Modèles réduits fiables et efficaces pour la planification et l'optimisation de mouvement des robots à pattes en environnements contraints

Pierre Fernbach

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ FÉDÉRALE TOULOUSE MIDI-PYRÉNÉES

Délivré par :
l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)

Présentée et soutenue le *15/10/2018* **par :**

Pierre FERNBACH

Modèles réduits fiables et efficaces pour la planification et l'optimisation de mouvement des robots à pattes en environnements contraints

---

JURY

Ludovic RIGHETTI	Professor W2	Rapporteur
Abderrahmane KHEDDAR	Directeur de Recherche	Rapporteur
Marie-Paule CANI	Professeur des Universités	Membre du Jury
Julien PETTRE	Directeur de Recherche	Membre du Jury
Michel TAÏX	Maître de conférences	Directeur de Thèse
Steve TONNEAU	Post-doctorant	coDirecteur de Thèse

École doctorale et spécialité :
EDSYS : Robotique 4200046
Unité de Recherche :
Laboratoire d'analyse et d'architecture des systèmes
Directeur(s) de Thèse :
Michel TAÏX et *Steve TONNEAU*
Rapporteurs :
Ludovic RIGHETTI et *Abderrahmane KHEDDAR*

Remerciements

Je souhaite remercier en premier lieu mon Directeur de Thèse Michel Taïx, qui fut également mon professeur de Master et qui m'a donné l'envie de me lancer dans la robotique grâce à ses cours très intéressants et pédagogiques. Malgré son emploi du temps plus que chargé il a toujours su se montrer présent et disponible durant ces trois années. Je lui suis très reconnaissant pour ses grandes qualités scientifiques et pédagogiques ainsi que sa franchise et sa sympathie.

Au même titre je souhaite remercier mon co-Directeur de thèse Steve Tonneau pour tous ses conseils avisés et ses idées brillantes ainsi que son attention de tout instant sur mes travaux. Merci pour ton énergie et ta sympathie, ce fut un grand plaisir de travailler avec toi. Surtout n'arrête jamais d'être aussi enthousiaste dès que tu as un début de piste d'idée, et même si cela n'abouti pas toujours du premier coup.

Malgré le fait que je sois le premier doctorant de Steve, Michel et lui ont su trouver l'encadrement qui me convenait le mieux, me permettant de passer ces trois années de thèse dans les meilleurs conditions possibles. J'ai beaucoup appris auprès d'eux et je les remercie encore pour cela.

Je souhaite également remercier mes rapporteurs Abderrahmane Kheddar et Ludovic Righetti pour l'intérêt qu'ils ont porté à mon travail. Je les remercie aussi, ainsi que Marie-Paule Cani et Julien Pettre, de l'honneur qu'ils m'ont fait en acceptant de faire partie de mon jury.

Je remercie tous les membres de l'équipe Gepetto avec qui j'ai collaboré durant ces trois années. En particulier Joseph Mirabel et Justin Carpentier pour avoir participé au développement des outils que j'ai utilisé, ainsi qu'Olivier Stasse pour son aide précieuse sur les plateformes robotiques.

Sans une bonne ambiance au sein de l'équipe, ces trois années de thèse ne se seraient pas aussi bien déroulées. Pour cela, je remercie tous les membres de Gepetto (ainsi que ceux de RAP qui traînent en salle Bauzil). En particulier tous les abonnés au V&B et l'équipe de JdR pour toutes ces soirées, ainsi que les joueurs de tarot sans qui la pause déjeuner serait bien moins intéressante (mais probablement beaucoup plus courte).

Je remercie bien sûr mes parents, qui m'ont toujours soutenu et encouragé durant toutes mes études et qui ont su cultiver ma curiosité scientifique dès le plus jeune âge. J'aimerais également remercier deux des enseignants qui m'ont donné l'envie de me lancer dans une carrière scientifique en me faisant partager leur passion : M. Beaumont et M. Roussigné.

Pour finir avec la plus importante, je remercie évidemment ma compagne Lauréline pour m'avoir supporté et soutenu tout au long de ces trois années.

Table des matières

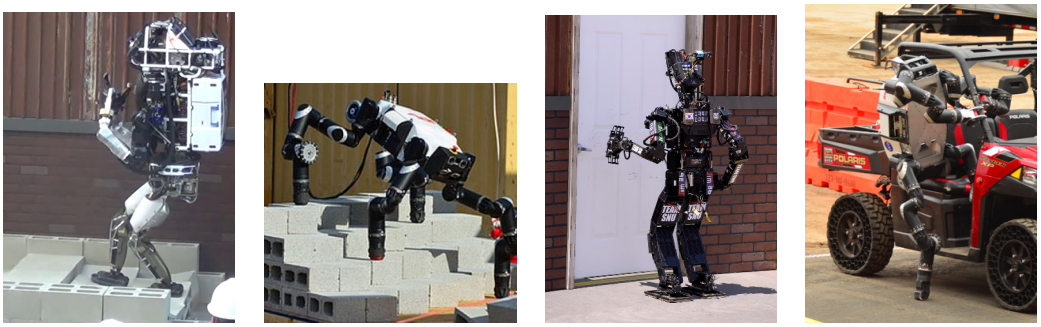
Introduction

Les robots occupent une place de plus en plus importante dans la société, à mesure qu'ils sortent des environnements clos et maîtrisés des usines et des laboratoires de recherche. Des bras manipulateurs dans les usines aux robots aspirateurs autonomes que l'on peut trouver chez de nombreux particuliers, en passant par les drones, les voitures ou même les tondeuses à gazon autonomes, il existe déjà de nombreux robots capables de réaliser des tâches diverses dans une grande variété de domaines d'application, avec plus ou moins d'autonomie.

Même si nous continuons à nous émerveiller devant l'arrivée des robots dans notre vie quotidienne, nous prenons rapidement l'habitude de leur présence et surtout de leur utilité.

Dans cette thèse nous nous intéressons en particulier aux **robots à pattes**, c'est à dire à l'ensemble des robots qui utilisent des pattes articulées pour se déplacer, les plus connus étant les robots humanoïdes ou les robots quadrupèdes. Or, la vision de **robots à pattes** évoluant naturellement au sein de la société civile reste à ce jour cantonnée à la science-fiction.

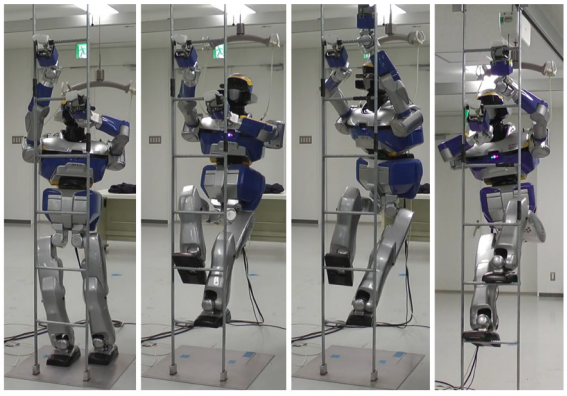
Pourtant, les capacités de locomotion des robots à pattes leur permettent de s'adapter et de naviguer dans des environnements très complexes. En effet, leur mode de locomotion les rend capables de naviguer aisément dans un environnement jonché d'obstacles, d'enjamber des débris (Figure 1-a et b), d'utiliser des escaliers ou des échelles (Figure 2), de prendre appui sur un mur ou une rambarde pour conserver l'équilibre (Figure 2-a), etc. Autant de scénarios inaccessibles pour des robots utilisant des roues. L'utilisation de pattes étend la variété de mouvements possibles pour un robot, ce qui motive le besoin de les étudier.



(a) Marche sur des débris (b) Enjambement de débris (c) Ouverture d'une porte (d) Sortie d'un véhicule

FIGURE 1 – Exemples de mouvements des robots candidats au Darpa Robotic Challenge.

(a) HRP-2 montant un escalier en s'aidant d'une rambarde. Image provenant de [Carpentier 2017c].



(b) HRP-2 montant une échelle. Image provenant de [Vaillant 2016].

FIGURE 2 – Exemples de mouvements multi-contacts avec HRP-2.

En plus de leurs capacités de locomotion, les robots humanoïdes sont dotés de bras leur permettant d'effectuer des tâches de manipulation. Ces robots seraient donc très utiles dans de nombreux scénarios et pourraient, à priori, accomplir une grande partie des tâches qu'un humain est capable de faire. Un des domaines où l'utilisation des robots humanoïdes serait fortement appréciée est l'intervention sur des lieux de catastrophes (appelé *disaster-response*) comme illustré sur la figure 3. Dans le cas d'environnements trop dangereux pour faire intervenir des secouristes humains, la possibilité d'utiliser des robots "secouristes" est étudiée depuis de nombreuses années. Malheureusement, malgré les avancées scientifiques et techniques, il n'y a eu que peu d'interventions utiles en conditions réelles de robots à pattes sur des sites de catastrophes.

FIGURE 3 – Vue d'artiste d'une équipe de robots humanoïdes intervenant après une catastrophe. Image provenant du site du DRC.

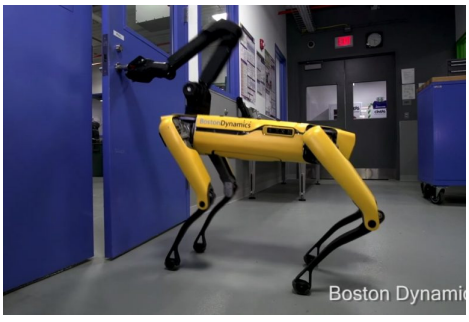
De nombreux autres scénarios bénéficieraient grandement de l'utilisation de robots à pattes, comme par exemple l'aide à domicile ou le rôle "d'ouvrier du futur" : un ouvrier robotique polyvalent, capable d'effectuer des tâches dangereuses ou pénibles pour les humains, et cela sans nécessiter d'aménagement particuliers de l'environnement.

Pourquoi, alors, les robots à pattes ne sont ils pas déployés à grande échelle ? Bien qu'il existe déjà de nombreux modèles de robots humanoïdes ou à pattes (comme par exemple ceux présentés dans la figure 4) capables physiquement et mécaniquement de réaliser tous les mouvements décrits précédemment, à ce jour il n'existe pas encore de méthode efficace et fiable capable de planifier, générer et contrôler le mouvement de ces robots de façon universelle dans n'importe quel type de scénario et sans intervention humaine. Autrement dit, ces robots n'ont pas l'autonomie nécessaire pour déterminer automatiquement quel mouvement il faut effectuer pour accomplir une tâche demandée.

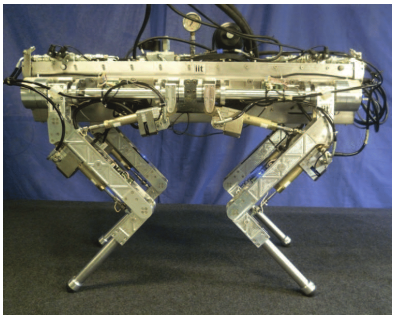
(a) ATLAS de Boston Dynamics

(b) HRP-2 de l'AIST

(c) TALOS de Pal-Robotics



(d) Spot-mini de Boston Dynamics



(e) HyQ de l'IIT

FIGURE 4 – Exemple de robots humanoïdes et de robots quadrupèdes.

En l'absence d'une telle méthode, ces robots ne sont pas prêts a être déployés de manière autonome dans un environnement quelconque en dehors des laboratoires. En effet, la majorité des mouvements exécutés sur des robots à pattes (comme par exemple les mouvements des candidats au *DARPA Robotic Challenge*, dont certains sont présentés sur la figure 1) sont soit télé-opérés avec une autonomie limitée, soit générés hors-ligne avec un temps de calcul élevé et avec l'intervention d'un expert humain pour définir certaines caractéristiques du mouvements (par exemple la position des pieds et des mains pour certains passage clés du mouvement). Le *DARPA Robotics Challenge* a mis en évidence que les erreurs humaines ont été la source de nombreux échecs ou chutes des robots [Atkeson 2015].

Pourquoi n'existe t'il pas de méthode générique permettant de réaliser de tels mouvements ? Afin de se déplacer un robot doit agir sur son environnement. En ce qui concerne les robots mobiles terrestres, cette action se fait via des forces exercées au niveau des points de contact entre le robot et l'environnement.

Dans le cas d'un robot à roues ces forces sont exercées par les roues sur le sol. La position des points de contact est donc généralement constante par rapport à l'origine du robot. De plus, la relation entre les variables de contrôle du robot (par exemple la vitesse de rotation des roues) et la résultante des forces exercées sur l'environnement peut être aisément calculée ou approximée dans la majorité des cas. Il est donc possible de déterminer un contrôle à appliquer au robot (par exemple une consigne de vitesse pour les moteurs des roues) par rapport à un déplacement désiré de l'origine du robot. Enfin, lors de la planification d'un mouvement pour des robots à roues (avec au moins trois roues), la contrainte d'équilibre peut souvent être ignorée puisque ces robots sont généralement conçus de façon à être en équilibre lorsque le robot est à l'arrêt. Toutefois ces simplifications ne peuvent pas toujours s'appliquer aux robots à roues, on peut par exemple trouver des cas avec des robots devant effectuer des mouvements très dynamiques ou devant manipuler des objets, ce qui demande de prendre en compte les contraintes d'équilibre ou des cas où il faut prendre en compte un possible glissement entre le terrain et les roues et où là relation entre le contrôle appliqué et les forces exercées n'est plus directe. Malgré cela, on trouve plusieurs méthodes dans la littérature capables de planifier des mouvements pour les robots à roues même dans les cas les plus compliqués.

Dans le cas d'un robot à pattes, cette tâche est bien plus complexe. D'une part, les contacts entre le robot et l'environnement sont intermittents et leurs positions ne sont pas constantes par rapport à l'origine du robot : il faut donc déterminer quelle patte (ou main, pied ...) doit établir un contact et où positionner ce contact dans l'environnement. La planification de contact doit donc choisir le contact à établir parmi une infinité de possibilités (une combinaison entre un ensemble discret de pattes et un ensemble continu de positions dans l'environnement). D'autre part, un robot à pattes est constitué de dizaines d'articulations. Ces articulations doivent se mouvoir de manière coordonnée afin de pouvoir exercer une force spécifique sur l'environnement. Cela implique que l'on ne peut pas déterminer aisément de

relation entre les variables de contrôle (le mouvement de chaque articulation) et le déplacement résultant de l'origine du robot à pattes. Enfin, le mouvement d'un robot à pattes doit respecter de nombreuses contraintes à la fois cinématiques (respect des limites articulaires, non collision avec l'environnement ou avec le robot lui même,...) et dynamiques (conservation de l'équilibre, condition de non glissement des contacts, ...).

Toutefois, il existe déjà dans la littérature des méthodes permettant de générer des mouvements pour des robots humanoïdes ou des robots à pattes pour certaines catégories de scénarios. Cependant chacune de ces méthodes présente des limites qui l'empêche d'être générique et de pouvoir résoudre tout type de scénarios sans intervention humaine.

Appellation et caractéristiques des différents types de mouvements

La caractéristique la plus importante d'un mouvement pour un robot à pattes est la nature des contacts entre le robot et l'environnement ainsi que la relation entre les différents contacts.

Le cas le plus simple est celui de la marche sur sol plat. Dans ce cas tous les contacts sont coplanaires, dans un plan orthogonal à l'axe vertical (l'axe selon lequel s'exerce la gravité). Pour cette catégorie de mouvement, des modèles simplifiés peuvent être utilisés, notamment le *Linear Inverse Pendulum Model* (LIPM) [Kajita 2003].

Une autre catégorie de marche, appelée la marche 3D, considère que les contacts sont toujours compris dans des plans orthogonaux à l'axe vertical mais l'élévation de ces plans peut varier. C'est le cas par exemple dans un scénario incluant un escalier.

Enfin, dans le cas général de la marche où les contacts ne sont plus coplanaires, le modèle simplifié du LIPM n'est plus valide [Caron 2017].

Une autre caractéristique importante du mouvement d'un robot à pattes est l'ordre des changements de contacts durant le mouvement. On peut différencier deux catégories de mouvements : cycliques et acycliques.

Dans le cas d'un mouvement cyclique, un cycle prédéfini de création et suppression de contacts va se répéter invariablement durant tout le mouvement. Pour ce type de mouvement, nous savons donc exactement quelle patte sera la prochaine à changer de contact à chaque instant du mouvement. Par exemple dans le cas de la marche bipède, si nous venons de poser le pied gauche nous savons que le prochain contact à chercher sera forcément pour le pied droit. Cela facilite grandement la planification en réduisant la dimension du problème.

Dans le cas général d'un mouvement acyclique, il n'y a aucune contrainte ou règle sur l'ordre des changements de contacts. Cela ajoute un problème de combinatoire durant la planification : décider quelle sera la prochaine patte à changer de contact.

On appelle **multi-contact** un mouvement pouvant présenter des contacts non coplanaires et une séquence de contacts acyclique, comme ceux montrés dans la figure 2. Autrement dit, aucune hypothèse n'est faite sur le type de contacts créés durant ce mouvement. Ce type de mouvement est particulièrement intéressant car il permet d'exploiter au maximum les capacités de locomotion des robots à pattes, et permet de réaliser des tâches complexes.

Dans cette thèse nous nous intéressons aux méthodes de planification pour des mouvements de locomotion en **multi-contact** pour des robots humanoïdes ou des robots à pattes, en limitant au maximum les interventions ou utilisations d'expertise humaine durant la planification. Une des contraintes pour pouvoir appliquer ces méthodes sur un robot réel est de pouvoir calculer le mouvement avec un temps de calcul inférieur à la durée totale du mouvement, nous parlerons dans ce cas de calcul en temps interactif.

État de l'art

En ce qui concerne la marche cyclique sur sol plat, de nombreuses méthodes existent déjà et le problème peut être considéré comme résolu. Ces méthodes se basent sur l'utilisation d'un modèle simplifié, valide seulement pour des contacts coplanaires et sur sol plat : le *Linear Inverse Pendulum Model* [Kajita 2003]. Dans ce cas, des méthodes efficaces tel que le *capture point* [Pratt 2006] peuvent être utilisées pour planifier la position du prochain contact.

Dans le cas de navigation parmi les obstacles ou sur un sol avec des obstacles, [Kuffner 2001] puis [Chestnutt 2003] ont introduit le problème du *footstep planning* qui consiste à planifier les positions successives des contacts des pieds afin d'obtenir un mouvement sans collision. Ces méthodes fonctionnent en discrétisant les mouvements possibles des jambes en un ensemble discret d'actions possibles. Puis, un algorithme de type A^* est utilisé pour trouver une séquence de pas optimale selon une heuristique. Cependant, la taille de l'ensemble d'actions possibles détermine le degré de ramification du graphe exploré par l'algorithme A^*. Pour résoudre des scénarios complexes, il faudrait utiliser un ensemble d'actions suffisamment grand, ce qui rendrait le problème impossible à traiter numériquement. De ce fait, ces méthodes n'ont de succès que pour la navigation sur du sol plat. [Perrin 2012] propose une méthode utilisant une variante de l'algorithme RRT afin de planifier des mouvements sans collision capables d'enjamber des obstacles.

[Grey 2017] propose une autre méthode de *footstep planning* où il utilise des graphes de possibilités pour planifier des mouvements de marches avec évitement d'obstacles. Cette méthode est également limitée à de la marche cyclique sur sol plat et en équilibre statique. Mais, l'intérêt de ce travail est que leur méthode est prouvée comme "complète en probabilité". Ce qui signifie que si un mouvement quasi-statique existe pour un scénario, la probabilité que leur méthode le trouve tend vers 1 quand le temps de calcul tend vers l'infini.

Le problème de planification de mouvements acycliques a été décrit et défini pour la première fois par Bretl [Bretl 2006b]. D'après sa définition, afin de résoudre le problème de la planification de mouvements acyclique il faut pouvoir résoudre deux sous-problèmes simultanément : planifier un chemin guide pour le centre du robot dans $SE(3)$, noté sous-problème $\mathcal{P}_1$, et aussi planifier une séquence discrète de configurations en contact le long de ce chemin, noté sous-problème $\mathcal{P}_2$. Dans le cadre des travaux de Bretl les configurations sont contraintes à être en équilibre statique. La figure 5 montre un exemple d'une telle séquence de configurations en contact dans le cas d'un mouvement **multi-contact**.

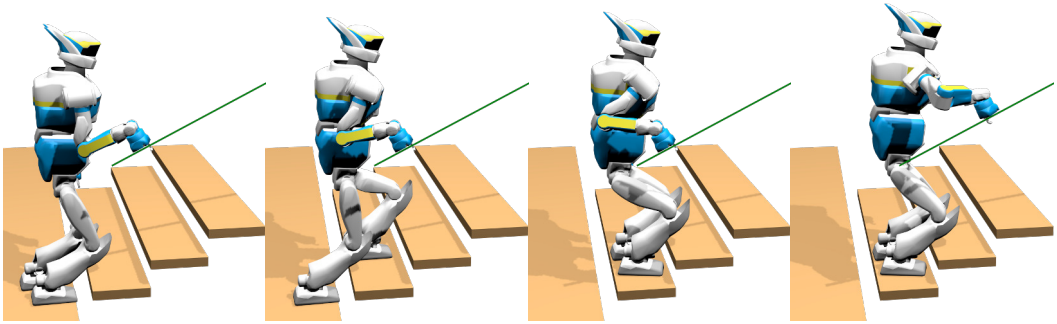


FIGURE 5 – Exemple de séquence de configurations en contact.

Un troisième problème non trivial, noté $\mathcal{P}_3$, est la génération d'un mouvement corps-complet en interpolant entre chaque configuration de la séquence précédente. Une illustration de ces trois sous-problèmes est présentée dans la figure 6.

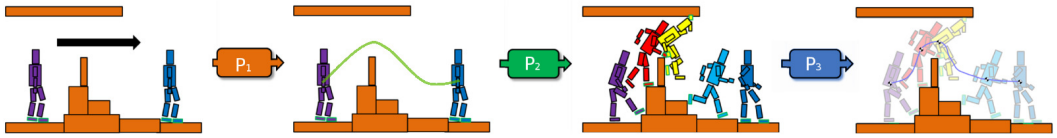


FIGURE 6 – Identification des trois sous-problèmes inclus dans le problème de la locomotion multi-contact.

La difficulté principale lorsque l'on doit résoudre ce problème de planification de mouvement acyclique est alors l'explosion combinatoire qui apparaît lorsque l'on considère en même temps le choix de contact parmi une infinité de possibilités et la planification du guide. [Bretl 2006b] propose le premier algorithme efficace capable de planifier des mouvements pour des scénarios simples, tels que de l'escalade avec un quadrupède. Mais cette méthode requiert de prédéfinir un ensemble discret de points de contact potentiels dans l'environnement, de ce fait elle n'est pas adaptable directement à un environnement quelconque.

En suivant la même approche, différentes méthodes ont été proposées, comme [Hauser 2006], pour résoudre le problème dans des scénarios spécifiques, toujours en limitant la combinatoire en fixant un ensemble discret de points de contact possibles

dans l'environnement.

La grande majorité des travaux suivants ceux de Bretl pour la planification de mouvement **multi-contact** ont exploré des formulations alternatives pour gérer la combinatoire du problème. La complexité de ce problème est parfaitement résumée dans [Bouyarmane 2018] et une étude détaillée de l'état de l'art y est présentée. On peut distinguer dans tous ces travaux deux approches différentes :

D'un coté, le problème complet est résolu via des méthodes d'optimisation. D'un autre coté, les deux sous-problèmes $\mathcal{P}_1$ (planification de la trajectoire du centre du robot) et $\mathcal{P}_2$ (planification de contact) sont découplés et résolus séparément.

D'autres travaux supposent que la séquence de contacts est donnée en entrée du problème. Ils se concentrent sur le problème de la génération de mouvement soit à partir d'un modèle réduit du robot, soit en considérant le corps complet (sous-problème $\mathcal{P}_3$) du robot.

Approche par optimisation

L'utilisation de méthodes d'optimisation locale permettant d'optimiser conjointement la position des contacts et la trajectoire du centre d'un robot à pattes est montrée dans [Mordatch 2012]. Cette méthode a été appliquée à un robot humanoïde sur des scénarios simples de marche [Mordatch 2015]. Mais cette méthode ne peut pas garantir de trouver l'optimal global et peut rester coincée dans un minimum local créé par des obstacles. Afin d'obtenir des temps de calcul raisonnables, la méthode proposée utilise un modèle dynamique simplifié. Malgré cela, la méthode requiert environ une minute de calcul pour 20 contacts. Plus récemment, [Winkler 2018] propose une méthode efficace permettant également d'optimiser les instants de changement de contacts. Cette méthode permet d'avoir un temps de calcul interactif et ne nécessite que quelques secondes par pas, pour un robot avec peu de degrés de liberté (18). Cependant, elle ne considère pour l'instant que des contacts ponctuels, approxime les butées articulaires du robot et ne prend pas en compte la collision avec l'environnement. De plus, comme les autres méthodes d'optimisation non linéaires, elle peut rester bloquée dans un minimum local et nécessite une initialisation correcte. Une méthode similaire a été utilisée pour de la manipulation dans [Gabiccini 2015].

[Deits 2014] propose de résoudre le problème de planification de contact en le formulant comme un problème de *mixte entier*. L'espace libre est décomposé en régions convexes pouvant contenir une boite englobante du robot. Puis à l'aide de technique de programmation à variables mixtes, il optimise le placement des points de contact dans ces régions tout en optimisant un critère global. Mais, cet optimal ne peut être atteint que si le mouvement est entièrement dans les régions convexes de l'espace. De plus, cette méthode est limitée aux mouvements cycliques bipèdes et les contraintes d'équilibres ne sont pas considérées. [Aceituno-Cabezas 2018] étend

cette méthode aux robots à pattes dans le cas général et permet de choisir le meilleur cycle de contact parmi un ensemble prédéfini, en intégrant des contraintes d'équilibre.

[Ponton 2016] propose une relaxation convexe de la dynamique centroïdale, qui lui permet de formuler le problème de la dynamique centroïdale comme un QCQP (*quadratically-constrained quadratic program*) convexe. En utilisant cette reformulation, il étend ensuite les travaux de [Deits 2014] en incluant le modèle dynamique dans la formulation du problème. Cela permet de générer des positions de contacts qui prennent en compte les contraintes dynamiques du robot.

Basé sur le travail de [Posa 2014], [Dai 2014] propose une méthode utilisant le modèle dynamique centroïdal et le modèle cinématique complet du robot, permettant d'optimiser la séquence de contacts et les instants de transitions entre chaque contact. L'utilisation du modèle cinématique complet permet d'assurer le respect de toutes les contraintes cinématiques, notamment les butées articulaires. Mais, cette méthode requiert plusieurs minutes à plusieurs heures de calcul pour des scénarios avec une dizaine de contacts.

Le domaine de l'animation graphique présente des problématiques similaires à celui de la robotique. En effet, dans ces deux domaines nous nous intéressons à planifier le mouvement d'un corps articulé, qu'il s'agisse d'un avatar ou d'un robot.

De ce fait, certaines méthodes peuvent être intéressantes pour les deux domaines, par exemple [Peng 2017, Holden 2017] utilisent des méthodes d'apprentissage afin de générer des mouvements dynamiques avec des contacts non-coplanaires et des temps de calcul très faible. Cependant, ces méthodes nécessitent l'utilisation d'une base de donnée de mouvements de référence.

D'autres travaux se sont concentrés sur la manière dont un expert humain peut définir des caractéristiques du mouvement. Par exemple, [Guay 2015] propose une méthode permettant de générer un mouvement corps-complet en spécifiant seulement une ou plusieurs courbes définissants le mouvement global souhaité.

Approche découplée

A notre connaissance, les premiers travaux à montrer l'intérêt de l'utilisation d'une approche découplée sont [Escande 2008, Escande 2013] où une trajectoire guide pour l'origine du robot est manuellement définie afin de guider la planification de contact. Cette méthode permet de résoudre des scénarios contraints mais avec de longs temps de calcul (de quelques minutes à plusieurs heures). De plus, la qualité du mouvement produit dépend de la qualité du guide manuellement défini.

Ensuite, les travaux de [Bouyarmane 2009, Bouyarmane 2012] se sont intéressés au calcul automatique de cette trajectoire guide en garantissant qu'il existe des configurations en équilibre statique le long de ce guide. Cependant, l'existence de

ces configurations ne suffit pas à garantir qu'il existe un mouvement faisable entre elles.

[Chung 2015] propose une autre méthode basée sur l'approche découplée, en jugeant la capacité du robot à créer des contacts avec un environnement discrétisé grâce à l'espace atteignable des membres du robot durant la phase de planification du guide. Cependant, leur méthode de planification de guide ne considère pas les collisions possibles, et une re-planification doit avoir lieu après chaque échec de la planification de contact.

[Tonneau 2015b, Tonneau 2018a] propose la première méthode découplée permettant d'atteindre des performances interactives. Cette méthode permet de planifier rapidement un chemin guide dans un espace de faible dimension. Durant cette planification, une heuristique géométrique basée sur l'espace d'accessibilité de chaque membre du robot est utilisée afin de donner une forte probabilité à l'existence d'une séquence de configurations en contact cinématiquement valides et sans collisions le long du guide planifié, sans toutefois fournir de garanties de succès.

Une limitation de toutes ces méthodes est qu'elles ne considèrent que des configurations en **équilibre statique**, c'est à dire avec une vitesse et une accélération nulle. Cela ne signifie pas forcément que le mouvement complet est quasi-statique, mais la séquence de contacts planifiée ne peut contenir que des configurations en contact en équilibre statique.

De ce fait, ces méthodes ne considèrent qu'une partie des problèmes possibles puisqu'elles ne peuvent pas trouver des séquences de contacts exploitant la dynamique du robot permettant de résoudre des scénarios sans solutions quasi-statiques, comme par exemple de la marche sur des plans très inclinés.

Génération de mouvement corps complet

De nombreux travaux [Hauser 2014, Herzog 2015, Park 2016, Carpentier 2016, Carpentier 2017c, Caron 2018] se sont concentrés sur la génération du mouvement corps complet en supposant que la séquence de contacts était une entrée donnée du problème.

[Carpentier 2017c] propose une méthode d'optimisation de trajectoire centroïdale en formulant un problème de contrôle optimal résolue par une méthode de *multiple-shooting*. Puis, un mouvement corps-complet est généré via une méthode de cinématique inverse du second ordre prenant la trajectoire centroïdale calculée précédemment comme référence. Cette méthode permet d'obtenir des performances interactives.

Dans [Ponton 2016], une relaxation convexe est proposée afin de résoudre le problème de la dynamique centroïdale de manière plus rapide. Ce travail est étendu dans [Ponton 2018] afin de pouvoir prendre en compte le moment cinétique et de pouvoir optimiser les durées des différentes phases de contact. Pour générer un mouvement corps complet, une méthode itérative est utilisée [Herzog 2016]. Ces travaux

proposent d'alterner entre une phase d'optimisation de la trajectoire centroïdale et une phase d'optimisation du mouvement corps complet, prenant en compte les contraintes cinématiques du robot.

[Caron 2018] propose une méthode plus performante qui permet de réduire le temps de calcul d'au moins un ordre de grandeur par rapport aux méthodes précédentes. Mais, cette méthode est limitée aux transitions entre des postures en simple-support (avec un seul contact).

Présentation de l'approche découplée utilisée dans l'équipe Gepetto

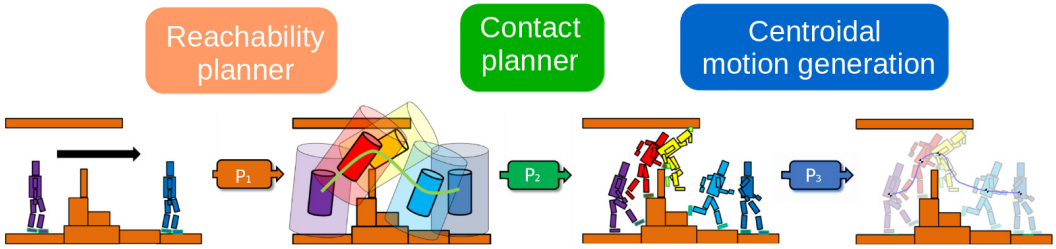


FIGURE 7 – Vue globale de l'approche découplée utilisée.

Depuis quelques années, le problème de la planification de mouvement multi-contact pour robots à pattes est abordé dans l'équipe Gepetto, via une approche découplée, au travers du projet Loco-3D [Carpentier 2017b].

Ce projet vise à terme à obtenir un robot entièrement autonome capable d'effectuer des tâches de locomotion multi-contact dans un environnement quelconque et inconnu, sans intervention humaine autre que la spécification d'un placement (position et orientation) final désiré. Dans le cadre de cette thèse, nous ne considérons pas l'aspect perception ni localisation, nous supposons que l'environnement et que la position du robot dans l'environnement sont parfaitement connus.

L'approche découplée proposée est illustrée par la figure 7. Nous y définissons trois sous-problèmes qui sont résolus séquentiellement :

$\mathcal{P}_1$: planification d'une trajectoire guide. Cette méthode prend en entrée une représentation de haut niveau de la tâche de locomotion : un environnement, un robot, un placement initial et final pour le centre du robot. Elle produit en sortie une trajectoire guide pour le tronc du robot. Cette méthode utilise un modèle réduit du robot afin de pouvoir planifier efficacement le guide dans un espace de faible dimension.

$\mathcal{P}_2$: planification de contacts. Cette méthode prend en entrée la trajectoire guide et va planifier une séquence discrète de configurations en contact qui suivent ce guide. Cette méthode considère le modèle complet du robot afin de pouvoir générer des configurations corps-complet en équilibre et sans collisions.

$\mathcal{P}_3$: la génération de mouvement corps-complet. Cette méthode prend en entrée la séquence de configurations en contact et génère une trajectoire centroïdale res-

pectant la dynamique du robot selon les contacts de la séquence et optimale selon un coût prédéfinie. Cette méthode ne considère que le modèle centroïdal du robot. Ensuite, une méthode de génération de mouvement corps complet prend en entrée cette trajectoire centroïdale et la séquence de contacts pour générer un mouvement corps-complet en utilisant le modèle complet du robot.

Des méthodes permettant de résoudre $\mathcal{P}_1$ et $\mathcal{P}_2$ ont été proposées dans [Tonneau 2018a]. La première est une méthode appelée *RB-RRT* qui permet de calculer un chemin guide durant $\mathcal{P}_1$, basée sur une heuristique assurant une forte probabilité pour qu'il existe des configurations cinématiquement faisables et en contact le long de ce chemin guide. Ensuite, un planificateur de contacts produit une séquence de configurations en contact, cinématiquement valides et en équilibre statique, qui permet de suivre exactement ce guide.

Afin de résoudre $\mathcal{P}_3$, [Carpentier 2017c] a proposé de résoudre le problème de la dynamique centroïdale via une méthode de *multiple shooting*. La génération de mouvement corps-complet est ensuite calculée avec une cinématique inverse du second ordre.

Bien que ces méthodes permettent effectivement de résoudre chacun des trois sous-problèmes avec des résultats prometteurs, elles présentent des limitations. Tout d'abord la planification du guide ne considère que des critères géométriques, il n'y a donc aucune garantie qu'il existe une séquence de configurations en équilibre le long du guide planifié.

Ensuite, la planification de contacts produit une séquence de configurations en contact où chaque configuration est en équilibre statique, cette méthode ne permet pas de résoudre des scénarios dynamiques sans solution quasi-statique. De plus, ce planificateur de contact ne peut pas garantir qu'il existe un mouvement corps-complet respectant les contraintes cinématiques et dynamiques du robot entre chaque configuration de la séquence.

Enfin, la méthode de cinématique inverse utilisée pour générer le mouvement corps-complet ne peut pas gérer l'évitement de collisions avec l'environnement. De plus, elle nécessite de prendre en entrée des trajectoires de référence pour les organes terminaux qui sont actuellement définies manuellement.

Bilan

Basé sur cette étude de l'état de l'art, nous pouvons déduire qu'il n'existe actuellement aucune méthode permettant de planifier un mouvement corps complet à partir seulement d'une tâche de haut niveau telle qu'une position but pour le centre du robot et présentant toutes les caractéristiques suivantes simultanément :

— Génère des mouvements **multi-contact** (acyclique et avec des contacts non coplanaires) ;
— N'est pas limité aux mouvements quasi-statiques ;

— Produit un mouvement dans un environnement complexe avec des obstacles non convexes ;
— Produit un mouvement respectant les contraintes dynamiques ;
— Produit un mouvement respectant les contraintes cinématiques et sans (auto-)collisions ;
— Ne nécessite pas d'intervention humaine pour prédéfinir des caractéristiques du mouvement ;
— Est fiable (présente une garantie de convergence ou un fort taux de succès) ;
— **Présente un temps de calcul interactif** (le temps de calcul d'un mouvement est inférieur à la durée d'exécution de ce mouvement) ;

Les principales différences entre les méthodes par optimisation et les méthodes découplées sont que l'approche découplée permet de gagner en performance grâce à des approximations successives et en formulant des problèmes moins complexes. Mais, cette approche nécessite de faire des choix intermédiaires sans pouvoir prendre en compte le problème dans sa globalité. L'approche par optimisation permet une formulation unique du problème, mais elle doit également reposer sur des approximations pour obtenir des temps calcul raisonnables [Winkler 2018].

En général les méthodes d'optimisation explorent moins bien l'espace que les méthodes de planification probabilistes, associées aux approches découplées, ce qui peut les amener à rester bloquées dans des minima locaux, notamment causés par la présence d'obstacles. Dans cette thèse nous sommes surtout intéressés par la capacité d'exploration des méthodes découplées et par le fait qu'elles puissent gérer des environnements très contraints.

Cependant, si le découplage des sous-problèmes permet de gagner en efficacité, il introduit également un nouveau type de problème : **le problème de faisabilité**. En effet, **comment garantir que la solution d'un sous-problème sera une entrée faisable pour le sous-problème suivant, puisque chaque sous problème utilise des modèles différents du robot ?**

Ce problème n'est pas propre à notre méthode mais apparaît dès que l'on utilise des modèles réduits et approximés du robot (comme par exemple le modèle centroïdal couramment utilisé). D'autres auteurs sont arrivés à une conclusion similaire dans [Bouyarmane 2018] en observant que les méthodes de génération de séquence de contacts de l'état de l'art peuvent produire des séquences de contacts pour lesquelles il n'existe pas de mouvement corps-complet valide. Il s'agit là du problème de faisabilité entre $\mathcal{P}_2$ et $\mathcal{P}_3$. Ici nous soutenons que ce problème de faisabilité existe entre chacun des sous-problèmes.

Les travaux de cette thèse visent donc à résoudre ce problème de faisabilité dans le contexte de l'architecture découplée proposée par l'équipe Gepetto.

Le problème de la faisabilité

Afin de pouvoir utiliser l'architecture découplée présentée dans la figure 7 de manière efficace, il faut être capable de garantir que chaque solution trouvée pour un sous-problème est une entrée **faisable** pour le sous-problème suivant. C'est à dire qu'il existe une solution à ce sous-problème avec cette entrée.

Il faut donc être capable de garantir qu'il existe une séquence de configurations en contact, sans collision et en équilibre, permettant de suivre la trajectoire guide calculée pendant $\mathcal{P}_1$. De même, il faut pouvoir garantir qu'il existe une trajectoire centroïdale valide permettant de suivre la séquence de contacts calculée pendant $\mathcal{P}_2$. Le mouvement corps-complet doit ensuite être capable de suivre cette trajectoire centroïdale.

Une solution naïve permettant de contourner ce problème serait d'utiliser une stratégie de type "essais et erreurs" : en autorisant chacun des sous-problèmes à échouer et en rebouclant avec le sous-problème précédent après chaque échec pour calculer une nouvelle solution. Cette stratégie ne permettra pas d'atteindre notre objectif avec un temps de calcul interactif car elle nécessitera de nombreuses itérations de chaque méthode avant de trouver une solution.

C'est pourquoi nous nous intéressons donc à résoudre ce problème de faisabilité, en proposant des **critères de faisabilité** permettant de garantir que chaque sous-problème produit une solution qui sera dans l'espace de faisabilité du sous-problème suivant. Autrement dit, que chaque solution d'un sous-problème est une entrée faisable pour le sous-problème suivant, et donc qu'il existe une solution avec cette entrée.

Contributions

Dans la première partie de ce manuscrit, nous proposons une extension kinodynamique du planificateur RB-RRT prenant en compte la dynamique du robot durant la planification, capable de résoudre des scénarios pour lesquels aucune solution quasi-statique n'existe. Afin de pouvoir mettre en œuvre cette version kinodynamique, nous avons proposé une méthode de planification locale capable de produire des trajectoires optimales en temps connectant exactement deux états du modèle réduit utilisé durant $\mathcal{P}_1$ et qui prend en compte les contraintes dynamiques appliquées au robot, dépendantes de l'état courant du robot et de ses contacts actuels avec l'environnement. Nous étendons également la méthode de planification de contact afin qu'elle génère des configurations qui respectent les contraintes dynamiques en suivant la trajectoire guide, sans être contraintes à être en équilibre statique.

Cette version kinodynamique permet donc d'offrir des garanties qu'il existe des configurations corps-complet respectant les contraintes dynamiques du robot le long de la trajectoire guide planifiée et donc que la méthode de planification de contact pourra trouver une solution avec ce guide en entrée, contrairement à la méthode

RB-RRT classique. Cette contribution permet donc de résoudre le problème de la faisabilité entre les sous problèmes $\mathcal{P}_1$ et $\mathcal{P}_2$. De plus, notre contribution permet de produire des séquences de configurations en contact qui ne sont plus limités à être en équilibre statique.

Dans la seconde partie de ce manuscrit, nous nous intéressons à la faisabilité d'une transition de contact en proposant un critère qui permet de répondre à la question suivante : existe-t-il une trajectoire centroïdale valide connectant deux configurations avec des contacts différents ? Nous proposons ensuite une méthode basée sur une reformulation conservatrice mais convexe permettant de répondre à cette question avec un temps de calcul inférieur d'au moins un ordre de grandeur aux méthodes de génération de trajectoire centroïdale. De plus cette méthode ne nécessite pas d'approximations autres que l'utilisation du modèle centroïdal, elle assure également que l'ensemble de la trajectoire centroïdale satisfait les contraintes du problème sans recourir à la discrétisation de cette trajectoire.

Nous allons ensuite utiliser cette méthode durant la planification de contact afin de garantir que la séquence de configurations planifiée sera une entrée faisable pour la méthode de génération de trajectoire centroïdale. Résolvant ainsi le problème de la faisabilité entre $\mathcal{P}_2$ et $\mathcal{P}_3$.

Enfin, dans la troisième partie nous présentons l'architecture complète de planification de mouvement mise en œuvre durant cette thèse et utilisant les deux critères proposés. Cette architecture permet de produire automatiquement des mouvements corps-complet en **multi-contact** cinématiquement et dynamiquement valides à partir seulement de la définition de haut niveau d'une tâche de locomotion. Puis, nous présentons des résultats obtenus avec cette architecture, en simulation ou sur le robot réel HRP-2 et nous montrons que nous obtenons des performances interactives ainsi qu'un taux de succès élevé. Dans cette dernière partie, nous analysons la justesse et l'intérêt des deux critères de faisabilité proposés dans les parties précédentes.

Publications

Les travaux présentés dans le chapitre 2 ont donné lieu à la publication [Fernbach 2017] et ceux du chapitre 4 à la publication [Fernbach 2018]. Les travaux de la section 3.5.3 sont un résumé de l'article [Tonneau 2018b].

De plus, les travaux résumés dans la section 1.2.2 proviennent de l'article [Campana 2016a]. Les développements logiciels effectués durant la thèse ont permis de contribuer à l'article [Mirabel 2016].

Liste des publications

[Campana 2016] Mylène Campana, Pierre Fernbach, Steve Tonneau, Michel Taïx et Jean-Paul Laumond. *Ballistic motion planning for jumping superheroes.* Dans Motion in Games Conference, Burlingame, CA, United States, octobre 2016. (Cité en page 40.)

[Fernbach 2017] P. Fernbach, S. Tonneau, A. Del Prete et M. Taïx. *A kinodynamic steering-method for legged multi-contact locomotion.* Dans 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3701–3707, Sept 2017. (Cité en page 58.)

[Fernbach 2018] Pierre Fernbach, Steve Tonneau et Michel Taïx. *CROC : Convex Resolution Of Centroidal dynamics trajectories to provide a feasibility criterion for the multi contact planning problem.* Dans 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 2018. (Cité en pages 87, 90 et 119.)

[Mirabel 2016] J. Mirabel, S. Tonneau, P. Fernbach, A. K. Seppälä, M. Campana, N. Mansard et F. Lamiraux. *HPP : A new software for constrained motion planning.* Dans 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 383–389, Oct 2016. (Cité en pages 52, 67 et 139.)

[Tonneau 2018] Steve Tonneau, Pierre Fernbach, Andrea Del Prete, Julien Pettré et Nicolas Mansard. *2PAC : Two Point Attractors for Center of Mass Trajectories in Multi Contact Scenarios.* Transaction on Graphics, Jul 2018. (Cité en pages 76, 78, 79 et 101.)

Définitions et notations générales

Sommaire

Ce chapitre regroupe des définitions et des notations utilisées tout au long de ce manuscrit.

Représentation du modèle du robot

Un robot est constitué d'un ensemble de corps rigides reliés par des articulations. Ces articulations sont des liaisons mécaniques qui contraignent le mouvement relatif des corps qu'elles relient. On nomme *chaîne cinématique* l'ensemble des liaisons qui constituent le robot. L'une des extrémités de cette chaîne est reliée à une base (fixe ou mobile) et est appelée l'*origine* ou la *racine* du robot. L'autre extrémité est appelée *organe terminal* et est généralement doté d'un outil, d'un dispositif de préhension ou est conçu pour établir un contact avec l'environnement (dans le cas d'un pied ou d'une patte).

Les robots humanoïdes ou les robots à pattes sont constitués de chaînes cinématiques arborescentes, puisque chaque membre (jambe, bras, patte...) forme une chaîne cinématique. Des exemples de chaînes cinématiques pour des robots humanoïdes sont présentés sur la figure 8.

On appelle degrés de liberté d'une liaison les mouvements relatifs indépendants d'un des corps par rapport à l'autre autorisés par la liaison. Il est donc possible de décrire la transformation entre deux corps reliés par une liaison avec un nombre de valeurs égale au nombre de degrés de liberté de cette liaison. La quasi-totalité des liaisons constituant les robots humanoïdes ou à pattes sont des liaisons pivots, autorisant un seul degré de liberté en rotation.

Dans le cas d'une base mobile, celle-ci possède 6 degrés de liberté (translation et rotation selon les 3 axes). Une base fixe possède 0 degré de liberté.

Le nombre de degrés de liberté d'un robot est la somme des degrés de liberté de chacune de ses liaisons et de sa base.

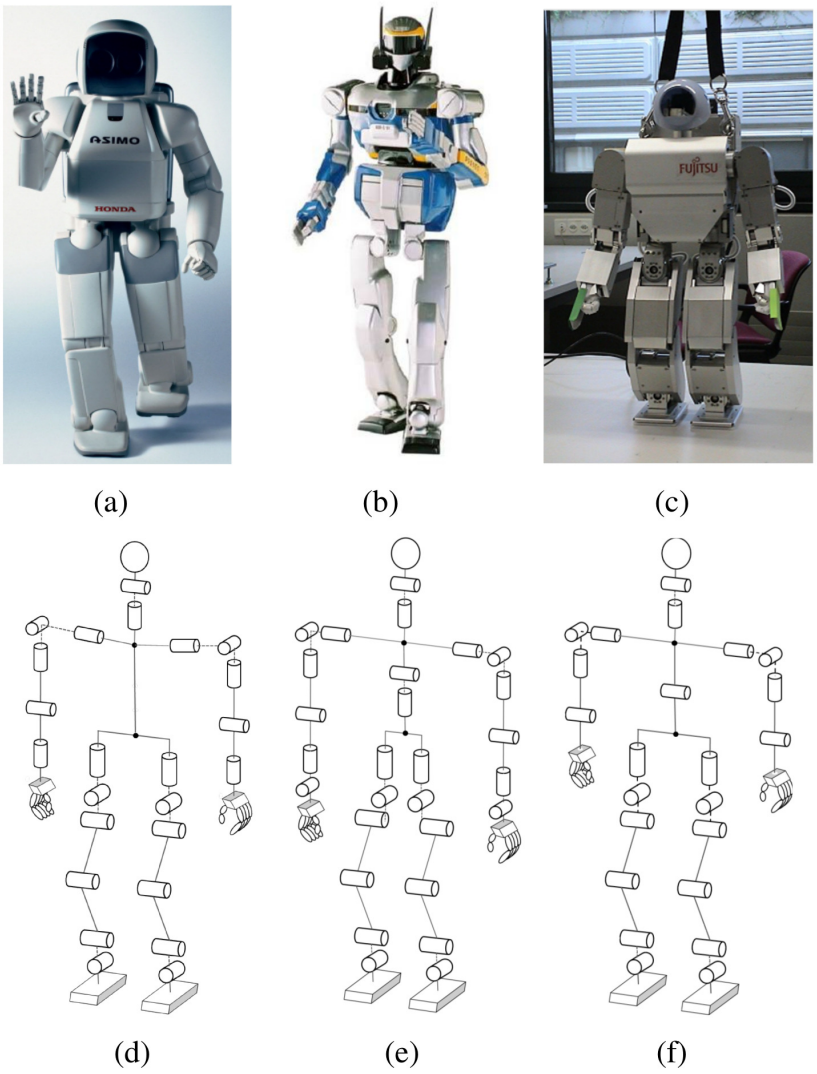


FIGURE 8 – Figure extraite de [Ali 2010]. Exemple de trois robots et des chaînes cinématiques les composant. (a) le robot ASIMO et sa chaîne cinématique (d). (b) le robot HRP-2 et sa chaîne cinématique (e). (c) le robot HOAP-2 et sa chaîne cinématique (f).

Espace des configurations

La position et l'orientation de chacun des corps du robot peut être décrite par un vecteur $\mathbf{q}$ (appelé une *configuration* du robot) dans l'espace des configurations $\mathcal{C} \in \mathbb{R}^N$. Avec N le nombre de degrés de liberté du robot.

Notion de contact

Le déplacement d'un robot à pattes ne peut se faire qu'en établissant des contacts avec l'environnement. Le robot peut alors exercer des forces sur l'envi-

ronnement en ces points de contact.

Dans cette section nous définissons la notion de contact, les phénomènes physiques impliqués et les notations utilisées dans le reste de ce manuscrit.

Définition d'un contact

L'interaction entre un robot et l'environnement est exprimé pour un ensemble discret de contacts. Un contact est défini par une position 3D $\mathbf{p}_i \in \mathbb{R}^3$ dans le repère monde (appelée point de contact) et la direction de la normale au contact (un vecteur unitaire orthogonal à la surface de contact) $\mathbf{n}_i \in \mathbb{R}^3$. A chaque point de contact, une force de contact $\mathbf{f}_i$ est définie, il s'agit de la force exercée par l'environnement sur le robot.

La majorité des contacts établis par un robot sont des contacts unilatéraux, ce qui signifie que le robot peut "pousser" sur l'environnement mais ne peut pas le "tirer". Plus formellement, le produit scalaire entre la force de contact $\mathbf{f}_i$ et la normale au contact $\mathbf{n}_i$ doit être positif. L'exception est le cas où le contact est établi avec un organe terminal préhensile, comme une main pouvant se fermer.

Les organes terminaux des robots ayant quatre pattes ou plus sont généralement conçus pour établir des contacts ponctuels avec l'environnement. Dans le cas des robots bipèdes la majorité des pieds sont rectangulaires, ils établissent alors des contacts planaires avec l'environnement. Dans ce cas, nous exprimons les points de contact aux quatre coins du pied, comme montré sur la figure 9.

Contrainte de non glissement

Une des contraintes principales que le robot doit respecter est la contrainte d'adhérence (ou de non-glissement) des contacts.

La loi de Coulomb définit la condition de non glissement par la relation suivante :

$$\mathbf{f}_t \leq \mu \mathbf{f}_n \tag{1}$$

Avec $\mathbf{f}_t$ la composante tangentielle de la force de contact (parallèle à la surface de contact) et $\mathbf{f}_n$ la composante normale de la force de contact (orthogonal à la surface de contact). μ est le coefficient de friction statique entre le robot et l'environnement, il s'agit d'une caractéristique intrinsèque des matériaux utilisés. Cette équation peut être écrite sous la forme suivante :

$$\sqrt{(\mathbf{t}_{i1}^\top \mathbf{f}_i)^2 + (\mathbf{t}_{i2}^\top \mathbf{f}_i)^2} \leq \mu_i \mathbf{n}_i^\top \mathbf{f}_i \quad \forall i \in [0; k] \tag{2}$$

Avec $\mathbf{n}_i \in \mathbb{R}^3$ la direction de la normale au i-ième point de contact et $(\mathbf{t}_{i1}, \mathbf{t}_{i2})$ les directions tangentes.

Cette condition peut également être exprimée par la relation suivante :

$$\mathbf{f}_i \in \mathcal{K}_i \tag{3}$$

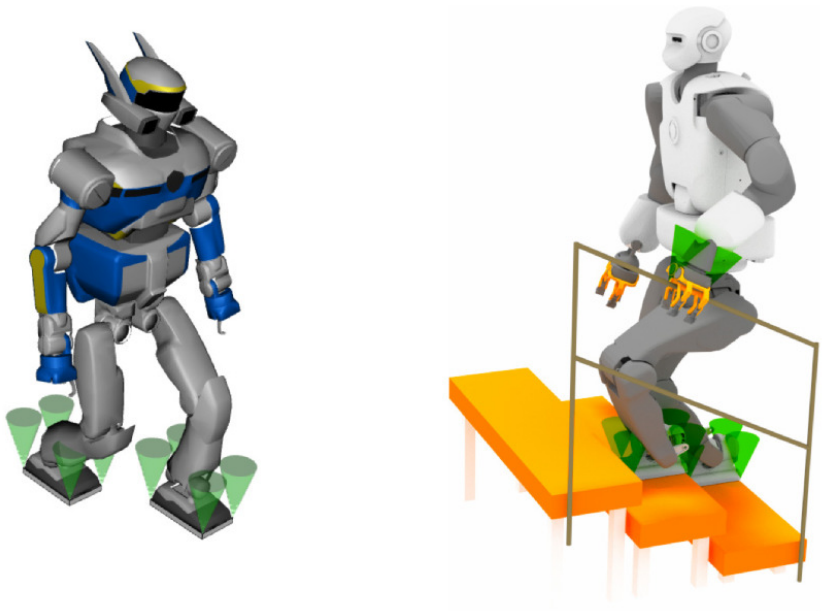


FIGURE 9 – Figure extraite de [Carpentier 2017c]. Illustration du robot HRP-2 (gauche) et TALOS [Stasse 2017] (droite) en contact avec l'environnement. En vert les cônes de friction pour un coefficient de friction de 0.3. Dans le cas des pieds rectangulaires, les points de contact sont exprimés aux quatre coins du pied (base des cônes de friction).

Où $\mathbf{f}_i$ est la force de contact exprimée au point $\mathbf{p}_i$ et $\mathcal{K}_i$ est le cône de friction également exprimé au point $\mathbf{p}_i$. L'expression de ce cône dépend de la normale au contact $\mathbf{n}_i$ et du coefficient de friction μ_i. La figure 9 montre des exemples de cônes de friction.

Phase de contact

Une *phase de contact* est définie comme un ensemble de contacts qui demeurent constants durant toute la phase de contact. Une phase de contact est notée $\{p\}$ avec $p \in \mathbb{N}$ et définit :

- $k^{\{p\}}$ le nombre de contacts ;
- $\mathbf{P}^{\{p\}} = [\mathbf{p}_1, \mathbf{p}_2, ...\mathbf{p}_k]$ l'ensemble des positions des points de contact $\mathbf{p}_i \in \mathbb{R}^3$;
- $\mathbf{N}^{\{p\}} = [\mathbf{n}_1, \mathbf{n}_2, ...\mathbf{n}_k]$ l'ensemble des normales aux contacts $\mathbf{n}_i \in \mathbb{R}^3$;

Chaque phase de contact contraint donc cinématiquement et dynamiquement les mouvements du robot puisque les contacts sont fixés.

En effet, les points de contact définis par une phase de contact vont limiter cinématiquement les mouvements possibles qui respectent le modèle de la chaîne cinématique du robot (ie. chaque corps est indéformable, et chaque liaison a un certain degré de liberté et des butées articulaires).

De plus, les positions et les normales de contact fixées vont définir des contraintes sur les forces de contact pour respecter la condition de non glissement.

Séquence de contacts

Le mouvement d'un robot à pattes est caractérisé par une succession de *phases de contact* que l'on nomme *séquence de contacts*. Au sein d'une séquence de contacts, chaque phase de contact diffère des phases adjacentes par exactement **un** changement de contact. Autrement dit, il ne peut y avoir qu'une seule création ou suppression de contact entre deux phases adjacentes. Dans le cas de contacts planaires, par simplicité nous contraignons les quatre points de contact exprimés aux quatre coins de l'organe terminal en contact à être supprimé ou créé en même temps.

Si l'on prend l'exemple d'un pas fait par un robot humanoïde, la séquence de contacts contient trois phases de contact : la phase initiale avec les deux pieds au sol (appelée *double support*), une phase en simple support avec la suppression du contact entre un des pieds et le sol, puis la phase finale en double support en créant un contact avec le pied qui a bougé à sa nouvelle position.

Lors d'un mouvement, le robot doit donc respecter les contraintes cinématiques et dynamiques imposées par la phase de contact courante. Le changement de phase de contact est instantané et le robot doit respecter les contraintes des deux phases de contact à cet instant.

Modèle centroïdal

Le centre de masse (CoM) d'un robot est un point très particulier pour l'expression de la dynamique du système. Il s'agit du point auquel s'exprime l'action de la gravité. Le CoM est aussi le point où le moment linéaire et cinétique du robot sont naturellement définis.

A partir de ces observations, de nombreux travaux se sont concentrés sur la *dynamique centroïdale* qui exprime la dynamique du robot projetée à son centre de masse. Une revue de ces contributions peut être trouvée dans [Orin 2013].

Notations

Nous définissons un *état* du modèle centroïdal par $\mathbf{x} = (\mathbf{c}, \dot{\mathbf{c}}, \ddot{\mathbf{c}}) \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3$ un triplet décrivant la position, vitesse et accélération du centre de masse.

A un état, nous pouvons associer une phase de contact $\{p\}$ avec $p \in \mathbb{N}$ (telle que défini dans la section précédente) en utilisant une notation avec un exposant : $\mathbf{x}^{\{p\}}$.

Lorsque l'on dit qu'un état associé à une phase de contact $\mathbf{x}^{\{p\}}$ est *cinématiquement valide* cela signifie que l'état centroïdal $\mathbf{x}$ satisfait toutes les contraintes

cinématiques imposées par la phase de contact $\{p\}$ (chaque corps est indéformable, et chaque liaison a un certain degré de liberté et des butées articulaires).

Lorsque l'on dit que $\mathbf{x}^{\{p\}}$ est *dynamiquement valide* cela signifie que l'état $\mathbf{x}$ permet de satisfaire les conditions de non glissement définis par la phase de contact $\{p\}$.

Lorsque l'on dit que $\mathbf{x}^{\{p\}}$ est *valide*, cela signifie que l'état respecte à la fois les contraintes cinématiques et dynamiques de la phase de contact $\{p\}$. Un même état peut être valide pour différentes phases de contact, il est alors noté $\mathbf{x}^{\{p,q\}}$.

Nous définissons une *trajectoire centroïdale* comme l'évolution de l'état centroïdal en fonction du temps, notée $\mathbf{x}(t)$. Une trajectoire centroïdale peut être associée à une séquence de contacts. Dans ce cas nous pouvons définir la *validité* de la trajectoire comme le fait qu'à chaque instant l'état centroïdal est *valide* pour la phase de contact courante.

Dynamique centroïdale

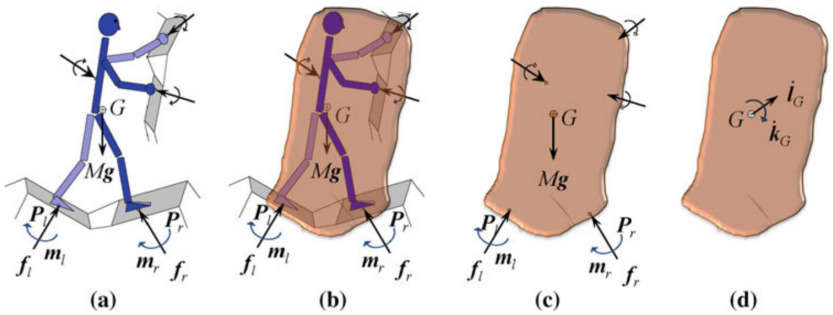


FIGURE 10 – Figure extraite de [Orin 2013]. (a) Modèle complet du robot soumis à des forces extérieures (gravité et interaction avec l'environnement) ainsi qu'à des forces internes (forces et couples exercés aux liaisons). (b) On suppose que le robot est un système fermé et on ne considère que les forces extérieures. (c) Forces extérieures appliquées au système. (d) La résultante de toutes les forces extérieures appliquée au centre de masse est une force et un moment.

L'idée principale de la représentation du modèle centroïdal est d'exprimer toutes les forces extérieures au centre de masse, comme présenté sur la figure 10. Le robot est alors considéré comme un seul corps rigide soumis à des forces extérieures.

La dynamique centroïdale peut alors être exprimée via les équations de Newton-Euler qui lient les variations du moment linéaire et cinétique d'un corps aux forces extérieures :

$$\begin{aligned} m\ddot{\mathbf{c}} &= \sum_{i=1}^{k} \mathbf{f}_i + m\mathbf{g} \\ \dot{L} &= \sum_{i=1}^{k} (\mathbf{p}_i - \mathbf{c}) \times \mathbf{f}_i \end{aligned} \tag{4}$$

Avec :
— $(\mathbf{c}, \dot{\mathbf{c}}, \ddot{\mathbf{c}})$ la position, vitesse et accélération du centre de masse ;
— $\dot{L}$ la dérivée du moment cinétique ;
— $\mathbf{g} = [0, 0, -9.81]$ le vecteur de gravité ;
— k le nombre de points de contact ;
— $\mathbf{f}_i$ la i-ième force de contact ;
— $\mathbf{p}_i$ la position du i-ième contact ;
— m la masse totale du robot.

Nous pouvons réarranger les équations de Newton-Euler comme suit :

$$\begin{aligned} m(\ddot{\mathbf{c}} - \mathbf{g}) &= \sum_{i=1}^{k} \mathbf{f}_i \\ \mathbf{c} \times \sum_{i=0}^{k} \mathbf{f}_i + \dot{L} &= \sum_{i=1}^{k} \mathbf{p}_i \times \mathbf{f}_i \end{aligned} \tag{5}$$

Puis remplacer $\sum_{i=1}^{k} \mathbf{f}_i$ dans l'équation du bas par son expression dans l'équation du haut et obtenir :

$$\begin{aligned} m(\ddot{\mathbf{c}} - \mathbf{g}) &= \sum_{i=1}^{k} \mathbf{f}_i \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) + \dot{L} &= \sum_{i=1}^{k} \mathbf{p}_i \times \mathbf{f}_i \end{aligned} \tag{6}$$

Ces équations peuvent finalement s'écrire sous une forme matricielle :

$$\underbrace{\begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) + \dot{\mathbf{L}} \end{bmatrix}}_{\mathbf{w}} = \begin{bmatrix} \mathbf{I}_3 & \dots & \mathbf{I}_3 \\ \hat{\mathbf{p}}_1 & \dots & \hat{\mathbf{p}}_k \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_k \end{bmatrix}}_{\mathbf{f}} \tag{7}$$

$\mathbf{w}$ est appelé le Gravito-Intertial-Wrench (GIW) [Caron 2015]. Les $\hat{\mathbf{p}}_i$ sont les matrices de préproduit vectoriel associés aux points de contact $\mathbf{p}_i$, $\mathbf{I}_3$ est la matrice Identité de taille 3×3.

Contraintes dynamiques

Afin de respecter les contraintes de non-glissement présentées dans la section précédente, il faut que chaque force $\mathbf{f}_i$ de l'équation (7) soit comprise dans le cône de friction associé au i-ième contact :

$$\mathbf{f}_i \in \mathcal{K}_i \quad , \quad \forall i \tag{8}$$

Cela revient à vérifier l'équation suivante :

$$\sqrt{(\mathbf{t}_{i1}^\top \mathbf{f}_i)^2 + (\mathbf{t}_{i2}^\top \mathbf{f}_i)^2} \leq \mu_i \mathbf{n}_i^\top \mathbf{f}_i \quad \forall i \in [0; k] \tag{9}$$

Avec μ_i le coefficient de friction, $\mathbf{n}_i \in \mathbb{R}^3$ la direction de la normale au i-ième point de contact et $(\mathbf{t}_{i1}, \mathbf{t}_{i2})$ les directions tangentes.

Pour vérifier si de telles forces existent, une approche possible serait de résoudre un *Second-Order Cone Program* (SOCP) [Bretl 2008], ou un *Quadratically Constrained Quadratic Program* (QCQP) plus efficace comme proposé par [Ponton 2016]. Une autre approche consiste à approximer les cônes de friction en les représentant par des polytopes, cette approche est couramment utilisée dans la littérature [Caron 2015, Qiu 2011, Del Prete 2016, Bretl 2006a] et permet d'utiliser des formulations linéaires plus simple.

On peut alors représenter les forces de contact sous la forme suivante :

$$\mathbf{f}_i = \mathbf{V}_i \boldsymbol{\beta}_i \text{ avec } \boldsymbol{\beta}_i \in \mathbb{R}^{n_g} \text{ et } \boldsymbol{\beta}_i \geq 0 \tag{10}$$

avec $\mathbf{V}_i \in \mathbb{R}^{3 \times n_g}$ une matrice définie par les génératrices du cône linéarisé ,où n_g est le nombre de génératrices utilisées lors de la linéarisation, comme fait classiquement nous fixons $n_g = 4$. En empilant toutes les forces de contact dans un vecteur, la contrainte d'appartenance des forces aux cônes linéarisés peut alors s'écrire comme un ensemble d'inégalités linéaires :

$$\mathbf{f} = \mathbf{V}\boldsymbol{\beta} \text{ avec } \boldsymbol{\beta} \in \mathbb{R}^{4k} \text{ et } \boldsymbol{\beta} \geq 0 \tag{11}$$

Avec $\mathbf{V} = diag(\begin{bmatrix} \mathbf{V}_1 & ... & \mathbf{V}_k \end{bmatrix}) \in \mathbb{R}^{3k \times 4k}$.

En utilisant les cônes de friction linéarisés, la contrainte de non glissement sur la dynamique centroïdale peut alors s'écrire :

$$\exists \boldsymbol{\beta} \geq 0 \textbf{ s.t. } \begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) + \dot{\mathbf{L}} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_3 & ... & \mathbf{I}_3 \\ \hat{\mathbf{p}}_1 & ... & \hat{\mathbf{p}}_k \end{bmatrix} \mathbf{V}\boldsymbol{\beta} \tag{12}$$

Vocabulaire spécifique

Chemin et trajectoire

Par abus de langage, durant cette thèse nous utiliserons les termes anglo-saxons afin de différencier un chemin d'une trajectoire.

Un chemin (de l'anglais *path*) est défini comme l'ensemble des positions successives occupées par un point. Un chemin ne contient aucune notion de temporalité et est un objet purement géométrique.

Une trajectoire (de l'anglais *trajectory*) est un chemin associé à une loi de mouvement. Dans notre définition une trajectoire est donc paramétrisée temporellement et donne l'évolution de la position d'un point en fonction du temps.

Différents types de planification

Dans l'ensemble de cette thèse le terme planification sera employé pour la planification de chemin, de trajectoire ou de mouvement. Bien que ces termes soient proches, ils ne sont pas interchangeables :

— La planification de **chemin** désigne une approche purement géométrique. Le **chemin** résultant n'ayant aucune notion de temporalité ou de dynamique.
— La planification de **trajectoire** désigne une approche planifiant l'évolution de la position d'un point en fonction du temps.
— Le terme planification de **mouvement** est généralement utilisé dans le cas de systèmes articulés, pour désigner une méthode qui génère un mouvement pour l'ensemble des articulations.

Temps interactif

Le terme de *temps de calcul interactif* ou *performances interactives* signifie que le temps de calcul d'une trajectoire ou d'un mouvement est inférieur à la durée de ce mouvement.

Ce terme est à différencier du terme *temps réel* qui signifie qu'une méthode peut s'exécuter à une fréquence fixée (généralement la fréquence du contrôle, de quelques kilohertz).

Quasi-Statique

Un état ou une configuration *statique* est un état pour lequel l'accélération et la vitesse sont nulles. Un état en équilibre *statique* est un état qui respecte les contraintes dynamiques présentées dans la section Contraintes dynamiques avec une vitesse et une accélération nulle.

Par extension une trajectoire ou un mouvement *quasi-statique* désigne une trajectoire lente, durant laquelle la vitesse et l'accélération sont supposées négligeables.

Ce type de trajectoire est bien plus simple à générer car les équations de la dynamique peuvent se simplifier grandement en négligeant les termes liées à l'accélération.

Première partie

Planification d'une trajectoire guide dynamiquement faisable

Le premier sous-problème ($\mathcal{P}_1$) de l'approche découplée présentée dans l'introduction consiste à planifier une trajectoire guide. Cette trajectoire est ensuite utilisée en entrée de la méthode de planification de contacts ($\mathcal{P}_2$) qui tente de générer des configurations en contact le long de cette trajectoire guide.

L'objectif de cette partie du manuscrit est de proposer une méthode de planification pour le sous-problème $\mathcal{P}_1$ permettant de garantir que la trajectoire guide planifiée sera une entrée **faisable** pour le problème de planification de contact $\mathcal{P}_2$. C'est à dire qu'il existe une séquence de configurations en contact, cinématiquement et dynamiquement valides, le long de la trajectoire guide planifiée.

Chapitre 1

Introduction aux algorithmes de planification

Sommaire

Afin de déplacer un robot dans un environnement contraint, il est nécessaire de générer un mouvement sans collision. Ce problème est couramment défini comme de la planification de mouvement et est illustré dans le problème du "déménageur de piano" présenté par [Schwartz 1983] : «Étant donnés un objet, un environnement et un placement (position et orientation) initial et final pour l'objet, une méthode de planification de mouvement doit, soit générer un mouvement tel que l'objet peut rejoindre le placement final sans entrer en collision avec l'environnement, soit conclure qu'un tel mouvement est impossible.»

La complexité de ce problème vient de l'infinité de placements possibles pour l'objet. Si l'objet devient articulé, le nombre de degrés de liberté augmente et la complexité du problème également. Dans la section suivante, les algorithmes de base de planification de mouvement, sur lesquelles se basent les méthodes actuelles, sont introduits brièvement.

En plus de l'aspect géométrique, nous nous intéressons aussi à planifier un mouvement qui soit dynamiquement valide. Certains algorithmes de planification existants peuvent considérer la dynamique du robot. Dans la section 1.2 nous présentons l'état de l'art de ces algorithmes, permettant de résoudre ce que l'on appelle un problème de planification kinodynamique.

Dans la section 1.3 nous montrons pourquoi ces algorithmes de planifications classiques ne peuvent pas être utilisés directement pour planifier des mouvements de robots à pattes.

Il est donc nécessaire de développer de nouvelles méthodes spécifiques pour ce type de mouvements. L'approche que nous avons choisie étant de découpler le problème et de commencer par planifier une trajectoire guide (sous problème $\mathcal{P}_1$), un état de l'art des méthodes existantes est fait dans la section 1.4.

Cet état de l'art nous permet de sélectionner la méthode qui nous semble la plus avantageuse à développer par rapport à notre approche de la planification pour la locomotion de robots à pattes, cette méthode nommée RB-RRT est détaillée dans la section 1.5.

Enfin, nous montrons que le problème de la planification d'une trajectoire guide pouvant être utilisée par la génération de contact reste un problème ouvert. En effet, aucune des méthodes présentées ne peut garantir l'existence de configurations en contact valides, sans se limiter au cas quasi-statique, le long de la trajectoire guide planifiée.

1.1 Planification géométrique

Nous rappelons que nous appelons un robot une chaîne cinématique articulée composée de N degrés de liberté dont la position et l'orientation de chacun de ses corps est décrite par une configuration $\mathbf{q} \in \mathcal{C}$. L'espace libre est l'espace défini par $\mathcal{C}_{free} \subset \mathcal{C}$ tel que $\forall \mathbf{q} \in \mathcal{C}_{free}$, $\mathbf{q}$ est sans collision avec l'environnement.

Le problème de planification revient donc à trouver un chemin entre deux points $\mathbf{q}_s \in \mathcal{C}_{free}$ et $\mathbf{q}_g \in \mathcal{C}_{free}$ tel que ce chemin est entièrement compris dans l'espace libre.

En se basant sur cette notion d'espace des configurations, de nombreux algorithmes probabilistes ont été développés. Une étude précise de l'état de l'art peut être trouvé dans le livre [LaValle 2006]. On peut noter les deux algorithmes les plus connus : *Probabilistic Road Maps* (PRM) [Kavraki 1996] et *Rapidly Exploring Random Tree* (RRT) [LaValle 1998].

Dans le reste de cette section, l'algorithme RRT est détaillé puisque nos développements se basent sur cette méthode. Cependant, de nombreuses notions détaillées seront communes avec les autres algorithmes de planification probabiliste.

Rapidly Exploring Random Tree

Comme les autres algorithmes de planification probabiliste, RRT procède en générant aléatoirement des configurations $\mathbf{q}_{rand} \in \mathcal{C}_{free}$ puis en essayant de les connecter dans une structure de graphe nommée Roadmap. Dans une roadmap, chaque nœud est une configuration $\mathbf{q} \in \mathcal{C}_{free}$ et chaque arc est un chemin valide qui connecte les configurations des nœuds qu'il relie. La définition de la validité d'un chemin ou d'une configuration est propre à chaque problème, il s'agit généralement

d'éviter les collisions mais toutes sortes d'autres contraintes peuvent être ajoutées au problème.

Dans nos travaux nous utilisons la variante BIdirectionnal-RRT qui construit simultanément deux arbres : l'un ayant pour racine la configuration initiale, et l'autre la configuration finale. L'algorithme s'arrête lorsque les deux arbres sont connectés entre eux. Différent choix d'implémentation existent, nous ne montrons dans l'algorithme 1 que les choix retenus et implémentés sans entrer dans les détails.

Algorithm 1 BI-RRT($\mathbf{q}_{init}$, $\mathbf{q}_{goal}$)

Ensure: Solution path $\tau | (\tau(0) = \mathbf{q}_{init}, \tau(1) = \mathbf{q}_{goal})$ or *fail*
1: $\mathcal{G}_a$.init($\mathbf{q}_{init}$) ; $\mathcal{G}_b$.init($\mathbf{q}_{goal}$) ;
2: **while** ¬ solved **and** $iter < N_{max}$ **do**
3: $\mathbf{q}_{rand} \leftarrow$ RANDOM_CONFIG() ;
4: $S_a \leftarrow$ EXTEND($\mathcal{G}_a$,$\mathbf{q}_{rand}$) ;
5: $S_b \leftarrow$ EXTEND($\mathcal{G}_b$,$\mathbf{q}_{rand}$) ;
6: **if** ($S_a = Reached$) **and** ($S_b = Reached$) **then**
7: **Return** PATH($\mathcal{G}_a$,$\mathcal{G}_b$,$\mathbf{q}_{init}$, $\mathbf{q}_{goal}$) ;
8: **end if**
9: **end while**
10: **Return** *fail* ;

Algorithm 2 EXTEND($\mathcal{G}$, $\mathbf{q}$)

1: $\mathbf{q}_{near} \leftarrow$ NEAREST_NEIGHBOR($\mathcal{G}$, $\mathbf{q}$) ;
2: $\tau \leftarrow$ STEERING_LOCAL($\mathbf{q}_{near}$,$\mathbf{q}$) ;
3: $\tau', \mathbf{q}_{new} \leftarrow$ PATH_VALIDATION(τ) ;
4: **if** $\mathbf{q}_{new} \neq \mathbf{q}_{near}$ **then**
5: $\mathcal{G}$.add_vertex($\mathbf{q}_{new}$) ;
6: $\mathcal{G}$.add_edge($\mathbf{q}_{near}$,$\mathbf{q}_{new}$,τ') ;
7: **if** $\mathbf{q}_{new} = \mathbf{q}$ **then**
8: **Return** *Reached* ;
9: **else**
10: **Return** *Advanced* ;
11: **end if**
12: **else**
13: **Return** *Trapped* ;
14: **end if**

Dans l'algorithme 1, la fonction RANDOM_CONFIG retourne une configuration $\mathbf{q} \in \mathcal{C}_{free}$. La fonction PATH utilise un algorithme de type A^* afin de parcourir le graphe et de trouver le chemin qui permet de rejoindre $\mathbf{q}_{goal}$ depuis $\mathbf{q}_{init}$.

L'algorithme 2 est la méthode d'extension de la roadmap. Elle essaie de connecter une configuration aléatoire donnée à un graphe et peut obtenir trois résultats :

— Réussir à connecter la configuration donnée, dans ce cas elle retourne *Reached* et ajoute la configuration au graphe ;

— Progresser dans la direction de la configuration donnée mais sans l'atteindre, dans ce cas elle ajoute une nouvelle configuration au graphe et retourne *Advanced* ;
— Échouer et retourner *Trapped*.

Dans l'algorithme 2, la fonction NEAREST_NEIGHBOR retourne le nœud appartenant du graphe considéré qui est le plus proche de la configuration donnée en paramètre, selon une métrique définie par le problème.

La méthode STEERING_LOCAL est appelée la **méthode de planification locale**, il s'agit d'une méthode qui génère un chemin connectant deux configurations données. Dans les cas d'utilisation classique, il s'agit d'une simple interpolation linéaire.

La méthode PATH_VALIDATION vérifie si un chemin donné est sans collision et respecte des potentielles contraintes du problème (par exemple des bornes sur les limites articulaires). Elle retourne la partie valide du chemin et la dernière configuration valide.

La figure 1.1 montre la progression de l'algorithme au fur et à mesure des itérations. Comme montré sur la figure 1.1 en bas à gauche, la solution trouvée par un algorithme de planification n'est jamais optimale puisqu'elle connecte des points échantillonnés aléatoirement. Pour remédier à cela, des algorithmes d'optimisation sont utilisés sur la solution trouvée, des détails sur ces algorithmes peuvent être trouvés dans le livre [LaValle 2006]. L'un des plus couramment utilisé est appelé "Random shortcut" et consiste à essayer de remplacer des portions du chemin trouvé par des segments de droite. Le résultat de cette optimisation est montré sur la figure 1.1 en bas à droite.

1.2 Planification Kinodynamique

La planification kinodynamique désigne les méthodes de planification qui considèrent l'aspect dynamique du robot en plus de l'aspect géométrique. Un algorithme RRT kinodynamique est présenté dans [LaValle 1999]. L'algorithme est identique à celui présenté dans la section précédente à la différence qu'au lieu de considérer l'espace des configurations avec $\mathbf{q} \in \mathbb{R}^N$, il considère l'espace d'état avec $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{2\times N}$.

De plus, au lieu de produire un chemin il produit une trajectoire dépendante du temps, qui respecte la dynamique du système définie par l'équation :

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \tag{1.1}$$

Avec $\mathbf{u}$ la variable de contrôle du système.

La différence fondamentale réside dans la méthode de planification locale choisie. En effet, dans le cas de trajectoires on ne peut plus se contenter d'utiliser une mé-

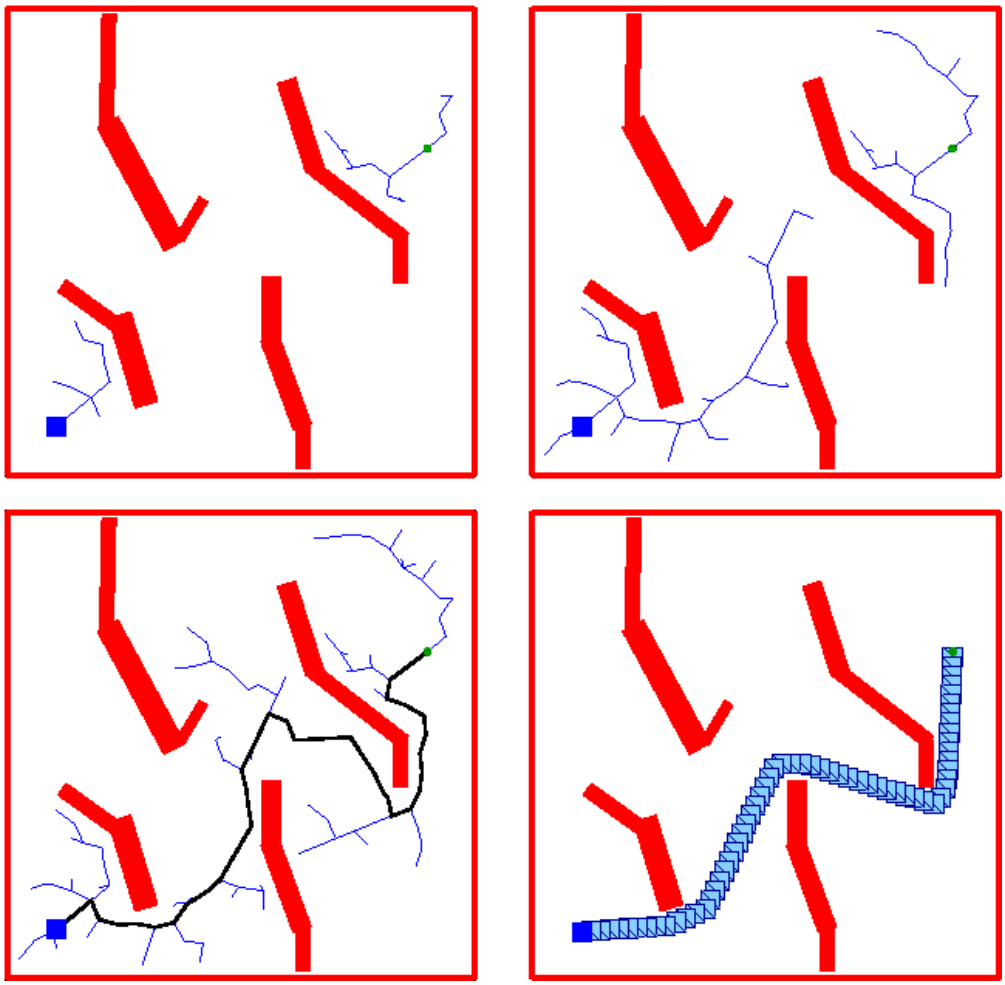


FIGURE 1.1 – Figure extraite de [Kuffner Jr 2000]. Les figures du haut montrent l'état des Roadmaps pour deux nombres différents d'itérations de l'algorithme. Le figure en bas à gauche montre l'état des Roadmap au moment de leur connection (succès de l'algorithme) et en noir le chemin trouvé dans le graphe via l'algorithme A^*. La figure en bas à droite montre le chemin final après optimisation.

thode d'interpolation linéaire. Il faut utiliser une méthode permettant de connecter exactement deux états en respectant les vitesses et accélérations initiales et finales avec une trajectoire qui respecte la dynamique du système de l'équation (1.1).

Dans nos travaux, nous nous sommes intéressés à la méthode Double Integrator Minimum Time (DIMT) qui est présentée dans [Kröger 2006, Hauser 2010]. [Kunz 2014] donne les détails des équations et de l'implémentation qui n'étaient pas présents dans les articles précédents et nous nous sommes basés sur son travail qui sera détaillé dans la sous-section suivante.

L'intérêt de la méthode DIMT est de calculer des trajectoires connectant exactement deux états en respectant des bornes d'accélération et de vitesse. La trajectoire

calculée est optimale en temps et se compose de phases avec une amplitude de l'accélération maximale ou nulle (couramment appelée "trajectoire bang-bang").

Un autre intérêt de la méthode DIMT est qu'elle propose une métrique de l'espace d'état peu coûteuse à calculer : il s'agit du temps minimal requis pour connecter deux états via une trajectoire produite par la méthode de planification locale.

1.2.1 Double Integrator Minimum Time (DIMT)

Étant donné des bornes manuellement définies et symétriques sur la dynamique du centre de masse selon trois axes orthogonaux :

$$\begin{array}{rcl} -\dot{c}^{max}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq & \dot{c}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq & \dot{c}^{max}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \\ -\ddot{c}^{max}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq & \ddot{c}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq & \ddot{c}^{max}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \end{array} \tag{1.2}$$

et étant donné un état centroïdal initial $\mathbf{x}_0 =< \mathbf{c}_0, \dot{\mathbf{c}}_0, \mathbf{0} >$ et un état final $\mathbf{x}_1 =< \mathbf{c}_1, \dot{\mathbf{c}}_1, \mathbf{0} >$, la méthode DIMT calcul une trajectoire optimale en temps $\mathbf{x}(t)$ qui connecte exactement $\mathbf{x}_0$ et $\mathbf{x}_1$. Cette trajectoire consiste, selon chaque axe, en une succession de phases à accélération constante ou nulle. L'accélération à l'état initial ou final est sans importance pour ce problème, puisque cette méthode produit une trajectoire où l'accélération n'est pas continue mais constante par morceaux, et que nous n'imposons aucune contrainte sur le jerk (la dérivée de l'accélération). Par simplicité nous la fixons à 0.

La trajectoire calculée est garantie de respecter les bornes de vitesse et d'accélération imposées. Mais la méthode ne considère pas l'environnement et ne garantit pas que la trajectoire soit sans collision.

La méthode *Double Integrator Minimum Time* fonctionne en deux étapes : premièrement le calcul du temps minimum requis pour atteindre l'état final selon chaque axe puis le calcul du temps minimum global nécessaire. Deuxièmement, le calcul des trajectoires pour chaque axe qui permet de rejoindre l'état final au temps imposé.

1.2.1.1 Calcul du temps minimum nécessaire

Pour commencer, il faut calculer le temps requis par chaque axe pour atteindre l'état final en supposant une trajectoire qui ne comporte que deux phases : accélération puis décélération maximale selon la borne imposée (aussi appelé trajectoire "bang-bang"). Ensuite, il faut vérifier si la borne de vitesse n'est pas atteinte et ajouter si nécessaire une phase à accélération nulle et vitesse constante.

Enfin, il faut calculer les possibles intervalles infaisables : il s'agit d'intervalles de temps propres à chaque axe, supérieur au temps minimum requis pour cet axe, durant lesquels l'état final ne peut pas être atteint.

Finalement, le temps minimum global est le maximum des temps requis pour chaque axe qui ne se trouve pas dans un intervalle infaisable.

La première étape pour calculer le temps minimum à accélération maximale pour un axe donné est de déterminer le signe de l'accélération initiale nécessaire. Pour cela, l'équation suivante est utilisée :

$$\begin{aligned} \Delta p_{acc} &= \frac{1}{2}(\dot{c}_0 + \dot{c}_1)\frac{|\dot{c}_1 - \dot{c}_0|}{\ddot{c}^{max}} \\ \sigma &= sign(c_1 - c_0 - \Delta p_{acc}) \end{aligned} \tag{1.3}$$

Où σ donne le signe de l'accélération de la première phase. Nous pouvons donc calculer une trajectoire composée de deux phases à accélération constante :

$$\ddot{c}(t) = \begin{cases} \sigma\ddot{c}^{max} & ,\forall t \in [0; t_{a_0}] \\ -\sigma\ddot{c}^{max} & ,\forall t \in]t_{a_0}; t_{a_0} + t_{a_1}] \end{cases} \tag{1.4}$$

Il faut à présent calculer ces temps t_{a_0} et t_{a_1}, pour cela il faut résoudre l'équation quadratique suivante :

$$\sigma\ddot{c}^{max}t_{a_0}^2 + 2\dot{c}_0 t_{a_0} + \frac{\dot{c}_1^2 - \dot{c}_0^2}{-2\sigma\ddot{c}^{max}} - (c_1 - c_0) = 0 \tag{1.5}$$

Cette équation admet deux solutions mais une seule est valide. En effet t_{a_0} et t_{a_1} doivent être positifs et t_{a_1} s'exprime de la manière suivante :

$$t_{a_1} = \frac{\dot{c}_1 - \dot{c}_0}{-\sigma\ddot{c}^{max}} + t_{a_0} \tag{1.6}$$

La condition $t_{a_1} > 0$ peut alors être transformée en borne minimale sur t_{a_0} comme suit :

$$t_{a_0} > \frac{\dot{c}_1 - \dot{c}_0}{\sigma\ddot{c}^{max}} \tag{1.7}$$

Ce qui permet de choisir la seule solution correcte pour t_{a_0} de l'équation (1.5).

Il faut maintenant vérifier si cette trajectoire viole les bornes de vitesse imposées. Pour cela, il suffit de vérifier la vitesse au moment du changement de signe de l'accélération t_{a_0} :

$$\dot{c}(t_{a_0}) = \dot{c}_0 + t_{a_0}\sigma\ddot{c}^{max} \tag{1.8}$$

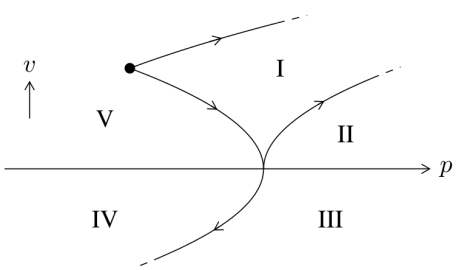
Si $-\dot{c}^{max} \leq \dot{c}(t_{a_0}) \leq \dot{c}^{max}$ alors le temps minimum requis pour cet axe est $T = t_{a_0} + t_{a_1}$. Sinon, il faut ajouter une phase à accélération nulle et vitesse constante et égale à la vitesse limite, de durée t_v. Les temps de chacune des nouvelles phases sont donnés par les equations suivantes :

$$
\begin{aligned}
t_{a_0} &= \frac{\sigma \dot{c}^{max} - \dot{c}_0}{\sigma \ddot{c}^{max}} \\
t_v &= \frac{\dot{c}_0^2 + \dot{c}_1^2 - 2\sigma(\dot{c}^{max})^2}{2\dot{c}^{max}\ddot{c}^{max}} + \frac{c_1 - c_0}{\sigma \dot{c}^{max}} \\
t_{a_1} &= \frac{\dot{c}_1 - \sigma \dot{c}^{max}}{-\sigma \ddot{c}^{max}}
\end{aligned}
\tag{1.9}
$$

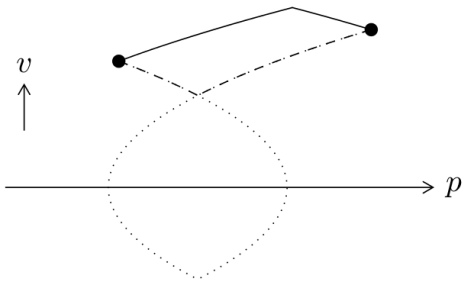
et la durée minimale pour cet axe est $T = t_{a_0} + t_v + t_{a_1}$

1.2.1.2 Intervalles infaisables

Un axe présente un intervalle infaisable si et seulement si l'état final appartient à la région I représentée dans la figure 1.2 - a. La figure 1.2 - b montre la raison de l'existence d'un tel intervalle. En effet, il est possible d'augmenter la durée de la trajectoire pour atteindre l'état final en diminuant l'amplitude de l'accélération appliquée, mais seulement entre la trajectoire en trait plein et celle en tirets. Pour augmenter davantage la durée de la trajectoire tout en atteignant exactement l'état final il faut alors effectuer un arrêt complet, inverser le signe de la vitesse puis accélérer à nouveau en direction de l'état final. Cette trajectoire prend alors un certain temps supplémentaire, ce qui donne lieu à cet intervalle de temps infaisable durant lequel on ne peut pas atteindre l'état final (la position et la vitesse finale simultanément).



(a) Différentes régions du plan de phase selon la position de l'état final par rapport à l'état initial. Les trajectoires représentées sont toutes à accélération maximale ou minimale.



(b) Exemple d'intervalle infaisable, avec les trajectoires calculée pour le temps minimal (trait plein) la borne minimum de intervalle infaisable (tirets) et la borne supérieur de intervalle infaisable (pointillés).

FIGURE 1.2 – Intervalle infaisable : Figures provenant de [Kunz 2014]

Afin de calculer les bornes de cet intervalle quand il existe, il faut résoudre à nouveau les équations (1.5 à 1.9) en inversant le signe de σ. Les bornes inférieures et supérieures sont alors données par les deux solutions possibles pour le temps total T.

Il faut noter que ces intervalles n'existent que parce que la méthode DIMT ne considère que des trajectoires en deux segments à accélération constante ou à trois segments avec le segment du milieu à accélération nulle.

Exemple d'intervalle infaisable

Considérons le problème suivant, selon un seul axe : l'état initial est $(c_0 = 0; \dot{c}_0 = 1)$ et l'état final est $(c_1 = 1; \dot{c}_1 = 1)$. Les bornes d'accélération sont $\ddot{c}_{max} = 0.5$ et de vitesse $\dot{c}_{max} = 2$. Le temps minimum pour connecter ces deux états est de $T = 0.9$ secondes, avec une trajectoire en deux segments : $\ddot{c}_0 = 0.5$, $t_{a0} = 0.45$ et $\ddot{c}_1 = -0.5$, $t_{a1} = 0.45$.

Maintenant il est possible d'augmenter la durée de cette trajectoire de façon continue jusqu'à $T = 1.172$ secondes, en diminuant progressivement l'accélération jusqu'à $\ddot{c}_0 = -0.5$. La trajectoire obtenue pour $T = 1.172$ contient également deux segments : $\ddot{c}_0 = -0.5$, $t_{a0} = 0.5858$ et $\ddot{c}_1 = 0.5$, $t_{a1} = 0.5858$.

Cependant si l'on désire augmenter encore la durée de la trajectoire, il faudrait exercer une accélération inférieure à la borne minimale. La seule autre solution pour augmenter la durée totale de la trajectoire est de faire une "boucle", comme la trajectoire en pointillés de la figure 1.2-b. Cette boucle consiste à inverser le signe de la vitesse, à s'éloigner de l'état final puis à accélérer à nouveau en direction de l'état final jusqu'à l'atteindre exactement.

Dans notre exemple, la durée minimum de cette boucle (faite avec une amplitude de l'accélération maximale possible) est de $T = 6.83$ secondes, avec la trajectoire en deux segments suivante : $\ddot{c}_0 = -0.5$, $t_{a0} = 3.415$ et $\ddot{c}_1 = 0.5$, $t_{a1} = 3.415$. Après cette durée minimale pour réaliser la boucle, il est à nouveau possible d'augmenter de manière continue la durée de la trajectoire, en diminuant progressivement l'amplitude de l'accélération.

Il y a donc un intervalle infaisable pour $T \in [1.172; 6.83]$ durant lequel il est impossible d'atteindre en même temps la position et la vitesse finale, avec les bornes d'accélérations utilisées $[-0.5; 0.5]$.

1.2.1.3 Calcul de trajectoires pour temps fixé

Le temps minimum global requis est donc le maximum entre les temps minimaux pour chaque axe, et qui ne se trouve pas dans un intervalle infaisable.

Maintenant que ce temps a été déterminé, il convient de calculer des trajectoires pour chaque axe qui atteignent l'état final pour ce temps fixé. Il existe une infinité de solutions possibles à ce problème mais nous avons choisi d'utiliser la même solution que [Kunz 2014] et [Hauser 2010] : la trajectoire qui minimise la valeur absolue maximale de l'accélération.

Pour calculer une telle trajectoire, il faut tout d'abord déterminer la valeur de l'accélération de la première phase $\ddot{c}_0$ requise en résolvant l'équation quadratique

suivante :

$$T^2\ddot{c}_0^2 + (2T(\dot{c}_0 + \dot{c}_1) - 4(c_1 - c_0))\ddot{c}_0 - (\dot{c}_1 - \dot{c}_0)^2 = 0 \tag{1.10}$$

La solution avec la plus grande valeur absolue donne la valeur de $\ddot{c}_0$ et $\ddot{c}_1 = -\ddot{c}_0$. Les durées de chacune des phases sont données par les équations suivantes :

$$\begin{aligned} t_{a_0} &= \frac{1}{2}\left(\frac{\dot{c}_1 - \dot{c}_0}{\ddot{c}_0} + T\right) \\ t_{a_1} &= T - t_{a_0} \end{aligned} \tag{1.11}$$

De même que précédemment, il faut vérifier si la trajectoire respecte les bornes de vitesse. Si ce n'est pas le cas les nouvelles accélérations sont données par l'équation suivante :

$$\ddot{c}_0 = -\ddot{c}_1 = \frac{(\dot{c}^{limit} - \dot{c}_0)^2 + (\dot{c}^{limit} - \dot{c}_1)^2}{2(\dot{c}^{limit}T - (c_1 - c_0))} \tag{1.12}$$

Avec $\dot{c}^{limit} = sign(\ddot{c}_0)\dot{c}^{max}$. Les durées de chaque phase sont calculées via les équations (1.9) avec les nouvelles valeurs d'accélération calculées ci-dessus.

La trajectoire obtenue permet alors d'atteindre les positions et vitesses désirées pour chacun des axes simultanément.

1.2.2 Génération de trajectoires balistiques

Un cas extrême de mouvement dynamique est le saut. Bien que la capacité de planifier ce genre de mouvements ne soit pas de la plus haute importance pour la robotique, cela reste un problème intéressant pour ce domaine. Par contre, dans le domaine de l'animation graphique la planification de sauts est une composante importante et très intéressante.

Afin de pouvoir planifier des sauts, il faut tout d'abord pouvoir générer une trajectoire balistique connectant deux positions. Dans [Campana 2016b], un formalisme pour générer de telles trajectoires est présenté. En utilisant ce formalisme nous avons proposé une méthode de planification reposant entièrement sur des trajectoires balistiques, planifiant ainsi une série de sauts respectant des contraintes dynamiques et cinématiques et permettant de connecter une position initiale et une position but dans [Campana 2016a].

Dans le reste de cette sous-section nous résumons donc le formalisme utilisé pour générer des trajectoires balistiques présenté dans [Campana 2016b] et qui sera utilisé dans la section 2.3.2.

1.2.2.1 Définition et notations

Une trajectoire balistique, ou trajectoire de vol, est définie comme une trajectoire durant laquelle la seule force s'appliquant au système est la force de pesanteur.

Dans la formulation que nous allons présenter, nous utilisons un modèle de masse ponctuelle : toute la masse du modèle est exprimée en son centre géométrique.

Ainsi, nous pouvons obtenir l'équation de la trajectoire balistique en dérivant la seconde loi de Newton :

$$\mathbf{c}(t) = \frac{1}{2}\mathbf{g}t^2 + \dot{\mathbf{c}}_s t + \mathbf{c}_s \tag{1.13}$$

Où $\mathbf{c}(t)$ est la trajectoire du centre du robot, $\mathbf{c}_s$ et $\dot{\mathbf{c}}_s$ sont la position et la vitesse initiale du robot. Les développements suivants sont extraits de [Campana 2016b].

La trajectoire balistique décrite par l'équation (1.13) forme une parabole contenue dans un plan, noté π_θ. Le problème peut donc être transformé en un problème 2D en opérant les changements de variables présentés sur la figure 1.3.

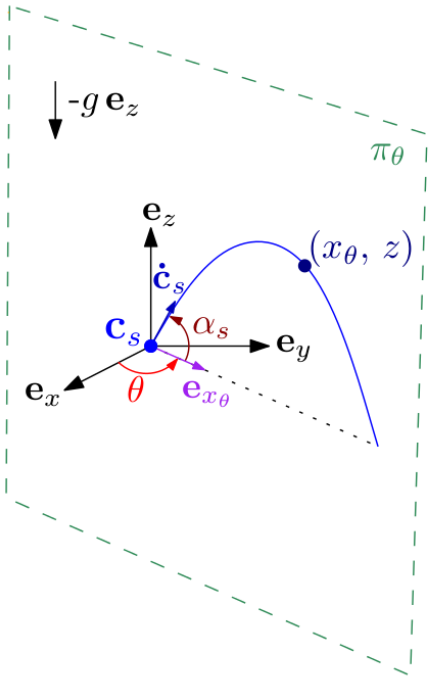


FIGURE 1.3 – Figure extraite de [Campana 2016b]. La parabole est comprise dans le plan π_θ définie par $(\mathbf{c}_0; \mathbf{e}_{x_\theta}; \mathbf{e}_z)$, avec $\mathbf{e}_{x_\theta} = cos(\theta)\mathbf{e}_x + sin(\theta)\mathbf{e}_y$.

Nous pouvons ensuite ré-écrire l'équation (1.13) sous forme paramétrique :

$$z = z_0 - \frac{1}{2}g\frac{(x_\theta - x_{\theta s})^2}{\dot{x}_{\theta s}^2} + tan(\alpha_s)(x_\theta - x_{\theta s}) + z_s \tag{1.14}$$

Avec :

$$
\begin{aligned}
tan(\alpha_s) &= \frac{\dot{z}_s}{\dot{x}_{\theta s}} \\
\frac{\dot{z}}{\dot{x}_{\theta s}} &= -g\frac{(x_\theta - x_{\theta s})}{\dot{x}_{\theta s}^2} + tan(\alpha_s)
\end{aligned}
\tag{1.15}
$$

Ce qui montre que chaque parabole est uniquement définie par l'angle de tir initial α_s, la vitesse horizontale initiale $\dot{x}_{\theta s}$ et l'angle θ.

1.2.2.2 Trajectoires rejoignant une position but

Nous voulons maintenant planifier une trajectoire balistique qui connecte exactement deux positions $\mathbf{c}_s$ et $\mathbf{c}_g$. Dans ce cas, l'angle θ est connu et on a la relation suivante entre entre $\dot{x}_{\theta s}$ et $tan(\alpha_s)$:

$$
\dot{x}_{\theta s} = \sqrt{\frac{g(x_{\theta g} - x_{\theta s})^2}{2(x_{\theta g} - x_{\theta s})tan(\alpha_s) - (z_g - z_s)}}
\tag{1.16}
$$

Il existe bien une infinité de solutions pour connecter $\mathbf{c}_s$ et $\mathbf{c}_g$ via une parabole mais chaque parabole est uniquement définie par la valeur de l'angle initiale de tir α_s. La figure 1.4 montre un faisceau de paraboles générées pour différentes valeurs de α_s.

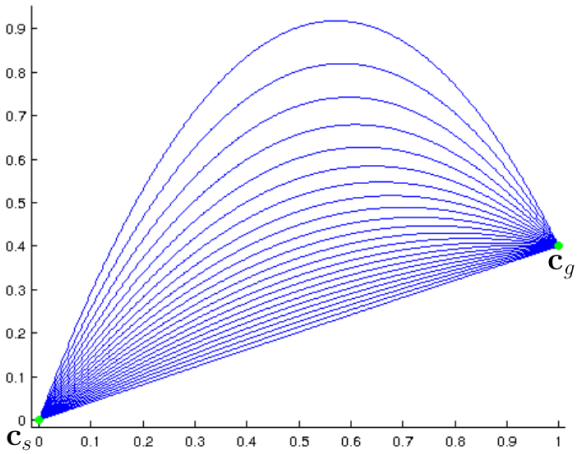


FIGURE 1.4 – Figure extraite de [Campana 2016b]. Faisceau de paraboles connectant $\mathbf{c}_s$ et $\mathbf{c}_g$ (points verts), pour différentes valeurs de α_s, dans le plan π_θ.

1.2.2.3 Contraintes de vitesse

Nous allons maintenant introduire des contraintes sur l'amplitude de la vitesse initiale et finale, afin de ne considérer que des trajectoires physiquement plausibles.

Ces contraintes sont exprimées de la manière suivante :

$$||\dot{\mathbf{c}}_s|| \leq V_{s_{max}} \text{ et } ||\dot{\mathbf{c}}_g|| \leq V_{g_{max}} \tag{1.17}$$

Pour la vitesse initiale, nous pouvons écrire :

$$\sqrt{\dot{x}_{\theta s}^2 + \dot{z}_s^2} \leq V_{s_{max}} \tag{1.18}$$

Ce qui est équivalent à :

$$1 + tan(\alpha_s)^2 \leq \frac{V_{s_{max}}^2}{\dot{x}_{\theta s}^2} \tag{1.19}$$

En utilisant les équations (1.14) nous pouvons alors obtenir :

$$g(x_{\theta g}-x_{\theta s})^2 tan(\alpha_s)^2 - 2(x_{\theta g}-x_{\theta s})V_{s_{max}}^2 tan(\alpha_s) + g(x_{\theta g}-x_{\theta s})^2 tan(\alpha_s)^2 + 2(z_g-z_s)V_{s_{max}}^2 \leq 0 \tag{1.20}$$

qui est une équation quadratique avec pour variable $tan(\alpha_s)$. Si cette équation n'admet pas de solution, alors il n'existe pas de parabole permettant de connecter $\mathbf{c}_s$ et $\mathbf{c}_g$ qui respecte les contraintes de vitesse initiale maximale. Sinon, les deux solutions à l'équation (1.20) nous donnent des bornes sur les valeurs possibles de α_s.

En appliquant un développement identique pour la vitesse finale et en intersectant les bornes trouvées, nous pouvons obtenir des bornes sur α_s qui permettent de produire des paraboles qui vérifient les contraintes de vitesse imposées sur la vitesse initiale et finale.

1.2.2.4 Tests de collision

Nous obtenons alors un faisceau de paraboles connectant exactement $\mathbf{c}_s$ et $\mathbf{c}_g$ et qui respectent les contraintes d'amplitude de vitesse initiale et finale maximales. Chacune des paraboles étant uniquement définie par la valeur de l'angle de tir initial α_s.

Nous allons maintenant vérifier les collisions pour ces paraboles en procédant par dichotomie jusqu'à en trouver une valide ou à avoir épuisé tout les candidats.

1.3 Planification pour robots à pattes

La catégorie des robots à pattes regroupe les robots humanoïdes, les quadrupèdes, hexapodes ou tout autre robot possédant des membres articulés qu'il utilise pour se déplacer en créant des contacts avec l'environnement.

La problème de la locomotion de ce type de robot sur du sol plat et via des cycles de contacts prédéfinis étant considéré comme résolue [Kajita 2003], la communauté

robotique s'intéresse maintenant à la planification de mouvements plus complexes, appelés **multi-contact** et décrits dans l'introduction de cette thèse.

Les techniques de planification habituelles présentées dans les sections ci-dessus ne sont pas utilisables directement pour générer des mouvements pour des robots à pattes pour plusieurs raisons.

Premièrement, ces robots ont un grand nombre de degrés de liberté ce qui augmente trop la dimension du problème de planification pour qu'il soit solvable dans des temps raisonnables. En effet, la complexité algorithmique des méthodes de planification probabilistes est exponentiel en le nombre de degrés de libertés du robot [LaValle 1999].

Deuxièmement, la probabilité d'échantillonner aléatoirement une configuration en contact avec l'environnement est nulle [Siméon 2004]. En effet, l'espace des configurations en contact avec l'environnement peut être défini comme la frontière de l'espace des configurations en collision $\mathcal{C}_{obs}$: $\partial\mathcal{C}_{obs}$. Or, la mesure de cet espace étant nulle, il faut donc recourir a des méthodes de projection pour obtenir une configuration dans cet espace [Bouyarmane 2009].

Enfin, dans le cas d'un mouvement de robot à pattes, le robot doit être capable de rester en équilibre en satisfaisant des contraintes dynamiques. La probabilité d'échantillonner aléatoirement ce genre de configurations et de trouver de tels mouvements via une méthode probabiliste tend vers 0.

Comme détaillé dans l'introduction de ce manuscrit, nous utilisons une approche découplée dans laquelle le premier sous-problème ($\mathcal{P}_1$) consiste à planifier une trajectoire qui servira de guide pendant la génération de contact ($\mathcal{P}_2$).

Il se pose alors le problème de la **faisabilité** : comment garantir que la trajectoire que l'on calcule durant ($\mathcal{P}_1$) sera un guide faisable pour la génération de contact ($\mathcal{P}_2$) ?

1.4 Planification d'une trajectoire guide

A notre connaissance [Escande 2008, Escande 2013] sont les premiers à utiliser un chemin guide pendant la planification de contacts, sous la forme d'une fonction de potentiel qui permet d'orienter la planification, définie comme un champ de potentiel autour d'un chemin (dans l'espace des configurations $\mathcal{C}$) donné. Ce chemin est défini préalablement par une interpolation entre plusieurs postures clés données par l'utilisateur.

D'autres approches ont ensuite formulé le problème de génération de contact et de génération de mouvement via des méthodes d'optimisations [Mordatch 2012, Winkler 2017, Posa 2016]. Comme toute méthode d'optimisation, elles requièrent une initialisation qui est soit fournie par l'utilisateur, soit calculée automatiquement par une approche naïve qui ne considère pas l'environnement. Une mauvaise

initialisation peut entraîner ces méthodes d'optimisation à rester coincées dans des minima locaux dû aux obstacles de l'environnement.

Pour éviter cela [Bouyarmane 2009, Bouyarmane 2012] propose une méthode pour planifier ce chemin guide automatiquement. La planification est faite dans l'espace $\mathcal{C}_{guide}$ définie comme le sous espace de $\mathcal{C}$ tel que toute configuration $\mathbf{q} \in \mathcal{C}_{guide}$ est suffisamment proche du contact et est en équilibre statique.

Comme la probabilité d'échantillonner aléatoirement dans $\mathcal{C}_{guide}$ reste très faible, les auteurs ont recourt à une méthode pour projeter une configuration de $\mathcal{C}_{free}$ dans $\mathcal{C}_{guide}$. Cependant, les méthodes de projection sont coûteuses en temps de calcul. De plus, bien que la méthode de planification de guide proposée dans [Bouyarmane 2009] fonctionne dans un sous espace de l'espace de configuration, la dimension du problème reste identique et est trop grande pour les algorithmes de planification probabiliste. Leurs résultats le prouvent en nécessitant d'une dizaine de minutes à plusieurs heures pour planifier un guide.

Enfin, le planificateur RB-RRT présenté dans [Tonneau 2015b, Tonneau 2018a] propose une approche heuristique pour planifier un chemin guide pour un modèle réduit du robot dans un espace de faible dimension, ce qui permet d'obtenir des temps de calcul interactifs. Les développements réalisés durant cette thèse se sont basés sur la formulation de RB-RRT, c'est pourquoi cette méthode est décrite succinctement dans la section suivante.

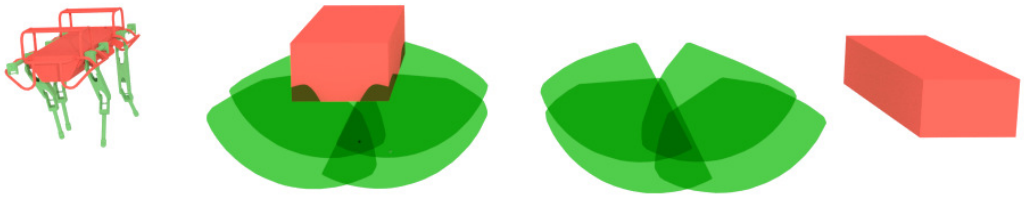
1.5 Reachability Based - RRT

Reachability Based RRT (RB-RRT) [Tonneau 2015b, Tonneau 2018a] est une méthode de planification basée sur l'algorithme de planification probabiliste BI-RRT et qui planifie un chemin guide pour un modèle simplifié du robot, réduisant ainsi la dimension de l'espace considéré pendant la planification.

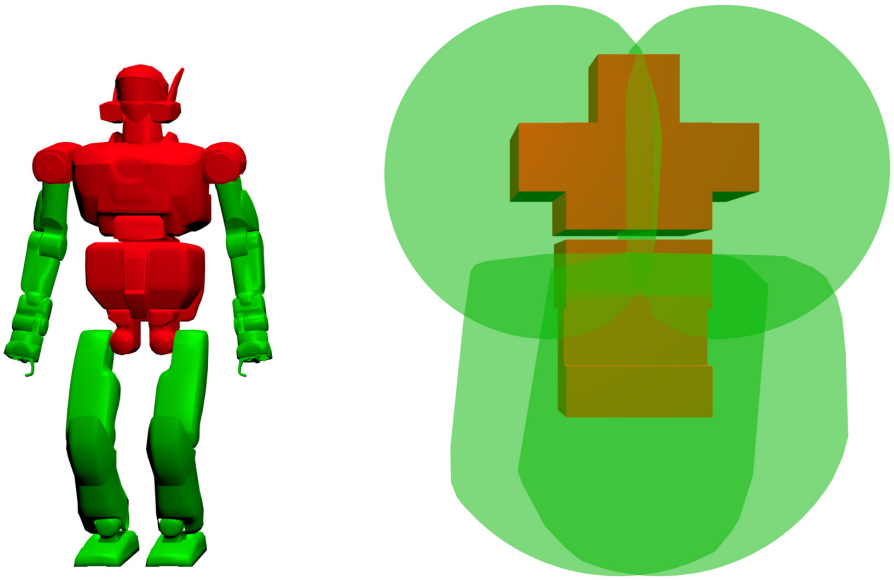
Cette méthode garantit néanmoins une forte probabilité d'existence d'une séquence de configurations en contact cinématiquement valides et sans collision le long du chemin guide calculé grâce à une approche heuristique. Cette heuristique est basée sur la notion d'accessibilité en utilisant l'idée suivante : pour pouvoir créer un contact avec l'environnement, l'origine du robot doit être placé suffisamment près d'un obstacle pour qu'il puisse le toucher avec une de ses pattes (pieds, mains, ...) mais pas trop près sinon il ne pourra pas éviter la collision avec l'obstacle.

Afin de déterminer si l'origine du robot est dans cet espace "près mais pas trop près" des obstacles, on utilise un modèle simplifié du robot qui est décomposé en deux types de formes géométriques, comme montré sur la figure 1.5. La forme rouge est une boite englobante de la géométrie du tronc du robot, les formes vertes sont les espaces atteignables de chaque membre du robot, ces formes vertes sont appelées "Range Of Motion" (ROM).

Le tronc du robot peut être articulé, comme c'est le cas par exemple pour le robot HRP-2 ou la majorité des robots humanoïdes. La forme rouge doit alors également être articulée avec les même degrés de liberté comme montré sur la figure 1.5-b.



(a) HyQ et sa représentation simplifiée



(b) HRP-2 et sa représentation simplifiée, les deux formes rouges du modèle simplifié sont reliés par deux liaisons pivots, selon les axes z et y, comme le torse du robot HRP-2.

FIGURE 1.5 – Le robot HyQ (a) et HRP-2 (b) et leurs représentations simplifiées utilisées pendant le problème $\mathcal{P}_1$. Les formes vertes représentent l'espace atteignable pour chaque patte du robot tandis que la forme rouge est la boite englobante du tronc du robot et doit rester sans collision.

Dans RB-RRT, une configuration du modèle simplifié est donc valide si les formes vertes sont en collision avec l'environnement (équivalent à la notion de "près") et si la forme rouge n'est pas en collision (équivalent à la notion de "pas trop près"). Des exemples de configurations sont montrés sur la figure 1.6, la configuration de gauche est invalide car les ROM ne sont pas en collision et la configuration du milieu est invalide car la forme englobante du tronc est en collision.

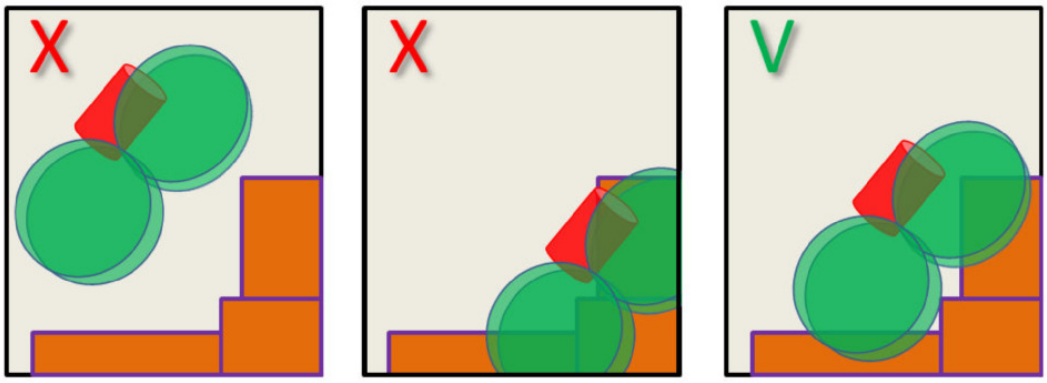


FIGURE 1.6 – Exemple de configurations possible pour le modèle réduit. Seule la configuration de droite respecte les conditions de validité définies par RB-RRT.

L'algorithme RB-RRT permet donc de grandement réduire la dimension du problème de planification en ne considérant qu'un problème de dimension $SE(3) \times \mathbb{R}^{n_{trunk}}$ puisqu'il ne planifie un chemin que pour la base mobile du modèle réduit du robot dans $SE(3)$ (la position et l'orientation 3D du centre du modèle réduit) et pour les n_{trunk} degrés de liberté du tronc du robot ($n_{trunk} = 2$ dans le cas de HRP-2).

Cependant deux limitations apparaissent dans cette méthode : premièrement, l'heuristique sur l'accessibilité utilisée est une condition nécessaire mais non suffisante pour la création de contact, c'est à dire qu'il n'y a pas de garantie qu'une configuration du modèle réduit valide permettra effectivement de créer cinématiquement des contacts avec le modèle complet du robot. Cette heuristique a tout de même était vérifiée empiriquement dans [Tonneau 2018a] et permet d'atteindre une très forte probabilité (entre plus de 99% pour les scénarios peu contraints et 85% dans les pires cas) de générer une configuration en contact cinématiquement valide pour le modèle complet.

Deuxièmement, l'heuristique est seulement une condition géométrique qui exprime donc seulement les contraintes cinématiques du modèle complet du robot. Elle n'offre aucune garantie sur l'existence d'une configuration en équilibre du modèle complet du robot. Il est montré empiriquement dans [Tonneau 2018a] que grâce au recours à d'autres heuristiques sur le type de surfaces de l'environnement considérées, on atteint une probabilité de 85% (dans le pire cas) de trouver une configuration du modèle complet en équilibre statique à partir d'une configuration valide du modèle réduit. Cette probabilité est élevée mais seulement parce que les scénarios considérés lors de ces tests ne nécessitent pas d'exploiter la dynamique du robot et sont tous solvables via un mouvement quasi-statique.

Enfin, la génération de configurations en contact est contrainte à vérifier l'équilibre **statique** du robot puisque la planification fournit un chemin guide qui ne donne aucune information sur la dynamique du système.

1.6 Conclusion

Il apparaît donc qu'aucune méthode existante ne permet de planifier un guide pour la génération de contact dans le cas d'un scénario multi-contact sans se limiter à des contraintes d'équilibre statique, avec des temps de calcul interactif et tout en garantissant la **faisabilité** de ce guide vis à vis de la génération de contact.

La solution qui nous semble la plus intéressante à développer afin de diminuer les limitations décrites dans le paragraphe précédent est la méthode RB-RRT. En effet, elle permet déjà d'obtenir des temps de calcul interactifs grâce à l'utilisation d'un modèle simplifié de faible dimension et garantit de forte probabilité de trouver une séquence de configurations en contact cinématiquement faisables et sans collision le long du chemin planifié.

Dans le domaine de la planification kinodynamique, la méthode d'extension locale DIMT est intéressante car elle permet de générer des trajectoires respectant des contraintes d'accélération et de vitesse. Cependant, cette méthode n'est pas utilisable directement dans l'algorithme RB-RRT puisque les contraintes dynamiques appliquées ne sont pas constantes et ne sont pas connues à priori.

En effet, contrairement aux problèmes de planification pour des bras manipulateurs présentés dans [Kunz 2014] où les bornes d'accélérations de chaque axe ne dépendent que du modèle du robot, dans le cas de la planification pour la locomotion de robots à pattes les bornes en accélération sont définies par les contraintes de non glissement appliquées aux points de contact. Comme présenté dans la section Contraintes dynamiques, ces contraintes de non glissement vont imposer des bornes sur l'accélération admissible. La difficulté dans ce cas est que les contraintes dynamiques dépendent de l'état courant du robot et de la phase de contact, ces bornes ne sont donc pas constantes mais sont dépendantes de l'état du robot.

La contribution développée dans le chapitre suivant est une extension **kinodynamique** de l'algorithme RB-RRT. Pour ce faire, nous avons proposé une formulation efficace permettant de calculer les bornes d'accélération admissibles en fonction de l'état et de la phase de contact du robot. Puis, nous avons proposé une méthode d'extension locale basée sur la méthode DIMT et utilisant ce calcul automatique des bornes d'accélérations admissibles afin de ne produire que des trajectoires respectant les contraintes dynamiques du robot.

Nous avons ensuite intégré cette méthode de planification locale au sein de l'algorithme RB-RRT afin d'en proposer une version kinodynamique capable de prendre en compte les contraintes dynamiques du robot liées aux conditions de non glissement.

Chapitre 2

Planification de trajectoires guides dynamiques pour la locomotion multi-contact

Sommaire

Dans ce chapitre nous allons présenter une version kinodynamique du planificateur RB-RRT permettant de planifier des trajectoires guides durant $\mathcal{P}_1$ dans l'espace d'état du modèle réduit, en garantissant de fortes probabilités qu'une séquence de configurations en contact, cinématiquement et dynamiquement valides et sans collision, existe le long de cette trajectoire.

Comme le résultat du sous-problème $\mathcal{P}_1$ sera alors une trajectoire, avec une paramétrisation temporelle et des informations sur la dynamique du modèle réduit du robot utilisé, la planification de contact ($\mathcal{P}_2$) n'est maintenant plus contrainte à produire des configurations en équilibre statique mais peut produire des configurations respectant les conditions de non glissement avec une accélération non nulle.

Afin de proposer cette version kinodynamique du planificateur, il faut tout d'abord une nouvelle méthode d'extension de la roadmap qui permet de relier les nœuds du graphe via des trajectoires qui respectent les contraintes cinématiques et

dynamiques des robots à pattes. La prise en compte des contraintes cinématiques est déjà faite grâce à l'heuristique basée sur l'accessibilité proposée par la méthode RB-RRT présentée dans la section 1.5. La contribution de ce chapitre porte donc essentiellement sur la prise en compte des contraintes dynamiques qui sont ignorées dans la méthode RB-RRT originale.

Une telle méthode d'extension locale est proposée dans la section suivante pour le cas général. Puis, une application de cette méthode dans le cadre du planificateur RB-RRT est proposée, nécessitant le recours à des approximations pour utiliser notre méthode d'extension avec le modèle réduit de RB-RRT.

Enfin, nous présentons des exemples de trajectoires et de séquences de contact produites et évaluons les performances des méthodes proposées.

2.1 Définition du problème

Nous définissons le problème de planification de trajectoire centroïdale comme le problème de trouver une trajectoire dynamiquement valide $\mathbf{x}(t)$ entre deux états $\mathbf{x}_{init}$ et $\mathbf{x}_{goal}$ pour un robot à pattes. Nous définissons la validité dynamique par le fait de respecter les contraintes de non glissement du robot (détaillées dans la section Contraintes dynamiques), y compris lorsque son accélération n'est pas nulle.

Il nous faut définir une nouvelle méthode de planification locale et de validation de trajectoires afin de respecter la dynamique du robot dans la méthodologie du RB-RRT.

Dans ce but, notre méthode de planification locale génère une trajectoire qui est dynamiquement valide dans le voisinage de l'état à partir duquel la roadmap est étendue. Puis, notre méthode de validation de trajectoire détermine précisément la partie de la trajectoire qui est réellement valide afin d'obtenir une roadmap où tous les états sont connectés par une trajectoire dynamiquement valide.

2.1.1 Utilisation du modèle centroïdal

Nous rappelons que la méthode de planification RB-RRT considère un modèle simplifié du robot et ne planifie que pour la base mobile de ce modèle dans $SE(3)$ et pour les n_{root} degrés de liberté du tronc du robot. Lors de la planification nous n'avons donc pas accès à la dynamique réelle du modèle complet du robot. Nous allons donc faire l'hypothèse que la masse des membres est négligeable par rapport à la masse totale du robot [Mordatch 2012] et considérer que la position du centre de masse du modèle complet du robot coïncide avec une position fixe dans le repère du modèle simplifié (par simplicité, nous prenons le centre géométrique du modèle simplifié).

Avec cette hypothèse, il est possible d'approximer la dynamique centroïdale du robot en ne considérant que le modèle simplifié pendant la planification. Ainsi, dans le reste des développements de ce chapitre nous ne considérons que le modèle centroïdal.

2.1.2 Notations

Nous rappelons que nous définissons un *état centroïdal* par $\mathbf{x} = (\mathbf{c}, \dot{\mathbf{c}}, \ddot{\mathbf{c}}) \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3$ un triplet décrivant la position, vitesse et accélération du centre de masse.

Il est possible d'associer une *phase de contact* $\{p\}$, $p \in \mathbb{N}$, à un état en la notant avec un exposant : $\mathbf{x}^{\{p\}}$.

Nous rappelons qu'une phase de contact est définie comme un ensemble de contacts actifs qui demeurent constants durant toute la phase de contact. Une phase de contact $\{p\}$ définit donc :

— $k^{\{p\}}$ le nombre de contact actifs ;
— $\mathbf{P}^{\{p\}} = [\mathbf{p}_1, \mathbf{p}_2, ...\mathbf{p}_k]$ l'ensemble des positions des points de contact $\mathbf{p}_i \in \mathbb{R}^3$;
— $\mathbf{N}^{\{p\}} = [\mathbf{n}_1, \mathbf{n}_2, ...\mathbf{n}_k]$ l'ensemble des normales aux contact $\mathbf{n}_i \in \mathbb{R}^3$;

2.2 Méthode d'extension pour robots à pattes

A chaque itération de l'algorithme de planification, une méthode de planification locale (aussi nommée *steering-method*) est appelée. Cette méthode de planification locale essaie de connecter exactement deux états via une trajectoire valide. Les variables d'entrée de cette méthode sont les deux états à connecter ainsi que les deux phases de contact associées $\mathbf{x}_0^{\{p\}}$ et $\mathbf{x}_1^{\{q\}}$. La sortie de la méthode est une trajectoire optimale en temps, de durée t_f, qui connecte exactement les deux états :

$$\mathbf{x} : t \in [0, t_f] \mapsto < \mathbf{c}(t), \dot{\mathbf{c}}(t), \ddot{\mathbf{c}}(t) >$$

Tel que $\mathbf{x}(0) = \mathbf{x}_0$ et $\mathbf{x}(t_f) = \mathbf{x}_1$, et $\mathbf{x}(t)$ est contrainte de respecter les bornes d'accélération du centre de masse imposées par les contraintes de non glissement à l'état $\mathbf{x}_0^{\{p\}}$.

La méthode de validation de trajectoire est ensuite appelée, avec en entrée la trajectoire générée par la méthode de planification locale. La sortie de cette méthode de validation est la partie de la trajectoire qui respecte les contraintes du problème.

Nous commençons par décrire la méthode de validation de trajectoire dans la sous-section suivante car sa formulation sert de base à la formulation de la méthode de planification locale, présentée à la fin de cette section.

2.2.1 Validation de trajectoire

2.2.1.1 Définition

La méthode de validation de trajectoire doit prendre en entrée deux états $\mathbf{x}_0^{\{p\}}$ et $\mathbf{x}_1^{\{q\}}$ et une trajectoire qui les connecte $\mathbf{x}(t)$. De plus nous considérons un ensemble discret, de taille inconnue, de phases de contact le long de la trajectoire, représenté

par $\mathbf{P}(t)$ et $\mathbf{N}(t)$, avec :

$$\mathbf{P}(0) = \mathbf{P}^{\{p\}}, \quad \mathbf{P}(t_f) = \mathbf{P}^{\{q\}}$$
$$\mathbf{N}(0) = \mathbf{N}^{\{p\}}, \quad \mathbf{N}(t_f) = \mathbf{N}^{\{q\}}$$

La définition de l'instant de transition entre les phases de contact, ainsi que l'existence possible de phases de contact intermédiaires, n'est pas traité ici. Dans la section 2.3 nous proposons une approche heuristique, propre à notre cas d'utilisation au sein de l'algorithme RB-RRT.

La sortie de la méthode de validation de trajectoire est la partie valide de la trajectoire initiale, notée $\mathbf{x}'(t)$ et tel que :

$$\mathbf{x}' : t \in \left[0, t'_f \leq t_f\right] \mapsto \mathbf{x}(t) \tag{2.1}$$

Dans le cas où $t'_f = 0$, la trajectoire est entièrement invalide. L'algorithme de planification la rejette et recommence une nouvelle itération. Dans le cas où $t'_f > 0$, une partie de la trajectoire est valide. L'algorithme de planification étend la roadmap et ajoute alors $\mathbf{x}'_1 = \mathbf{x}'(t'_f)$ au graphe et le connecte à $\mathbf{x}_0$ via la trajectoire $\mathbf{x}'(t)$.

Afin de vérifier la validité de la trajectoire nous procédons de manière classique en discrétisant la trajectoire et en vérifiant les contraintes à chaque pas de discrétisation. Ici, une trajectoire valide doit respecter les contraintes géométriques liées à l'heuristique sur l'accessibilité utilisée par RB-RRT ainsi que les contraintes dynamiques de non glissement. Les tests de collisions sont gérés par le projet Humanoid Path Planner (HPP) [Mirabel 2016] qui utilise la bibliothèque Flexible Collision Library (FCL) pour effectuer les tests d'intersections entre objets 3D. Dans la suite de cette section, nous nous intéresserons donc seulement à la vérification des contraintes dynamiques.

2.2.1.2 Contraintes dynamiques

En nous basant sur les travaux détaillés dans la section Contraintes dynamiques, nous formulons le test pour vérifier les contraintes dynamiques sous la forme d'un programme linéaire (LP). Nous rappelons que les contraintes de non-glissement appliquées au modèle centroïdale peuvent se formuler :

$$\exists \boldsymbol{\beta} \geq 0 \ \textbf{s.t.} \ \begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) + \dot{\mathbf{L}} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_3 & ... & \mathbf{I}_3 \\ \hat{\mathbf{p}_1} & ... & \hat{\mathbf{p}_k} \end{bmatrix} \mathbf{V}\boldsymbol{\beta} \tag{2.2}$$

Où :

— m est le poids total du robot ;
— k est le nombre total de point de contact ;
— $\mathbf{f} = \left[\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_k\right]^T \in \mathbb{R}^{3k}$ est le vecteur des forces de contact $\mathbf{f}_i$, appliquées aux points de contact $\mathbf{p}_i$;
— $\mathbf{g} = \begin{bmatrix} 0 & 0 & -9.81 \end{bmatrix}^T$ est le vecteur de gravité ;
— $\mathbf{L} \in \mathbb{R}^3$ est le moment cinétique (exprimé au centre de masse $\mathbf{c}$) ;

— $\hat{\mathbf{p}}_i$ sont les matrices de préproduit vectoriel associées aux positions de contact $\mathbf{p}_i$;
— $\mathbf{V} = diag(\begin{bmatrix}\mathbf{V}_1 & ... & \mathbf{V}_k\end{bmatrix}) \in \mathbb{R}^{3k \times 4k}$ avec les $\mathbf{V}_i$ les matrices définies par les génératrices des cônes de friction linéarisés ;
— $\boldsymbol{\beta} \in \mathbb{R}^{4k}$ un vecteur de coefficients.

Comme fait classiquement [Caron 2015, Del Prete 2016], nous posons le moment cinétique $\dot{\mathbf{L}} = 0$ afin de reformuler (2.2) comme suit :

$$\underbrace{m \begin{bmatrix}\mathbf{I}_3 \\ \hat{\mathbf{c}}\end{bmatrix}}_{\mathbf{H}} \ddot{\mathbf{c}} + \underbrace{m \begin{bmatrix}-\mathbf{g} \\ \mathbf{c} \times -\mathbf{g}\end{bmatrix}}_{\mathbf{h}} = \underbrace{\begin{bmatrix}\mathbf{I}_3 & ... & \mathbf{I}_3 \\ \hat{\mathbf{p}_1} & ... & \hat{\mathbf{p}_k}\end{bmatrix}}_{\mathbf{G}} \mathbf{V}\,\boldsymbol{\beta} \tag{2.3}$$

Ainsi, si il existe un $\boldsymbol{\beta}^*$ tel que $\boldsymbol{\beta}^* \geq \mathbf{0}$ et que l'équation (2.3) est satisfaite, alors le système respecte les contraintes de non glissement et donc respecte nos conditions de *validité dynamique*. Notre test revient donc à résoudre le LP suivant :

$$\begin{aligned} &\text{find } \boldsymbol{\beta} \\ &\text{s.t. } \mathbf{G}\boldsymbol{\beta} = \mathbf{H}\ddot{\mathbf{c}} + \mathbf{h} \\ &\quad \boldsymbol{\beta} \geq 0 \end{aligned} \tag{2.4}$$

2.2.2 Méthode de planification locale

Notre méthode de planification locale est basée sur la méthode décrite dans la sous-section 1.2.1, nommée Double Integrator Minimum Time (DIMT)[Kröger 2006, Hauser 2010, Kunz 2014]. Nous introduisons maintenant notre extension pour la locomotion des robots à jambes.

Nous rappelons que la méthode DIMT prend en entrée deux états $\mathbf{x}_0 = (\mathbf{c}_0, \dot{\mathbf{c}}_0, \ddot{\mathbf{c}}_0)$ et $\mathbf{x}_1 = (\mathbf{c}_1, \dot{\mathbf{c}}_1, \ddot{\mathbf{c}}_1)$ ainsi que des bornes d'accélération et de vitesse (constantes et symétriques) pour chaque degrés de liberté du système. Ici, nous considérons le modèle centroidal et nos degrés de liberté sont donc les trois translations selon les trois axes du repère cartésien. Ces bornes sont donc représentées ainsi :

$$\begin{aligned} -\dot{c}^{max}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq\ & \dot{c}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq\ \dot{c}^{max}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \\ -\ddot{c}^{max}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq\ & \ddot{c}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq\ \ddot{c}^{max}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \end{aligned} \tag{2.5}$$

Cette méthode produit en sortie une trajectoire $\mathbf{x}(t)$ définie par :

$$\mathbf{x} : t \in [0, t_f] \mapsto < \mathbf{c}(t), \dot{\mathbf{c}}(t), \ddot{\mathbf{c}}(t) >$$

Tel que $\mathbf{x}(0) = \mathbf{x}_0$ et $\mathbf{x}(t_f) = \mathbf{x}_1$, et $\mathbf{x}(t)$ respecte les bornes de vitesse et d'accélération données.

2.2.2.1 Problème de la locomotion de robots à pattes

Lors de la locomotion de robots à pattes, les contraintes dynamiques appliquées au centre de masse sont imposées par les contacts entre le robot et l'environnement. De ce fait, les bornes d'accélération du centre de masse ne sont ni constantes ni symétriques et elles ne peuvent pas être définies à priori sans connaissance des contacts actuels du robot. En effet, ces bornes vont correspondre aux accélérations admissibles du centre de masse. C'est à dire l'ensemble des accélérations qui permettent de satisfaire les contraintes de non glissement.

Or, l'accélération admissible dépend entre autre des points de contact et des normales de contact, la figure 2.1 montre un exemple intuitif : si l'on désire effectuer un mouvement vers la gauche du robot et que l'on a seulement des contacts entre les pieds et le sol, l'amplitude maximale de l'accélération qui permet de respecter les contraintes de non glissement est relativement faible. Si l'on rajoute un contact avec la main droite contre un mur, cette amplitude maximale augmente grandement.

D'autre part, comme décrit par l'équation (2.3), l'accélération admissible dépend également de la position du centre de masse comme on peut l'observer sur la figure 2.2.

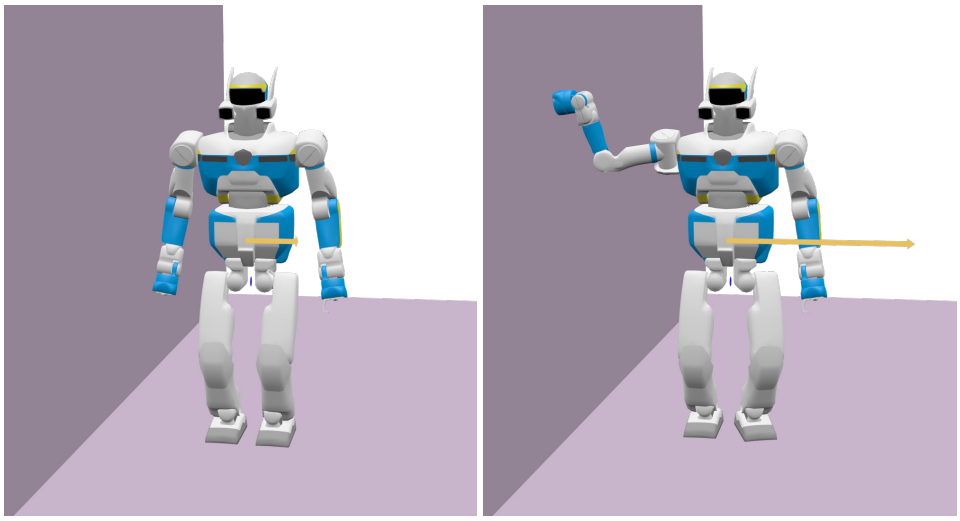


FIGURE 2.1 – Exemple de contraintes dynamiques dépendantes des contacts actifs pour HRP-2. Sur la figure de gauche, seuls les pieds sont en contact, sur la figure de droite la main droite est en contact avec le mur. Le vecteur jaune représente l'accélération admissible maximale vers la gauche du robot.

Afin d'être capable de gérer des bornes en accélération dépendantes de l'état et qui varient donc durant la planification, nous avons proposé de calculer ces bornes seulement pour l'état et la phase de contact initial de chaque appel à la méthode de planification locale : $\mathbf{x}_0^{\{p\}}$.

Ainsi, la trajectoire produite par la méthode de planification locale est valide dans le voisinage de l'état initial, et la méthode de validation de trajectoire présen-

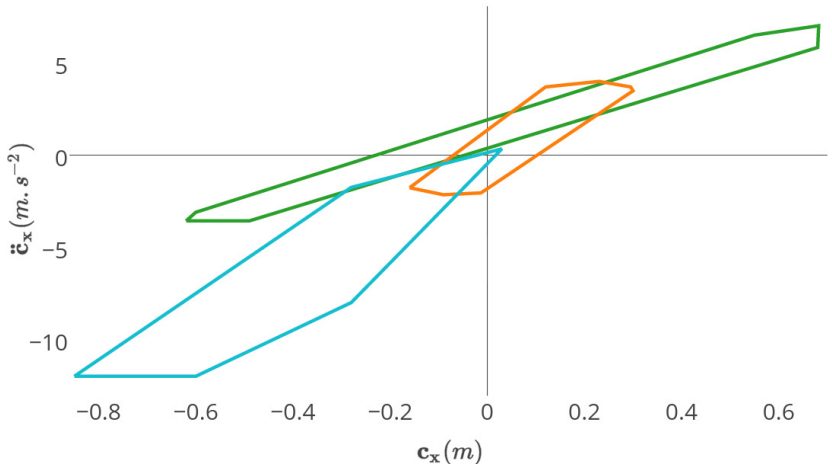


FIGURE 2.2 – Exemple de contraintes dynamiques dépendantes de l'état pour HRP-2. Chaque couleur correspond à une phase de contact différente (ie. différents points de contact avec des normales au contact différentes) pour une position de l'origine du robot identique. Les polygones représentent la relation entre l'accélération admissible du centre de masse et sa position, selon l'axe **x**.

tée dans la section précédente permet de déterminer exactement les limites de ce voisinage.

Bien que les bornes d'accélération utilisées ne soient valides que pour l'état initial de la planification locale, le fait d'utiliser ces bornes plausibles augmente grandement la probabilité que la méthode de planification locale produise des trajectoires valides par rapport à l'utilisation de bornes naïves constantes durant toute la planification, comme montré par les résultats de la section 2.4.2.3.

2.2.2.2 Calcul des bornes d'accélération

Plutôt que de calculer explicitement toutes les bornes d'accélération admissibles en utilisant une méthode de double description, nous préférons utiliser une méthode d'optimisation afin de calculer la borne dans une direction spécifique car cette approche se révèle moins coûteuse en temps de calcul [Del Prete 2016].

Pour estimer les bornes d'accélération admissibles dans le voisinage de l'état $\mathbf{x}_0^{\{p\}}$, nous allons procéder selon les étapes suivantes :

Premièrement, la méthode DIMT est appelée avec des bornes d'accélération arbitrairement grandes, définies manuellement :

$$-\ddot{c}^{\infty}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq \ddot{c}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq \ddot{c}^{\infty}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \tag{2.6}$$

Ces bornes permettent d'imposer une amplitude d'accélération maximale autorisée, mais ne prennent pas en compte les contraintes dynamiques.

La méthode DIMT génère alors une trajectoire composée d'une succession de

phases à accélération constante, nous nous intéressons à la direction de la première phase, notée par le vecteur unitaire $\mathbf{a}$.

Ensuite, nous calculons l'accélération maximale selon la direction trouvée qui satisfasse les contraintes de non glissement à l'état $\mathbf{x}_0^{\{p\}}$:

$$\ddot{\mathbf{c}}^{max} = \alpha^* \mathbf{a} \ , \ \alpha^* \in \mathbb{R}^+ \tag{2.7}$$

Enfin, la méthode DIMT est appelée une seconde fois, en utilisant comme bornes d'accélération la projection de $\ddot{\mathbf{c}}^{max}$ selon les axes $(\mathbf{x}, \mathbf{y}, \mathbf{z})$:

$$-\ddot{c}^{max}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq \ddot{c}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq \ddot{c}^{max}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \tag{2.8}$$

Avec $\ddot{c}^{max}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq \ddot{c}^{\infty}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}}$.

Si la direction de l'accélération $\mathbf{a}'$ calculée par le second appel de DIMT est telle que $\mathbf{a}' = \mathbf{a}$, alors les bornes utilisées sont correctes et la trajectoire produite sera dynamiquement valide dans le voisinage de $\mathbf{x}_0$. Dans la section 2.4.2.1 nous montrons empiriquement que dans la majorité des cas nous obtenons bien l'égalité $\mathbf{a}' = \mathbf{a}$ et que dans tout les cas les bornes restent valides dans le voisinage de $\mathbf{x}_0$.

Afin de calculer α^*, nous reformulons le LP (2.4) utilisé par notre test des contraintes dynamiques présenté dans la section précédente. La première étape étant de réécrire l'équation (2.3) en posant $\ddot{\mathbf{c}} = \alpha \mathbf{a}$:

$$\underbrace{m \begin{bmatrix} \mathbf{I}_3 \\ \hat{\mathbf{c}} \end{bmatrix}}_{\mathbf{H}} \boldsymbol{\alpha}\mathbf{a} + \underbrace{m \begin{bmatrix} -\mathbf{g} \\ \mathbf{c} \times -\mathbf{g} \end{bmatrix}}_{\mathbf{h}} = \underbrace{\begin{bmatrix} \mathbf{I}_3 & \dots & \mathbf{I}_3 \\ \hat{\mathbf{p}_1} & \dots & \hat{\mathbf{p}_k} \end{bmatrix}}_{\mathbf{G}} \mathbf{V}\,\boldsymbol{\beta} \tag{2.9}$$

Nous obtenons ensuite l'équation suivante, définissant les contraintes de non glissement :

$$\mathbf{H}\alpha\mathbf{a} + \mathbf{h} = \mathbf{G}\boldsymbol{\beta} \tag{2.10}$$

Dans cette équation, le membre de droite $\mathbf{G}$ dépend uniquement de la phase de contact courante (ie. des contacts courants entre le robot et l'environnement) et le membre de gauche dépend de la position du centre de masse du robot. Ici, nous calculons ces membres en fonction de l'état et de la phase de contact initial $\mathbf{x}_0^{\{p\}}$.

Comme pour le test des contraintes dynamiques, nous avons $\boldsymbol{\beta} \in \mathbb{R}^{4k}$, $\boldsymbol{\beta} \geq 0$ comme variable de notre problème. Mais ici, nous introduisons une nouvelle variable $\alpha \in \mathbb{R}^+$ qui représente l'amplitude de l'accélération. L'égalité ci-dessus peut alors être réécrite sous la forme suivante :

$$\begin{bmatrix} \mathbf{G} & -(\mathbf{H}\mathbf{a}) \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta} \\ \alpha \end{bmatrix} = \mathbf{h} \tag{2.11}$$

Nous pouvons maintenant poser le problème LP suivant :

$$\begin{aligned} &\text{find} \quad \begin{bmatrix} \boldsymbol{\beta} \\ \alpha \end{bmatrix} \\ &\min \quad -\alpha \\ &\text{s. t.} \quad \begin{bmatrix} \mathbf{G} & -(\mathbf{Ha}) \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta} \\ \alpha \end{bmatrix} = \mathbf{h} \\ &\qquad \begin{bmatrix} \boldsymbol{\beta} \\ \alpha \end{bmatrix} \geq \mathbf{0} \end{aligned} \tag{2.12}$$

Si le LP (2.12) admet une solution, alors la position du centre de masse dans l'état initial, $\mathbf{c}_0$, permet de satisfaire les contraintes dynamiques imposées par la phase de contact $\{p\}$ et la valeur optimale α^* donne la valeur maximale de l'amplitude de l'accélération selon la direction $\mathbf{a}$ qui permet de satisfaire ces contraintes : $\ddot{\mathbf{c}}^{max} = \mathbf{a}\alpha^*$.

Nous pouvons maintenant appeler la méthode DIMT entre les deux états $\mathbf{x}_0$ et $\mathbf{x}_1$, avec ces bornes d'accélération calculées.

Nous avons donc développé dans cette section un test permettant de vérifier si un état du robot est *dynamiquement valide* (c'est à dire qu'il respecte les contraintes de non glissement) et une méthode permettant de calculer l'accélération maximale admissible dans une direction donnée en fonction d'un état du robot. Ces deux méthodes étant formulées comme des problèmes LP, nous pouvons utiliser des méthodes de résolution du commerce afin de les résoudre efficacement.

Grâce à ces méthodes nous avons écrit une méthode de planification locale qui génère une trajectoire optimale en temps, connectant exactement deux états $\mathbf{x}_0^{\{p\}}$ et $\mathbf{x}_1^{\{q\}}$, dynamiquement valide dans le voisinage de $\mathbf{x}_0^{\{p\}}$. La partie effectivement valide de cette trajectoire est ensuite déterminée par la méthode de validation de trajectoire.

2.3 Application à la planification de guide avec RB-RRT

Nous rappelons que la méthode présentée en introduction découple le problème de planification de contact en deux phases : tout d'abord, la planification d'un chemin guide ($\mathcal{P}_1$) grâce à la méthode RB-RRT utilisant un modèle simplifié du robot (Fig. 1.5) présentée dans la section 1.5. Ensuite, la génération d'une séquence discrète de configurations corps complet en équilibre statique qui suivent le chemin trouvé précédemment ($\mathcal{P}_2$).

Grâce à l'intégration des méthodes présentées dans la section précédente au sein de RB-RRT, nous transformons la problème de planification de chemin $\mathcal{P}_1$ en un problème de planification de trajectoire. Ce qui nous permet alors de supprimer la contrainte qui imposait les configurations produites par $\mathcal{P}_2$ d'être en

équilibre statique. Nous obtenons ainsi une version **kinodynamique** de RB-RRT [Fernbach 2017].

Cela nous permet de nous intéresser à résoudre d'autres types de problèmes, qui n'auraient pas de solution quasi-statique. Par exemple un mouvement incluant un saut, ou de devoir descendre une pente qui n'est pas *presque plate* (c'est à dire dont la normale est orientée telle que le cône de friction ne contient pas la direction opposée à la gravité).

De plus, grâce à l'utilisation de ces méthodes nous pouvons désormais garantir de forte probabilité que des configurations en équilibre existent le long de la trajectoire planifiée.

Dimensions et notations de la trajectoire guide

Dans le cas géométrique, RB-RRT planifie une trajectoire pour la base mobile du modèle simplifié dans $SE(3)$ (la position et l'orientation 3D du centre du modèle réduit) et pour les n_{trunk} degrés de liberté du tronc du robot. La trajectoire est donc représenté par $\mathbf{q}^{trunk}(t)$ avec $\mathbf{q}^{trunk} \in SE(3) \times \mathbb{R}^{n_{root}}$ une configuration du modèle réduit.

Dans le cas kinodynamique, nous rajoutons les variables d'état $\dot{\mathbf{c}} \in \mathbb{R}^3$ et $\ddot{\mathbf{c}} \in \mathbb{R}^3$ respectivement la vitesse et l'accélération du centre de masse du modèle réduit. Or, la position du centre de masse du modèle réduit $\mathbf{c}$ est uniquement définie par $\mathbf{q}^{trunk}$, nous avons donc bien une trajectoire centroïdale sous sa forme classique $\mathbf{x}(t) =< \mathbf{c}, \dot{\mathbf{c}}, \ddot{\mathbf{c}} >$.

Au final, la trajectoire guide est notée par :

$$< \mathbf{x}(t) =< \mathbf{c}, \dot{\mathbf{c}}, \ddot{\mathbf{c}} >, \mathbf{q}^{trunk}(t) > \tag{2.13}$$

Lors de la planification locale, la trajectoire centroïdale $\mathbf{x}(t) =< \mathbf{c}, \dot{\mathbf{c}}, \ddot{\mathbf{c}} >$ est générée via la méthode de planification locale présentée dans la section précédente tandis que les autres degrés de libertés $\mathbf{q}^{trunk}$ sont gérés comme dans la méthode RB-RRT géométrique : une interpolation dans $SO(3)$ pour l'orientation de la base mobile et une interpolation linéaire pour les degrés de liberté du tronc du robot.

2.3.1 Planification de trajectoire pendant $\mathcal{P}_1$

2.3.1.1 Intégration de la méthode de planification locale

L'efficacité de la méthode RB-RRT s'explique en partie par le découplage entre la planification de trajectoire guide et la génération de contact. En effet, durant $\mathcal{P}_1$ les contacts ne sont pas générés explicitement afin d'éviter la complexité combinatoire que cela impliquerait.

Cependant, la méthode de planification locale présentée dans la section précédente nécessite de connaître les contacts actuels entre le robot et l'environnement, puisqu'elle prend en entrée des états associés à des phases de contact $\mathbf{x}^{\{p\}}$ qui définissent donc $\mathbf{P}^{\{p\}}$ et $\mathbf{N}^{\{p\}}$, respectivement la position des points de contact et

la normale des surfaces de contact pour la phase de contact $\{p\}$. Afin de pouvoir utiliser notre méthode d'extension locale au sein de RB-RRT nous allons donc approximer ces contacts durant $\mathcal{P}_1$ comme suit :

Pour une position du centre du modèle réduit $\mathbf{c}$ donnée, l'intersection entre les espaces atteignables de chaque membre et l'environnement est calculée, et l'on suppose que le contact existe au centre de cette intersection, comme présenté sur la figure 2.3. Dans le cas où l'espace atteignable d'un membre intersecte l'environnement en plusieurs surfaces disjointes, la position du contact est approximée au centre de la surface la plus proche de la configuration de référence de ce membre.

Une autre approximation est faite pour la position du centre de masse, qui est placée au centre géométrique du modèle simplifié. Cette dernière approximation est équivalente à supposer que les membres du robot ont une masse nulle [Mordatch 2012].

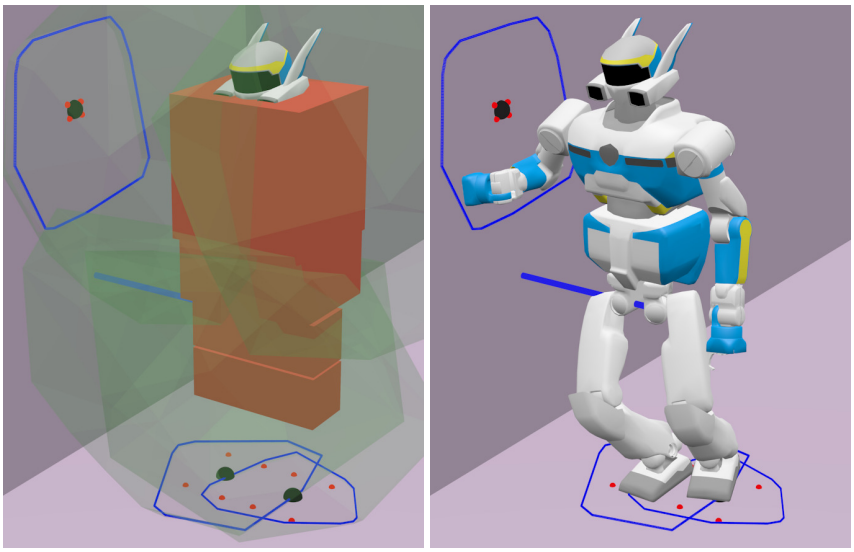


FIGURE 2.3 – Approximation des points de contact utilisée pendant $\mathcal{P}_1$. Gauche : les polygones bleus représentent les intersections entre l'espace atteignable d'un membre et l'environnement. Le centre de cette intersection (sphères noires) est la position du centre du pied ou de la main estimée durant $\mathcal{P}_1$, à partir de laquelle les points de contact (sphères rouges) sont extrapolés. Droite : configuration corps complet correspondante générée pendant $\mathcal{P}_2$.

2.3.1.2 Validation de trajectoires pendant $\mathcal{P}_1$

De la même manière, la méthode de validation de trajectoire prend en entrée un ensemble de phases de contact représenté par $\mathbf{P}(t)$ et $\mathbf{N}(t)$ et nous devons donc les approximer. Pour ce faire, nous calculons $\mathbf{P}(0)$ et $\mathbf{N}(0)$ via l'approximation proposée dans la sous-section précédente. Puis nous supposons que tant que la même surface de l'environnement intersecte l'espace atteignable des membres, les points de contact glissent en suivant la trajectoire du centre du modèle réduit $\mathbf{c}(t)$. Ceci revient à dire

que la position des points de contact dans le référentiel local du robot est constante.

Lorsque la surface sur laquelle la position des points de contact avait été estimée à $t = 0$ n'est plus dans l'espace atteignable d'un membre, on approxime de nouveaux points de contact avec la même méthode. Au final, $\mathbf{N}(t)$ est constant par morceaux et $\mathbf{P}(t)$ est différentiable par morceaux.

2.3.2 Génération de saut

Dans cette sous-section, nous montrons un cas d'utilisation particulier de notre planificateur : la génération de trajectoires comprenant des sauts.

Afin de pouvoir planifier une trajectoire comprenant un saut, il faut tout d'abord pouvoir générer une trajectoire balistique connectant exactement deux positions, pour cela nous nous basons sur les travaux de [Campana 2016b] résumés dans la sous-section 1.2.2.

Ici, nous nous intéressons à planifier une trajectoire qui ne comprend des sauts que lorsque cela est nécessaire, et nous montrons comment intégrer de telles trajectoires lors de la planification, tout en assurant la validité dynamique du résultat.

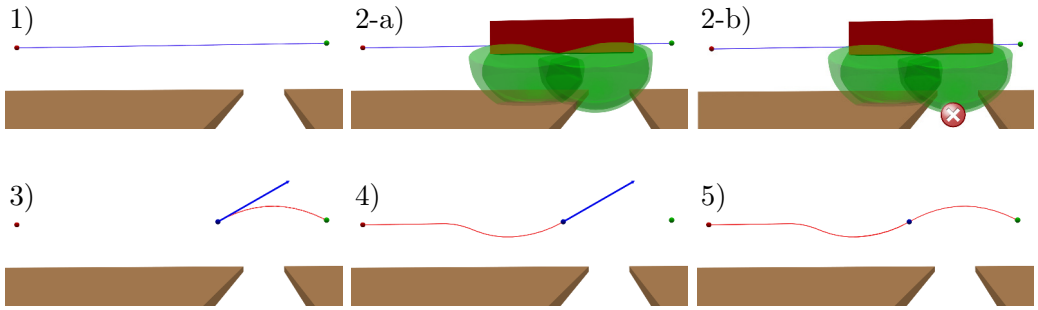


FIGURE 2.4 – Un saut est composé d'une phase balistique et d'une phase de préparation, planifié séquentiellement.

Notre méthode de planification locale peut produire des trajectoires $\mathbf{x}(t)$ entre deux états $\mathbf{x}_0$ et $\mathbf{x}_1$ qui résultent en des phases où le nombre de contacts possibles passe en dessous d'un seuil défini par l'utilisateur (Fig 2.4 - 1), c'est à dire que l'espace atteignable de certains membres n'est plus en collision avec l'environnement. Lorsque ce cas est détecté pendant la validation de trajectoire, le planificateur essaie de générer un saut comme suit :

Tout d'abord, il identifie le dernier état valide où suffisamment de zones atteignables sont encore en collision avec l'environnement $\mathbf{x}(t_{to})$ (Fig 2.4 2-a). La figure 2.4 2-b montre le premier état invalide : l'espace atteignable des deux pattes avant ne sont plus en collision, aucun contacts n'est donc possible pour ces pattes.

Ensuite, en utilisant la méthode présentée dans la sous-section 1.2.2, une trajectoire balistique est générée entre $\mathbf{c}_{to}$ et $\mathbf{c}_1$ en calculant la vitesse de décollage nécessaire en t_{to} : $\dot{\mathbf{c}}_{to}$ (Fig 2.4 3, le vecteur bleu est la vitesse de décollage requise).

Finalement, si une telle trajectoire balistique existe et est sans collision, alors notre méthode de planification locale est appelée pour connecter $\mathbf{x}_0$ à $\mathbf{x}_{to} = (\mathbf{c}_{to}, \dot{\mathbf{c}}_{to}, 0)$ avec une vitesse finale égale à la vitesse de décollage requise calculée précédemment (Fig 2.4 4).

Si la méthode de planification locale trouve une telle trajectoire et qu'elle est valide, alors une trajectoire comprenant une phase de prise d'élan et de préparation suivie d'une phase de vol a été générée avec succès.

2.3.3 Génération de configurations en contacts en équilibre dynamique

Durant la phase de planification de contact $\mathcal{P}_2$, une séquence discrète de configurations corps complet est générée le long du chemin guide trouvé durant $\mathcal{P}_1$. La description détaillée de la méthode de planification de contact est faite dans le chapitre 6.

Grâce à la version kinodynamique du RB-RRT, le résultat de $\mathcal{P}_1$ est maintenant une trajectoire paramétrisée temporellement $\mathbf{x}(t) =< \mathbf{c}(t), \dot{\mathbf{c}}(t), \ddot{\mathbf{c}}(t) >$ et nous avons donc des informations sur la dynamique du système lors de la génération de contacts pendant $\mathcal{P}_2$. Nous pouvons maintenant générer des configurations en *équilibre dynamique*, c'est à dire qui respectent les contraintes de non glissement avec une accélération pouvant être non nulle, en utilisant le même test que pour la validation de trajectoire (2.4).

La sortie du sous problème $\mathcal{P}_2$ est donc maintenant une séquence de configuration en équilibre dynamique, qui suivent la trajectoire guide planifiée pendant $\mathcal{P}_1$ et qui respectent les conditions de non glissement selon la dynamique du robot déterminée pendant $\mathcal{P}_1$.

2.4 Résultats

2.4.1 Scénarios

Nous avons évalué la version kinodynamique du planificateur RB-RRT sur divers scénarios avec les robots HyQ ou HRP-2. Chaque scénario met en avant une propriété particulière de notre planificateur kinodynamique. Dans l'ensemble des figures suivantes, les flèches bleues et jaunes représentent respectivement la vitesse et l'accélération du centre du robot et leur longueur est proportionnelle à la norme de ces vecteurs.

2.4.1.1 Évitement d'obstacles

Nous avons tout d'abord comparé les résultats produits entre la version originale de RB-RRT (géométrique) et notre version kinodynamique sur des scénarios dans des environnements contraints, nécessitant de slalomer entre les obstacles.

La figure 2.5 montre la différence entre les deux planificateurs. Comme la version géométrique n'a aucune contrainte sur l'accélération du centre du robot, elle peut produire des chemins avec un changement de direction instantané, qui ne semble pas naturel. De plus, la méthode de planification locale utilisée dans le cas géométrique est une interpolation linéaire, les chemins produits par cette méthode sont donc constitués de lignes brisées.

Notre version kinodynamique respecte des contraintes d'accélération et produit des trajectoires avec une accélération constante par morceaux et avec une vitesse continue. Ces trajectoires semblent bien plus naturelles et plus lisses.

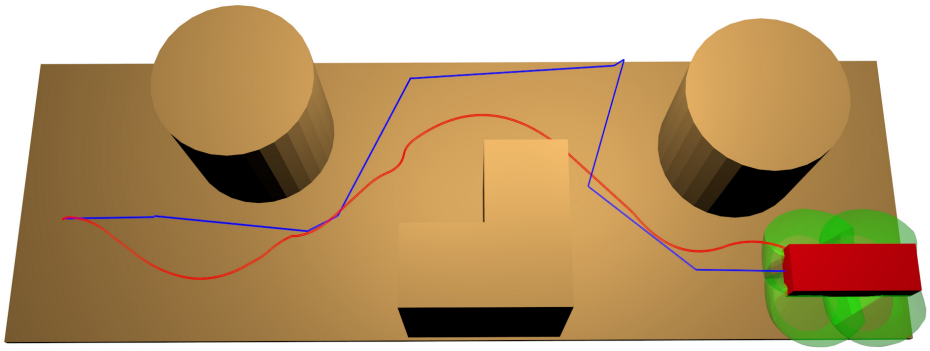


FIGURE 2.5 – Scénario *Slalom* : les trajectoires trouvées par la version kinodynamique (en rouge) sont plus lisses et apparaissent plus naturelles que celles trouvées par la version géométrique (en bleu).

Un second scénario présenté par la figure 2.6 montre un cas où une méthode de paramétrisation temporelle utilisée sur un chemin géométrique ne pourrait pas réussir à résoudre le problème. En effet, la position finale désirée se trouve sur la plate forme et nécessite un saut pour la rejoindre. Un point de passage a été placé manuellement avant le saut, mais afin d'être en mesure d'effectuer ce saut il faut atteindre le point de passage avec une vitesse suffisante.

Un planificateur géométrique va trouver le chemin le plus court pour rejoindre le point de passage. Mais une méthode d'optimisation qui essaierait de transformer ce chemin en une trajectoire ne pourra pas sortir du minimum local créé par l'obstacle et ne trouvera pas de solution qui permette d'atteindre le point de passage avec une vitesse suffisante.

Notre planificateur kinodynamique va trouver une trajectoire plus longue mais qui permet d'atteindre le point de passage à la vitesse désirée pour effectuer le saut.

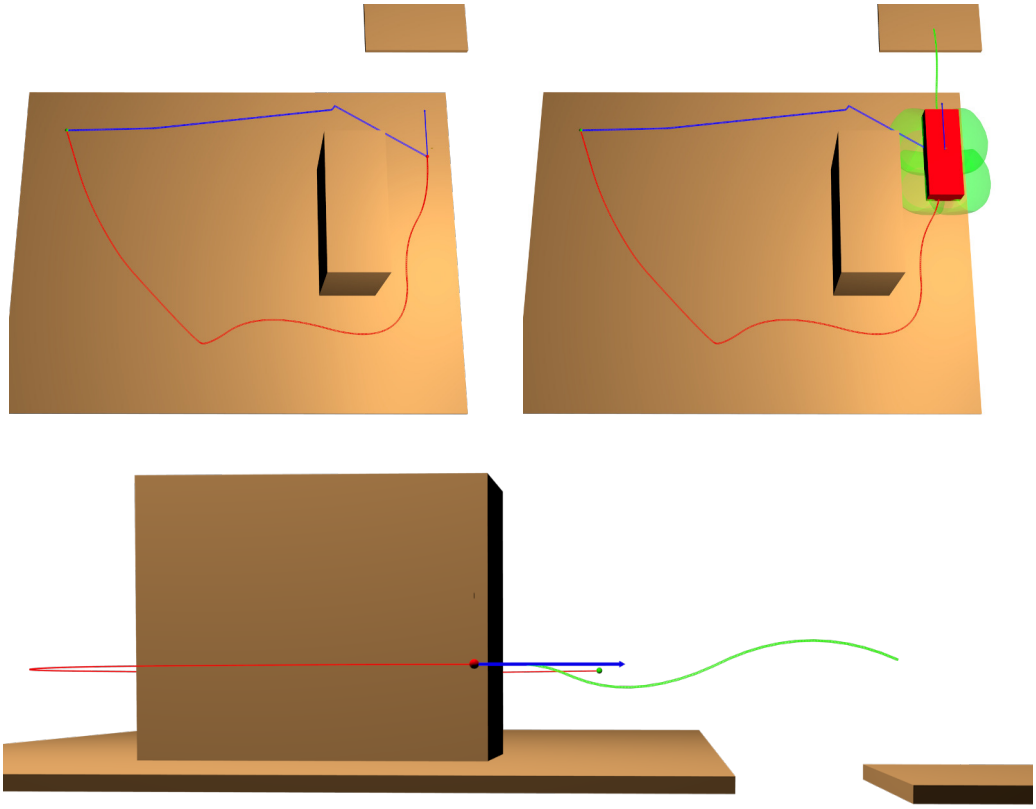


FIGURE 2.6 – Scénario *plate forme* : la position finale est sur la plate forme en haut à droite. Un point de passage (sphère rouge) est placé avant le saut, avec une vitesse non nulle (vecteur bleu). En bleu la trajectoire trouvée par la planificateur géométrique, en rouge celle trouvée par le planificateur kinodynamique. En vert la trajectoire entre le point de passage et la position finale (trouvée par le planificateur kinodynamique). Sur la figure en haut à droite, le robot est placé au point de passage.

Le seconde partie de la trajectoire entre le point de passage et l'état final (en vert sur la figure 2.6) est également planifiée via la version kinodynamique de RB-RRT qui permet de planifier la prise d'élan et le saut.

2.4.1.2 Push recovery

Dans ce scénario l'état initial de HRP-2 est à 1.5 mètres d'un mur, avec une vitesse initiale de $1.5 m.s^{-1}$ en direction du mur. La figure 2.7 montre que si l'on ne considère que les contacts entre les pieds et le sol, l'accélération maximale que l'on peut exercer vers la direction opposée au mur est trop faible pour pouvoir s'arrêter et changer de direction avant la collision.

En revanche, si l'on considère également un contact entre la main du robot et le

mur, l'accélération maximale admissible est bien plus important et permet d'éviter la collision. La séquence de contacts correspondante est montrée sur la figure 2.8.

Ce scénario montre bien l'intérêt de notre planificateur à calculer automatiquement les bornes d'accélération admissible en fonction des contacts possibles entre le robot et l'environnement.

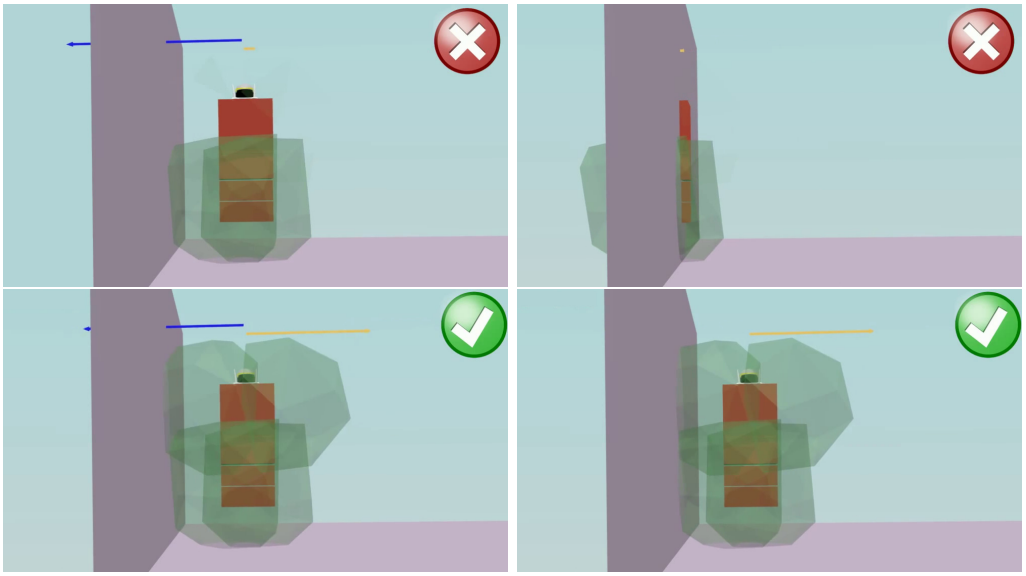


FIGURE 2.7 – HRP-2 est poussé vers le mur à une vitesse de $1.5m.s^{-1}$. La ligne du haut montre le cas où seuls les contacts avec les pieds sont autorisés, la ligne du bas montre le cas où on autorise les contacts avec les mains. La colonne de gauche montre l'état initial et la colonne de droite montre le premier état qui atteint une vitesse nulle. Les flèches bleues représentent la vitesse, les flèches jaunes représentent l'accélération.

2.4.1.3 Scénarios hautement dynamiques

Dans le scénario présenté par la figure 2.9 HRP-2 doit descendre une pente inclinée de 35°. Cette inclinaison est telle que la direction opposée à la gravité n'est pas comprise dans les cônes de friction entre les pieds et le sol. De ce fait, il n'existe aucune solution pour obtenir une configuration en équilibre statique dans la pente.

Le planificateur kinodynamique trouve une trajectoire solution qui consiste à accélérer dans la direction de la pente tant que le robot est sur celle-ci et à ralentir seulement une fois que le sol plat est atteint. En effet, la seule manière de satisfaire la condition de non glissement lorsque la direction opposée à la gravité n'est pas comprise dans le cône de friction est d'exercer une accélération telle que la résultante des forces de contact est ramenée dans le cône de friction. Dans ce cas, il faut accélérer dans la direction de la pente.

C'est également le comportement intuitif utilisé par les humains (ou d'autres animaux) lorsque nous sommes confrontés à une pente trop raide pour rester debout

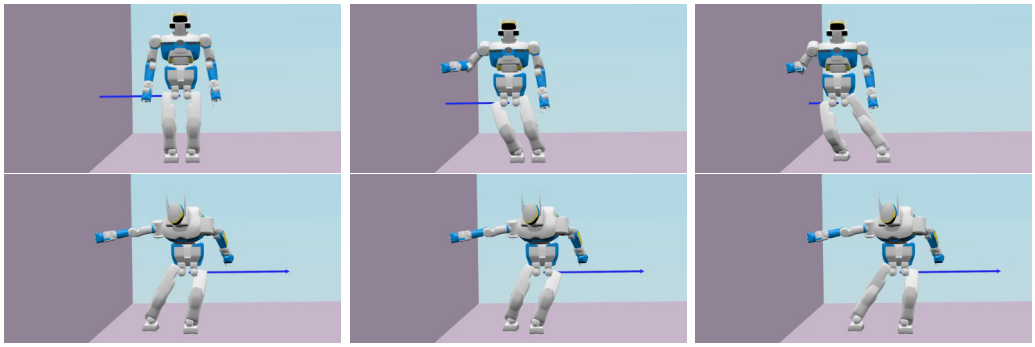


FIGURE 2.8 – Séquence de contacts générée pour le scénario du mur. Sur les vignettes 2-3-4 la main et les deux pieds sont en contact, sur les autres seul les pieds sont en contact.

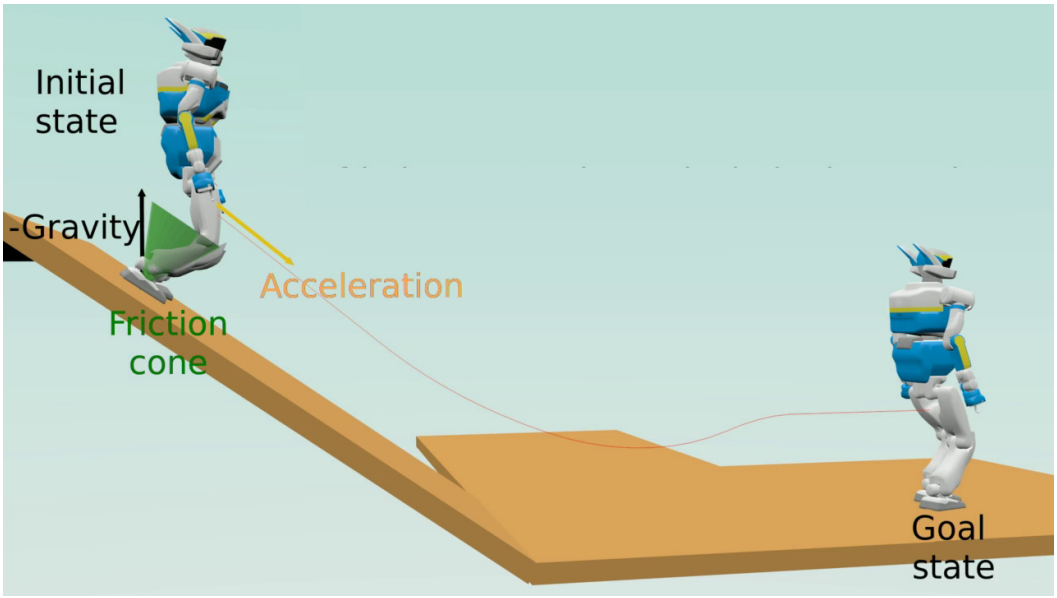


FIGURE 2.9 – Scénario *Pente* : HRP-2 doit descendre une pente à 38°. La direction opposée à la gravité n'est pas comprise dans les cônes de friction, de ce fait aucune configuration ne peut être en équilibre statique sur la pente. En rouge la trajectoire trouvée par le planificateur kinodynamique.

sans glisser : se laisser "entraîner" par la pente en continuant d'accélérer jusqu'à atteindre un sol suffisamment plat pour freiner.

La figure 2.10 montre la séquence de contacts générée le long de la trajectoire trouvée. On a bien une accélération vers le haut (vignette 7) ou vers l'arrière (vignette 11) seulement à partir de l'instant où le robot a au moins un pied sur le sol plat.

Il faut noter que, comme précisé dans la section 2.3.1, la position des points de contact n'est pas explicitée durant la phase $\mathcal{P}_1$ de RB-RRT mais qu'elle est

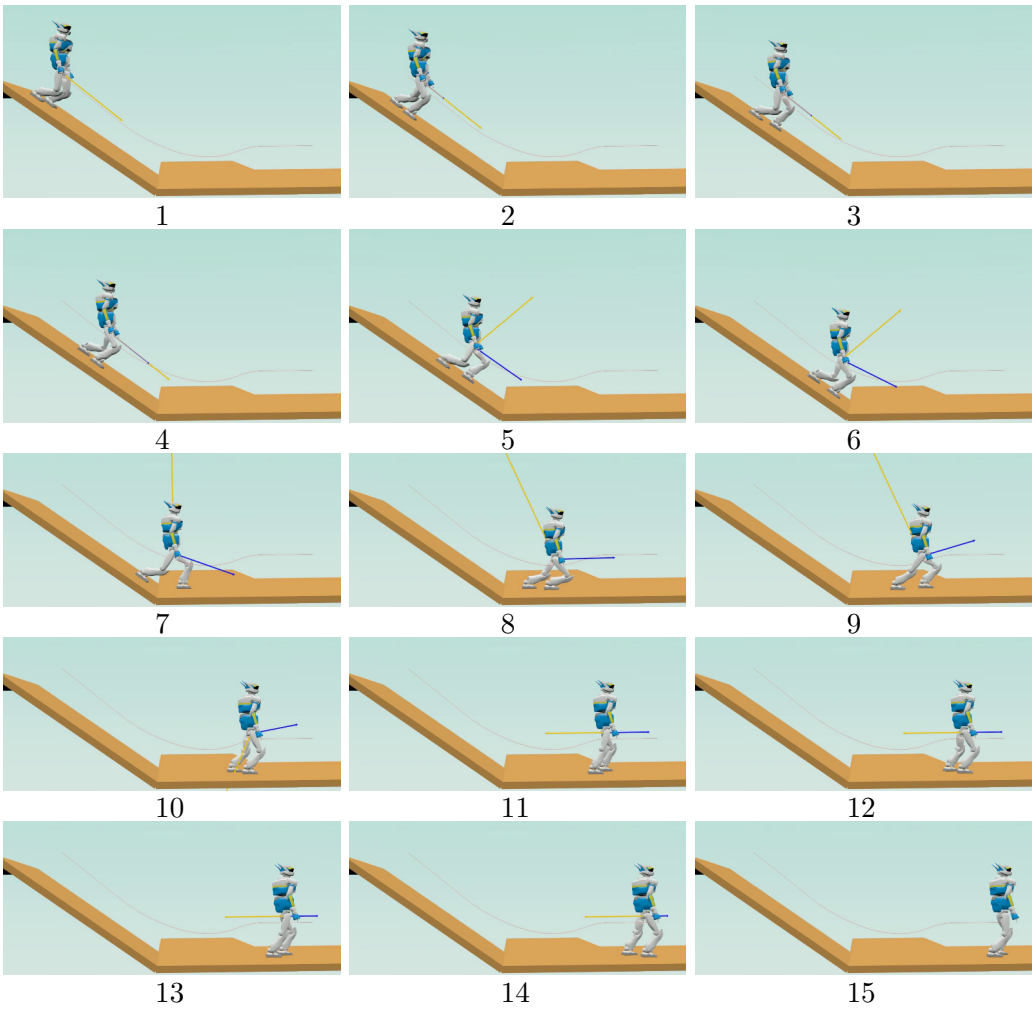


FIGURE 2.10 – Séquence de contacts générée pour le scénario de la pente.

approximée via une heuristique naïve pour les besoins des méthodes de planification locale et de validation de trajectoire.

Ce que signifie que nous n'avons aucune garantie que la position des points de contact approximée durant $\mathcal{P}_1$ soit la même (ou même qu'elle soit proche) de la position des contacts générés durant $\mathcal{P}_2$.

Mais, même dans un cas comme celui-ci où la position exacte des points de contact peut se révéler critique lors du passage entre le plan incliné et le sol plat (par exemple sur la vignette 7 de la figure 2.10), changeant drastiquement les contraintes dynamiques et donc l'accélération admissible, notre approximation reste correcte.

En effet, si à un instant de la trajectoire trouvée en $\mathcal{P}_1$ le robot exerce une accélération vers le haut, c'est qu'à cet instant la position de l'origine du robot était telle que l'espace atteignable d'au moins un pied intersectait avec le sol plat. Mais lors de la génération de configuration en contact pendant $\mathcal{P}_2$, pour cette même position de l'origine du robot, il existe peut être des solutions cinématiquement

valides telles que les pieds sont en contact avec le plan incliné et non avec le sol plat.

Cependant, comme les configurations générées à l'étape $\mathcal{P}_2$ sont contraintes à respecter les conditions de non glissement et à suivre l'accélération trouvée durant $\mathcal{P}_1$, seules les configurations avec au moins un pied sur le sol plat seront validées pour cet exemple puisque les autres configurations ne seraient pas dynamiquement valides.

Cela signifie que la dynamique calculée durant $\mathcal{P}_1$ sera bien consistante avec la séquence de contacts générée durant $\mathcal{P}_2$ malgré les approximations faites sur la position des points de contact.

2.4.2 Évaluation de performances

Afin de démontrer l'intérêt de nos travaux, nous avons effectué deux types de tests. Dans un premier temps nous avons évalué plusieurs critères de performance du planificateur RB-RRT kinodynamique sur des scénarios variés. Ensuite, nous avons comparé notre méthode de planification locale contre une approche naïve de l'état de l'art utilisant des bornes en accélération constantes définies par l'utilisateur.

Toute les développements ont été réalisés dans le projet logiciel Humanoid Path Planner [Mirabel 2016] en C++, tous les tests ont été effectués sur un ordinateur avec un processeur Intel Xeon CPU E5-1630 v3 à 3.7GHz (un seul cœur étant utilisé) avec 64Go de mémoire vive.

2.4.2.1 Performances du RB-RRT kinodynamique

	Path Planning				**Contact Generation**		**Total**
Scenario	Valid traj. (%)	Valid dir. (%)	num. nodes	time (s)	time (s)	success (%)	time (s)
Slalom (HyQ)	98.72 %	65.6 %	491	1.82	26.3	90 %	28.12
Plate forme (HyQ)	98.25 %	69.8 %	315	0.88	11.8	95%	12.68
Pente (25°) (HRP-2)	82.4 %	58.3 %	3895	88.1	37.5	85 %	125.6
Pente (38°) (HRP-2)	80.75 %	56.5 %	5533	373.1	75.9	90 %	449
Push recovery (HRP-2)	100 %	100 %	2	0.008	5.7	100 %	5.7

Tableau 2.1 – Temps de calcul et taux de succès par scénario, moyenne sur 50 essais (20 dans le cas de la pente). *Valid traj.* est la portion de trajectoire produite par la méthode de planification locale qui a été validée par la méthode de validation de trajectoire. *Valid dir.* est la moyenne des fois où l'égalité $\mathbf{a}' = \mathbf{a}$ a été vérifiée. *num nodes* la moyenne du nombre de nœuds contenus dans la roadmap après résolution. *success* est le taux de succès de la méthode de planification de contact.

Le tableau 2.1 montre le temps de calcul requis par notre méthode RB-RRT kinodynamique pour résoudre les scénarios présentés dans la sous-section précédente. La dernière colonne présente le temps total requis pour la planification de la trajectoire et la génération d'une séquence contacts. On peut observer des performances de quelques secondes à quelques minutes selon les scénarios, ce qui est plus rapide que l'état de l'art en locomotion dynamique en multi

contact [Deits 2014, Mordatch 2012] alors que nos scénarios sont plus longs et plus contraints.

Une comparaison directe des temps de calcul est complexe car les auteurs ne fournissent pas de temps de calcul précis et les scénarios utilisés sont différents.

En moyenne, le temps de calcul de notre méthode de planification locale est de $0.35ms$ et celui de la méthode de validation de trajectoire dynamique (en ignorant les tests de collisions) est de $0.26ms$. Ces temps sont négligeables face au temps requis par les tests de collisions qui est en moyenne de $6ms$ par itérations de l'algorithme de planification.

Dans le cas particulier du saut, le temps moyen observé est de $2.55ms$ pour la méthode de planification locale, la génération de trajectoire balistique et la validation de trajectoire dynamique.

Dans ce chapitre, nous nous sommes intéressés principalement à la partie planification de trajectoire, la partie planification de contact n'étant modifié que par le remplacement du test d'équilibre statique par l'équilibre dynamique (section 2.3.3). Nous n'analysons donc pas les performances de la méthode de planification de contacts dans ce chapitre, cela sera fait dans le chapitre 7.

Le temps de calcul de la partie planification de trajectoire est de l'ordre de la seconde pour des scénarios classiques et jusqu'à quelques minutes pour des scénarios très complexes.

La différence très importante de temps de calcul entre les scénarios sur sol plat (Slalom et Plate forme) et les scénarios sur la pente vient du fait que sur la pente, il n'existe aucune solution quasi-statique. C'est à dire que l'on ne peut avoir d'état valide sur la pente avec une accélération nulle mais seulement avec une accélération suffisante dans une direction proche de la direction de la pente.

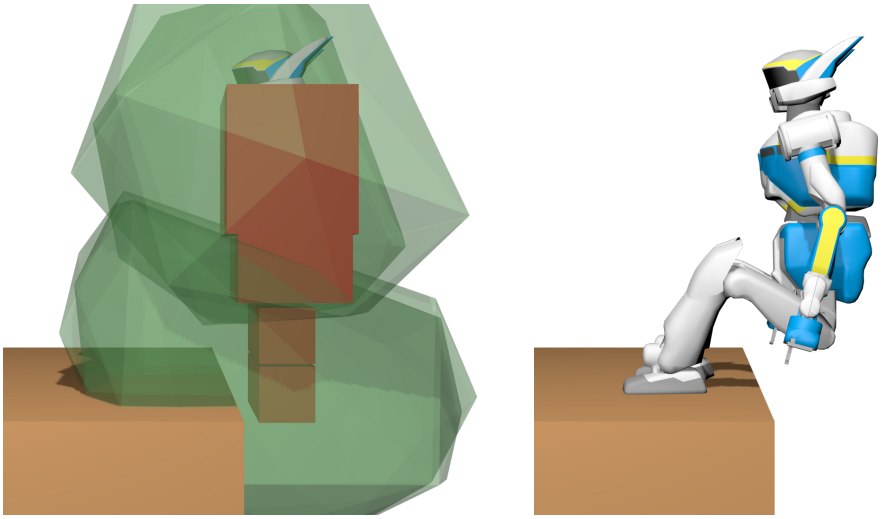
Il faut donc que le planificateur "découvre" ce comportement et trouve des trajectoires avec une accélération suffisante pour être dynamiquement valide, mais pas trop importante pour ne pas atteindre la vitesse maximale avant la fin de la pente, ce qui explique la difficulté de ces scénarios.

Le scénario de Push recovery est particulier car le planificateur peut trouver la solution en une seule itération, ce qui explique le temps de calcul de quelques millisecondes.

2.4.2.2 Taux de succès de la planification de contact

Comme soutenu dans l'introduction de ce chapitre, un des problèmes avec la méthode RB-RRT était que l'heuristique employée n'offrait aucune garantie sur l'existence de configurations en équilibre le long de la trajectoire planifiée. La figure 2.11 montre un exemple de cas dans lequel l'heuristique géométrique utilisée par RB-RRT est insuffisante.

En effet, la configuration du modèle réduit montrée sur la figure est valide selon l'heuristique, le chemin guide trouvée par RB-RRT pourrait donc contenir cette configuration. Cependant lors de la génération de contact, aucune configuration corps complet en équilibre statique ne peux être trouvée pour cette position. Donc, si un chemin guide contient ce genre de configuration du modèle réduit, la génération de contact va échouer pour ce chemin.



(a) (b)

FIGURE 2.11 – Exemple de cas mettant en échec l'heuristique géométrique : la configuration du modèle réduit (gauche) est valide selon l'heuristique car les espaces d'accessibilité des pieds sont bien en collision et la forme rouge n'est pas en collision. A droite une configuration corps complet trouvée pour la même position, cette configuration n'est pas en équilibre statique.

Dans le cas de la version kinodynamique proposée dans ce chapitre, l'état du modèle réduit montré sur la figure 2.11 ne serait valide qu'avec une importante accélération vers l'arrière. En effet, pour cette position du modèle réduit les contraintes de non glissement ne peuvent être respectées qu'en exerçant une accélération suffisante vers l'arrière.

Dans cet état le robot ne pourra donc ni rester statique ni accélérer vers l'avant, et la méthode d'extension proposée va correctement traduire ces contraintes sur les bornes d'accélération de la méthode de planification locale. Comme un tel état ne pourra certainement pas être connecté à d'autres états de la roadmap, il ne sera pas conservé.

L'avant dernière colonne du tableau 2.1 présente le taux de succès de la génération d'une séquence de contacts pour une trajectoire guide donnée. Un échec signifiant que la séquence de contacts n'a pas atteint la configuration finale demandée. Il suffit

donc d'un point dans la trajectoire guide planifiée qui ne peut pas donner lieu à une configuration en équilibre pour engendrer un échec pour ce scénario.

Les résultats montrent un taux de succès de 90% dans le pire cas sur des scénarios pour lesquels une solution quasi-statique existe, et de 85% pour les scénarios nécessitant d'exploiter la dynamique du robot.

La méthode RB-RRT kinodynamique offre donc bien de fortes probabilités que la trajectoire guide planifiée soit une entrée **faisable** pour la méthode de planification de contacts.

2.4.2.3 Intérêt du calcul des bornes d'accélération

Scénario : Slalom (HyQ)			
Bornes acc.	Temps (s)	Traj. valides (%)	Durée de la solution (s)
LP	1.82	98.32	4.75
6	time-out	0.0	X
4	106.75	9.19	5.11
2	2.68	68.15	6.98
1	4.02	76.87	8.41
Scénario : pente (38°) (HRP-2)			
Bornes acc.	Temps (s)	Traj. valides (%)	Durée de la solution (s)
LP	373.1	80.75	7.26
6	time-out	0.2	X
4	10992.6	7.6	12.53
2	time-out	8.8	X

Tableau 2.2 – Comparaison des performances de l'algorithme de planification obtenue avec des bornes d'accélération constantes définies par l'utilisateur par rapport à notre méthode (LP) en vert. *Temps* est la moyenne du temps de calcul pour résoudre le problème de planification, *Traj. valides* est la portion de trajectoire produite par la méthode de planification locale qui ont été validées par la méthode de validation de trajectoire. *Durée de la solution* est le temps total de la trajectoire trouvé en solution. Le time-out est de une heure pour le slalom et de 5 heures pour la pente.

Comme expliqué dans la section 2.2.2.1, les bornes d'accélération calculées ne sont valides que dans le voisinage de l'état initial de la méthode de planification locale. Or, la seconde colonne du tableau 2.1 montre bien que les trajectoires produites par la méthode de planification locale sont presque toutes valides.

En effet, à l'exception des scénarios sur la pente, le taux de trajectoires produites effectivement validé par notre test dynamique est proche de 100%. Dans le cas des scénarios de la pente, ce pourcentage diminue mais reste élevé. Ce qui montre que le voisinage dans lequel les bornes calculées sont valides englobe généralement l'état final de la méthode de planification locale et que donc toute la trajectoire produite est comprise dans ce voisinage.

Le tableau 2.2 montre l'intérêt de l'utilisation de notre méthode de planification locale, basée sur la méthode DIMT mais qui détermine automatiquement les bornes d'accélération admissible en fonction de l'état. Si l'on compare notre méthode avec la méthode naïve de l'état de l'art [Kunz 2014] qui utilise DIMT avec des bornes constantes et définies par l'utilisateur, on observe que le taux de succès de la méthode de validation de trajectoires est bien supérieur en utilisant notre méthode de planification locale qu'en utilisant des bornes constantes. Cela se traduit par un temps de résolution du problème de planification beaucoup plus rapide avec notre méthode.

De plus, les trajectoires produites par notre méthode ont une durée plus courte, ce qui montre que les solutions produites par notre planificateur exploitent mieux les capacités du robot. En effet, si l'on définit manuellement une borne d'accélération très faible, les trajectoires générées par la méthode DIMT seront très probablement valides mais la solution finale trouvée utilisera une accélération bien plus faible que nécessaire et la trajectoire aura une durée plus importante.

La principale différence entre les deux scénarios est que, dans le cas de la pente, les contraintes dynamiques sont très asymétriques et que les bornes d'accélérations calculées dans une direction spécifique ne sont pas valides dans la direction opposée.

Dans ce type de scénario, les résultats montrent bien que l'utilisation de notre méthode est nécessaire pour réussir à trouver une solution et que l'on ne peut pas utiliser des bornes constantes.

2.5 Discussion

Dans ce chapitre, nous avons présenté une méthode de planification locale et une méthode de validation de trajectoire propre au problème de la locomotion multi-contact des robots à pattes. Ces méthodes sont basées sur une formulation efficace de programme linéaire (LP) capturant la dynamique complexe des robots à pattes qui dépend de l'état et de la phase de contact.

Le développement théorique réalisé est applicable directement à n'importe quelle méthode de planification pour robots à pattes.

Ces méthodes ont ensuite été intégrées à l'algorithme de planification RB-RRT afin d'obtenir un **RB-RRT kinodynamique**.

Les spécificités de cet algorithme nous ont amené à faire des approximations sur la position des points de contact et à supposer que les contacts glissent le long de la trajectoire. Bien que ces approximations soit fausses dans le cas général, il faut bien considérer que dans l'architecture globale de planification utilisée, la trajectoire produite par le RB-RRT kinodynamique n'est pas la trajectoire finale utilisée par le robot mais une très bonne estimation servant de guide durant la génération de contact.

L'utilisation de nos méthodes permettant d'approximer la dynamique complexe des robots à pattes, alors que nous ne considérons que le modèle réduit du robot pendant la planification, nous permet d'une part de garantir une forte probabilité d'existence de configurations du modèle complet en équilibre le long de la trajectoire planifiée.

D'autre part, cela nous permet d'enlever la contrainte d'équilibre statique requise jusqu'à présent dans la méthode de planification de contacts.

Les résultats montrent que le RB-RRT kinodynamique est capable de planifier des trajectoires guides qui sont ensuite utilisées par le planificateur de contact pour produire des séquences de contacts pour des scénarios hautement dynamiques qui ne peuvent être résolus avec les planificateurs limités à des solutions quasi-statiques.

Ces résultats montrent également que la méthode RB-RRT permet de garantir de fortes probabilités que le guide planifié soit une entrée **faisable** pour la méthode de planification de contacts. En effet, nous avons la garantie que pour chaque état de la trajectoire guide $\mathbf{x}(t)$ planifiée il existe une phase de contact permettant de satisfaire les contraintes de non glissement pour cet état. Cependant, nous ne pouvons pas garantir qu'il existe une configuration corps-complet, cinématiquement valide et sans collision, permettant d'établir ces contacts. Nous ne pouvons que donner de fortes probabilités de l'existence d'une telle configuration via l'heuristique basée sur la notion d'accessibilité utilisée.

De plus, contrairement aux méthodes basées sur de l'optimisation, notre planificateur est capable d'éviter les minima locaux et de converger même dans un environnement très contraint par la présence d'obstacles.

En terme de performance, notre méthode permet d'atteindre des temps de calcul interactif (le temps de calcul d'une trajectoire est inférieur à la durée de la trajectoire) pour la plupart des scénarios mais nécessite jusqu'à quelques minutes pour les scénarios les plus complexes sans solution quasi-statique. L'atteinte d'un temps de calcul interactif dans le pire cas reste un de nos objectif, mais on peut noter que les performances actuelles sont meilleures que celles des autres méthodes de l'état de l'art [Deits 2014, Mordatch 2012].

Enfin, en plus de son intégration à notre architecture de planification de mouvement, nous pensons que notre méthode présente un intérêt pour produire une solution initiale pour des méthodes d'optimisation telles que [Deits 2014, Mordatch 2012]. En effet, en plus de réduire le temps de convergence de ces méthodes, une solution initiale évitant les minima locaux leur permettrait de pouvoir gérer les environnement contraints.

Deuxième partie

Critère de faisabilité d'une transition de contact

Dans l'introduction de cette thèse, nous avons présenté notre approche du problème de la locomotion multi-contact. Nous rappelons que notre approche décompose le problème en trois sous-problèmes : $\mathcal{P}_1$ la planification d'une trajectoire guide pour le centre du robot, $\mathcal{P}_2$ la planification d'une séquence de contacts, $\mathcal{P}_3$ l'optimisation de la trajectoire centroïdale du robot puis la génération d'un mouvement corps complet.

Dans la première partie de cette thèse, nous nous sommes intéressés au problème de la **faisabilité** d'une trajectoire guide pour la planification de contact. Autrement dit, à l'établissement d'un critère de faisabilité devant être respecté par la solution trouvée par le sous-problème $\mathcal{P}_1$ pour être une entrée valide au sous-problème $\mathcal{P}_2$.

Dans cette seconde partie, nous nous intéressons à la **faisabilité** d'une séquence de contacts par rapport à la génération d'une trajectoire centroïdale. Autrement dit, nous cherchons un critère permettant de garantir que la solution produite par le sous-problème $\mathcal{P}_2$ sera une entrée **faisable** pour le sous-problème $\mathcal{P}_3$.

Chapitre 3

Le problème de la faisabilité d'une transition de contact

Sommaire

3.1 Motivations

Ces dernières années, de nouvelles méthodes de génération de trajectoire ont été proposées [Carpentier 2017c, Herzog 2015, Dai 2014, Caron 2018], principalement basées sur l'utilisation du modèle centroïdal. Ces méthodes permettent de lever différentes contraintes s'appliquant aux méthodes historiques comme [Kajita 2003] et ne sont plus limitées à la génération de mouvements cycliques et sur sol plat mais peuvent générer des mouvements multi-contacts.

Cependant, ces méthodes supposent qu'une séquence de contacts *faisable* est donnée en entrée du problème. Dans ce cas, *faisable* signifie qu'il existe une trajectoire centroïdale qui respecte les contraintes dynamiques et cinématiques pour cette séquence de contacts et donc que les méthodes de génération de trajectoire vont être capables de converger avec cette séquence de contacts en entré.

D'autres contributions concernent la planification de contacts, c'est à dire le problème de trouver les contacts qui doivent être créés avec l'environnement pour déplacer le robot vers un but. [Bretl 2004, Hauser 2005, Escande 2013] posent les fondations théoriques de ce problème mais seulement pour le cas quasi-statique. De plus, ces travaux ne fournissent pas de méthodes efficaces en terme de temps de calcul pour résoudre le problème.

[Deits 2014] planifie bien une séquence de contacts, cependant la *faisabilité* de cette séquence n'est pas vérifiée avant d'essayer de générer un mouvement, ce qui entraîne forcément des échecs de la méthode de génération de trajectoire centroïdale. Cette contrainte est levée par [Ponton 2016] au prix d'une approximation de la dynamique centroïdale.

Pour pallier à ce problème, plusieurs approches ont été proposées. [Mordatch 2012] propose de générer conjointement la séquence de contacts et le mouvement via des méthodes d'optimisation utilisant un modèle simplifié. Cependant, cette méthode est trop coûteuse en temps de calcul, les résultats présentés demandant plusieurs minutes pour produire des mouvements de quelques pas.

Dans [Tonneau 2018a] des heuristiques sont utilisées pendant la sélection des contacts pour augmenter les probabilités de trouver une séquence de contacts faisable tout en permettant un temps de calcul interactif. Bien qu'il soit montré empiriquement l'utilité de ces heuristiques, elles ne sont pas suffisantes et aucune garantie sur la faisabilité des séquences de contacts n'est fournie.

Récemment nous avons proposé une méthode donnant une garantie formelle de l'existence d'une trajectoire centroïdale valide entre deux états en contact [Tonneau 2018b],résumé dans la section 3.5.3, mais cette méthode est limitée à des mouvements quasi-statiques.

Pour résoudre ce problème, une approche naïve pourrait consister à utiliser une méthode de planification de contacts de la littérature et à essayer une méthode de génération de trajectoire centroïdale sur la séquence de contacts produite. Puis, à modifier cette séquence de contacts après chaque échec potentiel de la génération de trajectoire jusqu'à obtenir une séquence faisable permettant de faire converger la méthode de génération de trajectoire.

Cependant, les méthodes de génération de trajectoire actuelles sont trop coûteuses en temps de calcul pour utiliser ce type d'approche par essais et erreurs durant la planification. En effet, les méthodes de génération de trajectoire centroïdale les plus efficaces à notre connaissance et permettant de gérer des mouvements multi-contact avec changement de phases de contact telles que [Carpentier 2017c] demandent plusieurs dizaines de millisecondes par phase de contact, [Ponton 2016] est plus rapide mais reste dans le même ordre de grandeur. D'autres contributions telles que [Caron 2018] obtiennent des temps de calcul d'un ordre de grandeur inférieur mais sont pour leur part limitées par des cas d'application restreints, ici de simple support à simple support. Or dans le cas de la planification de contacts, nous

sommes amenés à tester la faisabilité de dizaines, voire de centaines, de transitions possibles pour chaque nouvelle phase de contact potentielle. Ce qui impliquerait des temps de calcul bien trop élevés.

Notre contribution est donc de définir formellement le problème de la faisabilité d'une transition de contact. Ce problème étant plus simple que celui de la génération de trajectoire puisque nous ne cherchons qu'à déterminer l'existence d'au moins une trajectoire centroïdale *valide*, c'est à dire qui respecte les contraintes dynamiques et cinématiques du robot.

Puis, nous proposons un critère permettant de vérifier cette faisabilité d'une manière efficace via la formulation d'un LP (Linear Program) de faible dimension. Ce LP peut alors être résolu grâce à des méthodes propres aux problèmes linéaires, avec un temps de calcul d'au moins un ordre de grandeur inférieur aux méthodes de génération de trajectoires de l'état de l'art.

Ensuite, nous montrerons que nous pouvons appliquer ce critère de faisabilité d'une transition de contact pendant la planification de contact pour rejeter les contacts qui donneraient lieu à des transitions de contact infaisables. Ainsi nous pouvons produire une séquence de contacts **faisable**, garantissant l'existence d'une solution au problème de génération de trajectoire centroïdale pour cette séquence de contacts.

Une autre application de ce critère de faisabilité sont les problèmes du *0-step capturability* et *1-step capturability* [Koolen 2012] [Pratt 2012] [Del Prete 2018]. Ces problèmes consistent à déterminer s'il est possible, pour un robot dans un état donné avec une vitesse non-nulle, de parvenir à s'arrêter dans une configuration en équilibre statique sans changer de contact ou respectivement avec un seul changement de contact. La résolution rapide de ce problème est essentielle pour la sécurité du robot et de son environnement.

Une autre application possible de notre méthode est de fournir une solution initiale aux méthodes de résolutions non linéaires utilisées pendant la génération de trajectoire centroïdale [Carpentier 2017c].

3.2 Définition du problème

Nous définissons le problème de la faisabilité d'une transition de contact de la manière suivante. Étant donnés deux états du robot dans des phases de contact différentes, peut on garantir qu'il existe (ou qu'il n'existe pas) de mouvement respectant les contraintes cinématiques et dynamiques du robot, reliant ces deux états ?

Dans notre cas, nous considérons ce problème via le modèle centroïdal défini dans la section Modèle centroïdal, c'est à dire qu'un état est défini par $\mathbf{x} = (\mathbf{c}, \dot{\mathbf{c}}, \ddot{\mathbf{c}}) \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3$ un triplet décrivant la position, vitesse et accélération du centre de masse. A chaque état est associée une phase de contact $\{p\}$, placée en exposant : $\mathbf{x}^{\{p\}}$.

Nous rappelons qu'une phase de contact $\{p\}$, $p \in \mathbb{N}$, définit la position $\mathbf{P}^{\{p\}}$ et la normale $\mathbf{N}^{\{p\}}$ de tous les contacts entre le robot et l'environnement durant cette phase. Le nombre de contacts ainsi que leurs positions et normales demeure constant durant toute la phase de contact.

Le problème de faisabilité d'une transition de contact peut donc s'écrire plus formellement ainsi :

Soient deux états $\mathbf{x}_s^{\{p\}}$ et $\mathbf{x}_g^{\{q\}}$, avec $p \leq q$, existe-t-il une trajectoire *valide* $\mathbf{x}(t)$, $t \in \mathbb{R}^+$, de durée $T \in \mathbb{R}^+$ telle que $\mathbf{x}(0) = \mathbf{x}_s^{\{p\}}$ et $\mathbf{x}(T) = \mathbf{x}_g^{\{q\}}$?

Cette trajectoire doit donc présenter au moins un changement de phase de contact. Nous supposons qu'un changement entre deux phases de contact adjacentes est instantané. En effet il s'agit soit de rompre un contact avec l'environnement soit d'en créer un, ce qui est fait de manière instantanée.

A l'instant du changement de phase de contact, l'état $\mathbf{x}$ doit être *valide* pour les deux phases de contact, nous le notons $\mathbf{x}^{\{p,q\}}$.

Un état est *valide* si et seulement si il satisfait les contraintes définies par la phase de contact courante, qui impose des contraintes cinématiques et dynamiques sur le centre de masse.

Bien qu'il soit relativement simple de vérifier qu'un état du modèle complet du robot satisfasse ces contraintes, les exprimer pour le modèle centroïdal reste une tâche ardue qui n'est pas encore entièrement résolue [Tonneau 2018b, Herzog 2016].

3.3 Contraintes cinématiques appliquées au centre de masse

Vérifier la faisabilité cinématique d'un état $\mathbf{x}^{\{p\}} = (\mathbf{c}, \dot{\mathbf{c}}, \ddot{\mathbf{c}})$ revient à vérifier qu'il existe une configuration corps complet du robot $\mathbf{q} \in \mathcal{C}$ qui permet d'amener la position du centre de masse du robot à $\mathbf{c}$ et que les positions des points de contact $\mathbf{P}^{\{p\}}$ définies par la phase de contact soient respectées. De plus, la configuration $\mathbf{q}$ doit respecter les limites articulaires du robot et ne doit pas être en auto-collision.

Il est possible de vérifier cette contrainte explicitement via des méthodes de projection, en contraignant la position du centre de masse et des points de contact définie par $\mathbf{x}^{\{p\}}$ et en essayant de projeter une configuration $\mathbf{q}$ sur ces contraintes. La résolution de cette cinématique inverse, résolue par programmation quadratique ou par descente de Newton, reste trop coûteuse en temps de calcul pour notre problème [Baerlocher 2004, Dai 2014].

A notre connaissance, il n'existe pas de méthode exacte permettant d'exprimer les contraintes cinématiques imposées par une phase de contact sur la position possible du centre de masse. Mais, différentes approches existent dans la littérature pour les approximer.

Certaines approches expriment la faisabilité cinématique comme un coût. Par exemple, dans [Carpentier 2017a] la densité de probabilité de trouver une configuration corps complet pour des positions de contact et une position du centre de masse donnée est apprise et représentée par des mélanges de Gaussiennes. Cela permet d'utiliser cette probabilité comme un coût durant l'optimisation.

Cependant, la principale limitation de cette approche consiste en une hypothèse d'indépendance entre toutes les pattes du robot : les contraintes cinématiques imposées par la position d'un organe terminal sont indépendantes de la position des autres organes terminaux. De plus, l'introduction dans la fonction de coût de la faisabilité cinématique lors de l'optimisation de la dynamique centroïdale ne garantit pas que la trajectoire du centre de masse trouvée permettra de satisfaire ces contraintes cinématiques. En effet, cette méthode peut trouver des trajectoires centroïdales qui donnent lieu à des violations des limites articulaires lorsque l'on génère le mouvement corps complet. Enfin, cette méthode requiert de régler correctement les différents poids de la fonction de coût.

3.3.1 Approximation convexe des contraintes cinématiques

Dans [Tonneau 2018b] nous avons proposé une approximation des contraintes cinématiques, représentées par un polytope 3D convexe dépendant seulement des positions et orientations des organes terminaux des membres en contact. Tester si une position du centre de masse est faisable d'après cette approximation revient donc à tester si elle est à l'intérieur de ce polytope.

Ce polytope est calculé de la manière suivante : pour chaque membre i, un grand nombre de configurations cinématiquement valides et sans auto-collision est échantillonné aléatoirement hors ligne. Les positions du centre de masse exprimées dans le repère local de l'organe terminal du membre sont alors enregistrées. Puis, l'enveloppe convexe de ces positions est calculée. On obtient donc pour chaque membre, un polytope convexe exprimé dans le repère local de l'organe terminal.

Ensuite, durant l'exécution, pour chaque phase de contact ces polytopes sont transformés dans le repère monde en fonction de la position et de l'orientation de l'organe terminal en contact pour obtenir un polytope $\boldsymbol{\mathcal{K}}_i^{\{p\}}$.

L'appartenance d'un point à l'intérieur de ce polytope peut alors être exprimée par un ensemble d'inégalités linéaires :

$$\mathbf{c} \in \boldsymbol{\mathcal{K}}_i^{\{p\}} \Leftrightarrow \mathbf{K}_i^{\{p\}}\mathbf{c} \leq \mathbf{k}_i^{\{p\}} \tag{3.1}$$

avec $\mathbf{K}_i^{\{p\}} \in \mathbb{R}^{n_f \times 3}$ et $\mathbf{k}_i^{\{p\}} \in \mathbb{R}^{n_f}$ respectivement une matrice et un vecteur définis par les équations des faces du polytope $\boldsymbol{\mathcal{K}}_i^{\{p\}}$, avec n_f le nombre de faces. En ajoutant toutes les contraintes pour chaque membre en contact i, on obtient alors :

$$\mathbf{c} \in \boldsymbol{\mathcal{K}}^{\{p\}} \Leftrightarrow \mathbf{K}^{\{p\}}\mathbf{c} \leq \mathbf{k}^{\{p\}} \tag{3.2}$$

Ce qui revient à vérifier l'appartenance de $\mathbf{c}$ à l'intersection de tout les polytopes : $\boldsymbol{\mathcal{K}}^{\{p\}} = \cap_{i=0}^{k} \boldsymbol{\mathcal{K}}_i^{\{p\}}$.

Deux approximations sont faites par cette méthode : l'espace des positions valides du centre de masse est convexe, et les contraintes cinématiques imposées par la position d'un organe terminal sont indépendantes de la position des autres organes terminaux.

Cette approximation est donc une condition non suffisante sur la position du centre de masse $\mathbf{c}$ à l'existence d'une configuration corps complet cinématiquement valide pour une phase de contact donnée. En appliquant un facteur d'échelle au polytope calculée afin de s'assurer qu'il englobe bien toutes les positions du centre de masse valides, nous obtenons une condition nécessaire mais non suffisante.

Cependant, cette approximation a l'avantage de pouvoir être exprimée par un ensemble d'inégalités linéaires sur $\mathbf{c}$. La figure 3.1 montre la représentation 3D du polytope $\boldsymbol{\mathcal{K}}^{\{p\}}$.

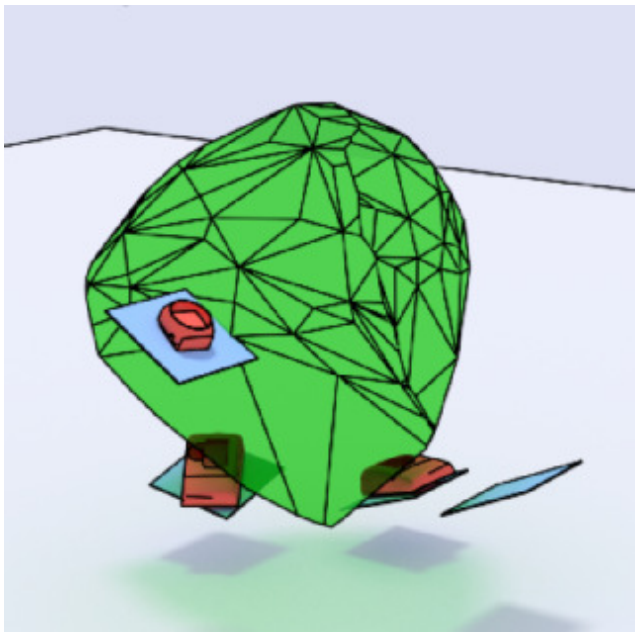


FIGURE 3.1 – Exemple de l'approximation convexe des contraintes cinématiques. En rouge, les points de contact entre le robot et l'environnement définissant la phase de contact $\{p\}$. En vert, le polytope $\boldsymbol{\mathcal{K}}^{\{p\}}$. Vérifier si $\mathbf{c}$ satisfait les contraintes cinématiques approximées reviens à vérifier si sa position est à l'intérieur du polytope vert.

3.4 Étude des différentes formulations de la dynamique centroïdale

Contrairement aux contraintes cinématiques, la formulation des contraintes dynamiques pour le modèle centroïdal est exacte et ne requiert pas d'approximation. Ces contraintes sont imposées par les forces de contact qui doivent être à l'intérieur des cônes de friction de chaque contact et qui sont données par les équations de Newton-Euler (détaillées dans la section Contraintes dynamiques).

$$\underbrace{\begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) + \dot{\mathbf{L}} \end{bmatrix}}_{\mathbf{w}} = \begin{bmatrix} \mathbf{I}_3 & ... & \mathbf{I}_3 \\ \hat{\mathbf{p}}_1 & ... & \hat{\mathbf{p}}_k \end{bmatrix} \mathbf{f} \tag{3.3}$$

Où :
- m est la masse totale du robot ;
- k est le nombre total de points de contact ;
- $\mathbf{p}_i \in \mathbb{R}^3, 0 \leq i \leq k$ est la position du i-ième point de contact ;
- $\mathbf{f} = \begin{bmatrix} \mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_k \end{bmatrix}^T \in \mathbb{R}^{3k}$ est le vecteur de forces de contact appliquées à chaque point de contact ;
- $\mathbf{g} = \begin{bmatrix} 0 & 0 & -9.81 \end{bmatrix}^T$ est le vecteur de gravité ;
- $\dot{\mathbf{L}} \in \mathbb{R}^3$ est la dérivée du moment cinétique (exprimé au point $\mathbf{c}$) ;
- $\hat{\mathbf{p}}_i$ est la matrice de préproduit vectoriel de $\mathbf{p}_i$;
- $\mathbf{w} \in \mathbb{R}^6$ est appelé le *Gravito-Inertial-Wrench* (GIW).

Il existe dans la littérature deux formulations différentes pour exprimer ces contraintes. Ces deux formulations sont équivalentes en théorie mais présentent des différences pratiques [Del Prete 2016]. Elles sont détaillées dans les sous-sections suivantes.

3.4.1 Formulation en force

La manière la plus évidente de vérifier si un état est en équilibre est de chercher s'il existe des forces de contact vérifiant les équations de Newton-Euler et qui sont comprises dans les cônes de frictions de chaque contact. Il s'agit de la formulation détaillée dans la section Contraintes dynamiques et utilisée dans la première partie de ce manuscrit.

Nous rappelons que vérifier l'existence de ces forces de contact respectant les conditions de non glissement est équivalent à vérifier l'existence d'un vecteur $\boldsymbol{\beta} \in \mathbb{R}^{4k}$ tel que :

$$\underbrace{\begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) + \dot{\mathbf{L}} \end{bmatrix}}_{\mathbf{w}} = \begin{bmatrix} \mathbf{I}_3 & ... & \mathbf{I}_3 \\ \hat{\mathbf{p}}_1 & ... & \hat{\mathbf{p}}_k \end{bmatrix} \mathbf{V}\boldsymbol{\beta} \ , \ \boldsymbol{\beta} \geq 0 \tag{3.4}$$

Avec $\mathbf{V} = diag(\begin{bmatrix}\mathbf{V}_1 & ... & \mathbf{V}_k\end{bmatrix}) \in \mathbb{R}^{3k\times 4k}$ et les $\mathbf{V}_i$ des matrices définies par les génératrices des cônes de friction linéarisés.

3.4.2 Méthode de double description

Une autre formulation des contraintes dynamiques consiste à calculer le *Gravito Interial Wrench Cone* (GIWC) sous la forme d'un ensemble d'inégalités linéaires, puis à tester si $\mathbf{w}$ appartient bien à ce cône. Cette formulation est basée sur l'observation qu'un cône linéarisé $\mathcal{P} \subseteq \mathbb{R}^n$ peut être représenté de deux manières équivalentes :

$$\begin{aligned}\mathcal{P} =& \{x \in \mathbb{R}^n : x = P_s\lambda, \lambda \geq 0\} \\ & \{x \in \mathbb{R}^n : P_f x \geq 0\}\end{aligned} \tag{3.5}$$

Où les matrices P_f et P_s sont appelées respectivement description des faces et génératrices de $\mathcal{P}$. La conversion entre les représentations par génératrices ou par faces est possible via un algorithme dédié [Fukuda 1996] et est appelé la méthode de double description. Cependant, cette méthode a un coût de calcul non négligeable et est sujette à des échecs occasionnels.

Comme l'ensemble des forces de contact admissibles défini par l'équation (3.4) forme une cône, il est possible de transformer l'égalité (3.4) en une inégalité via la méthode de double description. Cette formulation a été introduite par [Qiu 2011] puis décrite plus en détails dans [Caron 2015].

L'inégalité obtenue est de la forme suivante :

$$\mathbf{H}(\mathbf{p}_1, .., \mathbf{p}_k, \mathbf{n}_1, .., \mathbf{n}_k, \mu_1, .., \mu_k) \underbrace{\begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) + \dot{\mathbf{L}} \end{bmatrix}}_{\mathbf{w}} \leq \mathbf{h}(\mathbf{p}_1, .., \mathbf{p}_k, \mathbf{n}_1, .., \mathbf{n}_k, \mu_1, .., \mu_k) \tag{3.6}$$

Où $\mathbf{H} \in \mathbb{R}^{n_d\times 6}$ et $\mathbf{h} \in \mathbb{R}_{n_d}$ sont une matrice et un vecteur dépendants uniquement de la phase de contact (positions et normales des points de contact) et du coefficient de friction. La valeur de n_d dépend en partie du nombre de points de contact k mais l'algorithme de double description ne garantit pas de relation directe entre le nombre de points de contact et n_d [Fukuda 1996].

Empiriquement, nous avons observé que $n_d \in [32; 200]$ pour $k = 8$ (deux contacts rectangulaires générant 4 points de contact chacun). Nous avons également observé que n_d tend à se rapprocher de sa borne minimale dans le cas où les contacts sont coplanaires et tend à augmenter dans le cas contraire.

Par simplicité nous notons cette matrice et ce vecteur avec un exposant décrivant la phase de contact pour laquelle ils ont étés calculés :

$$\mathbf{H}^{\{p\}} \underbrace{\begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) + \dot{\mathbf{L}} \end{bmatrix}}_{\mathbf{w}} \leq \mathbf{h}^{\{p\}} \tag{3.7}$$

3.5 Cas quasi-statique

3.5.1 Formulation en force

Dans le cas particulier de l'équilibre statique, nous avons $\dot{\mathbf{L}} = 0$ et $\ddot{\mathbf{c}} = 0$. La matrice $\mathbf{w}$ de l'équation (3.4) peut alors se simplifier ainsi :

$$\mathbf{w} = \begin{bmatrix} \mathbf{0}_{3\times3} \\ m\hat{\mathbf{g}} \end{bmatrix} \mathbf{c} + \begin{bmatrix} -m\mathbf{g} \\ 0 \end{bmatrix} \tag{3.8}$$

Afin de tester si une position du centre de masse $\mathbf{c}$ permet d'être en équilibre statique, il suffit alors de vérifier l'existence de forces de contact valides en résolvant le LP suivant :

$$\begin{aligned} &\text{find } \boldsymbol{\beta} \\ &\text{s.t. } \boldsymbol{\beta} \geq 0 \\ &\quad \begin{bmatrix} \mathbf{0}_{3\times3} \\ m\hat{\mathbf{g}} \end{bmatrix} \mathbf{c} + \begin{bmatrix} -m\mathbf{g} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{I}_3 & \dots & \mathbf{I}_3 \\ \hat{\mathbf{p}}_1 & \dots & \hat{\mathbf{p}}_k \end{bmatrix} \mathbf{V}\boldsymbol{\beta} \end{aligned} \tag{3.9}$$

Si ce LP converge, alors une répartition admissible des forces de contact existe pour cette position du centre de masse et l'équilibre statique peut être atteint.

3.5.2 Méthode de double description

Avec la formulation par inégalités, vérifier l'équilibre statique pour une position du centre de masse $\mathbf{c}$ revient à vérifier un ensemble d'inégalités linéaires :

$$\mathbf{H} \begin{bmatrix} \mathbf{0}_{3\times3} \\ m\hat{\mathbf{g}} \end{bmatrix} \mathbf{c} \leq \mathbf{h} - \mathbf{H} \begin{bmatrix} -m\mathbf{g} \\ 0 \end{bmatrix} \tag{3.10}$$

ce qui est extrêmement rapide en temps de calcul.

Ces contraintes définissent donc un polytope 3D convexe auquel $\mathbf{c}$ doit appartenir, noté $\mathcal{A}$. La figure 3.2 (seconde et troisième ligne, en jaune) montre des exemples d'un tel polytope.

3.5.3 Problème de faisabilité d'une transition de contact

Dans le cas particulier d'un mouvement quasi-statique, l'existence d'un chemin $\mathbf{c}(t)$ connectant deux positions du centre de masse appartenant à deux phases de contact différentes $\mathbf{c}_s^{\{p\}}$ et $\mathbf{c}_g^{\{q\}}$ et respectant les contraintes associées à ces phases,

peut alors être prouvée géométriquement. En effet, en supposant que les positions initiales et finales sont valides, un tel chemin existe si et seulement si l'intersection des polytopes des contraintes pour les phases de contact $\{p\}$ et $\{q\}$ n'est pas nulle.

L'existence d'un point $\mathbf{c}_{switch}^{\{p,q\}}$ valide à la fois pour les contraintes de la phase $\{p\}$ et de la phase $\{q\}$ garantit l'existence d'un chemin valide connectant $\mathbf{c}_s^{\{p\}}$ et $\mathbf{c}_g^{\{q\}}$.

Ce raisonnement est résumé dans la figure 3.2 : chaque phase de contact (ligne du haut) impose des contraintes d'équilibre statique $\mathcal{A}^{\{j\}}$ et cinématiques $\mathcal{K}^{\{j\}}$ (seconde ligne). Les contraintes d'équilibre statique sont définies via la représentation par polytopes décrite dans la sous-section précédente. Les contraintes cinématiques sont représentées via l'approximation convexe présentée dans la section 3.3.1.

L'intersection de ces contraintes est notée $\mathcal{C}^{\{j\}}$ et définit un polytope 3D (troisième ligne). Une position du centre de masse $\mathbf{c}^{\{j\}}$ satisfait les contraintes de la phase de contact $\{j\}$ si et seulement si $\mathbf{c}^{\{j\}} \in \mathcal{C}^{\{j\}}$.

L'intersection de deux polytopes de contraintes pour deux phases de contact adjacentes est notée $\mathcal{T}^{\{j,j+1\}}$. Si ce polytope n'est pas nul, alors il existe une position du centre de masse $\mathbf{c}_{switch}^{\{p,q\}} \in \mathcal{T}^{\{p,q\}}$ qui garantit l'existence d'un chemin solution.

Dans ce cas, le chemin solution consiste à connecter $\mathbf{c}_s^{\{p\}}$ à $\mathbf{c}_{switch}^{\{p\}}$ via une ligne droite, puis à effectuer le changement de phase de contact de $\{p\}$ vers $\{q\}$ en conservant la position du centre de masse à $\mathbf{c}_{switch}$, et enfin à connecter $\mathbf{c}_{switch}^{\{q\}}$ à $\mathbf{c}_g^{\{q\}}$. Ce chemin est montré sur la dernière ligne de la figure 3.2.

Ce chemin est valide car le changement de phase de contact s'effectue pour une position du centre de masse telle que les contraintes sont respectées pour les deux phases de contact. De plus, les contraintes sont forcément respectées le long des deux segments de droites puisque les contraintes sont convexes et que les deux extrémités des segments respectent ces contraintes.

Dans le cas du problème de la faisabilité d'une transition de contact, nous ne nous intéressons pas à la génération de ce chemin ou à son optimisation. Nous ne nous intéressons donc pas à trouver une position pour $\mathbf{c}_{switch}^{\{p,q\}}$. En effet, nous cherchons seulement à prouver que l'intersection des contraintes $\mathcal{T}^{\{p,q\}}$ n'est pas nulle, ce qui garantit l'existence d'au moins une solution et donc que la transition de la phase de contact $\{p\}$ à la phase $\{q\}$ est **faisable**.

3.6 Discussion

Il apparaît donc qu'il existe différentes formulations dans la littérature pour exprimer les contraintes (cinématiques ou dynamique) s'appliquant à la trajectoire centroïdale.

En ce qui concerne les contraintes cinématiques, il s'agit d'approximations ou de relaxation des contraintes réelles pour pouvoir les appliquer au modèle centroïdal.

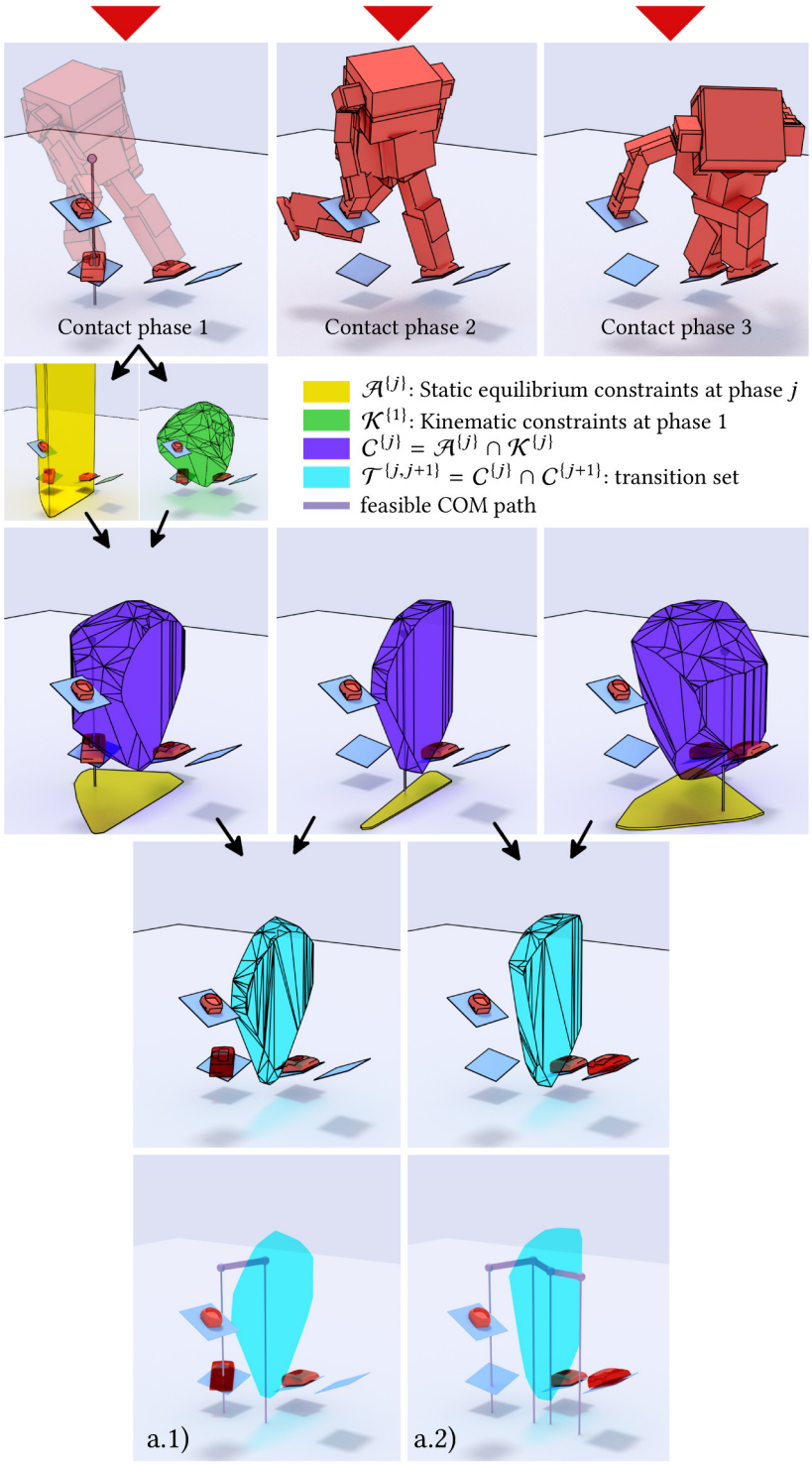


FIGURE 3.2 – Vue globale de la méthode de résolution du problème de la faisabilité d'une transition de contact dans le cas quasi-statique.

Dans les chapitres suivants nous avons décidé d'utiliser la condition nécessaire mais non suffisante présentée dans la sous-section 3.3.1 car elle permet d'exprimer les contraintes comme des inégalités linéaires sur la position du centre de masse. La validité et la justesse de cette approximation seront évaluées dans la section 4.6.6.

Pour les contraintes dynamiques, la différence majeure entre les deux formulations présentées est la dimension du problème. Dans le cas de la formulation des contraintes par des inégalités, la dimension est grandement réduite, mais il faut passer par l'étape de la double description qui est non-triviale et nécessite une méthode coûteuse en temps de calcul et sensible numériquement.

Dans le cas particulier d'un mouvement quasi-statique, nous avons proposé une méthode permettant de résoudre le problème de la faisabilité d'une transition de contact de manière géométrique et exacte pour les contraintes utilisées, mais nos contraintes cinématiques sont approximées. Cependant, cette méthodologie ne peut pas être appliquée directement au cas général puisque les contraintes dynamiques ne peuvent plus être représentées par des polytopes convexes.

3.6.1 Non convexité des contraintes dynamiques

Dans le cas général où l'accélération du centre de masse n'est pas négligeable, l'expression du Gravito-Interial-Wrench $\mathbf{w}$ est non-linéaire à cause du produit vectoriel entre $\mathbf{c}$ et $\ddot{\mathbf{c}}$. Les contraintes dynamiques sont donc non-convexes, et cela indépendamment de la formulation utilisée. Pour pouvoir résoudre ce problème, il faut donc utiliser des méthodes de résolution non-linéaires [Carpentier 2017c, Winkler 2018] qui présentent plusieurs inconvénients.

Tout d'abord, ces méthodes sont généralement plus coûteuses en temps de calcul que des méthodes propres aux problèmes linéaires. Ensuite, ces méthodes n'ont pas de garanties de convergence, et elles peuvent rester coincées dans des minima locaux.

Enfin, elles requièrent une initialisation avec une solution initiale. La qualité de cette solution initiale peut affecter le temps et le taux de convergence.

Pour pallier les problèmes engendrés par l'utilisation de méthode de résolution non-linéaire, nous proposons une reformulation convexe mais conservatrice des contraintes dynamiques dans le chapitre suivant. Les méthodes de résolution de problèmes convexes offrant des garanties de convergence ainsi que des temps de calcul inférieurs aux méthode non-linéaires, cette reformulation est très intéressante.

Une reformulation convexe du problème de l'optimisation de la dynamique centroïdale a déjà été proposé dans [Ponton 2016] puis [Ponton 2018] mais dans ces travaux une relaxation de la dynamique centroïdale est utilisée. Bien que leurs résultats semble montrer que cette relaxation n'introduit pas d'erreurs importantes, il s'agit tout de même d'une approximation. Dans notre cas, la convexité est obtenue sans aucune approximation mais avec une réduction de l'espace des solutions possibles, ce qui rend notre méthode conservatrice.

3.6.2 Le problème de la discrétisation

Un autre problème non résolu dans l'état de l'art est que les différentes formulations des contraintes cinématiques ou dynamiques présentées depuis le début de ce chapitre permettent de vérifier la validité d'un état $\mathbf{x}^{\{p\}} = (\mathbf{c}, \dot{\mathbf{c}}, \ddot{\mathbf{c}})$ donné pour une phase de contact $\{p\}$ donnée. Or, nous nous intéressons ici à vérifier la validité d'une **trajectoire** $\mathbf{x}(t)$ donnée.

Classiquement, les méthodes de résolution de ce type de problème trouvée dans la littérature [Carpentier 2017c, Herzog 2015, Dai 2014, Caron 2018, Fernbach 2018] fonctionnent en discrétisant la trajectoire et en ajoutant des contraintes pour chaque état $\mathbf{x}(j\Delta t)$ discrétisé le long de la trajectoire.

Ce fonctionnement entraîne plusieurs problèmes. Tout d'abord, la validité de la trajectoire n'est pas entièrement garantie. En effet, il est possible que la trajectoire viole des contraintes entre deux points de discrétisations valides, comme montré sur la figure 3.3. Ce problème a déjà été identifié lors de la résolution de problèmes de génération de trajectoires, par exemple par [Arisumi 2008, Lengagne 2011].

De plus, certaines méthodes de résolution classiques ont tendance à saturer les contraintes, et si une contrainte exprimée en deux points de discrétisation adjacent est saturée, il y a un risque élevé que cette contrainte soit violée par la trajectoire entre ces deux points.

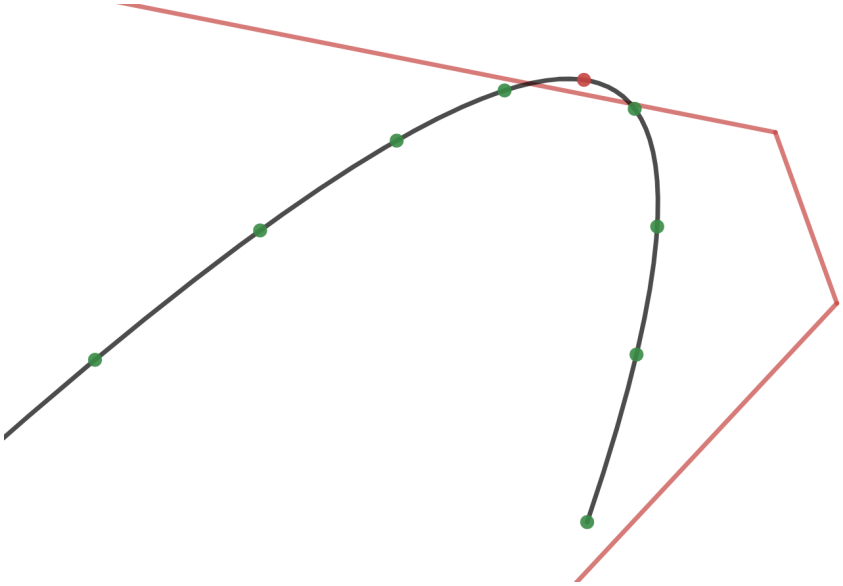


FIGURE 3.3 – Exemple de trajectoire invalide alors que chaque point de discrétisation est valide. En noir la trajectoire, en rouge les contraintes. Les points vert sont les points de discrétisation, le point rouge est un point invalide.

D'autre part, le nombre de contraintes du problème complet va dépendre du nombre de points de discrétisation choisis, et le temps nécessaire pour la résolution d'un problème dépend du nombre de contraintes de ce problème. Il faut donc faire

un compromis entre le temps de calcul et le risque d'obtenir des solutions qui ne seraient pas valides.

Une exception notable est [Lengagne 2013] qui vérifie que pour chaque intervalle de temps discrétisé la borne supérieur des contraintes est bien valide, garantissant ainsi la validité de toute la trajectoire, peu importe le pas de discrétisation choisi. Cette vérification est faite en représentant chaque contrainte entre chaque intervalle de temps par une B-spline grâce à une approximation polynomiale de la fonction de contrainte par série de Taylor. Puis, en utilisant la propriété de convexité des B-Spline, la borne supérieure de la courbe est contrainte.

Dans le chapitre 5 nous proposons une formulation continue des contraintes basée sur une idée similaire à ces travaux mais qui ne nécessite pas d'utiliser des approximations polynomiales à chaque itération, qui demandent un temps de calcul supplémentaire.

Cette formulation continue permet de garantir exactement la validité de toute la trajectoire avec un nombre fixe de contraintes.

3.7 Conclusion

Nous rappelons que le but de cette seconde partie du manuscrit est de résoudre le problème de la **faisabilité** d'une transition de contact, afin de pouvoir garantir que la séquence de contacts planifiée pendant la résolution du sous problème $\mathcal{P}_2$ soit une entrée valide au problème de génération de trajectoire centroïdale $\mathcal{P}_3$.

Dans ce but, nous avons défini formellement le problème de la faisabilité d'une transition de contacts avant de détailler les différentes formulations liées à ce problème puis de proposer une méthode de résolution dans le cas quasi-statique.

Dans les deux chapitres suivants nous allons proposer une méthode permettant de le résoudre efficacement dans le cas général. Finalement, l'intégration de ce problème comme un **critère de faisabilité** lors de la planification de contact sera décrit dans la troisième partie de ce manuscrit.

Chapitre 4

Reformulation convexe du problème de la dynamique centroïdale (CROC)

Sommaire

Dans ce chapitre, nous nous intéressons à reformuler les équations de la dynamique centroïdale présentées dans le chapitre précédent de manière convexe mais

sans se limiter au cas quasi-statique. Pour cela, nous imposons une condition conservatrice sur $\mathbf{c}(t)$.

Ensuite, nous utilisons cette reformulation convexe pour résoudre le problème de la faisabilité d'une transition de contact : déterminer l'existence une trajectoire centroïdale respectant les contraintes cinématiques et dynamiques du robot et connectant exactement deux états appartenant à des phases de contact différentes. Nous montrons que ce problème peut alors être formulé comme un LP de faible dimension, ce qui permet d'atteindre des temps de calcul d'au moins un ordre de grandeur inférieur aux méthodes de résolutions non-linéaires.

Cette méthode est appelée *CROC* pour "Convex Resolution Of Centroïdal dynamic trajectories" [Fernbach 2018].

Comme présenté dans le chapitre précédent il existe deux formulations principales des contraintes dynamiques, dans ce chapitre nous détaillerons les calculs avec la formulation par inégalité (en utilisant la méthode de double description) car les développements sont plus intuitifs à comprendre. Cependant, la même démarche peut-être réalisée avec l'autre formulation, et dans la section 4.6.5 nous comparons les deux formulations et discutons de leurs avantages et inconvénients respectifs.

4.1 Représentation de la trajectoire centroïdale par une courbe de Bezier

L'idée principale exploitée pour reformuler le problème de la dynamique centroïdale de manière convexe est de représenter la trajectoire centroïdale par une courbe de Bézier, puis d'exploiter les propriétés de ce type de courbe pour exprimer le GIW $\mathbf{w}(t)$ de manière linéaire.

Dans ce travail, nous faisons l'hypothèse que la trajectoire centroïdale est décrite par un polynôme. Comme les courbes de splines, les courbes de Bézier peuvent représenter n'importe quel polynôme et permettent d'exprimer aisément les contraintes aux bornes d'une courbe. Nous choisissons une représentation par des courbes de Bézier pour cette raison, mais également pour leurs propriétés de convexité qui seront utilisées pour permettre d'imposer aisément des contraintes additionnelles sur l'ensemble du polynôme ainsi que ses dérivées. Ces propriétés seront par ailleurs mises à contribution dans le chapitre suivant, où nous proposerons une formulation continue et générale du problème centroïdal.

Dans le cas général, une courbe de Bézier $\mathbf{c}(t)$ de degré n et de durée T s'exprime de la manière suivante :

$$\mathbf{c}(t) = \sum_{i=0}^{n} B_i^n(t/T)\mathbf{P}_i \tag{4.1}$$

où les B_i^n sont les polynômes de Bernstein, définis par :

$$B_i^n(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i},\ u \in [0;1] \tag{4.2}$$

Les $\mathbf{P}_i$ sont les points de contrôle de la courbe de Bézier. Il y a donc $n+1$ points de contrôle pour une courbe de degré n.

4.1.1 Conditions aux limites d'une courbe de Bézier

Les courbes de Bézier permettent de contraindre aisément la valeur de la courbe et de ses dérivées au point initial $\mathbf{c}(0)$ ou final $\mathbf{c}(T)$ en définissant la valeur d'un point de contrôle par contrainte.

Dans le cas général, une trajectoire centroïdale doit connecter exactement deux états $\mathbf{x}_s = (\mathbf{c}_s, \dot{\mathbf{c}}_s, \ddot{\mathbf{c}}_s)$ et $\mathbf{x}_g = (\mathbf{c}_g, \dot{\mathbf{c}}_g, \ddot{\mathbf{c}}_g)$. Nous devons donc imposer des contraintes sur la position, la vitesse et l'accélération initiales et finales. Il faut alors un minimum de 6 points de contrôle, et donc $n \geq 5$.

Les positions de ces points de contrôle sont données par les équations suivantes :

- $\mathbf{P}_0 = \mathbf{c}_s$ et $\mathbf{P}_n = \mathbf{c}_g$ garantissent que les positions initiale et finale sont respectées ;
- $\mathbf{P}_1 = \frac{\dot{\mathbf{c}}_s T}{n} + \mathbf{P}_0$ et $\mathbf{P}_{n-1} = \mathbf{P}_n - \frac{\dot{\mathbf{c}}_g T}{n}$ garantissent que les vitesses initiale et finale sont respectées ;
- $\mathbf{P}_2 = \frac{\ddot{\mathbf{c}}_s T^2}{n(n-1)} + 2\mathbf{P}_1 - \mathbf{P}_0$ et $\mathbf{P}_{n-2} = \frac{\ddot{\mathbf{c}}_g T^2}{n(n-1)} + 2\mathbf{P}_{n-1} - \mathbf{P}_n$ garantissent que les accélérations initiale et finale sont respectées.

Les développements menant à ces équations sont détaillés en annexe A.2.

Nous avons alors les valeurs des trois premiers et des trois derniers points de contrôle qui sont contraintes par les conditions aux limites. Leur valeur est donc constante et ne dépend que des états initial et final du problème.

Pour certaines instances particulières du problème, une ou plusieurs de ces contraintes peuvent être supprimées. Par exemple, on peut chercher à rejoindre une position finale sans contraindre la vitesse ou l'accélération atteinte. Un autre cas particulier est celui du *N-Step capturability* où le problème consiste à atteindre une vitesse et une accélération nulle, sans contrainte sur la position finale.

Dans ce chapitre, nous considérons le cas où toutes les contraintes sont actives. Le raisonnement proposé est valide quel que soit le nombre de contraintes : seuls le degré des courbes utilisées et le détail des calculs change.

4.1.2 Reformulation conservatrice mais convexe

Nous rappelons que les contraintes dynamiques appliquées au modèle centroïdal pendant une phase de contact $\{p\}$ sont définies par :

$$\mathbf{H}^{\{p\}} \underbrace{\begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) + \dot{\mathbf{L}} \end{bmatrix}}_{\mathbf{w}} \leq \mathbf{h}^{\{p\}} \tag{4.3}$$

Ces contraintes ne sont pas linéaires à cause du produit vectoriel $\mathbf{c} \times \ddot{\mathbf{c}}$.

Nous allons maintenant reformuler le problème de la faisabilité en appliquant une condition sur $\mathbf{c}(t)$ en la représentant par une courbe de Bézier de degré fixe $n = 6$. Cette condition est conservatrice puisque nous fixons à présent le degré de la courbe. Avec cette reformulation, $\mathbf{c}(t)$ ne couvre plus l'ensemble des solutions possibles, nous discuterons de la perte possible engendrée par cette reformulation dans la section 4.6.2.

En fixant le degré de la courbe ainsi nous avons un seul point de contrôle libre, les six autres étant contraints par les conditions aux limites. Nous notons ce point de contrôle $\mathbf{P}_3 = \mathbf{y}$:

$$\mathbf{c}(t, \mathbf{y}) = \sum_{i \in \{0,1,2,4,5,6\}} B_i^6(t/T)\mathbf{P}_i + B_3^6(t/T)\mathbf{y} \tag{4.4}$$

Avec cette reformulation, $\mathbf{y}$ et T sont maintenant les seules variables du problème. Pour l'instant, nous supposons que la durée T est donnée et nous montrerons comment l'obtenir dans la section 4.5. $\mathbf{y}$ devient alors la seule variable du problème, grâce à cela nous allons exprimer $\mathbf{c} \times \ddot{\mathbf{c}}$ de manière analytique et linéaire. Les développements nécessaires sont décrits dans le reste de cette sous-section.

4.1.2.1 Dérivation d'une courbe de Bézier

La première étape est de calculer l'expression de $\ddot{\mathbf{c}}(t, \mathbf{y})$. Or, une des propriétés des courbes de Bézier est que la dérivée d'une courbe de Bézier de degré n est également une courbe de Bézier, de degré $n - 1$ [Risler 1992]. De plus, l'expression des points de contrôle de la dérivée s'obtient à partir des points de contrôle de la courbe originale via la formule suivante :

$$\mathbf{P}'_i = n(\mathbf{P}_{i+1} - \mathbf{P}_i) \tag{4.5}$$

Avec $\mathbf{P}'_i$ les points de contrôle de $\dot{\mathbf{c}}(t, \mathbf{y})$ et $\mathbf{P}_i$ ceux de $\mathbf{c}(t, \mathbf{y})$.

Il est donc possible de dériver deux fois $\mathbf{c}(t, \mathbf{y})$ et d'obtenir une courbe de Bézier $\ddot{\mathbf{c}}(t, \mathbf{y})$ de degré $n-2$. L'expression de tous les points de contrôle de $\ddot{\mathbf{c}}(t, \mathbf{y})$ peut être calculée de manière analytique. Ils sont soit constants, soit linéairement dépendants de $\mathbf{y}$. On peut donc représenter les points de contrôle $\mathbf{P}''_i$ de $\ddot{\mathbf{c}}(t, \mathbf{y})$ ainsi :

$$\mathbf{P}''_i(\mathbf{y}) = \mathbf{P}''^y_i \mathbf{y} + \mathbf{P}''^s_i \tag{4.6}$$

Avec $\mathbf{P}''^y_i$ et $\mathbf{P}''^s_i$ des constantes calculées analytiquement à partir des valeurs des $\mathbf{P}_i$ et de T. En Annexe A.3, le calcul analytique de ces points de contrôle est développé.

Au final, on peut représenter $\ddot{\mathbf{c}}(t, \mathbf{y})$ comme suit :

$$\ddot{\mathbf{c}}(t, \mathbf{y}) = \sum_{i=0}^{n-2} B_i^{n-2}(t/T)(\mathbf{P}''^y_i \mathbf{y} + \mathbf{P}''^s_i) \tag{4.7}$$

4.1.2.2 Produit vectoriel entre courbes de Bézier

Maintenant que nous avons l'expression de $\mathbf{c}(t,\mathbf{y})$ et de $\ddot{\mathbf{c}}(t,\mathbf{y})$, nous pouvons calculer leur produit vectoriel. Or, l'une des raisons ayant motivé le choix de l'utilisation des courbes de Bézier est que le produit vectoriel entre deux courbes de Bézier de degré n_1 et n_2 est une courbe de Bézier de degré n_1+n_2, la démonstration de cette propriété est présentée en annexe A.4. Dans notre cas particulier du produit vectoriel entre $\mathbf{c}$ et sa dérivée seconde, une simplification supplémentaire peut-être faite et le degré final de la courbe obtenue est $2n-3$.

Encore une fois, l'expression des points de contrôle de $\mathbf{c}(t,\mathbf{y})\times\ddot{\mathbf{c}}(t,\mathbf{y})$ peut être obtenue analytiquement (voir annexe A.5) et ces points de contrôle sont linéairement dépendants de $\mathbf{y}$ puisque le produit vectoriel de $\mathbf{y}$ par lui même s'annule.

Nous pouvons donc représenter $\mathbf{c}(t,\mathbf{y})\times\ddot{\mathbf{c}}(t,\mathbf{y})$ par une courbe de Bézier de degré $2n-3$, dont l'expression analytique des points de contrôle est connue et linéairement dépendante de $\mathbf{y}$.

4.1.2.3 Reformulation convexe du GIW

Nous rappelons que l'expression du GIW $\mathbf{w}(t)$ est :

$$\mathbf{w}(t)=\begin{bmatrix} m(\ddot{\mathbf{c}}-\mathbf{g}) \\ m\mathbf{c}\times(\ddot{\mathbf{c}}-\mathbf{g})+\dot{\mathbf{L}}\end{bmatrix}=\begin{bmatrix} m\ddot{\mathbf{c}}-m\mathbf{g} \\ m\mathbf{c}\times\ddot{\mathbf{c}}+m\hat{\mathbf{g}}\ddot{\mathbf{c}}+\dot{\mathbf{L}}\end{bmatrix} \tag{4.8}$$

Or, nous avons montré que $\mathbf{c}$, $\ddot{\mathbf{c}}$ et $\mathbf{c}\times\ddot{\mathbf{c}}$ sont des courbes de Bézier respectivement de degré n, $n-2$ et $2n-3$. De plus, la somme de courbes de Bézier est une courbe de Bézier de degré égale à celui de la courbe du plus haut degré dans la somme. La multiplication d'une courbe par un scalaire et la somme avec des constantes résulte également en une courbe de Bézier de degré inchangé.

Grâce à toutes ces propriétés, nous pouvons alors exprimer $\mathbf{w}(t)$ comme une courbe de Bézier. Nous rappelons, que nous avons fixé le degré $n=6$, $\mathbf{w}(t)$ est donc de degré $2n-3=9$:

$$\mathbf{w}(t,\mathbf{y})=\sum_{i\in\{0..9\}} B_i^9(t/T)\mathbf{P}_{\mathbf{w}i}(\mathbf{y})+\begin{bmatrix}\mathbf{0}\\ \dot{\mathbf{L}}(t)\end{bmatrix} \tag{4.9}$$

Avec $\mathbf{P}_{\mathbf{w}i}(\mathbf{y})\in\mathbb{R}^6$ les points de contrôle de $\mathbf{w}(t,\mathbf{y})$. L'expression de ces points peut être calculée analytiquement, ils dépendent linéairement de $\mathbf{y}$ et peuvent donc être exprimés ainsi :

$$\mathbf{P}_{\mathbf{w}i}(\mathbf{y})=\mathbf{P}^y_{\mathbf{w}i}\mathbf{y}+\mathbf{P}^s_{\mathbf{w}i} \tag{4.10}$$

Où $\mathbf{P}^y_{\mathbf{w}i}\in\mathbb{R}^{6\times3}$ et $\mathbf{P}^s_{\mathbf{w}i}\in\mathbb{R}^6$ sont des constantes calculées analytiquement. Elles dépendent des deux états initial et final $\mathbf{x}_s=(\mathbf{c}_s,\dot{\mathbf{c}}_s,\ddot{\mathbf{c}}_s)$ et $\mathbf{x}_g=(\mathbf{c}_g,\dot{\mathbf{c}}_g,\ddot{\mathbf{c}}_g)$ ainsi que de T,m et $\mathbf{g}$, comme détaillé dans l'annexe A.6.3.

Dans la suite de ce chapitre, par simplicité, nous supposons que $\dot{\mathbf{L}}(t) = \mathbf{0}$ et nous discutons de cette hypothèse dans la sous-section 4.7.2. Ainsi, grâce à la formulation du GIW de l'équation (4.9) les contraintes dynamiques de l'équation (4.3) peuvent être formulées de manière linéaires par rapport à notre variable $\mathbf{y}$.

Ceci n'est possible que parce que notre reformulation ne laisse qu'un seul point de contrôle variable pour $\mathbf{c}(t)$.

4.2 Test de faisabilité d'une transition de contact

Nous rappelons que le problème de la faisabilité d'une transition de contact consiste à déterminer l'existence d'une trajectoire centroïdale $\mathbf{c}(t)$ connectant exactement deux états $\mathbf{x}_s^{\{p\}}$ et $\mathbf{x}_g^{\{q\}}$ pouvant appartenir à des phases de contact différentes.

Cette trajectoire doit satisfaire les contraintes dynamiques et cinématiques imposées par les phases de contact.

Avec la reformulation présentée dans la section précédente, la variable du problème de faisabilité d'une transition de contact devient maintenant la position $\mathbf{y} \in \mathbb{R}^3$ du point de contrôle de la courbe de Bézier décrivant la trajectoire du centre de masse $\mathbf{c}(t)$.

Grâce à cette reformulation, il devient possible de formuler les contraintes dynamiques appliquées pendant une phase de contact $\{p\}$ (4.3) comme des inégalités linéaires sur $\mathbf{y}$:

$$\mathbf{H}^{\{p\}}\mathbf{w}(t, \mathbf{y}) \leq \mathbf{h}^{\{p\}} \tag{4.11}$$

Avec :

$$\mathbf{w}(t, \mathbf{y}) = \sum_{i=0}^{9} B_i^9(t/T)(\mathbf{P}_{\mathbf{w}i}^y \mathbf{y} + \mathbf{P}_{\mathbf{w}i}^s) \tag{4.12}$$

Nous rappelons que les contraintes cinématiques appliquées pendant une phase de contact $\{p\}$ peuvent être approximées par des inégalités linéaires sur $\mathbf{c}(t)$, comme présenté dans la section 3.3.1 :

$$\mathbf{K}^{\{p\}}\mathbf{c}(t, \mathbf{y}) \leq \mathbf{k}^{\{p\}} \tag{4.13}$$

Avec :

$$\mathbf{c}(t, \mathbf{y}) = \sum_{i \in \{0,1,2,4,5,6\}} B_i^6(t/T)\mathbf{P}_i + B_3^6(t/T)\mathbf{y} \tag{4.14}$$

L'existence d'une trajectoire du centre de masse $\mathbf{c}(t)$ satisfaisant les contraintes imposées par une phase de contact $\{p\}$ peut maintenant être déterminée en résolvant un problème convexe.

4.2.1 Application à un mouvement sans changement de contact

Si l'on considère le cas où $p = q$, le problème de faisabilité revient à trouver une valeur de $\mathbf{y}$ telle que :

$$\begin{aligned} \mathbf{K}^{\{p\}}\mathbf{c}(t,\mathbf{y}) &\leq \mathbf{k}^{\{p\}} \quad , \forall t \in [0;T] \\ \mathbf{H}^{\{p\}}\mathbf{w}(t,\mathbf{y}) &\leq \mathbf{h}^{\{p\}} \quad , \forall t \in [0;T] \end{aligned} \tag{4.15}$$

Afin de vérifier ces contraintes, nous procédons de manière similaire à toutes les méthodes existantes dans la littérature en discrétisant $\mathbf{c}(t,\mathbf{y})$ et $\mathbf{w}(t,\mathbf{y})$ sur $[0;T]$ selon un pas Δt :

$$\begin{aligned} \mathbf{c}(j\Delta t,\mathbf{y}) &= \mathbf{c}_j^y\mathbf{y} + \mathbf{c}_j^s \\ \mathbf{w}(j\Delta t,\mathbf{y}) &= \mathbf{w}_j^y\mathbf{y} + \mathbf{w}_j^s \end{aligned} \tag{4.16}$$

Où $\mathbf{c}_j^y$, $\mathbf{c}_j^s$,$\mathbf{w}_j^y$, $\mathbf{w}_j^s$ sont des constantes dépendantes des deux états initial et final $\mathbf{x}_s = (\mathbf{c}_s, \dot{\mathbf{c}}_s, \ddot{\mathbf{c}}_s)$ et $\mathbf{x}_g = (\mathbf{c}_g, \dot{\mathbf{c}}_g, \ddot{\mathbf{c}}_g)$ ainsi que de T,m ,$\mathbf{g}$ et du temps $j\Delta t$. Avec $j \in J$ l'ensemble défini par $\{j \in \mathbb{N} : 0 \leq j\Delta t \leq T\}$.

Nous pouvons maintenant exprimer les contraintes de l'équation (4.15) de manière discrétisée :

$$\begin{aligned} \mathbf{K}^{\{p\}}\mathbf{c}_j^y\mathbf{y} &\leq \mathbf{k}^{\{p\}} - \mathbf{K}^{\{p\}}\mathbf{c}_j^s \quad , \forall j \in J \\ \mathbf{H}^{\{p\}}\mathbf{w}_j^y\mathbf{y} &\leq \mathbf{h}^{\{p\}} - \mathbf{H}^{\{p\}}\mathbf{w}_j^s \quad , \forall j \in J \end{aligned} \tag{4.17}$$

Que nous pouvons réécrire ainsi :

$$\underbrace{\begin{bmatrix} \mathbf{K}^{\{p\}}\mathbf{c}_j^y \\ \mathbf{H}^{\{p\}}\mathbf{w}_j^y \end{bmatrix}}_{\mathbf{E}_j^{\{p\}}} \mathbf{y} \leq \underbrace{\begin{bmatrix} \mathbf{k}^{\{p\}} - \mathbf{K}^{\{p\}}\mathbf{c}_j^s \\ \mathbf{h}^{\{p\}} - \mathbf{H}^{\{p\}}\mathbf{w}_j^s \end{bmatrix}}_{\mathbf{e}_j^{\{p\}}} , \forall j \in J \tag{4.18}$$

Au final, nous pouvons déterminer l'existence d'une trajectoire $\mathbf{c}(t)$ satisfaisant les contraintes imposées par la phase de contact $\{p\}$ en cherchant s'il existe une valeur de $\mathbf{y} \in \mathbb{R}^3$ satisfaisant les contraintes (4.18). Comme ces contraintes sont des inégalités linéaires, nous pouvons vérifier l'existence d'un tel $\mathbf{y}$ via la formulation d'un LP (Linear Program) :

$$\begin{aligned} \textbf{find} \quad & \mathbf{y} \\ \text{subject to} \quad & \mathbf{E}_j^{\{p\}}\mathbf{y} \leq \mathbf{e}_j^{\{p\}} \quad \forall j \in J \end{aligned} \tag{4.19}$$

Des bibliothèques du commerce nous permettent de résoudre efficacement de type de problème. Si ce LP converge, alors il existe un $\mathbf{y}$ satisfaisant toutes les contraintes. Cela signifie qu'il existe une trajectoire centroïdale $\mathbf{c}(t)$ valide connectant exactement les deux états et phases de contact initial et final $\mathbf{x}_s^{\{p\}}$ et $\mathbf{x}_g^{\{p\}}$ du problème.

4.2.2 Application à un mouvement avec un changement de contact

Si l'on considère le cas où $q = p + 1$, il est nécessaire de connaître le temps de transition entre les phases de contact. Comme expliqué dans la définition du problème, nous considérons pour l'instant ce temps comme connu et nous expliquons dans la section (4.5) comment le déterminer.

Nous pouvons donc définir $T^{\{p\}}$ et $T^{\{q\}}$ le temps passé dans chaque phase de contact, tel que $T = T^{\{p\}} + T^{\{q\}}$. Les contraintes s'expriment alors de la même manière que dans l'équation (4.15) mais avec deux ensembles de contraintes distincts, un pour chaque phase :

$$\begin{aligned}
\mathbf{K}^{\{p\}}\mathbf{c}(t,\mathbf{y}) &\leq \mathbf{k}^{\{p\}} \quad &, \forall t \in [0; T^{\{p\}}] \\
\mathbf{H}^{\{p\}}\mathbf{w}(t,\mathbf{y}) &\leq \mathbf{h}^{\{p\}} \quad &, \forall t \in [0; T^{\{p\}}] \\
\mathbf{K}^{\{q\}}\mathbf{c}(t,\mathbf{y}) &\leq \mathbf{k}^{\{q\}} \quad &, \forall t \in [T^{\{p\}}; T] \\
\mathbf{H}^{\{q\}}\mathbf{w}(t,\mathbf{y}) &\leq \mathbf{h}^{\{q\}} \quad &, \forall t \in [T^{\{p\}}; T]
\end{aligned} \tag{4.20}$$

A l'instant de transition $t = T^{\{p\}}$, les contraintes des deux phases doivent être appliquées. Nous définissons maintenant les ensembles $J^{\{p\}}$ et $J^{\{q\}}$ tel que :

$$\begin{aligned}
J^{\{p\}} &: \{j \in \mathbb{N}, 0 \leq j\Delta t \leq T^{\{p\}}\} \\
J^{\{q\}} &: \{j \in \mathbb{N}, T^{\{p\}} \leq j\Delta t \leq T\}
\end{aligned} \tag{4.21}$$

Qui nous permettent de représenter les contraintes de manière discrétisée, comme dans la sous-section précédente :

$$\begin{aligned}
\mathbf{K}^{\{p\}}\mathbf{c}_j^y\mathbf{y} &\leq \mathbf{k}^{\{p\}} - \mathbf{K}^{\{p\}}\mathbf{c}_j^s \quad &, \forall j \in J^{\{p\}} \\
\mathbf{H}^{\{p\}}\mathbf{w}_j^y\mathbf{y} &\leq \mathbf{h}^{\{p\}} - \mathbf{H}^{\{p\}}\mathbf{w}_j^s \quad &, \forall j \in J^{\{p\}} \\
\mathbf{K}^{\{q\}}\mathbf{c}_j^y\mathbf{y} &\leq \mathbf{k}^{\{q\}} - \mathbf{K}^{\{q\}}\mathbf{c}_j^s \quad &, \forall j \in J^{\{q\}} \\
\mathbf{H}^{\{q\}}\mathbf{w}_j^y\mathbf{y} &\leq \mathbf{h}^{\{q\}} - \mathbf{H}^{\{q\}}\mathbf{w}_j^s \quad &, \forall j \in J^{\{q\}}
\end{aligned} \tag{4.22}$$

Au final, nous pouvons à nouveau formuler un LP permettant de vérifier l'existence d'une position pour le point de contrôle $\mathbf{y}$ satisfaisant les contraintes imposées par chacune des phases de contact sur la portion de la trajectoire concernée :

$$\begin{aligned}
&\textbf{find} \quad \mathbf{y} \\
&\text{subject to} \quad \mathbf{E}_j^{\{z\}}\mathbf{y} \leq \mathbf{e}_j^{\{z\}} \quad , \forall j \in J^{\{z\}}, \forall z \in \{p, q\}
\end{aligned} \tag{4.23}$$

Donc d'après notre définition du problème de la faisabilité d'une transition de contact, si ce LP converge la transition de contact entre $\mathbf{x}_s^{\{p\}}$ et $\mathbf{x}_g^{\{q\}}$ est **faisable**.

4.2.3 Application dans le cas général

Si l'on considère le cas général où $q > p$, nous pouvons ajouter de nouvelles phases de contact au problème, en définissant à chaque fois leur durée $T^{\{z\}}$ et l'ensemble $J^{\{z\}}$. Le LP dans le cas général s'écrit de la manière suivante :

$$\begin{aligned} \textbf{find} \quad & \mathbf{y} \\ \text{subject to} \quad & \mathbf{E}_j^{\{z\}}\mathbf{y} \leq \mathbf{e}_j^{\{z\}} \quad , \forall j \in J^{\{z\}}, \forall z \in \{p, \ldots, q\} \end{aligned} \tag{4.24}$$

Cependant, il faut noter qu'avec notre reformulation la courbe représentant la trajectoire du centre de masse n'a qu'un seul point de contrôle variable qui peut agir sur la forme de la courbe. De ce fait, la courbe obtenue ne peut avoir qu'un seul point d'inflexion au maximum (en plus des inflexions imposées par les contraintes du problème). Cela signifie que plus on considère de phases de contact dans un même problème, plus notre reformulation est conservatrice.

Dans notre cas, nous nous intéressons à déterminer la faisabilité d'une transition de contact. Les problèmes que nous formulons ont donc deux phases de contact dans le cas d'une création ou d'une suppression de contact ou trois phases de contact dans le cas d'un repositionnement de contact (le contact est alors supprimé puis créé à une autre position après une certaine durée).

Nous résolvons un nouveau problème pour chaque transition de contact à tester.

4.2.4 Contraintes additionnelles et fonction de coût

La formulation proposée permet d'ajouter aisément des bornes fixes sur $\mathbf{c}(t)$ ou n'importe laquelle de ses dérivées, pour une ou plusieurs phases de contact. En effet, il suffit de discrétiser la courbe comme fait dans l'équation (4.16), l'expression de ces points discrétisés étant toujours linéairement dépendante de $\mathbf{y}$ (ou constante). Ensuite, on peut ajouter comme contraintes les bornes désirées et d'obtenir des inégalités sur la position de $\mathbf{y}$.

Nous notons ces contraintes de façon générique, pour une phase de contact $\{z\}$, par :

$$\mathbf{O}_j^{\{z\}}\mathbf{y} \leq \mathbf{o}_j^{\{z\}} \tag{4.25}$$

Dans notre implémentation nous utilisons des bornes sur la vitesse et l'accélération du centre de masse.

Bien que vérifier l'existence d'un $\mathbf{y}$ respectant toutes les contraintes suffit à résoudre le problème de la faisabilité d'une transition, il peut être intéressant de générer une trajectoire qui optimise une certaine fonction de coût $l(\mathbf{y})$. Cette trajectoire pourra, entre autre, être utilisée comme initialisation d'une méthode de résolution non linéaire comme présenté dans la section 4.6.4. Des détails sur les différentes fonctions de coût seront donnés dans la section 4.4.

Nous pouvons finalement reformuler le LP de l'équation (4.24) en ajoutant la fonction de coût et les contraintes additionnelles :

$$\begin{aligned} \textbf{find} \quad & \mathbf{y} \\ \textbf{minimize} \quad & l(\mathbf{y}) \\ \text{subject to} \quad & \forall j \in J^{\{z\}}, \forall z \in \{p, \dots, q\} : \\ & \mathbf{E}_j^{\{z\}} \mathbf{y} \leq \mathbf{e}_j^{\{z\}} \\ & \mathbf{O}_j^{\{z\}} \mathbf{y} \leq \mathbf{o}_j^{\{z\}} \end{aligned} \tag{4.26}$$

Dans le reste de ce chapitre, nous appelons *CROC* (Convex Resolution Of Centroïdal dynamic trajectories) la méthode consistant à formuler puis résoudre le LP (4.26).

4.3 Reformulation du problème en force

Dans la section précédente, nous avons reformulé le problème de la faisabilité d'une transition de contact de manière linéaire en nous basant sur la formulation des contraintes dynamiques par des inégalités, autrement dit la méthode "double description".

Dans cette section nous montrons que le même développement peut être fait avec la formulation des contraintes par égalité (aussi appelée formulation en force).

Nous rappelons qu'avec cette formulation les contraintes dynamiques s'écrivent de la manière suivante (voir section 3.4.1 pour les développements détaillés) :

$$\underbrace{\begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) + \dot{\mathbf{L}} \end{bmatrix}}_{\mathbf{w}} = \underbrace{\begin{bmatrix} \mathbf{I}_3 & \dots & \mathbf{I}_3 \\ \hat{\mathbf{p}}_1 & \dots & \hat{\mathbf{p}}_k \end{bmatrix}}_{\mathbf{A}^{\{p\}}} \mathbf{V}\boldsymbol{\beta} \ , \ \boldsymbol{\beta} \geq 0 \tag{4.27}$$

Nous rappelons que $\mathbf{V}$ ne dépend que des normales de contact et des coefficients de friction, donc $\mathbf{A}^{\{p\}}$ dépend uniquement de la phase de contact. $\boldsymbol{\beta} \in \mathbb{R}^{4k}$ est une variable.

Avec la reformulation proposée dans la section précédente cette égalité peut être réécrite pour chaque point discrétisé :

$$\mathbf{w}_j^y \mathbf{y} + \mathbf{w}_j^s = \mathbf{A}^{\{p\}} \boldsymbol{\beta}_j \ , \ \boldsymbol{\beta}_j \geq 0 \ , \ \forall j \in J^{\{p\}} \tag{4.28}$$

Avec cette formulation par égalité, les variables du problème deviennent alors $\mathbf{y}$ et un vecteur $\boldsymbol{\beta}_j \in \mathbb{R}^{4k}$ pour chaque point discrétisé :

$$\boldsymbol{\lambda} = \begin{bmatrix} \mathbf{y} \\ \boldsymbol{\beta}_0 \\ \vdots \\ \boldsymbol{\beta}_{n_d} \end{bmatrix} \tag{4.29}$$

Avec $\boldsymbol{\lambda}$ la nouvelle variable de notre problème et n_d le nombre de points de discrétisation tel que $(n_d \Delta t) = T$.

Finalement, le LP (4.24) peut se reformuler ainsi :

$$\begin{aligned} \textbf{find} \quad & \boldsymbol{\lambda} \\ \text{subject to} \quad & \forall j \in J^{\{z\}}, \forall z \in \{p, \dots, q\} : \\ & \mathbf{K}^{\{z\}} \mathbf{c}_j^y \mathbf{y} \leq \mathbf{k}^{\{z\}} - \mathbf{K}^{\{z\}} \mathbf{c}_j^s \\ & \boldsymbol{\beta}_j \geq 0 \\ & \mathbf{w}_j^y \mathbf{y} = \mathbf{A}^{\{z\}} \boldsymbol{\beta}_j - \mathbf{w}_j^s \end{aligned} \tag{4.30}$$

4.3.1 Dimension du problème

La différence entre les deux formulations est donc dans la dimension du problème. Dans le cas de la formulation en force la variable du problème est de taille $\lambda \in \mathbb{R}^{3+(4k)n_d}$. De plus, $6n_d$ égalités sont ajoutées aux contraintes.

Par contre, le nombre de contraintes d'inégalités est réduit puisque les inégalités $\mathbf{Hw} \leq \mathbf{h}$ n'apparaissent plus.

Bien que ce problème soit de grande dimension, les matrices de contrainte et de variables du problème (4.30) sont creuses (c'est à dire majoritairement composée de 0). Il existe une bibliothèque libre du commerce qui permet d'exploiter ce genre de structure [1] et ainsi de résoudre efficacement ce LP en des temps de calcul raisonnables.

4.4 Fonction de coût

Les formulations de LP proposées dans les deux sections précédentes permettent seulement d'exprimer une fonction de coût linéaire, de la forme suivante :

$$l(\mathbf{y}) = \mathbf{By} + \mathbf{b} \tag{4.31}$$

Avec une augmentation minime du temps de calcul requis, nous pouvons reformuler le problème LP en un QP (Quadratic Programming) et utiliser une fonction de coût quadratique permettant plus de possibilités dans le choix des caractéristiques du mouvement à optimiser. Des exemples de fonctions de coût quadratique pourraient par exemple minimiser la norme de la distance parcourue, la vitesse ou le contrôle (l'accélération).

Cependant, comme expliqué dans la sous-section précédente, il n'existe pas de bibliothèque libre permettant de résoudre des problèmes QP tout en exploitant la structure creuse des différentes matrices du problème. A cause de cela, le LP de la formulation en force (4.30) ne peut pas être transformé en QP sans une augmentation très importante du temps de calcul nécessaire. L'utilisation d'une

1. GLPK : `https://www.gnu.org/software/glpk/`

fonction de coût quadratique sera donc reservée à la formulation par inégalités utilisant la méthode de double description.

Dans notre implémentation nous avons choisi de minimiser l'intégrale de la norme au carré de l'accélération. Comme l'accélération $\ddot{\mathbf{c}}(t)$ est représentée par une courbe de Bézier dont nous connaissons l'expression analytique des points de contrôle, linéairement dépendants de $\mathbf{y}$, il est possible d'écrire ce coût de manière continue et exacte.

En effet, ce coût s'écrit de la manière suivante :

$$l(\mathbf{y}) = \int_0^T ||\ddot{\mathbf{c}}(t, \mathbf{y})||^2 dt \tag{4.32}$$

Or, dans notre cas il est possible de calculer la courbe $\ddot{\mathbf{c}}^2$ puis sa primitive et enfin l'expression de sa primitive aux extrémités, le tout analytiquement comme détaillé en annexe A.7. Nous pouvons ensuite représenter le coût sous la forme quadratique classique :

$$l(\mathbf{y}) = \mathbf{y}^\top \mathbf{A}\mathbf{y} + 2\mathbf{b}^\top \mathbf{y} \tag{4.33}$$

Avec $\mathbf{A} \in \mathbb{R}^{3\times3}$ et $\mathbf{b} \in \mathbb{R}^3$ une matrice et un vecteur qui ne dépendent que des positions des points de contrôle constants $\mathbf{P}_i$ de la courbe $\mathbf{c}(t, \mathbf{y})$.

D'autres coûts peuvent être définis pour agir sur le *style* du mouvement. En effet, la forme de la trajectoire centroïdale générée a une influence direct sur l'esthétique du mouvement corps complet généré en suivant cette trajectoire, comme montré sur la figure 4.1. Dans le domaine de l'animation graphique être capable de faire varier le style du mouvement, et d'avoir un style consistant sur plusieurs mouvements différents, est une propriété très intéressante.

Les figures 4.2, 4.3, 4.4 et 4.5 montrent des exemples de coûts simples donnant lieu à des mouvements à l'esthétique très différente alors que la même séquence de contacts est utilisée.

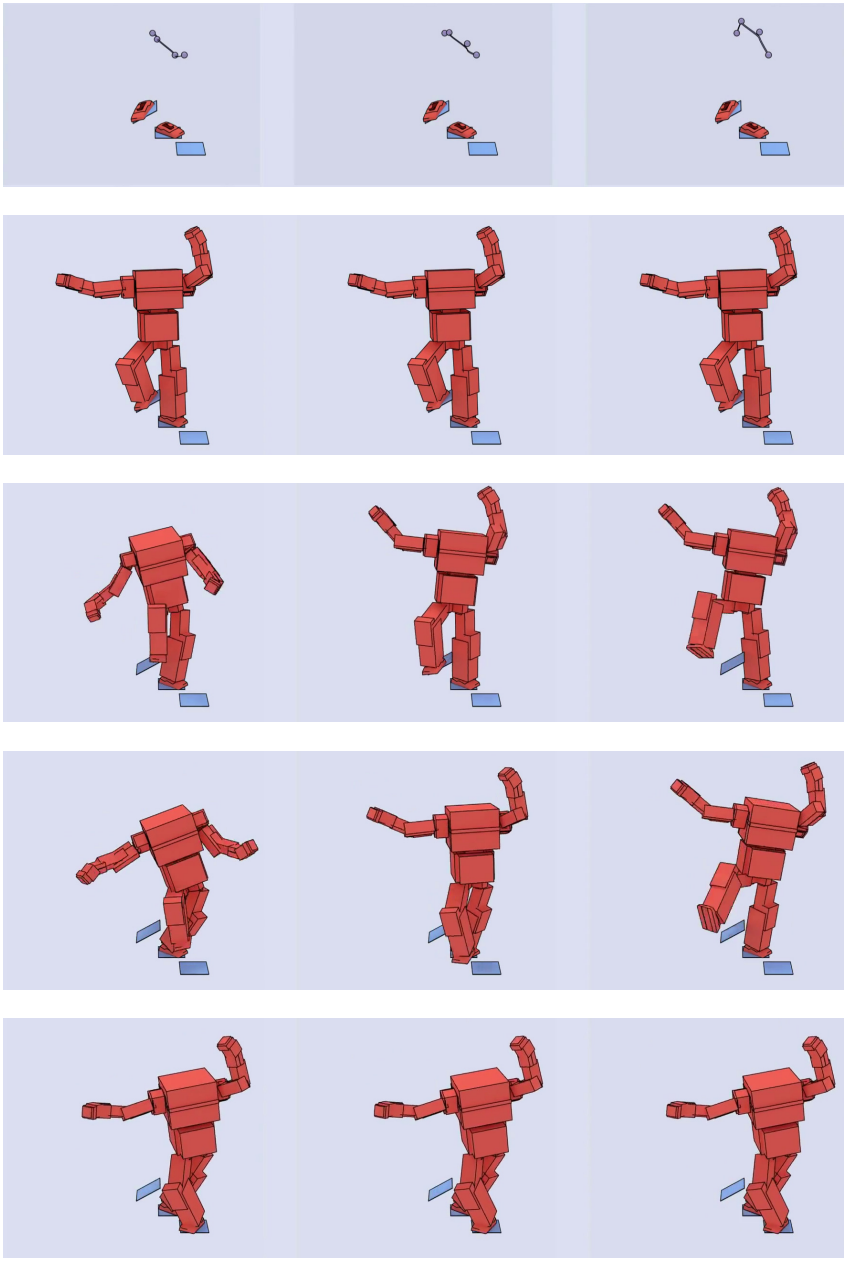


FIGURE 4.1 – Figure provenant de [Tonneau 2018b]. Exemple de trois mouvements générés avec la même séquence de contacts mais une trajectoire centroïdale différente : de gauche à droite la trajectoire est de plus en plus haute (vers les z positifs). La première ligne montre les trajectoires centroïdales. Les lignes suivantes montrent chacun des trois mouvements au même instant (la durée totale des trois mouvements est identique).

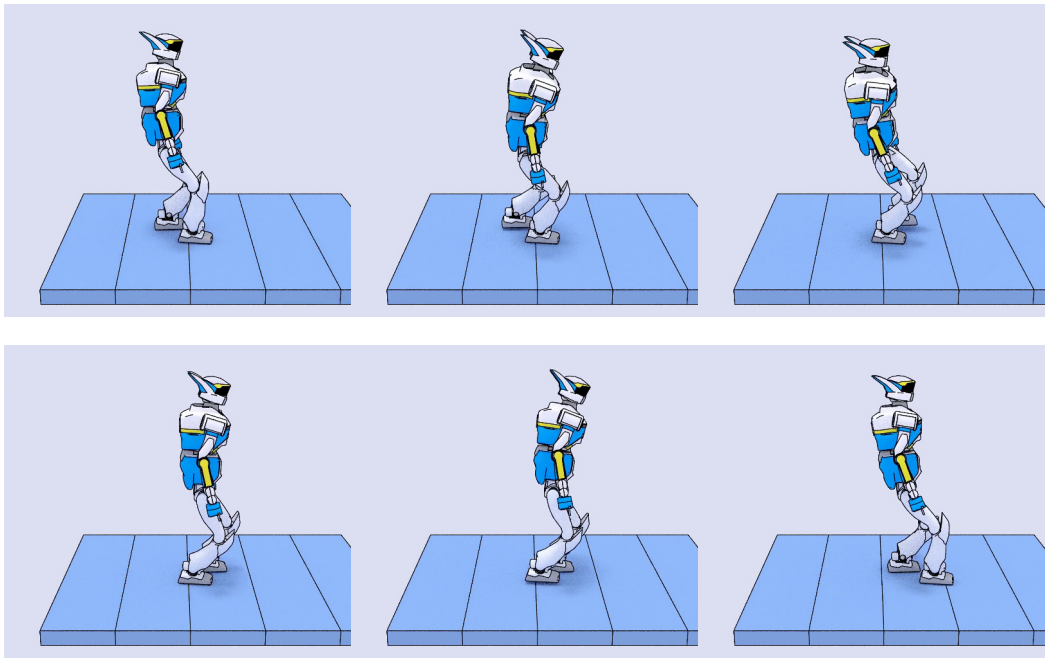


FIGURE 4.2 – Mouvement avec un style "fatigué" : généré avec une fonction de coût minimisant la distance parcourue par le centre de masse.

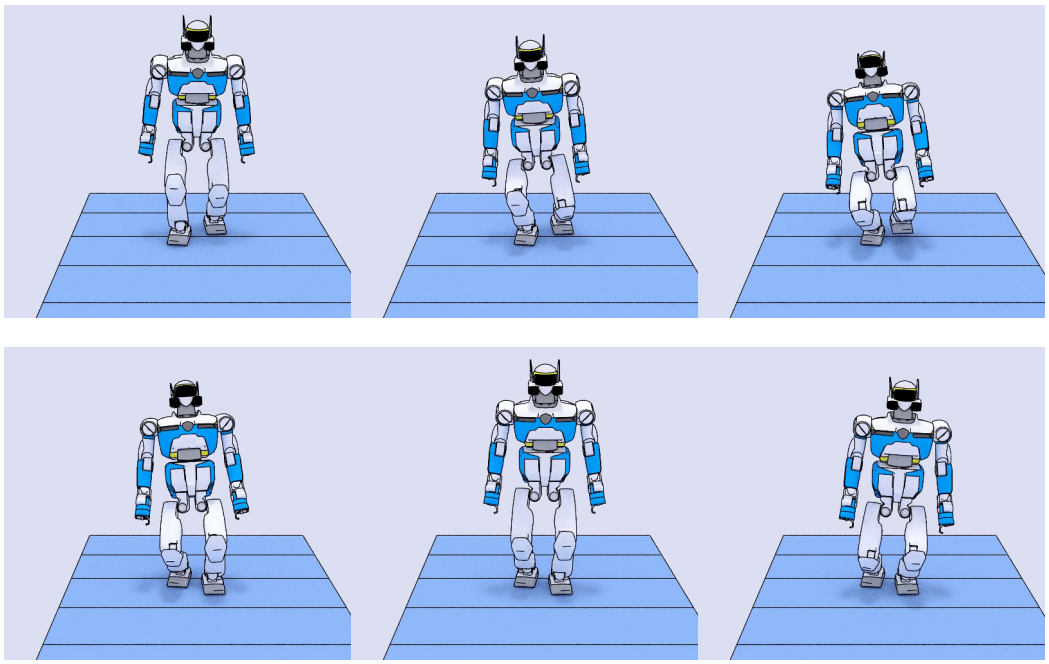


FIGURE 4.3 – Mouvement avec un style "groovy" : généré avec une fonction de coût maximisant le déplacement latéral.

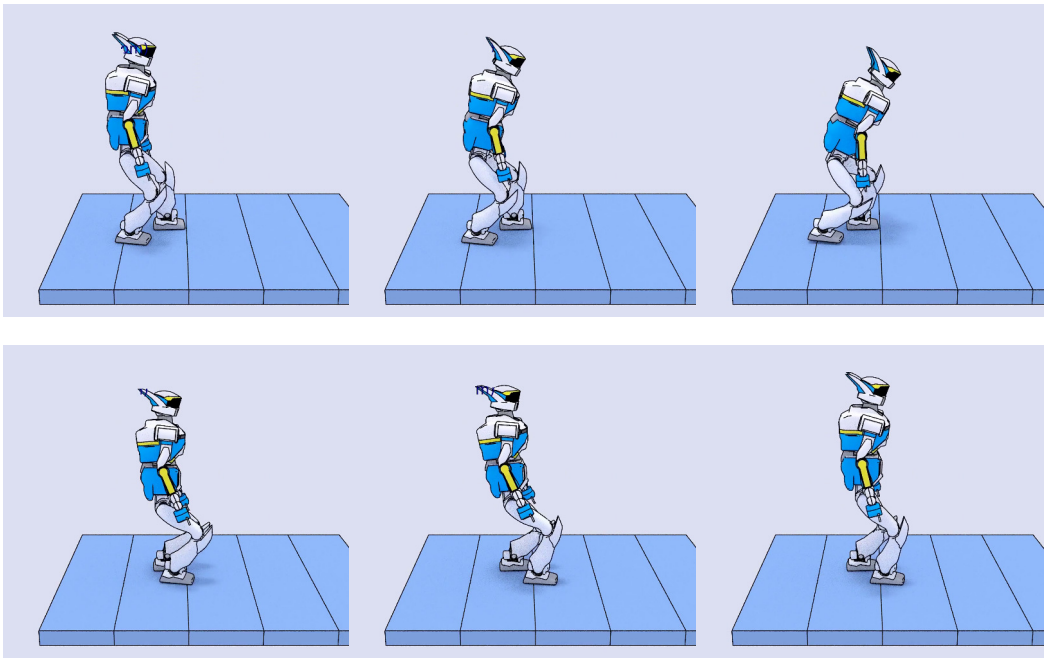


FIGURE 4.4 – Mouvement avec un style "accroupi" : généré avec une fonction de coût minimisant la hauteur du point de contrôle variable.

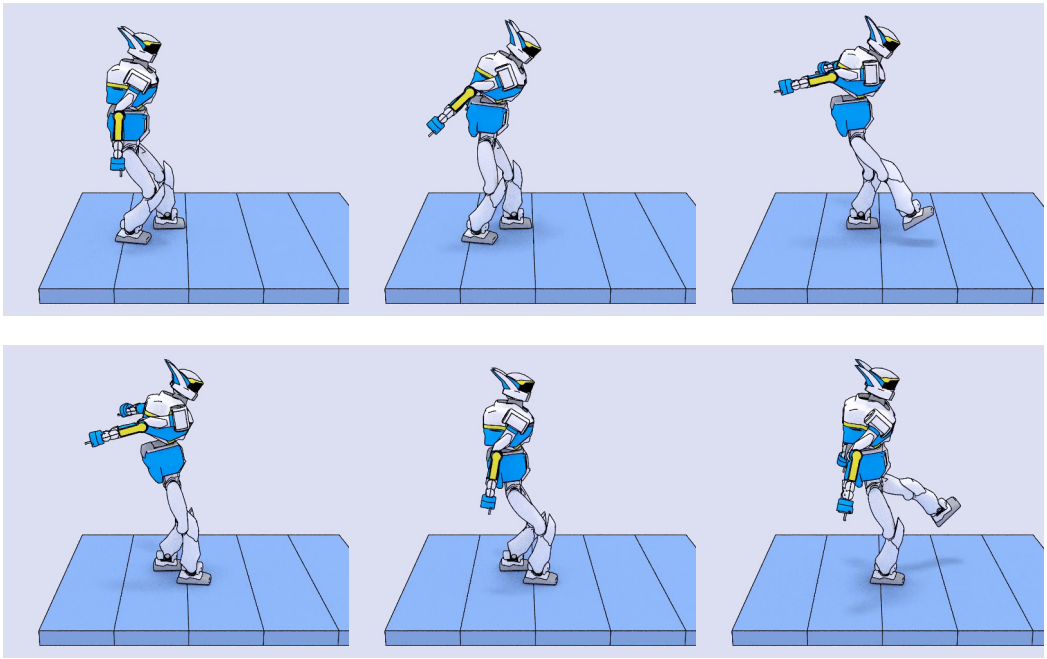


FIGURE 4.5 – Mouvement avec un style "joueur" : généré avec une fonction de coût plaçant le point de contrôle variable le plus en arrière possible.

4.5 Échantillonnage du temps de transition

Dans le début de ce chapitre, nous avons supposé que la durée totale de la trajectoire T ainsi que les durées de chaque phase de contact $T^{\{p\}}$ étaient connues. Cette hypothèse était nécessaire pour demeurer convexe, mais il faut à présent une manière de déterminer ces durées.

Afin de conserver un problème convexe et exact sans relaxation des contraintes, contrairement à [Ponton 2018], nous avons fait le choix d'échantillonner ces durées sur un intervalle de temps jusqu'à trouver une solution. En théorie, afin de rester complet, il faudrait échantillonner un nombre infini de combinaisons.

Nous avons préféré choisir une approche pragmatique, ne garantissant plus la complétude théorique mais offrant le même taux de succès empiriquement, en procédant comme suit. Nous avons échantillonné des temps pour des séquences de trois phases de contact $T^{\{p\}}, T^{\{p+1\}}, T^{\{p+2\}}$ dans un intervalle de 0.1 à 2 secondes pour les phases sans mouvement des pattes et entre 0.5 à 2 secondes pour les phases avec un mouvements des pattes, avec un incrément de 50ms. Sur trois phases de contact, cela donne un total de 43320 combinaisons de durées possibles.

Nous avons alors testé CROC avec toutes ces combinaisons sur des problèmes divers : avec HRP-2 ou HyQ sur des scénarios sur sol plat ou en multi-contact. Cela pour des milliers de séquences de trois phases de contact.

L'analyse des taux de convergence en fonction des combinaisons de durées utilisées a révélé un résultat intéressant : une petite liste de combinaisons (5 dans notre cas, présentés dans le tableau 4.1) couvre exactement 100% des succès obtenues pendant nos tests. Autrement dit, lorsque CROC a convergé avec au moins une des 43320 combinaisons de durées, au moins une des 5 combinaisons de cette liste fait partie des combinaisons permettant à CROC de converger sur cette instance du problème.

Durées (s)			Taux de succès (%)
$T^{\{p\}}$	$T^{\{p+1\}}$	$T^{\{p+2\}}$	
1	0.8	0.8	91.2
1	0.75	0.9	89.2
0.8	0.8	0.9	88.3
0.7	0.5	0.85	77.7
1.2	0.6	1.1	70.8

Tableau 4.1 – Taux de succès obtenus avec chacune des cinq combinaisons de durées retenus. Ici les 100% de taux de succès correspondent à l'utilisation des 43320 combinaisons possibles.

Comme montré dans le tableau 4.1, aucune de ces combinaisons prises indépendamment ne permet d'atteindre les 100% de taux de succès. Mais comme montré par la figure 4.6, lorsque l'on teste les cinq, nous obtenons 100% de taux de succès comme si l'on avait testé les 43320 combinaisons.

La figure 4.6 montre qu'à partir de 3 combinaisons testé nous dépassons les 99% de succès. Il serait donc envisageable de ne tester que ces 3 combinaisons pour une perte minime de solutions.

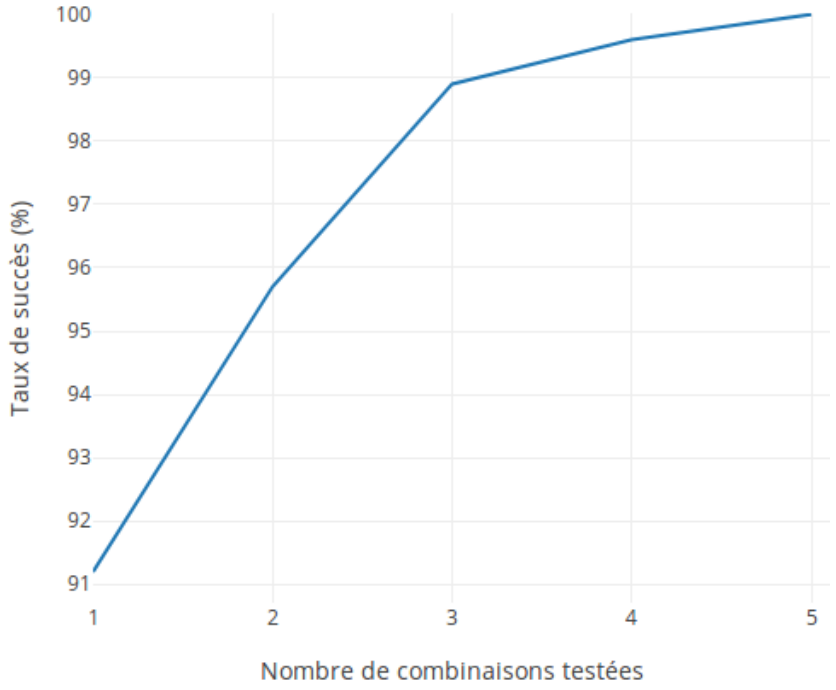


FIGURE 4.6 – Evolution du taux de succès en fonction du nombre de combinaisons de durées testées. Ici les 100% de taux de succès correspondent à l'utilisation des 34200 combinaisons possibles.

Au final, il suffit donc de tester au maximum 5 QPs, avec cette liste de combinaisons de durées, pour chaque problème afin de déterminer s'il existe une solution.

4.6 Résultats

Dans cette section nous nous intéressons à mesurer les performance de la méthode CROC proposée dans ce chapitre. De par la reformulation utilisée, nous savons que notre méthode est conservatrice puisque notre représentation de la trajectoire centroïdale par une courbe de Bézier de degré fixé ne peut couvrir l'ensemble des solutions au problème.

Cependant, déterminer le taux de succès de notre méthode pour la résolution du problème de la faisabilité d'une transition de contact n'est pas aisé car, à ce jour, il n'y a aucune méthode permettant de déterminer avec exactitude si une transition de contact est faisable. En effet, bien qu'il existe plusieurs méthodes de génération de trajectoire centroïdale dans la littérature aucune n'est complète et ne

garantit la convergence car elles utilisent soit des approximations soit des méthodes de résolution non linéaires.

Nous n'avons donc pas de "vérité terrain" permettant de calculer le taux de succès d'une méthode.

En réalité, nous ne nous intéressons pas exactement au taux de succès de notre méthode par rapport à la "vérité terrain" mais par rapport au taux de succès de la méthode de génération de trajectoire centroïdale que nous allons utiliser.

En effet, comme expliqué dans le chapitre 3, l'intérêt de proposer une méthode pour vérifier la faisabilité d'une transition de contact est de garantir que la méthode de génération de trajectoire centroïdale utilisée ensuite va converger avec la séquence de contacts fournie. Nous nous intéressons donc à comparer le taux de succès de CROC par rapport au taux de succès de la méthode de résolution non linéaire utilisée par la méthode de génération de trajectoire centroïdale.

Dans notre architecture, nous utilisons la méthode de génération de trajectoire proposée dans [Carpentier 2017c], qui sera décrite dans la partie suivante.

Note sur la terminologie employée :

Dans le reste de cette section, nous employons le terme *nombre de contact* pour désigner le nombre de pattes (jambe, bras, ...) en contact. Or, nous considérons ici des contacts rectangulaires, ce qui signifie que chaque patte en contact génère 4 points de contact : un à chaque sommet de l'organe terminal en contact (pied, main, ...).

4.6.1 Protocole expérimental

Afin de mesurer les performances de notre méthode, nous avons généré aléatoirement des paires d'états associés à des phases de contact tel que :

— Les deux états sont en équilibre statique et sont cinématiquement valides ;
— Les deux états ont les même pattes en contact ;
— Il y a exactement UN repositionnement de contact entre les deux états, et pas d'autre variation de contact.

A partir d'une telle paire d'états nous pouvons donc générer une séquence de trois phases de contact en introduisant la phase intermédiaire où le contact qui va être repositionné est supprimé. Il n'y a aucune garantie, ni contrainte, qu'un état en équilibre quasi-statique existe pour cette phase intermédiaire.

Nous avons utilisé deux méthodes d'échantillonnage aléatoire différentes. La première génère des contacts coplanaires. La seconde génère des contacts véritablement aléatoires, ce qui résulte en des contacts non coplanaires et des transitions de contact qui peuvent demander des mouvements très complexes.

Pour chaque séquence de trois phases de contact, nous testons ensuite la méthode CROC (avec la formulation en force ou par double description des

contraintes dynamiques) et la méthode de génération de trajectoire centroïdale de [Carpentier 2017c]. Comme toutes les méthodes non linéaires, cette dernière méthode requiert une initialisation avec une solution initiale. Dans [Carpentier 2017c] les auteurs proposent une manière de calculer cette solution initiale à partir de conditions géométriques sur les positions des points de contact. Nous utilisons cette même initialisation, mais si CROC converge nous testons également la méthode non linéaire en lui donnant comme solution initiale la solution trouvée par CROC.

La "vérité terrain" considérée pour le calcul des taux de succès est donc la méthode de résolution non linéaire [Carpentier 2017c], initialisée avec la solution de CROC quand elle est disponible, avec la solution initiale naïve sinon.

Pas de discrétisation :

Dans notre implémentation, nous avons choisi d'utiliser un pas de discrétisation variable mais un nombre de points de discrétisation fixe. Ce choix a été fait afin de conserver la taille du problème fixe, et donc son temps de résolution, indépendamment de la durée de la trajectoire.

Dans les résultats présentés dans le reste de cette section, nous avons choisi d'utiliser 7 points de discrétisation par phases de contact, pour la méthode CROC et celle de génération de trajectoire centroïdale de [Carpentier 2017c]. Selon la durée de la trajectoire considérée, le pas de discrétisation est alors compris entre 50 et 200 millisecondes, ce qui est comparable aux pas utilisés dans la littérature [Dai 2014, Ponton 2016].

Implémentation :

L'implémentation de ces problèmes a été faite en C++ [2], afin de résoudre les problèmes LP nous utilisons la bibliothèque GLPK et pour résoudre les problèmes QP nous utilisons QUADPROG.

Toutes les mesures de performances ont été réalisées sur un ordinateur avec un processeur Intel Xeon CPU E5-1630 v3 à 3.7Ghz et 64Go de mémoire vive. Aucun calcul n'est parallélisé et un seul coeur du processeur est utilisé.

4.6.2 A quel point est-on conservateur ?

Comme attendu, les résultats du tableau 4.2 montrent que notre méthode trouve moins de solutions que la méthode de résolution non linéaire. Cela est dû au fait que notre représentation de la trajectoire centroïdale par une courbe de Bézier de degré fixé ne permet pas de couvrir l'ensemble de l'espace des solutions.

Cependant, nous sommes tout de même capable de trouver plus de 60% des solutions dans le pire cas, et cela pour un temps de calcul requis bien inférieur.

2. `https://gitlab.com/stonneau/bezier_COM_traj`

Méthode	**Coplanaire** succès (%)	**Non-coplanaire** succès (%)	Temps LP/QP (ms)	Temps total (ms)
CROC (double description)	89.7	60.6	0.049	3.93
CROC (force)	89.7	60.6	7.51	13.01
OCP avec init. naïve	100	94.1	-	$\simeq 150$
OCP avec init. de CROC	100	100	-	$\simeq 130$

Tableau 4.2 – Comparaison entre la méthode CROC proposée dans ce chapitre et la méthode de résolution non-linéaire proposée dans [Carpentier 2017c], avec une initialisation par une solution naïve ou par la solution trouvée par CROC quand elle est disponible. La comparaison des taux de succès est faite pour des scénarios avec des contacts coplanaires ou non-coplanaires.

La grande différence de taux de succès de la méthode CROC entre les scénarios coplanaires et non-coplanaires s'explique par le fait que pour ces derniers une trajectoire centroïdal complexe, avec parfois plusieurs changement de directions, peut être nécessaire pour pouvoir résoudre le problème. Comme la forme de la trajectoire centroïdale est contrainte avec la représentation utilisée par CROC, notre méthode peut ne pas trouver de solution dans ces cas puisqu'elle ne couvre pas tout l'espace des solutions.

4.6.3 Analyse de performance

Notre méthode CROC est entre un et deux ordres de grandeur plus rapide que la méthode de génération de trajectoire. Grâce à cette efficacité, il devient réaliste d'utiliser notre méthode durant la planification de contact.

Dans le cas de la formulation par double description, on peut noter que le temps nécessaire pour résoudre le LP de l'équation (4.26) est extrêmement faible ($\sim 50\mu s$), le reste du temps est utilisé pour calculer les différentes matrices de contraintes ($\sim 0.4ms$) et la méthode de double description elle même ($\sim 3.5ms$).

Dans les deux cas, une partie non négligeable du temps total est passée dans le calcul des matrices de contraintes. Ce temps est principalement pris par des multiplications matricielles de grande dimension. L'implémentation actuelle pourrait certainement être optimisée pour réduire grandement cette portion du temps de calcul. Notamment en éliminant de possibles contraintes redondantes afin de réduire la dimensions des matrices.

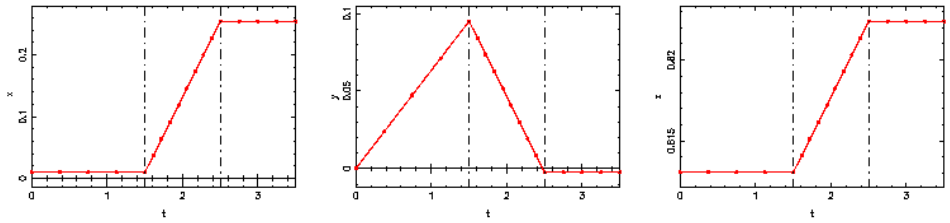
4.6.4 Initialisation d'une méthode de résolution non linéaire

Les méthodes de résolution non linéaires nécessitent d'être initialisées avec une solution initiale. Le choix de cette solution initiale peut changer drastiquement les performances de la méthode de résolution non linéaire, ce choix peut être complexe dans les cas de mouvements en multi-contact.

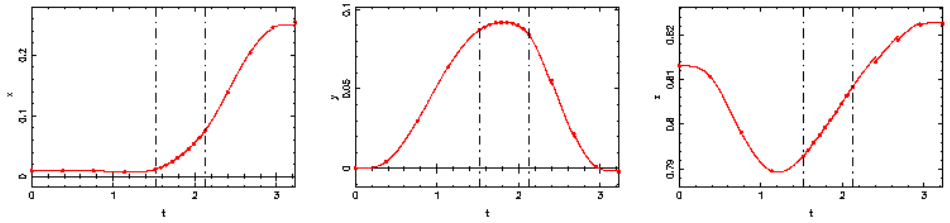
La figure 4.7-a montré un exemple d'initialisation naïve proposée dans [Carpentier 2017c]. Cette solution initiale est composée de segments de droite pas-

sant par le centre des polygones de support de chaque phase de contact, à une hauteur selon l'axe z prédéfinie. Avec une vitesse constante par phase de contact permettant de suivre cette trajectoire. Cette solution est donc valide seulement dans le cas de contacts coplanaires avec un sol plat ou presque plat, de plus l'accélération requise pour cette solution peut être très importante.

La figure 4.7-b montre une trajectoire trouvée par CROC, donnée comme solution initiale à la méthode de génération de trajectoires.



(a) solution initiale naïve



(b) solution initiale fournie par CROC

FIGURE 4.7 – Exemple de solutions initiales données à la méthode de résolution non linéaire, dans le cas d'un pas de 50cm sur sol plat. Les colonnes représentent les axes (x, y, z), les traits pointillées verticaux représentent les changements de phases de contact. Les points rouges sont les points de discrétisation de la méthode de génération de trajectoire.

L'analyse des résultats obtenue suggère un fait intéressant : l'ensemble des solutions trouvées par CROC n'est pas strictement inclue dans l'ensemble des solutions trouvées par la méthode non linéaire avec une initialisation naïve. Mais lorsque l'on utilise cette solution trouvée par CROC comme solution initiale de la méthode non-linéaire, cette dernière converge bien.

Comme montré dans le tableau 4.2, l'utilisation de la solution trouvée par CROC pour initialiser la méthode de résolution non linéaire permet un gain à la fois en terme de taux de succès et de temps de calcul. En effet, on constate une réduction d'environ 20% du temps de calcul requis lorsque l'on initialise la méthode de résolution non linéaire avec la solution de CROC.

Cette augmentation des taux de succès n'apparaît que pour les scénarios avec des contacts non-coplanaires, puisque dans le cas coplanaire la solution initiale naïve proposée dans [Carpentier 2017c] est toujours proche d'une solution valide.

L'initialisation avec la solution fournie par CROC n'a par contre pas d'impact sur la solution finale et optimale obtenue par la méthode de génération de trajectoire quand elle converge.

4.6.5 Comparaison entre la formulation en force et par la double description

Formulation	Temps de calcul (ms)	Nombre de contacts		
		2	3	4
Double Description	Double Description	3.56	14.88	28.16
	QP	0.049	0.092	0.158
	Total	3.93	16.18	31.41
Force	LP	7.51	13.94	26.01
	Total	13.01	25.28	47.65

Tableau 4.3 – Comparaison entre la formulation en force et par la double description. Nombre de contacts désigne le nombre de pattes en contact, chacune générant 4 points de contact.

Comme expliqué dans la section 4.4, nous n'avons implémenté qu'un LP pour la formulation en force et non un QP, contrairement à la formulation avec la double description.

Comme présenté dans le tableau 4.3, la formulation utilisant la double description semble la plus rapide pour un nombre de contact compris entre 2 et 4. La résolution du QP de l'équation (4.26) est extrêmement rapide dans ce cas car ce QP est de faible dimensions, contrairement au LP de l'équation (4.30) de la formulation en force.

Cependant, dans le cas de l'utilisation de la méthode de double description, on peut noter qu'environ 85% du temps total est passé dans la méthode de double description elle même.

A mesure que le nombre de contacts augmente, l'écart entre les deux formulations diminue. En effet, le temps de calcul requis par la double description dépend de manière cubique du nombre de contacts, alors que celui de la formulation en force double pour chaque contact additionnel. De plus, comme mentionné dans le chapitre 3.4.2, la méthode de double description est instable numériquement et peut échouer occasionnellement. Pour toutes ces raisons, nous serions plutôt enclins à utiliser la formulation en force malgré le fait qu'elle soit plus lente pour un faible nombre de contacts.

Cependant, dans notre cas nous souhaitons utiliser une fonction de coût quadratique afin d'obtenir une trajectoire très utile pour la suite. Cela nécessite donc de formuler un QP ce qui rendrait le temps de calcul pour la formulation en force bien trop élevé. Nous avons donc décidé d'utiliser la formulation par double description pour l'instant.

4.6.6 Validité des contraintes cinématiques utilisées

Dans ce chapitre, nous avons utilisé l'approximation des contraintes cinématiques présentée dans la section 3.3.1. Nous nous intéressons maintenant à évaluer leur justesse.

Nous rappelons que le but de cette approximation est de traduire les contraintes cinématiques du modèle complet du robot pour le modèle centroïdale. L'approximation utilisée est une condition nécessaire mais non suffisante sur la position du centre de masse pour l'existence d'une configuration corps complet cinématiquement faisable.

Afin de tester la justesse de cette approximation, pour chaque séquence de trois phases de contact pour lesquelles CROC a convergé, nous testons explicitement la faisabilité cinématique de la trajectoire résultante. Pour tester cette faisabilité, nous utilisons une méthode d'inverse cinématique en contraignant les contacts définis par les phases de contact à être fixes et en contraignant le centre de masse à suivre la trajectoire trouvée par CROC.

Ces tests ont montré que 17.5% des trajectoires trouvées par CROC étaient partiellement infaisables cinématiquement. Cela confirme que notre représentation des contraintes cinématiques n'est pas suffisante. Cependant, ce n'est pas une limitation de la méthode proposée dans ce chapitre. En effet, nous pourrions utiliser d'autres types de représentation des contraintes cinématiques tant que celles-cis peuvent se représenter linéairement en fonction de la position du centre de masse.

De plus, en faisant les mêmes tests mais sans considérer aucune contraintes cinématiques dans la formulation du problème CROC, nous obtenons 78.3% de trajectoires infaisables cinématiquement. Cela montre bien l'intérêt de l'utilisation de ces contraintes cinématiques, même si elles ne sont pas suffisantes.

4.6.7 Exemples de trajectoires

La figure 4.8 montre des exemples de trajectoires trouvées par la méthode CROC, avec l'espace des solutions valides pour $\mathbf{y}$ et la solution choisie selon notre coût qui minimise la norme de l'accélération. Sur la ligne du haut, HRP-2 monte une marche de 15cm en prenant appui avec sa main sur une rambarde. La ligne du milieu montre HyQ marchant entre deux planches inclinées de 45°.

Sur la dernière ligne, HRP-2 enjambe un trou en marchant sur une planche inclinée, ce mouvement n'admet aucune solution quasi-statique contrairement aux deux premiers, car il est impossible de trouver une configuration en équilibre statique avec seulement un pied sur la plate-forme inclinée. Cet exemple montre bien que CROC est capable de générer des trajectoires dynamiques.

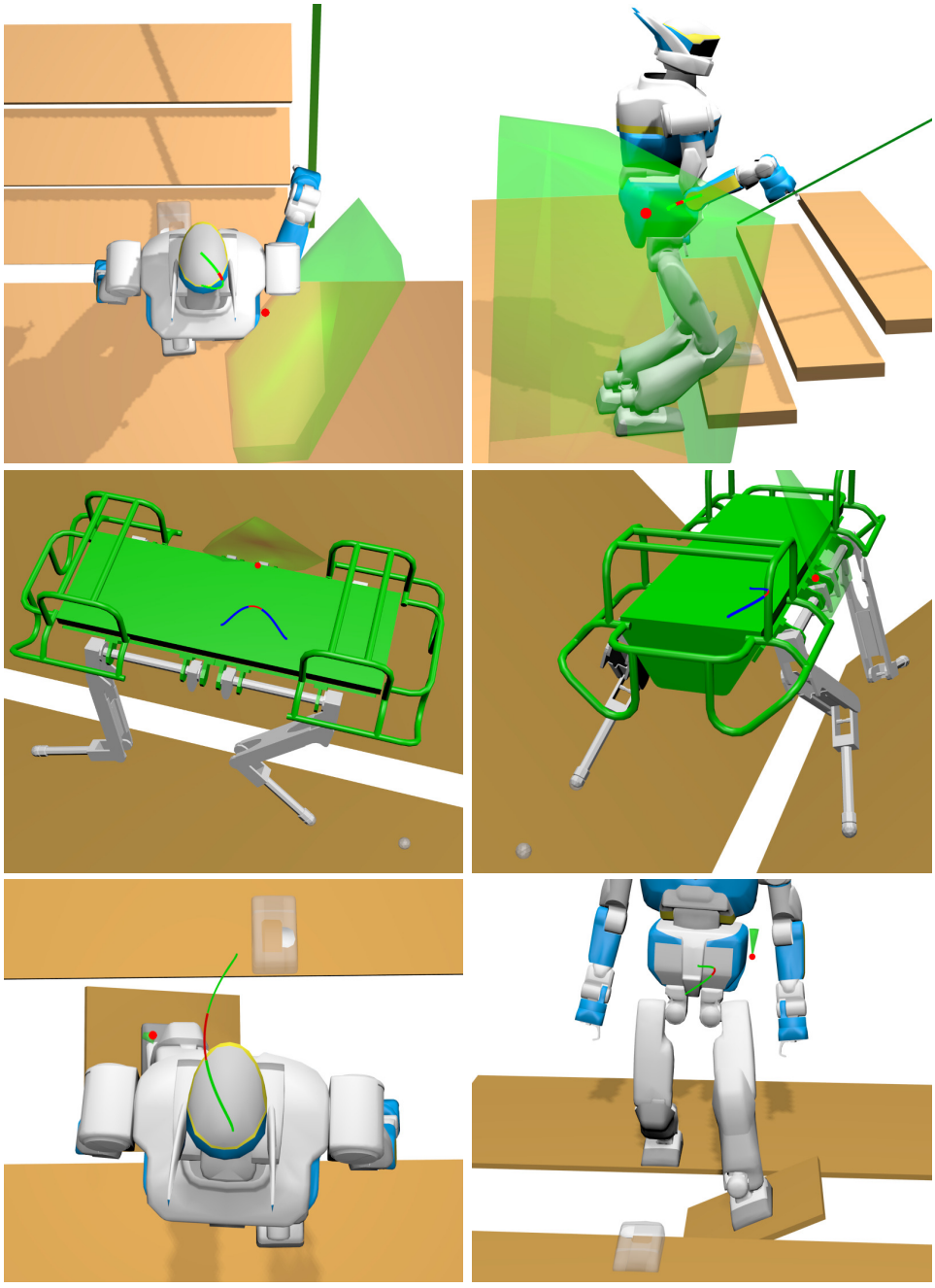
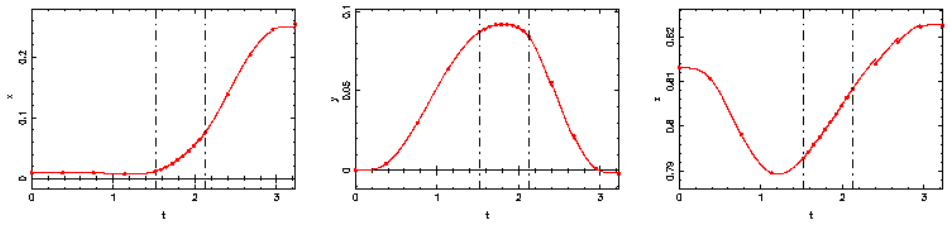
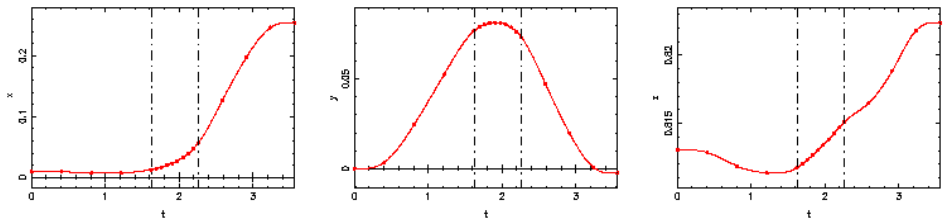


FIGURE 4.8 – Exemples de trajectoires centroïdales trouvées par la méthode CROC. La position montrée est la position initiale, le prochain contact est montré en transparent. Les polytopes verts représentent l'espace des positions de $\mathbf{y}$ qui respectent les contraintes. La sphère rouge : la solution choisie pour $\mathbf{y}$ selon le coût utilisé. Les trajectoires vertes et rouges ou bleues et rouges sont les trajectoires centroïdales générées, avec un changement de couleur à chaque changement de phase de contact.

4.6.7.1 Comparaison avec une méthode de génération de trajectoire non linéaire



(a) solution de CROC



(b) solution de la méthode de génération de trajectoire non linéaire

FIGURE 4.9 – Comparaison des trajectoires trouvées par CROC et la méthode de génération de trajectoire non linéaire [Carpentier 2017c], dans le cas d'un pas de 50cm sur sol plat. Les colonnes représentent les axes (x, y, z), les traits pointillés verticaux représentent les changements de phases de contact. Les points rouges sont les points de discrétisation de la méthode de génération de trajectoire.

La figure 4.9 compare un exemple de solution trouvée par CROC, à la solution trouvée par la méthode de génération de trajectoire [Carpentier 2017c] initialisée avec la solution de CROC. Pour cet exemple, les deux méthodes utilisent la même fonction de coût : minimiser la norme de l'accélération. Visuellement les deux solutions sont proches, cependant l'analyse de la valeur finale de la fonction de coût montre une différence de 22% en faveur de la méthode de génération de trajectoire de [Carpentier 2017c].

En effet, notre représentation de la trajectoire centroïdale par une courbe de Bézier de degré 6 ne nous permet pas de couvrir l'ensemble de l'espace des solutions et contraint la forme de la trajectoire trouvée par CROC alors que la méthode de génération de trajectoire non linéaire n'impose pas de conditions sur la forme de la courbe. On peut observer cette différence notamment selon l'axe z sur la figure 4.9 où il apparaît clairement que la trajectoire trouvée par la méthode de génération de trajectoire (b) pour cet axe ne peut pas être représentée par une courbe de Bézier de degré 6.

4.7 Discussion

Dans ce chapitre, nous avons proposé une formulation exact et efficace de la dynamique centroïdale des robots à pattes, qui est convexe et conservatrice mais non limitée au cas quasi-statique. A notre connaissance c'est la première formulation à présenter ces propriétés simultanément.

Nous avons ensuite utilisé cette formulation pour proposer une méthode résolvant le problème de la faisabilité d'une transition de contact, c'est à dire déterminer s'il existe au moins une trajectoire centroïdale valide connectant deux états appartenant à des phases de contact différentes.

Grâce à la convexité de notre formulation, nous pouvons résoudre ce problème de manière très efficace en quelques millisecondes pour une séquence de trois phases de contact. Cela en fait la première méthode assez rapide pour pouvoir être intégrée dans un planificateur de contact, nous détaillerons cette intégration et discuterons de son intérêt dans la troisième partie de ce manuscrit.

Contrairement aux formulations obtenant un problème convexe grâce à des approximations ou relaxations des contraintes, il est important de noter que notre formulation de la dynamique est exacte et que la convexité est obtenue par un choix judicieux de représentation de la trajectoire centroïdale. Cela signifie que les solutions trouvées par CROC sont garanties d'être exactes par rapport à la dynamique centroïdale. Cette propriété a bien été validée par nos tests puisque lors de la comparaison avec une méthode de génération de trajectoire non-linéaire de l'état de l'art nous n'avons obtenue aucun faux-positifs. C'est a dire que pour toutes les instances de problèmes pour lesquelles CROC a convergé, la méthode non-linéaire a également convergé.

Cependant, cette convexité a un prix et notre représentation ne peut couvrir l'ensemble de l'espace des solutions, notre reformulation est donc conservatrice. Nous avons toutefois montré empiriquement que nous pouvons résoudre entre 90% et 60% des problèmes résolues par une méthode de résolution non linéaire, selon la difficulté du mouvement. Cela, pour un temps de calcule de 10 à 50 fois plus rapide.

Un autre point intéressant est d'utiliser la solution trouvée par CROC pour initialiser la méthode de résolution non linéaire utilisée pendant la génération de trajectoire. Des résultats quantitatifs montrent une amélioration des taux de succès ainsi qu'une diminution du temps de calcul requis par la génération de trajectoire.

4.7.1 Application au problème du 0 et 1 step capturability

Le problème du *N-step capturability* consiste à déterminer s'il est possible au robot de s'arrêter dans une position en équilibre statique à partir d'un état donné, en faisant au maximum N pas.

Avec notre formulation, nous pouvons aisément changer les contraintes définies par les conditions aux limites dans la section 4.1.1. Pour le problème du N-step

capturability, il ne faut plus considérer de contraintes sur la position finale $\mathbf{c}_g$ et imposer les contraintes $(\dot{\mathbf{c}}_g = 0, \ddot{\mathbf{c}}_g = 0)$.

Avec cet ensemble de contraintes, la résolution du LP (4.19) permet de déterminer le 0-step capturability. De même, le LP (4.24) peut résoudre le problème de 1-step capturability.

4.7.2 Prise en compte du moment cinétique

Dans nos développements nous avons supposé que $\dot{\mathbf{L}}(t) = \mathbf{0}$. Mais ce n'est pas une limitation de notre formulation. En effet, il serait possible de prendre en compte le moment cinétique $\dot{\mathbf{L}}$ avec notre formulation. Si $\dot{\mathbf{L}}(t)$ est représenté par une courbe de Bézier, nous pouvons l'inclure dans l'expression du GIW.

Cependant, il faudrait être capable de garantir qu'il est possible de générer un mouvement corps complet qui suit la courbe $\dot{\mathbf{L}}(t)$ donnée. Cela requiert des informations sur la dynamique du modèle corps complet dont nous ne disposons pas avec le modèle centroïdal. Une approche possible serait d'itérer avec une méthode de génération de mouvement corps complet permettant de calculer ce moment cinétique [Herzog 2016].

4.7.3 Approximations et incertitudes liées au modèle centroïdal

Bien que notre formulation soit exacte par rapport au modèle de la dynamique centroïdale considérée, notre formulation du problème contient encore des approximations partagées avec toutes les autres méthodes de l'état de l'art utilisant seulement le modèle centroïdal. En effet, certaines contraintes exprimées par le modèle corps-complet du robot tel que les contraintes cinématiques sont approximées dans notre formulation. D'autres contraintes comme les limites de couple ou le moment cinétique sont ignorées.

Une solution proposée dans la littérature serait d'itérer entre une phase d'optimisation sur le modèle centroïdal et une phase considérant le modèle corps complet [Herzog 2016]. Cependant, cette approche entraîne une augmentation bien trop importante du temps de calcul qui n'est pas possible dans notre contexte du problème de faisabilité d'une transition de contact.

Une autre possibilité permettant d'améliorer la qualité de ces approximations serait de considérer les contraintes de couple comme proposé dans [Orsolino 2018, Samy 2017]. Il faudrait toutefois être capable d'exprimer ces contraintes sans connaissance de la configuration corps-complet mais seulement à partir des informations du modèle centroïdal. Ceci est une piste de recherche ouverte.

Finalement, il serait intéressant de garantir que notre méthode produit des solutions robustes aux incertitudes du monde réel. Pour réussir cela, une approche possible est d'ajouter une variable d'écart au problème qui permettrait de maximiser la distance à certaines contraintes et de rejeter les solutions qui ne satisfont pas une distance minimale. Une formulation similaire est proposée dans [Tonneau 2018a]. Le résultat serait une formulation encore plus conservatrice mais garantissant la

robustesse de la solution trouvée.

4.7.4 Utilisation de CROC pour la génération de trajectoire

Avec l'introduction d'une fonction de coût comme proposée dans la section 4.2.4, CROC peut générer des trajectoires centroïdales valides minimisant ce coût. Cependant, même avec une fonction de coût identique notre méthode trouvera une solution moins optimale (ie. la valeur finale du coût sera supérieure) qu'une méthode de génération de trajectoire non-linéaire. Tout d'abord car notre représentation ne couvre pas tout l'espace des solutions mais aussi parce que nous ne pouvons pas optimiser la durée de chaque phase du mouvement.

De plus, comme la courbe représentant la trajectoire centroïdale ne peut pas présenter plus d'un point d'inflexion, il n'est pas possible de considérer de nombreuses phases de contact dans un même problème. En effet, dans un mouvement il y a généralement un point d'inflexion dans la trajectoire centroïdale par pas ou par repositionnement de contact. Cela nous oblige à résoudre séquentiellement des problèmes avec un faible nombre de phases de contact, dans notre implémentation nous nous sommes limité à 3.

Dans le cas du problème de faisabilité d'une transition de contact, cela n'est pas limitant. Mais, dans le cas de la génération de trajectoire, cela nous empêche de considérer la trajectoire complète pendant l'optimisation et donc de pouvoir optimiser correctement cette trajectoire en fonction des changements de contacts futurs.

Malgré cela, la méthode CROC pourrait être utilisée comme méthode de génération de trajectoire dans les cas où la garantie de convergence et la vitesse d'exécution prime sur l'optimalité de la solution. C'est par exemple le cas dans le domaine de l'animation graphique où CROC pourrait être utilisée directement pour générer des trajectoires centroïdales. En effet, l'utilisation de méthodes de résolution linéaire garantissent la convergence contrairement aux méthodes non-linéaires avec un temps de calcul d'au moins un ordre de grandeur inférieur aux méthodes de génération de trajectoires actuelles. Ces deux propriétés sont plus importantes dans le domaine de l'animation graphique que l'optimalité de la solution.

De plus, il est possible d'agir sur la forme de la solution produite et donc sur le *style* du mouvement final en agissant seulement sur la fonction de coût, comme détaillé dans la section 4.4. Cette propriété est commune avec les méthodes non linéaires mais ici notre variable et nos contraintes peuvent être représentées en 3D comme montré sur la figure 4.8. Grâce à cette représentation en 3D du problème, il est bien plus aisé de visualiser directement l'influence des changements apportés à la fonction de coût sur la position du point du contrôle et donc sur la trajectoire générée.

Il serait également possible de permettre une édition manuelle de la position du point de contrôle (la sphère rouge dans la figure 4.8), à l'intérieur des contraintes

3D, pour agir directement sur le style du mouvement tout en garantissant que la trajectoire générée sera valide.

Chapitre 5

Formulation continue du problème de faisabilité

Sommaire

Dans le chapitre précédent, nous avons proposé une reformulation convexe de la dynamique centroïdale grâce une représentation de la trajectoire centroïdale par une courbe de Bézier. Cette reformulation étant exacte et ne nécessitant aucune relaxation ou approximation de la dynamique, les trajectoires obtenues devraient toujours être valides.

Or, ce n'est pas le cas. En effet, comme toutes les autres méthodes de l'état de l'art [Carpentier 2017c, Herzog 2015, Dai 2014, Caron 2018, Fernbach 2018], nous avons dû recourir à la discrétisation de la trajectoire afin de tester les contraintes. Le résultat est que les contraintes ne sont vérifiées que pour les points de discrétisation, et que la trajectoire solution n'est garantie d'être valide que pour ces points. Or, la figure 5.1 montre bien qu'il est possible d'avoir une trajectoire invalide entre deux points de discrétisation valides.

La taille du problème, et donc le temps requis pour le résoudre, dépend du nombre de points de discrétisation utilisés. Le nombre de points de discrétisation est donc un paramètre important définissant un compromis entre le temps de calcul et le risque d'obtenir des trajectoires partiellement invalides.

Pour remédier à ce problème nous proposons dans ce chapitre une méthode permettant de vérifier de manière continue la validité d'une trajectoire tout en

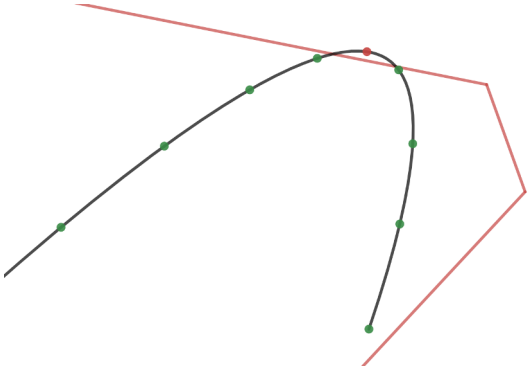


FIGURE 5.1 – Exemple de trajectoire invalide alors que chaque point de discrétisation est valide. En noir la trajectoire, en rouge les contraintes. Les points verts sont les points de discrétisation, le point rouge est un point invalide car il viole les contraintes.

évaluant les contraintes pour un nombre fixe de points, en exploitant une propriété particulière des courbes de Bézier. Cette méthode n'est pas conservatrice dans le cas général et n'entraîne donc aucune perte de solution.

Dans la suite de ce chapitre, nous détaillons cette formulation continue dans le cadre de CROC, qui utilise une courbe de Bézier de degré fixe, avec un unique point de contrôle comme variable.

Cependant, nous insistons par la suite sur le caractère générique de cette reformulation, qui peut être appliquée à n'importe quelle méthode d'optimisation de trajectoire centroïdale, convexe ou non-linéaire, tant que la trajectoire centroïdale est représentée par un polynôme. L'utilisation de cette formulation permet en théorie de supprimer définitivement le problème de la discrétisation, aussi nous pensons que l'application de cette formulation continue aux méthodes de génération de trajectoire centroïdale de la littérature est une piste de recherche prometteuse.

5.1 Reformulation continue

5.1.1 Enveloppe convexe d'une courbe de Bézier

Par construction, une courbe de Bézier est entièrement comprise dans l'enveloppe convexe de ses points de contrôle. Cela signifie que pour vérifier si une courbe de Bézier respecte un ensemble de contraintes linéaires, une condition suffisante est que ses points de contrôle vérifient ces contraintes, comme illustré par la figure 5.2.

Cette condition n'est toutefois pas nécessaire. En effet, il est possible qu'une courbe de Bézier soit entièrement comprise dans un ensemble de contraintes alors qu'un ou plusieurs de ses points de contrôle ne satisfont pas ces contraintes. Un tel exemple est montré dans la figure 5.3.

Dans la section 5.3 nous montrons que bien que cette condition soit conserva-

trice, nous pouvons l'utiliser de manière à ne perdre aucune solution. Notre reformulation continue n'est donc pas conservatrice dans le cas général.

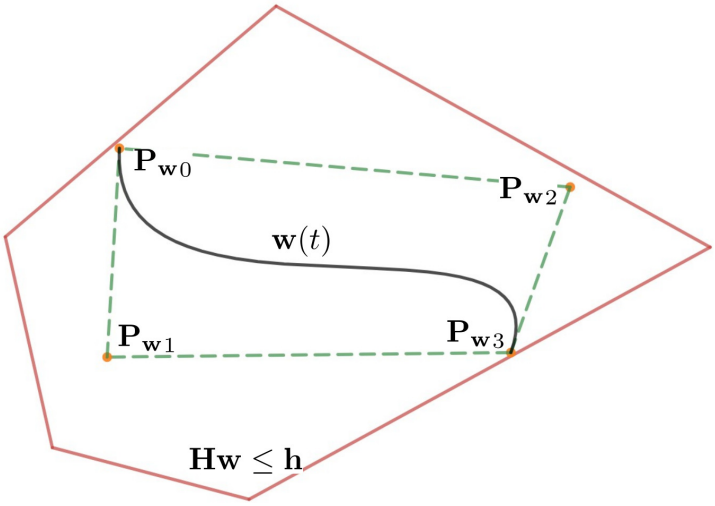


FIGURE 5.2 – Dans cette vue abstraite, le polygone rouge représente les contraintes 6D sur $\mathbf{w}(t)$. La courbe noire est une courbe de Bézier de degré 3 définie par les quatre points de contrôle $\mathbf{P}_{\mathbf{w}i}$ en orange. Les pointillés vert représentent l'enveloppe convexe des points de contrôle, contenant entièrement la courbe de Bézier.

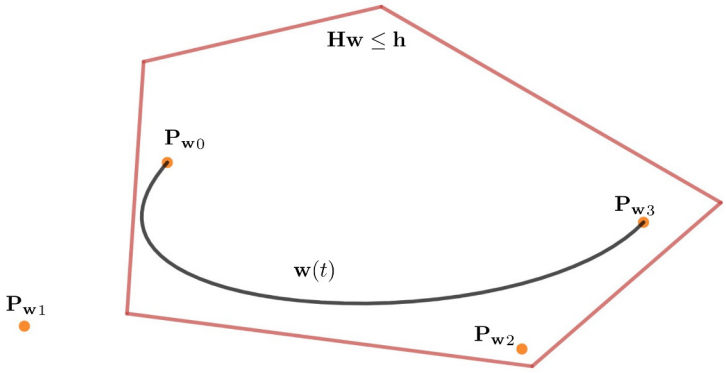


FIGURE 5.3 – Dans cette vue abstraite, le polygone rouge représente les contraintes 6D sur $\mathbf{w}(t)$. La courbe noire est une courbe de Bézier de degré 3 définie par les quatre points de contrôle $\mathbf{P}_{\mathbf{w}i}$ en orange. La courbe $\mathbf{w}(t)$ satisfait entièrement les contraintes alors que $\mathbf{P}_{\mathbf{w}1}$ ne les satisfait pas.

5.1.2 Application à un mouvement sans changement de contact

Dans le cas d'un mouvement sans changement de contact, les contraintes restent constantes pour toute la trajectoire :

$$\begin{aligned} \mathbf{K}^{\{p\}}\mathbf{c}(t,\mathbf{y}) &\leq \mathbf{k}^{\{p\}} \quad , \forall t \in [0;T] \\ \mathbf{H}^{\{p\}}\mathbf{w}(t,\mathbf{y}) &\leq \mathbf{h}^{\{p\}} \quad , \forall t \in [0;T] \end{aligned} \tag{5.1}$$

En utilisant la propriété qu'une courbe de Bézier est entièrement comprise dans l'enveloppe convexe de ses points de contrôle, il suffit de vérifier si les points de contrôles de $\mathbf{c}(t,\mathbf{y})$ vérifient les contraintes cinématiques et si les points de contrôle de $\mathbf{w}(t,\mathbf{y})$ vérifient les contraintes dynamiques.

Dans le cas de $\mathbf{c}(t,\mathbf{y})$, tout les points de contrôles sauf un sont constants et dépendent seulement des conditions aux limites. En supposant que les états initiaux et finaux du problème sont valides (autrement le problème n'a aucune solution) il suffit de vérifier si le point de contrôle variable $\mathbf{y}$ satisfait les contraintes cinématiques.

Pour $\mathbf{w}(t,\mathbf{y})$, nous rappelons que nous pouvons calculer analytiquement l'expression des points de contrôles $\mathbf{P}_{\mathbf{w}i}$ et que, comme montré dans la section 4.1.2.3, ils peuvent s'exprimer de la manière suivante :

$$\mathbf{P}_{\mathbf{w}i}(\mathbf{y}) = \mathbf{P}^{y}_{\mathbf{w}i}\mathbf{y} + \mathbf{P}^{s}_{\mathbf{w}i} \tag{5.2}$$

Nous pouvons alors déterminer l'existence d'une trajectoire centroïdale satisfaisant l'ensemble des contraintes, de manière certaine et sans avoir recours à la discrétisation de la trajectoire, grâce au LP suivant :

$$\begin{aligned} \textbf{find} \quad & \mathbf{y} \\ \text{subject to} \quad & \mathbf{K}^{\{p\}}\mathbf{y} \leq \mathbf{k}^{\{p\}} \\ & \mathbf{H}^{\{p\}}\mathbf{P}^{y}_{\mathbf{w}i}\mathbf{y} \leq \mathbf{h}^{\{p\}} - \mathbf{H}^{\{p\}}\mathbf{P}^{s}_{\mathbf{w}i} \ , \forall i \in [0; 2n-3] \end{aligned} \tag{5.3}$$

L'intérêt de cette formulation est que la trajectoire est garantie d'être entièrement valide. De plus, la dimension des contraintes ne dépend plus d'un pas de discrétisation manuellement défini mais est fixe. En effet, les contraintes cinématiques ne doivent être vérifiées que pour un seul point, et les contraintes dynamiques pour chaque point de contrôle de $\mathbf{w}(t,\mathbf{y})$, soit $2n-2$ points (10 dans notre cas car $n=6$).

5.1.3 Application à un mouvement avec un changement de contact

Dans le cas d'un mouvement avec un changement de contact entre les phases de contact $\{p\}$ et $\{q\}$ qui a lieu au temps $t = T^{\{p\}}$, nous avons deux ensembles distincts de contraintes s'appliquant sur des intervalles de temps différents :

$$\begin{aligned} \mathbf{K}^{\{p\}}\mathbf{c}(t,\mathbf{y}) &\leq \mathbf{k}^{\{p\}} \quad , \forall t \in [0;T^{\{p\}}] \\ \mathbf{H}^{\{p\}}\mathbf{w}(t,\mathbf{y}) &\leq \mathbf{h}^{\{p\}} \quad , \forall t \in [0;T^{\{p\}}] \\ \mathbf{K}^{\{q\}}\mathbf{c}(t,\mathbf{y}) &\leq \mathbf{k}^{\{q\}} \quad , \forall t \in [T^{\{p\}};T] \\ \mathbf{H}^{\{q\}}\mathbf{w}(t,\mathbf{y}) &\leq \mathbf{h}^{\{q\}} \quad , \forall t \in [T^{\{p\}};T] \end{aligned} \tag{5.4}$$

L'idée est de se ramener au cas ou un seul ensemble de contraintes s'applique sur toute une courbe. Pour parvenir à cela nous allons scinder les courbes $\mathbf{c}(t, \mathbf{y})$ et $\mathbf{w}(t, \mathbf{y})$ en deux à l'instant du changement de phases de contact $t = T^{\{p\}}$. Puis vérifier si chacune des courbes satisfait les contraintes définies par la phase de contact correspondante en vérifiant ces contraintes pour leur points de contrôle. La figure 5.4 illustre notre démarche.

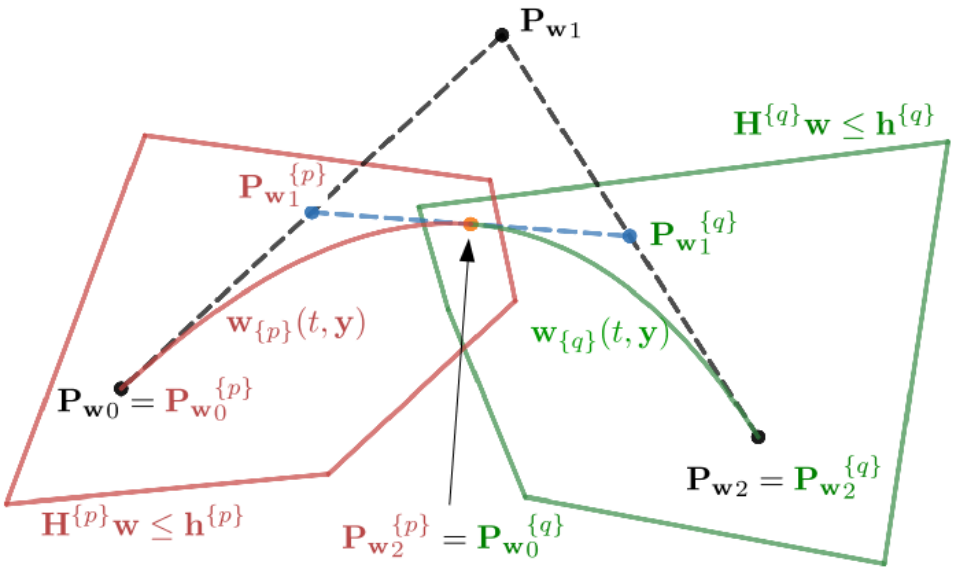


FIGURE 5.4 – Exemple de décomposition de la courbe $\mathbf{w}$ de degré 2 définie par les points de contrôle $\mathbf{P}_{\mathbf{w}i}$ (en noir), à l'instant de transition $t = T^{\{p\}}$ (point orange). La courbe est décomposée en deux courbes de degré 2 également, $\mathbf{w}_{\{p\}}$ définie par les points de contrôle $\mathbf{P}_{\mathbf{w}i}^{\{p\}}$ (en rouge) et $\mathbf{w}_{\{q\}}$ définie par les points de contrôle $\mathbf{P}_{\mathbf{w}i}^{\{q\}}$ (en vert). Chacun de ces ensembles de points de contrôle est alors contraint à satisfaire les contraintes de la phase de contact correspondante. Le résultat est que la courbe satisfait les contraintes correspondantes à la phase de contact courante à chaque instant. On peut noter que les points de contrôle de la courbe originale ne sont pas contraints à satisfaire les contraintes.

Afin de réussir cela, il faut pouvoir calculer l'expression analytique des points de contrôle des nouvelles courbes scindées. Pour ce faire, nous utilisons l'algorithme de De Casteljau (présenté dans l'annexe A.8) qui garantit que la décomposition des courbes est strictement équivalente à la courbe originale. Nous avons donc :

$$\begin{cases} \mathbf{c}_{\{p\}}(t, \mathbf{y}) = \mathbf{c}(t, \mathbf{y}) & \forall t \in [0; T^{\{p\}}] \\ \mathbf{c}_{\{q\}}(t, \mathbf{y}) = \mathbf{c}(t, \mathbf{y}) & \forall t \in [T^{\{p\}}; T] \\ \mathbf{c}_{\{p\}}(T^{\{p\}}, \mathbf{y}) = \mathbf{c}_{\{q\}}(T^{\{p\}}, \mathbf{y}) & \end{cases} \tag{5.5}$$

$$\begin{cases} \mathbf{w}_{\{p\}}(t,\mathbf{y}) = \mathbf{w}(t,\mathbf{y}) & \forall t \in [0;T^{\{p\}}] \\ \mathbf{w}_{\{q\}}(t,\mathbf{y}) = \mathbf{w}(t,\mathbf{y}) & \forall t \in [T^{\{p\}};T] \\ \mathbf{w}_{\{p\}}(T^{\{p\}},\mathbf{y}) = \mathbf{w}_{\{q\}}(T^{\{p\}},\mathbf{y}) & \end{cases} \tag{5.6}$$

De plus, les courbes scindées ont le même degré que la courbe originale et les points de contrôle de ces courbes sont linéairement dépendants des points de contrôle de la courbe originale. Nous pouvons donc exprimer chaque courbe $\mathbf{c}_{\{z\}}(t,\mathbf{y})$, scindée à partir de $\mathbf{c}(t,\mathbf{y})$, qui correspond à l'intervalle de temps de la phase $\{z\}$ ainsi :

$$\mathbf{c}_{\{z\}}(t,\mathbf{y}) = \sum_{i=0}^{n} B_i^n(t/T^{\{z\}})\mathbf{P}_i^{\{z\}}(\mathbf{y}) \qquad ,\forall z \in \{p,q\} \tag{5.7}$$

Où les points de contrôle des courbes scindées $\mathbf{P}_i^{\{z\}}$ peuvent s'exprimer :

$$\mathbf{P}_i^{\{z\}}(\mathbf{y}) = \mathbf{P}_i^{y\{z\}}\mathbf{y} + \mathbf{P}_i^{s\{z\}} \tag{5.8}$$

Avec $\mathbf{P}_i^{y\{z\}}$ et $\mathbf{P}_i^{s\{z\}}$ des constantes dont l'expression analytique est donnée par l'algorithme de De Casteljau en fonction des points de contrôle de la courbe originale, les développements sont présentés en annexe A.8.

Le même développement peut être fait pour $\mathbf{w}(t,\mathbf{y})$:

$$\begin{gathered} \mathbf{w}_{\{z\}}(t,\mathbf{y}) = \sum_{i=0}^{2n-3} B_i^{2n-3}(t/T^{\{z\}})\mathbf{P}_{\mathbf{w}i}^{\{z\}}(\mathbf{y}) \\ \text{Avec } \mathbf{P}_{\mathbf{w}i}^{\{z\}}(\mathbf{y}) = \mathbf{P}_{\mathbf{w}i}^{y\{z\}}\mathbf{y} + \mathbf{P}_{\mathbf{w}i}^{s\{z\}} \quad ,\forall z \in \{p,q\} \end{gathered} \tag{5.9}$$

Maintenant que nous avons les expressions des points de contrôle de chaque courbe scindée, nous pouvons exprimer les contraintes cinématiques de l'équation (5.4) pour chaque phase de contact de la manière suivante :

$$\underbrace{\mathbf{K}^{\{z\}}\mathbf{P}_i^{y\{z\}}}_{\mathbf{A}_i^{\{z\}}}\mathbf{y} \leq \underbrace{\mathbf{k}^{\{z\}} - \mathbf{K}^{\{z\}}\mathbf{P}_i^{s\{z\}}}_{\mathbf{a}_i^{\{z\}}}, \forall i \in [0;n], \forall z \in \{p,q\} \tag{5.10}$$

et les contraintes dynamiques s'expriment :

$$\underbrace{\mathbf{H}^{\{z\}}\mathbf{P}_{\mathbf{w}j}^{y\{z\}}}_{\mathbf{D}_j^{\{z\}}}\mathbf{y} \leq \underbrace{\mathbf{h}^{\{z\}} - \mathbf{H}^{\{z\}}\mathbf{P}_{\mathbf{w}j}^{s\{z\}}}_{\mathbf{d}_j^{\{z\}}}, \forall j \in [0;2n-3], \forall z \in \{p,q\} \tag{5.11}$$

Ensuite, nous pouvons empiler toutes ces contraintes pour chaque point de contrôle et chaque phase :

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_0^{\{p\}} \\ \vdots \\ \mathbf{A}_n^{\{p\}} \\ \mathbf{A}_0^{\{q\}} \\ \vdots \\ \mathbf{A}_n^{\{q\}} \end{bmatrix} \mathbf{a} = \begin{bmatrix} \mathbf{a}_0^{\{p\}} \\ \vdots \\ \mathbf{a}_n^{\{p\}} \\ \mathbf{a}_0^{\{q\}} \\ \vdots \\ \mathbf{a}_n^{\{q\}} \end{bmatrix} \mathbf{D} = \begin{bmatrix} \mathbf{D}_0^{\{p\}} \\ \vdots \\ \mathbf{D}_{2n-3}^{\{p\}} \\ \mathbf{D}_0^{\{q\}} \\ \vdots \\ \mathbf{D}_{2n-3}^{\{q\}} \end{bmatrix} \mathbf{d} = \begin{bmatrix} \mathbf{d}_0^{\{p\}} \\ \vdots \\ \mathbf{d}_{2n-3}^{\{p\}} \\ \mathbf{d}_0^{\{q\}} \\ \vdots \\ \mathbf{d}_{2n-3}^{\{q\}} \end{bmatrix} \tag{5.12}$$

Nous rappelons que dans notre cas $n = 6$.

Finalement, nous pouvons formuler le problème de la faisabilité d'une transition de contact comme un LP :

$$\begin{aligned} \textbf{find} \quad & \mathbf{y} \\ \text{subject to} \quad & \mathbf{A}\mathbf{y} \leq \mathbf{a} \\ & \mathbf{D}\mathbf{y} \leq \mathbf{d} \end{aligned} \tag{5.13}$$

Les contraintes sont donc que chaque point de contrôle de chaque courbe scindée doit satisfaire les contraintes imposées par la phase de contact correspondante.

Si ce LP converge, nous avons la garantie qu'il existe une position pour $\mathbf{y}$ telle que la trajectoire centroïdale $\mathbf{c}(t, \mathbf{y})$ soit entièrement valide, c'est à dire qu'elle respecte exactement les contraintes de l'équation (5.4) à chaque instant t.

5.1.4 Application dans le cas général

Dans le cas général, la même idée peut s'appliquer. Nous scindons alors $\mathbf{c}(t, \mathbf{y})$ et $\mathbf{w}(t, \mathbf{y})$ en autant de courbes qu'il y a de phases de contact.

De manière identique au chapitre précédent, il est possible d'ajouter des contraintes supplémentaires sur $\mathbf{c}(t)$ ou sur ses dérivées. Cette fois ci, nous vérifions également ces contraintes de manière continue en utilisant la formulation présentée ci-dessus. Nous notons ces contraintes :

$$\mathbf{O}\mathbf{y} \leq \mathbf{o} \tag{5.14}$$

Comme décrit dans le chapitre précédent, il est également possible d'ajouter une fonction de coût et de formuler le problème comme un QP :

$$\begin{aligned} \textbf{find} \quad & \mathbf{y} \\ \textbf{min} \quad & l(\mathbf{y}) \\ \text{subject to} \quad & \mathbf{A}\mathbf{y} \leq \mathbf{a} \\ & \mathbf{D}\mathbf{y} \leq \mathbf{d} \\ & \mathbf{O}\mathbf{y} \leq \mathbf{o} \end{aligned} \tag{5.15}$$

Dans le reste de ce manuscrit, la méthode permettant de résoudre le QP (5.15) est appelée la méthode *CROC continue*. Par défaut, lorsque nous parlons de la méthode *CROC* sans précision, il s'agit de cette méthode continue.

5.2 Comparaison avec la formulation discrétisée

Dans cette section nous allons comparer la formulation continue de CROC avec la version discrétisée présentée dans le chapitre précédent. D'après la propriété utilisée sur l'appartenance de la courbe à l'enveloppe convexe des points de contrôle présentée dans la section 5.1.1, nous savons que théoriquement cette formulation est plus conservatrice que la formulation discrétisée. Nous allons chercher à évaluer statistiquement à quel point nous perdons des solutions valides.

5.2.1 Protocole expérimental

La protocole expérimental est le même que pour le chapitre précédent : nous évaluons nos méthodes avec des séquences de trois phases de contact échantillonnées aléatoirement, avec des contacts coplanaires et non-coplanaires.

La différence étant qu'ici nous comparons la version continue de CROC avec la version discrétisée de CROC présentée dans le chapitre précédent. Pour cette dernière version, nous utilisons trois valeurs différentes pour le nombre de points de discrétisation par phases de contact.

Pour toutes les méthodes nous utilisons la formulation des contraintes dynamiques par inégalités, utilisant la méthode de double description.

Afin de valider les trajectoires solutions produites, nous vérifions les contraintes dynamiques à posteriori, le long des trajectoires solutions produites, avec un pas de discrétisation très petit ($1ms$).

Méthode	**Coplanaire**		**Non-coplanaire**		Temps QP (ms)	Temps total (ms)
	Taux de succès (%)	Solutions invalides (%)	Taux de succès (%)	Solutions invalides (%)		
Discrétisé (3 points)	89.7	10.6	61.4	19.7	0.026	0.20
Discrétisé (7 points)	89.7	6.7	60.6	9.3	0.049	0.37
Discrétisé (15 points)	89.1	4.2	60.6	6.9	0.092	0.75
Continue	88.4	0	57.2	0	0.047	0.41

Tableau 5.1 – Comparaison entre la méthode CROC continue proposée dans ce chapitre et la méthode discrétisée proposée dans le chapitre précédent (avec différentes valeurs de nombres de points de discrétisation par phase de contact), la "vérité terrain" utilisée pour calculer le taux de succès est la méthode de résolution non linéaire [Carpentier 2017c]. La comparaison des taux de succès est faite pour des scénarios avec des contacts coplanaires ou non-coplanaires. La colonne "Temps QP" désigne le temps requis pour résoudre le problème QP, la colonne "Temps total" désigne le temps requis par toute la méthode CROC, sans prendre en compte le temps requis pour la méthode de double description (qui est de $3.56ms$ en moyenne, identique pour toutes les formulations).

5.2.2 Temps de calcul

Comme attendu, les résultats du tableau 5.1 montrent que le temps de calcul augmente avec le nombre de points de discrétisation. Théoriquement le nombre de contraintes augmente linéairement en fonction du nombre de points de discrétisation, les résultats obtenus confirment que le temps de calcul augmente également de façon linéaire.

Le temps de calcul de la méthode continue est comparable à celui de la méthode discrétisée avec 7 points de discrétisation par phases de contact. Ce qui est logique puisqu'ici nous avons le degré de la courbe $n = 6$ et donc avec la formulation continue nous avons 7 points de contrôles par phases de contact à tester pour les contraintes cinématiques et 10 par phases pour les contraintes dynamiques. On a donc approximativement le même nombre de contraintes dans le problème continu et le problème discrétisé avec 7 points de discrétisation par phases.

5.2.3 Solutions de la formulation discrétisée invalides

Comme avancé dans l'introduction de ce chapitre, les méthodes discrétisées peuvent produire des solutions pour lesquelles une partie de la trajectoire ne satisfait pas les contraintes. La troisième colonne du tableau 5.1 le montre. En fonction du nombre de points de discrétisation choisis, le pourcentage de solutions invalides atteint des valeurs non négligeables (jusqu'à 20% de solutions invalides). Cela motive bien l'intérêt de notre vérification continue des contraintes, afin d'éviter de trouver des trajectoires partiellement invalides.

Dans le cas de la formulation continue, nous pouvons vérifier que toutes les trajectoires produites sont bien valides.

5.2.4 Perte possible de solutions

Contraindre tous les points de contrôle de la courbe à satisfaire les contraintes est une condition non nécessaire, et donc conservatrice. En effet, il est possible qu'il existe une courbe entièrement valide mais dont les points de contrôle ne soient pas valides, comme montré dans la figure 5.3.

En faisant le choix de scinder la trajectoire en une courbe par phase de contact, les résultats du tableau 5.1 montrent une diminution du taux de succès par rapport à la méthode discrétisée. En effet, dans le cas non coplanaire il y a 5.6% de cas où la méthode discrétisée trouve une solution mais pas la méthode continue. Cette perte semble faible en regard du nombre de trajectoires invalides générées par l'approche discrétisée.

De plus, nous pouvons reformuler le problème afin de théoriquement supprimer cette perte.

5.3 Formulation continue et sans perte

Dans la formulation proposée, nous scindons la trajectoire en une courbe de Bézier par phase de contact. Sur la figure 5.4, nous observons déjà qu'avec cette division, le point de contrôle de la courbe noire n'est plus contraint à appartenir à l'ensemble des contraintes. Les contraintes portent alors sur les points de contrôle des courbes scindées. L'espace atteignable par $\mathbf{P}_{w1}$ dans la figure 5.4 est de ce fait augmenté.

Dans la figure 5.5 (a) les points de contrôle de la courbe sont situés sur les bornes de l'ensemble de contraintes, mais ils ne sont pas atteints. La courbe ne peut donc pas explorer plus d'espace dans cette direction, et en particulier, ne peut jamais atteindre la limite de l'ensemble de contraintes. Dans la figure 5.5 (b) la courbe est scindée en deux, les contraintes s'appliquent donc maintenant sur les points de contrôle de chacune des courbes scindées, et les points de contrôle de la courbe originale ne sont plus contraints. Dans ce cas, la courbe peut explorer plus d'espace que précédemment. En particulier, il est alors possible pour la courbe d'atteindre sa limite de validité.

A chaque division de la courbe, on augmente ainsi l'espace atteignable pour le point de contrôle $\mathbf{y}$ de $\mathbf{c}(t)$, jusqu'à, à l'infini, couvrir la totalité de son espace admissible.

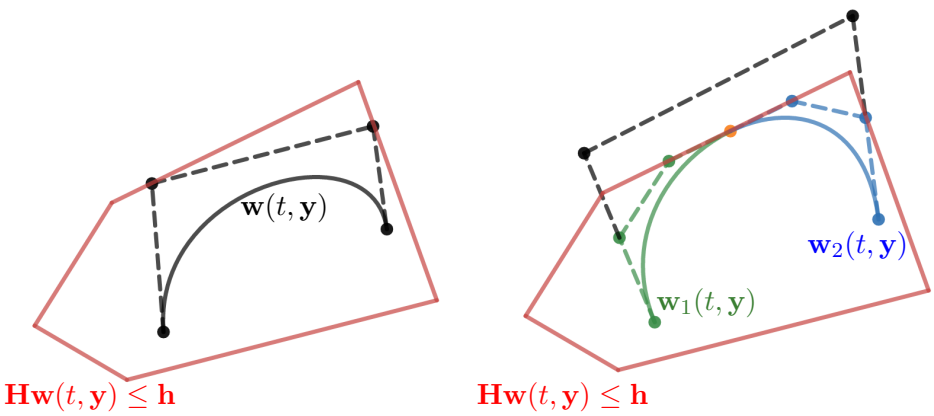


FIGURE 5.5 – Exemple de décomposition d'une courbe en deux afin de pouvoir explorer plus de solutions. En rouge les contraintes. A gauche en noir, la courbe originale définie par les points de contrôle représentés par des points noirs. A droite en vert et bleu les courbes scindées définies respectivement par les points de contrôles en vert et bleu, le point orange est un point de contrôle des deux courbes.

Pour l'utilisateur, il y a donc un choix à faire sur le nombre de divisions à introduire dans la courbe. Un grand nombre de divisions permet de mieux couvrir l'espace des solutions, mais résulte en une augmentation du nombre de contraintes.

Un parallèle peut être fait ici avec le choix du pas de discrétisation, à la diffé-

rence qu'une solution trouvée sera toujours valide. On peut aussi observer qu'augmenter le nombre de divisions revient essentiellement à autoriser la courbe solution à se rapprocher plus de la limite de l'espace des contraintes, et donc à proposer une solution moins robuste. On s'intéressera donc à choisir le nombre minimum de divisions permettant de trouver une solution. Dans le cas de CROC, conforté par les résultats du tableau 5.1, nous avons fait le choix de ne diviser la courbe qu'aux changements de phase de contact, comme expliqué précédemment. Dans le cas d'autres méthodes que CROC, il sera intéressant d'étudier la paramétrisation de la reformulation continue.

5.4 Conclusion

Dans ce chapitre, nous avons proposé une formulation continue des contraintes pour le problème de la faisabilité d'une transition de contact, applicable à n'importe quelle formulation du problème de la dynamique centroïdale. Cette formulation garantit que l'ensemble de la courbe satisfait les contraintes. De plus, elle n'est pas conservatrice dans le cas général et n'entraîne aucune perte de solutions.

Cette formulation exploite la propriété des courbes de Bézier garantissant que la courbe est entièrement comprise dans l'enveloppe convexe de ses points de contrôles. Il suffit alors de vérifier si les points de contrôle de la courbe satisfont les contraintes pour garantir que l'ensemble de la courbe satisfait ces contraintes.

Dans le cas d'un changement de phase de contact, les contraintes à vérifier vont changer à un instant donné de la trajectoire. Dans ce cas, nous scindons la trajectoire en plusieurs courbes de Bézier, une pour chaque ensemble de contraintes. Puis nous exprimons chaque ensemble de contraintes sur les points de contrôles des courbes correspondantes. Grâce à l'algorithme de De Casteljau, cette décomposition est exacte et strictement identique à la courbe originale.

Le résultat est une méthode garantissant que la solution est entièrement valide.

L'intérêt de cette formulation a été montré quantitativement. Les méthodes discrétisées nécessitent de choisir un nombre de points de discrétisation résultant d'un compromis entre le temps de calcul et le risque d'obtenir une trajectoire invalide. Pour un nombre de points de discrétisation donnant lieu à un temps de calcul similaire à celui de notre méthode continue, nous avons mesuré qu'environ 10% de solutions produites par la méthode discrétisée qui se sont avérées partiellement invalides.

Application de la formulation continue au cas général

Dans ce chapitre nous avons détaillé cette formulation en utilisant notre reformulation convexe du problème de la dynamique centroïdale proposée au chapitre précédent, mais elle peut s'appliquer à n'importe quelle autre méthode de génération de trajectoire, y compris à des méthodes non-linéaires.

Nous pensons que cette formulation continue présente un intérêt pour toutes les méthodes de génération de trajectoires. En effet, les méthodes trouvées dans la littérature [Carpentier 2017c, Herzog 2015, Ponton 2018, Dai 2014, Caron 2018, Winkler 2018] ont toutes recours à la discrétisation de la trajectoire pour évaluer les contraintes. D'autres auteurs ont déjà insisté sur l'intérêt d'une formulation continue des contraintes [Lengagne 2013]. Les résultats présentés dans ce chapitre confirment clairement que la discrétisation d'une trajectoire peut donner lieu à un pourcentage non négligeable de solutions invalides.

Comme la majorité de ces méthodes représentent déjà la trajectoire centroïdale par un polynôme, il serait aisé d'utiliser notre formulation continue dans ces méthodes. En effet, il suffirait de représenter les trajectoires centroïdales par une courbe de Bézier, cette fois ci sans fixer le degré et en utilisant un degré suffisamment grand pour pouvoir représenter exactement le polynôme original. Ensuite, l'application de la méthode proposée dans ce chapitre est identique, à l'exception que les points de contrôle de la courbe auront une expression non-linéaire en fonction des variables du problème. Il suffira ensuite d'exprimer les contraintes à ces points de contrôle au lieu de les exprimer à des points de discrétisation arbitraires, la manière d'exprimer ces contraintes est propre à chaque méthode et restera identique au cas discrétisé.

Selon le degré nécessaire pour que la courbe de Bézier représente exactement la trajectoire polynomiale générée par ces problèmes et le nombre de point de discrétisation précédemment utilisés, il est possible que l'on observe une dégradation des performances de ces méthodes en utilisant la formulation continue. Mais cela sera contrebalancé par la garantie que les contraintes sont respectées sur l'ensemble de la courbe.

Nous pensons donc que l'application de notre formulation continue des contraintes à d'autres méthodes de génération de trajectoires de la littérature est une piste de recherche prometteuse.

Troisième partie

Architecture expérimentale et résultats

Dans l'introduction de cette thèse, nous avons présenté notre approche au problème de la locomotion multi-contact. Nous rappelons que notre approche décompose le problème en trois sous-problèmes : $\mathcal{P}_1$ la planification d'une trajectoire guide pour le centre du robot, $\mathcal{P}_2$ la planification d'une séquence de contacts, $\mathcal{P}_3$ l'optimisation de la trajectoire centroïdale du robot puis la génération d'un mouvement corps complet.

Dans la première partie de cette thèse, nous nous sommes intéressés au problème de la **faisabilité** d'une trajectoire guide pour la planification de contact. Autrement dit, à l'établissement d'un **critère de faisabilité** devant être respecté par la solution trouvée par le sous-problème $\mathcal{P}_1$ pour être une entrée valide au sous-problème $\mathcal{P}_2$.

Dans la seconde partie, nous avons formulé le problème de la **faisabilité** d'une transition de contact qui consiste à déterminer l'existence d'au moins une trajectoire centroïdale valide connectant deux états appartenant à des phases de contact différentes. Nous avons ensuite proposé une méthode permettant de résoudre ce problème de manière linéaire et continue avec des temps de calcul de l'ordre de la milliseconde.

Dans cette partie nous allons montrer comment utiliser cette méthode comme un **critère de faisabilité** pendant la planification de contact ($\mathcal{P}_2$) pour garantir la convergence de la méthode de génération de trajectoire centroïdale ($\mathcal{P}_3$).

Grâce à ces contributions, nous pouvons maintenant compléter l'architecture globale de planification pour la **locomotion multi-contact de robots à pattes** évoquée dans l'introduction.

Cette partie détaille les différentes méthodes de l'architecture implémentée et montre les résultats expérimentaux obtenus.

Chapitre 6

Architecture expérimentale

Sommaire

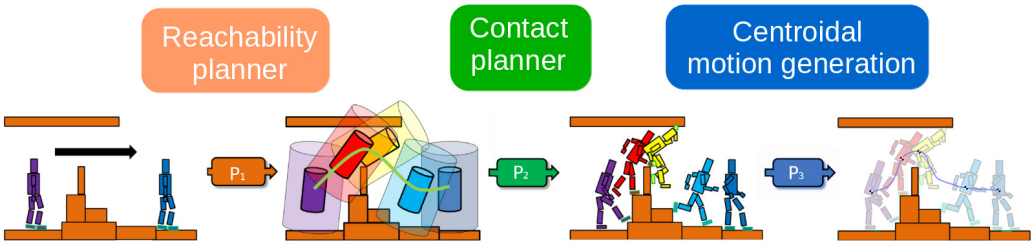


Figure 6.1 – Résumé de l'approche découplée utilisée. En entrée (à gauche) des tâches de haut niveau, comme un placement 6D but pour le centre du robot. En sortie (à droite) un mouvement corps-complet valide.

Dans ce chapitre, nous allons détailler l'architecture complète de planification de mouvement proposée par l'équipe Gepetto dans le cadre du projet loco-3D [Carpentier 2017b] et mise en œuvre durant cette thèse, dont l'approche globale est résumée sur la figure 6.1. Nous précisons également la manière dont sont intégrées les contributions des deux parties précédentes dans l'architecture complète.

Plusieurs contributions ont été faites pour résoudre chacun des sous-problèmes de la figure 6.1 [Tonneau 2018a, Carpentier 2017c] mais le lien automatique entre

les différents sous-problèmes n'avait pas encore été fait. En effet comme décrit dans l'introduction de ce manuscrit, afin que l'architecture globale puisse fonctionner correctement il faut que la sortie de chaque sous-problème soit une entrée faisable au sous-problème suivant.

Dans cette thèse nous avons introduit la notion de **faisabilité**, qui permet de garantir ou de donner de fortes probabilités qu'une solution trouvée par un sous-problème soit une entrée permettant au sous-problème suivant de trouver une solution. Dans le cas de la faisabilité entre le problème $\mathcal{P}_1$ et $\mathcal{P}_2$ traité dans la première partie de ce manuscrit, nous devons approximer certains éléments du modèle réduit. Notre critère de faisabilité n'est donc pas exact mais assure de fortes probabilités de faisabilité. Dans le cas de la faisabilité entre le problème $\mathcal{P}_2$ et $\mathcal{P}_3$ traité dans la seconde partie de ce manuscrit, notre critère est exact. Nous garantissons donc que la sortie du problème $\mathcal{P}_2$ est une entrée faisable au problème de génération de trajectoire centroïdale du sous-problème $\mathcal{P}_3$.

Dans la section suivante, nous détaillons les relations entre chaque bloc constituant notre architecture et la manière dont ils communiquent entre eux. Puis, dans le reste de ce chapitre nous détaillons chacun de ces blocs.

6.1 Vue globale

Afin de résoudre le problème de la planification de mouvement multi-contact pour robots à pattes notre méthode doit être capable de produire un mouvement corps-complet pour le robot respectant les contraintes dynamiques, cinématiques et de non collision, à partir seulement d'une tâche de haut niveau comme un placement (position et orientation 3D) but pour le centre du robot.

La figure 6.2 présente l'architecture complète mise en place durant cette thèse permettant de résoudre ce problème. Les entrées sont un modèle du robot et de l'environnement, des bornes de vitesse et accélération pour le centre du robot ainsi qu'une configuration corps-complet initiale et la phase de contact initiale. Ensuite, la tâche de locomotion est définie par un placement final désiré pour le centre du robot. La sortie est un mouvement corps-complet valide.

Le premier bloc *RB-RRT Kinodynamic* présenté dans le chapitre 2 planifie une trajectoire guide pour l'état centroïdal $\mathbf{x}_{planning}(t) =< \mathbf{c}_{planning}(t), \dot{\mathbf{c}}_{planning}(t), \ddot{\mathbf{c}}_{planning}(t) >$ et pour les degrés de liberté du tronc du robot $\mathbf{q}^{trunk}(t)$, en utilisant un modèle simplifié permettant de planifier dans un espace de faible dimension tout en assurant une forte probabilité de la faisabilité de la solution planifiée. Ce bloc résout donc le sous-problème $\mathcal{P}_1$.

Le second bloc est le générateur de contact. Cette méthode prend en entrée la trajectoire guide et produit une séquence de contacts permettant de suivre ce guide, elle produit également un estimation initiale de la durée de chaque phase de contact. Ce bloc résout donc le sous-problème $\mathcal{P}_2$.

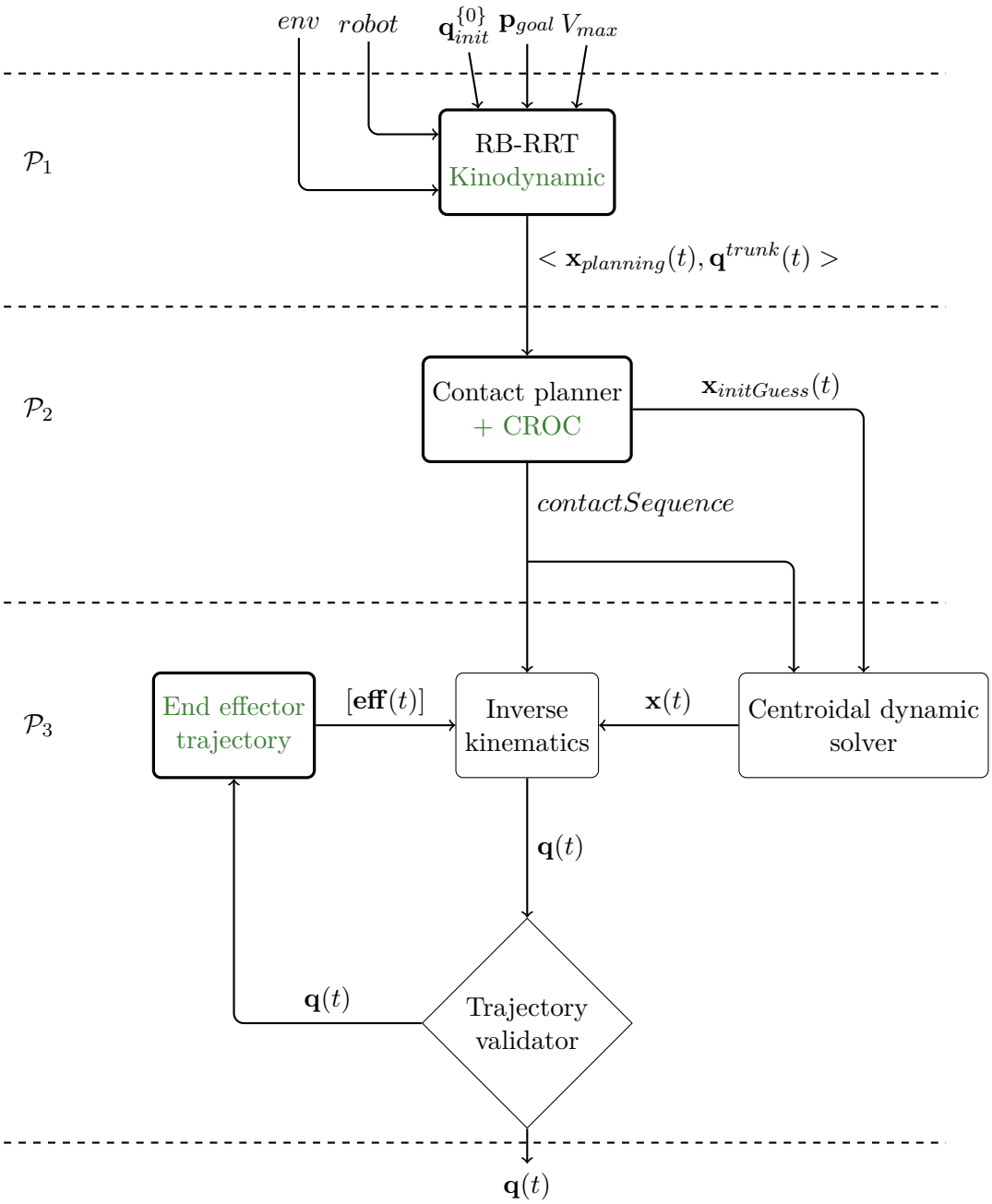


FIGURE 6.2 – Architecture complète utilisée. En vert les contributions scientifiques de cette thèse et en gras les blocs utilisant ces contributions.

Grâce à la méthode CROC proposée dans la seconde partie de ce manuscrit, nous pouvons définir un critère de faisabilité et l'utiliser durant la planification de contact afin de ne produire que des séquences de contact faisables. Cette intégration est détaillée dans la section 6.4 suivante. Avec un faible coût supplémentaire, CROC

produit une trajectoire centroïdale valide pour chaque transition de la séquence de contacts : $\mathbf{x}(t)_{initGuess}$. Ces trajectoires sont utilisées pour initialiser la méthode de résolution non-linéaire utilisée par le bloc suivant.

Le bloc *centroïdal dynamic solver* prend en entrée une séquence de contacts avec une durée initiale pour chaque phase de contact et une solution initiale pour la trajectoire centroïdale. Il produit en sortie une trajectoire centroïdale $\mathbf{x}(t)$ respectant les contraintes dynamiques et optimale selon un coût prédéfini. La méthode utilisée est celle présentée dans [Carpentier 2017c].

Le bloc *Inverse Kinematics* va prendre en entrée cette trajectoire centroïdale ainsi que la séquence de contacts et va produire un mouvement corps complet $\mathbf{q}(t)$ avec une méthode de cinématique inverse du second ordre. Ce bloc doit également prendre en entrée une trajectoire pour chaque organe terminal en contact.

Afin de produire des trajectoires sans collisions pour les organes terminaux, nous proposons un nouveau bloc dédié qui va fonctionner par itérations successives avec la cinématique inverse. Lors de la première itération, le bloc *End effector trajectory* donne une trajectoire par défaut pour les organes terminaux sans considérer l'environnement. Ensuite, le mouvement produit par la cinématique inverse est vérifié pour tester les collisions et les butées articulaires. Si une partie du mouvement s'avère invalide, de nouvelles trajectoires sont générées pour les organes terminaux pour cette partie du mouvement, en prenant cette fois l'environnement en compte.

Ensemble, ces derniers blocs résolvent le sous-problème $\mathcal{P}_3$.

Implémentation

Les connexions entre chacun de ces blocs sont assurées par un script python utilisant les API python de chaque méthode. Les objets figurant sur les flèches du schéma 6.2 sont tous représentés par des classes python ou C++ avec une API python exposée.

L'exception est la connexion avec la méthode de génération de trajectoire centroïdale. En effet, l'API actuelle nous impose de communiquer via des fichiers : un fichier formulant le problème doit être donné en entrée et la trajectoire solution est écrite dans un fichier en sortie. L'écriture et la lecture de ces fichiers sont automatiquement gérées par ce script.

Détails sur les différentes trajectoires centroïdales

Dans l'architecture utilisée (figure 6.2), 3 trajectoires centroïdales différentes sont calculées. La figure 6.3 compare ces différentes trajectoires pour un exemple de mouvement.

La courbe $\mathbf{x}(t)_{planning}$ est une estimation grossière de la trajectoire centroïdale planifiée par la méthode RB-RRT kinodynamique, elle n'est valide qu'avec les approximations faites sur les positions des contacts (que nous avons appelé *contact*

glissant) et sur la position du centre de masse (confondu avec le centre géométrique) présentées dans la section 2.3 de la première partie. Il s'agit du guide qui est utilisé pendant la génération de contact.

La courbe $\mathbf{x}(t)_{initGuess}$ est produite par CROC lors de la validation de la séquence de contacts. Elle correspond à une trajectoire valide du point de vue de la dynamique centroïdale pour les contacts choisis. Les trajectoires calculées par CROC ne considèrent que 3 phases de contact successives, et non pas l'ensemble de la trajectoire. La courbe présentée sur la figure 6.3 est donc une concaténation de plusieurs trajectoires produites par CROC. Bien que cette concaténation forme une trajectoire valide, elle peut être raffinée par une méthode considérant l'ensemble des phases de contact. La concaténation des trajectoires fournies par CROC est donc utilisée comme solution initiale de la méthode de génération de trajectoire centroïdale. Une discussion plus approfondie des raisons de la sous-optimalité de cette trajectoire est présentée dans la section 4.7.4.

Sur la figure 6.3, la trajectoire $\mathbf{x}(t)_{initGuess}$ est donc la concaténation de trois trajectoires trouvées par CROC, entre chacun des points oranges et les états initial et final. la position de ces points de jonction est fixe et donnée par la méthode de génération de contact. Cela peut résulter en des trajectoires avec des déplacements inutiles, comme montré sur la figure.

Enfin, $\mathbf{x}(t)$ est le résultat de la méthode de génération de trajectoire centroïdale [Carpentier 2017c]. Contrairement à la méthode CROC, la méthode de génération de trajectoire optimise l'ensemble de la trajectoire en un seul problème car il n'y a pas de points intermédiaires fixés, ce qui résulte en une trajectoire plus lisse.

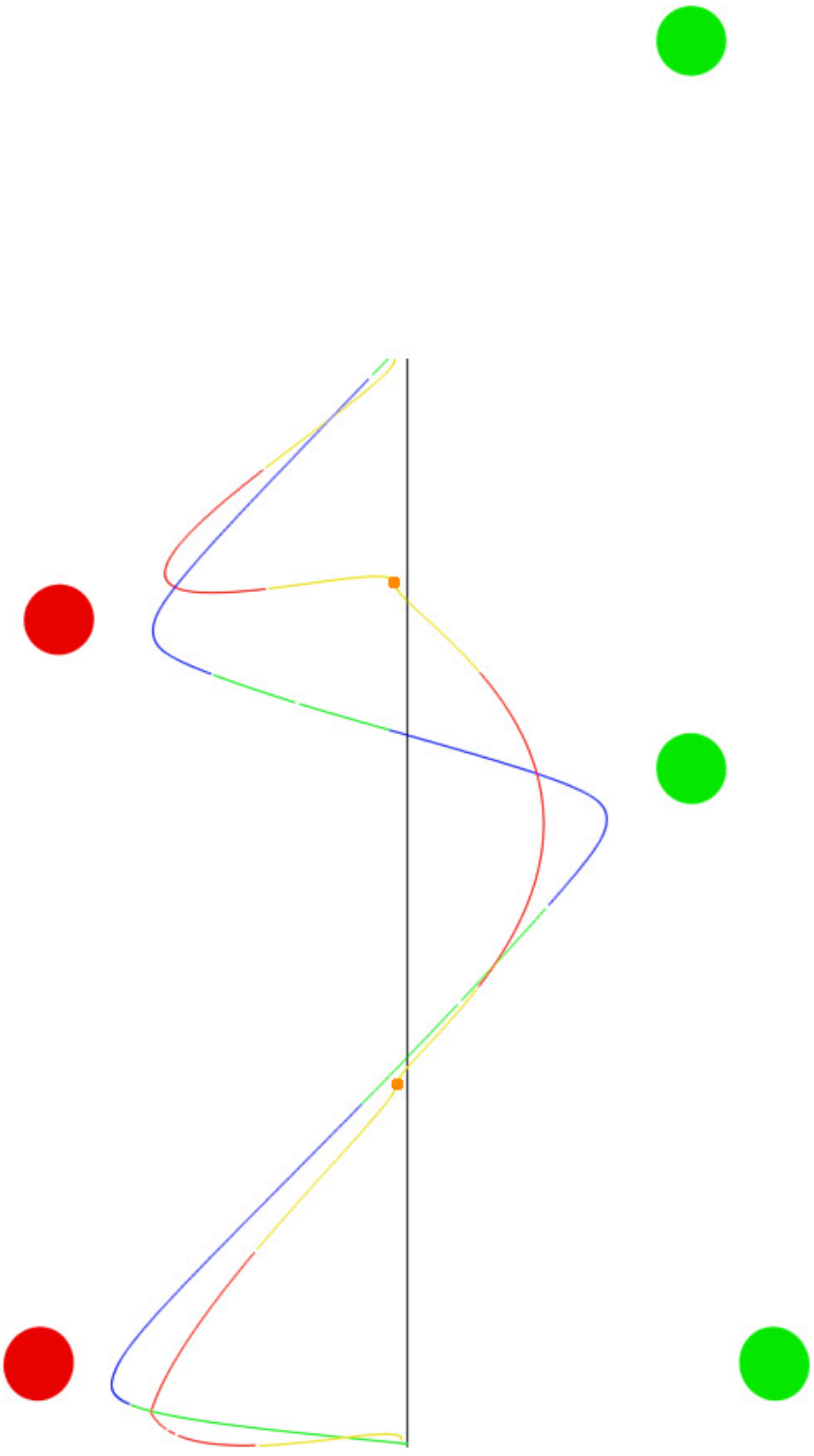


FIGURE 6.3 – Exemple des trois types de trajectoires centroïdales utilisées, projetées dans le plan (x, y), dans un cas de marche bipède sur sol plat. Les cercles rouge et vert représentent les positions des pieds gauche et droit. La ligne noire est $\mathbf{x}(t)_{planning}$. La courbe jaune et orange est $\mathbf{x}(t)_{initGuess}$ avec en jaune les phases de double support et en orange les phases de simple support, les points oranges sont les points de jonction entre chaque trajectoires trouvées par CROC. La courbe verte et bleu est $\mathbf{x}(t)$ avec en vert les phases de double support et en bleu les phases de simple support.

6.2 Planification d'une trajectoire guide

Nous rappelons que la méthode RB-RRT kinodynamique présentée dans le chapitre 2 planifie une trajectoire dans un espace de faible dimension en utilisant un modèle réduit du robot. Cette méthode prend en entrée le modèle du robot et de l'environnement, des bornes de vitesse et d'accélération maximales pour le centre du modèle réduit et des positions et orientations initiales et finales définissant le problème de planification.

La trajectoire guide est planifiée pour la position et l'orientation de l'origine du modèle réduit du robot ainsi que pour les n_{trunk} degrés de liberté du tronc du robot : $\mathbf{q}^{trunk} \in SE(3) \times \mathbb{R}^{n_{trunk}}$, avec $\mathbf{q}_{trunk}$ une configuration du modèle réduit. Comme nous utilisons une méthode kinodynamique, nous planifions également dans l'espace d'état avec la vitesse et l'accélération du centre du robot : $\dot{\mathbf{c}} \in \mathbb{R}^3$ et $\ddot{\mathbf{c}} \in \mathbb{R}^3$.

Comme la position du centre du modèle réduit $\mathbf{c}$ est uniquement définie par $\mathbf{q}^{trunk}$, nous avons bien une trajectoire centroïdale sous sa forme classique $\mathbf{x}(t) =< \mathbf{c}, \dot{\mathbf{c}}, \ddot{\mathbf{c}} >$. Au final, la sortie de cette méthode est une trajectoire guide définie par :

$$< \mathbf{x}_{planning}(t) =< \mathbf{c}_{planning}(t), \dot{\mathbf{c}}_{planning}(t), \ddot{\mathbf{c}}_{planning}(t) >, \mathbf{q}^{trunk}(t) > \tag{6.1}$$

Implémentation :

Ce planificateur a été implémenté en C++ avec une API python dans le paquet HPP-RBPRM [1] de la suite logicielle HPP (Humanoid Path Planner) [2][Mirabel 2016].

La connexion entre le planificateur de trajectoire guide et le planificateur de contact se fait directement dans la suite HPP en C++.

6.3 Planification de contacts

La méthode de planification de contacts considère le modèle complet du robot. Elle prend en entrée une trajectoire guide $< \mathbf{x}(t), \mathbf{q}^{trunk}(t) >$ et va générer une séquence de configurations corps complet, en contact avec l'environnement et en équilibre, qui permettent de suivre la trajectoire guide [Tonneau 2018a].

Afin de planifier cette séquence de configurations, le guide $< \mathbf{x}(t), \mathbf{q}^{trunk}(t) >$ est discrétisé. Ensuite, à partir d'une configuration corps complet et d'une phase de contact initiale $\mathbf{q}_0^{\{0\}}$, pour chaque configuration discrétisée du modèle réduit $\mathbf{q}_i^{trunk}$ une configuration corps complet associée à une phase de contact $\mathbf{q}_i^{\{k\}}$ va être générée par le générateur de contact.

Nous rappelons qu'une phase de contact $\{k\}$, $k \in \mathbb{N}$ définit la position et la normale de chaque point de contact. Ici, la consistance entre la configuration $\mathbf{q}_i$, la

1. `https://github.com/humanoid-path-planner/hpp-rbprm`
2. `https://humanoid-path-planner.github.io/hpp-doc/`

phase de contact $\{k\}$ et l'environnement doit être assurée. C'est à dire que lorsque l'on note $\mathbf{q}_i^{\{k\}}$, il faut que les positions des points de contact définies par la phase $\{k\}$ correspondent aux positions des contacts entre le robot et l'environnement quand le robot est dans la configuration $\mathbf{q}_i$. De plus, les normales au contact associées à ces positions doivent correspondre aux normales de la surface de l'environnement en contact.

C'est pourquoi, durant la planification de contact les phases de contact sont définies automatiquement à partir d'une configuration et d'une liste d'organes terminaux en contact.

Le générateur de contact prend en entrée la configuration corps complet et la phase de contact précédente $\mathbf{q}_{i-1}^{\{j\}}$, la configuration du modèle réduit courante $\mathbf{q}_i^{trunk}$ et l'état centroïdal courant $\mathbf{x}_i$.

Il produit en sortie une configuration corps complet et une phase de contact $\mathbf{q}_i^{\{k\}}$ telle que les degrés de libertés communs entre le modèle réduit et le modèle complet sont identiques. Autrement dit, le générateur de contact conserve les valeurs de la position et l'orientation de l'origine du robot ainsi que les valeurs des degrés de liberté du tronc fournies par le guide et génère des configurations pour chaque pattes du robot.

De plus, le générateur de contact suit les règles suivantes :

— Une patte en contact pour $\mathbf{q}_{i-1}^{\{j\}}$ est maintenue en contact pour $\mathbf{q}_i^{\{k\}}$ si les contraintes cinématiques peuvent être respectées ;
— $\mathbf{q}_i$ est sans collision avec l'environnement ni en auto-collision ;
— $\mathbf{x}_i$ satisfait les contraintes dynamiques de non glissement imposées par la phase de contact $\{k\}$;
— au maximum **un** contact n'est pas maintenu (supprimé) entre $\mathbf{q}_{i-1}^{\{j\}}$ et $\mathbf{q}_i^{\{k\}}$;
— au maximum **un** contact est créé (ajouté) entre $\mathbf{q}_{i-1}^{\{j\}}$ et $\mathbf{q}_i^{\{k\}}$;

Le générateur de contact fonctionne en utilisant une liste de configurations candidates classées à l'aide d'une heuristique, qu'il évalue jusqu'à trouver un candidat satisfaisant les règles ci-dessus. Les détails des méthodes utilisées par le générateur de contact se trouvent en annexe B, en reprenant l'algorithme proposé dans [Tonneau 2018a].

A partir de la séquence de configurations en contact planifiée, nous pouvons extraire une séquence de contacts telle que chaque phase de contact diffère des phases adjacentes par exactement **un** changement de contact (un ajout ou une suppression). Pour parvenir à cela, il faut supprimer toutes les configurations adjacentes ayant une phase de contact identique, et ajouter les éventuelles phases de contact intermédiaires.

Implémentation :

Le générateur de contact est également implémenté en C++ dans le paquet HPP-RBPRM [3]. Les tests de collisions sont réalisés avec la bibliothèque Flexible Collision Library (FCL) [4]. Les tests d'équilibre sont réalisés avec la bibliothèque centroïdal-dynamics-lib [5]. Toutes ces bibliothèques sont implémentées en C++.

6.4 Intégration du test de faisabilité dans le planificateur de contact

L'intégration de la méthode CROC, présentée dans la seconde partie de ce manuscrit, comme critère de faisabilité durant la planification de contact a été réalisée simplement en ajoutant la règle suivante dans le générateur de contact décrit dans la section précédente :

La transition entre $\mathbf{x}_{i-1}^{\{j\}}$ *et* $\mathbf{x}_{i}^{\{k\}}$ *doit être faisable.*

Afin de vérifier cela lors de la création de contact, en plus de tester chaque candidat pour les collisions et les contraintes de non glissement comme décrit dans l'annexe B.4, on teste également si la transition entre l'état précédent et l'état candidat est faisable.

Nous avons donc l'état associé à une phase de contact précédent fixé $\mathbf{x}_{i-1}^{\{j\}}$ et un état candidat $\mathbf{x}_{i}^{\{k\}}$. D'après les règles du générateur de contact, nous avons $j \leq k \leq j+2$. Nous avons donc au maximum une séquence de 3 phases de contact. Si $j \neq k$ nous utilisons directement CROC pour vérifier la faisabilité de la transition de contact entre l'état précédent et le candidat.

Si CROC converge, nous avons la garantie que la transition est faisable. Si CROC ne converge pas nous n'avons aucune garantie car CROC utilise une formulation conservatrice. Cependant, nous considérons dans ce cas que la transition n'est pas faisable. Si la transition n'est pas garantie comme faisable, le candidat est classé comme *invalide* et un nouveau candidat est testé.

Implémentation :

Nous utilisons la bibliothèque Spline [6] développée au sein de l'équipe pour représenter les courbes de Bézier. L'implémentation de la méthode CROC est faite dans la bibliothèque bezier_COM_traj [7]. Ces deux bibliothèques sont implémentées en C++ avec une interface python.

Les problèmes LP et QP sont résolus via la bibliothèque QuadProg. Les LP de grande dimension avec des matrices creuses sont résolus avec GLPK.

3. `https://github.com/humanoid-path-planner/hpp-rbprm`
4. `https://github.com/flexible-collision-library/fcl`
5. `https://github.com/stonneau/centroidal-dynamics-lib`
6. `https://github.com/stonneau/spline`
7. `https://gitlab.com/stonneau/bezier_COM_traj`

6.5 Génération de trajectoire centroïdale

La méthode de génération de trajectoire centroïdale utilisée est celle présentée dans [Carpentier 2017c]. Cette méthode prend en entrée une séquence de contacts avec une valeur initiale pour la durée de chaque phase de contact et une solution initiale pour la trajectoire centroïdale. Elle donne en sortie une trajectoire centroïdale valide et optimale ainsi que les durées de chaque phases de contact.

Cette méthode formule un problème de contrôle optimal non linéaire qui prend en compte les contraintes dynamiques du modèle centroïdal, avec une fonction de coût spécifique. La fonction de coût contient entre autre un coût permettant de maximiser la faisabilité cinématique de la trajectoire centroïdale produite.

Ce problème de contrôle optimal est résolu grâce à une méthode de *multiple shooting*.

Implémentation :

La méthode de *multiple shooting* est implémentée par la bibliothèque MUSCOD-II [Leineweber 2003]. Un programme permettant de formuler automatiquement un problème de contrôle optimal à partir d'une séquence de contacts, dans un format utilisable par MUSCOD, a été écrit par les auteurs de [Carpentier 2017c] puis nous l'avons adapté et rendu plus générique pour notre utilisation.

Le problème de génération de trajectoire centroïdale n'étant pas le cœur de cette thèse, nous utilisons actuellement une implémentation peu optimisée de cette méthode dans notre architecture. Les temps de calcul mesurés sont supérieurs à ceux présentés dans [Carpentier 2017c] mais les solutions trouvées sont identiques.

Perspectives futures

L'utilisation de la bibliothèque MUSCOD-II pose plusieurs problèmes. Tout d'abord, il s'agit d'une bibliothèque propriétaire nécessitant une licence. Ensuite, il s'agit d'un programme très complexe nécessitant le réglage minutieux de nombreux paramètres sensibles.

Pour ces raisons, nous voulons étudier deux pistes afin de pouvoir nous passer de cette bibliothèque. D'une part, l'utilisation de CROC directement comme méthode de génération de trajectoire centroïdale.

D'autre part, l'intégration des travaux proposés dans [Ponton 2018]. L'intérêt de cette méthode est de pouvoir optimiser efficacement la durée de chaque phase de contact. De plus, leur reformulation convexe permet un gain en terme de temps de calcul par rapport aux méthodes non-linéaires.

6.6 Cinématique inverse

La méthode de cinématique inverse prend en entrée la séquence de contacts, la trajectoire centroïdale et des trajectoires pour chacun des organes terminaux en

contact. Cette méthode génère en sortie une trajectoire articulaire $\mathbf{q}(t)$ qui donne l'évolution des valeurs de tous les degrés de liberté du robot, afin de produire un mouvement qui respecte la séquence de contacts et les trajectoires de référence données en entrée.

Cette méthode est une cinématique inverse du second ordre similaire à la méthode proposée dans [Saab 2013]. Elle fonctionne en définissant une pile de tâches hiérarchisées, chaque tâche est alors optimisée dans l'espace nul des tâches de priorité plus importante. C'est à dire que l'optimisation d'une tâche se fera toujours sans interférer avec les tâches de plus haute priorité.

Dans l'implémentation choisie, les tâches sont les suivantes (par ordre de priorité) :

— La position des organes terminaux en contact pour la phase de contact courante est fixe ;
— La trajectoire du centre de masse suit la référence donnée en entrée ;
— Les trajectoires des organes terminaux suivent les références données en entrée ;
— la distance à une configuration de référence est minimisée.

La dernière tâche est appelée une *tâche posturale*, elle permet de limiter les mouvements inutiles du robot et d'obtenir un mouvement esthétiquement plus agréable.

Implémentation :

Cette méthode de cinématique inverse a été implémentée en python. Une version C++ appelée TSID[8] a été implémentée au sein de l'équipe, son utilisation dans l'architecture de planification est prévue dans le futur.

6.7 Génération de trajectoires pour les pattes

La méthode de cinématique inverse servant à générer un mouvement corps complet doit prendre en entrée des trajectoires de référence pour les organes terminaux en contact. Pour chaque phase où un organe terminal est repositionné entre deux points de contact, il faut fournir une trajectoire 6D (position et orientation 3D) de référence pour cet organe terminal.

Afin de pouvoir planifier la locomotion de robots à pattes dans des environnements contraints, nous devons être capable de générer des trajectoires *valides* pour les organes terminaux automatiquement car nous ne pouvons pas nous appuyer sur des trajectoires conçues manuellement. Une trajectoire *valide* doit respecter les contraintes cinématiques du robot et toute la patte considérée doit être sans collision pendant que l'organe terminal suit cette trajectoire.

De plus, afin de limiter les risques de chutes dus aux incertitudes et parce que nous considérons des contacts planaires, la trajectoire doit être colinéaire à la normale de la surface en contact pendant un petit laps de temps après le décollage et

8. `https://github.com/stack-of-tasks/tsid/`

avant la pose de l'organe terminal, comme montré sur la figure 6.4. Cela permet d'assurer que toute la surface de l'organe terminale entre en contact avec l'environnement au même instant. L'impact entre le sol et l'organe terminal doit également être fait à vitesse et accélération nulle. Enfin, la trajectoire doit respecter des contraintes en vitesse, accélération et jerk (la dérivée de l'accélération) afin d'être réalisable par le robot.

FIGURE 6.4 – Exemple de trajectoire respectant les contraintes imposées. En rouge la trajectoire de l'organe terminal dans le plan (x, z). En noir en trait plein les surfaces de contact en noir pointillé l'instant ou la trajectoire n'est plus contrainte à être colinéaire à la normale de la surface de contact.

On trouve dans la littérature beaucoup de travaux utilisant des trajectoires de référence pour les organes terminaux conçues manuellement. Par exemple, [Naveau 2016] utilise des B-Splines permettant d'assurer une vitesse et une accélération nulles au moment de la pose du pied, mais les points de contrôle de ces B-Splines doivent être ajustés manuellement pour chaque scénario. [Verrelst 2006] utilise des *Clamped Cubic Spline* afin de générer des trajectoires d'enjambement d'obstacles. Ces trajectoires sont générées à partir des primitives géométriques des obstacles mais ne fonctionnent que pour des obstacles rectangulaires sur du sol plat. De plus, il n'y a aucune garantie que la patte entière soit sans collision.

D'autres travaux [Focchi 2013, García Armada 2003] détectent la collision entre la patte en mouvement et l'environnement en ligne et appliquent une stratégie de "mouvement réflexe" consistant à adapter la hauteur du pas afin de passer l'obstacle. Il existe également des travaux basés sur des méthodes d'optimisation, comme [Kalakrishnan 2011], qui peuvent générer des trajectoires sans collisions mais se basent sur l'utilisation de la fonction distance entre deux objets qui présente des inconvénients. En effet, cette fonction n'est pas différentiable dans le cas général et il faut utiliser des algorithmes spécifiques pour calculer les deux points les plus proches entre deux objets. [Brossette 2017] propose une méthode permettant d'éviter l'utilisation de cette fonction de distance, à condition de représenter l'environnement par un ensemble de formes convexes. Cependant, elle ne permet pas de prendre en compte l'ensemble des contraintes cinématiques de la jambe.

Enfin, certaines méthodes [Nishiwaki 2014, Qiu 2012] planifient le mouvement de la patte via des algorithmes probabilistes (comme le *Bi-RRT*). Cela garantit un mouvement respectant toutes les contraintes cinématiques et de non collision. Cependant, ces algorithmes ne considèrent que des contraintes géométriques et ne

peuvent donc pas garantir les contraintes sur la dynamique de la trajectoire décrites plus haut. De plus, ils vont produire des trajectoires engendrant des mouvements inutiles pour la patte du robot et avec de possibles discontinuités dans la vitesse ou les dérivées supérieures. Enfin, elles requièrent beaucoup de temps de calcul.

Dans cette section, nous proposons une méthode à mi-chemin entre la planification, qui ne peut considérer les contraintes sur la dynamique de la trajectoire et qui produit des mouvements non lisses, et les méthodes basées sur l'optimisation, qui requièrent des modèles de collisions convexes et qui ne considèrent pas toutes les contraintes cinématiques. Cette méthode repose sur l'utilisation d'une trajectoire par défaut pour chaque type de patte, caractérisée par des paramètres définis manuellement. Cette trajectoire par défaut sera ensuite déformée via une méthode d'optimisation afin d'éviter les obstacles, en utilisant un coût basé sur la solution d'une méthode de planification géométrique.

Concevoir manuellement des trajectoires respectant toutes les contraintes énumérées plus haut peut être long et laborieux, de plus il faut les adapter manuellement à chaque nouvel environnement contraint. Cependant, il peut être utile de concevoir une trajectoire de référence *valide* qui ne considère pas l'environnement et qui est efficace pour de la marche sur sol plat. Cette trajectoire peut alors être utilisée par défaut dans tous les cas où elle reste *valide*. Comme décrit dans la présentation de l'architecture de planification, cette trajectoire par défaut sera la première à être utilisée par la méthode de cinématique inverse et elle ne sera modifiée que si elle s'avère invalide.

Afin de vérifier si une trajectoire d'organe terminal est valide, il faut vérifier si elle donne lieu à un mouvement corps complet qui respecte les butées articulaires et est sans collision ni auto-collision. Pour cela, la méthode de validation de trajectoire de la figure 6.2 vérifie le mouvement corps complet et indique les portions de trajectoires et les pattes qui ne sont pas valides.

Pour les phases où le mouvement d'une patte n'est pas valide, nous utilisons un algorithme nommé *Limb-RRT* pour produire un chemin pour l'organe terminal tel que le mouvement de la patte qui suit ce chemin soit sans collision et respecte les butées articulaires. Comme il s'agit d'un chemin planifié par un algorithme probabiliste qui ne considère que des contraintes géométriques, il ne sera pas utilisé directement comme référence pour l'organe terminal. Nous allons alors progressivement déformer la trajectoire par défaut pour la faire correspondre au chemin trouvé par le *Limb-RRT* jusqu'à ce que la trajectoire résultante permette de générer un mouvement valide.

Ces méthodes sont présentées dans les sous-sections suivantes.

6.7.1 Trajectoire par défaut

La trajectoire par défaut est la première trajectoire testée pour la génération de mouvement corps complet. Elle doit être générée rapidement en fonction des positions et orientations initiale et finale de l'organe terminal en contact pour chaque

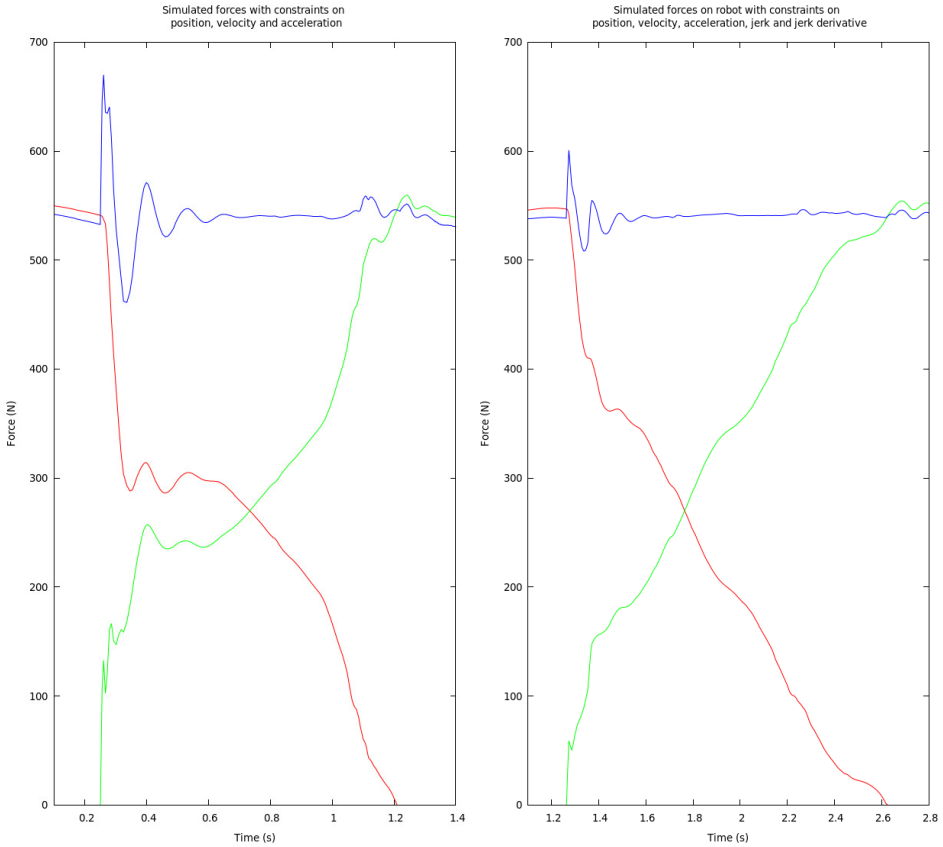
phase où un organe terminal doit se déplacer. Cette trajectoire est conçue pour être efficace pour de la marche sur sol plat, mais peut être utilisée tant qu'elle permet d'obtenir un mouvement qui respecte les contraintes cinématiques.

Nous avons choisi de représenter les trajectoires des organes terminaux par des courbes de Bézier. Notre trajectoire est composée de trois courbes : une pour le décollage du pied, une pour la pose du pied et une entre les deux, comme montré sur la figure 6.4.

Les trajectoires de décollage et de pose du pied doivent satisfaire certaines contraintes. En effet, afin d'assurer que toute la surface du pied entre en contact (respectivement quitte le contact) avec l'environnement au même instant, il faut assurer que la vitesse et l'accélération pendant ces phases sont colinéaires à la normale de la surface en contact comme montré sur la figure 6.4. Cela permet également de minimiser les risques dus aux incertitudes pendant le mouvement. De plus, la trajectoire de décollage (respectivement de pose du pied) doit avoir une accélération et une vitesse initiale (respectivement finale) nulle.

Des expériences réalisées sur le robot HRP-2 ont montré que nous pouvions obtenir un mouvement beaucoup plus lisse si nous contraignions également la dérivée et la dérivée seconde de l'accélération. Les résultats de ces tests sont montrés par les courbes des figures 6.5. Ces courbes représentent les forces exercées sur les pieds du robot, selon l'axe z. La somme des forces des deux pieds devrait être constante durant le mouvement puisque le poids du robot reste constant. Or, on peut observer des oscillations de la valeur de la somme lors de la pose du pied et du transfert du poids d'un pied à l'autre. Dans le cas où l'on ne contraint que la vitesse et l'accélération, cette oscillation est bien plus importante. Cela est dû à l'impact entre le pied et le sol, lors de la pose du pied, qui est plus violent dans ce cas car le robot n'a pas été capable de suivre exactement la trajectoire de référence car elle présentait des valeurs des dérivées de l'accélération trop élevées. Dans le cas où l'on contraint également la dérivée et la dérivée seconde de l'accélération, l'impact est bien moindre et les oscillations minimes car la trajectoire de référence peut être suivie plus fidèlement. Les courbes 6.6 montrent que, à l'exception du bruit, les courbes simulées et mesurées sur le robot réel sont semblables.

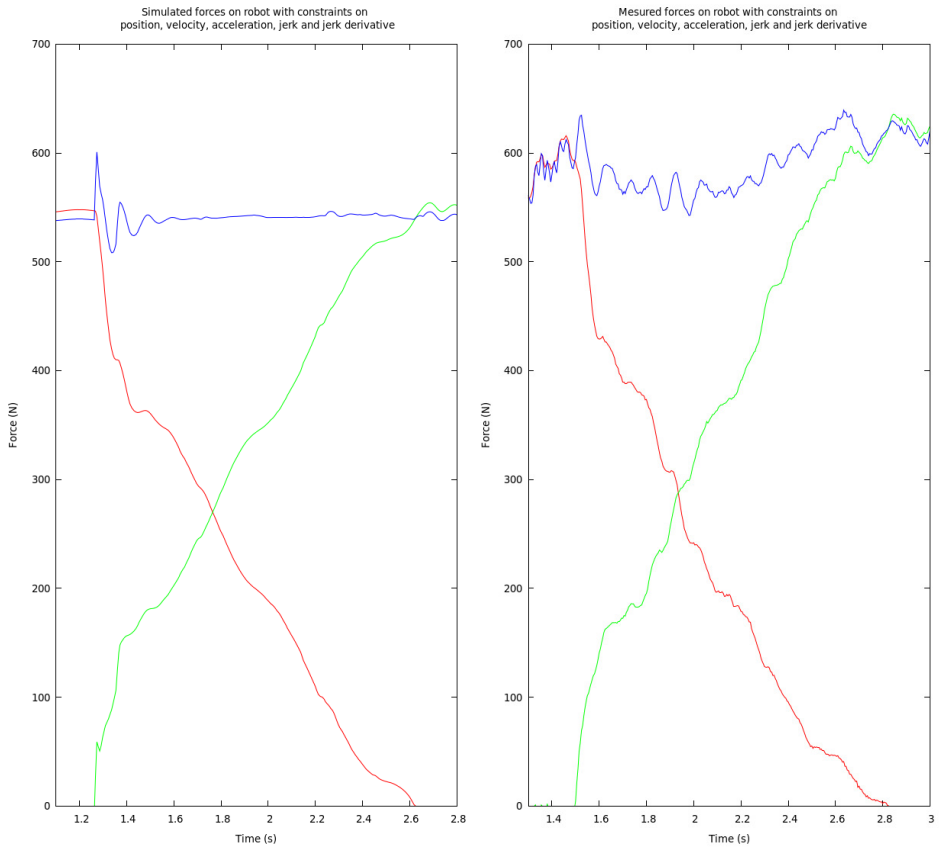
Au final, nos phases de décollage et de pose du pied sont générées en imposant une dérivée seconde de l'accélération constante, colinéaire à la normale de la surface de contact et avec une amplitude et une durée définies manuellement. Ensuite, les points de contrôle de la courbe de Bézier sont calculés automatiquement pour satisfaire ces deux contraintes.



(a) Avec des contraintes sur la vitesse et l'accélération.

(b) Avec des contraintes sur la vitesse, l'accélération, le jerk et la dérivée du jerk.

FIGURE 6.5 – Courbe des forces appliquées selon l'axe z sur les pieds de HRP-2 pendant un pas sur du sol plat, générées avec le simulateur Open-HRP. En rouge le pied gauche, en vert le pied droit, en bleu la somme des deux.



(a) Simulé sur Open-HRP. (b) Mesuré sur le robot réel.

FIGURE 6.6 – Courbe des forces appliquées selon l'axe z sur les pieds de HRP-2 pendant un pas sur du sol plat, générées avec le simulateur Open-HRP ou mesurées sur le robot réel. En rouge le pied gauche, en vert le pied droit, en bleu la somme des deux.

La trajectoire du milieu, entre les phases de décollage et de pose du pied, est représentée par une courbe de Bézier de degré 8 : les positions, vitesses, accélérations et dérivés de l'accélération initiales et finales sont contraintes à être égales aux valeurs finales et la trajectoire de décollage et initiales de la trajectoire de pose du pied. Cela afin de garantir la continuité jusqu'à l'ordre 4 de la trajectoire complète. Ensuite, un point de contrôle libre $\mathbf{P_4} = \mathbf{y}$ est ajouté permettant d'agir sur la forme courbe. La trajectoire peut alors s'exprimer ainsi :

$$\mathbf{eff}(t, \mathbf{y}) = \sum_{i \in \{0,1,2,3,5,6,7,8\}} B_i^8(t/T)\mathbf{P}_i + B_4^8(t/T)\mathbf{y} \tag{6.2}$$

Avec B les polynômes de Bernstein, $\mathbf{P}_i$ les points de contrôle de la courbe et T la durée totale de la trajectoire, comme défini dans la section 4.1.

En utilisant le fait qu'une courbe de Bézier est comprise dans l'enveloppe convexe de ses points de contrôle, nous pouvons aisément imposer des contraintes sur la vitesse, l'accélération ou la dérivée de l'accélération de l'organe terminal comme fait dans le chapitre 5. Ces contraintes se représentent de la manière suivante :

$$\mathbf{O}\mathbf{y} \leq \mathbf{o} \tag{6.3}$$

Enfin pour obtenir une trajectoire la plus lisse possible, nous pouvons optimiser la position du point de contrôle libre $\mathbf{y}$ afin de minimiser un coût donné. Après plusieurs expérimentations, nous avons choisi de minimiser la norme au carré de la vitesse de l'organe terminal car cela va tendre à minimiser le moment cinétique généré par la patte en mouvement. Le coût est donc :

$$l(\mathbf{y}) = \int_0^T ||\dot{\mathbf{eff}}(t, \mathbf{y})||^2 dt \tag{6.4}$$

Grâce à la formulation des courbes de Bézier, nous pouvons calculer cette primitive analytiquement et l'évaluer aux extrémités, ce qui nous permet de formuler ce coût de manière continue et exacte et de le représenter sous la forme quadratique suivante :

$$l(\mathbf{y}) = \mathbf{y}^\top \mathbf{H}_s \mathbf{y} + 2\mathbf{g}_s^\top \mathbf{y} \tag{6.5}$$

Les expressions de $\mathbf{H}_s$ et $\mathbf{g}_s$ sont calculées analytiquement avec la même méthode que celle présentée en annexe A.7.

Au final, on peut formuler un QP pour trouver la position optimale de $\mathbf{y}$ comme fait dans le chapitre 4 :

$$\begin{aligned} \textbf{min} \quad & \mathbf{y}^\top \mathbf{H}_s \mathbf{y} + 2\mathbf{g}_s^\top \mathbf{y} \\ \text{subject to} \quad & \mathbf{O}_{vel}\mathbf{y} \leq \mathbf{o}_{vel} \\ & \mathbf{O}_{acc}\mathbf{y} \leq \mathbf{o}_{acc} \\ & \mathbf{O}_{jerk}\mathbf{y} \leq \mathbf{o}_{jerk} \end{aligned} \tag{6.6}$$

Des exemples de trajectoires par défaut sont montrés sur la figure 6.7. Ces exemples sont fait pour une marche sur sol plat, la normale au contact est donc alignée avec l'axe z. Le mouvement durant les phases de décollage et de pose du pied ne se fait donc que selon l'axe z. Pour ces exemples avec les jambes de HRP 2 la durée de ces phases est fixée à 0.3 secondes.

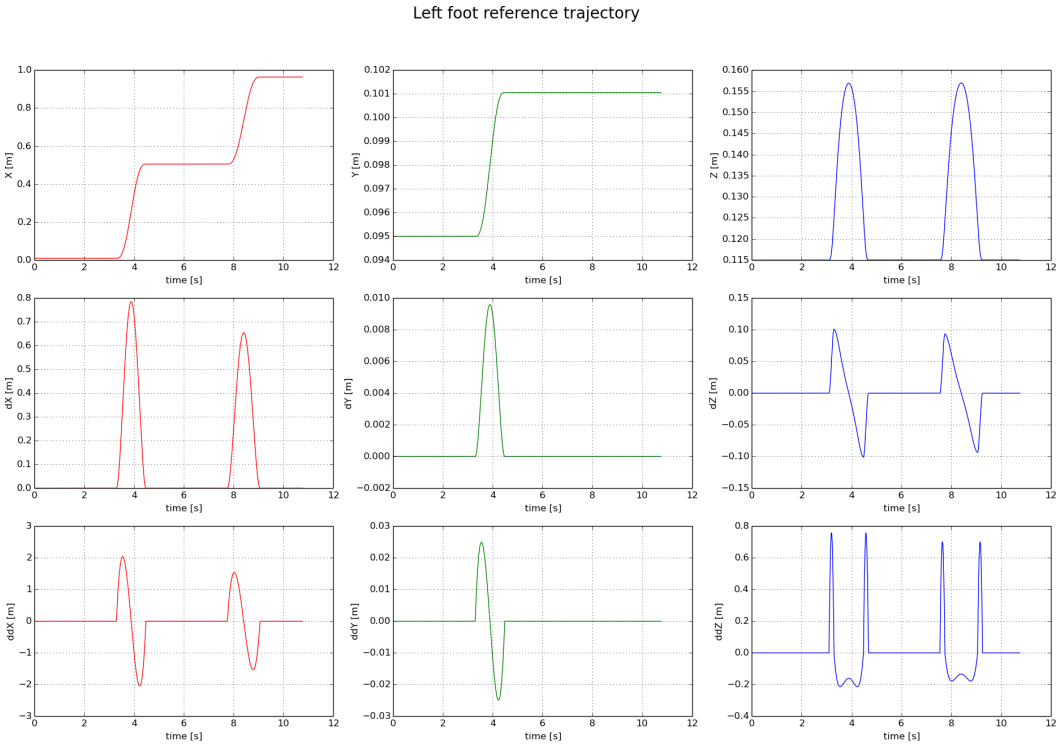


FIGURE 6.7 – Exemples de trajectoires par défaut, pour deux pas sur sol plat. Les lignes sont (de haut en bas) la position, la vitesse en l'accélération de l'organe terminale et les colonnes sont (de gauche à droite) les axes x, y, z.

6.7.2 Limb-RRT

L'algorithme *Limb-RRT* est une adaptation de l'algorithme original introduit par [Qiu 2012]. Cet algorithme est utilisé pour chaque phase où un organe terminal doit être repositionné d'une position en contact à une autre et où la trajectoire par défaut est invalide. Il prend en entrée les configurations corps complet initiale et finale ainsi que la trajectoire centroïdale entre ces deux configurations calculée par la méthode de génération de trajectoire.

L'algorithme produit en sortie un mouvement corps complet cinématiquement valide et sans collision qui connecte les deux configurations et qui suit la trajectoire centroïdale fournie en entrée.

Pour réaliser cela, un RRT orienté [LaValle 2006] est utilisé pour la patte en mouvement en contraignant le centre de masse du robot à suivre la trajectoire

centroïdale de référence. Ce RRT considère toutes les contraintes cinématiques et de non collision du modèle complet du robot.

Après application des méthodes d'optimisation classiques pour les planificateurs probabilistes (par exemple le raccourci aléatoire ou le raccourci partiel, une étude de ces différentes méthodes est présentée dans [Geraerts 2007]) la trajectoire 6D de l'organe terminal est extraite à partir du mouvement corps complet. Cependant, comme tout planificateur probabiliste, le *Limb-RRT* va produire une trajectoire peu lisse et pouvant présenter des mouvements inutiles même après utilisation de méthodes d'optimisation probabilistes, comme montré par la trajectoire noire sur les figures 6.8 et 6.9. De plus, le *Limb-RRT* est une méthode seulement géométrique, le chemin produit peut donc présenter des discontinuités de vitesse ou d'accélération.

A cause de tout cela, nous avons constaté que cette trajectoire de l'organe terminal ne pouvait pas servir de référence pour le robot réel car il sera incapable de la suivre exactement.

6.7.3 Déformation de la trajectoire par défaut

Afin de résoudre le problème précédent et de pouvoir utiliser la trajectoire trouvée par le *Limb-RRT* comme une référence que nous savons cinématiquement valide et sans collision, nous proposons une méthode d'optimisation de trajectoire. Cette méthode va progressivement déformer la trajectoire par défaut afin de la faire correspondre à la trajectoire trouvée par la méthode *Limb-RRT*. Pour parvenir à cela, nous définissons une nouvelle fonction de coût : l'intégrale de la distance à la solution du *Limb-RRT* notée $\mathbf{s}(t)$:

$$l(\mathbf{y}) = \int_0^T ||\mathbf{eff}(t, \mathbf{y}) - \mathbf{s}(t)||^2 dt \tag{6.7}$$

En remplaçant $\mathbf{eff}(t, \mathbf{y})$ par son expression dans l'équation (6.2), nous pouvons réécrire ce coût sous la forme quadratique suivante (via un développement similaire à celui de l'annexe A.7) :

$$l(\mathbf{y}) = \mathbf{y}^\top \mathbf{H}_d \mathbf{y} + 2\mathbf{g}_d^\top \mathbf{y} \tag{6.8}$$

Comme la trajectoire $\mathbf{s}(t)$ trouvée par le *Limb-RRT* est garantie d'être cinématiquement valide, si cette fonction de coût devient nulle la trajectoire $\mathbf{eff}(t, \mathbf{y})$ est également garantie d'être cinématiquement valide. Cependant, cela peut résulter en une trajectoire avec des mouvements inutiles et difficiles à suivre par le robot réel car elle présenterait une accélération ou un jerk important.

Nous allons donc procéder en définissant un nouveau coût :

$$\begin{aligned} \mathbf{H}(\alpha) &= \alpha \mathbf{H}_d + (1-\alpha)\mathbf{H}_s \\ \mathbf{g}(\alpha) &= \alpha \mathbf{g}_d + (1-\alpha)\mathbf{g}_s \end{aligned} \tag{6.9}$$

Où $\alpha \in [0; 1]$ est un poids qui définit le compromis entre le fait que la trajectoire soit lisse et sa correspondance avec la solution cinématiquement faisable

du *Limb-RRT*. $\mathbf{H}_s$ et $\mathbf{g}_s$ définissent le coût utilisé pour la trajectoire par défaut (dans l'équation (6.5)). Quand la trajectoire par défaut n'est pas valide, nous allons procéder itérativement en générant des trajectoires avec le QP suivant :

$$\begin{aligned} \textbf{min} \quad & \mathbf{y}^\top \mathbf{H}(\alpha)\mathbf{y} + 2\mathbf{g}(\alpha)^\top \mathbf{y} \\ \text{subject to} \quad & \mathbf{O}_{vel}\mathbf{y} \leq \mathbf{o}_{vel} \\ & \mathbf{O}_{acc}\mathbf{y} \leq \mathbf{o}_{acc} \\ & \mathbf{O}_{jerk}\mathbf{y} \leq \mathbf{o}_{jerk} \end{aligned} \tag{6.10}$$

En incrémentant α jusqu'à générer une trajectoire de référence $\mathbf{eff}(t, \mathbf{y})$ valide. Pour vérifier si une trajectoire de référence est valide, nous exécutons la méthode de cinématique inverse avec cette trajectoire et nous vérifions si le mouvement corps complet obtenu respecte les butés articulaires et est sans collision.

La figure 6.8 montre plusieurs courbes de Bézier obtenues pour différentes valeurs de α ainsi que la solution trouvée par le *Limb-RRT* $\mathbf{s}(t)$. On peut observer que même pour $\alpha = 1$, la courbe ne correspond pas exactement à $\mathbf{s}(t)$. Cela est dû au fait que le degré de la courbe de Bézier est fixé. Cependant, ce n'est pas une limitation de notre méthode car le degré de la courbe de Bézier représentant la trajectoire de l'organe terminal peut être augmenté en ajoutant des points de contrôle libres au milieu de la courbe.

Lorsqu'un seul point de contrôle variable n'est pas suffisant pour obtenir une trajectoire de l'organe terminal valide, nous ajoutons des variables au QP (6.10) afin d'augmenter le degré de la courbe pour que la solution corresponde mieux à la solution du *Limb-RRT*, comme montré sur la figure 6.9.

Dès que la méthode de cinématique inverse produit un mouvement cinématiquement valide et sans collision avec la trajectoire de référence fournie, la méthode s'arrête et cette trajectoire est utilisée. Ce fonctionnement permet d'éviter au maximum les mouvements inutiles des pattes du robot et d'utiliser la trajectoire la plus lisse possible tout en étant sans collision.

De part le fonctionnement itératif de cette méthode, elle nécessite beaucoup de temps de calcul, comme mesuré dans la section 7.3.4. Cependant, dans beaucoup de cas la trajectoire par défaut est valide et cette méthode n'est utilisée que dans les environnements contraints.

Implémentation

La méthode *Limb-RRT* est implémentée dans le paquet HPP-RBPRM [9]. Pour la gestion des courbes de Bézier, nous utilisons les même outils que ceux utilisés par CROC.

9. `https://github.com/humanoid-path-planner/hpp-rbprm`

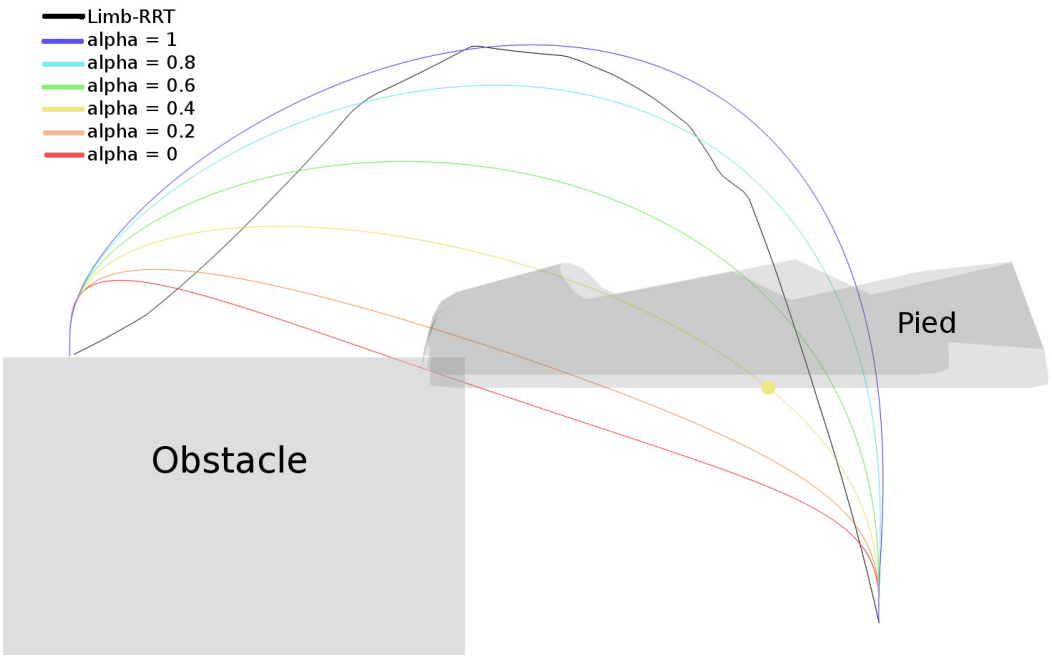


FIGURE 6.8 – Faisceau de courbes de Bézier pour la trajectoire du centre d'un pied de HRP-2 (dans le plan (x, z)) générées pour différentes valeurs de α. En gris transparent le pied de HRP-2 suivant la trajectoire pour $\alpha = 0.4$, en collision avec l'obstacle. Les courbes avec $\alpha \leq 0.4$ génèrent des mouvements en collision avec l'obstacle.

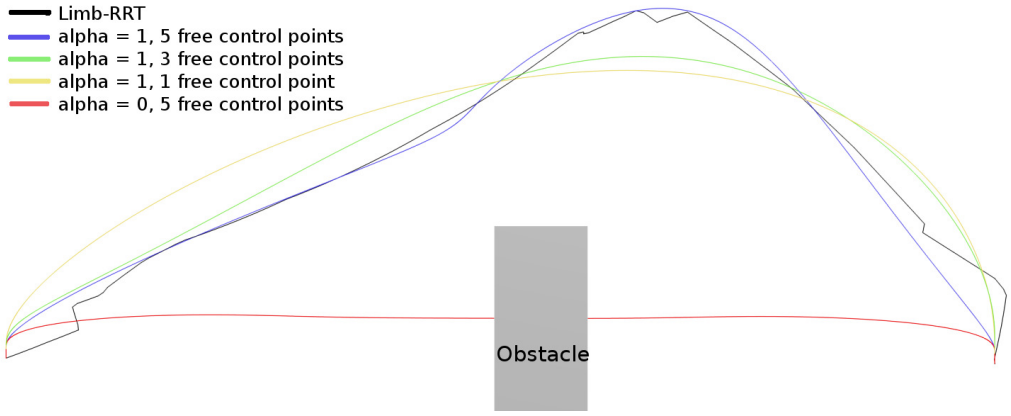


FIGURE 6.9 – Faisceau de courbes de Bézier pour la trajectoire du centre d'un pied de HRP-2 (dans le plan (x, z)) générées avec un nombre différent de points de contrôle libres.

6.8 Conclusion

Dans ce chapitre, nous avons présenté une architecture complète de planification de mouvement pour robot à pattes dans des scénarios multi-contact et non limitée aux mouvements quasi-statiques. L'architecture proposée ne prend en entrée qu'une tâche de locomotion de haut niveau : un placement final pour le centre du robot et produit automatiquement un mouvement corps complet dynamiquement consistant et sans collision.

Cette architecture utilise l'approche découplée présentée en introduction de ce manuscrit et résout chaque sous problème avec des méthodes de l'état de l'art ou des méthodes proposées durant cette thèse.

De plus, les deux critères de faisabilité proposés dans les deux premières parties de cette thèse sont intégrés à cette architecture afin d'assurer la **faisabilité** de la sortie de chaque bloc par rapport au bloc suivant.

6.8.1 Connexion avec la plateforme robotique

La sortie de la méthode proposée est une trajectoire articulaire décrivant l'évolution dans le temps de la valeur de l'angle de chaque articulation. Il est ensuite possible de donner cette trajectoire comme référence à un contrôleur bas-niveau afin d'exécuter le mouvement sur le robot.

Dans le cas de HRP-2, nous utilisons un contrôle en boucle fermée, avec les capteurs de forces des pieds du robot, fourni avec le robot (appelé "stabilisateur") [Kajita 2001, Kajita 2005]. Le but de ce contrôle est de stabiliser les mouvements des flexibilités présentes dans les chevilles du robot.

Cependant, de par sa formulation, ce stabilisateur est très limité dans le type de mouvement qu'il est capable de gérer. En effet, sa formulation suppose des mouvements bipèdes sur sol plat.

Dans le cas de mouvements utilisant les mains du robot, il est possible de désactiver ce stabilisateur et de réussir tout de même à exécuter le mouvement en boucle ouverte sans perte d'équilibre grâce au support additionnel apporté par les contacts des mains du robot.

Mais dans le cas de la marche bipède sur sol incliné ou avec des contacts non coplanaires, cela n'est pas possible.

Le développement d'une méthode de contrôle bas-niveau plus générique est en cours au sein de l'équipe mais n'est pas actuellement utilisable. Par conséquent, les mouvements planifiés testés sur le robot réel ont dû être limités aux mouvements avec des contacts coplanaires.

6.8.2 Prise en compte du moment cinétique

Dans notre mise en oeuvre de l'architecture proposée, les premiers sous-problèmes imposent un moment cinétique nul. En effet, comme nous utilisons le

modèle centroïdal nous n'avons pas les informations nécessaires pour calculer ce moment cinétique exactement puisqu'il dépend entre autre du mouvement des pattes du robot. Nous ne pouvons le calculer qu'à partir de la méthode de génération de mouvement corps complet.

Une hypothèse réaliste est de supposer que l'influence du moment cinétique ne va pas invalider la séquence de contacts grâce au critère de robustesse [Del Prete 2016] utilisé pendant la génération de contact. Par contre, il est fort probable que ce moment cinétique puisse invalider la trajectoire centroïdale calculée en le négligeant, ou tout du moins la rendre sous-optimale.

Afin de prendre en compte le moment cinétique pendant la génération de trajectoire centroïdale, une approche proposée dans la littérature ([Kajita 2003] dans le cas de la marche sur sol plat [Herzog 2016, Farshidian 2016] dans le cas non-coplanaire) est d'itérer entre la méthode de génération de trajectoire centroïdale et celle de génération de mouvement corps complet. Lors de la première itération, la méthode de génération de trajectoire centroïdale n'a aucune information sur le moment cinétique généré par le mouvement des pattes du robot. Ce moment est calculé par la méthode de génération de mouvement corps complet qui prend en entrée la trajectoire centroïdale puis est transmis comme référence à la méthode de génération de trajectoire centroïdale lors de la seconde itération. La trajectoire centroïdale va alors être modifiée pour prendre en compte ce moment. Cette méthode est également appelée "filtre dynamique".

Ces deux méthodes vont itérer jusqu'à ce que les changements dans la trajectoire centroïdale deviennent minimes.

L'intégration de cette méthode dans notre architecture est faisable et ne demande plus que du travail d'implémentation, prévu dans un futur proche.

Chapitre 7

Résultats

Sommaire

Maintenant que nous disposons d'une architecture complète de planification de mouvement, nous allons pouvoir évaluer les performances de chacun des blocs de cette architecture, ainsi que de l'architecture complète. Nous nous intéressons principalement à évaluer la justesse et l'intérêt de l'utilisation de notre méthode CROC comme critère de faisabilité pendant la planification de contact.

Nous rappelons que des résultats propres à la planification de trajectoire guide kinodynamique sont présentés à la fin de la première partie de ce manuscrit (section 2.4), nous ne revenons donc pas en détails sur cette méthode dans ce chapitre.

Dans la section suivante nous présentons différents scénarios utilisés afin d'évaluer notre méthode de planification de mouvement. Chaque scénario illustrant des capacités spécifiques de nos méthodes. Pour chaque scénario nous présentons les

résultats qualitatifs obtenus, tel que des mouvements corps-complet exécutés en simulation ou sur le robot HRP-2 réel.

Ensuite, nous allons mesurer empiriquement l'impact et l'intérêt de l'utilisation de CROC comme critère de faisabilité pendant la génération de contact. Nous montrons que ce critère permet d'assurer la faisabilité de la séquence de contacts produite donnée en entrée de la méthode de génération de trajectoire centroïdale et cela pour une très faible augmentation du temps de calcul requis. Nous montrons également que sans ce critère de nombreuses séquences de contacts infaisables peuvent être produites.

Finalement, nous analysons les performances de l'ensemble de l'architecture de planification de mouvement présentée dans le chapitre 6. Nous montrons que grâce à l'utilisation des deux critères de faisabilité présentés dans les deux premières parties de cette thèse, nous obtenons un taux de succès proche de 100% pour l'architecture complète. De plus, les méthodes utilisées permettent d'obtenir des performances interactives.

Tous les tests ont été effectués sur un ordinateur avec un processeur Intel Xeon CPU E5-1630 v3 à 3.7GHz (un seul cœur étant utilisé) avec 64Go de mémoire vive. Les détails d'implémentations sont donnés dans le chapitre 6.

7.1 Résultats expérimentaux

Dans cette section, nous présentons les divers scénarios utilisés pour évaluer les capacités de notre architecture de planification de mouvement. Pour chaque scénario nous présentons les résultats intéressants obtenus : séquence de contacts, mouvement corps complet, trajectoire des organes terminaux ...

La quasi-totalité de ces scénarios mettent en scène le robot HRP-2. Ce choix a été fait car le but de ces travaux était de valider les mouvements sur un robot réel et que l'équipe Gepetto possède ce robot. De plus, il s'agit du robot dont nous ayons la meilleur connaissance actuellement au sein de l'équipe et toute l'architecture de contrôle pour ce robot était déjà fonctionnelle au début de cette thèse.

7.1.1 Navigation sur sol plat

Le scénario présenté sur la figure 7.1 est un simple problème de navigation sur sol plat avec le robot HRP-2. Comme montré sur la figure, la méthode de planification de trajectoire trouve un guide permettant d'éviter les obstacles et de rejoindre la position finale désirée. Il s'agit d'un problème de marche bipède classique, déjà résolu par de nombreuses méthodes de l'état de l'art. Le but de ce scénario est simplement de valider que notre architecture de planification (en particulier notre planificateur de contacts), conçue pour des mouvements acycliques et multi-contact, reste capable de gérer ce type de mouvement simple. La quasi-totalité des séquences de contact trouvées pour ce scénario (comme celle présentée dans la figure 7.2)

étaient cycliques, sans que nous ayons besoin de contraindre le planificateur de contact.

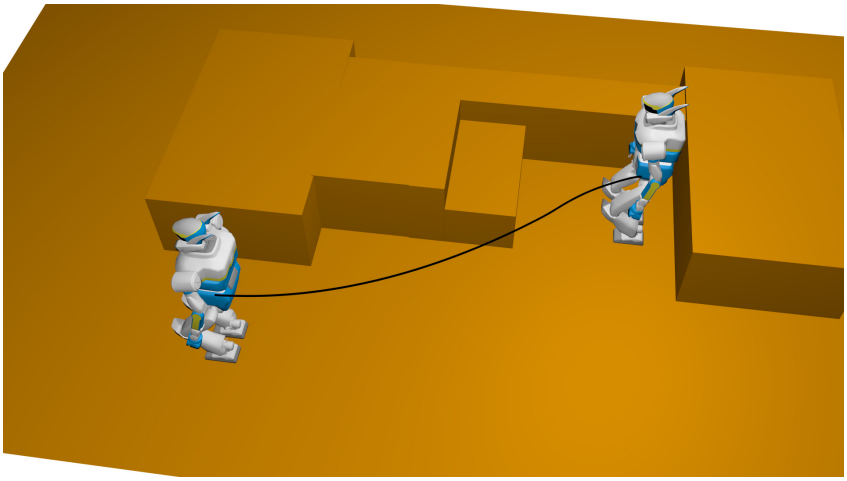


FIGURE 7.1 – Présentation du scénario de navigation sur sol plat. La position initiale est à droite et la position finale à gauche. En noir, un exemple de trajectoire guide planifiée.

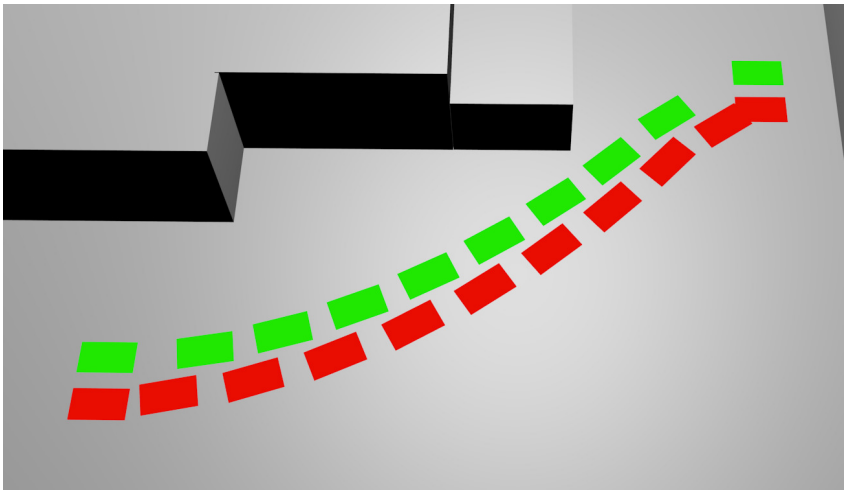


FIGURE 7.2 – Séquence de contacts trouvée pour le scénario de navigation sur sol plat. En vert les positions successives du pied droit et en rouge celles du pied gauche. La position initiale est en haut à droite, la position finale à gauche.

La figure 7.3 montre la trajectoire centroïdale générée à partir de cette séquence de contacts. On peut observer que le mouvement produit est très proche d'un mouvement quasi-statique. En effet, la trajectoire centroïdale présentée sur la figure 7.3 passe bien au-dessus du centre du pied en contact et pendant les phases de simple support le *Zero Moment Point* (ZMP) [Kajita 2003] reste proche du polygone de

support du pied en contact.

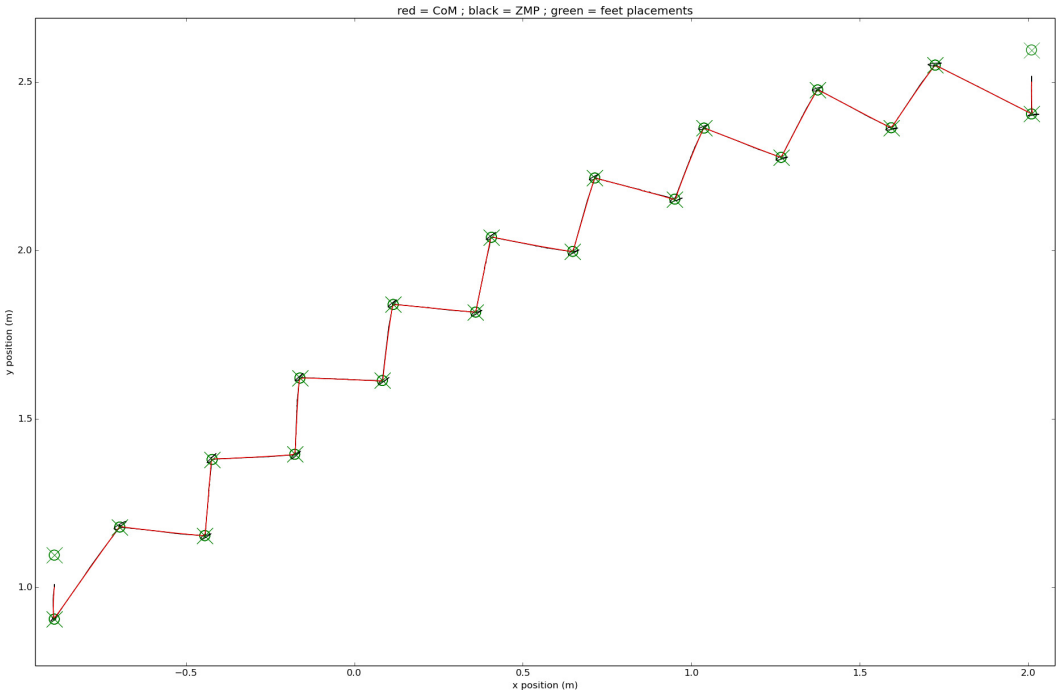


FIGURE 7.3 – Trajectoire centroïdale (en rouge) et trajectoire du ZMP (en noir) générée pour le scénario de navigation sur sol plat. En vert les positions des contacts. La position initiale est en bas à gauche, la position finale à droite.

Ce comportement est forcé artificiellement en imposant une durée minimale pour les phases de simple support. Ce choix a été fait afin de minimiser l'impact du moment cinétique que nous ne sommes pas encore en mesure de gérer correctement. En effet comme expliqué dans la section 6.8.2, le moment cinétique généré par le mouvement des membres du robot n'est actuellement pas pris en compte pendant l'étape de génération de trajectoire centroïdale. Or, plus le mouvement du pied est rapide pendant la phase de simple support, plus ce moment cinétique est important et risque de rendre la trajectoire centroïdale générée invalide.

C'est pourquoi nous imposons une durée minimale relativement élevée (1 seconde dans ce scénario) pour les phases de simple support. Cette limitation ne sera plus nécessaire une fois que le filtre dynamique sera intégré à notre architecture.

Nous avons exécuté avec succès des mouvements pour ce scénario sur le robot réel, comme montré sur la figure 7.4.

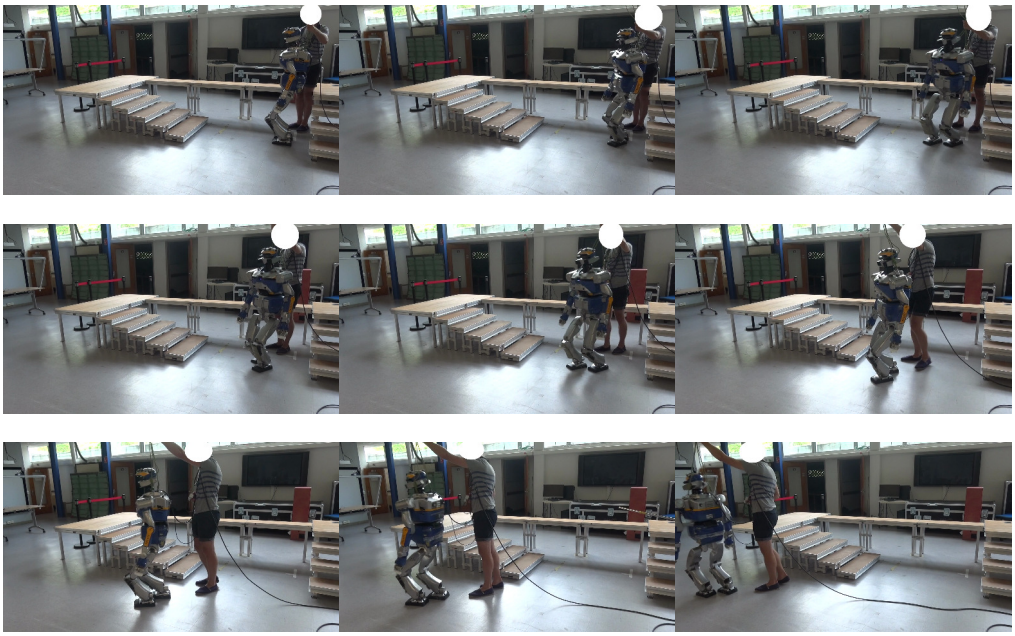


FIGURE 7.4 – Captures du mouvement corps complet exécuté sur le robot HRP-2 pour le scénario de navigation sur sol plat.

7.1.2 Navigation sur sol plat en environnement contraint

Ce second scénario est identique au scénario précédent à l'exception de l'ajout de petits obstacles au niveau du sol. La figure 7.5 montre la séquence de contacts obtenue pour ce scénario et la figure 7.6 montre la trajectoire centroïdale.

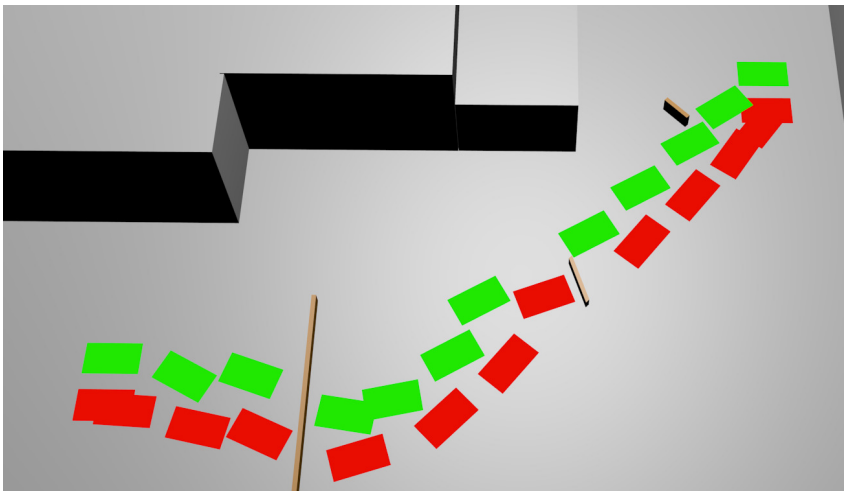


FIGURE 7.5 – Séquence de contacts trouvée pour le scénario de navigation parmi les obstacles. En vert les positions successives du pied droit et en rouge celles du pied gauche. La position initiale est en haut à droite, la position finale à gauche.

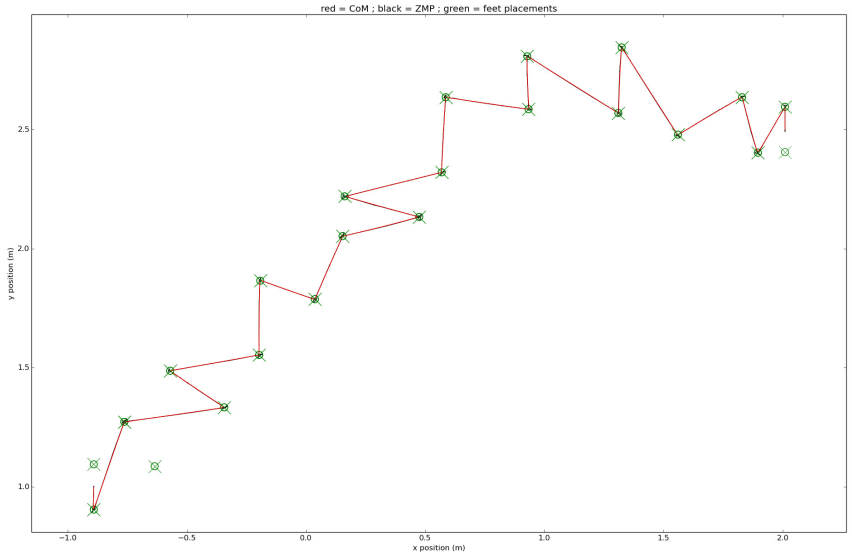


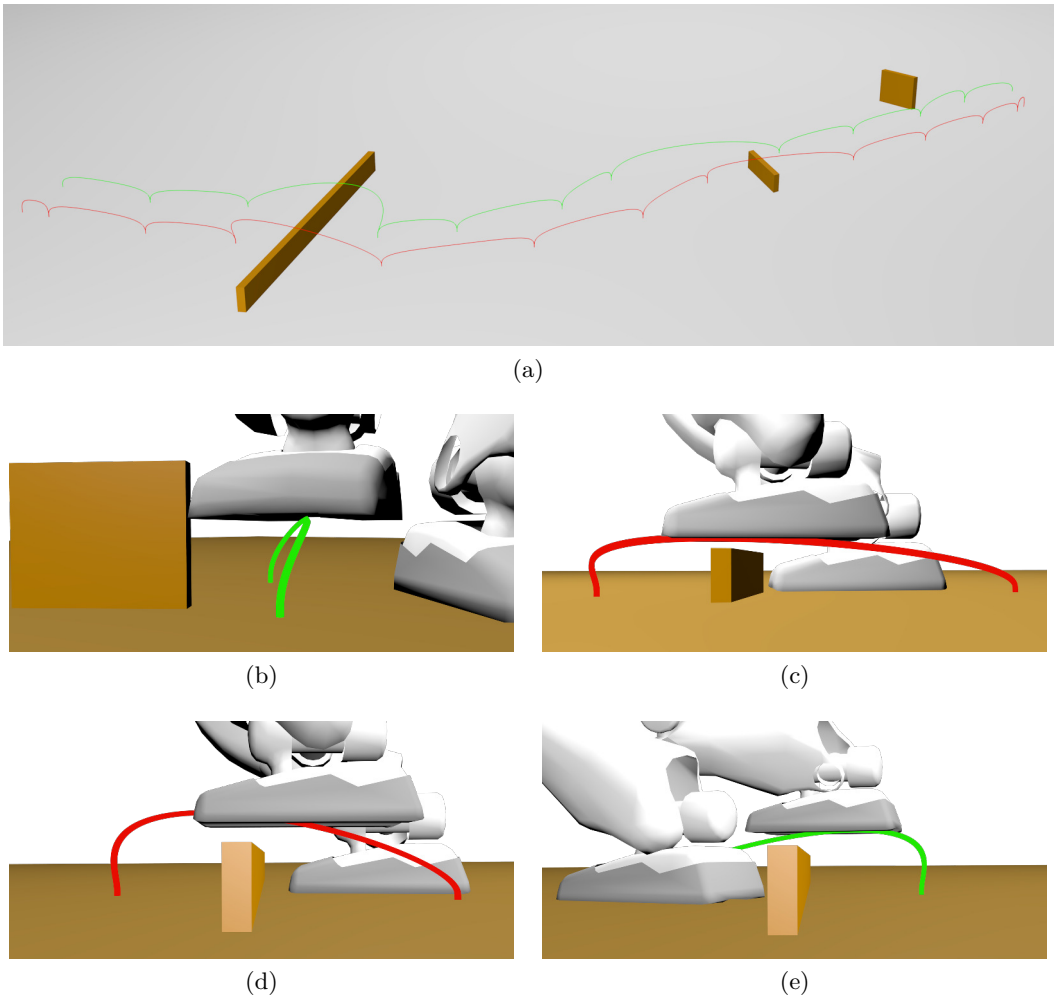
FIGURE 7.6 – Trajectoire centroïdale (en rouge) et trajectoire du ZMP (en noir) générée pour le scénario de navigation parmi les obstacles. En vert les positions des contacts. La position initiale est en bas à gauche, la position finale à droite.

Par rapport au scénario précédent, la trajectoire guide planifiée n'est pas modifiée par la présence de ces obstacles au niveau du sol. En effet, la planification de guide ne va éviter que les obstacles gênants le mouvement du tronc du robot. Notre planificateur de contacts est ensuite capable de trouver des positions pour les pieds qui évitent la collision entre les jambes et ces obstacles.

La réelle difficulté de ce scénario est la génération de trajectoires pour les pieds du robot. En effet, la présence d'obstacles empêche l'utilisation d'une trajectoire par défaut. La figure 7.7 montre les trajectoires des pieds générées automatiquement pour ce scénario. Sur la ligne du haut (a) on peut observer que la trajectoire par défaut est utilisée pour la majorité des mouvements de pieds. Seul les cas où cette trajectoire engendre une collision avec l'environnement nécessitent l'appel à la méthode de génération de trajectoire des organes terminaux présentée dans la section 6.7.

La figure 7.7(b) montre que contrairement à la majorité des méthodes d'évitement d'obstacles pour les pieds (comme [Focchi 2013, García Armada 2003]), *Limb-RRT* ne se contente pas de modifier la hauteur de la trajectoire mais peut générer des trajectoires évitant l'obstacle grâce à un mouvement latéral.

Les figures (d) et (e) montrent que la trajectoire des pieds dépend de la configuration corps complet. En effet, selon si le pied en mouvement est le pied "à l'avant" du robot ou "à l'arrière", la trajectoire valide n'est pas la même.



(a) (b) (c) (d) (e)

FIGURE 7.7 – Trajectoires des pieds pour le scénario de navigation parmi les obstacles. En vert la trajectoire du pied droit et en rouge celle du pied gauche.

La figure 7.8 montre des captures des moments clés du mouvement corps complet généré pour ce scénario, ce mouvement a été validé sur le simulateur Open-HRP.

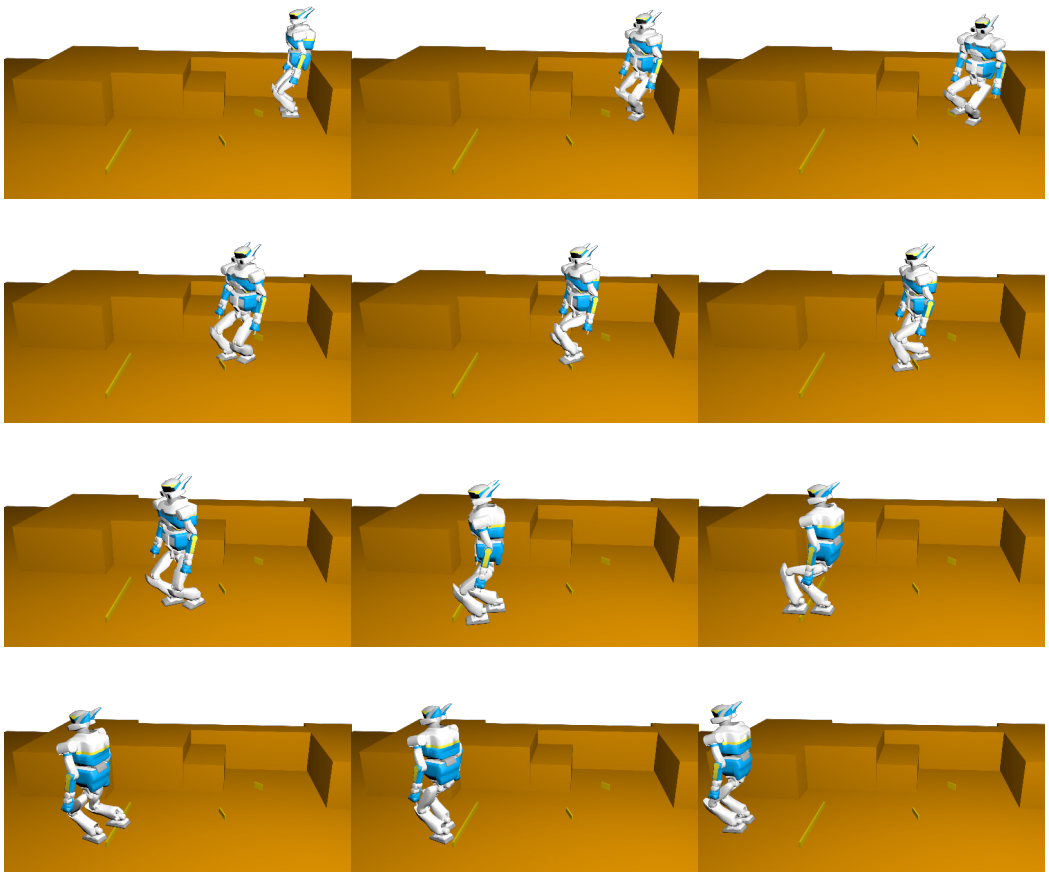


FIGURE 7.8 – Captures du mouvement corps complet sur le scénario de navigation parmi les obstacles.

7.1.3 Montée des marches

Ce scénario consiste à faire monter au robot HRP-2 un escalier de six marches d'une hauteur de 10cm et d'une profondeur de 30cm chacune. Les contraintes cinématiques de HRP-2 le forcent à poser les deux pieds sur chaque marche pendant la montée. Ce scénario avait déjà été résolu dans [Carpentier 2017c] mais avec une séquence de contacts et des trajectoires des pieds définies manuellement. Ici, la seule entrée est la position initiale et finale du centre du robot.

La figure 7.9 montre un exemple de séquence de contacts planifiée, la figure 7.10 montre la trajectoire centroïdale pour les deux premières marches. Sur cette dernière figure, on peut observer que la trajectoire du ZMP (en noir) sort légèrement de la zone sous la flexibilité de la cheville (cercle vert). Cela signifie que nous sommes en limite de validité du mouvement pour le stabilisateur de HRP-2.

Cet écart entre le ZMP simulé en trait plein noir et le ZMP de référence en pointillés est dû au moment cinétique. En effet, nous rappelons que jusqu'à la génération du mouvement corps complet, ce moment est négligé. Nous voyons ici l'intérêt de mettre en place une boucle entre la génération de trajectoire centroïdale et la génération de mouvement corps complet afin de mieux prendre en compte le moment cinétique, comme suggéré dans la section 6.8.2.

Ce mouvement a été validé avec le robot HRP-2 en situation réelle. Des captures de ce mouvement sont présentées dans la figure 7.11.

La démarche "accroupie" que l'on peut observer sur cette figure est due aux limites cinématiques du robot : si la trajectoire du centre de masse était plus haute durant ce mouvement, à chaque montée de marche la jambe en arrière s'approcherait trop de la butée articulaire. Cela n'est pas toléré par le contrôleur bas niveau du robot, qui entrerait alors en oscillation.

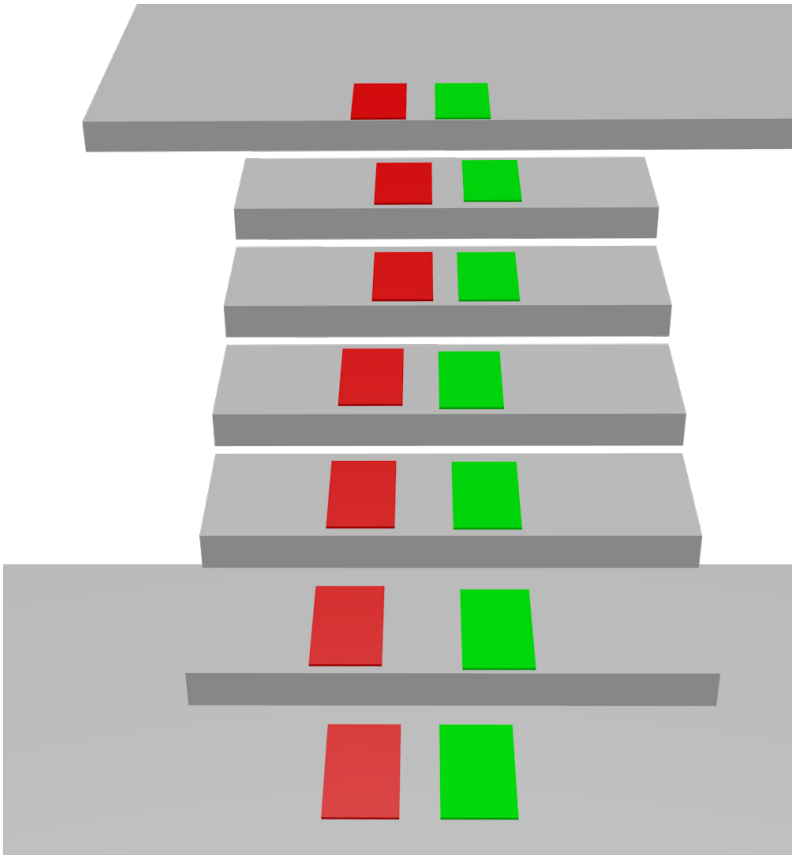


FIGURE 7.9 – Séquence de contacts pour la montée des marches de 10cm. En vert les positions successives du pied droit et en rouge celles du pied gauche.

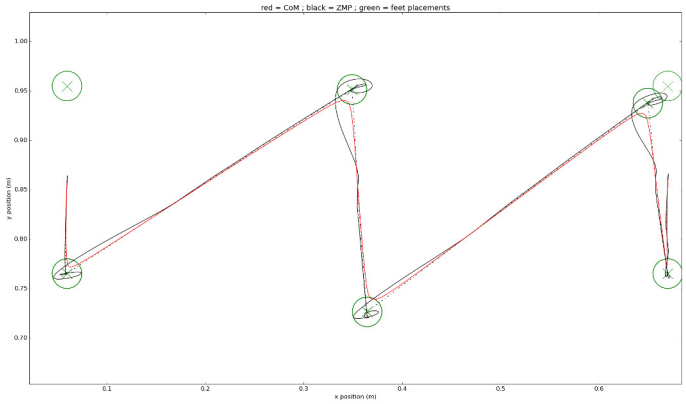


FIGURE 7.10 – Trajectoire centroïdale (en rouge) et trajectoire du ZMP (en noir) générée pour la montée de deux marches de 10cm de hauteur et de 30cm de profondeur. En vert les positions des contacts.

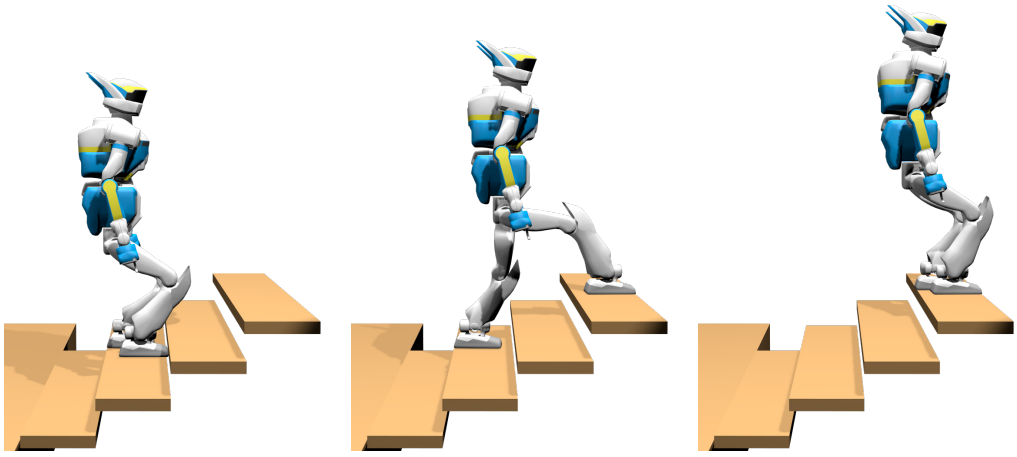
FIGURE 7.11 – Captures du mouvement corps complet réalisé par le robot HRP-2 pour le scénario de la montée des marches.

Intérêt du critère de faisabilité d'une transition de contact

Pour ce scénario, l'impact de l'utilisation de la méthode CROC comme critère de faisabilité pendant la génération de contact peut s'observer de manière qualitative. La figure 7.12 montre un exemple de séquence de contacts qui peut être trouvée par la méthode de génération de contacts classique.

Toutes les configurations de cette séquence sont cinématiquement et dynamiquement valides et respectent les règles de la génération de contact avant l'introduction du test de faisabilité. Or, il n'existe aucune trajectoire centroïdale valide permettant de monter deux marches en un pas comme sur cette figure.

L'introduction de CROC comme critère de faisabilité pendant la génération de contacts permet de détecter ce genre de cas et de rejeter de tels candidats.



(a) (b) (c)

FIGURE 7.12 – Exemple de séquence de contacts infaisable détectée par CROC. Les trois configurations sont cinématiquement valides et en équilibre mais il n'existe aucune trajectoire centroïdale valide entre les configurations (b) et (c).

7.1.4 Montée des marches avec rambarde

Ce scénario est un autre escalier, cette fois-ci avec des marches de 15cm de hauteur pour 30cm de profondeur. Le robot peut saisir une rambarde présente sur la droite de l'escalier avec sa main pour monter les marches. Il s'agit donc d'un scénario multi-contact avec des surfaces de contact non colinéaires. Ce scénario avait déjà été résolu dans [Carpentier 2016], avec le planificateur de contacts de [Tonneau 2018a]. Cependant en l'absence de CROC, le filtrage manuel des séquences de contact infaisables avait nécessité de nombreux essais et un temps considérable, avant de trouver une séquence de contacts faisable. Par ailleurs, les trajectoires des organes terminaux étaient définies manuellement.

Une séquence de contacts est présentée sur la figure 7.13. Sur la figure 7.14 la séquence de configurations en contact trouvée par le planificateur est montrée. On peut observer sur cette figure que le mouvement est acyclique : il n'y a pas de cycle prédéfini de création et surpression de contact entre les deux pieds et la main. Le planificateur de contact sélectionne le meilleur contact à repositionner selon la faisabilité cinématique et dynamique, et selon l'heuristique de choix de contact utilisée.

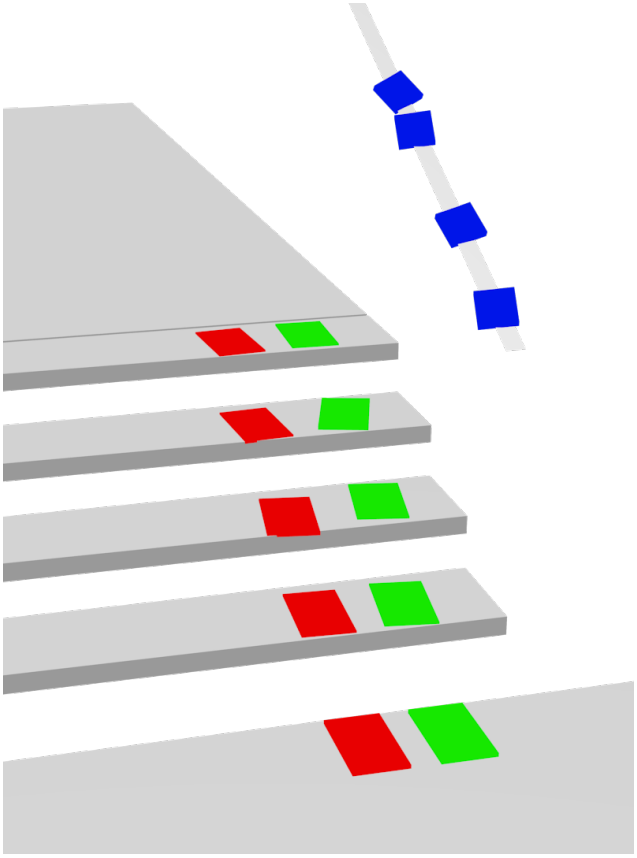


FIGURE 7.13 – Séquence de contacts pour la montée des marches avec rambarde. En vert les positions successives du pied droit et en rouge celles du pied gauche en bleu celles de la main droite.

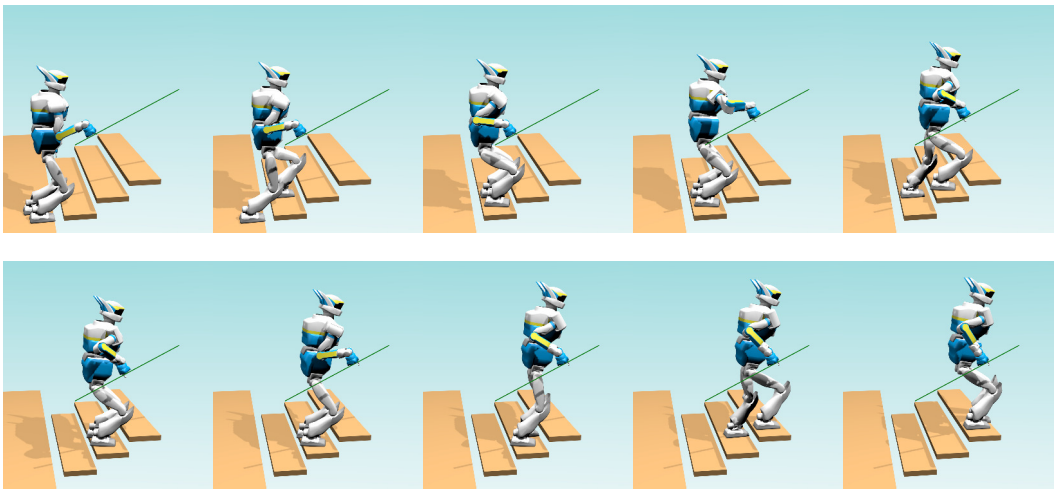


FIGURE 7.14 – Captures du mouvement corps complet réalisé par le robot HRP-2 pour le scénario de la montée des marches avec rambarde.

Planification de la saisie avec la main

Les contraintes imposées par un contact où la main saisit un objet de l'environnement comme la rambarde sont très différentes des contraintes où la main serait simplement posée sur une surface de l'environnement. En effet, dans le premier cas il s'agit d'un contact bilatéral : des forces peuvent être exercées dans les deux sens selon la direction de la normale au contact.

Notre méthode de planification de contacts n'est pas encore capable de déterminer si un contact doit être considéré comme une saisie ou non. Dans le cas de ce scénario nous avons dû prédéfinir que tout les contacts établis avec la main droite correspondaient à des saisies.

De plus, l'actionnement de la pince de la main du robot a dû être défini manuellement lors de la génération du mouvement corps complet.

Dans le futur, il serait intéressant d'intégrer les travaux de [Mirabel 2017] liés à la manipulation d'objets au sein du planificateur de contact pour déterminer si il est possible de saisir une prise. L'intérêt supplémentaire de ces travaux est de planifier un mouvement de saisie et une position de saisie valide.

7.1.5 Débris

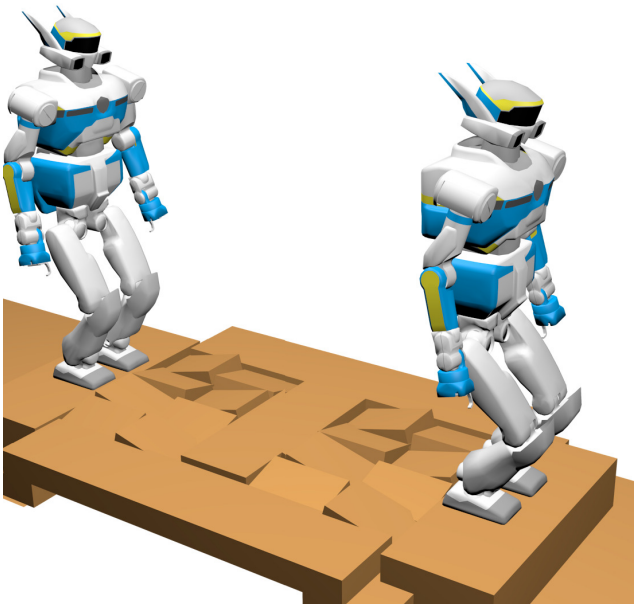


FIGURE 7.15 – Présentation du scénario simulant des débris : la position initiale se trouve à gauche et la position finale à droite.

Ce scénario simule des débris ou un sol jonché d'obstacles. Comme présenté dans la figure 7.15 il s'agit de rangées de plateformes de 30cm de long pour 20cm de large, inclinées de 15° selon l'axe x ou y. Ce scénario imite une des tâches du *Darpa Robotic Challenge*, présentée dans la figure 1-a en introduction.

Une séquence de contacts faisable est montrée par la figure 7.16 et des captures du mouvement corps complet sont présentées sur la figure 7.18. Ce mouvement a été validé sur le simulateur Open-HRP mais nous avons observé que ce mouvement est en limite du domaine de validité du stabilisateur du robot. En effet, comme expliqué dans le chapitre 6, ce stabilisateur est conçu pour de la marche sur sol plat.

Ce scénario est particulièrement difficile pour le planificateur de contacts à cause de toutes les collisions potentielles entre le pied ou le bas de la jambe du robot avec l'environnement. Les pieds de HRP-2 mesurant 24cm de long et 14cm de large, ces plateformes ne sont que quelques centimètres plus grandes que la taille du pied du robot. Par conséquent, très peu de candidats pendant la génération de contacts donnent lieu à une configuration sans collision.

Pour les même raisons, la génération de trajectoire sans collision pour les pieds est complexe. Notre méthode parvient néanmoins à trouver automatiquement ces trajectoires montrées sur la figure 7.17.

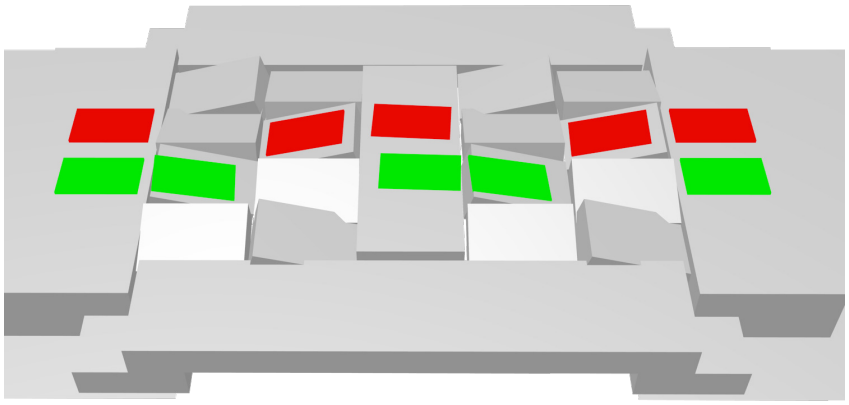


FIGURE 7.16 – Séquence de contacts pour le scénario des débris. En vert les positions successives du pied droit et en rouge celles du pied gauche.

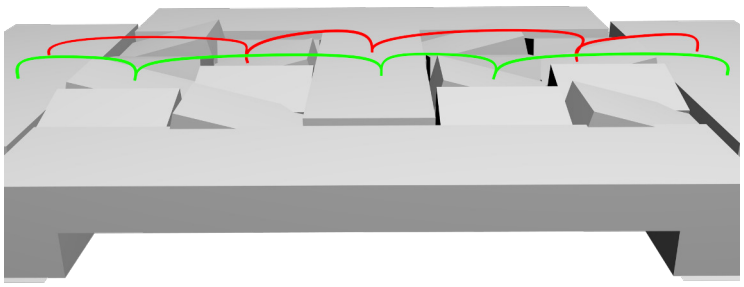


FIGURE 7.17 – Trajectoires des pieds pour le scénario des débris. En vert la trajectoire du pied droit et en rouge celle du pied gauche.

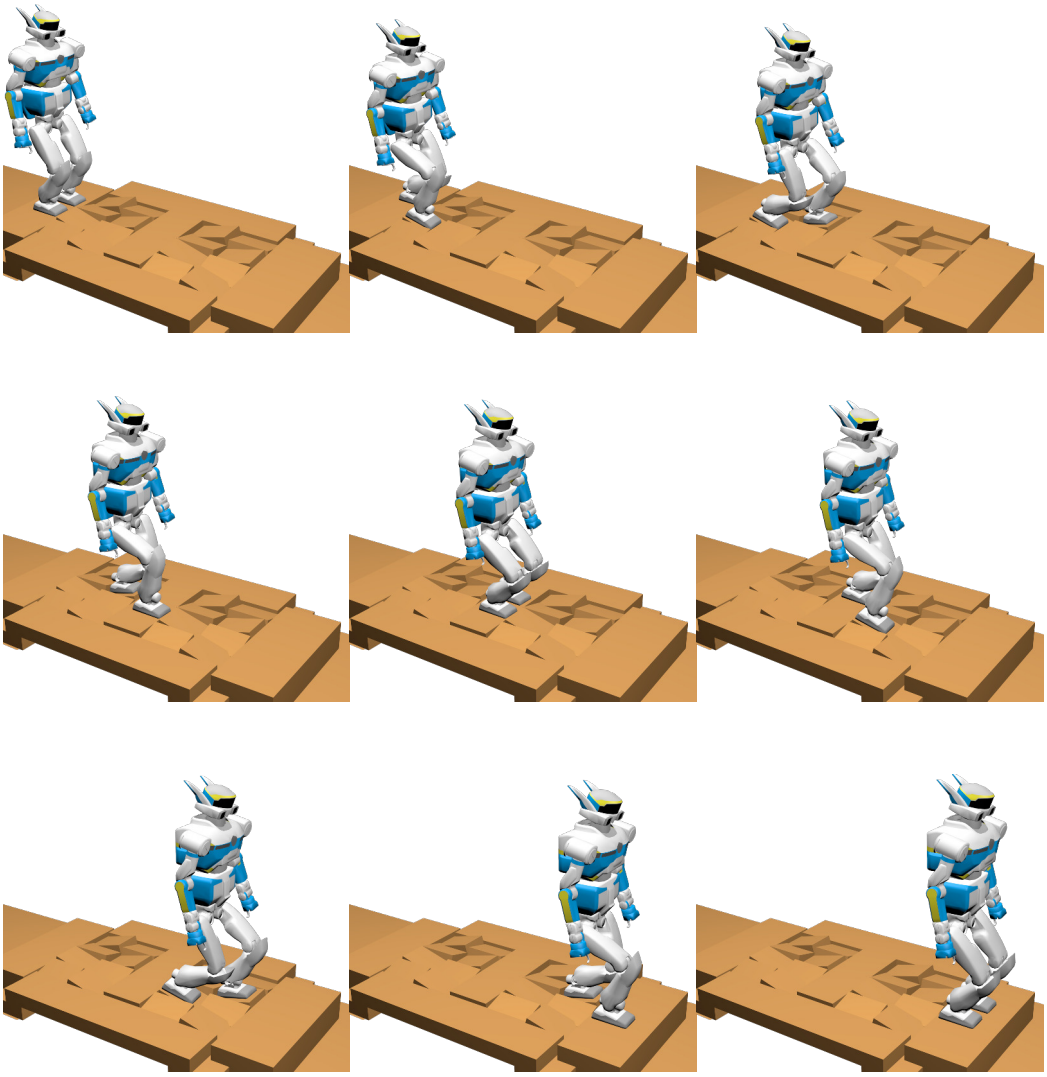


FIGURE 7.18 – Captures du mouvement corps complet réalisé par le robot HRP-2 pour le scénario des débris.

Intérêt du critère de faisabilité d'une transition de contact

Pour ce scénario, l'impact de l'utilisation de CROC comme critère de faisabilité durant la génération de contact s'est avéré très pertinent. Sans ce critère, de nombreuses séquences de contact présentaient des transitions pour lesquelles il n'existait aucune trajectoire centroïdale valide. Des exemples sont montrés sur la figure 7.19.

L'analyse empirique présentée dans la section 7.2 confirme ces observations.

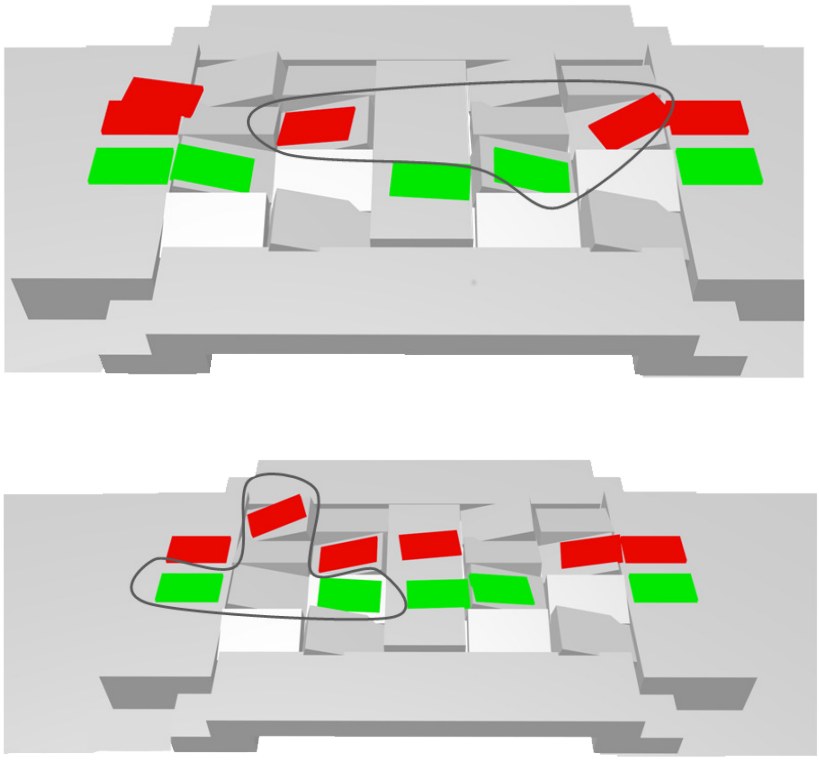


FIGURE 7.19 – Exemple de séquences de contacts infaisables détectées par CROC. Les transitions de contacts entourées en noir sur les figures donnent lieu à un problème de génération de trajectoire centroïdale infaisable.

7.1.6 Plateforme inclinée

Ce scénario est constituée d'un trou trop large pour pouvoir être traversé en une seule enjambée. Un plateforme est accessible au milieu de ce trou, mais elle est inclinée de telle sorte qu'aucune configuration en équilibre statique n'existe avec seulement un pied sur la plateforme.

Une séquence de contacts donnant lieu à un mouvement corps complet valide est montrée sur la figure 7.20. La figure 7.21 montre la trajectoire centroïdale générée pour ce scénario, on peut observer sur cette figure la prise d'élan effectuée jusqu'à la phase de simple support sur la plateforme (le second segment bleu du milieu).

En effet, à cause de l'inclinaison de la plateforme la seule manière de ne pas avoir de glissement du pied lors de la phase avec un seul contact sur la plateforme est d'exercer une accélération suffisante pour ramener les forces de contact dans le cône de friction. Dans ce cas, l'accélération doit être exercée vers la droite du robot, ce qui explique la prise d'élan permettant de commencer cette phase avec une vitesse suffisamment importante vers la gauche et l'avant.

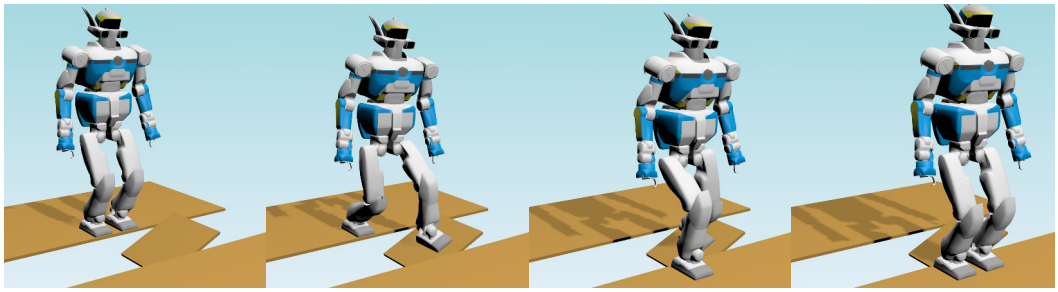


FIGURE 7.20 – Captures du mouvement pour le scénario de la plateforme inclinée.

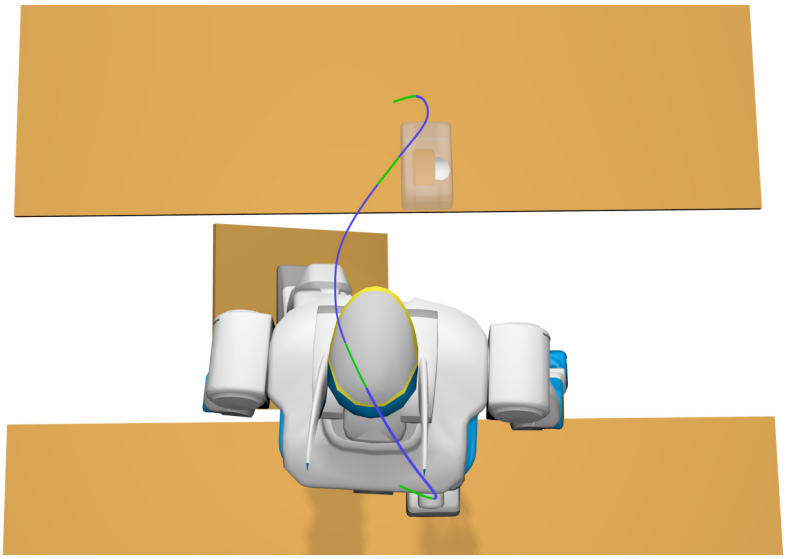


FIGURE 7.21 – Scénario de la plateforme inclinée, avec HRP-2 dans une configuration intermediraire, le prochain contact est affiché en transparent. La trajectoire centroïdale générée est affichée en vert pour les phases de contact en double support et en bleu pour les phases de simple support.

La figure 7.22 montre cette trajectoire centroïdale avec la vitesse et l'accélération, on y observe la prise d'élan permettant de commencer la phase de contact sur la plateforme avec une importante vitesse vers la gauche (vers les y positifs) et l'accélération importante vers la droite (y négatifs) pendant cette phase.

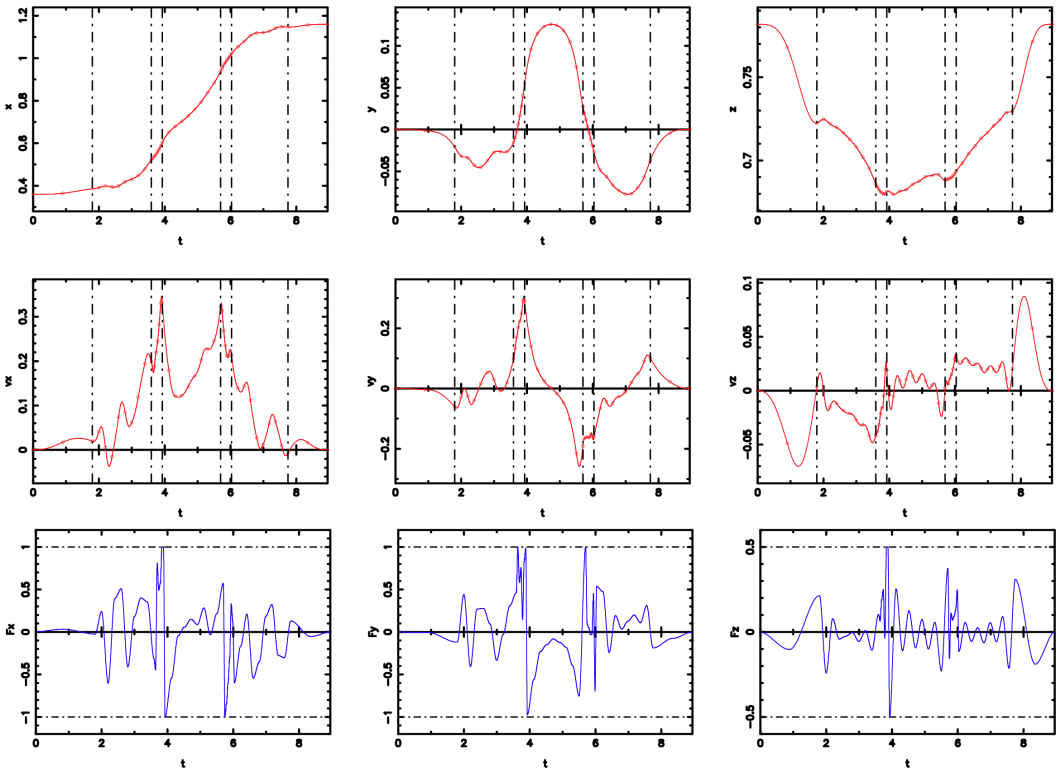
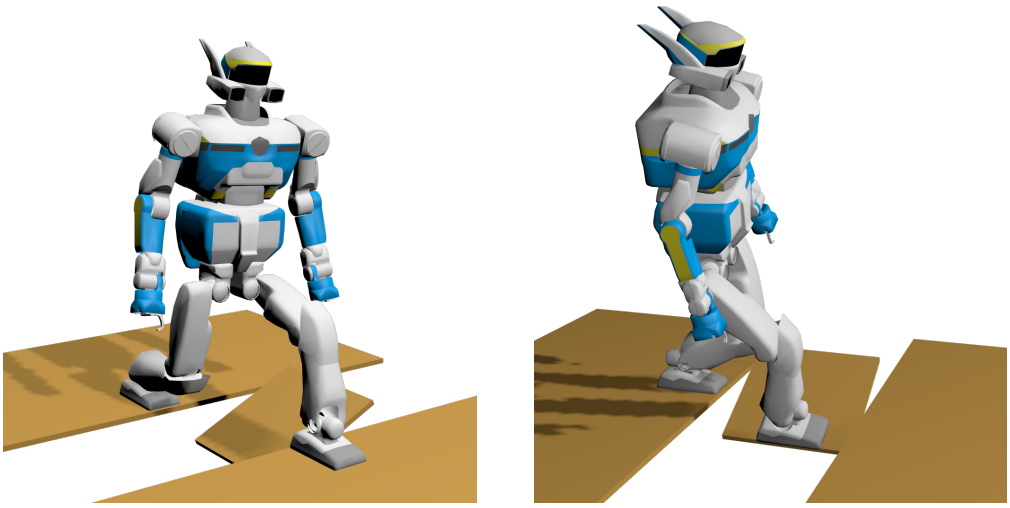


FIGURE 7.22 – Trajectoire centroïdale pour le scénario de la plateforme inclinée. Chaque colonne affiche la trajectoire selon un axe ((x, y, z) de gauche à droite) et les lignes affichent la position, la vitesse et l'accélération de haut en bas. Les pointillés verticaux représentent les changements de phase de contacts.

Intérêt du critère de faisabilité d'une transition de contact

Sans le critère de faisabilité d'une transition de contact proposée durant cette thèse, le planificateur de contacts peut trouver des séquences de contacts qui, soit essaient d'enjamber directement le trou (Fig. 7.19(a)), soit utilisent la plateforme mais avec le pied droit (Fig. 7.19(b)).

Dans les deux cas, chaque configuration de la séquence de contacts est valide cinématiquement et dynamiquement mais il n'existe aucune trajectoire centroïdale valide permettant de connecter les configurations montrées sur la figure 7.19 avec les configurations initiale et finale du problème.



(a) (b)

FIGURE 7.23 – Deux exemples de transitions de contact infaisables détectés par CROC. (a) le robot tente de traverser l'espace en une seule enjambée, (b) le robot utilise la plateforme mais avec le pied droit.

7.1.7 Marche entre plans inclinés

Afin de montrer la généricité de notre approche, nous avons testé notre méthode avec le robot quadrupède HyQ entre deux plans inclinés à 45°. La figure 7.24 montre une séquence de configurations en contact trouvée par notre méthode.

On peut observer sur cette figure que le mouvement produit est acyclique. En effet, il n'y a pas de cycle fixé définissant l'ordre des pattes à bouger. De plus, cet exemple montre que notre planificateur de contact peut gérer aussi bien les robots humanoïdes que les quadrupèdes avec une méthode identique.

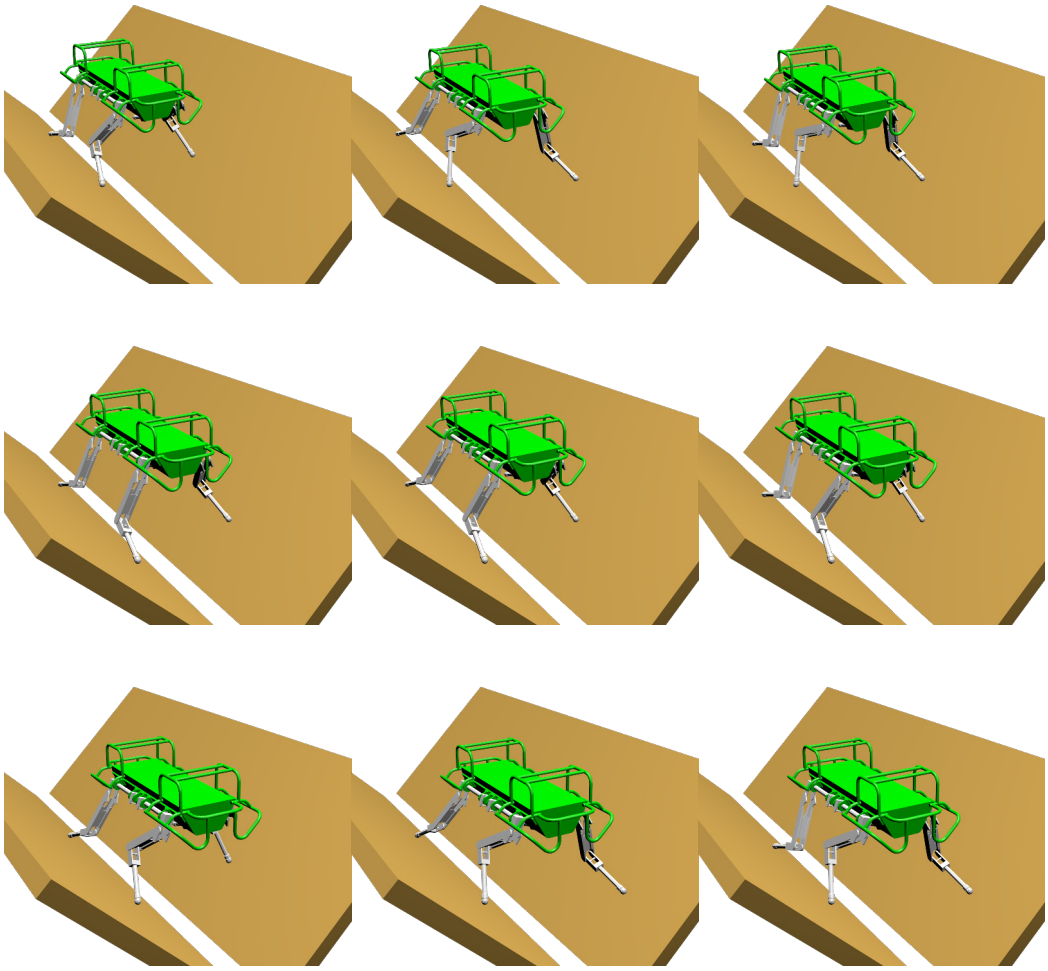


FIGURE 7.24 – Séquence de configurations en contact planifiée pour le scénario de la marche entre plans inclinés

7.1.8 Escalier industriel très contraint

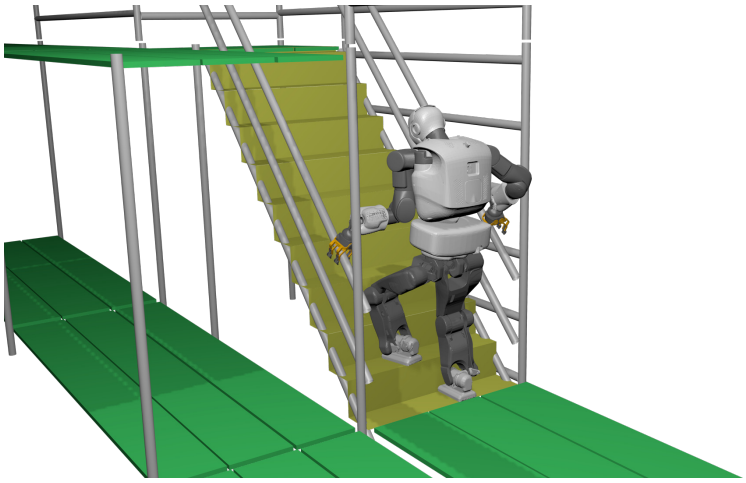


FIGURE 7.25 – Présentation du scénario de l'escalier industriel, avec le robot TALOS.

Ce dernier scénario présenté sur la figure 7.25 est un cas industriel réel, fourni par Airbus. Il s'agit d'un escalier avec des marches de 20cm de haut et de profondeur, avec une rambarde de chaque côté. Comme nous savions que l'escalier avec des marches de 15cm présenté au début de ce chapitre était la limite cinématique du robot HRP-2, nous avons essayé de résoudre ce scénario avec le robot TALOS qui est plus grand que HRP-2 (1.75m contre 1.54m) et dont les jambes ont un espace atteignable plus important.

Ce scénario est très complexe car très contraint. Tout d'abord, les marches sont plus petites que les pieds du robot TALOS [Stasse 2017] qui sont de 21cm de long pour 13cm de large. Du plus, la présence de nombreux éléments autour des rambardes génère de forte probabilités de collision pour les bras du robot.

Enfin, la configuration des marches est telle que trouver une transition de contact pour passer un pied d'une marche à la suivante est complexe. En effet si la trajectoire guide est trop haute (comme c'est le cas sur la figure 7.25) lorsque le robot va avancer, la jambe en arrière va entrer en butée articulaire (voir la jambe droite de la figure 7.25) avant que le centre de masse ne soit passé au dessus de la marche suivante. Cela signifie que lorsque le robot va lever le pied arrière, le centre de masse ne pourra pas être au dessus du pied en contact sur la marche de devant. Les forces de contacts nécessaires pour maintenir le robot en équilibre pendant le mouvement pour passer la jambe sur la marche suivante seront alors majoritairement exercées au niveau des mains saisissant les rambardes.

Il s'agit en réalité du genre de mouvement produit par les humains confrontés à ce genre d'escalier très raides : les bras sont fortement exploités pour monter et nous sommes rarement en équilibre sur notre seul pied d'appui lorsque nous montons une

marche. Cependant, dans le cas du robot TALOS, son poids total est de 95 Kg et ses poignets ne sont conçus que pour supporter un poids de 6Kg. Cette stratégie n'est donc pas viable avec ce robot car cela exercerait une trop grande force sur ses poignets.

Si au contraire, on abaisse la trajectoire guide, on obtient des configurations comme celle de la figure 7.26. Dans ce cas, à cause de la hauteur des marches, la cheville est en collision avec la marche suivante.

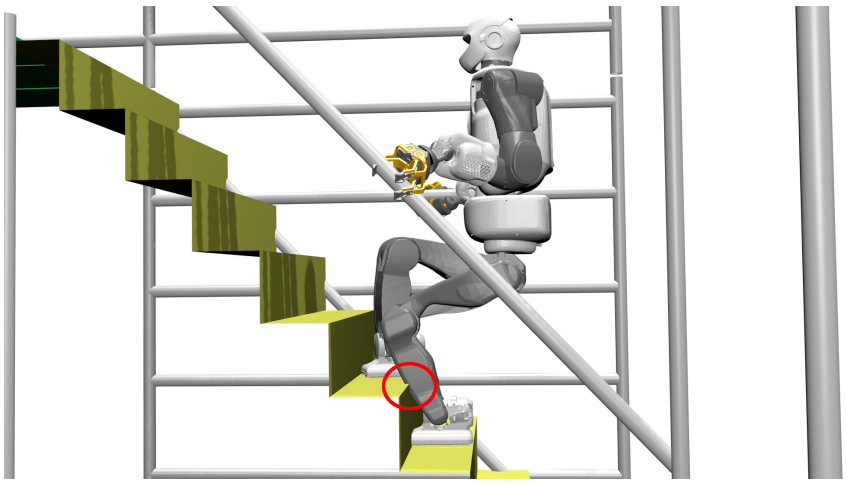
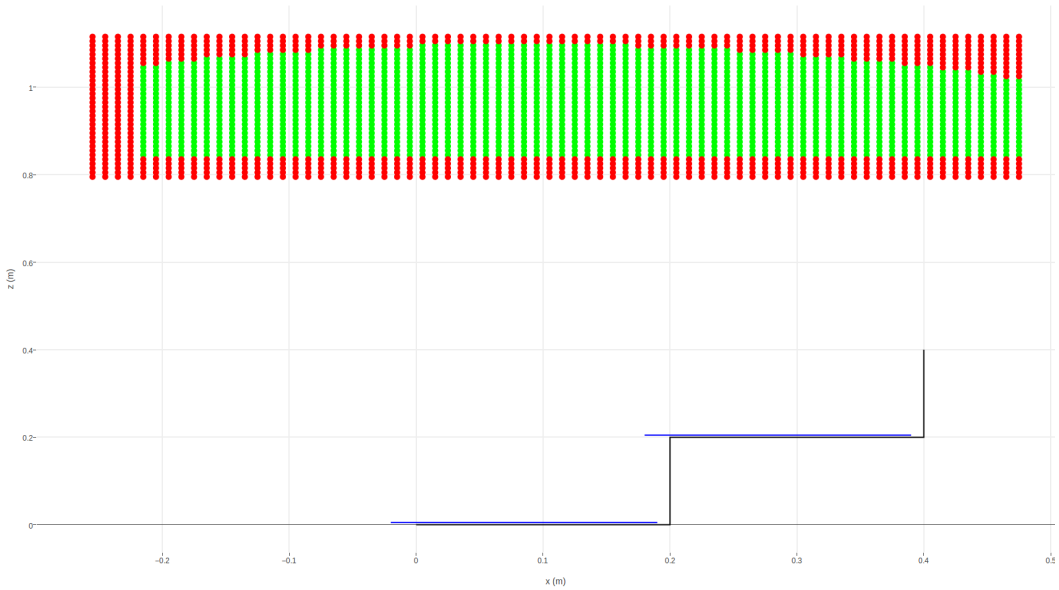


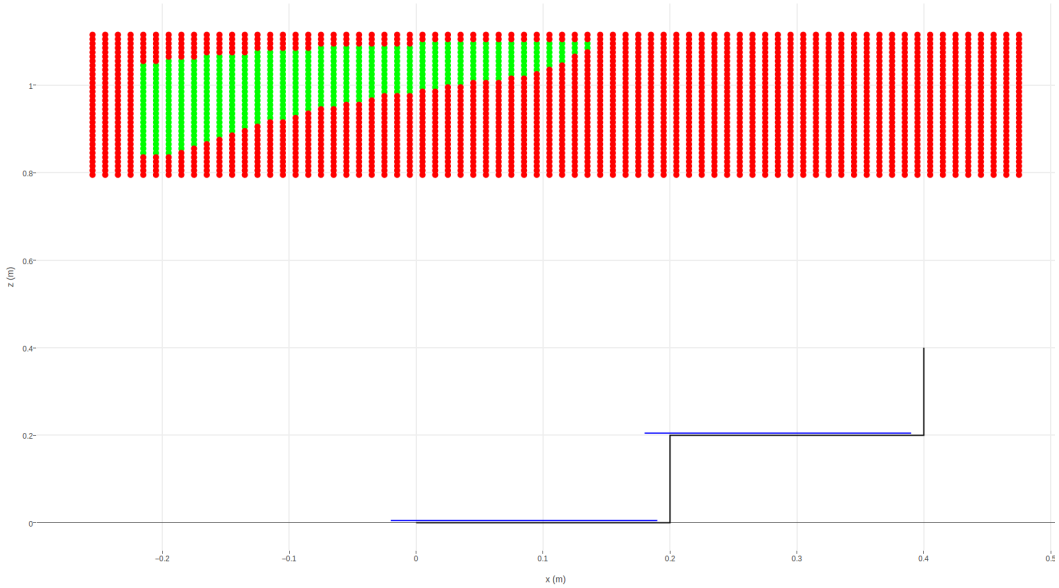
FIGURE 7.26 – Exemple de configuration en collision : la cheville gauche est en collision avec la marche suivante.

Afin d'évaluer la difficulté de ce scénario, nous avons testé la faisabilité cinématique des positions de l'origine du robot avec des contacts fixés : le robot est placé avec un pied sur chaque marche dans une configuration similaire à celle de la figure 7.25. La position des pieds est alors fixée et nous faisons varier la position de l'origine du robot selon les axes x (avant/arrière) et z (vertical) en vérifiant l'existence des configurations cinématiquement valides pour chaque position via une méthode de cinématique inverse. Puis, la collision entre les jambes du robot et les marches est vérifiée.

Les résultats sont présentés sur les graphiques 7.27. On observe qu'une grande plage de position de l'origine donne lieu à des configurations cinématiquement valides (7.27 (a)), mais que très peu de ces configurations sont sans collisions pour les jambes du robot(7.27(b)). Comme supposé, nous constatons qu'il n'existe pas de configuration cinématiquement valide et sans collision permettant de placer l'origine du robot au dessus de la marche suivante avant de lever le pied de la marche précédente.



(a) Existence d'une configuration cinématiquement valide



(b) Existence d'une configuration cinématiquement valide et sans collision

FIGURE 7.27 – Existence d'une configuration corps complet cinématique valide (a) et sans collision entre les jambes du robot et les marches (b) selon la position de l'origine du robot (dans le plan (x, z), avec la valeur de y placée entre les positions des deux pieds), avec des contacts fixés. En vert les positions de l'origine pour lesquelles il existe une configuration valide, en rouge celles où il n'en existe pas. Les traits noirs sont les marches de l'escalier, les traits bleus les positions des pieds.

En variant l'heuristique de génération de contact utilisée par le planificateur de contact, nous avons pu trouver une séquence de contacts inattendue.

Cette séquence est montrée sur la figure 7.28. Elle consiste à monter l'escalier avec les pieds orientés parallèlement aux marches. Effectivement, avec cette démarche les risques de collisions entre la cheville et la marche suivante sont très diminués, comme prouvé par la figure 7.29 qui montre la faisabilité cinématique des positions de l'origine. Si l'on compare la zone de positions valides cinématiquement avec cette posture (7.29(a)) avec celle dans le cas de la position "classique" de montée des marches (7.27(a)) on constate que dans le cas de l'orientation des pieds parallèlement aux marches la zone valide est plus réduite, notamment vers l'arrière du robot (vers les x négatifs). Cela est dû aux butées articulaires de la cheville autour de l'axe x.

Par contre, si on considère les collisions, la zone valide dans le cas de l'orientation des pieds parallèlement aux marches (7.29(b)) est bien plus importante qu'avec la posture "classique" (7.27(b)). De plus, cette zone est mieux placée car elle est au dessus des deux marches. Cette démarche augmente donc grandement l'espace des positions possibles de l'origine du robot permettant de générer des configurations en contact cinématiquement valides et sans collisions.

Ce mouvement n'a pas pu être testé dans un simulateur dynamique car nous n'avons pas actuellement d'implémentation pouvant simuler TALOS avec des contacts de saisie et des contacts non coplanaires avec précision. Cela sera à vérifier dans le futur quand un tel simulateur sera disponible.

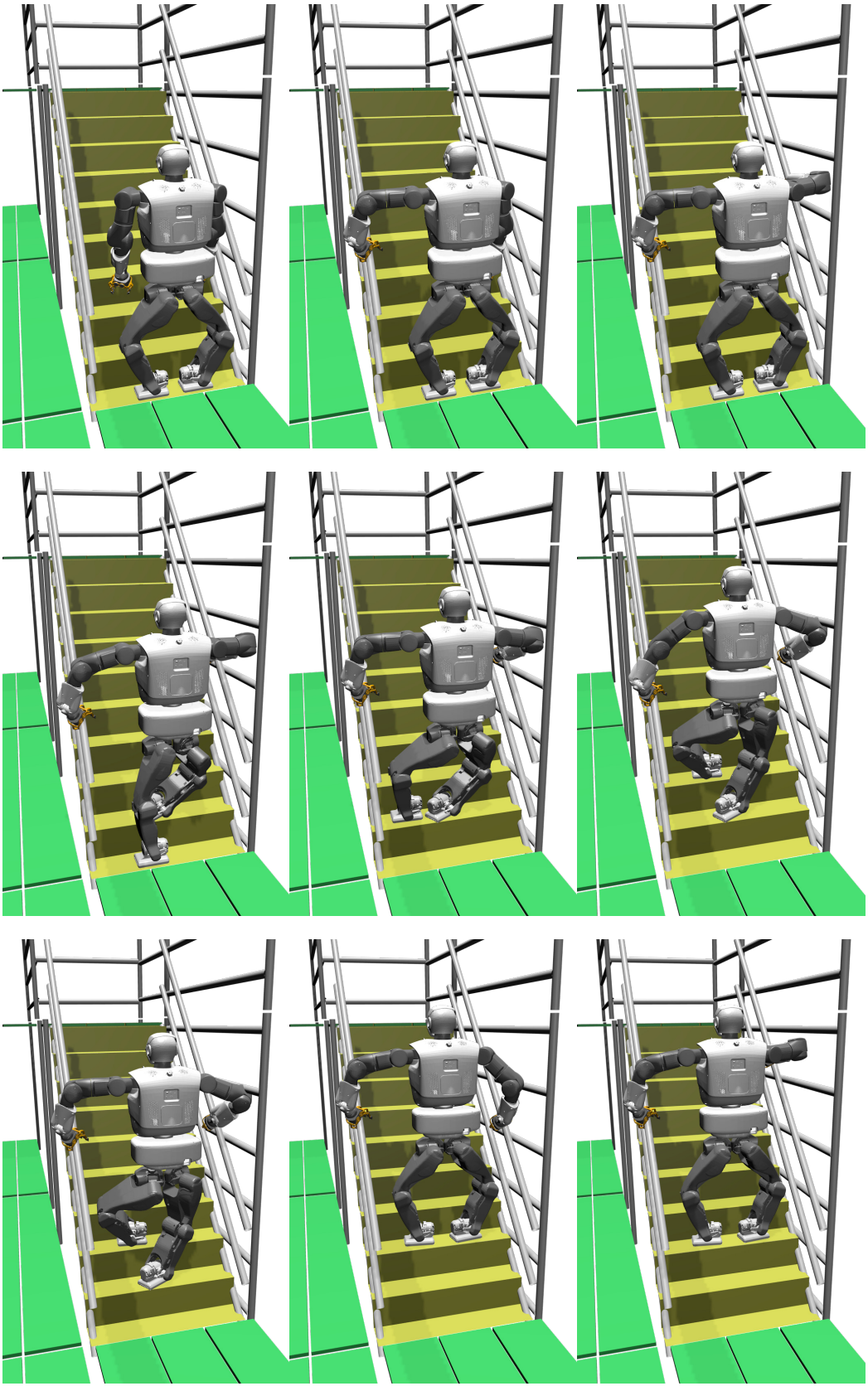
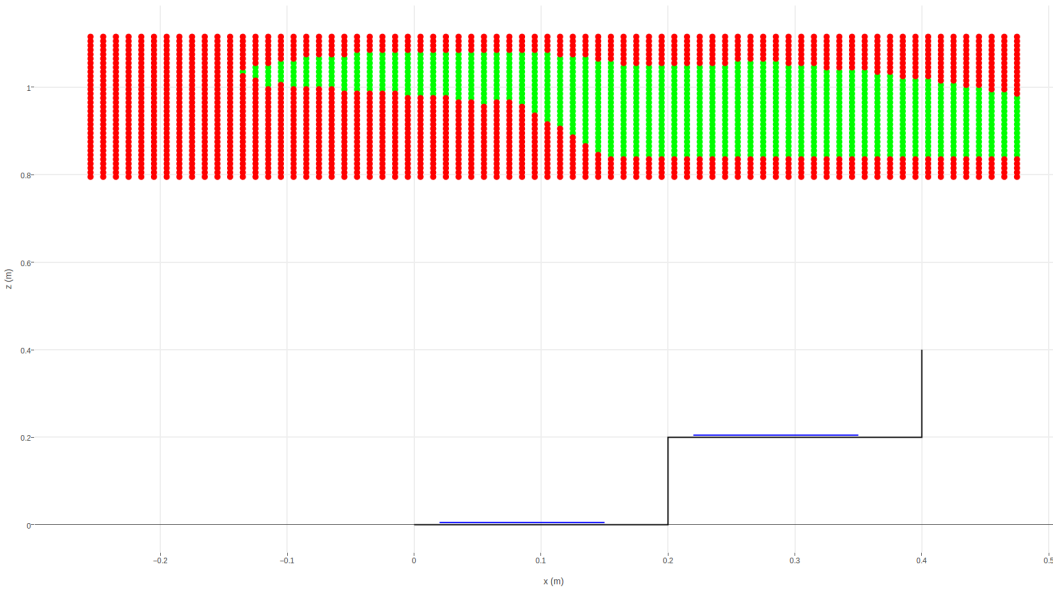
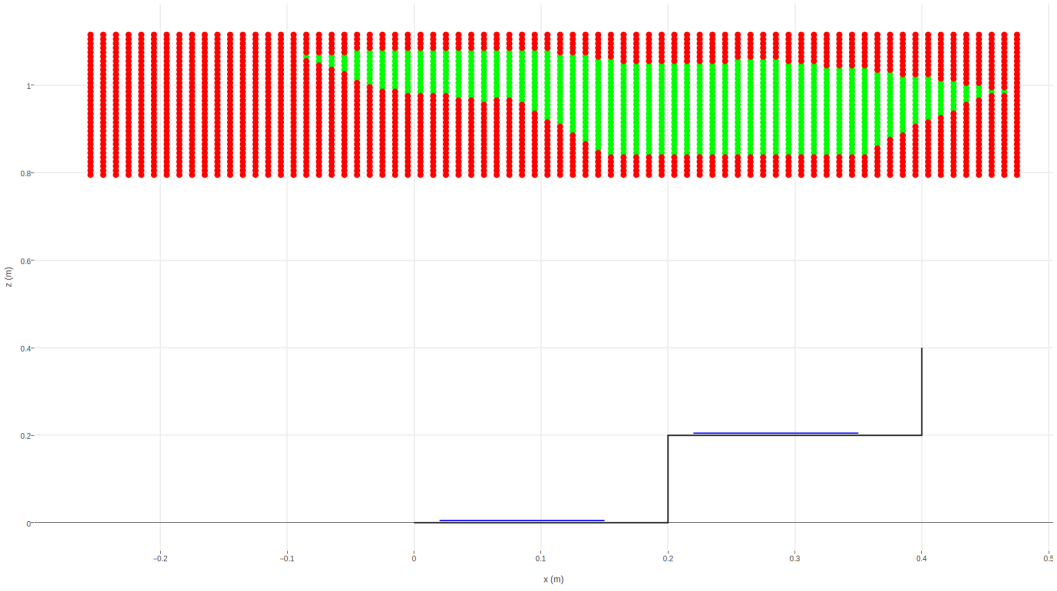


FIGURE 7.28 – Exemple de séquence de configurations en contact trouvée par notre planificateur de contact.



(a) Existence d'une configuration cinématiquement valide



(b) Existence d'une configuration cinématiquement valide et sans collision

FIGURE 7.29 – Cas du placement des pieds parallèlement aux marches : existence d'une configuration corps complet cinématique valide (a) et sans collision entre les jambes du robot et les marches (b) selon la position de l'origine du robot (dans le plan (x, z), avec la valeur de y placée entre les positions des deux pieds), avec des contacts fixés. En vert les positions de l'origine pour lesquelles il existe une configuration valide, en rouge celles où il n'en existe pas. Les traits noirs sont les marches de l'escalier, les traits bleus les positions des pieds.

Bilan

Cet exemple montre que le planificateur de contact peut trouver des comportements ou des démarches auxquels un humain ne penserait pas forcément et qui demanderait beaucoup de travail pour être testés et mis en oeuvre manuellement. Cependant, cela montre aussi que notre planificateur de contact est dépendant de l'heuristique utilisée par le générateur de contact.

De plus, les graphiques présentés montrent que dans ce type de scénarios contraints, la position de l'origine du robot est un paramètre très sensible lors de la génération de configurations en contact et sans collision. Or, nous rappelons que dans notre méthode de génération de contacts la position de l'origine est contrainte à suivre exactement le guide fourni en entrée.

Grâce à l'heuristique utilisée pour planifier ce guide, nous savons qu'il permet de générer des configurations cinématiquement valides, autrement dit la trajectoire guide sera comprise dans la zone valide du graphique 7.27(a). Mais, nous ne pouvons pas garantir que ce guide permettra de générer des configurations sans collisions (autrement dit qu'il est compris dans la zone valide de la figure 7.27(b)).

Ces résultats suggèrent qu'il serait intéressant de pouvoir relâcher la contrainte de suivre exactement la trajectoire guide lors de la planification de contact, surtout selon l'axe z qui semble très sensible.

7.2 CROC comme critère de faisabilité pendant la génération de contacts

Dans cette section, nous nous intéressons à évaluer empiriquement l'intérêt de l'utilisation de la méthode CROC comme critère de faisabilité pendant la génération de contact, ainsi que son impact en termes de performance. Pour quantifier l'impact de cette intégration, nous avons mesuré empiriquement les performances de la méthode de planification de contact et le taux de succès de la méthode génération de trajectoire centroïdale avec ou sans l'utilisation de la méthode CROC pendant la génération de contact.

Le résultats obtenus sont présentés dans le tableau 7.1. Pour chacun des scénarios, nous avons exécuté 100 fois la méthode de planification de contacts pour la même trajectoire guide en entrée, avec et sans utiliser CROC comme critère de faisabilité. Lorsqu'une séquence de contacts est produite, nous appelons la méthode de génération de trajectoire centroïdale de [Carpentier 2017c] avec cette séquence de contacts et nous mesurons son taux de succès.

Scenario	Méthode	Planification de contact				Traj. centroïdale
		Succès (%)	Temps (s)	candidats	config.	succès (%)
Marche (sol plat)	Sans CROC	100	0.58	8.2	6.3	98
	Avec CROC	100	0.63	21.9	7.0	100
Escalier (3 marches)	Sans CROC	100	0.61	24.4	6.1	52
	Avec CROC	94	0.82	87.3	7.3	100
Escalier (avec rambarde)	Sans CROC	98	1.24	144.3	11.6	31
	Avec CROC	84	1.57	322.6	13.2	100
Débris	Sans CROC	47	1.84	319.2	9.3	15
	Avec CROC	32	2.43	969.6	9.8	100

Tableau 7.1 – Évaluation de la faisabilité des séquences de contacts générées avec ou sans utiliser CROC comme critère de faisabilité durant la génération de contact, ainsi que de l'impact de l'utilisation de CROC sur les performances du générateur de contact. La colonne *Planification de contact* montre le taux de succès du planificateur (ie. quand la séquence de contacts produite atteint la position but), le temps de calcul, le nombre moyen de contacts candidats évalués et le nombre moyen de configurations dans la séquence produite. La dernière colonne montre le taux de succès de la méthode de génération de trajectoire centroïdale avec la séquence de contacts planifiée.

7.2.1 Impact de CROC sur les performances du planificateur de contacts

On peut observer sur le tableau 7.1 que l'utilisation de CROC pendant la génération de contacts dégrade légèrement les performances de la méthode de planification de contact. En effet, le temps de calcul est légèrement augmenté (de 8 à 30% selon les scénarios) et le taux de succès diminue.

L'augmentation du temps de calcul peut s'expliquer d'une part par l'ajout du temps de calcul nécessaire pour résoudre la méthode CROC. Mais cette augmentation s'explique surtout par le fait que l'algorithme doit évaluer beaucoup plus de candidats avant d'en trouver un valide, comme montré dans la cinquième colonne du tableau 7.1.

En réalité, la portion du temps de calcul utilisée par CROC est faible par rapport au temps total de planification de contact : entre 7 et 16% du temps total est utilisé par CROC, le reste est surtout utilisé par les méthodes de projection par cinématique inverse et de tests de collision. Cela montre bien que la méthode CROC est suffisamment efficace pour être utilisée pendant la planification de contact sans trop affecter les performances de la méthode.

7.2.2 Intérêt de CROC pendant la génération de contacts

La réduction du taux de succès du planificateur de contact en utilisant CROC est en fait virtuelle : l'introduction de CROC réduit le nombre de séquences de contact trouvées, mais garantit que le problème centroïdal défini par la suite est faisable, comme l'indique la dernière colonne du tableau 7.1.

Le premier point que l'on peut observer est que dans le cas de l'utilisation du critère de faisabilité, le taux de succès de la méthode de génération de trajectoire centroïdale est invariablement de 100%. Cela valide les développements théoriques de la seconde partie de ce manuscrit, et prouve que la méthode CROC est exacte : il n'y a aucune approximation supplémentaire par rapport à la méthode de génération de trajectoire centroïdale résolvant un problème non linéaire.

Dans le cas de la marche sur sol plat, l'intérêt de l'utilisation de CROC comme critère de faisabilité n'est pas évident car l'augmentation du taux de succès est marginale. Cela grâce aux heuristiques utilisées pendant la génération de contact qui donnent déjà de fortes probabilités de générer une séquence de contacts faisable.

Cependant dans tous les autres scénarios, l'intérêt de l'utilisation de CROC est mis en évidence. Le taux de succès de la génération de trajectoire centroïdale pour les séquences générées sans ce critère peut être extrêmement faible dans les scénarios complexes avec des contacts non coplanaires.

Conclusion sur l'utilisation de CROC comme critère de faisabilité pendant la génération de contact

Les résultats de cette section prouvent les arguments avancés dans la seconde partie de ce manuscrit : pour les scénarios complexes il est nécessaire d'utiliser un critère de faisabilité pendant la génération de contact afin d'obtenir un taux de succès élevé pour les blocs suivants de l'architecture de planification de mouvement.

La méthode CROC proposée permet de formuler un critère de faisabilité garantissant de manière exacte la faisabilité de la séquence de contacts produite et garantissant la convergence de la méthode de génération de trajectoire centroïdale avec cette séquence de contacts en entrée.

De plus, sa formulation linéaire par un LP de faible dimension permet à CROC d'être suffisamment peu coûteux en temps de calcul pour être intégré pendant la génération de contact sans trop réduire les performances de la méthode de planification de contact.

7.3 Performances de l'architecture complète

Afin de mesurer les performances de l'architecture complète de planification présentée dans le chapitre précédent nous avons testé cette architecture sur cinq scénarios typiques parmi ceux présentés au début de ce chapitre, avec 500 essais pour chaque scénario : une marche de 3 pas sur du sol plat, la navigation sur sol plat (un long mouvement de plus d'une vingtaine de pas), les débris, la montée des marches de 10cm et celle des marches de 15cm avec la rambarde. Puis, nous avons mesuré les performances et le taux de succès pour chaque bloc de l'architecture. Ces résultats sont présentés dans le tableau 7.2.

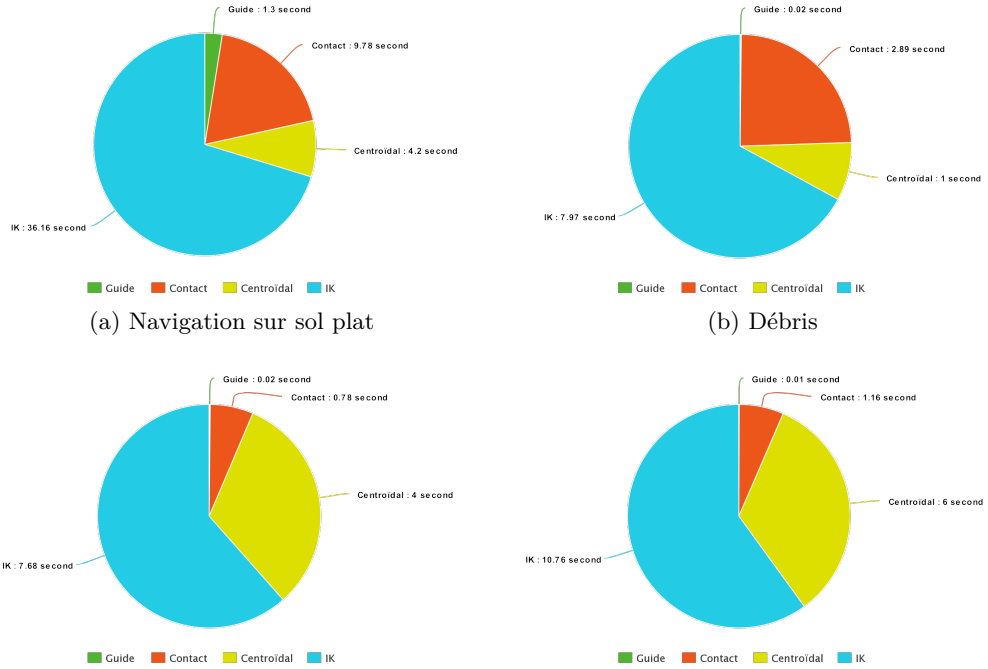
Scénario	Marche (3 pas)	
n. phases	7 - **7** - 7	
Durée mouvement (s)	7.2 - **7.7** - 8.3	
Méthode	temps (s)	succès (%))
Guide Kinodynamique	0.005 - **0.007** - 0.008	100.0
Contact + CROC	0.29 - **0.42** - 2.53	100.0
Traj. centroïdale*	$\sim$ **0.7**	100.0
Cinématique inverse	3.11 - **3.30** - 3.96	100.0
TOTAL	$\sim$ **4.43**	**100.0**

Scénario	Navigation sur sol plat		Débris	
n. phases	27 - **42.23** - 67		9.0 - **9.86** - 15.0	
Durée mouvement (s)	31.36 - **55.02** - 69.85		13.69 - **14.94** - 30.74	
Méthode	temps (s)	succès (%)	temps (s)	succès (%)
Guide Kinodynamique	0.87 - **1.3** - 11.49	100.0	0.01 - **0.02** - 0.03	100.0
Contact + CROC	1.4 - **9.78** - 115.15	99.4	1.07 - **2.89** - 18.78	32.67
Traj. centroïdale*	$\sim$ **4.2**	99.4	$\sim$ **1**	32.67
Cinématique inverse	18.19 - **36.16** - 68.01	98.67	5.93 - **7.97** - 16.73	27.28
TOTAL	$\sim$ **51.44**	**98.67**	$\sim$ **11.88**	**27.28**

Scénario	Escalier (3 marches)		Escalier+rambarde(3 m.)	
n. phases	13.0 - **13.93** - 19.0		16.0 - **17.81** - 23.0	
Durée mouvement (s)	15.25 - **16.23** - 21.9		19.92 - **23.13** - 31.53	
Méthode	temps (s)	succès (%)	temps (s)	succès (%)
Guide Kinodynamique	0.02 - **0.02** - 0.04	100	0.01 - **0.01** - 0.02	100
Contact + CROC	0.44 - **0.78** - 9.67	90.73	0.49 - **1.16** - 15.6	88.04
Traj. centroïdale*	$\sim$ **4**	90.73	$\sim$ **6**	88.04
Cinématique inverse	5.81 - **7.68** - 17.85	82.1	6.38 - **10.76** - 17.32	77.52
TOTAL	$\sim$ **12.48**	**82.1**	$\sim$ **17.93**	**77.52**

Tableau 7.2 – Analyse de performance de l'architecture complète de planification ainsi que de chaque bloc, pour 500 essais par scénario. Mis à part pour les taux de succès, les résultats sont écrits de la manière suivante : *min* - **moyenne** - *max*. Les temps de la ligne "TOTAL" ne considèrent que les temps des essais qui ont réussi. Les deux premières lignes de chaque tableau montrent le nombre de phases de contacts et la durée totale du mouvement pour chaque scénario. Les taux de succès affichés pour chaque blocs sont les taux de succès de la chaîne complète jusqu'à ce bloc et non le taux de succès de ce bloc seulement. * : les temps de calcul pour la méthode de génération de trajectoire centroïdale sont ceux présentés dans [Carpentier 2017c], extrapolés à partir des résultats présentés car les scénarios testés ne sont pas identiques.

Dans les sous sections suivantes nous analysons les performances de chaque bloc ainsi que la validité de la solution produite par le bloc précédent. Nous rappelons que nous utilisons le terme *temps de calcul interactif* pour désigner un temps de calcul qui est inférieur à la durée du mouvement produit.



(a) Navigation sur sol plat (b) Débris

(c) Escalier (d) Escalier + rambarde

FIGURE 7.30 – Répartition du temps de calcul pour chaque bloc de l'architecture complète.

Note sur le taux de succès des méthodes probabilistes

En ce qui concerne les taux de succès présentés dans le tableau 7.2 il faut faire une distinction importante entre les méthodes de planification de guide et de contact qui sont probabilistes et les méthodes de génération de trajectoire centroïdale et de cinématique inverse qui sont déterministes. Dans ce dernier cas, si une méthode déterministe échoue à résoudre un problème, cela signifie qu'elle échouera toujours avec les mêmes entrées. Au contraire dans le cas des méthodes probabilistes, un échec ne signifie pas forcément que la méthode ne peut résoudre ce problème. Relancer la méthode probabiliste pour le même problème avec les mêmes entrées peut éventuellement donner lieu à un succès.

Ici, les performances sont mesurées en n'autorisant qu'un seul essai pour chaque méthode. Mais il serait possible d'atteindre les 100% de réussite jusqu'à la planification de contacts en relançant les méthodes de planification après chaque échec. Évidement, le temps de calcul serait multiplié par le nombre d'essais requis. Les performances obtenues avec une telle stratégie sont montrées dans la sous-section 7.3.2.

7.3.1 Planification d'une trajectoire guide

Des résultats plus détaillés concernant la méthode de planification de trajectoire guide sont présentés à la fin de la première partie de ce manuscrit 2.4. Ici, les cinq scénarios sont relativement simples vis à vis de la planification de trajectoire guide. En effet, dans tous les cas nous obtenons un taux de succès de 100% et un temps de calcul très rapide de quelques millisecondes à quelques secondes pour le scénario le plus complexe.

Ces résultats valident néanmoins le fait que la planification de trajectoire guide kinodynamique est mise en oeuvre de manière efficace et résout le problème en un temps bien inférieur à la durée du mouvement.

7.3.2 Planification de contacts

D'après les résultats du tableau 7.2, la méthode de planification de contacts semble être la cause principale d'échec dans la chaîne complète de planification de mouvement.

Mais, puisque cette méthode est très rapide et trouve une solution avec un temps de calcul inférieur d'environ un ordre de grandeur à la durée du mouvement, nous pouvons la relancer après chaque échec jusqu'à ce qu'elle trouve une solution. En effet cette méthode étant probabiliste, il est possible qu'elle trouve une solution à un problème où elle a précédemment échoué si on la relance avec les mêmes entrées. Les performances obtenues en appliquant cette stratégie sont montrées dans le tableau 7.3.

Scénario	Nombre d'essais moyen	Temps de planification moyen (s)
Navigation sur sol plat	1.006	9.84
Débris	3.06	8.84
Escalier	1.10	0.86
Escalier + rambarde	1.14	1.32

Tableau 7.3 – Analyse des performances de la méthode de planification de contacts en adoptant une stratégie où le planificateur est relancé avec les mêmes entrées après chaque échec jusqu'à trouver une solution. Avec cette stratégie le taux de succès de ce bloc est toujours de 100% pour les scénarios considérés.

On peut donc observer que si on autorise la méthode de planification de contact à être relancée après chaque échec, on peut obtenir un taux de succès de 100% pour cette méthode avec seulement une augmentation minime du temps de calcul. Si l'on insère les temps de calcul du tableau 7.3 dans le tableau de l'architecture complète 7.2 on constate que le temps total de génération de mouvement est toujours inférieur à la durée du mouvement, à l'exception du scénario des débris où il sera supérieur de quelques secondes.

Bien que les temps de calcul moyens soient relativement faibles et bien inférieurs à la durée totale du mouvement (comme montré dans le tableau 7.2), ces temps de

calcul présentent un écart type très important avec des temps dans le pire cas plus important que la durée du mouvement. Afin de minimiser les temps dans le pire cas, un time-out peut être utilisé afin de relancer la méthode quand celle-ci semble prendre trop de temps pour converger. La valeur de ce time-out peut être définie proportionnellement à la durée de la trajectoire guide.

Améliorations possibles

On peut tout de même s'intéresser à essayer d'améliorer cette méthode afin d'augmenter son taux de succès à chaque essai. Pour cela il faut tout d'abord étudier les causes des échecs de la méthode.

Une cause possible pourrait être que la trajectoire guide fournie en entrée n'est pas faisable. En effet, bien que nous ayons montré théoriquement dans le chapitre 2 qu'il existe forcément des phases de contact permettant de suivre le guide, nous ne pouvons pas certifier qu'il existe des transitions faisables entre ces phases de contact. De plus, nous n'avons aucune garantie que les positions des points de contact de ces phases soient accessibles via une configuration respectant les contraintes cinématiques du robot et sans collision, seule une forte probabilité est assurée.

Cependant, ici en relançant plusieurs fois la méthode avec le même guide elle finit par trouver une solution. De plus, si on analyse les résultats obtenus dans le tableau 7.1 où la méthode de planification de contact est testée de nombreuses fois avec la **même** trajectoire guide, on observe que la méthode de planification de contact n'est pas complète et peut échouer avec une trajectoire guide pour laquelle il est prouvé qu'une solution existe.

Cela peut s'expliquer par le fait que cette méthode utilise un algorithme de type *glouton* : à chaque fois qu'un choix doit être fait, la méthode choisit la meilleure possibilité selon une heuristique et ne remet jamais ce choix en question. Autrement dit, une fois un contact valide choisi il sera forcément présent dans la séquence de contacts, il n'y a pas de retour en arrière possible pour changer ce contact.

Bien que ce fonctionnement permette d'obtenir des temps de calcul très rapides malgré l'importante dimension combinatoire du problème, c'est la cause des échecs de l'algorithme. En effet, il arrive que l'algorithme puisse se retrouver bloqué sans aucun candidat valide à cause d'un mauvais choix de contact fait précédemment.

Une possibilité proposée dans la littérature [Escande 2013, Short 2018] est de générer une structure de graphe en explorant différents choix en parallèle. Utiliser une méthode d'exploration similaire pourrait réduire les risques de rester bloqué sans choix de contacts valides. Mais cela se fera vraisemblablement au prix d'une forte augmentation du temps de calcul requis à cause de l'importante combinatoire du problème.

Pour cette raison nous préférons conserver l'algorithme actuel qui permet de réduire grandement la combinatoire du problème. Il est tout de même possible

d'ajouter la possibilité de changer le dernier contact choisi si celui-ci se révèle bloquant.

Plusieurs solutions sont également envisageables pour améliorer cette méthode. Tout d'abord la création de meilleures heuristiques de choix de contact, plus informées sur l'état futur du robot ou sur l'environnement proche, permettraient de choisir de meilleurs contacts. Cela augmenterait probablement le taux de succès en réduisant le choix de contact à priori valide mais qui devient bloquant dans le futur. Cependant, cela augmente l'importance du facteur humain dans l'architecture de planification complète en nécessitant un choix d'heuristique spécifique au type de mouvement désiré. De plus, cela rendrait la méthode moins générique ce qui va à l'encontre de la philosophie développée tout au long de cette thèse.

Un autre point est que la contrainte imposée de suivre exactement la trajectoire guide pendant la génération de contact est trop restrictive. Dans des cas complexes notamment dans les escaliers, il serait intéressant d'autoriser une liberté autour de guide surtout selon l'axe z.

Enfin, grâce aux approximations des contraintes cinématiques utilisées par la méthode CROC nous pourrions déterminer quel sera la prochaine patte en contact à devenir cinématiquement invalide. Cette information peut alors être utilisée lors du choix du contact à repositionner, afin d'essayer en priorité de repositionner ce contact pour éviter les changements inutiles de contact des autres pattes. L'expression de ces contraintes cinématiques pourrait également être utilisée pour déterminer les zones où le contact resterait cinématiquement valide pendant une durée suffisamment grande, puisque l'on connaît le déplacement futur du tronc le long de la trajectoire guide.

7.3.3 Cinématique inverse

On observe que l'étape de génération de mouvement corps-complet via la méthode de cinématique inverse peut échouer occasionnellement. Il s'agit soit d'échecs de la méthode de cinématique inverse (moins de 2%) soit de cas où la méthode produit un mouvement ne respectant pas les butés articulaires du robot.

De plus, cette méthode présente des temps de calcul relativement longs. En effet, il s'agit de la méthode la plus coûteuse en temps de calcul de l'architecture, comme montré sur la figure 7.30. Bien qu'elle présente des performances interactives, elle nécessite un temps de calcul d'environ 80% de la durée totale du mouvement, ce qui laisse trop peu de temps aux autres méthodes de l'architecture.

Afin d'améliorer les performances, l'utilisation d'une librairie C++ est envisagée dans un futur proche pour remplacer le script python actuel.

De plus, une autre méthode visant à remplacer la cinématique inverse est en cours de développement au sein de l'équipe. Il s'agit d'une méthode de *Differential Dynamic Programming* (DDP) [Budhiraja 2018]. Une fois ces travaux aboutis, son

intégration dans l'architecture complète pourra se faire aisément puisque les entrées et les sorties seront identiques au bloc de cinématique inverse actuel.

A terme, ce DDP pourra également optimiser les trajectoires des organes terminaux.

7.3.4 Génération de trajectoires pour les organes terminaux

En ce qui concerne la génération de trajectoires pour les organes terminaux, la production des trajectoires par défaut requiert moins de $20\mu s$ de calcul, ce qui est négligeable. Par contre, le temps de calcul nécessaire pour produire des trajectoires sans collision en environnement contraint est actuellement supérieur à la durée totale du mouvement, principalement à cause des itérations nécessaires avec la méthode de cinématique inverse.

En effet, la première étape pour la génération de trajectoire des organes terminaux décrite dans la section 6.7 est la méthode *Limb-RRT*. Cette méthode requiert entre $300ms$ et 2.5 secondes avec une moyenne d'environ 1 seconde, pour chaque phase où un organe terminal bouge d'un contact vers un autre. La méthode de déformation de la courbe est elle très rapide et ne requiert qu'environ $50\mu s$.

Ensuite, il faut exécuter à nouveau la méthode de cinématique inverse avec la nouvelle référence pour la trajectoire de l'organe terminal. Or, comme expliqué dans la sous-section précédente, cette méthode est la plus coûteuse de l'architecture.

Le fait d'itérer avec la cinématique inverse pour tester plusieurs trajectoires des organes terminaux semble donc être une approche trop coûteuse en terme de temps de calcul. En effet, selon les scénarios il faut entre 3 et 10 itérations avant de trouver une trajectoire valide, et chaque itération nécessite un temps de calcul seulement légèrement inférieur à la durée du mouvement.

En plus d'une optimisation de l'implémentation, une solution possible serait de toujours prendre la courbe qui correspond le mieux à la trajectoire trouvée par le Limb-RRT. Ainsi, il n'y aurait qu'une seule itération de l'inverse cinématique. Cependant, cette solution reviendrait à utiliser des trajectoires nécessitant des mouvements inutilement complexes du robot.

Toutefois, comme dans de nombreux cas la trajectoire par défaut est valide, il n'y a qu'une faible portion des phases de contact du mouvement complet qui nécessitent l'utilisation du Limb-RRT et des itérations avec la cinématique inverse. Dans ces cas, le temps de calcul total n'est que faiblement impacté. l'exception est bien sûr les scénarios où la trajectoire par défaut n'est jamais valide, comme par exemple dans le cas des escaliers.

Faisabilité des mouvements des organes terminaux

La méthode de génération de trajectoires pour les organes terminaux peut échouer car un aspect important de la faisabilité du mouvement est actuellement

ignoré dans toutes les étapes précédentes de l'architecture présentée. Il s'agit de la faisabilité du mouvement des organes terminaux : existe-t-il une trajectoire pour les organes terminaux entre deux points de contact, tel que le robot puisse produire un mouvement cinématiquement valide et sans collision en suivant cette trajectoire ?

Cet aspect de la faisabilité devrait principalement être pris en compte lors de la génération de la séquence de contacts. En effet, il faut assurer la faisabilité du mouvement des organes terminaux entre chaque point de contact planifié.

Il est également important de la prendre en compte pendant la génération de trajectoire centroïdale car en imposant une certaine trajectoire pour le centre de masse il est possible de rendre infaisable un mouvement corps complet alors qu'il existait une solution pour une autre trajectoire centroïdale.

De manière pratique, pour les environnements considérés, la faisabilité du mouvement des organes terminaux est rarement un problème. En effet, le seul scénario où la méthode de génération de trajectoire pour les organes terminaux a échoué durant nos tests est le scénario des débris.

Dans certaines séquences de contact trouvées pour ce scénario, le pied était placé très au bord des plateformes inclinées. Le résultat est que lors du mouvement pour poser ou lever le pied, il entrait invariablement en collision avec une des plateformes adjacentes.

Ce problème a été réglé de manière pragmatique en appliquant un facteur d'échelle sur les organes terminaux du robot pendant la planification de contact afin de laisser une marge de sécurité entre ces organes terminaux et les obstacles. Cela a également comme intérêt d'augmenter la robustesse du mouvement par rapport aux imprécisions.

7.3.5 Conclusion sur l'architecture de planification de mouvement

Si on analyse les performances globales de l'architecture complète de planification de mouvement, en relançant la méthode de planification de contact après chaque échec jusqu'à ce qu'elle trouve une solution, nous obtenons un taux de succès relativement élevé, d'environ 85% dans le pire cas, comme montré dans le tableau 7.4.

Si l'on adopte cette stratégie la seule source d'échec possible restante dans l'architecture proposée deviens la méthode de génération de mouvement corps complet par cinématique inverse.

En ce qui concerne le temps de calcul, nous obtenons bien des performances interactives en moyenne puisque le temps de calcul total est inférieur à la durée du mouvement. L'exception ici est le scénario des débris dont le temps de calcul est supérieur de quelques secondes à la durée du mouvement.

Nous estimons que nous pouvons atteindre de meilleures performances afin de garantir un temps interactif dans le pire cas simplement par une optimisation de

l'implémentation de certaines méthodes, notamment la méthode de cinématique inverse.

Scénario	Durée du mouvement (s)	Temps de calcul total (s)	Taux de succès (%)
Marche (3 pas)	7.7	4.43	100
Navigation sur sol plat	55.02	51.5	99.3
Débris	14.94	17.83	83.5
Escalier	16.23	12.56	90.5
Escalier + rambarde	23.13	18.09	88.05

Tableau 7.4 – Analyse des performances de l'architecture de planification de mouvement complète, en adoptant une stratégie où le planificateur de contact est relancé avec les mêmes entrées après chaque échec jusqu'à trouver une solution.

Les performances mesurées dans le cas du scénario de marche avec 3 pas montrent qu'il est tout à fait possible d'utiliser notre méthode avec une formulation similaire à celle de la commande prédictive qui utilise un horizon temporel de quelques pas dans le futur. Dans ce cas, nous sommes capables de générer la position des contacts pour les 3 prochains pas en environ $400ms$, ce qui est équivalent à la méthode proposée dans [Ponton 2016] utilisant une méthode d'optimisation de mixte entier.

7.4 Utilisation du robot réel

A cause des limitations imposées par le stabilisateur bas-niveau du robot HRP-2 décrites dans le chapitre précédent, peu de nouveaux mouvements ont pu être exécutés sur le robot réel.

Deux mouvements typiques ont toutefois été exécutés avec succès sur le robot : de la navigation sur sol plat et la montée des escaliers. Bien que ces mouvements ne soient pas originaux, dans notre cas ils ont été automatiquement et entièrement générés seulement à partir d'une position but pour le centre du robot. Ces mouvements se sont également montrés répétables avec un très bon taux de succès sur le robot réel.

Conclusion

Le but de cette thèse était de faire progresser les méthodes de **planification** de mouvements pour la **locomotion de robots à pattes**, dans le cas de scénarios **multi-contact** (avec une séquence de contacts acyclique et des contacts non coplanaires) et **dynamiques**. Les travaux de cette thèse s'inscrivent dans le contexte de l'architecture de planification découplée proposée par l'équipe Gepetto pour le projet Loco-3D. Cette architecture consiste à découpler le problème de la locomotion en trois sous-problèmes résolus séquentiellement. La planification d'une trajectoire guide (noté $\mathcal{P}_1$), la planification d'une séquence de contacts suivant ce guide (noté $\mathcal{P}_2$) puis la génération d'une trajectoire centroïdale et enfin d'un mouvement corps complet suivant cette séquence de contacts (noté $\mathcal{P}_3$).

La problématique, pour pouvoir utiliser cette approche, est de garantir que la solution de chaque sous-problème est une entrée **faisable** pour le sous problème suivant. C'est à dire, qu'il existe une solution au sous-problème suivant avec cette entrée.

Cette problématique existe pour toutes les méthodes utilisant des modèles réduits ou simplifiés du robot, cependant cette thèse est focalisée sur le contexte de l'architecture proposée pour le projet Loco-3D.

Durant cette thèse nous avons proposé deux **critères de faisabilité**. Dans la première partie nous avons proposé une méthode locale capable de générer une trajectoire optimale en temps pour le centre du robot, avec une accélération admissible selon la condition de non glissement des organes terminaux en contact.

Nous avons ensuite utilisé cette méthode de planification locale pour proposer l'algorithme **RB-RRT kinodynamique** capable de résoudre le sous-problème $\mathcal{P}_1$. L'algorithme RB-RRT original offrait déjà une forte probabilité qu'il existe une séquence de configurations en contact cinématiquement valides le long du guide planifié, mais sans considérer la faisabilité dynamique. La nouvelle version kinodynamique offre maintenant une forte probabilité qu'il existe une séquence de configurations, cinématiquement et dynamiquement valides.

Cette contribution permet donc d'obtenir une forte probabilité que la sortie du sous-problème $\mathcal{P}_1$ sera une entrée **faisable** pour le sous-problème $\mathcal{P}_2$.

Dans la seconde partie de ce manuscrit nous avons défini le problème de la faisabilité d'une transition de contact : existe-t'il une trajectoire centroïdale valide permettant de connecter exactement deux états du robot, associés à deux configurations en contact ?

Ensuite, nous avons proposé une méthode fiable et efficace permettant de résoudre ce problème avec un temps de calcul de l'ordre de la milliseconde, nommée **CROC**. L'efficacité de cette méthode est obtenue grâce à une reformulation convexe du problème de la dynamique centroïdale. Cette reformulation est conservatrice,

mais contrairement aux autres reformulations convexes de l'état de l'art, elle n'est pas limitée au cas quasi-statique et ne fait pas d'approximation ou de relaxation des contraintes dynamiques. Dans notre cas, la convexité est obtenue grâce à un choix judicieux de représentation de la trajectoire centroïdale : une courbe de Bézier avec un seul point de contrôle libre utilisé comme variable de notre problème.

Le faible temps de calcul de notre méthode (au moins un ordre de grandeur inférieur aux méthodes de génération de trajectoire centroïdale de l'état de l'art) nous permet d'utiliser cette méthode comme un critère de faisabilité durant la planification de contact sans augmenter de manière significative le temps de calcul de la méthode de planification de contact. Cela nous permet de garantir qu'il existe une trajectoire centroïdale valide entre chaque configuration en contact présente dans la séquence produite par notre planificateur de contact.

Cette contribution permet donc de garantir exactement la **faisabilité** de la sortie du sous-problème $\mathcal{P}_2$ par rapport à la méthode de génération de trajectoire centroïdale du sous-problème $\mathcal{P}_3$.

Dans la dernière partie de ce manuscrit nous avons détaillé l'architecture complète de planification de mouvement multi-contact mise en oeuvre durant la thèse, utilisant les deux critères de faisabilité proposés précédemment. Nous avons également proposé une méthode de génération de trajectoires de vol entre deux positions en contact pour les pattes du robot, permettant de leur assurer un mouvement cinématiquement valide et sans collision. Puis, nous avons évalué les performances obtenues ainsi que la justesse et l'intérêt de ces critères de faisabilité et nous avons montré des exemples de scénarios résolus via notre méthode.

En ce qui concerne la faisabilité entre $\mathcal{P}_1$ et $\mathcal{P}_2$, l'intérêt de notre contribution est difficile à quantifier empiriquement car la méthode de planification de contact ($\mathcal{P}_2$) est probabiliste et n'est pas complète. A cause de cela, nous ne pouvons pas déduire précisement si la cause des échecs durant la résolution de $\mathcal{P}_2$ est dû à une solution infaisable trouvée pour $\mathcal{P}_1$. Toutefois pour les scénarios évalués dans le chapitre 7, il a été possible d'obtenir un taux de succès de 100 % en relançant suffisamment de fois la méthode de planification de contact avec le même guide jusqu'à obtenir un succès. Cela montre que le guide était toujours valide. De plus, les résultats qualitatifs sur des scénarios précis présentés dans la première partie du manuscrit montrent clairement l'intérêt de la méthode **RB-RRT kinodynamique** proposée durant cette thèse, cette dernière étant notamment capable de résoudre des scénarios hors de portée de la version originale de RB-RRT.

Concernant le critère de faisabilité entre $\mathcal{P}_2$ et la génération de trajectoire centroïdale de $\mathcal{P}_3$, nos résultats théoriques nous assurent que le critère proposé est exact. Cette propriété est vérifiée empiriquement puisque dans l'ensemble des cas considérés, la solution trouvée pour le sous-problème $\mathcal{P}_2$ est une entrée faisable pour la méthode de génération de trajectoire centroïdale utilisée pendant le sous-problème $\mathcal{P}_3$, et ce dernier présente un taux de succès de 100% dans notre architecture. Par ailleurs, il a été empiriquement démontré qu'au sein de notre architecture, l'utili-

sation de **CROC** comme critère de faisabilité lors de la planification de contacts résulte en une nette amélioration du taux de succès obtenu par notre architecture dans son ensemble.

La faisabilité entre la méthode de génération de trajectoire centroïdale et la méthode de génération de mouvement corps complet n'a pas été étudiée durant cette thèse, qui s'est appuyée sur des travaux existants pour compléter notre chaîne de génération de mouvements [Carpentier 2017c]. En l'état, la méthode de génération de mouvement corps complet peut échouer ou produire un mouvement cinématiquement invalide à cause d'une mauvaise trajectoire centroïdale de référence en de rares cas, et des travaux ultérieurs tenteront de proposer un critère de faisabilité pertinent pour ce problème.

Finalement, notre contribution technique permet d'obtenir une architecture complète de planification de mouvement en **multi-contact** et non limitée aux cas quasi-statiques.

Bilan

Si l'on compare les caractéristiques de l'architecture complète obtenues avec la liste de caractéristiques désirées qui a été évoquée durant l'introduction (section Bilan de l'introduction), nous constatons que nous remplissons la majorité des points requis, mais qu'il reste encore des progrès possibles pour certains points :

Génère des mouvements multi-contact : notre méthode est bien capable de générer des mouvements acycliques et avec des contacts non coplanaires, comme montré sur de nombreux exemples.

N'est pas limité aux mouvements quasi-statiques : nous sommes capables de produire des mouvements dynamiques et de résoudre des scénarios sans solutions quasi-statiques. Cela est montré dans plusieurs scénarios d'exemples.

Produit un mouvement dans un environnement complexe avec des obstacles non convexes : l'utilisation d'un algorithme de planification probabiliste nous permet de pouvoir considérer un environnement complexe sans aucune contrainte sur la forme ou la convexité des obstacles. De plus, cette méthode permet dans le cas échéant de ne pas tomber dans des minima locaux.

Produit un mouvement respectant les contraintes dynamiques : les mouvements générés respectent les contraintes dynamiques appliquées au robot. Ceci a été prouvé en simulation et sur le robot réel HRP-2.

Produit un mouvement respectant les contraintes cinématiques et sans (auto-)collisions : les contraintes cinématiques et de non collision sont correctement prises en compte sauf à l'étape de la cinématique inverse. La séquence de

configurations en contact planifiée est bien sans collision mais le mouvement corps-complet connectant chacune de ces configurations n'est pas garanti d'être sans collision pour le tronc du robot. La trajectoire guide assure toutefois qu'il existe une solution sans collision pour la boite englobante utilisée pour représenter le tronc du robot. Mais, la méthode de cinématique inverse ne considérant pas les obstacles, elle peut trouver un mouvement différent qui entre en collision. Les contraintes cinématiques et de non collision sont bien respectées pour les membres du robot grâce à notre méthode de génération de trajectoire pour les organes terminaux, mais si le tronc entre en collision la cinématique inverse n'est actuellement pas capable de produire un mouvement évitant l'obstacle.

Ne nécessite pas d'intervention humaine pour prédéfinir des caractéristiques du mouvement : les interventions humaines nécessaires sont très réduites durant l'ensemble de l'architecture mais pas encore inexistantes. En plus de la définition de la tâche de locomotion en entrée de notre méthode (position initiale et finale du centre du robot) deux autres choix de paramètres doivent être faits par un expert humain. D'une part, le choix de l'heuristique utilisée lors de la création de contacts. D'autre part, le choix de la fonction de coût utilisée pour la méthode d'optimisation de trajectoire centroïdale et des différents poids de celle-ci. Cependant des choix très génériques peuvent être faits, par exemple pour les quatre scénarios analysés dans la section 7.3 la même heuristique de création de contact est utilisée pour les quatre scénarios et la fonction de coût est identique avec seulement un changement des valeurs des poids entre le scénario de marche sur sol plat et les autres.

Est fiable : Bien que nous n'ayons aucune garantie théorique de convergence, les résultats empiriques montrent que le taux de succès est proche de 100% et est d'environ 85% dans le pire cas pour les scénarios complexes. La principale cause d'échec est la méthode de génération de mouvement corps complet. Il faut noter toutefois que la méthode de planification de contact n'est pas complète et peut nécessiter d'être relancée plusieurs fois avant de trouver une solution. De plus, la méthode de génération de trajectoire centroïdale utilise une méthode de résolution non linéaire qui n'offre aucune garantie de convergence, mais l'initialisation proposée dans le chapitre 4 permet d'obtenir un taux de succès de 100% de cette méthode au sein de notre architecture. Enfin, en ce qui concerne la méthode de planification de guide, la méthode de planification locale est complète mais le RB-RRT kinodynamique n'est pas complet à cause des heuristiques et des approximations utilisées.

Présente un temps de calcul interactif : Comme montré par les résultats du chapitre 7, notre méthode atteint des performances interactives en moyenne, mais le temps de calcul dans les pires cas peut dépasser la durée totale du mouvement (en restant dans le même ordre de grandeur).

Perspectives futures

Les méthodes proposées durant cette thèse présentent plusieurs limitations et nous serions intéressé à continuer de chercher comment les dépasser.

Le principal problème de la méthode RB-RRT Kinodynamique est que la méthode de planification locale nécessite de connaître exactement les contacts entre le robot et l'environnement mais que cette information n'est pas disponible durant la résolution de $\mathcal{P}_1$. Dans le chapitre 2 nous avons proposé une heuristique permettant d'approximer grossièrement la position des contacts durant la planification de guide. Bien que nous ayons montré que cette heuristique n'introduit pas d'erreurs, elle limite les solutions que l'on peut trouver. Il serait alors intéressant d'utiliser une méthode plus efficace pour estimer la position des contacts durant $\mathcal{P}_1$. Une approche qui nous semble prometteuse serait d'utiliser des méthodes d'apprentissage afin d'apprendre la distribution de probabilité du placement des contacts en fonction de l'état courant, de l'environnement proche et de la direction de déplacement future.

Une autre piste à explorer serait d'utiliser CROC pendant la planification locale du RB-RRT kinodynamique. Une application directe et naïve n'est pas possible car elle nécessiterait d'introduire la combinatoire du choix de contact dans le sous-problème $\mathcal{P}_1$, ce que nous cherchons à éviter. Cependant, la méthode de planification locale actuelle et la méthode CROC résolvent des problèmes proches et nous pensons qu'il serait intéressant d'étudier plus en détail les interactions possibles entre ces deux méthodes.

La méthode CROC présente deux limitations principales. Tout d'abord, la durée de chaque phase de contact est fixée et est une entrée du problème. Dans le chapitre 4 nous avons proposé une méthode permettant de sélectionner des combinaisons de durée afin de ne pas perdre de solutions. Mais cette méthode nécessite d'évaluer chaque type de robot et d'environnement. Il est possible de trouver des environnements présentant de nouvelles propriétés et nécessitant d'utiliser des combinaisons de durées que nous n'avions pas trouvées précédemment. L'intégration des instants de transition comme variables du problème a déjà été fait dans la littérature [Ponton 2018, Winkler 2018] mais cela augmente le temps de résolution, qui est critique dans notre cas d'application consistant à utiliser CROC durant la planification de contact. Nous pourrions plutôt rechercher des stratégies d'échantillonnage des durées plus efficaces.

Ensuite, la convexité de la formulation de CROC vient du fait que l'on a qu'un seul point de contrôle variable pour la courbe représentant la trajectoire centroïdale. Cela contraint fortement le problème et réduit l'espace des solutions explorées, notamment en ne permettant qu'un seul point d'inflexion possible sur la trajectoire centroïdale. Cette limitation nous empêche de considérer une séquence de contacts avec de nombreuses phases car généralement dans un mouvement il y a un point d'inflexion sur la trajectoire centroïdale par repositionnement de contact. Cela nous force alors à fixer des états intermédiaires, qui sont les état finaux ou initiaux pour

chaque problème considérant des séquences de deux ou trois phases de contact, comme montré dans la figure 6.3. Pour considérer des séquences de contact plus longues, afin d'obtenir une trajectoire plus lisse sans points intermédiaires fixés, il faudrait ajouter plusieurs points de contrôles variables au problème défini dans le chapitre 4. Cela rendrait le problème non linéaire mais nous souhaitons explorer la faisabilité et l'efficacité d'une méthode de génération de trajectoire basée sur cette formulation.

Enfin, une dernière limitation qui n'est pas propre à la méthode CROC mais commune à toutes les méthodes de génération de trajectoire centroïdale est l'expression des contraintes cinématiques. Dans le chapitre 3 nous avons proposé une approximation convexe de ces contraintes, qui est nécessaire mais non suffisante. A notre connaissance il n'y a pas de méthode permettant d'exprimer exactement et efficacement les contraintes cinématiques du modèle complet pour le modèle centroïdal. Il s'agit d'une piste de recherche intéressante. Les méthodes permettant d'itérer entre le modèle dynamique centroïdal et le modèle cinématique complet comme [Dai 2014] permettent de résoudre ce problème dans le cas d'utilisation des méthodes de génération de trajectoires. Mais le sur-coût en temps de calcul n'est pas possible dans notre cas d'utilisation pour résoudre le problème de la faisabilité d'une transition durant la planification de contact.

En ce qui concerne la méthode d'optimisation de trajectoire centroïdale, deux possibilités s'offrent à nous. La première serait d'utiliser directement CROC, avec les améliorations évoquées dans le paragraphe précédent. Cette approche conviendrait dans le domaine de l'animation graphique mais ne serait peut être pas suffisante pour optimiser les fonctions de coût complexes utilisées en robotique et n'est pas capable d'optimiser la durée de chaque phase de contact. La seconde serait d'opter pour la méthode proposée dans [Ponton 2018], en essayant d'y appliquer notre formulation continue des contraintes. Dans tous les cas, il faudra mettre en place une méthode similaire au filtre dynamique, qui permette d'itérer entre la méthode de génération de trajectoire centroïdale et celle de génération de mouvement corps-complet afin de correctement prendre en compte le moment cinétique.

Un point bloquant de l'architecture actuelle est que la méthode de génération de mouvements corps complet utilisée n'est pas capable de produire un mouvement évitant les collisions pour le tronc du robot. Des méthodes existantes permettent bien de produire des mouvements corps complet avec évitement d'obstacles, mais elles ne permettent pas d'obtenir des performances interactives tout en étant capable de produire un mouvement dynamiquement consistant et non limité au cas quasi-statique.

Une autre piste de recherche qui nous semble prometteuse est l'application de la formulation continue des contraintes, proposée dans le chapitre 5, à d'autres méthodes de génération de trajectoire centroïdale. Cette formulation devrait pouvoir s'appliquer à n'importe quelle méthode de l'état de l'art, y compris les méthodes

non linéaires, et permettre de vérifier exactement les contraintes pour toute la trajectoire sans recourir à la discrétisation. Une étude de la faisabilité et de l'impact de l'utilisation de cette formulation est toutefois nécessaire.

Finalement, un travail d'optimisation peut être fait sur différentes méthodes de l'architecture complète. Il s'agit soit d'une amélioration des méthodes (pour le planificateur de contact) soit d'une optimisation de l'implémentation (pour la cinématique inverse et la génération des trajectoires des pattes). Nous espérons que ces travaux permettront de conserver des performances interactives même dans le pire cas d'exécution.

Le projet Loco-3D

La vision à long terme du projet Loco-3D [Carpentier 2017b] dans lequel s'inscrit cette thèse est d'embarquer cette architecture de planification dans le robot et de la connecter aux méthodes de perception en entrée et de contrôle en sortie. Le résultat serait un robot pouvant effectuer des tâches de locomotion dans un environnement quelconque et inconnu, capable de produire des mouvements en **multi-contact** sans interventions humaines.

Pour parvenir à cela, il faut tout d'abord finir le développement d'un contrôleur bas-niveau pour notre robot, capable de réaliser des mouvements avec des contacts non-coplanaires et non limités au sol plat, contrairement au contrôleur actuel.

La connexion avec des méthodes de perception en entrée ne changera pas notre méthode. Au lieu de prendre un modèle 3D de l'environnement manuellement construit en entrée, notre méthode prendra un modèle reconstruit à partir des diverses données des capteurs. De même, au lieu de spécifier une position initiale celle-ci sera estimée par les méthodes de perception. La grande différence sera que ces données ne seront plus exactes mais imprécises et incertaines, il faudra donc assurer la robustesse du mouvement produit par rapport à ces incertitudes.

Finalement, le bouclage entre la perception et l'action nous permettra de corriger les erreurs et les imprécisions durant le mouvement. Mais pour être capable d'atteindre ce but, notre méthode de planification de mouvement doit être capable de re-planifier le mouvement en fonction des nouvelles informations. De plus, la méthode de planification de mouvement doit pouvoir être utilisée avec un horizon temporel pour planifier seulement la prochaine dizaine de secondes de mouvement et non le mouvement complet jusqu'à la position finale comme actuellement.

Nous pensons que tout ces développements vont permettre de faire un premier pas vers une méthode générique de planification de mouvement multi-contact pour robots à pattes et qu'ainsi nous pourrons nous rapprocher de l'arrivée de robots à pattes autonomes, capables de se déplacer librement et automatiquement dans n'importe quel environnement.

Annexe A

Développements liés aux courbes de Bézier

Dans cette annexe, nous considérons une courbe de Bézier $\mathbf{c}(t)$ de degré n et de durée T qui s'exprime de la manière suivante :

$$\mathbf{c}(t) = \sum_{i=0}^{n} B_i^n(t/T)\mathbf{P}_i \tag{A.1}$$

Où les $\mathbf{P}_i$ sont les points de contrôle de la courbe de Bézier. Il y a donc $n+1$ points de contrôle pour une courbe de degré n. Les B_i^n sont les polynômes de Bernstein, définis par :

$$B_i^n(u) = \binom{n}{i} u^i(1-u)^{n-i},\ u \in [0;1] \tag{A.2}$$

Avec $\binom{n}{i}$ les coefficients binomiaux, ce qui peut se développer ainsi :

$$B_i^n(u) = \frac{n!}{i!(n-i)!}u^i(1-u)^{n-i},\ u \in [0;1] \tag{A.3}$$

Tel que :

$$\sum_{i=0}^{n} B_i^n(u) = 1,\ \forall u \in [0;1] \tag{A.4}$$

Dans nos travaux nous considérons des courbes de dimension supérieur à 1, la dimension d'une courbe est définie par la dimension de ses points de contrôle. Une courbe de dimension m est définie par des points de contrôle $\mathbf{P}_i \in \mathbb{R}^m$.

A.1 Dérivation d'une courbe de Bézier

Propriété : La dérivée d'une courbe de Bézier $\mathbf{c}(t)$ de degré n et de durée T est une courbe de Bézier $\dot{\mathbf{c}}(t)$ de degré $n-1$ et de durée identique T. La valeur des points de contrôle $\mathbf{P}'_i$ de $\dot{\mathbf{c}}(t)$ dépend seulement des points de contrôle $\mathbf{P}_i$ de $\mathbf{c}(t)$:

$$\mathbf{P}'_i = \frac{n}{T}(\mathbf{P}_{i+1} - \mathbf{P}_i),\ \forall i \in [0, n-1] \tag{A.5}$$

A.2 Conditions aux limites

Dans cette section nous considérons une courbe $\mathbf{c}(t)$ devant connecter exactement deux états $\mathbf{x}_s = (\mathbf{c}_s, \dot{\mathbf{c}}_s, \ddot{\mathbf{c}}_s)$ et $\mathbf{x}_g = (\mathbf{c}_g, \dot{\mathbf{c}}_g, \ddot{\mathbf{c}}_g)$, en respectant les positions, vitesses et accélérations initiales et finales. Pour cela, il suffit d'imposer la valeur initiale et finale de $\mathbf{c}(t)$, $\dot{\mathbf{c}}(t)$ et $\ddot{\mathbf{c}}(t)$:

$$\begin{cases} \mathbf{c}(0) & = \mathbf{c}_s \\ \mathbf{c}(T) & = \mathbf{c}_g \\ \dot{\mathbf{c}}(0) & = \dot{\mathbf{c}}_s \\ \dot{\mathbf{c}}(T) & = \dot{\mathbf{c}}_g \\ \ddot{\mathbf{c}}(0) & = \ddot{\mathbf{c}}_s \\ \ddot{\mathbf{c}}(T) & = \ddot{\mathbf{c}}_g \end{cases} \tag{A.6}$$

Or, par construction la valeur initiale (respectivement finale) d'une courbe de Bézier ne dépend que de son premier (respectivement dernier) point de contrôle. En effet d'après l'équation (A.4), nous avons :

$$\begin{aligned} B_0^n(0) = 1 \text{ et } B_i^n(0) = 0 \ , \forall i \in [1; n] \\ B_n^n(1) = 1 \text{ et } B_i^n(1) = 0 \ , \forall i \in [0; n-1] \end{aligned} \tag{A.7}$$

D'après la formulation générale d'une courbe de Bézier (A.1) nous avons bien :

$$\mathbf{c}(0) = \mathbf{P}_0 \text{ et } \mathbf{c}(T) = \mathbf{P}_n \tag{A.8}$$

Les conditions (A.6) peuvent donc se réécrire ainsi :

$$\begin{cases} \mathbf{P}_0 & = \mathbf{c}_s \\ \mathbf{P}_n & = \mathbf{c}_g \\ \mathbf{P}'_0 & = \dot{\mathbf{c}}_s \\ \mathbf{P}'_{n-1} & = \dot{\mathbf{c}}_g \\ \mathbf{P}''_0 & = \ddot{\mathbf{c}}_s \\ \mathbf{P}''_{n-2} & = \ddot{\mathbf{c}}_g \end{cases} \tag{A.9}$$

Avec respectivement $\mathbf{P}_i$, $\mathbf{P}'_i$,$\mathbf{P}''_i$ les points de contrôle de $\mathbf{c}(t)$, $\dot{\mathbf{c}}(t)$ et $\ddot{\mathbf{c}}(t)$.

Or, grâce à la propriété de dérivation de la section précédente, nous pouvons exprimer ces points de contrôle en fonction seulement de ceux de $\mathbf{c}(t)$ car :

$$\begin{aligned} \mathbf{P}'_i &= \frac{n}{T}(\mathbf{P}_{i+1} - \mathbf{P}_i) \\ \mathbf{P}''_i &= \frac{n(n-1)}{T^2}(\mathbf{P}_{i+2} - 2\mathbf{P}_{i+1} + \mathbf{P}_i) \end{aligned} \tag{A.10}$$

On obtient alors :

$$\begin{cases} \mathbf{P}'_0 & = \frac{n}{T}(\mathbf{P}_1 - \mathbf{P}_0) \\ \mathbf{P}'_{n-1} & = \frac{n}{T}(\mathbf{P}_n - \mathbf{P}_{n-1}) \\ \mathbf{P}''_0 & = \frac{n(n-1)}{T^2}(\mathbf{P}_2 - 2\mathbf{P}_1 + \mathbf{P}_0) \\ \mathbf{P}''_{n-2} & = \frac{n(n-1)}{T^2}(\mathbf{P}_n - 2\mathbf{P}_{n-1} + \mathbf{P}_{n-2}) \end{cases} \tag{A.11}$$

En remplaçant les points de contrôle de l'équation (A.9) par leur expression dans l'équation (A.11), nous obtenons les équations suivantes :

$$\begin{cases} \mathbf{P}_0 & = \mathbf{c}_s \\ \mathbf{P}_n & = \mathbf{c}_g \\ \frac{n}{T}(\mathbf{P}_1 - \mathbf{P}_0) & = \dot{\mathbf{c}}_s \\ \frac{n}{T}(\mathbf{P}_n - \mathbf{P}_{n-1}) & = \dot{\mathbf{c}}_g \\ \frac{n(n-1)}{T^2}(\mathbf{P}_2 - 2\mathbf{P}_1 + \mathbf{P}_0) & = \ddot{\mathbf{c}}_s \\ \frac{n(n-1)}{T^2}(\mathbf{P}_n - 2\mathbf{P}_{n-1} + \mathbf{P}_{n-2}) & = \ddot{\mathbf{c}}_g \end{cases} \tag{A.12}$$

Par substitution, nous pouvons obtenir :

$$\begin{cases} \frac{n}{T}(\mathbf{P}_1 - \mathbf{c}_s) & = \dot{\mathbf{c}}_s \\ \frac{n}{T}(\mathbf{c}_g - \mathbf{P}_{n-1}) & = \dot{\mathbf{c}}_g \end{cases} \tag{A.13}$$

$$\Leftrightarrow \begin{cases} \mathbf{P}_1 & = \mathbf{c}_s + \frac{\dot{\mathbf{c}}_s T}{n} \\ \mathbf{P}_{n-1} & = \mathbf{c}_g - \frac{\dot{\mathbf{c}}_g T}{n} \end{cases} \tag{A.14}$$

Ainsi que :

$$\begin{cases} \frac{n(n-1)}{T^2}(\mathbf{P}_2 - 2(\mathbf{c}_s + \frac{\dot{\mathbf{c}}_s T}{n}) + \mathbf{c}_s) & = \ddot{\mathbf{c}}_s \\ \frac{n(n-1)}{T^2}(\mathbf{c}_g - 2(\mathbf{c}_g - \frac{\dot{\mathbf{c}}_g T}{n}) + \mathbf{P}_{n-2}) & = \ddot{\mathbf{c}}_g \end{cases} \tag{A.15}$$

$$\Leftrightarrow \begin{cases} \mathbf{P}_2 & = \frac{\ddot{\mathbf{c}}_s T^2}{n(n-1)} + 2(\mathbf{c}_s + \frac{\dot{\mathbf{c}}_s T}{n}) - \mathbf{c}_s \\ \mathbf{P}_{n-2} & = \frac{\ddot{\mathbf{c}}_g T^2}{n(n-1)} + 2(\mathbf{c}_g - \frac{\dot{\mathbf{c}}_s T}{n}) - \mathbf{c}_g \end{cases} \tag{A.16}$$

$$\Leftrightarrow \begin{cases} \mathbf{P}_2 & = \frac{\ddot{\mathbf{c}}_s T^2}{n(n-1)} + 2(\mathbf{c}_s + \frac{\dot{\mathbf{c}}_s T}{n}) - \mathbf{c}_s \\ \mathbf{P}_{n-2} & = \frac{\ddot{\mathbf{c}}_g T^2}{n(n-1)} + 2(\mathbf{c}_g - \frac{\dot{\mathbf{c}}_s T}{n}) - \mathbf{c}_g \end{cases} \tag{A.17}$$

$$\Leftrightarrow \begin{cases} \mathbf{P}_2 & = \frac{\ddot{\mathbf{c}}_s T^2}{n(n-1)} + 2\frac{\dot{\mathbf{c}}_s T}{n} + \mathbf{c}_s \\ \mathbf{P}_{n-2} & = \frac{\ddot{\mathbf{c}}_g T^2}{n(n-1)} - 2\frac{\dot{\mathbf{c}}_g T}{n} + \mathbf{c}_g \end{cases} \tag{A.18}$$

Au final nous obtenons les 6 contraintes suivantes pour les 3 premiers et 3 derniers points de contrôle de $\mathbf{c}(t)$:

— $\mathbf{P}_0 = \mathbf{c}_s$ et $\mathbf{P}_n = \mathbf{c}_g$ garantissent que les positions initiale et finale sont respectées ;

— $\mathbf{P}_1 = \frac{\dot{\mathbf{c}}_s T}{n} + \mathbf{c}_s$ et $\mathbf{P}_{n-1} = \mathbf{c}_g - \frac{\dot{\mathbf{c}}_g T}{n}$ garantissent que les vitesses initiale et

finale sont respectées ;
— $\mathbf{P}_2 = \frac{\ddot{\mathbf{c}}_s T^2}{n(n-1)} + 2\frac{\dot{\mathbf{c}}_s T}{n} + \mathbf{c}_s$ et $\mathbf{P}_{n-2} = \frac{\ddot{\mathbf{c}}_g T^2}{n(n-1)} - 2\frac{\dot{\mathbf{c}}_g T}{n} + \mathbf{c}_g$ garantissent que les accélérations initiale et finale sont respectées.

A.3 Calcul des points de contrôle de $\ddot{\mathbf{c}}(t)$

Avec l'application de la propriété de dérivation d'une courbe de Bézier, $\ddot{\mathbf{c}}(t)$ est une courbe de degré $n-2$ dont les points de contrôle $\mathbf{P}''_i$ sont définies par l'équation suivante :

$$\begin{aligned}\mathbf{P}''_i &= \frac{(n-1)}{T}(\mathbf{P}'_{i+1} - \mathbf{P}'_i) \\ &= \frac{(n-1)}{T}(\frac{n}{T}(\mathbf{P}_{i+2} - \mathbf{P}_{i+1}) - \frac{n}{T}(\mathbf{P}_{i+1} - \mathbf{P}_i)) \\ &= \frac{n(n-1)}{T^2}(\mathbf{P}_{i+2} - 2\mathbf{P}_{i+1} + \mathbf{P}_i)\end{aligned} \tag{A.19}$$

A partir de maintenant et pour le reste de cette annexe nous allons fixer $n = 6$ pour tous les développements, comme dans le chapitre 4. Nous obtenons alors les expressions suivantes pour les points de contrôle de $\ddot{\mathbf{c}}(t)$:

$$\begin{cases} \mathbf{P}''_0 &= \frac{30}{T^2}(\mathbf{P}_0 - 2\mathbf{P}_1 + \mathbf{P}_2) \\ \mathbf{P}''_1 &= \frac{30}{T^2}(\mathbf{P}_1 - 2\mathbf{P}_2 + \mathbf{P}_3) \\ \mathbf{P}''_2 &= \frac{30}{T^2}(\mathbf{P}_2 + \mathbf{P}_4 - 2\mathbf{P}_3) \\ \mathbf{P}''_3 &= \frac{30}{T^2}(-2\mathbf{P}_4 + \mathbf{P}_5 + \mathbf{P}_3) \\ \mathbf{P}''_4 &= \frac{30}{T^2}(\mathbf{P}_4 - 2\mathbf{P}_5 + \mathbf{P}_6) \end{cases} \tag{A.20}$$

En substituant les expressions des points de contrôle définies par les conditions aux limites, nous obtenons :

$$\begin{cases} \mathbf{P}''_0 &= \frac{30}{T^2}(\mathbf{c}_s - 2(\frac{\dot{\mathbf{c}}_s T}{6} + \mathbf{c}_s) + \frac{\ddot{\mathbf{c}}_s T^2}{30} + 2\frac{\dot{\mathbf{c}}_s T}{6} + \mathbf{c}_s) \\ \mathbf{P}''_1 &= \frac{30}{T^2}(\frac{\dot{\mathbf{c}}_s T}{6} + \mathbf{c}_s - 2(\frac{\ddot{\mathbf{c}}_s T^2}{30} + 2\frac{\dot{\mathbf{c}}_s T}{6} + \mathbf{c}_s) + \mathbf{P}_3) \\ \mathbf{P}''_2 &= \frac{30}{T^2}(\frac{\ddot{\mathbf{c}}_s T^2}{30} + 2\frac{\dot{\mathbf{c}}_s T}{6} + \mathbf{c}_s + \frac{\ddot{\mathbf{c}}_g T^2}{30} - 2\frac{\dot{\mathbf{c}}_g T}{6} + \mathbf{c}_g - 2\mathbf{P}_3) \\ \mathbf{P}''_3 &= \frac{30}{T^2}(-2(\frac{\ddot{\mathbf{c}}_g T^2}{30} - 2\frac{\dot{\mathbf{c}}_g T}{6} + \mathbf{c}_g) + \mathbf{c}_g - \frac{\dot{\mathbf{c}}_g T}{6} + \mathbf{P}_3) \\ \mathbf{P}''_4 &= \frac{30}{T^2}(\frac{\ddot{\mathbf{c}}_g T^2}{30} - (2\frac{\dot{\mathbf{c}}_g T}{6} + \mathbf{c}_g) - 2\mathbf{c}_g - \frac{\dot{\mathbf{c}}_g T}{6} + \mathbf{c}_g) \end{cases} \tag{A.21}$$

$$\Leftrightarrow \begin{cases} \mathbf{P}''_0 &= \ddot{\mathbf{c}}_s \\ \mathbf{P}''_1 &= \frac{30}{T^2}(-\mathbf{c}_s - 3\frac{\dot{\mathbf{c}}_s T}{6} - 2\frac{\ddot{\mathbf{c}}_s T^2}{30} + \mathbf{P}_3) \\ \mathbf{P}''_2 &= \frac{30}{T^2}(\mathbf{c}_s + \mathbf{c}_g + \frac{T}{6}(2\dot{\mathbf{c}}_s - 2\dot{\mathbf{c}}_g) + \frac{T^2}{30}(\ddot{\mathbf{c}}_s + \ddot{\mathbf{c}}_g) - 2\mathbf{P}_3) \\ \mathbf{P}''_3 &= \frac{30}{T^2}(-\mathbf{c}_g + 3\frac{\dot{\mathbf{c}}_g T}{6} - 2\frac{\ddot{\mathbf{c}}_g T^2}{30} + \mathbf{P}_3) \\ \mathbf{P}''_4 &= \ddot{\mathbf{c}}_g \end{cases} \tag{A.22}$$

$$\Leftrightarrow \begin{cases} \mathbf{P}''_0 &= \ddot{\mathbf{c}}_s \\ \mathbf{P}''_1 &= -\frac{30}{T^2}(\mathbf{c}_s) - \frac{15}{T}\dot{\mathbf{c}}_s - 2\ddot{\mathbf{c}}_s + \frac{30}{T^2}\mathbf{P}_3) \\ \mathbf{P}''_2 &= \frac{30}{T^2}(\mathbf{c}_s + \mathbf{c}_g) + \frac{10}{T}(\dot{\mathbf{c}}_s - \dot{\mathbf{c}}_g) + (\ddot{\mathbf{c}}_s + \ddot{\mathbf{c}}_g) - \frac{60}{T^2}\mathbf{P}_3 \\ \mathbf{P}''_3 &= -\frac{30}{T^2}(\mathbf{c}_g) + \frac{15}{T}\dot{\mathbf{c}}_g - 2\ddot{\mathbf{c}}_g + \frac{30}{T^2}\mathbf{P}_3) \\ \mathbf{P}''_4 &= \ddot{\mathbf{c}}_g \end{cases} \tag{A.23}$$

D'après la formulation présentée dans le chapitre 4, nous avons posé $\mathbf{P}_3 = \mathbf{y}$ notre variable. Nous pouvons alors réécrire les points de contrôle de $\ddot{\mathbf{c}}(t, \mathbf{y})$ ainsi :

$$\begin{cases} \mathbf{P}''_0 &= \ddot{\mathbf{c}}_s \\ \mathbf{P}''_1 &= -\frac{30}{T^2}(\mathbf{c}_s) - \frac{15}{T}\dot{\mathbf{c}}_s - 2\ddot{\mathbf{c}}_s + \frac{30}{T^2}\mathbf{y} \\ \mathbf{P}''_2 &= \frac{30}{T^2}(\mathbf{c}_s + \mathbf{c}_g) + \frac{10}{T}(\dot{\mathbf{c}}_s - \dot{\mathbf{c}}_g) + (\ddot{\mathbf{c}}_s + \ddot{\mathbf{c}}_g) - \frac{60}{T^2}\mathbf{y} \\ \mathbf{P}''_3 &= -\frac{30}{T^2}(\mathbf{c}_g) + \frac{15}{T}\dot{\mathbf{c}}_g - 2\ddot{\mathbf{c}}_g + \frac{30}{T^2}\mathbf{y} \\ \mathbf{P}''_4 &= \ddot{\mathbf{c}}_g \end{cases} \tag{A.24}$$

On observe que chaque point de contrôle de $\ddot{\mathbf{c}}(t, \mathbf{y})$ est linéairement dépendant de la variable $\mathbf{y}$:

$$\mathbf{P}''_i(\mathbf{y}) = \mathbf{P}''^{y}_i \mathbf{y} + \mathbf{P}''^{s}_i \tag{A.25}$$

Avec :

$$\begin{cases} \mathbf{P}''^{y}_0 = 0 & \mathbf{P}''^{s}_0 = \ddot{\mathbf{c}}_s \\ \mathbf{P}''^{y}_1 = \frac{30}{T^2} & \mathbf{P}''^{s}_1 = -\frac{30}{T^2}(\mathbf{c}_s) - \frac{15}{T}\dot{\mathbf{c}}_s - 2\ddot{\mathbf{c}}_s \\ \mathbf{P}''^{y}_2 = -\frac{60}{T^2} & \mathbf{P}''^{s}_2 = \frac{30}{T^2}(\mathbf{c}_s + \mathbf{c}_g) + \frac{10}{T}(\dot{\mathbf{c}}_s - \dot{\mathbf{c}}_g) + (\ddot{\mathbf{c}}_s + \ddot{\mathbf{c}}_g) \\ \mathbf{P}''^{y}_3 = \frac{30}{T^2} & \mathbf{P}''^{s}_3 = -\frac{30}{T^2}(\mathbf{c}_g) + \frac{15}{T}\dot{\mathbf{c}}_g - 2\ddot{\mathbf{c}}_g \\ \mathbf{P}''^{y}_4 = 0 & \mathbf{P}''^{s}_4 = \ddot{\mathbf{c}}_g \end{cases} \tag{A.26}$$

A.4 Le produit vectoriel entre deux courbes de Bézier est une courbe de Bézier

Soit deux courbes de Bézier $\mathbf{a}(t)$ et $\mathbf{b}(t)$ respectivement de degré n et m, de durée $T = 1$ et de dimension identique. Nous avons :

$$\begin{aligned} \mathbf{a}(t) &= \sum_{i=0}^{n} \mathbf{P}_i B_i^n(t) \\ \mathbf{b}(t) &= \sum_{i=0}^{m} \mathbf{Q}_i B_i^m(t) \end{aligned} \tag{A.27}$$

Nous avons alors :

$$\begin{aligned}\mathbf{a}(t) \times \mathbf{b}(t) &= (\sum_{i=0}^{n} \mathbf{P}_i B_i^n(t)) \times (\sum_{i=0}^{m} \mathbf{Q}_i B_i^m(t)) \\ &= \sum_{i=0}^{n}\sum_{j=0}^{m} B_i^n(t) B_j^m(t)(\mathbf{P}_i \times \mathbf{Q}_j)\end{aligned} \tag{A.28}$$

Or, un courbe de Bézier de degré n est définie par un polynôme de même degré, nous pouvons donc écrire $\mathbf{a}(t)$ et $\mathbf{b}(t)$ sous la forme suivante :

$$\begin{aligned}\mathbf{a}(t) &= \mathbf{u}_0 + \mathbf{u}_1 t + \mathbf{u}_2 t^2 + \cdots + \mathbf{u}_n t^n \\ \mathbf{b}(t) &= \mathbf{v}_0 + \mathbf{v}_1 t + \mathbf{v}_2 t^2 + \cdots + \mathbf{v}_m t^m\end{aligned} \tag{A.29}$$

avec les $\mathbf{u}_i$ et $\mathbf{v}_i$ des coefficients de même dimension que les courbes de Bézier et $\mathbf{u}_n \neq \mathbf{0}$, $\mathbf{v}_m \neq \mathbf{0}$.

Nous pouvons ensuite calculer le produit vectoriel entre ces deux expressions et le développer par distributivité :

$$\mathbf{a}(t) \times \mathbf{b}(t) = \mathbf{u}_0 \times \mathbf{v}_0 + \mathbf{u}_0 \times \mathbf{v}_1 t + \cdots + \mathbf{u}_n \times \mathbf{v}_m t^{n+m} \tag{A.30}$$

Nous obtenons alors un polynôme de degré $n+m$. Or, tout polynôme de degré inférieur ou égal à n peut être décomposé en des polynômes de Bernstein de degré n et donc représenté par une courbe de Bézier de degré n.

Ici, $\mathbf{a}(t) \times \mathbf{b}(t)$ peut donc être représenté par une courbe de Bézier de degré $n+m$.

A.5 Expressions des points de contrôle de $\mathbf{c} \times \ddot{\mathbf{c}}$

Afin de développer les calculs de l'équation (A.30) dans le cas du produit vectoriel $\mathbf{c} \times \ddot{\mathbf{c}}$, puis de simplifier l'équation obtenue et enfin d'identifier les expressions des différents points de contrôle de la courbe résultante, nous avons utilisé un bibliothèque de calcul symbolique nommée *Sympy*.

Nous rappelons que les expressions des points de contrôle de $\ddot{\mathbf{c}}(t)$ sont données dans les équations (A.20). Avec ces expressions, nous obtenons les expressions des points de contrôle $\mathbf{Q}_i$ de la courbe $\mathbf{c} \times \ddot{\mathbf{c}}$:

$$
\begin{aligned}
\mathbf{Q}_0 &= \mathbf{P}_0 \times (-60\mathbf{P}_1 + 30\mathbf{P}_2)\frac{1}{T^2} \\
\mathbf{Q}_1 &= (-30\mathbf{P}_0 \times \mathbf{P}_2 + \frac{40}{3}\mathbf{P}_0 \times \mathbf{P}_3 + 20\mathbf{P}_1 \times \mathbf{P}_2)\frac{1}{T^2} \\
\mathbf{Q}_2 &= (5\mathbf{P}_0 \times \mathbf{P}_4 - \frac{40}{3}\mathbf{P}_0 \times \mathbf{P}_3 - 20\mathbf{P}_1 \times \mathbf{P}_2 + 20\mathbf{P}_1 \times \mathbf{P}_3)\frac{1}{T^2} \\
\mathbf{Q}_3 &= (-5\mathbf{P}_0 \times \mathbf{P}_4 + \frac{10}{7}\mathbf{P}_0 \times \mathbf{P}_5 + \frac{90}{7}\mathbf{P}_1 \times \mathbf{P}_4 - 20\mathbf{P}_1 \times \mathbf{P}_3 + \frac{100}{7}\mathbf{P}_2 \times \mathbf{P}_3)\frac{1}{T^2} \\
\mathbf{Q}_4 &= (-\frac{10}{7}\mathbf{P}_0 \times \mathbf{P}_5 + \frac{5}{21}\mathbf{P}_0 \times \mathbf{P}_6 - \frac{90}{7}\mathbf{P}_1 \times \mathbf{P}_4 + \frac{40}{7}\mathbf{P}_1 \times \mathbf{P}_5 + \frac{125}{7}\mathbf{P}_2 \times \mathbf{P}_4 \\
&\quad - \frac{100}{7} \times \mathbf{P}_3)\frac{1}{T^2} \\
\mathbf{Q}_5 &= (-\frac{5}{21}\mathbf{P}_0 \times \mathbf{P}_6 - \frac{40}{7}\mathbf{P}_1 \times \mathbf{P}_5 + \frac{10}{7}\mathbf{P}_1 \times \mathbf{P}_6 - \frac{125}{7}\mathbf{P}_2 \times \mathbf{P}_4 + \frac{90}{7}\mathbf{P}_2 \times \mathbf{P}_5 \\
&\quad - \frac{100}{7}\mathbf{P}_4 \times \mathbf{P}_3)\frac{1}{T^2} \\
\mathbf{Q}_6 &= (-\frac{10}{7}\mathbf{P}_1 \times \mathbf{P}_6 - \frac{90}{7}\mathbf{P}_2 \times \mathbf{P}_5 + 5\mathbf{P}_2 \times \mathbf{P}_6 + \frac{100}{7}\mathbf{P}_4 \times \mathbf{P}_3 - 20.0\mathbf{P}_5 \times \mathbf{P}_3)\frac{1}{T^2} \\
\mathbf{Q}_7 &= (-5.0\mathbf{P}_2 \times \mathbf{P}_6 + 20\mathbf{P}_4 \times \mathbf{P}_5 + 20\mathbf{P}_5 \times \mathbf{P}_3 - \frac{40}{3}\mathbf{P}_6 \times \mathbf{P}_3)\frac{1}{T^2} \\
\mathbf{Q}_8 &= (-20.0\mathbf{P}_4 \times \mathbf{P}_5 + 30\mathbf{P}_4 \times \mathbf{P}_6 + \frac{40}{3}\mathbf{P}_6 \times \mathbf{P}_3)\frac{1}{T^2} \\
\mathbf{Q}_9 &= \mathbf{P}_6 \times (30\mathbf{P}_4 - 60\mathbf{P}_5)\frac{1}{T^2}
\end{aligned}
\tag{A.31}
$$

Le résultat devrait être de degré $n + n - 2$, mais dans ce cas une simplification apparaît car les coefficients de plus haut degré de $\mathbf{c}(t)$ et $\ddot{\mathbf{c}}(t)$ sont linéairement dépendant, leur produit vectoriel s'annule donc et $\mathbf{c} \times \ddot{\mathbf{c}}$ est de degré $2n - 3$.

A.6 Expressions des points de contrôle du Gravito Intertial Wrench

Nous rappelons que l'expression du GIW est la suivante :

$$
\mathbf{w}(t) = \begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) \end{bmatrix} = \begin{bmatrix} m\ddot{\mathbf{c}} - m\mathbf{g} \\ m(\mathbf{c} \times \ddot{\mathbf{c}} + \mathbf{g} \times \mathbf{c}) \end{bmatrix} \tag{A.32}
$$

A.6.1 Partie inférieure

L'expression de $\mathbf{c} \times \ddot{\mathbf{c}}$ ayant déjà été calculée dans la section précédente, il faut à présent calculer $\mathbf{g} \times \mathbf{c}$. Or, le produit scalaire entre une courbe de Bézier et une constante résulte en une courbe de Bézier où chacun des points de contrôle est le produit scalaire entre cette constante et les points de contrôle de la courbe originale.

De plus, la somme de deux courbes de Bézier résulte en une courbe de Bézier de degré égale au plus au degré des deux courbes additionnées. Et la multiplication

par un scalaire consiste à multiplier tout les points de contrôle par ce scalaire.

Au final, on obtient les expressions suivantes pour les points de contrôle $\mathbf{R}_i$ de la courbe de Bézier représentant $m(\mathbf{c} \times \ddot{\mathbf{c}} + \mathbf{g} \times \mathbf{c})$.

$$
\begin{aligned}
\mathbf{R}_0 &= (\mathbf{g} \times \mathbf{P}_0 T^2 - 60\mathbf{P}_0 \times \mathbf{P}_1 + 30\mathbf{P}_0 \times \mathbf{P}_2)\frac{m}{T^2} \\
\mathbf{R}_1 &= (\frac{1}{3}\mathbf{g} \times \mathbf{P}_0 T^2 + \frac{2}{3}\mathbf{g} \times \mathbf{P}_1 T^2 - 30\mathbf{P}_0 \times \mathbf{P}_2 + \frac{40}{3}\mathbf{P}_0 \times \mathbf{P}_3 + 20\mathbf{P}_1 \times \mathbf{P}_2)\frac{m}{T^2} \\
\mathbf{R}_2 &= (\frac{1}{12}\mathbf{g} \times \mathbf{P}_0 T^2 + \frac{1}{2}\mathbf{g} \times \mathbf{P}_1 T^2 + \frac{5}{12}\mathbf{g} \times \mathbf{P}_2 T^2 + 5\mathbf{P}_0 \times \mathbf{P}_4 - \frac{40}{3}\mathbf{P}_0 \times \mathbf{P}_3 \\
&\quad - 20\mathbf{P}_1 \times \mathbf{P}_2 + 20\mathbf{P}_1 \times \mathbf{P}_3)\frac{m}{T^2} \\
\mathbf{R}_3 &= (\frac{1}{84}\mathbf{g} \times \mathbf{P}_0 T^2 + \frac{3}{14}\mathbf{g} \times \mathbf{P}_1 T^2 + \frac{15}{28}\mathbf{g} \times \mathbf{P}_2 T^2 + \frac{5}{21}\mathbf{g} \times \mathbf{P}_3 T^2 - 5\mathbf{P}_0 \times \mathbf{P}_4 \\
&\quad + \frac{10}{7}\mathbf{P}_0 \times \mathbf{P}_5 + \frac{90}{7}\mathbf{P}_1 \times \mathbf{P}_4 - 20\mathbf{P}_1 \times \mathbf{P}_3 + \frac{100}{7}\mathbf{P}_2 \times \mathbf{P}_3)\frac{m}{T^2} \\
\mathbf{R}_4 &= (\frac{1}{21}\mathbf{g} \times \mathbf{P}_1 T^2 + \frac{5}{15}\mathbf{g} \times \mathbf{P}_2 T^2 + \frac{5}{42}\mathbf{g} \times \mathbf{P}_4 T^2 + \frac{10}{21}\mathbf{g} \times \mathbf{P}_3 T^2 - \frac{10}{7}\mathbf{P}_0 \times \mathbf{P}_5 \\
&\quad + \frac{5}{21}\mathbf{P}_0 \times \mathbf{P}_6 - \frac{90}{7}\mathbf{P}_1 \times \mathbf{P}_4 + \frac{40}{7}\mathbf{P}_1 \times \mathbf{P}_5 + \frac{125}{7}\mathbf{P}_2 \times \mathbf{P}_4 - \frac{100}{7}\mathbf{P}_2 \times \mathbf{P}_3)\frac{m}{T^2} \\
\mathbf{R}_5 &= (\frac{5}{42}\mathbf{g} \times \mathbf{P}_2 T^2 + \frac{5}{14}\mathbf{g} \times \mathbf{P}_4 T^2 + \frac{1}{21}6\mathbf{g} \times \mathbf{P}_5 T^2 + \frac{10}{21}\mathbf{g} \times \mathbf{P}_3 T^2 + -\frac{5}{21}\mathbf{P}_0 \times \mathbf{P}_6 \\
&\quad - \frac{40}{7}\mathbf{P}_1 \times \mathbf{P}_5 + \frac{10}{7}\mathbf{P}_1 \times \mathbf{P}_6 - \frac{125}{7}\mathbf{P}_2 \times \mathbf{P}_4 + \frac{90}{7}\mathbf{P}_2 \times \mathbf{P}_5 - \frac{100}{7}\mathbf{P}_4 \times \mathbf{P}_3)\frac{m}{T^2} \\
\mathbf{R}_6 &= (\frac{15}{28}\mathbf{g} \times *T^2\mathbf{P}_4 + \frac{3}{14}\mathbf{g} \times \mathbf{P}_5 T^2 + \frac{1}{84}\mathbf{g} \times \mathbf{P}_6 T^2 + \frac{5}{21}\mathbf{g} \times \mathbf{P}_3 T^2 - \frac{10}{7}\mathbf{P}_1 \times \mathbf{P}_6 \\
&\quad - \frac{90}{7}\mathbf{P}_2 \times \mathbf{P}_5 + 5\mathbf{P}_2 \times \mathbf{P}_6 + \frac{100}{7}\mathbf{P}_4 \times \mathbf{P}_3 - 20\mathbf{P}_5 \times \mathbf{P}_3)\frac{m}{T^2} \\
\mathbf{R}_7 &= (\frac{5}{12}\mathbf{g} \times \mathbf{P}_4 T^2 + \frac{1}{2}\mathbf{g} \times \mathbf{P}_5 T^2 + \frac{1}{12}\mathbf{g} \times \mathbf{P}_6 T^2 - 5\mathbf{P}_2 \times \mathbf{P}_6 + 20\mathbf{P}_4 \times \mathbf{P}_5 \\
&\quad + 20\mathbf{P}_5 \times \mathbf{P}_3 - \frac{40}{3}\mathbf{P}_6 \times \mathbf{P}_3)\frac{m}{T^2} \\
\mathbf{R}_8 &= (\frac{2}{3}\mathbf{g} \times \mathbf{P}_5 T^2 + \frac{1}{3}\mathbf{g} \times \mathbf{P}_6 T^2 - 20\mathbf{P}_4 \times \mathbf{P}_5 + 30\mathbf{P}_4 \times \mathbf{P}_6 + \frac{40}{3}\mathbf{P}_6 \times \mathbf{P}_3)\frac{m}{T^2} \\
\mathbf{R}_9 &= (\mathbf{g} \times \mathbf{P}_6 T^2 - 30\mathbf{P}_4 \times \mathbf{P}_6 + 60\mathbf{P}_5 \times \mathbf{P}_6)\frac{m}{T^2}
\end{aligned}
\tag{A.33}
$$

D'après la formulation présentée dans le chapitre 4, nous avons posé $\mathbf{P}_3 = \mathbf{y}$ notre variable. Les expressions des points de contrôle $\mathbf{R}_i$ présentées ci-dessus sont donc linéairement dépendantes de notre variable $\mathbf{y}$:

$$\mathbf{R}_i(\mathbf{y}) = \mathbf{R}_i^y \mathbf{y} + \mathbf{R}_i^s \tag{A.34}$$

Avec :

$$
\begin{aligned}
\mathbf{R}_0^y &= 0 \\
\mathbf{R}_0^s &= (\mathbf{g}\times\mathbf{P}_0T^2 - 60\mathbf{P}_0\times\mathbf{P}_1 + 30\mathbf{P}_0\times\mathbf{P}_2)\frac{m}{T^2} \\
\mathbf{R}_1^y &= (\frac{40}{3}\hat{\mathbf{P}}_0)\frac{m}{T^2} \\
\mathbf{R}_1^s &= (\frac{1}{3}\mathbf{g}\times\mathbf{P}_0T^2 + \frac{2}{3}\mathbf{g}\times\mathbf{P}_1T^2 - 30\mathbf{P}_0\times\mathbf{P}_2 + 20\mathbf{P}_1\times\mathbf{P}_2)\frac{m}{T^2} \\
\mathbf{R}_2^y &= (-\frac{40}{3}\hat{\mathbf{P}}_0 + 20\hat{\mathbf{P}}_1)\frac{m}{T^2} \\
\mathbf{R}_2^s &= (\frac{1}{12}\mathbf{g}\times\mathbf{P}_0T^2 + \frac{1}{2}\mathbf{g}\times\mathbf{P}_1T^2 + \frac{5}{12}\mathbf{g}\times\mathbf{P}_2T^2 + 5\mathbf{P}_0\times\mathbf{P}_4 \\
&\quad - 20\mathbf{P}_1\times\mathbf{P}_2)\frac{m}{T^2} \\
\mathbf{R}_3^y &= (+\frac{5}{21}\hat{\mathbf{g}}T^2 - 20\hat{\mathbf{P}}_1 + \frac{100}{7}\hat{\mathbf{P}}_2)\frac{m}{T^2} \\
\mathbf{R}_3^s &= (\frac{1}{84}\mathbf{g}\times\mathbf{P}_0T^2 + \frac{3}{14}\mathbf{g}\times\mathbf{P}_1T^2 + \frac{15}{28}\mathbf{g}\times\mathbf{P}_2T^2 - 5\mathbf{P}_0\times\mathbf{P}_4 \\
&\quad + \frac{10}{7}\mathbf{P}_0\times\mathbf{P}_5 + \frac{90}{7}\mathbf{P}_1\times\mathbf{P}_4)\frac{m}{T^2} \\
\mathbf{R}_4^y &= (\frac{10}{21}\hat{\mathbf{g}}T^2 - \frac{100}{7}\hat{\mathbf{P}}_2)\frac{m}{T^2} \\
\mathbf{R}_4^s &= (\frac{1}{21}\mathbf{g}\times\mathbf{P}_1T^2 + \frac{5}{15}\mathbf{g}\times\mathbf{P}_2T^2 + \frac{5}{42}\mathbf{g}\times\mathbf{P}_4T^2 - \frac{10}{7}\mathbf{P}_0\times\mathbf{P}_5 \\
&\quad + \frac{5}{21}\mathbf{P}_0\times\mathbf{P}_6 - \frac{90}{7}\mathbf{P}_1\times\mathbf{P}_4 + \frac{40}{7}\mathbf{P}_1\times\mathbf{P}_5 + \frac{125}{7}\mathbf{P}_2\times\mathbf{P}_4)\frac{m}{T^2} \\
\mathbf{R}_5^y &= (\frac{10}{21}\hat{\mathbf{g}}T^2 - \frac{100}{7}\hat{\mathbf{P}}_4)\frac{m}{T^2} \\
\mathbf{R}_5^s &= (\frac{5}{42}\mathbf{g}\times\mathbf{P}_2T^2 + \frac{5}{14}\mathbf{g}\times\mathbf{P}_4T^2 + \frac{1}{21}6\mathbf{g}\times\mathbf{P}_5T^2 - \frac{5}{21}\mathbf{P}_0\times\mathbf{P}_6 \\
&\quad - \frac{40}{7}\mathbf{P}_1\times\mathbf{P}_5 + \frac{10}{7}\mathbf{P}_1\times\mathbf{P}_6 - \frac{125}{7}\mathbf{P}_2\times\mathbf{P}_4 + \frac{90}{7}\mathbf{P}_2\times\mathbf{P}_5)\frac{m}{T^2} \\
\mathbf{R}_6^y &= (\frac{5}{21}\hat{\mathbf{g}}T^2 + \frac{100}{7}\hat{\mathbf{P}}_4 - 20\hat{\mathbf{P}}_5)\frac{m}{T^2} \\
\mathbf{R}_6^s &= (\frac{15}{28}\mathbf{g}\times *T^2\mathbf{P}_4 + \frac{3}{14}\mathbf{g}\times\mathbf{P}_5T^2 + \frac{1}{84}\mathbf{g}\times\mathbf{P}_6T^2 - \frac{10}{7}\mathbf{P}_1\times\mathbf{P}_6 \\
&\quad - \frac{90}{7}\mathbf{P}_2\times\mathbf{P}_5 + 5\mathbf{P}_2\times\mathbf{P}_6)\frac{m}{T^2} \\
\mathbf{R}_7^y &= (20\hat{\mathbf{P}}_5 - \frac{40}{3}\hat{\mathbf{P}}_6)\frac{m}{T^2} \\
\mathbf{R}_7^s &= (\frac{5}{12}\mathbf{g}\times\mathbf{P}_4T^2 + \frac{1}{2}\mathbf{g}\times\mathbf{P}_5T^2 + \frac{1}{12}\mathbf{g}\times\mathbf{P}_6T^2 - 5\mathbf{P}_2\times\mathbf{P}_6 + 20\mathbf{P}_4\times\mathbf{P}_5)\frac{m}{T^2} \\
\mathbf{R}_8^y &= (\frac{40}{3}\hat{\mathbf{P}}_6)\frac{m}{T^2} \\
\mathbf{R}_8^s &= (\frac{2}{3}\mathbf{g}\times\mathbf{P}_5T^2 + \frac{1}{3}\mathbf{g}\times\mathbf{P}_6T^2 - 20\mathbf{P}_4\times\mathbf{P}_5 + 30\mathbf{P}_4\times\mathbf{P}_6)\frac{m}{T^2} \\
\mathbf{R}_9^y &= 0 \\
\mathbf{R}_9^s &= (\mathbf{g}\times\mathbf{P}_6T^2 - 30\mathbf{P}_4\times\mathbf{P}_6 + 60\mathbf{P}_5\times\mathbf{P}_6)\frac{m}{T^2}
\end{aligned}
\tag{A.35}
$$

A.6.2 Partie supérieure

On s'intéresse ici à calculer la courbe de Bézier représentant $m(\ddot{\mathbf{c}} - \mathbf{g})$. D'après les expressions des points de contrôle de $\ddot{\mathbf{c}}(t)$ de l'équation (A.20) les points de contrôle $\mathbf{S}_i$ de $m(\ddot{\mathbf{c}} - \mathbf{g})$ sont :

$$\begin{aligned}
\mathbf{S}_0 &= (T^2\mathbf{g} + 30\mathbf{P}_0 - 60\mathbf{P}_1 + 30\mathbf{P}_2)\frac{m}{T^2} \\
\mathbf{S}_1 &= (T^2\mathbf{g} + 30\mathbf{P}_1 - 60\mathbf{P}_2 + 30\mathbf{P}_3)\frac{m}{T^2} \\
\mathbf{S}_2 &= (T^2\mathbf{g} + 30\mathbf{P}_2 + 30\mathbf{P}_4 - 60\mathbf{P}_3)\frac{m}{T^2} \\
\mathbf{S}_3 &= (T^2\mathbf{g} - 60\mathbf{P}_4 + 30\mathbf{P}_5 + 30\mathbf{P}_3)\frac{m}{T^2} \\
\mathbf{S}_4 &= (T^2\mathbf{g} + 30\mathbf{P}_4 - 60\mathbf{P}_5 + 30\mathbf{P}_6)\frac{m}{T^2}
\end{aligned} \tag{A.36}$$

Cette courbe est de degré 4, or la courbe de la partie inférieure de $\mathbf{w}(t)$ est de degré 9, afin de pouvoir empiler correctement ces deux parties dans une même courbe, il faut élever le degré de la partie supérieure.

Pour élever une courbe de Bézier d'un degré, la règle est la suivante :

$$\sum_{i=0}^{n} B_i^n(t)\mathbf{p}_i = \sum_{i=0}^{k} B_i^k(t)\frac{(k-i)\mathbf{p}_i + i\mathbf{p}_{i-1}}{k}, k = n+1, \mathbf{p}_{-1} = 0$$

Ici, il faut élever jusqu'au degré 9, nous obtenons alors les nouvelles expressions pour les points de contrôle $\mathbf{U}_i$ de la courbe $m(\ddot{\mathbf{c}} - \mathbf{g})$ élevé au degré 9 :

$$
\begin{aligned}
\mathbf{U}_0 &= (T^2\mathbf{g} + 30\mathbf{P}_0 - 60\mathbf{P}_1 + 30\mathbf{P}_2)\frac{m}{T^2} \\
\mathbf{U}_1 &= (T^2\mathbf{g} + \frac{50}{3}\mathbf{P}_0 - 20\mathbf{P}_1 - 10\mathbf{P}_2 + \frac{40}{3}\mathbf{P}_3)\frac{m}{T^2} \\
\mathbf{U}_2 &= (T^2\mathbf{g} + \frac{25}{3}\mathbf{P}_0 - 20\mathbf{P}_2 + 5\mathbf{P}_4 + \frac{2}{3}\mathbf{P}_3)\frac{m}{T^2} \\
\mathbf{U}_3 &= (T^2\mathbf{g} + \frac{25}{7}\mathbf{P}_0 + \frac{50}{7}\mathbf{P}_1 - \frac{100}{7}\mathbf{P}_2 + \frac{55}{7}\mathbf{P}_4 + \frac{10}{7}\mathbf{P}_5 - \frac{40}{7}\mathbf{P}_3)\frac{m}{T^2} \\
\mathbf{U}_4 &= (T^2\mathbf{g} + \frac{25}{21}\mathbf{P}_0 + \frac{50}{7}\mathbf{P}_1 - \frac{25}{7}\mathbf{P}_2 + 5.0\mathbf{P}_4 + \frac{30}{7}\mathbf{P}_5 + \frac{5}{21}\mathbf{P}_6 - \frac{100}{7}\mathbf{P}_3)\frac{m}{T^2} \\
\mathbf{U}_5 &= (T^2\mathbf{g} + \frac{5}{21}\mathbf{P}_0 + \frac{30}{7}\mathbf{P}_1 + 5.0\mathbf{P}_2 - \frac{25}{7}\mathbf{P}_4 + \frac{50}{7}\mathbf{P}_5 + \frac{25}{21}\mathbf{P}_6 - \frac{100}{7}\mathbf{P}_3)\frac{m}{T^2} \\
\mathbf{U}_6 &= (T^2\mathbf{g} + \frac{10}{7}\mathbf{P}_1 + \frac{55}{7}\mathbf{P}_2 - \frac{100}{7}\mathbf{P}_4 + \frac{50}{7}\mathbf{P}_5 + \frac{25}{7}\mathbf{P}_6 - \frac{40}{7}\mathbf{P}_3)\frac{m}{T^2} \\
\mathbf{U}_7 &= (T^2\mathbf{g} + 5\mathbf{P}_2 - 20\mathbf{P}_4 + \frac{25}{3}\mathbf{P}_6 + \frac{2}{3}\mathbf{P}_3)\frac{m}{T^2} \\
\mathbf{U}_8 &= (T^2\mathbf{g} - 10\mathbf{P}_4 - 20\mathbf{P}_5 + \frac{50}{3}\mathbf{P}_6 + \frac{40}{3}\mathbf{P}_3)\frac{m}{T^2} \\
\mathbf{U}_9 &= (T^2\mathbf{g} + 30\mathbf{P}_4 - 60\mathbf{P}_5 + 30\mathbf{P}_6)\frac{m}{T^2}
\end{aligned}
\tag{A.37}
$$

Comme précédemment, les expressions des points de contrôle $\mathbf{U}_i$ présentées ci-dessus sont linéairement dépendantes de notre variable $\mathbf{y}$:

$$\mathbf{U}_i(\mathbf{y}) = \mathbf{U}_i^y\mathbf{y} + \mathbf{U}_i^s \tag{A.38}$$

Avec :

$$
\begin{aligned}
\mathbf{U}_0^y &= 0 \\
\mathbf{U}_0^s &= (T^2\mathbf{g} + 30\mathbf{P}_0 - 60\mathbf{P}_1 + 30\mathbf{P}_2)\frac{m}{T^2} \\
\mathbf{U}_1^y &= \frac{40}{3}\frac{m}{T^2} \\
\mathbf{U}_1^s &= (T^2\mathbf{g} + \frac{50}{3}\mathbf{P}_0 - 20\mathbf{P}_1 - 10\mathbf{P}_2)\frac{m}{T^2} \\
\mathbf{U}_2^y &= \frac{2}{3}\frac{m}{T^2} \\
\mathbf{U}_2^s &= (T^2\mathbf{g} + \frac{25}{3}\mathbf{P}_0 - 20\mathbf{P}_2 + 5\mathbf{P}_4)\frac{m}{T^2} \\
\mathbf{U}_3^y &= -\frac{40}{7}\frac{m}{T^2} \\
\mathbf{U}_3^s &= (T^2\mathbf{g} + \frac{25}{7}\mathbf{P}_0 + \frac{50}{7}\mathbf{P}_1 - \frac{100}{7}\mathbf{P}_2 + \frac{55}{7}\mathbf{P}_4 + \frac{10}{7}\mathbf{P}_5)\frac{m}{T^2} \\
\mathbf{U}_4^y &= -\frac{100}{7}\frac{m}{T^2} \\
\mathbf{U}_4^s &= (T^2\mathbf{g} + \frac{25}{21}\mathbf{P}_0 + \frac{50}{7}\mathbf{P}_1 - \frac{25}{7}\mathbf{P}_2 + 5.0\mathbf{P}_4 + \frac{30}{7}\mathbf{P}_5 + \frac{5}{21}\mathbf{P}_6)\frac{m}{T^2} \\
\mathbf{U}_5^y &= -\frac{100}{7}\frac{m}{T^2} \\
\mathbf{U}_5^s &= (T^2\mathbf{g} + \frac{5}{21}\mathbf{P}_0 + \frac{30}{7}\mathbf{P}_1 + 5.0\mathbf{P}_2 - \frac{25}{7}\mathbf{P}_4 + \frac{50}{7}\mathbf{P}_5 + \frac{25}{21}\mathbf{P}_6)\frac{m}{T^2} \\
\mathbf{U}_6^y &= -\frac{40}{7}\frac{m}{T^2} \\
\mathbf{U}_6^s &= (T^2\mathbf{g} + \frac{10}{7}\mathbf{P}_1 + \frac{55}{7}\mathbf{P}_2 - \frac{100}{7}\mathbf{P}_4 + \frac{50}{7}\mathbf{P}_5 + \frac{25}{7}\mathbf{P}_6)\frac{m}{T^2} \\
\mathbf{U}_7^y &= \frac{2}{3}\frac{m}{T^2} \\
\mathbf{U}_7^s &= (T^2\mathbf{g} + 5\mathbf{P}_2 - 20\mathbf{P}_4 + \frac{25}{3}\mathbf{P}_6)\frac{m}{T^2} \\
\mathbf{U}_8^y &= \frac{40}{3}\frac{m}{T^2} \\
\mathbf{U}_8^s &= (T^2\mathbf{g} - 10\mathbf{P}_4 - 20\mathbf{P}_5 + \frac{50}{3}\mathbf{P}_6)\frac{m}{T^2} \\
\mathbf{U}_9^y &= 0 \\
\mathbf{U}_9^s &= (T^2\mathbf{g} + 30\mathbf{P}_4 - 60\mathbf{P}_5 + 30\mathbf{P}_6)\frac{m}{T^2}
\end{aligned}
\tag{A.39}
$$

A.6.3 Expression du GIW

Au final, les points de contrôle $\mathbf{P}_{\mathbf{w}i}$ de la courbe de Bézier représentant $\mathbf{w}(t)$, de degré 9 s'obtiennent en empilant les points de controles de la partie supérieure (A.39) et inférieure (A.35) :

$$\mathbf{P}_{\mathbf{w}i}(\mathbf{y}) = \begin{bmatrix} \mathbf{U}_i^y \\ \mathbf{R}_i^y \end{bmatrix} \mathbf{y} + \begin{bmatrix} \mathbf{U}_i^s \\ \mathbf{R}_i^s \end{bmatrix} \ , \forall i \in [0;9] \tag{A.40}$$

Les expressions de ces points de contrôle sont donc bien linéairement dépendantes de $\mathbf{y}$.

A.7 Calcul analytique d'un coût intégral

Beaucoup de fonctions de coût considérées lors de l'optimisation de trajectoires contiennent des intégrales. Dans cette section nous montrons comment représenter ces fonctions de coût sous une forme quadratique classique $\mathbf{y}^\top \mathbf{A}\mathbf{y} + 2\mathbf{b}^\top \mathbf{y}$ en prenant comme exemple le coût visant à minimiser l'intégrale de la norme au carré de l'accélération $\ddot{\mathbf{c}}(t, \mathbf{y})$:

$$\int_0^T ||\ddot{\mathbf{c}}(t, \mathbf{y})||^2 dt \tag{A.41}$$

Tout d'abord, il faut calculer l'expression de $||\ddot{\mathbf{c}}(t, \mathbf{y})||$:

$$||\ddot{\mathbf{c}}(t, \mathbf{y})|| = \sqrt{\sum_{d=\{x,y,z\}} \ddot{\mathbf{c}}_d^2(t, \mathbf{y})} \tag{A.42}$$

avec :

$$\ddot{\mathbf{c}}_d(t, \mathbf{y}) = \sum_{i=0}^{4} B_i^4(t/T)\mathbf{Q}_{i,d} \tag{A.43}$$

avec $\mathbf{Q}_i \in \mathbb{R}^3$ les points de contrôle de $\ddot{\mathbf{c}}(t, \mathbf{y})$ et $\mathbf{Q}_{i,d} \in \mathbb{R}$ la valeur des points de contrôle selon l'axe d. $\ddot{\mathbf{c}}_d(t, \mathbf{y})$ est donc une courbe de dimension 1.

Ensuite, nous pouvons calculer :

$$\int_0^T ||\ddot{\mathbf{c}}(t, \mathbf{y})||^2 dt = \int_0^T \sum_{d=\{x,y,z\}} \ddot{\mathbf{c}}_d^2(t, \mathbf{y}) dt = \sum_{d=\{x,y,z\}} (\int_0^T \ddot{\mathbf{c}}_d^2(t, \mathbf{y}) dt) \tag{A.44}$$

L'expression des points de contrôle de $\ddot{\mathbf{c}}_d^2(t, \mathbf{y})$ s'obtient analytiquement en mettant au carré la courbe $\ddot{\mathbf{c}}_d(t, \mathbf{y})$ et en identifiant l'expression des nouveaux points de contrôles.

Il faut ensuite calculer l'intégrale de $\ddot{\mathbf{c}}_d^2(t, \mathbf{y})$. Pour cela, il faut tout d'abord calculer l'expression de sa primitive.

A.7.1 Primitive d'une courbe de Bézier

La primitive d'une courbe de Bézier de degré n définie par des points de contrôle $\mathbf{Q}_i$ et de durée T est une courbe de Bézier de durée égale, de degré $n+1$ et définie par les points de contrôle $\mathbf{R}_i$ obtenus via la relation suivante :

$$\mathbf{R}_i = \sum_{j=0}^{i-1} T\mathbf{Q}_j/(n+1) \ , \forall i \in [1; n+1] \tag{A.45}$$

Et $\mathbf{R}_0$ le terme constant déterminé via les conditions initiales, dans la suite nous supposons que $\mathbf{R}_0 = 0$.

L'expression de cette primitive peut donc se calculer analytiquement.

A.7.2 Évaluation de la primitive aux extrémités

Maintenant que nous avons l'expression analytique de la primitive de $\ddot{\mathbf{c}}_d^2(t, \mathbf{y})$ notée $\ddot{\mathbf{C}}_{2d}(t, \mathbf{y})$, il suffit de l'évaluer pour $t = 0$ et $t = T$. Nous avons alors $\ddot{\mathbf{C}}_{2d}(0, \mathbf{y}) = 0$ et :

$$\begin{aligned}
\ddot{\mathbf{C}}_{2d}(T, \mathbf{y}) = 1440\frac{\mathbf{y}_d^2}{7T^3} \\
+ (1800\mathbf{P}_{0,d} - 4320\mathbf{P}_{1,d} + 1080\mathbf{P}_{2,d} + 1080\mathbf{P}_{4,d} - 4320\mathbf{P}_{5,d} + 1800\mathbf{P}_{6,d})\frac{\mathbf{y}_d}{7T^3} \\
+ (6300\mathbf{P}_{0,d}^2 - 18900\mathbf{P}_{0,d}\mathbf{P}_{1,d} + 2700\mathbf{P}_{0,d}\mathbf{P}_{2,d} + 1080\mathbf{P}_{0,d}\mathbf{P}_{4,d} + 540\mathbf{P}_{0,d}\mathbf{P}_{5,d} \\
+180\mathbf{P}_{0,d}\mathbf{P}_{6,d} + 16200\mathbf{P}_{1,d}^2 - 8100\mathbf{P}_{1,d}\mathbf{P}_{2,d} - 1620\mathbf{P}_{1,d}\mathbf{P}_{4,d} + 540\mathbf{P}_{1,d}\mathbf{P}_{6,d} \\
+3240\mathbf{P}_{2,d}^2 - 1620\mathbf{P}_{2,d}\mathbf{P}_{4,d} - 1620\mathbf{P}_{2,d}\mathbf{P}_{5,d} + 1080\mathbf{P}_{2,d}\mathbf{P}_{6,d} + 3240\mathbf{P}_{4,d}^2 \\
-8100\mathbf{P}_{4,d}\mathbf{P}_{5,d} + 2700\mathbf{P}_{4,d}\mathbf{P}_{6,d} + 16200\mathbf{P}_{5,d}^2 - 18900\mathbf{P}_{5,d}\mathbf{P}_{6,d} + 6300\mathbf{P}_{6,d}^2)\frac{1}{7T^3}
\end{aligned} \tag{A.46}$$

Avec les $\mathbf{P}_i \in \mathbb{R}^3$ les points de contrôle de $\mathbf{c}(t, \mathbf{y})$ définis dans la section A.2 et $\mathbf{P}_{i,d} \in \mathbb{R}$ leur valeur selon l'axe d. $\mathbf{y}_d \in \mathbb{R}$ est la valeur de la variable $\mathbf{y}$ selon l'axe d.

A.7.3 Mise sous forme quadratique

Nous pouvons maintenant exprimer le coût ainsi :

$$\int_0^T ||\ddot{\mathbf{c}}(t, \mathbf{y})||^2 dt = \sum_{d=\{x,y,z\}} (\ddot{\mathbf{C}}_{2d}(T, \mathbf{y}) - \ddot{\mathbf{C}}_{2d}(0, \mathbf{y})) = \sum_{d=\{x,y,z\}} \ddot{\mathbf{C}}_{2d}(T, \mathbf{y}) \tag{A.47}$$

Nous pouvons développer cette expression, en négligeant la partie constante (indépendante de $\mathbf{y}$) :

$$\begin{aligned}
\int_0^T ||\ddot{\mathbf{c}}(t, \mathbf{y})||^2 dt = \frac{1440}{7T^3} \sum_{d=\{x,y,z\}} \mathbf{y}_d^2 \\
+ \sum_{d=\{x,y,z\}} ((1800\mathbf{P}_{0,d} - 4320\mathbf{P}_{1,d} + 1080\mathbf{P}_{2,d} + 1080\mathbf{P}_{4,d} - 4320\mathbf{P}_{5,d} + 1800\mathbf{P}_{6,d})\frac{\mathbf{y}_d}{7T^3})
\end{aligned} \tag{A.48}$$

Au final, en factorisant cette expression par la variable $\mathbf{y}$ puis en simplifiant, nous pouvons reformuler le coût ainsi :

$$\int_0^T ||\ddot{\mathbf{c}}(t,\mathbf{y})||^2 dt = \mathbf{y}^\top \mathbf{A}\mathbf{y} + 2\mathbf{b}^\top \mathbf{y} \tag{A.49}$$

Avec :

$$\begin{aligned} \mathbf{A} &= \mathbb{I}_3 \frac{1440}{7T^3} \\ 2\mathbf{b} &= (1800\mathbf{P}_0 - 4320\mathbf{P}_1 + 1080\mathbf{P}_2 + 1080\mathbf{P}_4 - 4320\mathbf{P}_5 + 1800\mathbf{P}_6)\frac{1}{7T^3} \end{aligned} \tag{A.50}$$

Avec $\mathbb{I}_3$ la matrice Identité de taille 3×3.

A.8 Algorithme de De Casteljau

L'algorithme de De Casteljau est une méthode récursive permettant d'évaluer une courbe de Bézier $\mathbf{c}(u)$ de degré n en un point u_0. Dans cette section, par simplicité, nous prenons le cas d'une courbe de Bézier définie pour $u \in [0;1]$, dans le cas général d'une courbe de durée T il suffit d'utiliser la relation $u = t/T$ pour évaluer la courbe en un point t_0.

Afin de calculer $\mathbf{c}(u_0) = \mathbf{P}_0^n$, cet algorithme utilise la relation de récurrence suivante :

$$\mathbf{P}_i^j = (1-u_0)\mathbf{P}_i^{j-1} + u_0\mathbf{P}_{i+1}^{j-1} \tag{A.51}$$

avec $\mathbf{P}_i^0$ les points de contrôle de $\mathbf{c}$.

Cet algorithme s'explique géométriquement avec la figure A.1. Chaque figure montre une itération de l'algorithme : à chaque itération j les segments de droite connectant les points de l'itération précédente $\mathbf{P}_i^{j-1}$ sont divisés selon un ratio $u_0 : (1-u_0)$ et des nouveaux points $\mathbf{P}_i^j$ sont créés puis reliés entre eux.

A la dernière itération $j = n$ il ne reste qu'un point $\mathbf{P}_0^n$, il s'agit de l'évaluation de la courbe au paramètre u_0 donné : $\mathbf{c}(u_0)$.

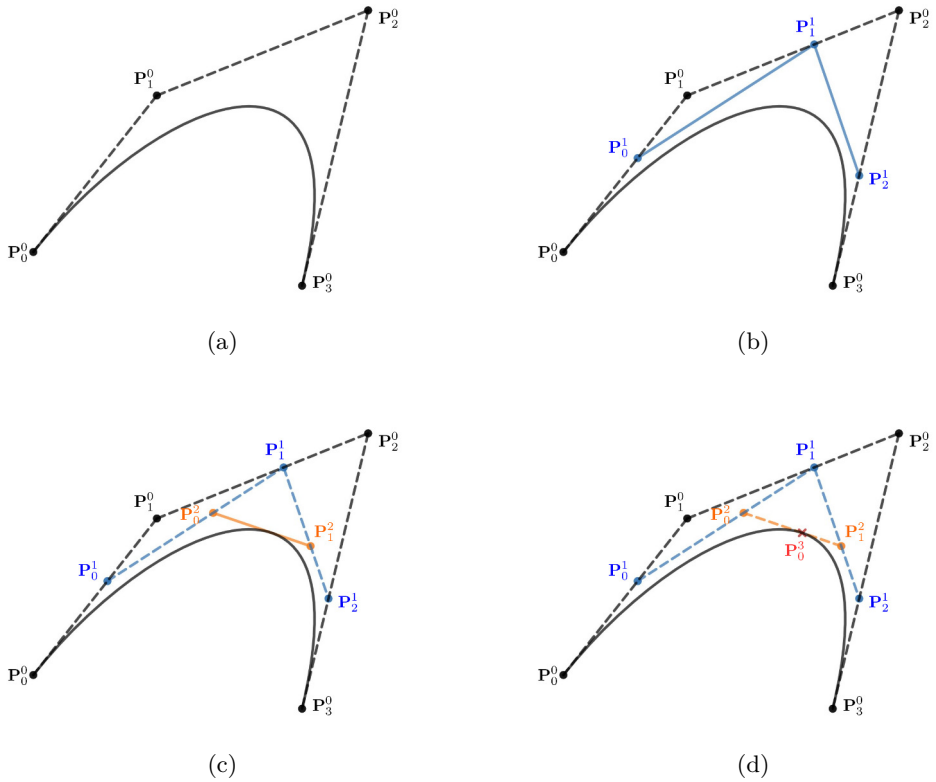


FIGURE A.1 – Itérations successives de l'algorithme de De Casteljau pour une courbe de Bézier de degré $n = 3$ définie par les points de contrôle $\mathbf{P}_i^0$, évaluée à $u_0 = 0.6$.

Le second intérêt de l'algorithme de De Casteljau est de permettre de scinder une courbe de Bézier $\mathbf{c}(u)$ en deux courbes $\mathbf{c}_1(u)$ et $\mathbf{c}_2(u)$ au point u_0. Cette décomposition est strictement équivalente à la courbe originale :

$$\begin{cases} \mathbf{c}_1(u) = \mathbf{c}(u) & \forall u \in [0; u_0] \\ \mathbf{c}_2(u) = \mathbf{c}(u) & \forall u \in [u_0; 1] \\ \mathbf{c}_1(u_0) = \mathbf{c}_2(u_0) \end{cases} \tag{A.52}$$

Les points de contrôle de ces deux courbes $\mathbf{c}_1(u)$ et $\mathbf{c}_2(u)$ sont calculés par l'algorithme de De Casteljau, il s'agit des points $\mathbf{P}_0^0, \mathbf{P}_0^1, ..., \mathbf{P}_0^n$ pour la courbe $\mathbf{c}_1(u)$ et des points $\mathbf{P}_0^n, \mathbf{P}_1^{n-1}, ..., \mathbf{P}_n^0$ pour la courbe $\mathbf{c}_2(u)$. Cette décomposition est illustrée sur la figure A.2.

D'après l'équation (A.51), il est possible de calculer par récurrence les points de contrôle de ces courbes scindées et ils sont linéairement dépendants des points de contrôle de la courbe originale. En effet, il n'y a aucune multiplication entre les points de contrôle, uniquement des multiplications par des scalaires et des additions entre points de contrôle.

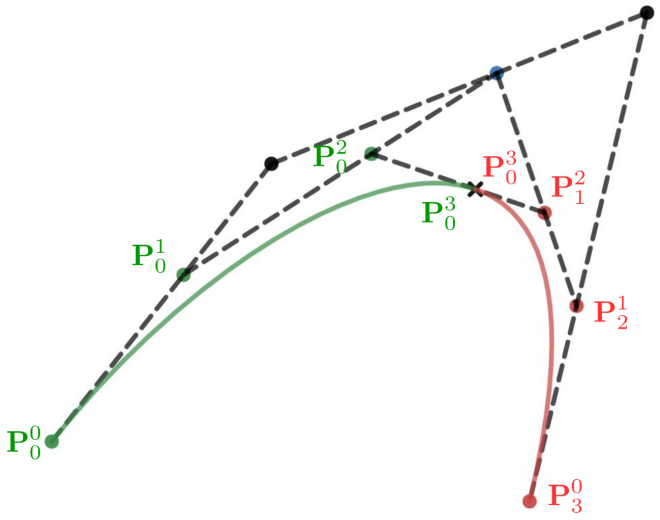


FIGURE A.2 – Exemple de décomposition d'une courbe de Bézier de degré $n = 3$ à $u_0 = 0.6$. En vert la courbe $\mathbf{c}_1(u)$ et ses points de contrôle, en rouge la courbe $\mathbf{c}_2(u)$ et ses points de contrôle.

Annexe B

Détail des méthodes de planification de contact

Cette annexe décrit les méthodes utilisées par le générateur de contact pour assurer les règles exprimées dans la section 6.3. Ces méthodes ont été proposés dans [Tonneau 2018a]. Nous rappelons que les règles imposées au générateur de contact sont les suivantes :

- Une patte en contact pour $\mathbf{q}_{i-1}^{\{j\}}$ est maintenue en contact pour $\mathbf{q}_i^{\{k\}}$ si les contraintes cinématiques peuvent être respectées ;
- $\mathbf{q}_i$ est sans collision avec l'environnement ni auto-collision ;
- $\mathbf{x}_i$ satisfait les contraintes dynamique de non glissement imposées par la phase de contact $\{k\}$;
- au maximum **un** contact n'est pas maintenu (supprimé) entre $\mathbf{q}_{i-1}^{\{j\}}$ et $\mathbf{q}_i^{\{k\}}$;
- au maximum **un** contact est créé (ajouté) entre $\mathbf{q}_{i-1}^{\{j\}}$ et $\mathbf{q}_i^{\{k\}}$;

Les notations sont celles définies dans la section 6.3.

B.1 Maintenir un contact

Afin d'essayer de maintenir les contacts de la phase de contact précédente $\{j\}$ pour la nouvelle configuration $\mathbf{q}_i$, nous utilisons une méthode de cinématique inverse du premier ordre avec contraintes.

La cinématique inverse est initialisée avec la configuration $\mathbf{q}_{i-1}$. Les positions des organes terminaux en contact définies par la phase de contact $\{j\}$ sont alors contraintes à rester constantes. Puis, la position et l'orientation de l'origine du robot ainsi que les valeurs des degrés de liberté du tronc sont projetées vers leurs valeurs dans $\mathbf{q}_i^{trunk}$. Cette méthode est illustrée par la figure B.1.

Si la cinématique inverse échoue on essais à nouveau en supprimant un contact, c'est à dire en enlevant une des contraintes fixant la position d'un organe terminal en contact, jusqu'à ce que la cinématique inverse converge. Les contacts les plus anciens sont essayés en premier.

Si plusieurs contacts doivent être supprimés, des configurations intermédiaires sont ajoutées afin de garantir qu'il y a toujours un maximum d'un contact supprimé entre chaque configuration.

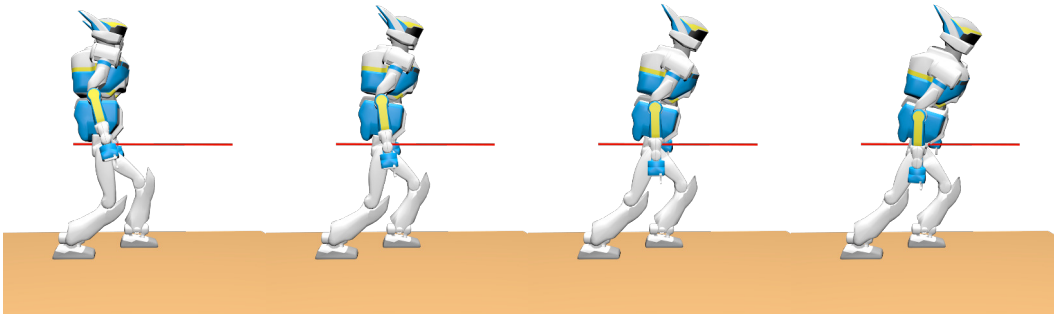


FIGURE B.1 – Illustration de la méthode de maintien de contact par cinématique inverse. L'origine du robot suit le guide en rouge tandis que les contacts entre les pieds et le sol sont maintenus constant. La vignette de droite montre la première configuration ne permettant pas de satisfaire les contraintes cinématiques : le genou droit est en butée articulaire.

B.2 Test de collision

Afin de vérifier si une configuration est en (auto-)collision, la bibliothèque FCL (Flexible Collision Library) est utilisée. La trajectoire guide assure que le tronc du robot est toujours sans collision, à cette étape il ne peut donc y avoir que les pattes du robot qui génèrent des collisions.

Lorsqu'une collision pour une des pattes est détectée, si il s'agit d'une patte qui n'est pas en contact des configurations aléatoires pour cette patte sont échantillonnées jusqu'à en trouver une sans collision. Si il s'agit d'une patte en contact, le contact est tout d'abord supprimé puis des configurations aléatoires sont échantillonnées comme précédemment.

B.3 Test des contraintes dynamiques

Afin de vérifier si une configuration associée une phase de contact $\mathbf{q}_i^{\{k\}}$ permet d'être en équilibre, il faut vérifier si les contraintes dynamiques de non glissement imposées par la phase de contact $\{k\}$ sont satisfaites pour l'état centroïdal courant $\mathbf{x}_i^{\{k\}}$. Pour cela, nous utilisons le LP défini dans la section 2.3.3 en y ajoutant le critère de robustesse proposé par [Del Prete 2016] afin de garantir une marge de robustesse aux incertitudes et imprécisions.

Ici $\mathbf{x}_i$ est fixé par la trajectoire guide, donc si $\mathbf{x}_i^{\{k\}}$ ne satisfait pas les contraintes dynamiques il faut modifier la phase de contact $\{k\}$. Pour cela, s'il reste des pattes qui ne sont pas en contact mais dont l'espace atteignable intersecte avec l'environnement, le générateur de contact essaie de créer des contacts avec ces pattes en essayant d'abord les pattes qui sont restées sans contact le plus longtemps.

Si il n'est pas possible d'ajouter des contacts la méthode essaie de repositionner

un contact existant, pour cela un contact est rompu puis on essaie de générer un nouveau contact pour la même patte. On essaie d'abord de repositionner le contact de la patte qui est en contact depuis le plus longtemps.

D'après le critère de faisabilité proposé dans le chapitre 2 et utilisé pendant la génération de la trajectoire guide, nous savons qu'il existe au moins une phase de contact $\{p\}$ telle que $\mathbf{x}_i^{\{p\}}$ satisfait les contraintes de non glissement et que les points de contact sont compris dans l'espace atteignable par les organes terminaux lorsque le tronc du robot est à la configuration $\mathbf{q}_i^{trunk}$.

Cependant, même si ces points de contact sont compris dans l'espace atteignable, nous n'avons pas de garantie qu'il existe une configuration corps complet cinématiquement valide et sans collision qui permet de créer ces points de contact. L'heuristique utilisée pendant la planification du guide permet seulement de donner une forte probabilité que cette configuration existe.

Cette étape peut donc échouer, dans ce cas l'algorithme s'arrête et retourne la séquence de configurations calculée jusqu'à présent.

B.4 Création de contact

Pour chaque patte, une base de donnée de configurations est générée hors ligne. Ces configurations sont stockées dans une structure d'octree en utilisant la position de l'organe terminal comme index (figure B.2, ligne du haut).

Durant l'exécution, lorsqu'une requête de création de contact est faite pour une patte, l'intersection entre l'octree et l'environnement est calculée (figure B.2-2). Cela permet d'obtenir une liste de configurations candidates pour la patte résultant en une position de l'organe terminal proche du contact (avec une distance maximale au contact dépendante de la résolution de l'octree). Cette liste de candidats est ensuite classée selon une heuristique définie par l'utilisateur.

Finalement, en commençant par les candidates ayant le meilleur score pour l'heuristique, une méthode de projection est utilisée afin de projeter les configurations pour que l'organe terminal soit en contact avec l'environnement. Cela jusqu'à trouver un candidat *valide* (figure B.2-3). Ici, une configuration est *valide* si elle est sans collision et permet au robot de satisfaire les contraintes dynamiques.

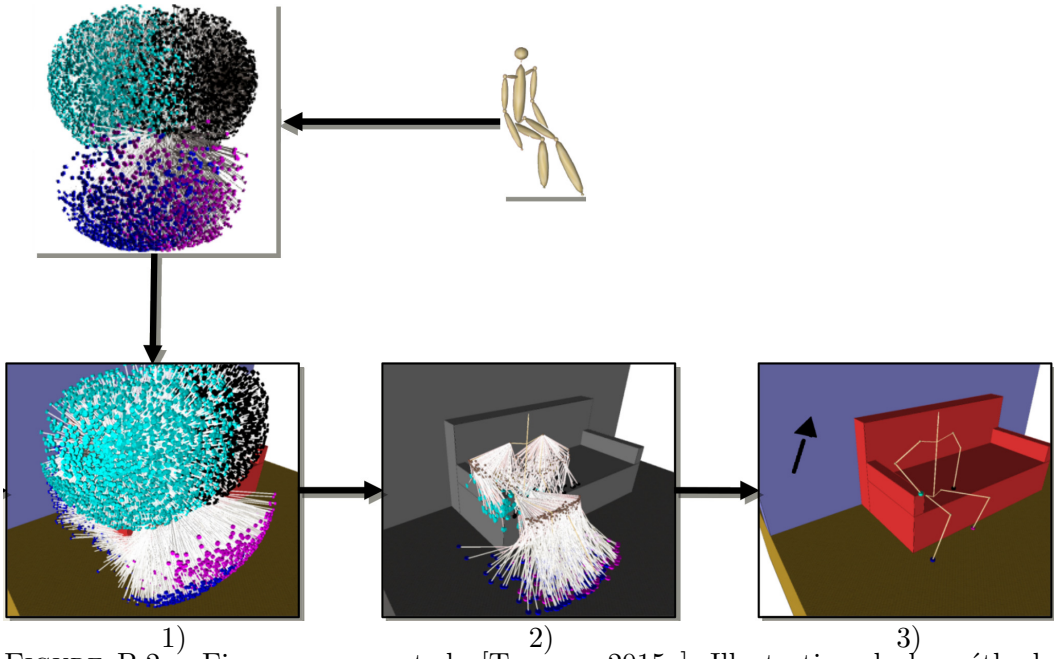


FIGURE B.2 – Figure provenant de [Tonneau 2015a]. Illustration de la méthode de génération de contact. Ligne du haut : la génération de l'octree contenant les configurations. Ligne du bas : l'origine de l'octree est placé dans l'environnement (1) puis l'intersection est calculée (2). En (3) la configuration trouvée est affichée.

Bibliographie

[Aceituno-Cabezas 2018] Bernardo Aceituno-Cabezas, Carlos Mastalli, Hongkai Dai, Michele Focchi, Andreea Radulescu, Darwin G. Caldwell, Juan C. Grieco, Gerardo Fernx00E1ndez-Lx00F3pez et Claudio Semini. *Simultaneous Contact, Gait, and Motion Planning for Robust Multilegged Locomotion via Mixed-Integer Convex Optimization.* IEEE Robotics and Automation Letters, vol. 3, pages 2531–2538, 2018. (Cité en page 8.)

[Ali 2010] M. A. Ali, H. Andy Park et C. S. G. Lee. *Closed-form inverse kinematic joint solution for humanoid robots.* Dans 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 704–709, Oct 2010. (Cité en page 20.)

[Arisumi 2008] H. Arisumi, S. Miossec, J. Chardonnet et K. Yokoi. *Dynamic lifting by whole body motion of humanoid robots.* Dans 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 668–675, Sept 2008. (Cité en page 87.)

[Atkeson 2015] C. G. Atkeson, B. P. W. Babu, N. Banerjee, D. Berenson, C. P. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin, M. Gennert, J. P. Graft, P. He, A. Jaeger, K. Knoedler J. Kim, L. Li, X. Long C. Liu, T. Padir, F. Polido, G. G. Tighe et X. Xinjilefu. *What Happened at the DARPA Robotics Challenge, and Why ?* Rapport technique, Carnegie Mellon University, Pittsburgh, USA, 2015. (Cité en page 4.)

[Baerlocher 2004] Paolo Baerlocher et Ronan Boulic. *An inverse kinematics architecture enforcing an arbitrary number of strict priority levels.* The Visual Computer, vol. 20, no. 6, juin 2004. (Cité en page 78.)

[Bouyarmane 2009] Karim Bouyarmane, Adrien Escande, Florent Lamiraux et Abderrahmane Kheddar. *Potential field guide for humanoid multicontacts acyclic motion planning.* Dans Proc. of IEEE Int. Conf. on Robot. and Auto (ICRA), pages 1165 – 1170, Kobe, Japan, 2009. (Cité en pages 9, 44 et 45.)

[Bouyarmane 2012] Karim Bouyarmane et Abderrahmane Kheddar. *Humanoid Robot Locomotion and Manipulation Step Planning.* Advanced Robotics, vol. 26, no. 10, pages 1099–1126, 2012. (Cité en pages 9 et 45.)

[Bouyarmane 2018] Karim Bouyarmane, Stéphane Caron, Adrien Escande et Abderrahmane Kheddar. *Multi-contact Motion Planning and Control.* Dans Ambarish Goswami et Prahlad Vadakkepat, editeurs, Humanoid Robotics : A Reference, pages 1–42. Springer Netherlands, Dordrecht, janvier 2018. (Cité en pages 8 et 13.)

[Bretl 2004] Timothy Bretl, Stephen Rock, Jean-Claude Latombe, Brett Kennedy et Hrand Aghazarian. *Free-Climbing with a Multi-Use Robot.* Dans Marcelo H. Ang Jr. et Oussama Khatib, editeurs, ISER, volume 21 of *Springer Tracts in Advanced Robot.*, pages 449–458. Springer, 2004. (Cité en page 76.)

[Bretl 2006a] T. Bretl et S. Lall. *A fast and adaptive test of static equilibrium for legged robots*. Dans Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., pages 1109–1116, May 2006. (Cité en page 26.)

[Bretl 2006b] Timothy Bretl. *Motion Planning of Multi-Limbed Robots Subject to Equilibrium Constraints : The Free-Climbing Robot Problem*. The Int. Journal of Robot. Research (IJRR), vol. 25, no. 4, pages 317–342, avril 2006. (Cité en page 7.)

[Bretl 2008] T. Bretl et S. Lall. *Testing Static Equilibrium for Legged Robots*. IEEE Transactions on Robotics, vol. 24, no. 4, pages 794–807, Aug 2008. (Cité en page 26.)

[Brossette 2017] Stanislas Brossette et Pierre-Brice Wieber. *Collision avoidance based on separating planes for feet trajectory generation*. Dans Humanoids 2017 - IEEE RAS International Conference on Humanoid Robots , pages 509–514, Birmingham, United Kingdom, novembre 2017. IEEE. (Cité en page 144.)

[Budhiraja 2018] Rohan Budhiraja, Justin Carpentier, Carlos Mastalli et Nicolas Mansard. Differential Dynamic Programming for Multi-Phase Rigid Contact Dynamics. working paper or preprint, juillet 2018. (Cité en page 192.)

[Campana 2016a] Mylène Campana, Pierre Fernbach, Steve Tonneau, Michel Taïx et Jean-Paul Laumond. *Ballistic motion planning for jumping superheroes*. Dans Motion in Games Conference, Burlingame, CA, United States, octobre 2016. (Cité en page 40.)

[Campana 2016b] Mylène Campana et Jean-Paul Laumond. *Ballistic motion planning*. Dans IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016), Daejeon, South Korea, octobre 2016. Finalist of the Best Paper Award on Safety Security and Rescue Robotics. (Cité en pages 40, 41, 42 et 60.)

[Caron 2015] Stéphane Caron, Quang-Cuong Pham et Yoshihiko Nakamura. *Leveraging Cone Double Description for Multi-contact Stability of Humanoids with Applications to Statics and Dynamics*. Dans Robotics, Science and Systems (RSS), 2015. (Cité en pages 25, 26, 53 et 82.)

[Caron 2017] Stéphane Caron, Quang-Cuong Pham et Yoshihiko Nakamura. *ZMP Support Areas for Multi-contact Mobility Under Frictional Constraints*. IEEE Transactions on Robotics, vol. 33, no. 1, pages 67–80, février 2017. (Cité en page 5.)

[Caron 2018] Stéphane Caron, Adrien Escande, Leonardo Lanari et Bastien Mallein. Capturability-based analysis, optimization and control of 3d bipedal walking. Submitted, janvier 2018. (Cité en pages 10, 11, 75, 76, 87, 119 et 130.)

[Carpentier 2016] Justin Carpentier, Steve Tonneau, Maximilien Naveau, Olivier Stasse et Nicolas Mansard. *A versatile and efficient pattern generator for*

generalized legged locomotion. Dans Proc. of IEEE Int. Conf. on Robot. and Auto (ICRA), Stockholm, Sweden, 2016. (Cité en pages 10 et 168.)

[Carpentier 2017a] Justin Carpentier, Rohan Budhiraja et Nicolas Mansard. *Learning Feasibility Constraints for Multi-contact Locomotion of Legged Robots*. Dans Robotics : Science and Systems, Cambridge, MA, United States, juillet 2017. (Cité en page 79.)

[Carpentier 2017b] Justin Carpentier, Andrea Del Prete, Steve Tonneau, Thomas Flayols, Florent Forget, Alexis Mifsud, Kevin Giraud, Dinesh Atchuthan, Pierre Fernbach, Rohan Budhiraja, Mathieu Geisert, Joan Solà, Olivier Stasse et Nicolas Mansard. *Multi-contact Locomotion of Legged Robots in Complex Environments – The Loco3D project*. Dans RSS Workshop on Challenges in Dynamic Legged Locomotion, page 3p., Boston, United States, juillet 2017. (Cité en pages 11, 133 et 203.)

[Carpentier 2017c] Justin Carpentier et Nicolas Mansard. *Multi-contact Locomotion of Legged Robots*. Rapport LAAS n° 17172. https ://hal.laas.fr/hal-01520248. Conditionally accepted for IEEE Trans. on Robotics, 2017. (Cité en pages 2, 10, 12, 22, 75, 76, 77, 86, 87, 106, 107, 108, 109, 113, 119, 126, 130, 133, 136, 137, 142, 165, 185, 188 et 199.)

[Chestnutt 2003] Joel E. Chestnutt, James J. Kuffner, Koichi Nishiwaki et Satoshi Kagami. *Planning Biped Navigation Strategies in Complex Environments*. Dans In IEEE Int. Conf. Hum. Rob., Munich, Germany, 2003. (Cité en page 6.)

[Chung 2015] Shu-Yun Chung et O. Khatib. *Contact-consistent elastic strips for multi-contact locomotion planning of humanoid robots*. Dans Proc. of IEEE Int. Conf. on Robot. and Auto (ICRA), pages 6289–6294, May 2015. (Cité en page 10.)

[Dai 2014] Hongkai Dai, Andrés Valenzuela et Russ Tedrake. *Whole-body motion planning with centroidal dynamics and full kinematics*. Dans Humanoid Robots (Humanoids), 14th IEEE-RAS Int. Conf. on, pages 295–302, Madrid, Spain, 2014. (Cité en pages 9, 75, 78, 87, 107, 119, 130 et 202.)

[Deits 2014] Robin Deits et Russ Tedrake. *Footstep planning on uneven terrain with mixed-integer convex optimization*. Dans Humanoid Robots (Humanoids), 14th IEEE-RAS Int. Conf. on, Madrid, Spain, 2014. (Cité en pages 8, 9, 68, 72 et 76.)

[Del Prete 2016] Andrea Del Prete, Steve Tonneau et Nicolas Mansard. *Fast Algorithms to Test Robust Static Equilibrium for Legged Robots*. Dans Proc. of IEEE Int. Conf. on Robot. and Auto (ICRA), Stockholm, Sweden, 2016. (Cité en pages 26, 53, 55, 81, 155 et 224.)

[Del Prete 2018] Andrea Del Prete, Steve Tonneau et Nicolas Mansard. *Zero Step Capturability for Legged Robots in Multi Contact*. Accepted on IEEE Trans on Robotics, 2018. (Cité en page 77.)

[Escande 2008] Adrien Escande, Abderrahmane Kheddar, Sylvain Miossec et Sylvain Garsault. *Planning Support Contact-Points for Acyclic Motions and Experiments on HRP-2.* Dans ISER'08 : 11th International Symposium on Experimental Robotics, pages 293–302, Athens, Greece, juillet 2008. (Cité en pages 9 et 44.)

[Escande 2013] Adrien Escande, Abderrahmane Kheddar et Sylvain Miossec. *Planning contact points for humanoid robots.* Robotics and Autonomous Systems, vol. 61, no. 5, pages 428–442, mai 2013. (Cité en pages 9, 44, 76 et 191.)

[Farshidian 2016] Farbod Farshidian, Michael Neunert, Alexander W. Winkler, Gonzalo Rey et Jonas Buchli. *An Efficient Optimal Planning and Control Framework For Quadrupedal Locomotion.* CoRR, vol. abs/1609.09861, 2016. (Cité en page 155.)

[Fernbach 2017] P. Fernbach, S. Tonneau, A. Del Prete et M. Taïx. *A kinodynamic steering-method for legged multi-contact locomotion.* Dans 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3701–3707, Sept 2017. (Cité en page 58.)

[Fernbach 2018] Pierre Fernbach, Steve Tonneau et Michel Taïx. *CROC : Convex Resolution Of Centroidal dynamics trajectories to provide a feasibility criterion for the multi contact planning problem.* Dans 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 2018. (Cité en pages 87, 90 et 119.)

[Focchi 2013] Michele Focchi, Victor Barasuol, Ioannis Havoutis, Jonas Buchli, Claudio Semini et Darwin G. Caldwell. *Local reflex generation for obstacle negotiation in quadrupedal locomotion.* Dans Nature-Inspired Mobile Robotics, pages 443–450, 08 2013. (Cité en pages 144 et 162.)

[Fukuda 1996] Komei Fukuda et Alain Prodon. Double description method revisited, pages 91–111. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996. (Cité en page 82.)

[Gabiccini 2015] Marco Gabiccini, Alessio Artoni, Gabriele Pannocchia et Joris Gillis. *A Computational Framework for Environment-Aware Robotic Manipulation Planning.* Dans Int. Symp. Robotics Research (ISRR), Sestri Levante, Italy, 2015. (Cité en page 8.)

[García Armada 2003] Elena García Armada, Joaquín Estremera et Pablo González de Santos. *A control architecture for humanitarian-demining legged robots.* Dans International conference on Climbing and Walking Robots (6º. 2003. Catania, Italia, 2003. (Cité en pages 144 et 162.)

[Geraerts 2007] Roland Geraerts et Mark H. Overmars. *Creating High-quality Paths for Motion Planning.* The International Journal of Robotics Research, vol. 26, no. 8, pages 845–863, 2007. (Cité en page 151.)

[Grey 2017] Michael X. Grey, Aaron D. Ames et C. Karen Liu. *Footstep and Motion Planning in Semi-unstructured Environments Using Randomized Possibility*

Graphs. Dans IEEE International Conference on Robotics and Automation (ICRA), 2017. (Cité en page 6.)

[Guay 2015] Martin Guay, Rémi Ronfard, Michael Gleicher et Marie-Paule Cani. *Space-time sketching of character animation*. ACM Transactions on Graphics, vol. 34, no. 4, page Article No. 118, août 2015. (Cité en page 9.)

[Hauser 2005] K Hauser, T Bretl et J.-C. Latombe. *Non-gaited humanoid locomotion planning*. Dans Humanoid Robots, 2005 5th IEEE-RAS Int. Conf. on, pages 7–12, 2005. (Cité en page 76.)

[Hauser 2006] Kris Hauser, Timothy Bretl, Kensuke Harada et Jean-Claude Latombe. *Using Motion Primitives in Probabilistic Sample-Based Planning for Humanoid Robots*. Dans Srinivas Akella, Nancy M. Amato, Wesley H. Huang et Bud Mishra, editeurs, WAFR, volume 47 of *Springer Tracts in Advanced Robot*. Springer, 2006. (Cité en page 7.)

[Hauser 2010] K. Hauser et V. Ng-Thow-Hing. *Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts*. Dans IEEE Inter. Conf. on Robotics and Automation (ICRA), May 2010. (Cité en pages 35, 39 et 53.)

[Hauser 2014] Kris Hauser. *Fast interpolation and time-optimization with contact*. The Int. Journal of Robot. Research (IJRR), vol. 33, no. 9, pages 1231–1250, août 2014. (Cité en page 10.)

[Herzog 2015] Alexander Herzog, Nicholas Rotella, Stefan Schaal et Ludovic Righetti. *Trajectory generation for multi-contact momentum-control*. Dans Humanoid Robots (Humanoids), 15h IEEE-RAS Int. Conf. on, novembre 2015. (Cité en pages 10, 75, 87, 119 et 130.)

[Herzog 2016] Alexander Herzog, Stefan Schaal et Ludovic Righetti. *Structured contact force optimization for kino-dynamic motion generation*. CoRR, vol. abs/1605.08571, 2016. (Cité en pages 10, 78, 115 et 155.)

[Holden 2017] Daniel Holden, Taku Komura et Jun Saito. *Phase-functioned Neural Networks for Character Control*. ACM Trans. Graph., vol. 36, no. 4, pages 42 :1–42 :13, juillet 2017. (Cité en page 9.)

[Kajita 2001] S. Kajita, K. Yokoi, M. Saigo et K. Tanie. *Balancing a humanoid robot using backdrive concerned torque control and direct angular momentum feedback*. Dans Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164), volume 4, pages 3376–3382 vol.4, May 2001. (Cité en page 154.)

[Kajita 2003] S Kajita, F Kanehiro, K Kaneko, K Fujiwara, K Harada, K Yokoi et H Hirukawa. *Biped Walking Pattern Generation by using Preview Control of Zero-Moment Point*. Dans Proc. of IEEE Int. Conf. Robot. and Auto (ICRA), Taipei, Taiwan, septembre 2003. (Cité en pages 5, 6, 43, 75, 155 et 159.)

[Kajita 2005] S. Kajita, T. Nagasaki, K. Kaneko, K. Yokoi et K. Tanie. *A Running Controller of Humanoid Biped HRP-2LR*. Dans Proceedings of the 2005

IEEE International Conference on Robotics and Automation, pages 616–622, April 2005. (Cité en page 154.)

[Kalakrishnan 2011] Mrinal Kalakrishnan, Jonas Buchli, Peter Pastor, Michael Mistry et Stefan Schaal. *Learning, planning, and control for quadruped locomotion over challenging terrain.* The International Journal of Robotics Research, vol. 30, no. 2, pages 236–258, 2011. (Cité en page 144.)

[Kavraki 1996] L E Kavraki, P Svestka, J.-C. Latombe et M H Overmars. *Probabilistic roadmaps for path planning in high-dimensional configuration spaces.* Robot. and Auto, IEEE Trans. on, vol. 12, no. 4, pages 566–580, 1996. (Cité en page 32.)

[Koolen 2012] Twan Koolen, Tomas de Boer, John R. Rebula, Ambarish Goswami et Jerry E. Pratt. *Capturability-based analysis and control of legged locomotion, Part 1 : Theory and application to three simple gait models.* I. J. Robotics Res., vol. 31, no. 9, pages 1094–1113, 2012. (Cité en page 77.)

[Kröger 2006] Torsten Kröger, Adam Tomiczek et Friedrich M. Wahl. *Towards On-Line Trajectory Computation.* Dans IEEE/RSJ Inter. Conf. on Intelligent Robots and Systems (IROS), 2006. (Cité en pages 35 et 53.)

[Kuffner Jr 2000] J.J. Kuffner Jr et S.M. LaValle. *RRT-connect : An efficient approach to single-query path planning.* Dans Robot. and Auto, 2000. Proceedings. ICRA'00. IEEE Int. Conf. on, volume 2, pages 995–1001. IEEE, 2000. (Cité en page 35.)

[Kuffner 2001] J. J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba et H. Inoue. *Footstep planning among obstacles for biped robots.* Dans Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems., volume 1, pages 500–505 vol.1, Oct 2001. (Cité en page 6.)

[Kunz 2014] T. Kunz et M. Stilman. *Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits.* Dans IEEE/RSJ Inter. Conf. on Intelligent Robots and Systems (IROS), 2014. (Cité en pages 35, 38, 39, 48, 53 et 71.)

[LaValle 1998] S M LaValle. *Rapidly-Exploring Random Trees : A New Tool for Path Planning.* In, vol. 129, no. 98-11, pages 98–11, 1998. (Cité en page 32.)

[LaValle 1999] S.M. LaValle et Jr. Kuffner J.J. *Randomized kinodynamic planning.* Dans Proc. of IEEE Int. Conf. on Robot. and Auto (ICRA), volume 1, pages 473–479 vol.1, Detroit, Michigan, USA, 1999. (Cité en pages 34 et 44.)

[LaValle 2006] Steven M LaValle. *Planning Algorithms.* Methods, vol. 2006, no. 10 : 0521862051, page 842, 2006. (Cité en pages 32, 34 et 150.)

[Leineweber 2003] Daniel B. Leineweber, Irene Bauer, Hans Georg Bock et Johannes P. Schlöder. *An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part 1 : theoretical aspects.* Computers and Chemical Engineering, vol. 27, no. 2, pages 157 – 166, 2003. (Cité en page 142.)

[Lengagne 2011] S. Lengagne, N. Ramdani et P. Fraisse. *Planning and Fast Replanning Safe Motions for Humanoid Robots.* IEEE Transactions on Robotics, vol. 27, no. 6, pages 1095–1106, Dec 2011. (Cité en page 87.)

[Lengagne 2013] Sébastien Lengagne, Joris Vaillant, Eiichi Yoshida et Abderrahmane Kheddar. *Generation of whole-body optimal dynamic multi-contact motions.* The International Journal of Robotics Research, vol. 32, no. 9-10, pages 1104–1119, 2013. (Cité en pages 88 et 130.)

[Mirabel 2016] J. Mirabel, S. Tonneau, P. Fernbach, A. K. Seppälä, M. Campana, N. Mansard et F. Lamiraux. *HPP : A new software for constrained motion planning.* Dans 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 383–389, Oct 2016. (Cité en pages 52, 67 et 139.)

[Mirabel 2017] Joseph Mirabel. *Manipulation planning for documented objects.* Theses, Institut National Polytechnique De Toulouse, février 2017. (Cité en page 170.)

[Mordatch 2012] Igor Mordatch, Emanuel Todorov et Zoran Popović. *Discovery of complex behaviors through contact-invariant optimization.* ACM Trans. on Graphics, vol. 31, no. 4, pages 43 :1–43 :8, 2012. (Cité en pages 8, 44, 50, 59, 68, 72 et 76.)

[Mordatch 2015] Igor Mordatch, Kendall Lowrey et Emanuel Todorov. *Ensemble-CIO : Full-Body Dynamic Motion Planning that Transfers to Physical Humanoids.* Dans Proc. of IEEE Int. Conf. on Robot. and Auto (ICRA), Seattle, USA, 2015. (Cité en page 8.)

[Naveau 2016] Maximilien Naveau. *Advanced human inspired walking strategies for humanoid robots.* Theses, Université Paul Sabatier - Toulouse III, septembre 2016. (Cité en page 144.)

[Nishiwaki 2014] Koichi Nishiwaki, Joel Chestnutt et Satoshi Kagami. Planning and control of a humanoid robot for navigation on uneven multi-scale terrain, pages 401–415. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. (Cité en page 144.)

[Orin 2013] David E. Orin, Ambarish Goswami et Sung-Hee Lee. *Centroidal dynamics of a humanoid robot.* Autonomous Robots, vol. 35, no. 2, pages 161–176, Oct 2013. (Cité en pages 23 et 24.)

[Orsolino 2018] R. Orsolino, M. Focchi, C. Mastalli, H. Dai, D. G. Caldwell et C. Semini. *Application of Wrench based Feasibility Analysis to the Online Trajectory Optimization of Legged Robots.* IEEE Robotics and Automation Letters, pages 1–1, 2018. (Cité en page 115.)

[Park 2016] Chonhyon Park, Jae Sun Park, Steve Tonneau, Nicolas Mansard, Franck Multon, Julien Pettré et Dinesh Manocha. *Dynamically Balanced and Plausible Trajectory Planning for Human-Like Characters.* Dans To appear in Proc. of I3D '16, Seatle, USA, 2016. (Cité en page 10.)

[Peng 2017] Xue Bin Peng, Glen Berseth, KangKang Yin et Michiel van de Panne. *DeepLoco : Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning*. ACM Transactions on Graphics (Proc. SIGGRAPH 2017), vol. 36, no. 4, 2017. (Cité en page 9.)

[Perrin 2012] Nicolas Perrin, Olivier Stasse, Leo Baudouin, Florent Lamiraux et Eiichi Yoshida. *Fast Humanoid Robot Collision-Free Footstep Planning Using Swept Volume Approximations*. IEEE Trans. Robotics, vol. 28, no. 2, pages 427–439, 2012. (Cité en page 6.)

[Ponton 2016] B. Ponton, A. Herzog, S. Schaal et L. Righetti. *A Convex Model of Humanoid Momentum Dynamics for Multi-Contact Motion Generation*. Dans Proceedings of the 2016 IEEE-RAS International Conference on Humanoid Robots, 2016. (Cité en pages 9, 10, 26, 76, 86, 107 et 195.)

[Ponton 2018] B. Ponton, A. Herzog, A. del Prete, S. Schaal et L. Righetti. *On Time Optimization of Centroidal Momentum Dynamics*. Dans Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2018. IEEE, mai 2018. (Cité en pages 10, 86, 104, 130, 142, 201 et 202.)

[Posa 2014] Michael Posa, Cecilia Cantu et Russ Tedrake. *A Direct Method for Trajectory Optimization of Rigid Bodies Through Contact*. The Int. Journal of Robot. Research (IJRR), vol. 33, no. 1, pages 69–81, janvier 2014. (Cité en page 9.)

[Posa 2016] M. Posa, S. Kuindersma et R. Tedrake. *Optimization and stabilization of trajectories for constrained dynamical systems*. Dans 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 1366–1373, May 2016. (Cité en page 44.)

[Pratt 2006] Jerry Pratt, J. Carff, S. Drakunov et Ambarish Goswami. *Capture Point : A Step toward Humanoid Push Recovery*. 2006 6th IEEE-RAS International Conference on Humanoid Robots, 2006. (Cité en page 6.)

[Pratt 2012] Jerry E. Pratt, Twan Koolen, Tomas de Boer, John R. Rebula, Sebastien Cotton, John Carff, Matthew Johnson et Peter D. Neuhaus. *Capturability-based analysis and control of legged locomotion, Part 2 : Application to M2V2, a lower-body humanoid*. I. J. Robotics Res., vol. 31, no. 10, pages 1117–1133, 2012. (Cité en page 77.)

[Qiu 2011] Zhapeng Qiu, Adrien Escande, Alain Micaelli et Thomas Robert. *Human motions analysis and simulation based on a general criterion of stability*. Dans Inter. Symposium on Digital Human Modeling, 2011. (Cité en pages 26 et 82.)

[Qiu 2012] Z. Qiu, A. Escande, A. Micaelli et T. Robert. *A hierarchical framework for realizing dynamically-stable motions of humanoid robot in obstacle-cluttered environments*. Dans 2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012), pages 867–874, Nov 2012. (Cité en pages 144 et 150.)

[Risler 1992] J. J. Risler. Mathematical methods for cad. Cambridge University Press, Cambridge, UK, 1992. (Cité en page 92.)

[Saab 2013] L. Saab, O. E. Ramos, F. Keith, N. Mansard, P. Souères et J. Y. Fourquet. *Dynamic Whole-Body Motion Generation Under Rigid Contacts and Other Unilateral Constraints*. IEEE Transactions on Robotics, vol. 29, no. 2, pages 346–362, April 2013. (Cité en page 143.)

[Samy 2017] V. Samy, S. Caron, K. Bouyarmane et A. Kheddar. *Post-impact adaptive compliance for humanoid falls using predictive control of a reduced model*. Dans 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), pages 655–660, Nov 2017. (Cité en page 115.)

[Schwartz 1983] Jacob T. Schwartz et Micha Sharir. *On the "piano movers'" problem I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers*. Communications on Pure and Applied Mathematics, vol. 36, no. 3, pages 345–398, 1983. (Cité en page 31.)

[Short 2018] A. Short et T. Bandyopadhyay. *Legged Motion Planning in Complex Three-Dimensional Environments*. IEEE Robotics and Automation Letters, vol. 3, no. 1, pages 29–36, Jan 2018. (Cité en page 191.)

[Siméon 2004] T. Siméon, J.P. Laumond, J. Cortes et A. Sahbani. *Manipulation planning with probabilistic roadmaps*. The Int. Journal of Robot. Research (IJRR), vol. 23, no. 7-8, pages 729–746, 2004. (Cité en page 44.)

[Stasse 2017] Olivier Stasse, Thomas Flayols, Rohan Budhiraja, Kevin Giraud-Esclasse, Justin Carpentier, Joseph Mirabel, Andrea Del Prete, Philippe Souères, Nicolas Mansard, Florent Lamiraux, Jean-Paul Laumond, Luca Marchionni, Hilario Tome et Francesco Ferro. *TALOS : A new humanoid research platform targeted for industrial applications*. Dans International Conference on Humanoid Robotics, ICHR, Birmingham 2017, IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)„ Birmingham, United Kingdom, novembre 2017. IEEE. (Cité en pages 22 et 179.)

[Tonneau 2015a] Steve Tonneau. *Motion planning and synthesis for virtual characters in constrained environments*. Theses, INSA de Rennes, 2015. (Cité en page 226.)

[Tonneau 2015b] Steve Tonneau, Nicolas Mansard, Chonhyon Park, Dinesh Manocha, Franck Multon et Julien Pettré. *A reachability-based planner for sequences of acyclic contacts in cluttered environments*. Dans Int. Symp. Robotics Research (ISRR), Sestri Levante, Italy, 2015. (Cité en pages 10 et 45.)

[Tonneau 2018a] S. Tonneau, A. D. Prete, J. Pettré, C. Park, D. Manocha et N. Mansard. *An Efficient Acyclic Contact Planner for Multiped Robots*. IEEE Transactions on Robotics, pages 1–16, 2018. (Cité en pages 10, 12, 45, 47, 76, 115, 133, 139, 140, 168 et 223.)

[Tonneau 2018b] Steve Tonneau, Pierre Fernbach, Andrea Del Prete, Julien Pettré et Nicolas Mansard. *2PAC : Two Point Attractors for Center of Mass Tra-*

jectories in Multi Contact Scenarios. Transaction on Graphics, Jul 2018. (Cité en pages 76, 78, 79 et 101.)

[Vaillant 2016] Joris Vaillant, Abderrahmane Kheddar, Hervé Audren, François Keith, Stanislas Brossette, Adrien Escande, Karim Bouyarmane, Kenji Kaneko, Mitsuharu Morisawa, Pierre Gergondet, Eiichi Yoshida, Suuji Kajita et Fumio Kanehiro. *Multi-contact vertical ladder climbing with an HRP-2 humanoid*. Autonomous Robots, vol. 40, no. 3, pages 561–580, 2016. (Cité en page 2.)

[Verrelst 2006] B. Verrelst, O. Stasse, K. Yokoi et B. Vanderborght. *Dynamically Stepping Over Obstacles by the Humanoid Robot HRP-2*. Dans 2006 6th IEEE-RAS International Conference on Humanoid Robots, pages 117–123, Dec 2006. (Cité en page 144.)

[Winkler 2017] A. W. Winkler, F. Farshidian, M. Neunert, D. Pardo et J. Buchli. *Online Walking Motion and Foothold Optimization for Quadruped Locomotion*. Dans IEEE Inter. Conf. on Robotics and Automation (ICRA), May 2017. (Cité en page 44.)

[Winkler 2018] Alexander W Winkler, Dario C Bellicoso, Marco Hutter et Jonas Buchli. *Gait and Trajectory Optimization for Legged Systems through Phase-based End-Effector Parameterization*. IEEE Robotics and Automation Letters (RA-L), vol. 3, pages 1560–1567, July 2018. (Cité en pages 8, 13, 86, 130 et 201.)

Abstract : The automatic synthesis of movements for legged robots is one of the long standing challenge of robotics, and its resolution is a prior to the safe deployment of robots outside of their labs. In this thesis, we tackle it with a divide and conquer approach, where several smaller sub-problems are identified and solved sequentially to generate motions in a computationally efficient manner.

This decoupling comes with a feasibility issue : how can we guarantee that the solution of a sub-problem is a valid input for the next sub-problem ? To address this issue, this thesis defines computationally efficient feasibility criteria, focused on the constraints on the Center Of Mass of the robot.

Simultaneously, it proposes a new formulation of the problem of computing a feasible trajectory for the Center Of Mass of the robot, given a contact sequence. This formulation is continuous, as opposed to traditional approaches that rely on a discretized formulation, which can result in constraint violations and are less computationally efficient. This general formulation could be straightforwardly used with any existing approach of the state of the art.

The framework obtained was experimentally validated both in simulation and on the HRP-2 robot, and presented a higher success rate, as well as computing performances order of magnitudes faster than the state of the art.

Keywords : Motion and kinodynamic planning, legged robots, multi-contact, optimization

Résumé : La synthèse automatique du mouvement de robots à pattes est un enjeu majeur de la robotique : sa résolution permettrait le déploiement des robots hors de leurs laboratoire. Pour y parvenir, cette thèse suit l'approche "diviser pour régner", où le problème est décomposé en plusieurs sous-problèmes résolus séquentiellement.

Cette décomposition amène alors la question nouvelle de la faisabilité : comment garantir que la solution d'un sous-problème, permet la résolution des suivants (dont elle sert d'entrée) ? Pour y répondre, cette thèse définit des critères de faisabilités efficaces, qui s'appuient sur la définition des contraintes qui s'appliquent au centre de masse du robot.

En parallèle, et de manière plus générale, elle propose une nouvelle formulation du problème du calcul d'une trajectoire valide pour le centre de masse du robot. Cette formulation, continue, présente le double avantage (par rapport aux méthodes discrètes classiques) de garantir la validité de la solution en tous points, tout en améliorant, grâce à une réduction de la dimensionnalité du problème, les performances des algorithmes de l'état de l'art.

L'architecture de planification de mouvement résultante a été validée en simulation, ainsi que sur le robot HRP-2, démontrant ainsi sa supériorité en termes de temps de calcul et de taux de succès par rapport à l'existant.

Mots clés : Planification de mouvement, robots à pattes, multi-contact, planification kinodynamique, optimisation