Robot hexapode : développement d'un algorithme de marche et tests

SION, Antoine

Publication date:
2020

UMONS
Université de Mons

Faculté Polytechnique

Robot hexapode : développement d'un algorithme de marche et tests

Rapport final

Travail de Fin d'Études présenté en vue de l'obtention du grade de Master Ingénieur civil mécanicien - spécialisation mécatronique

Antoine SION

POLYTECH MONS

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Sous la direction de
Monsieur le Professeur Alain VANDE WOUWER
Monsieur le Professeur Lionel BIRGLEN

Juin 2020

Abstract

This work is about the development of a gait algorithm for a hexapodal robot from the robotics laboratory of Polytechnique Montréal. First we present a state of the art on walking robots to see the research references in this domain. Another brief state of the art is presented on a new material called Gecko Skin, which is mimicking real gecko skin which can be applied to walking robots to move on walls. Next, we describe our robot and his components such as MX106T motors or the Odroid-C1 used to control the robot. Due to the COVID-19 pandemic, implementation of our algorithm on the real robot was impossible and we used a simulation program, CoppeliaSim, to pursue the work. We explain the methodology used to model the robot as well as the chosen physics engine, Vortex Studio. Then, we present our algorithm which is a fixed gait algorithm. It allows the robot to move along flat ground by choosing X and Y speeds and rotation speed along the Z axis. We also present an exact solution (to limit calculation time) to the inverse kinematics problem used to place the end of the 4 degrees of freedom legs. The last chapter is about the tests performed on the robot. The main focus was put on energy consumption to deduce some optimal parameters (optimal body height of 62,5 mm, step height influencing the needed power, optimal radius of the circle of initial positions of the legs of 300 mm for a step height of 50 or 25 mm and of 312,5 mm for a step height of 75 or 100 mm). We also deduce from torque and speeds graphs of all the joints that the most solicited joints for each leg while walking are the second and third joints. We explain the encountered difficulties to simulate Gecko Skin on the feet of the robot (JKR and Kendall models non applicable directly in CoppeliaSim). Another test is also made to evaluate the capacity of the hexapod to climb a slope. The maximum angle at which the hexapod can climb the slope is 45 ° for a friction coefficient of 1 on the feet of the hexapod. This test also shows that the first joint of every leg becomes the most solicited one when climbing a slope. A gait on rough terrain is also tested and shows the limits of our fixed gait algorithm. We finally conclude that we will need to test the Gecko Skin practically and that it will be necessary to study other gait algorithms (with force feedback for example) to get better results on rough terrain. This work will serve as a basis for future works on the implementation of our code on the robot.

Résumé

Ce travail porte sur le développement d'un algorithme de marche pour un robot hexapode du laboratoire de robotique de la Polytechnique de Montréal. Un état de l'art est d'abord dressé portant sur les robots marcheurs afin de parcourir les références en matière de recherche dans le domaine. Un bref état de l'art sur un matériau nouveau communément appelé Gecko Skin est aussi réalisé, ce dernier imite le comportement de la peau du gecko et peut être appliqué à des robots marcheurs afin de se déplacer sur des murs. Ensuite, on décrit notre robot et les composants le constituant, comme les moteurs MX106T et le contrôleur présent, un Odroid-C1. La pandémie de COVID-19 ayant empêché l'implémentation de l'algorithme sur le robot en laboratoire, un logiciel de simulation est ensuite présenté, CoppeliaSim ainsi que la méthode mise en oeuvre pour modéliser le robot. On explicite aussi le choix de Vortex Studio comme moteur physique. Après, l'algorithme développé pour notre robot est exposé, ce dernier étant un algorithme à démarche fixe. Celui-ci permet au robot de se déplacer sur sol plat en choisissant des vitesses de progression en X et Y ainsi qu'une vitesse de rotation selon Z. La solution exacte (choisie pour limiter le temps de calcul) de cinématique inverse est présentée, servant à positionner les extrémités des pattes à quatre degrés de liberté de notre robot. Le dernier chapitre est quant à lui consacré aux tests réalisés sur le robot en se concentrant sur la puissance consommée par ce dernier afin de déduire certains paramètres optimaux pour celui-ci (hauteur de corps optimale de 62,5 mm, hauteur du pas influençant la puissance à développer, rayon optimal du cercle des positions initiales des pattes de 300 mm pour une hauteur de pas de 50 ou 25 mm et de 312,5 mm pour une hauteur du pas de 75 ou 100 mm). Les liaisons les plus sollicitées lors des déplacements ont pu être observées grâce à des graphes représentant l'évolution des couples et vitesses de chacune des liaisons du robot : les deuxièmes et troisièmes liaisons de chaque patte sont ainsi les plus fortement utilisées. On explique ensuite les difficultés rencontrées pour simuler le comportement de la Gecko Skin sur les pattes de l'hexapode (modèles JKR et Kendall non applicables à CoppeliaSim). On évoque par après un test où l'hexapode grimpe une pente et où l'angle de décrochage apparaît être de 45 ° pour un coefficient de frottement de 1 sur les pattes de l'hexapode. Ce test a permis de montrer que la liaison la plus sollicitée de chaque patte était la première liaison lors d'une démarche sur pente. Une démarche sur terrain accidenté à également été testée et a permis de montrer les limites de notre algorithme. On conclut enfin par le fait qu'il faudra recourir à l'expérimentation pratique pour tester la Gecko Skin sur notre robot et qu'il faudra explorer d'autres algorithmes de démarche (feedback de force par exemple) afin d'obtenir de meilleurs résultats sur terrain accidenté. Ce travail servira ainsi de base à l'implémentation du code développé sur le robot pour des travaux futurs.

Remerciements

Tout d'abord, je tiens à remercier mes promoteurs Lionel Birglen et Alain Vande Wouwer pour leur suivi tout au long du projet et leurs conseils précieux pour la rédaction de mon rapport.

Je remercie aussi tous les stagiaires du laboratoire de robotique de Montréal pour les discussions intéressantes portant sur mon projet et les leurs. Une réflexion extérieure apporte toujours quelque chose en plus au travail de chacun.

Pour finir, je remercie ma famille pour m'avoir soutenu durant toute cette période compliquée de confinement.

Table des matières

Chapitre 1

Introduction

Ce travail porte sur un robot hexapode qu'a acquis la Faculté Polytechnique de Montréal au début de l'année 2020. Le but était d'étudier celui-ci afin de pouvoir le remettre en état de fonctionnement, aussi bien au niveau du hardware que du software en implémentant un algorithme lui permettant de se mouvoir de manière coordonnée. Suite à cette première étape, différents tests allaient pouvoir être mis en place sur ce robot, dont un nouveau matériau appelé Gecko Skin imitant la peau du gecko et ayant des propriétés intéressantes au niveau de l'adhérence. Le robot a ainsi pour vocation de devenir une plateforme modulable afin de permettre l'expérimentation de nouvelles méthodes et composants sur ce dernier. À cause de la pandémie de COVID-19, le travail sur le robot en lui-même n'a malheureusement duré qu'un mois et demi car le laboratoire a fermé. Une simulation numérique a alors été mise en place afin de pouvoir continuer à travailler indirectement sur le robot à distance.

Un état de l'art sur les robots marcheurs a d'abord été dressé afin de voir quelles étaient les références dans ce domaine et les différents types de robots à pattes existants. Un bref état de l'art sur la Gecko Skin a aussi été réalisé afin d'exposer son fonctionnement, ses principales techniques de fabrication et des exemples d'application, entre autres sur des robots. Cela a une importance car une des finalités de ce travail était d'étudier l'application de ce type de matériau sur les pattes du robot pour le faire grimper sur un mur.

Ensuite, un chapitre est consacré à la description de notre robot. Les composants le constituant y sont exposés et expliqués. Un autre chapitre est quant à lui dédié à la simulation du robot. On décrit dans celui-ci comment notre robot a été simulé, avec quel logiciel et quel moteur physique a été choisi. Puis un autre chapitre détaille le contrôle de notre robot et l'algorithme développé pour cela. Nous explicitons aussi quelques notions fondamentales liées aux robots marcheurs comme les types de démarches existants. Une section est aussi consacrée à la cinématique inverse des pattes de notre robot.

Le dernier chapitre est quant à lui consacré aux tests effectués sur notre robot et les résultats associés. Un intérêt particulier est porté à l'énergie consommée par le robot afin de déterminer certains paramètres optimaux pour ce dernier. On explicite aussi les difficultés rencontrées afin de simuler la Gecko Skin ainsi que le comportement de l'hexapode sur une pente et sur un terrain plus accidenté. Enfin, nous concluons sur le travail réalisé et les perspectives à développer sur le robot dans le futur.

Chapitre 2

État de l'art

Ce chapitre se divise en deux parties, une présentant un état de l'art sur les robots marcheurs et une autre présentant un bref état de l'art sur la Gecko Skin.

2.1 Les robots marcheurs

Cette section présente d'une manière qui se veut générale une chronologie des différents robots marcheurs ayant existé et ayant apporté une contribution significative à la recherche dans ce domaine. Il est tout d'abord bon de préciser les avantages et les désavantages dont disposent les robots marcheurs par rapport à d'autres robots mobiles à roues ou à chenilles par exemple, comme il en est question dans [1].

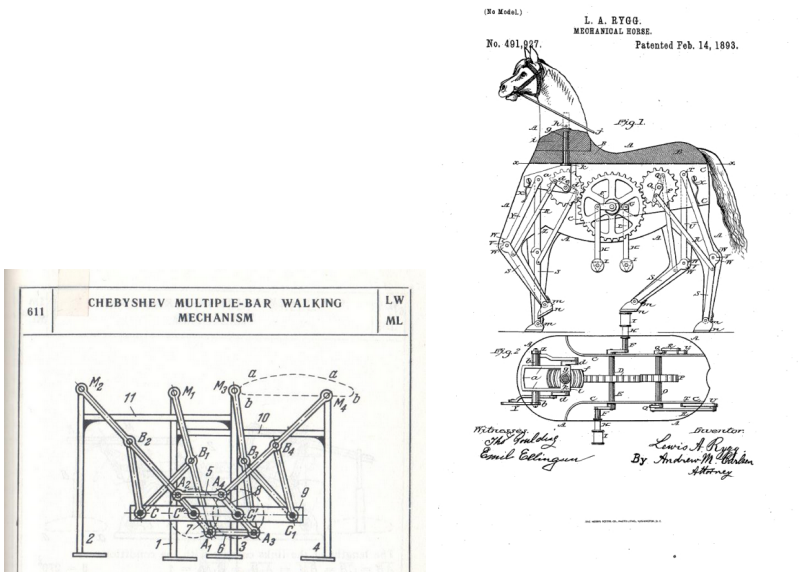
La principale différence avec ces derniers provient du nombre de degrés de liberté très élevé des robots disposant de pattes en comparaison avec le faible nombre de degrés de liberté des robots à roues/chenilles. Cela leur confère ainsi une liberté de mouvement plus grande. Ils ne sont pas limités à des terrains plats mais peuvent se déplacer sur des sols accidentés car leurs points de contact avec ceux-ci sont discrets, ils peuvent ainsi les choisir avec plus de flexibilité. Cela entraîne une logique de commande bien plus complexe, comme nous le verrons tout au long de ce travail. Certains robots marcheurs disposent aussi d'une perception tactile du sol grâce à des capteurs de toucher. Ce type de robot permet également d'endommager de façon minimale le sol, ce qui peut être recherché par exemple lors de l'exploitation forestière. Là où un robot à roues laisse une longue empreinte continue derrière lui, un robot à pattes ne laisse que quelques points de contact sur le sol.

Il existe aussi des inconvénients à ce genre de robots. Leur consommation en énergie est très importante du fait des nombreuses accélérations et décélérations survenant lors des déplacements de chaque membre. Ils ont généralement une faible charge utile. L'algorithme de contrôle est complexe vu le nombre de degrés de libertés qu'il faut coordonner, plusieurs processeurs sont ainsi souvent utilisés. Enfin, la vitesse de déplacement est faible.

Ce type de robot est donc particulièrement bien adapté à réaliser des missions en terrain accidenté où la mobilité est un point clé. Plusieurs exemples existent dont un projet européen, TELEMAN [2] ayant vu le jour dans les années 90 et encourageant le développement de ce type de robot pour la maintenance et la détection d'accident sur les sites nucléaires.

L'histoire des machines à pattes n'est pas aussi récente que l'on pourrait se l'imaginer. En témoignent plusieurs machines ayant été inventées depuis le dix-neuvième siècle. Par exemple la machine de Chebyshev [3] fut créée en 1850 par le mathématicien du même nom. Elle disposait de quatre pattes liées et synchronisées entre elles avec des liaisons en forme de barres (figure 2.1(a)). Un brevet déposé pour un cheval mécanique se déplaçant grâce à la force humaine a lui aussi été déposé en 1893 [4] (figure 2.1(b)).

Les robots à pattes de cette époque ne disposaient pas de contrôleurs et ne devaient leur déplacement qu'aux principes mécaniques directement mis en place dans leur conception. Il a fallu attendre la fin des années soixante avant de commencer à voir des robots recourant à des techniques de contrôle plus avancées.



(a) (b)

FIGURE 2.1 – (a) Machine de Chebyshev (1850) [3] (b) Cheval mécanique (1893) [4]

Le "General Electric Walking Truck" (1969) en est un exemple [5] (figure 2.2). Il était entièrement contrôlé par un humain à l'aide de valves hydrauliques et était l'un des premiers robots à implémenter un feedback de force à l'utilisateur.

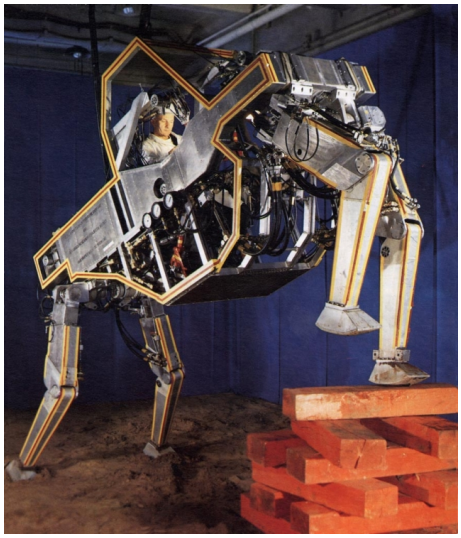
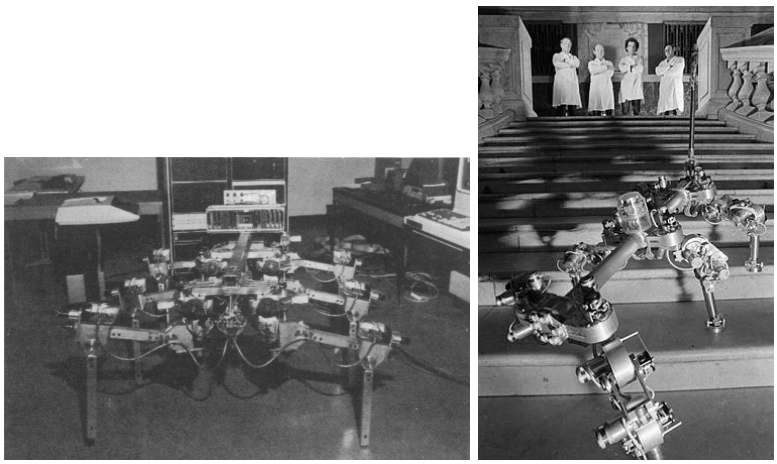


FIGURE 2.2 – General Electric Walking Truck [5]

Néanmoins, ce n'est qu'à partir des années septante que des travaux de recherche conséquents se sont vraiment mis en place pour les robots marcheurs. Cela a été permis par les progrès en informatique, permettant enfin de doter les robots de contrôleurs et de tester des stratégies de contrôle complexes.

Le tout premier robot à pattes doté d'un contrôle numérique fut l'"Ohio State University Hexapod"

(1976). C'est sur ce robot qu'ont été testés de nombreux algorithmes de contrôle nouveaux qui ont posé les bases de la recherche dans ce domaine comme le premier algorithme de free gait [6] et un contrôle de force [7]. Il avait 6 pattes munies chacune de 3 articulations et pesait 100 kg (figure 2.3(a)). Les Etats-Unis n'étaient pas les seuls à développer des robots marcheurs, durant la même période des chercheurs russes ont aussi créé un hexapode : Masha (aussi en 1976, figure 2.3(b)). Lui aussi a été utilisé pour tester un contrôle de force [8].



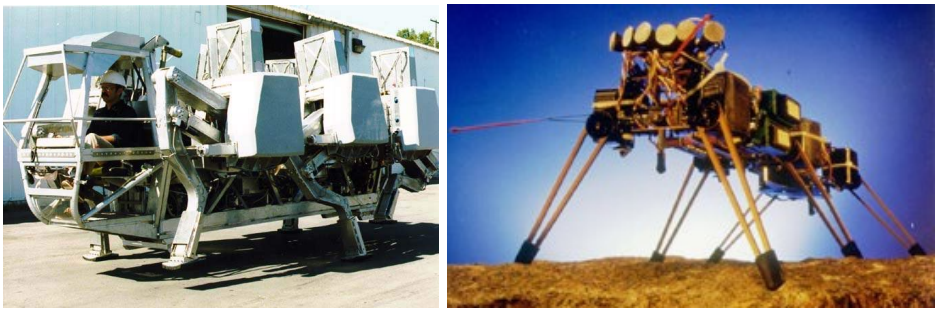
(a) (b)

FIGURE 2.3 – (a) OSU hexapod [6] (b) Masha [8]

Ces robots ont permis au domaine de recherche des robots marcheurs de se développer et de plus en plus d'autres équipes de recherche en ont créé au cours des années. Quelques-uns sont présentés ici.

L'Ohio State University a ainsi développé un autre robot marcheur plus imposant en 1989, l'ASV (Adaptive Suspension vehicle) [9]. Celui-ci était muni de 6 pattes mais était beaucoup plus grand que l'OSU hexapod : 5 m pour 2600 kg. Il pouvait ainsi transporter un opérateur et une charge allant jusque 225 kg, le tout étant alimenté par un moteur à combustion. Son autre particularité était l'utilisation d'un mécanisme à pantographe pour les pattes, permettant le découplage des mouvements horizontaux et verticaux avec des actionneurs linéaires. Nous pouvons voir cela à la figure 2.4(a).

D'autres universités américaines ont aussi développé leurs propres robots, comme au MIT par exemple. Leur laboratoire de robotique mobile a créé plusieurs robots dont les plus remarquables sont Genghis (1989), Hannibal et Attila (1990). Genghis (figure 2.4(b)) était un robot à six pattes dont chacune avait deux degrés de liberté. Il comportait un nombre important de capteurs : détection de contact avec le sol, paire de moustaches détectant les obstacles, cellules infrarouges et inclinomètres. Sa grande particularité était l'utilisation d'une stratégie de contrôle décentralisée sur plusieurs niveaux, appelée "subsumption architecture" qui profitait d'un fort couplage entre les capteurs et les actionneurs dans un bas niveau ce qui permettait de réaliser des actions plus complexes dans un haut niveau en associant les niveaux plus bas [10]. La même architecture a ainsi pu être testée sur Hannibal et Attila (figure 2.4(c) et (d)), qui constituent des versions plus évoluées du projet initial. Un contrôle robuste a ainsi permis de les faire marcher sur des terrains accidentés dans le but de les utiliser pour réaliser l'exploration d'autres planètes [11].

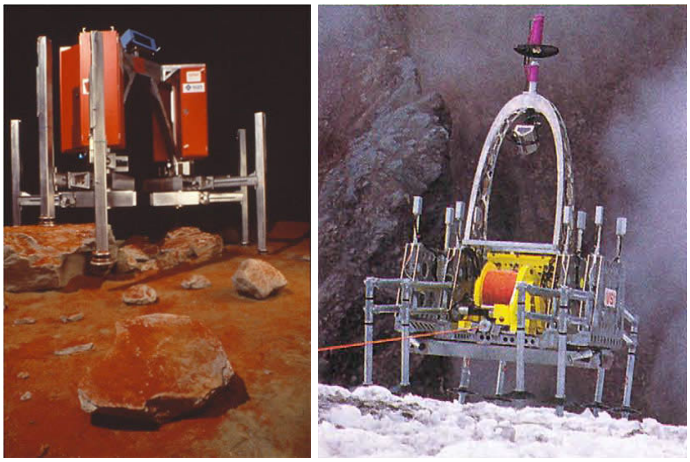


(a) (b)

(c) (d)

FIGURE 2.4 – (a) ASV [12] (b) Genghis [13] (c) Hannibal [14] (d) Attila [14]

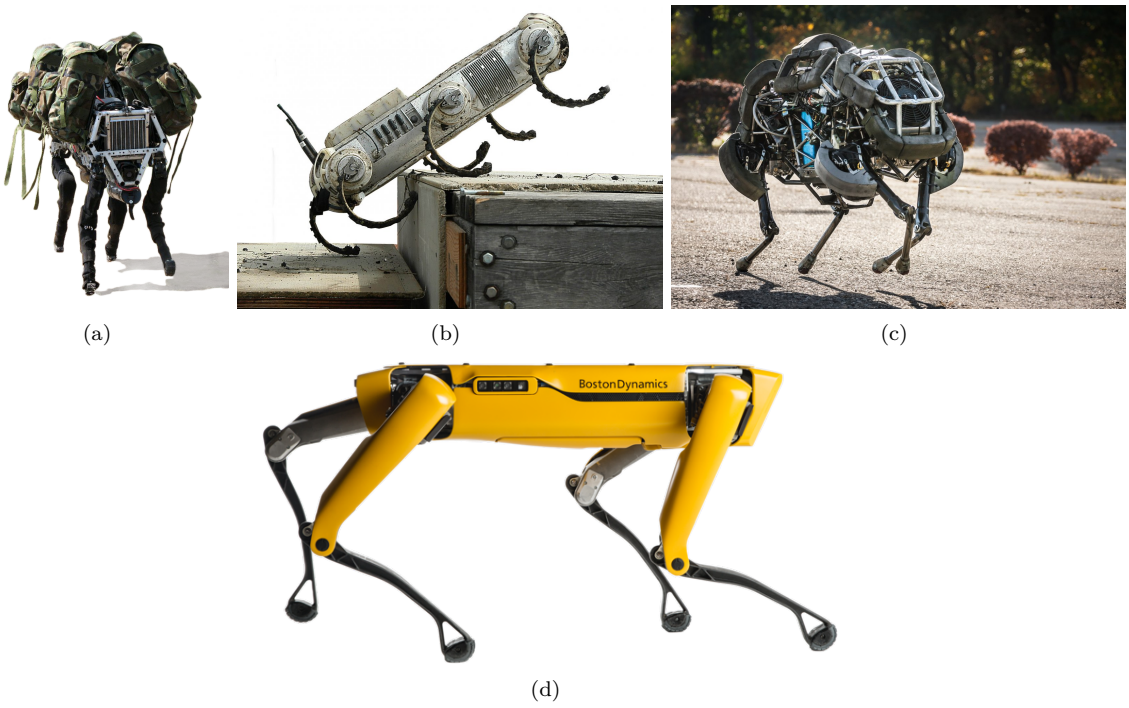
L'Université Carnegie Mellon (toujours aux Etats-Unis) a elle aussi inventé des robots dans un but d'exploration spatiale. Nous pouvons ainsi citer Ambler (1990) et Dante II (1994). Le premier est un robot pesant 2000 kg et pouvant porter jusqu'à 1000 kg (figure 2.5(a)). Il a été développé en vue d'une mission sur la planète Mars. Sa particularité réside en la configuration des pattes : elles sont groupées par trois autour de deux axes verticaux. Cela permet ainsi au robot de rester très stable lors de ses différentes démarches [15]. Le deuxième (figure 2.5(b)) dispose de huit pattes et a été utilisé pour explorer le cratère d'un volcan en Alaska, le Mount Spurr [16].



(a) (b)

FIGURE 2.5 – (a) Ambler [17] (b) Dante II [16]

Il serait impensable de réaliser un état de l'art sur les robots marcheurs sans mentionner les robots provenant de chez Boston Dynamics [18]. Ce sont actuellement les robots les plus connus par le grand public et les plus impressionants. On peut ainsi citer Big Dog (2004), leur premier robot qui était capable de grimper des pentes et de naviguer sur terrain accidenté en portant jusqu'à 150 kg (figure 2.6(a)). Ou encore RHEX (2007, figure 2.6(b)), ne disposant que d'un seul actionneur par patte. Il y a aussi Wild Cat (2013, figure 2.6(c)) qui est le robot quadrupède le plus rapide au monde, pouvant atteindre 32 km/h. Enfin, leur dernière création est Spot (2019), un robot pouvant monter des escaliers et réaliser de multiples tâches grâce à ses possibilités de customisation poussées (figure 2.6(d)).



(a) (b) (c)

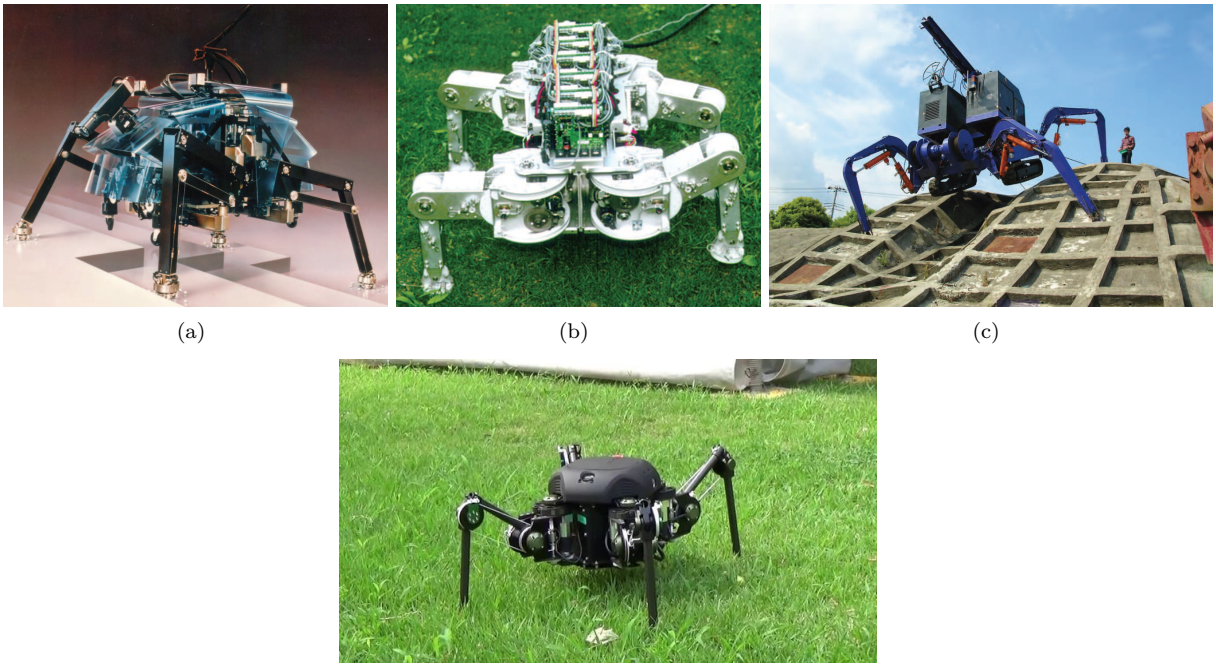
(d)

FIGURE 2.6 – (a) Big Dog [18] (b) RHEX [18] (c) Wild Cat [18] (d) Spot [18]

Les robots de Boston Dynamics sont donc commercialisés. Un autre robot marcheur en vente ayant un grand succès auprès des particuliers comme des universités est le PhantomX AX Metal Hexapod Mark III (figure 2.7). Ce robot provient de la société Trossen Robotics et a été mis en vente en 2018. C'est un hexapode où chaque patte comporte trois degrés de liberté [19]. Nous le mentionnons ici car le robot étudié dans ce travail a beaucoup de similarités avec ce dernier (même type de moteurs, architecture similaire).

Les robots marcheurs sont donc surtout développés aux Etats-Unis. Mais il en existe quand même dans d'autres pays du monde. Le Japon n'est ainsi pas en reste avec la série de robots quadrupèdes Titan développés depuis 1980 par "The Tokyo Institute of Technology" dont le laboratoire de robotique est dirigé par S. Hirose. On peut voir quelques exemples à la figure 2.8 [20]. Ces robots diffèrent par leur types d'actionneurs : certains sont munis d'actionneurs hydrauliques, d'autres par des systèmes de poulies et de fils et d'autres par de simples moteurs électriques.

FIGURE 2.7 – PhantomX AX Metal Hexapod Mark III [19]



(a) (b) (c)

(d)

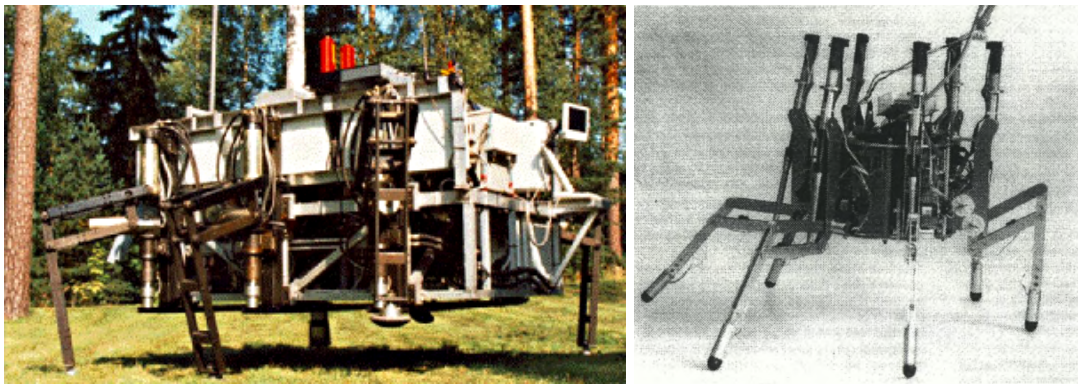
FIGURE 2.8 – (a) Titan IV [20] (b) Titan VIII [20] (c) Titan XI [20] (d) Titan XIII [20]

En Allemagne, plusieurs équipes de recherches ont développés des robots marcheurs. On peut citer Lauron V (2014), dernier en date de la série de robots du même nom provenant de "FZI Karlsruhe" [21]. C'est un hexapode possédant quatre degrés de liberté par jambe. Il peut se mouvoir sur des terrains accidentés et

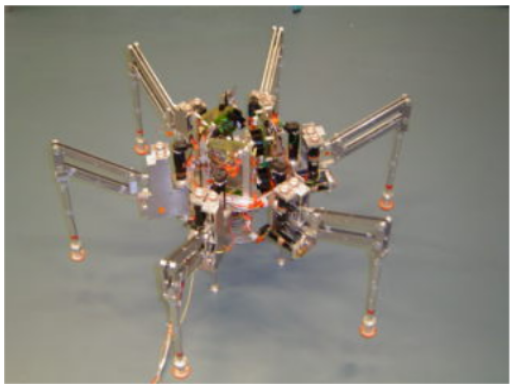
des pentes ainsi que saisir des objets (figure 2.9(a)). Il y a aussi Scorpion (2001), de l'université de Bremen (figure 2.9(b)). C'est un robot à huit pattes dont le contrôle est inspiré par le biomimétisme [22].

(a) (b)

FIGURE 2.9 – (a) Lauron V [21] (b) Scorpion [22]



(a) (b)



(c)

FIGURE 2.10 – (a) Mecant [23] (b) Silex [1] (c) AMRU [24]

En Finlande, au "Automation Technology Laboratory of Helsinki University", des algorithmes de "free gait" et d'adaptation au terrain ont été développés sur le robot Mecant (1994) [25]. Ce robot disposait de

six pattes et était hydraulique (figure 2.10(a)). Il a été développé pour l'exploitation forestière.

En Belgique, il y a aussi eu quelques études sur le sujet. Nous pouvons citer le robot Silex de l'Université Libre de Bruxelles et AMRU, un robot provenant de l'Ecole Royale Militaire et ayant aussi été étudié à l'UMONS. Les deux utilisent un mécanisme à pantographe et sont munis de six jambes. Le premier a été mis à profit principalement pour l'étude d'un algorithme de "free gait" [1] alors que le deuxième a été utilisé pour un contrôle "neuro-fuzzy" adaptatif et à des fins de simulation [24] [26]. Ils sont visibles à la figure 2.10.

Nous avons ainsi présenté un panel de robots marcheurs, des plus anciens aux plus récents. Il n'a bien évidemment pas été possible de lister tous les robots marcheurs existants, mais ceux présentés dans cette section permettent de se faire une bonne idée de l'évolution de la technique dans ce domaine et des références importantes existant en la matière.

2.2 Gecko Skin

Au cours des dernières années, de plus en plus de chercheurs se sont intéressés aux propriétés des pattes des geckos. Leurs pattes leur permettent de s'accrocher à la plupart des surfaces et de se déplacer en même temps sur celles-ci. Cette adhérence est rendue possible par une structure hiérarchique composée de setae (poils microscopiques) de 20 µm se séparant ensuite en spatulae d'une largeur de 200 nm [27]. Celles-ci sont visibles à la figure 2.11. Les spatulae exercent ainsi des interactions de Van Der Waals permettant au gecko d'adhérer à la surface.

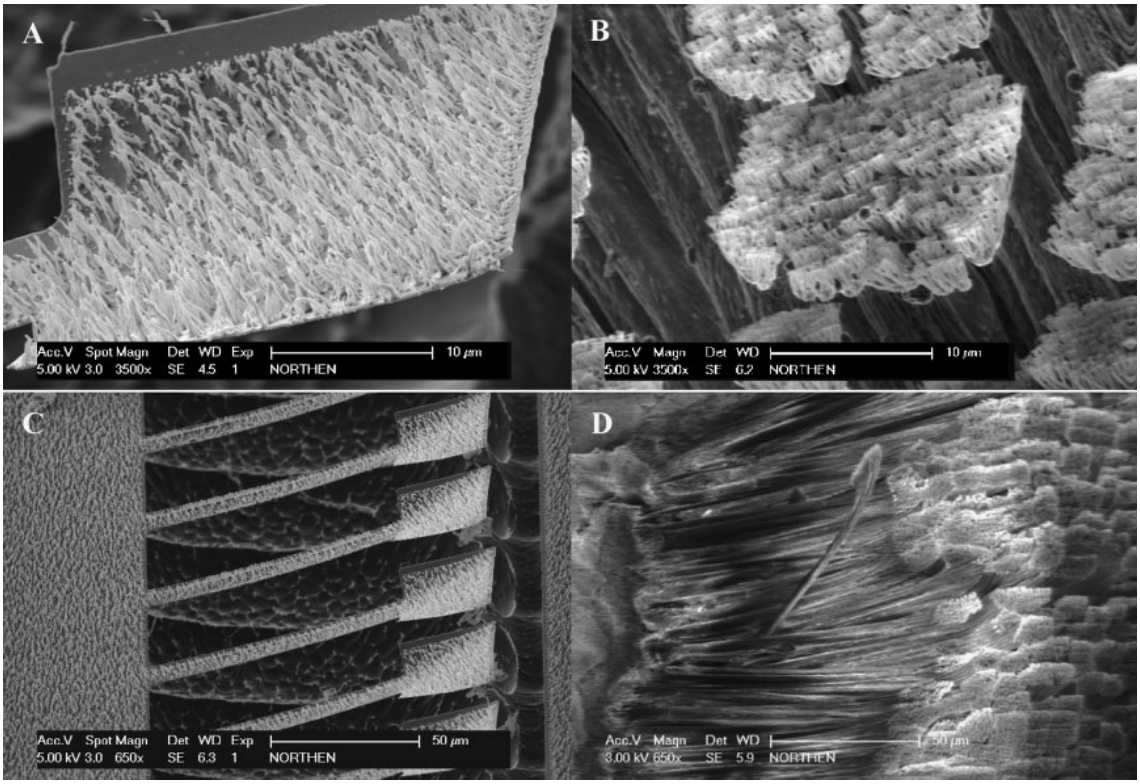


FIGURE 2.11 – Spatulae (B) et setae (D) d'un gecko ainsi que le matériau développé (A et C) [27]

Ce type de matériau présente un intérêt croissant qui est dû à certaines de ses propriétés. [28] nous explique ainsi que jusqu'ici, l'on avait recours à des adhésifs plus traditionnels sensibles à la pression afin de pouvoir adhérer à des surfaces de types différents, mais cela pose plusieurs problèmes : difficultés à décoller l'adhésif de la surface en question, pas de possibilités de réutiliser le matériau et dégâts sur la surface. La

Gecko Skin permet d'avoir un matériau réutilisable, facilement nettoyable et surtout facilement décollable de la surface grâce à sa structure hiérarchique anisotropique.

Plusieurs équipes se sont ainsi inspirées de ces structures afin de concevoir des matériaux synthétiques imitant leurs propriétés. Une revue générale de ce qui a été fait et les principales techniques employées pour fabriquer les matériaux sont présentées ici.

Certaines structures ont été pensées et réalisées pour ressembler à la structure hiérarchique des pattes du gecko. C'est le cas du matériau développé par [27]. Celui-ci est présent à la figure 2.11. Il permet l'adhésion mais aussi le retrait grâce à l'utilisation d'un champ magnétique permettant de réorienter des microstructures en nickel. La fabrication a été faite à l'aide de la lithographie.

D'autres structures se veulent plus symétriques et ne suivant pas de hiérarchie au niveau de l'échelle. On peut alors en discerner plusieurs types, en fonction du pas séparant les piliers, la hauteur de ces derniers et aussi leur forme. Des exemples sont repris à la figure 2.12.

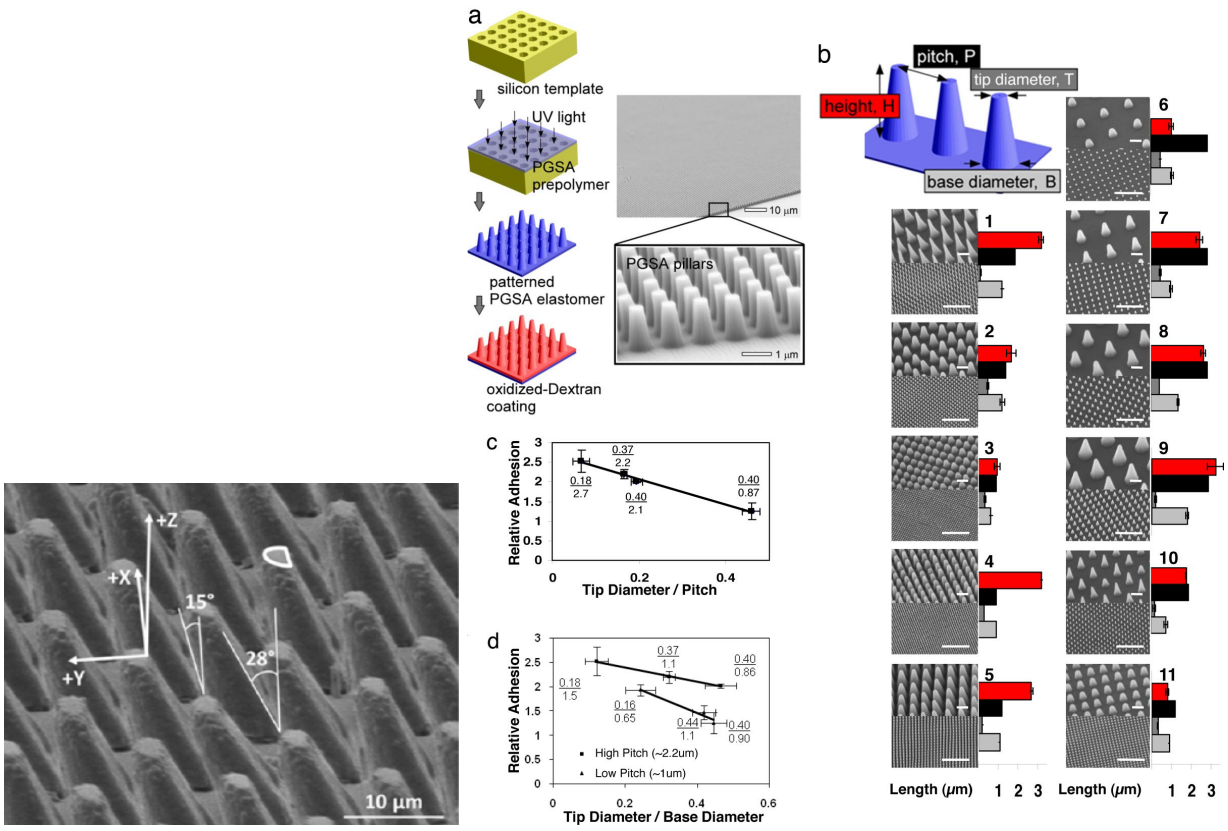


FIGURE 2.12 – (a) Vue microscopique du matériau de [29] (b) Vue microscopique du matériau de [30] ainsi que sa méthode de fabrication

Il existe des matériaux similaires mais présentant des surfaces adhésives des deux côtés (figure 2.13 et 2.14).

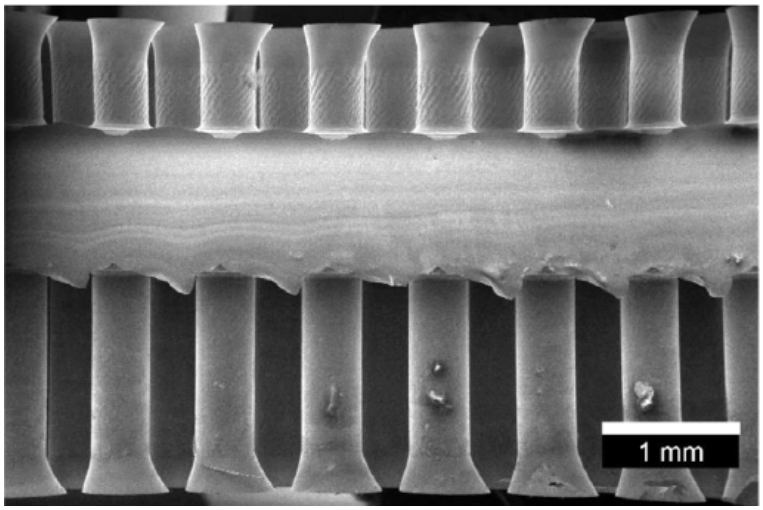


FIGURE 2.13 – Vue microscopique du matériau de [31], avec des piliers sur deux surfaces

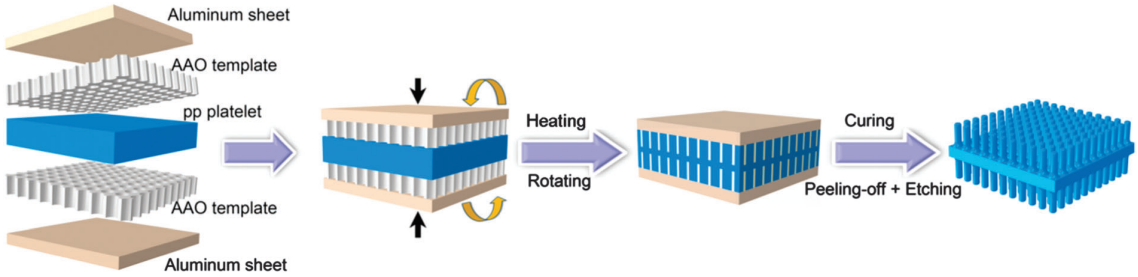


FIGURE 2.14 – Méthode de fabrication du matériau de [32], avec des piliers sur deux surfaces

La plupart de ces matériaux sont donc créés à partir de polymères versés dans un moule. Le moule est quant à lui fabriqué là aussi avec la lithographie. Il en résulte un matériau prenant la forme de minuscules piliers.

D'autres approches existent, comme ce matériau réalisé de la même manière mais incorporant en plus une matrice rigide en fibres de carbone dans un pad en élastomère souple (figure 2.15).

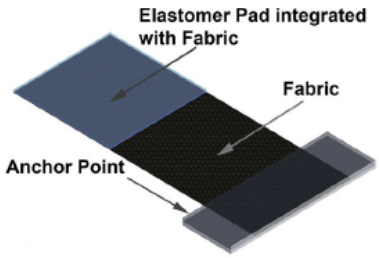


FIGURE 2.15 – Adhésif avec fibres de carbone développé par [28]

Quelques robots grimpeurs ont été développés en utilisant le type de matériaux énoncés précédemment. Un premier exemple est Stickybot (figure 2.16). Ce robot a pour but d'essayer de reproduire un gecko montant sur un mur. Le bout des pattes est ainsi recouvert d'un matériau présentant une structure hiérarchique

présentée auparavant, ils sont adhésifs dans une direction et peuvent se détacher si l'on tire dans l'autre direction.

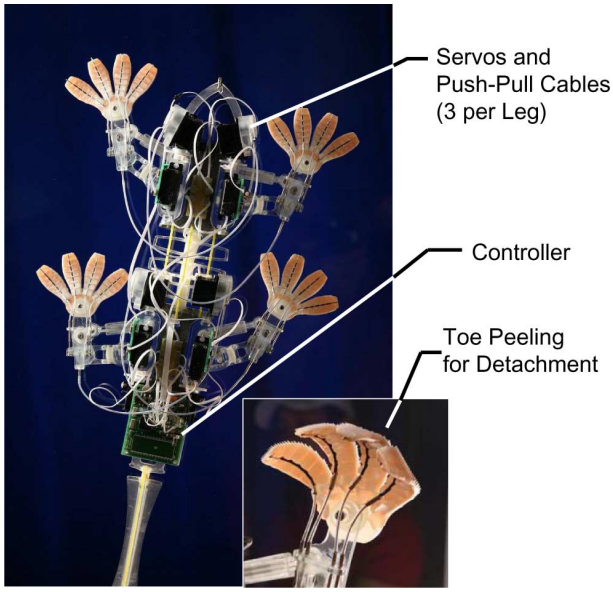


FIGURE 2.16 – Stickybot développé par [33]

Un autre robot est l'Acrobot (figure 2.17) qui a été développé dans le but d'être utilisé à bord de l'ISS dans un environnement sans gravité et dans des endroits étroits. Il utilise aussi des pads adhésifs pouvant être mis en état OFF ou ON en fonction de la direction de la charge appliquée.

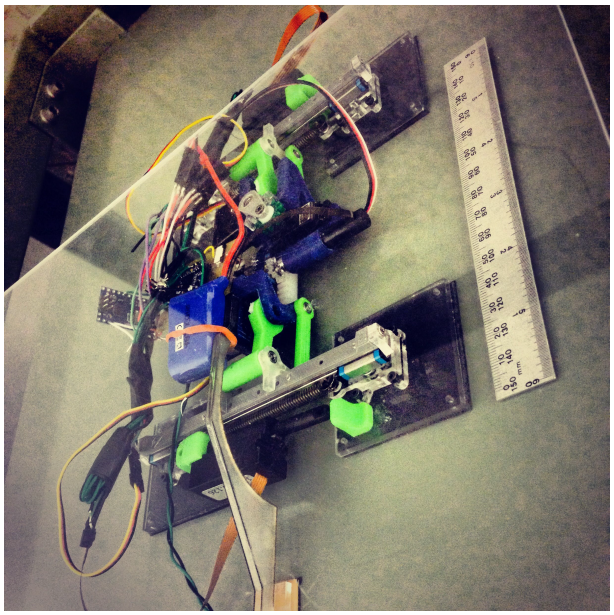


FIGURE 2.17 – Acrobot développé par [34]

Ce type de matériau commence à être commercialisé au niveau industriel mais aussi pour les particuliers.

Deux entreprises proposent ce type de produit, GeckSkin [35] et INNOCISE. Le produit d'INNOCISE appelé Gecomer [36] est visible à la figure 2.18. Les applications vont du post-it réutilisable en Gecko Skin aux manipulateurs robotiques utilisant ce matériau pour des applications Pick and Place.

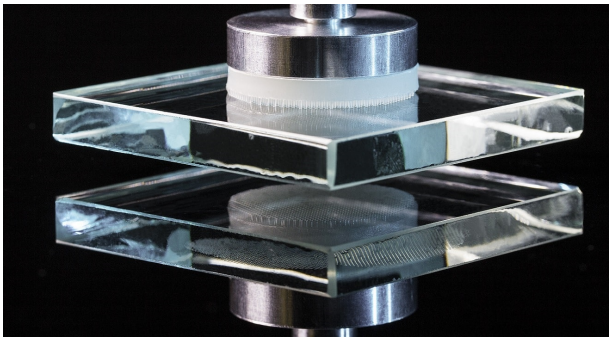


FIGURE 2.18 – Gecomer [36]

Finalement, certaines personnes ont tentés de réaliser des pads adhésifs utilisant le même principe mais de manière DIY. Un bon exemple est cette vidéo Youtube [37]. Dans celle-ci, on utilise des feuilles de réseau de diffraction comme moule (disponibles à bas prix) que l'on recouvre de silicone. C'est cette dernière méthode que nous aurions dû explorer dans ce travail afin de pouvoir l'appliquer à notre robot, mais cela n'a malheureusement pas été possible à cause de la pandémie de COVID-19.

Chapitre 3

Le robot hexapode

Le robot étudié dans ce travail est un robot à six pattes ayant été utilisé pour tourner dans le film Eye On Juliet (2017) [38]. On peut voir celui-ci à la figure 3.1. Ce robot n'a pas été commercialisé et a été créé juste pour le film. La Polytechnique de Montréal en a ainsi racheté un exemplaire mais sans avoir accès à ses plans.

FIGURE 3.1 – Le robot hexapode de Eye On Juliet

Ce chapitre va passer en revue le robot et les pièces le composant. On expliquera ainsi plusieurs aspects liés au contrôleur, aux moteurs et à l'alimentation.

3.1 Description du robot

Le robot est constitué d'un corps muni de six pattes, ainsi que d'une tête pouvant tourner par rapport au corps. Chaque patte présente quatre moteurs et la tête en a deux. Cette dernière peut se mouvoir en rotation selon deux directions alors que les pattes ont chacune quatre degrés de liberté (figure 3.2). La dénomination de chaque membre de la patte est notée sur la figure (coxa, fémur, tibia et tarsus).

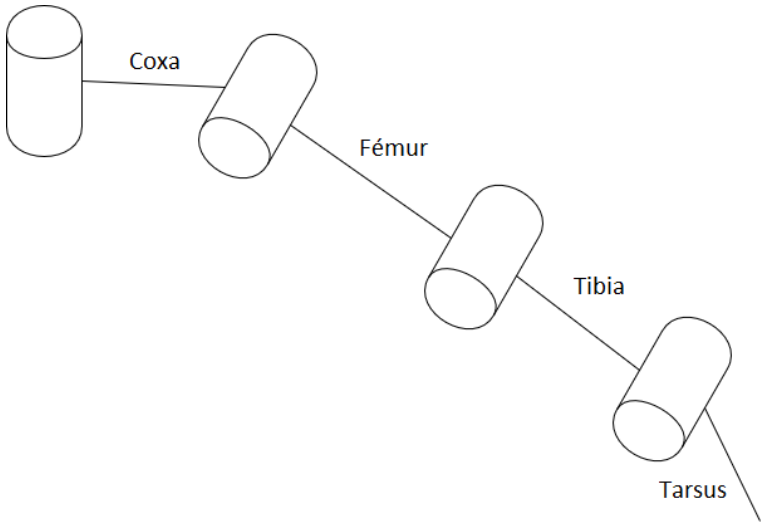


FIGURE 3.2 – Schéma cinématique d'une patte avec 4 degrés de liberté

Il pèse environ une dizaine de kilos et a une empreinte au sol d'approximativement 500 mm sur 500 mm. Les moteurs utilisés sont des servomoteurs haut de gamme MX106T et MX64T. Le contrôleur quant à lui est un SBC (single-board computer) l'Odroid-C1 [39]. Une autre carte est présente sur le robot, l'Arbotix Pro [40], permettant de communiquer avec les moteurs. Une photo des deux cartes est présente à la figure 3.3.

FIGURE 3.3 – Cartes présentes initialement sur le robot

(a) (b) (c)

FIGURE 3.4 – (a) Le servomoteur MX106T [42] (b) Câble utilisé [44] (c) Power hub de Trossen Robotics [45]

3.2 Servomoteurs

Les servomoteurs utilisés dans le robot proviennent de chez Robotis [41]. Il y a donc 24 MX106T présents dans les pattes du robot (4 par pattes) et 2 MX64T pour sa tête. Ce sont les mêmes types de servomoteurs si ce n'est que les MX64T ont un couple maximum plus petit que les MX106T. On peut voir ce type de moteur à la figure 3.4(a).

Ces servomoteurs disposent tous d'un microcontrôleur (ARM CORTEX-M3) permettant la communication avec un contrôleur principal. Ils sont munis d'encodeurs absolus sans contact. Chaque servomoteur peut aussi déterminer sa vitesse, sa position, sa température, son voltage et sa charge en temps réel. Ils sont munis d'un PID intégré permettant le contrôle en position du moteur. Ces moteurs nécessitent une alimentation sous 12V. Plus d'informations sont disponibles sur le manuel du fabricant disponible en ligne [42][43]. Un résumé de celui-ci est présent dans les annexes.

Le protocole de communication utilisé par ces servomoteurs et leur alimentation rendent leur utilisation très aisée. Il suffit de les connecter entre eux à l'aide de groupes de 3 câbles (un pour la communication, un pour l'alimentation VCC et un pour le ground). Les 4 moteurs présents dans chaque patte du robot sont ainsi reliés entre eux en chaîne par le type de câble présent à la figure 3.4(b). Les six groupes de pattes sont ensuite reliés à l'alimentation et au contrôleur à travers deux "power hubs" tels que celui à la figure 3.4(c).

Tout cela permet donc facilement d'alimenter et de communiquer avec les servomoteurs à travers un bus unique. Les positions des servomoteurs peuvent être actualisées en envoyant une seule instruction sur le bus, ce qui donne la possibilité de les faire se mouvoir tous au même moment.

3.3 Contrôleur

Le contrôleur principal utilisé est l'Odroid-C1. C'est à partir de celui-ci que l'algorithme du robot s'exécute. Il envoie ainsi des commandes au moteur en passant par l'Arbotix Pro qui joue le rôle d'intermédiaire entre les moteurs et le contrôleur.

L'Odroid dispose d'un environnement Linux permettant une assez grande liberté pour choisir le langage utilisé pour coder l'algorithme. Le contrôleur est équipé d'un connecteur Wi-Fi permettant de communiquer facilement avec celui-ci depuis un PC.

3.4 Alimentation

Les moteurs et le contrôleur sont alimentés par batterie vu que le robot est mobile et doit pouvoir se déplacer de manière autonome. On a choisi un type de batterie recommandé pour travailler avec ce type de moteur, une batterie Li-Po (Lithium-Polymère). Une photo de la batterie utilisée est représentée à la figure 3.5.

FIGURE 3.5 – Batterie Li-Po utilisée [46]

Chapitre 4

Simulation du robot

Ce travail aurait dû être réalisé entièrement sur le robot décrit au chapitre précédent. Nous aurions ainsi pu le programmer et effectuer des tests dessus, dans le laboratoire de robotique de la Polytechnique de Montréal. Malheureusement, à cause de la pandémie de COVID-19, il n'a pas été possible de continuer à travailler sur le robot car la Polytechnique de Montréal a fermé. Cela s'est passé un mois et demi après le début du travail sur le robot. Il a donc été décidé de simuler ce dernier sur ordinateur afin de pouvoir continuer le TFE. Ce chapitre décrit la démarche suivie.

4.1 Logiciels utilisés

Il existe plusieurs logiciels sur le marché permettant de faire de la simulation de robots. Existent ainsi par exemple Arctin, ARS, Gazebo, Morse,... [47]. Notre choix s'est porté sur le logiciel V-REP car il était gratuit et permettait de choisir entre 4 moteurs physiques (à savoir ODE, Bullet, Vortex ou Newton). Il est aussi bon de noter que V-REP a récemment été renommé CoppeliaSim dans la dernière mise à jour du logiciel.

On a aussi utilisé un logiciel de CAD classique afin de modéliser le plus fidèlement possible le robot : Solidworks.

4.2 Modélisation 3D

N'ayant plus accès au robot afin de prendre des mesures exactes de celui-ci, nous avons fait au mieux afin de le recréer sous Solidworks sans plans. Il a quand même été possible de mener à bien cela car les modèles CAD des moteurs sont disponibles en ligne et des mesures avaient été effectuées pour les distances séparant les moteurs d'une jambe et les distances séparant les points d'ancrage des jambes par rapport au corps du robot. Une jambe ainsi que le corps du robot ont pu être modélisés sous Solidworks (figure 4.1).

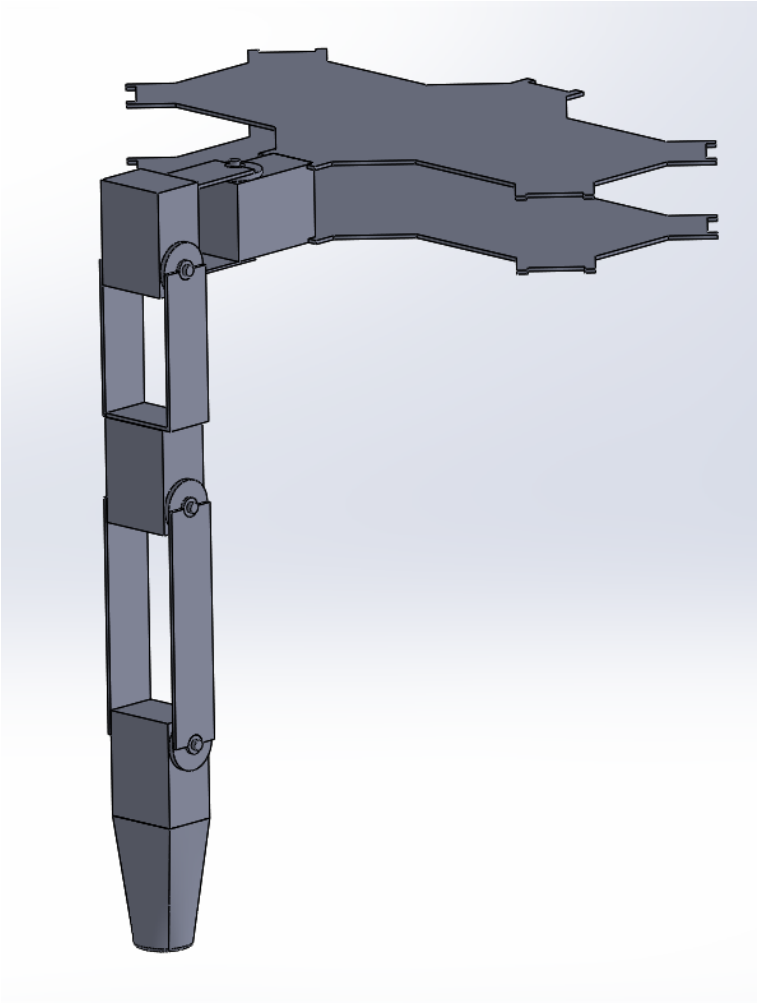


FIGURE 4.1 – Modélisation d'une jambe et du corps sous Solidworks

Nous n'avons modélisé qu'une seule jambe car vu qu'elles sont toutes identiques, il a été plus facile de n'en implémenter qu'une seule sur CoppeliaSim et de la copier/coller plusieurs fois aux autres endroits. Cette modélisation respecte ainsi les distances mesurées sur le robot pour ce qui est de la longueur des membres la composant (à lire en partant du corps du robot vers l'extrémité de la patte) : $L_{coxa} = 62$ mm, $L_{fémur} = 125$ mm, $L_{tibia} = 125$ mm et $L_{tarsus} = 110$ mm. Le parti a été pris de ne pas modéliser la partie supérieure du robot comprenant les deux moteurs MX64T car nous n'avions pas pris de mesures sur cela. De plus, la partie supérieure peut facilement être démontée et ne contribue pas à la démarche du robot.

4.3 Implémentation dans CoppeliaSim

Le modèle a ensuite été importé sous fichier STL sur CoppeliaSim. Nous nous sommes alors inspiré d'un tutoriel existant sur le site de CoppeliaSim afin de construire notre robot [48]. Cela consistait d'abord à placer les liaisons rotoïdes sur les arbres de chaque moteur. On associait ensuite aux formes importées des formes de géométrie plus simple qui seraient utilisées lors de la partie dynamique de la simulation. Cela est réalisé afin de minimiser la complexité des calculs lors de cette dernière. Il restait alors à associer les différents objets selon une chaîne parent/enfant et à copier/coller la jambe réalisée aux 5 autres endroits nécessaires.

Un script a alors été associé au robot permettant son contrôle. Le robot est présent à la figure 4.2 ainsi que la hiérarchie des différents éléments sur la gauche de la figure. Son équivalent "dynamique" réalisé avec des pièces à géométrie plus simple est représenté à la figure 4.3.

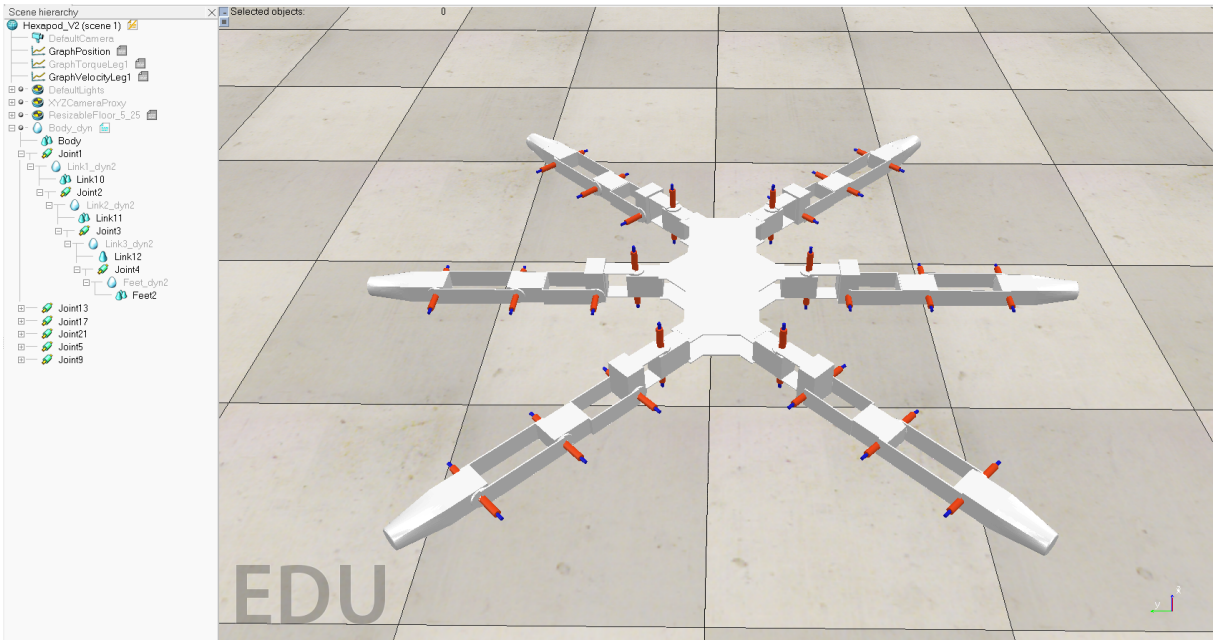


FIGURE 4.2 – Robot simulé sous CoppeliaSim

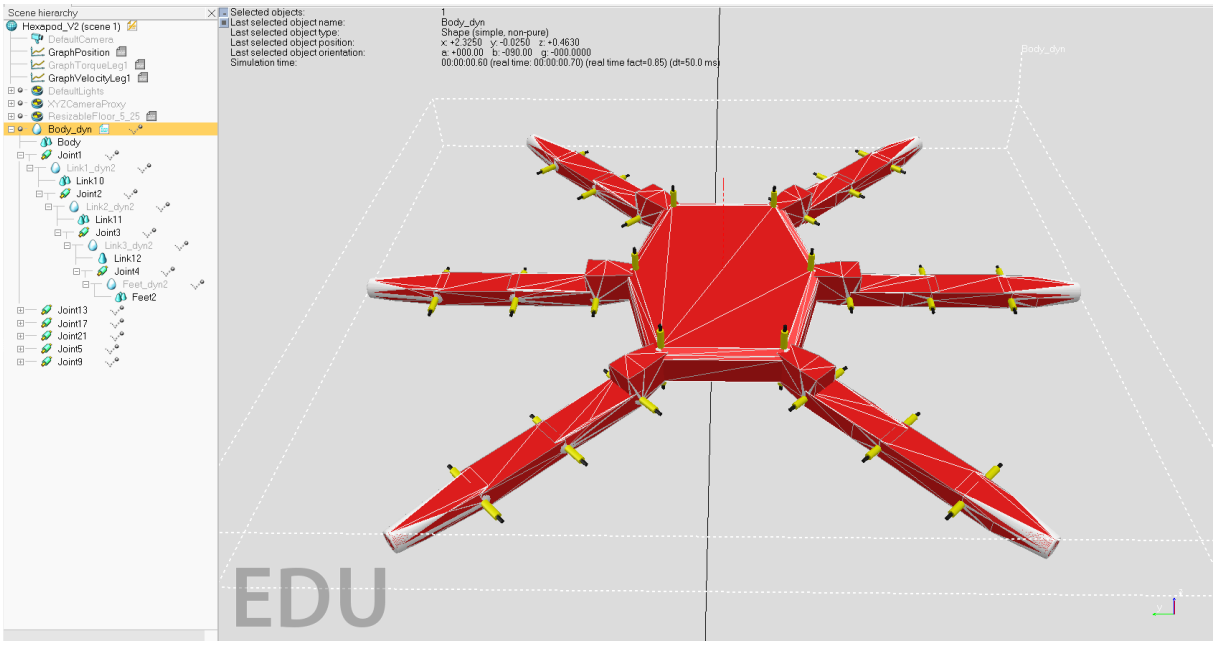


FIGURE 4.3 – Robot simulé sous CoppeliaSim - pièces à géométries simples

4.4 Moteurs physiques

CoppeliaSim permet l'utilisation de quatre moteurs physiques (figure 4.4). Nous allons les décrire et justifier celui qui a été choisi [49] :

FIGURE 4.4 – Moteurs physiques disponibles [49]

- Bullet : c'est un moteur open source permettant la détection de collisions et de la dynamique des corps rigides. Il a été développé pour les jeux vidéos et les effets visuels de films.
- ODE : similaire à Bullet, il apporte les mêmes fonctionnalités et est lui aussi utilisé dans les jeux vidéos.
- Vortex Studio : c'est un moteur commercialisé par CM Labs et qui propose des simulations physiques de haute qualité. Un avantage de Vortex Studio est qu'il propose des propriétés physiques dans des unités physiques, contrairement aux autres moteurs où ce n'est pas toujours le cas. Il est utilisé dans des applications industrielles et dans la recherche.
- Newton : similaire à Bullet et ODE, sauf qu'il possède un solveur déterministique et non basé sur des méthodes itératives. Il est aussi utilisé pour les jeux vidéos.

Une autre information utile pour faire le choix d'un moteur est de voir les propriétés d'un objet quelconque modifiables dans CoppeliaSim en fonction du moteur choisi. Un exemple est ainsi donné à la figure 4.5.

On a ici différentes propriétés relatives au contact entre objets comme la friction, l'adhérence, l'amortissement. Comme nous simulons un robot hexapode dont les pattes sont en contact permanent avec le sol, il est important de choisir un moteur physique permettant de simuler le contact précisément. Il est clair que Vortex Studio présente le plus grand nombre de paramètres réglables comparé aux autres moteurs.

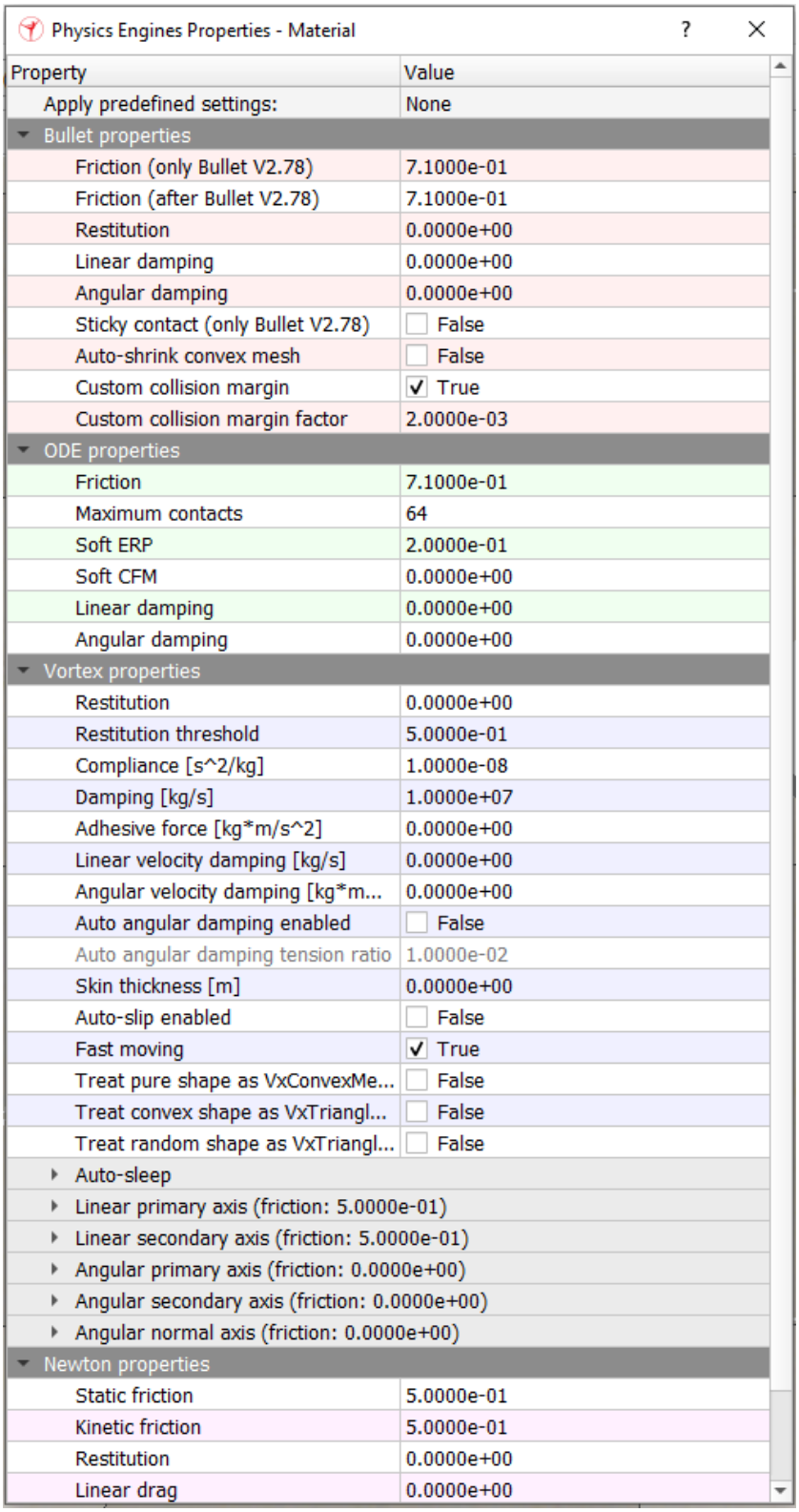
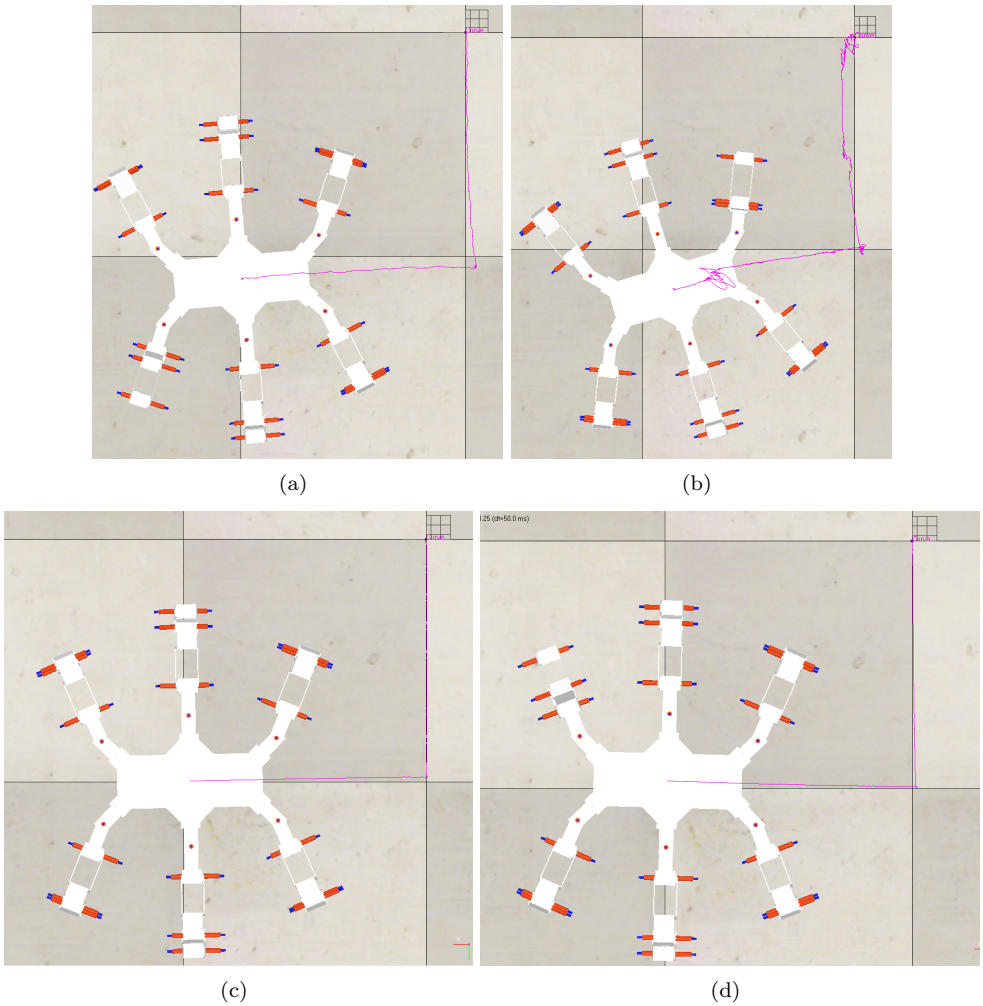


FIGURE 4.5 – Exemple de propriétés modifiables dans CoppeliaSim en fonction du moteur physique choisi



(a) (b)

(c) (d)

FIGURE 4.6 – (a) Trajectoire avec Bullet (b) Trajectoire avec ODE (c) Trajectoire avec Vortex Studio (d) Trajectoire avec Newton

Notre choix s'est donc porté sur Vortex Studio car outre son nombre important de propriétés pouvant être modifiées, il est aussi utilisé dans des applications industrielles et de recherche, contrairement aux autres moteurs. Nous avons évidemment expérimenté avec les différents moteurs afin de se rendre compte de leurs différences, et là aussi Vortex Studio présentait de meilleurs résultats (figure 4.6, la même trajectoire a été imposée au robot avec les quatre moteurs physiques, à savoir une ligne droite, une rotation de 90 degrés et une autre ligne droite). Ce moteur a donc été retenu pour notre simulation.

Chapitre 5

Contrôle et algorithme du robot

Afin de permettre à notre robot de se déplacer, il nous a fallu créer un algorithme permettant à celui-ci de coordonner ses pattes afin de savoir où et quand les placer. Le problème a ainsi été divisé en deux : d'abord une partie de l'algorithme génère la démarche de l'hexapode, c'est-à-dire où il doit placer ses pattes et en fonction de quelle séquence. Ensuite, l'autre partie de l'algorithme traduit ces positions voulues en paramètres de configuration permettant de positionner les moteurs aux positions angulaires nécessaires.

5.1 Génération de démarche

Avant de décrire l'algorithme utilisé afin de générer les démarches de l'hexapode, il faut d'abord expliquer la manière dont celui-ci se déplace. Ce qui permet au robot de se mouvoir dans une direction donnée, c'est le mouvement des pattes qui est divisé en deux phases : une phase de support et une phase de transfert (figure 5.1). Les deux points extrêmes de cette trajectoire sont appelés AEP et PEP (Anterior Extreme Position et Posterior Extreme Position) [1]. Pendant la phase de support, la patte recule dans une direction opposée à la direction de déplacement de l'hexapode, permettant à ce dernier de s'appuyer sur le sol en avançant. Pendant la phase de transfert, la patte s'élève du sol pour aller chercher un nouveau point de contact en avant. La trajectoire de la phase de support doit être une ligne droite pour un mouvement rectiligne, alors que la trajectoire de transfert peut être choisie librement. La trajectoire choisie a la forme d'une ellipse entre le PEP et l'AEP.

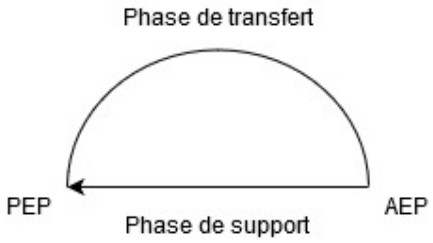


FIGURE 5.1 – Mouvement d'une patte

La durée de la phase de transfert d'une patte est notée t et celle de la phase de support s. Le temps d'un cycle total est donc de $T = s + t$. Un facteur de service $\beta = \frac{s}{T}$ peut être défini, facteur sur lequel nous allons pouvoir jouer afin d'imposer différentes démarches au robot [1].

Les pattes effectuent ainsi ces deux phases dans une séquence dépendant de la démarche choisie pour l'hexapode. Deux démarches ont été implémentées : un wave gait et un tripod gait. L'algorithme utilisé est le même pour les deux démarches, il suffit simplement de choisir les états initiaux (support ou transfert) des pattes afin d'obtenir la démarche voulue ainsi que d'imposer un $\beta = \frac{5}{6}$ pour le wave gait et un $\beta = \frac{1}{2}$ pour le tripod gait. Le wave gait fait que les pattes réalisent leur phase de transfert l'une après l'autre tandis que le tripod gait permet à trois pattes de réaliser leur phase de transfert en même temps. Les séquences représentant ces deux démarches sont schématisées aux figures 5.2 et 5.3.

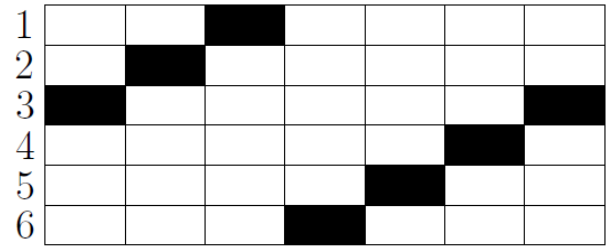


FIGURE 5.2 – Wave gait : les rectangles noirs représentent la phase de transfert [50]

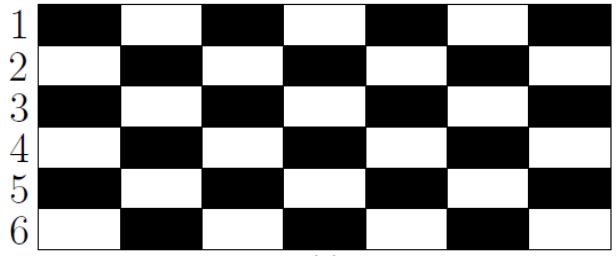


FIGURE 5.3 – Tripod gait : les rectangles noirs représentent la phase de transfert [50]

Il existe un lien entre les positions imposées par la génération de démarche (données dans le système de référence du corps de l'hexapode) et le calcul des positions angulaires des moteurs par la cinématique inverse. Cette étape est réalisée grâce à l'utilisation des matrices de transformations homogènes. Nous travaillons ainsi dans le repère du corps de l'hexapode (figure 5.4) lorsque l'on impose la position d'une patte $\{\bar{r}_{P/0}\}_0$, position qui est ensuite transformée par la matrice de transformation homogène correspondante $T_{0,i}$ selon la formule [51] :

$$\begin{pmatrix} \{\bar{r}_{P/0}\}_0 \\ 1 \end{pmatrix} = T_{0,i} \cdot \begin{pmatrix} \{\bar{r}_{P/i}\}_i \\ 1 \end{pmatrix}$$

ce qui permet d'obtenir la position $\{\bar{r}_{P/i}\}_i$ dans le repère de la patte et d'effectuer la cinématique inverse avec celle-ci. Le repère de la patte est situé sur la première rotoïde de la patte, toujours avec l'axe des x pointé vers l'extrémité de celle-ci. Les matrices $T_{0,i}$ sont ainsi tabulées et inversées à l'avance dans le code, ce qui permet d'avoir la position à imposer dans le système de référence de la patte choisie avant d'effectuer la cinématique inverse. Ces matrices sont reprises en annexe.

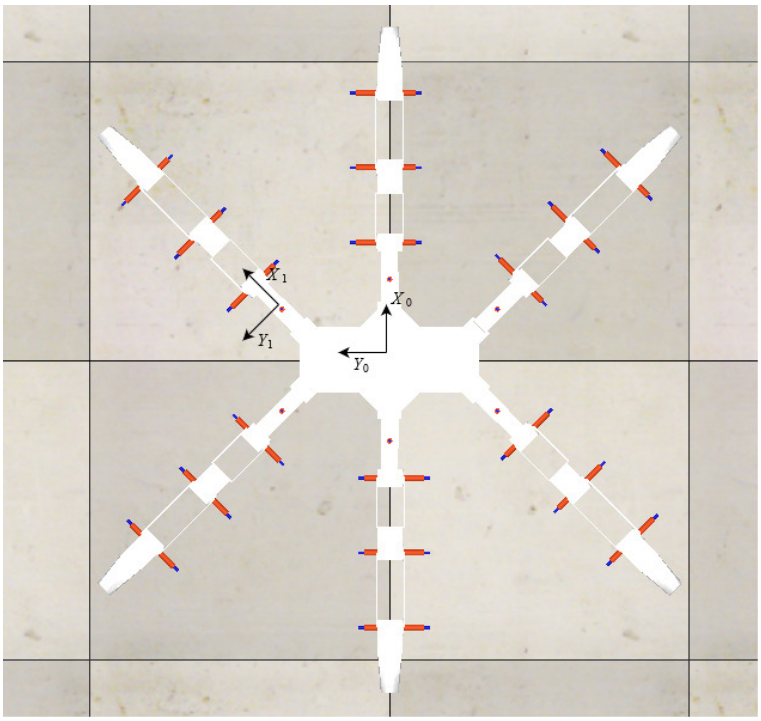


FIGURE 5.4 – Vue du dessus de l'hexapode : deux systèmes d'axes de référence, celui du corps de l'hexapode et celui d'une patte

5.2 Cinématique inverse d'une jambe

Un problème de cinématique inverse est un problème classique en robotique. Il consiste à déterminer les paramètres de configuration (ici q_1, q_2, q_3, q_4) du système pour que l'extrémité de celui-ci se place à une position X, Y, Z dans l'espace. Nous partons donc de ces coordonnées dans l'espace pour, dans notre cas, spécifier la position angulaire de nos moteurs.

Pour résoudre un problème de cinématique inverse, deux types de méthodes existent : les méthodes de résolution exactes et les méthodes de résolution numériques. Une méthode de résolution exacte a été choisie ici car elle est moins demandante en temps de calcul qu'une méthode numérique. Résoudre des équations déjà calculées à l'avance est en effet bien plus rapide qu'utiliser une méthode itérative bouclant plusieurs fois. Comme notre robot doit effectuer la cinématique inverse à une fréquence assez élevée (voir section 5.3), il est important de minimiser le temps de calcul du processeur.

Une jambe du robot comporte donc quatre degrés de liberté. Il est facile de trouver dans la littérature une résolution exacte pour le même problème à 3 degrés de liberté [52][53][54][55], mais un parti pris doit être choisi pour notre quatrième degré de liberté vu qu'il existe un trop grand nombre de solutions possibles si l'on ajoute celui-ci. On va donc résoudre le problème pour seulement 3 degrés de liberté et l'on reviendra sur le quatrième à la fin.

Vu la configuration de ceux-ci, le problème se décompose dans deux plans afin de déterminer d'abord le premier paramètre de configuration. On regarde la jambe vue du dessus dans le plan X, Y (figure 5.5).

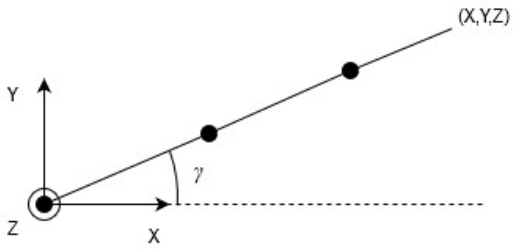


FIGURE 5.5 – Plan X, Y

L'angle γ représenté est notre premier paramètre de configuration. Nous pouvons écrire :

$$tan(\gamma) = \frac{Y}{X}$$

$$q_1 = \gamma = arctan(\frac{Y}{X})$$

Notre premier paramètre de configuration est défini et ne dépend que de la position voulue en X et en Y.

Nous nous plaçons maintenant dans le plan de la jambe (figure 5.6). Les trois parties de la jambe ont été désignées par les noms qu'on leur attribue généralement dans la littérature ; le coxa, le fémur et le tibia (le quatrième membre non représenté pour l'instant est le tarsus).

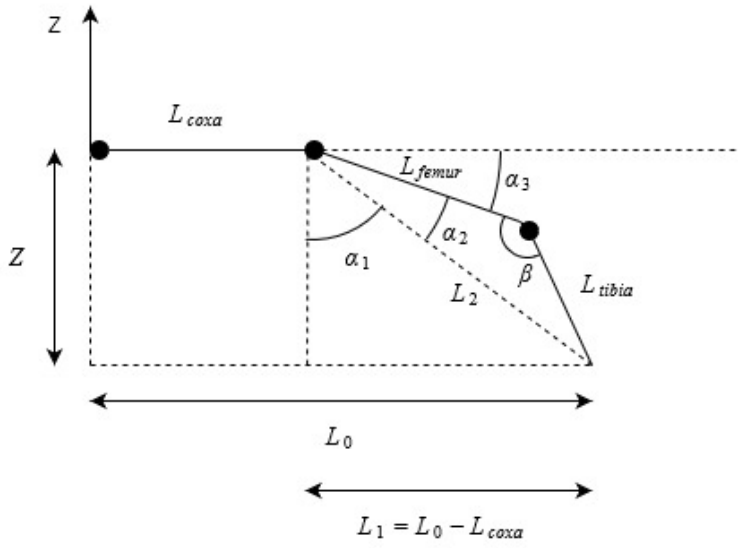


FIGURE 5.6 – Plan de la jambe

Comme nous sommes dans le plan de la jambe, L_0 est calculé à partir des coordonnées X et Y :

$$L_0 = \sqrt{X^2 + Y^2}$$

L_2 est ainsi déterminé :

$$L_2 = \sqrt{L_1^2 + Z^2}$$

Notre but est maintenant de trouver α_3, ce qui est réalisé en décomposant le problème en α_1 et α_2 :

$$\alpha_1 = arctan(\frac{L_1}{Z})$$

$$L_{tibia}^2 = L_2^2 + L_{femur}^2 - 2L_2 L_{femur} cos(\alpha_2)$$

$$\alpha_2 = arccos(\frac{L_{tibia}^2 - L_2^2 - L_{femur}^2}{-2L_2 L_{femur}})$$

Et donc :

$$q_2 = \alpha_3 = \frac{\pi}{2} - (\alpha_1 + \alpha_2)$$

Il nous reste à trouver le troisième paramètre de configuration. Il est directement déduit à partir de β :

$$\beta = arccos(\frac{L_2^2 - L_{femur}^2 - L_{tibia}^2}{-2L_{femur} L_{tibia}})$$

$$q_3 = \pi - \beta$$

Il est aussi bon de remarquer qu'il existe une autre solution symétrique à ce problème (figure 5.7). Les équations de q_2 et q_3 se modifient alors légèrement :

$$q_2 = \frac{\pi}{2} - (\alpha_1 + \alpha_2) + 2\alpha_2$$

$$q_3 = -(\pi - \beta)$$

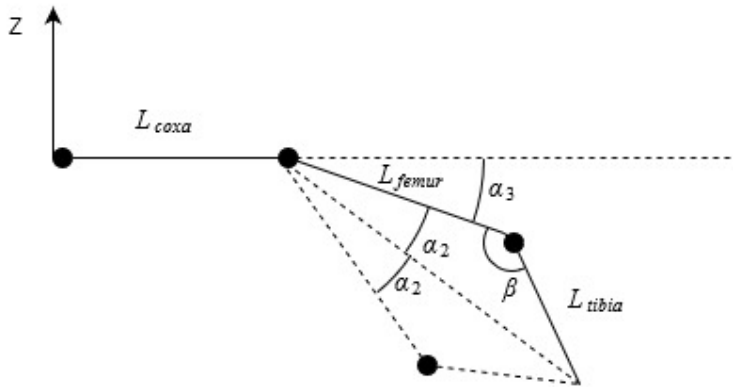


FIGURE 5.7 – Deuxième solution

Il paraît naturel de choisir la première solution dans le cas d'un robot marcheur comme le nôtre. Nous avons donc fait en sorte que la jambe se retrouve dans une telle configuration dans notre code.

Notre quatrième paramètre de configuration peut maintenant être imposé. Le choix de maintenir le tarsus (pied) de la jambe perpendiculaire au sol à tout moment a été fait. Cela permet ainsi un meilleur contact du pied du robot avec le sol.

Cela se traduit par une formule simple :

$$q_4 = \frac{\pi}{2} - q_2 - q_3$$

Il faut aussi modifier les coordonnées que l'on veut imposer en Z (pour prendre en compte la longueur rajoutée en Z du tarsus) :

$$Z_{new} = Z_{old} + L_{tarsus}$$

Nous disposons ainsi d'un algorithme permettant d'avoir les quatre paramètres de configuration de chaque moteur en fonction d'un point en coordonnées cartésiennes à imposer dans le système de référence de la jambe.

5.3 Considérations relatives au logiciel de simulation

L'algorithme qui a été développé permet d'envoyer les nouvelles positions voulues aux moteurs à une fréquence donnée. Par exemple, si la fréquence choisie est de 50 Hz, cela voudrait dire que des nouvelles positions seraient envoyées aux moteurs toutes les 20 ms. Cette fréquence dépend normalement du contrôleur utilisé et de sa puissance de calcul : la fréquence maximum est ainsi imposée par le temps avec lequel le contrôleur exécute le code. Si celui-ci exécute en 10 ms le code, la fréquence maximum à imposer est ainsi de 100 Hz.

Or, si cela est applicable dans la réalité, les choses sont un peu différentes avec un logiciel de simulation. CoppeliaSim fonctionne avec des pas de simulation, c'est-à-dire qu'un script principal s'exécute tous les pas de simulation (50 ms dans CoppeliaSim). Ce script se divise en quatre grandes catégories :

- L'initialisation : cette partie du script n'est exécutée qu'une seule fois au début de la simulation.
- L'actuation : cette partie est appelée à chaque pas de simulation. Elle s'occupe de la dynamique, des moteurs,...
- La détection : cette partie est appelée à chaque pas de simulation. Elle s'occupe des différents capteurs pouvant être présents dans la simulation.
- La restauration : cette partie du script n'est exécutée qu'une seule fois à la fin de la simulation. Elle restaure les configurations initiales de tous les objets présents dans la simulation.

Il paraît donc logique et intuitif de choisir le pas de simulation pour avoir notre fréquence d'actualisation des positions. Cela nous permettra d'écrire notre code dans un autre script, qui sera appelé à chaque pas de simulation par le script principal dans la partie actuation. Nous aurons ainsi une actualisation des positions toutes les 50 ms, soit à une fréquence de 20 Hz. De plus, cela ne change pas de beaucoup la logique d'implémentation sur un contrôleur réel, le code pourra ainsi être facilement transposable sur le vrai robot.

Il est aussi bon de noter que CoppeliaSim donne la possibilités d'écrire deux types de scripts : "threaded" et "non-threaded". Le deuxième type est un script simple qui est appelé par le script principal à chaque pas de simulation comme décrit auparavant. Le premier type est plus particulier, il agit comme une coroutine. C'est-à-dire qu'il s'exécute pendant un court laps de temps avant de rendre le contrôle au script principal ou un autre "threaded script". Cela lui permet de fonctionner "en parallèle" de tous les autre scripts. Nous en parlons ici car il a été choisi d'implémenter le contrôle de haut niveau sur un "threaded script", à savoir imposer les vitesses de déplacement du robot et le type de démarche à adopter. Cela permet d'écrire ce que l'on veut que le robot effectue comme trajectoire et pendant combien de temps facilement dans ce script. Les paramètres sont ensuite passés dans un "non-threaded script" où se trouve l'algorithme permettant de générer la démarche et la cinématique inverse qui lui est appelé tous les pas de simulation. Une représentation de cela est présentée à la figure 5.8.

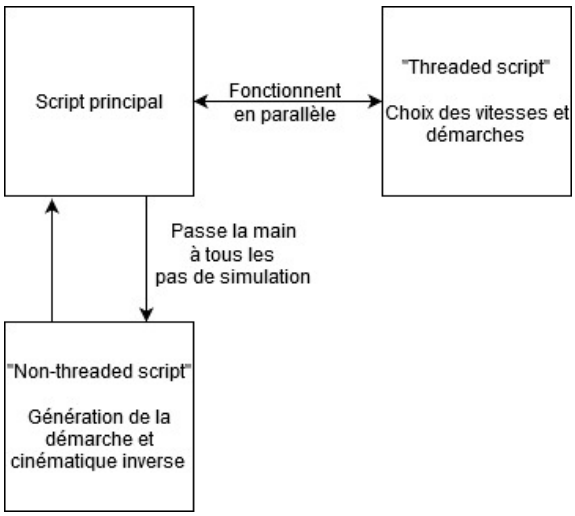


FIGURE 5.8 – Scripts utilisés par CoppeliaSim

5.4 Algorithme

La structure de l'algorithme est repris à la figure 5.9.

Celui-ci est divisé en deux parties, d'abord une étape d'initialisation suivie de la boucle principale. L'étape d'initialisation n'est donc appelée qu'une seule fois par le script principal au début de la simulation. Cela permet d'initialiser toutes les variables nécessaires ainsi que de placer l'hexapode dans sa position initiale en envoyant les commandes nécessaires aux moteurs. Certains paramètres y sont aussi réglables comme la hauteur du corps de l'hexapode, la hauteur de son pas et évidemment le temps de cycle choisi. Le temps nécessaire à la phase de support d'une patte est imposé et le temps de la phase de transfert est ainsi directement déduit :

$$t = s\frac{1-\beta}{\beta}$$

Il est aussi intéressant de noter que la configuration initiale des pattes du robot est un cercle centré sur le corps de ce dernier dont le rayon peut être réglé dans les paramètres. Ces positions sont donc celles que les jambes vont atteindre dès la fin de leur phase de transfert. Cela permet ainsi aux jambes de ne pas sortir de leur espace de travail en voulant effectuer un mouvement impossible car on les recentre à chaque fin de cycle.

La boucle principale est appelée à tous les pas de simulation par le script principal. On commence ainsi par récupérer les vitesses dictées par l'utilisateur. Ensuite, l'état de chaque patte est vérifié. Si la patte est en phase de support, on va calculer sa prochaine position. Cela est réalisé grâce à cette formule [50] :

$$P_i^{k+1} = R_z(\frac{\omega_z}{f}) \times (P_i^k - \frac{V}{f})$$

La vitesse voulue V normalisée par la fréquence f de la simulation est d'abord soustraite à la position actuelle. Le produit de la matrice de transformation homogène R_z et la nouvelle position calculée est ensuite effectué afin de pivoter selon l'axe z. Cette matrice de rotation utilise un angle de rotation $\frac{\omega_z}{f}$, car la vitesse de rotation est elle aussi normalisée par la fréquence.

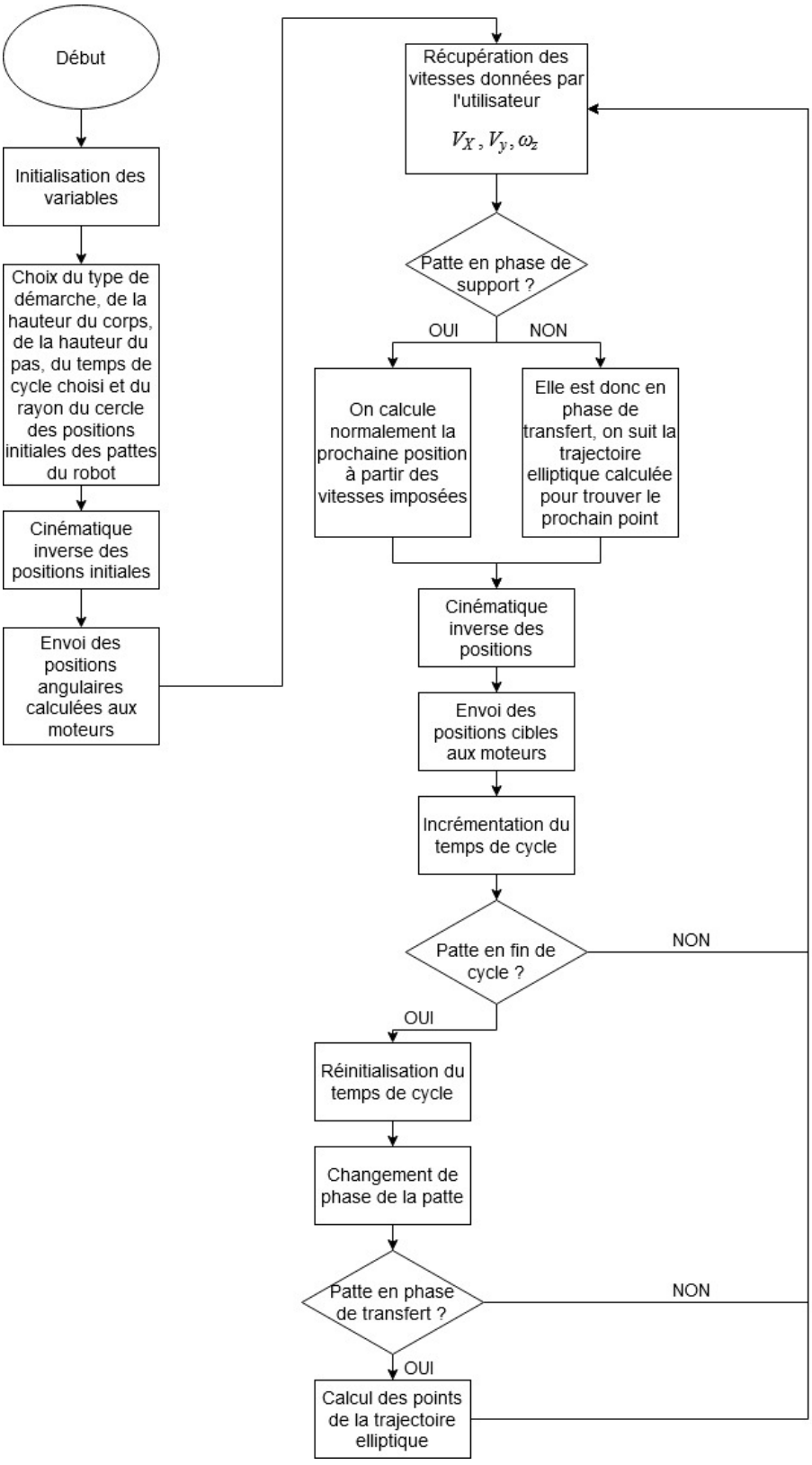


FIGURE 5.9 – Algorithme utilisé

Par contre, si la patte est en phase de transfert, sa position est tout simplement actualisée avec le prochain point de la trajectoire qu'elle doit suivre et qui est calculée d'un bloc (cela sera expliqué plus loin).

Les nouvelles positions de toutes les pattes ayant été calculées, on peut dès lors effectuer la cinématique inverse sur celles-ci afin de traduire cela en positions angulaires que l'on impose aux moteurs.

Une variable représentant le temps de cycle est alors incrémentée d'une quantité $\frac{1}{f}$. Cela permet de voir où nous nous trouvons temporellement dans le cycle ce qui sera utile pour imposer de façon synchrone les états de toutes les pattes. Une vérification est effectuée pour voir si des pattes se retrouvent en fin de cycle. Si oui, on active un "flag" exprimant qu'il faut faire passer ces pattes dans la phase suivante.

La variable du temps de cycle est remise à zéro si la phase de transfert a atteint son terme.

Les états des pattes concernées peuvent alors être modifiés comme expliqué précédemment. De plus, pour les pattes entrant en phase de transfert, les trajectoires elliptiques qu'elles devront suivre à partir de leur position actuelle pour arriver à la position initiale et les replacer au centre de la zone de travail sont calculées. La trajectoire elliptique est ainsi constituée d'un nombre de points fonction du temps t de la phase de transfert et de la fréquence f : $n = t \times f$. Il ne reste plus à la patte qu'à suivre cette série de points jusqu'à sa phase de support. La hauteur de l'ellipse est un paramètre réglable et correspond à la hauteur d'un pas que réalise l'hexapode.

L'algorithme revient ensuite au début de la boucle principale pour un nouveau calcul des positions.

Chapitre 6

Tests et résultats

Ce chapitre expose et décrit les tests que nous avons effectué à l'aide de la simulation. Une première section présente d'abord un exemple du type de trajectoire que notre hexapode peut accomplir. Nous nous sommes ensuite principalement concentrés sur l'aspect énergétique lié à la puissance mécanique appliquée sur les liaisons du robot. Il n'est malheureusement pas possible de simuler directement un moteur électrique à l'aide de sa fonction de transfert et de ses constantes classiques sur CoppeliaSim. Le logiciel permet d'appliquer sur chaque liaison un couple et une vitesse maximale. Lors d'un contrôle en position de chaque liaison comme on le fait ici, on peut alors utiliser un PID prenant en compte l'erreur sur la position afin de contrôler la vitesse de la liaison et cela à chaque pas de simulation. Le couple maximum est quant à lui gardé constant. Cela ne correspond donc pas exactement à la réalité et à nos moteurs utilisés, mais cela apporte quand même une information sur la puissance appliquée aux liaisons ce qui nous permet de faire des comparaisons entre les paramètres du robot, ses démarches et les situations dans lesquelles il se trouve.

Un couple maximal de 5 Nm et une vitesse maximale de 45 tr/min ont été imposés à nos liaisons afin d'essayer de rester cohérent avec nos moteurs (MX106T). Les paramètres PID choisis sont les suivants : $P = 0,2$, $I = 0,05$ et $D = 0$.

6.1 Exemple de trajectoire et vidéos

Notre algorithme permet à notre robot de se déplacer sur un sol plat, dans plusieurs directions. Il peut ainsi se déplacer selon X ou Y et aussi tourner autour de Z. Cela lui permet de réaliser des trajectoires comme celle reprise à la figure 6.1.

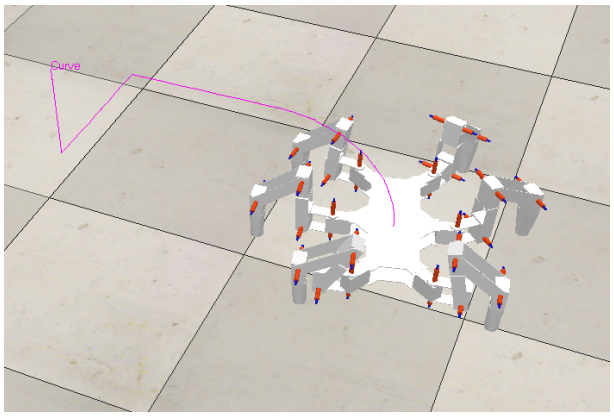


FIGURE 6.1 – Exemple de trajectoire réalisée par le robot

En guise d'illustration, on peut retrouver deux vidéos de notre robot aux adresses suivantes (l'une avec un tripod gait et l'autre avec un wave gait) :

— Tripod gait : `https://youtu.be/A2j3-6F_7K4`

— Wave gait : `https://youtu.be/vpdVf7q5l8A`

6.2 Démarche sur le sol

Cette section présente et interprète les graphes des couples, vitesses et puissances provenant d'un déplacement classique de l'hexapode. Celui-ci se déplace vers l'avant à une vitesse de 10 mm/s. Deux démarches ont été testées : le tripod gait et le wave gait. Le même temps de cycle total a été imposé pour ces deux démarches, six secondes. Les autres paramètres choisis étaient : une hauteur du corps de l'hexapode de 200 mm, une hauteur de pas de 100 mm et un rayon de 350 mm pour le positionnement initial des pattes sur un cercle. Les couples et vitesses de chaque liaison ont été relevés en fonction du temps ce qui nous a permis de calculer la puissance appliquée sur celles-ci, le tout sur un cycle (figures 6.2, 6.3 et 6.4).

Plusieurs conclusions intéressantes liées à l'observation de ces graphes peuvent être avancées. Tout d'abord, comme la démarche choisie est tripode, les pattes se lèvent par groupe de trois (pattes 1-3-5 et pattes 2-4-6). La phase de transfert dure trois secondes et la phase de support aussi trois secondes. La phase de transfert est celle durant laquelle les vitesses sont élevées et varient : de 16 à 19 secondes pour le premier groupe de pattes et de 19 à 22 secondes pour le deuxième. La puissance mesurée est ainsi élevée lors de la phase de transfert de la patte et quasiment nulle lors de la phase de support. Cela s'explique par le fait que la patte se déplace selon une trajectoire elliptique qui met en oeuvre tous les moteurs de la patte en phase de transfert, alors qu'en phase de support, seul le premier moteur est vraiment sollicité et les autres bougent à une vitesse moindre. Malgré cela, les couples sont plus élevés lors de la phase de support que lors de la phase de transfert : cela s'explique simplement parce que les pattes doivent supporter le poids du robot lors de la phase de support.

Le moteur le plus sollicité est le moteur numéro 2. Il reprend ainsi le plus d'efforts lors de la phase de support et atteint les plus hautes vitesses lors de la phase de transfert. C'est aussi lui qui consomme le plus de puissance lors de la phase de transfert. Lors de la phase de support, le moteur consommant le plus de puissance est le premier moteur.

Les graphes mesurés lors d'un wave gait sont visibles aux figures 6.5, 6.6 et 6.7. Dans ce cas-ci, c'est toujours le moteur 2 qui est le plus sollicité. La puissance est plus élevée lors de la phase de transfert d'un wave gait que lors d'un tripod gait de même temps de cycle. Cela n'est pas dû aux couples développés mais bien aux vitesses qui sont plus importantes lors d'un wave gait : cela s'explique par le fait qu'une patte en phase de transfert ne dispose que d'une seule seconde pour parcourir la distance parcourue en phase de support en 5 secondes dans la direction opposée afin de se replacer à sa position initiale. Les vitesses nécessaires sont donc plus importantes que lors d'un tripod gait où les phases de support et de transfert durent 3 secondes chacune.

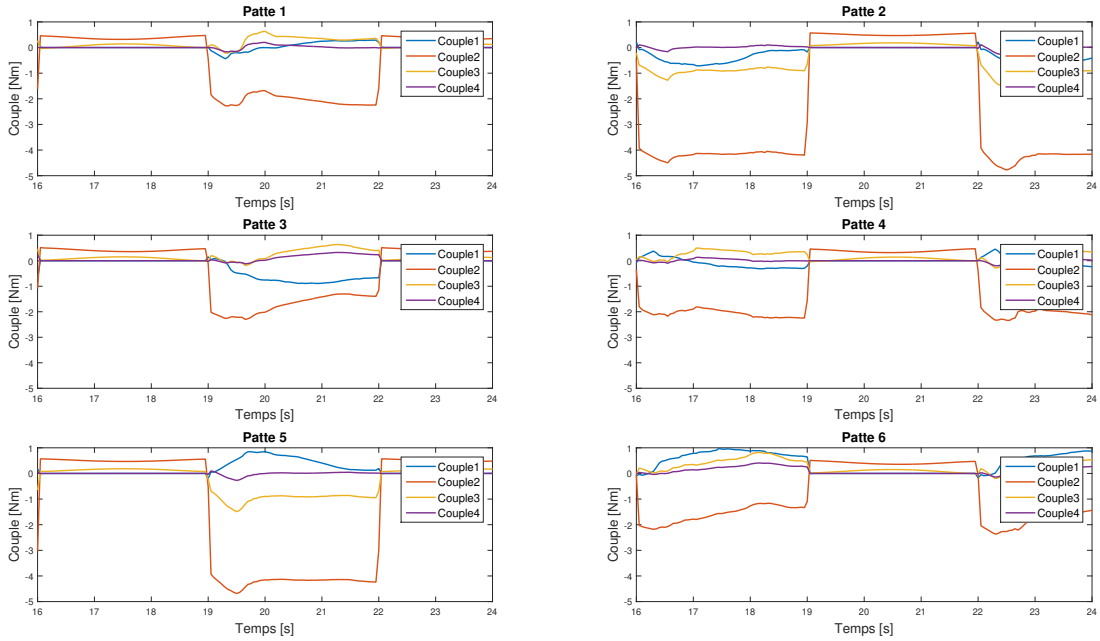


FIGURE 6.2 – Couples mesurés lors d'un tripod gait

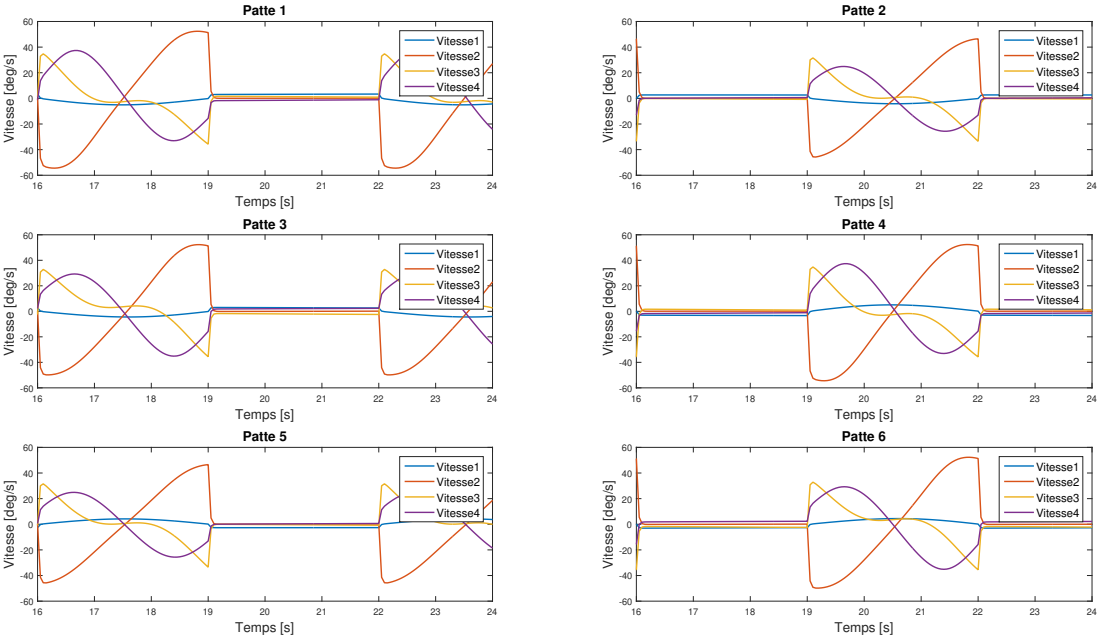


FIGURE 6.3 – Vitesses mesurées lors d'un tripod gait

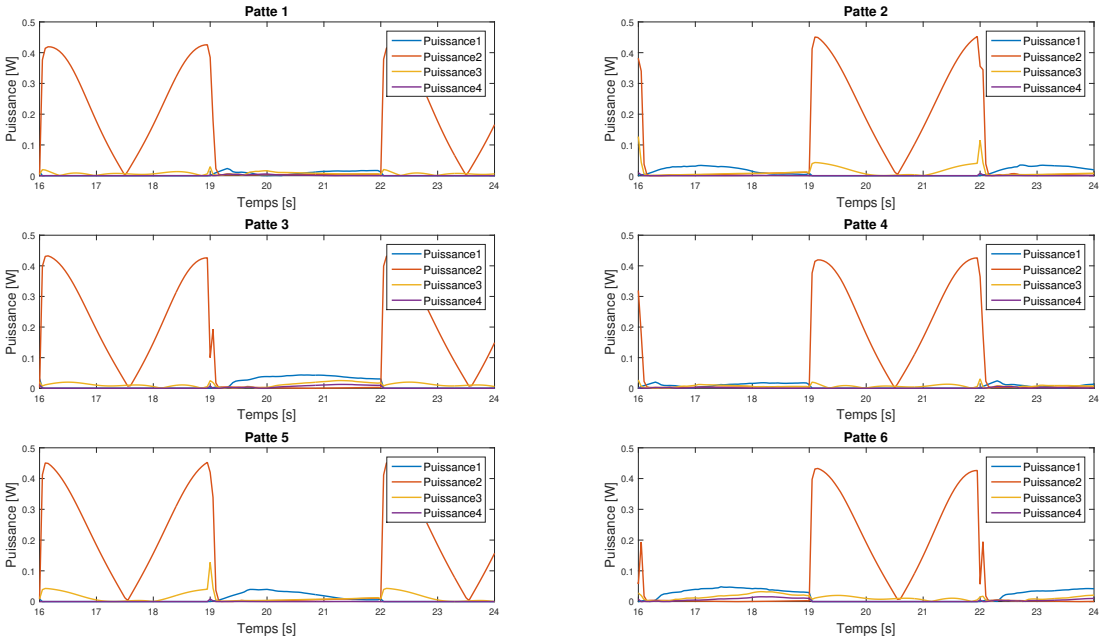


FIGURE 6.4 – Puissances mesurées lors d'un tripod gait

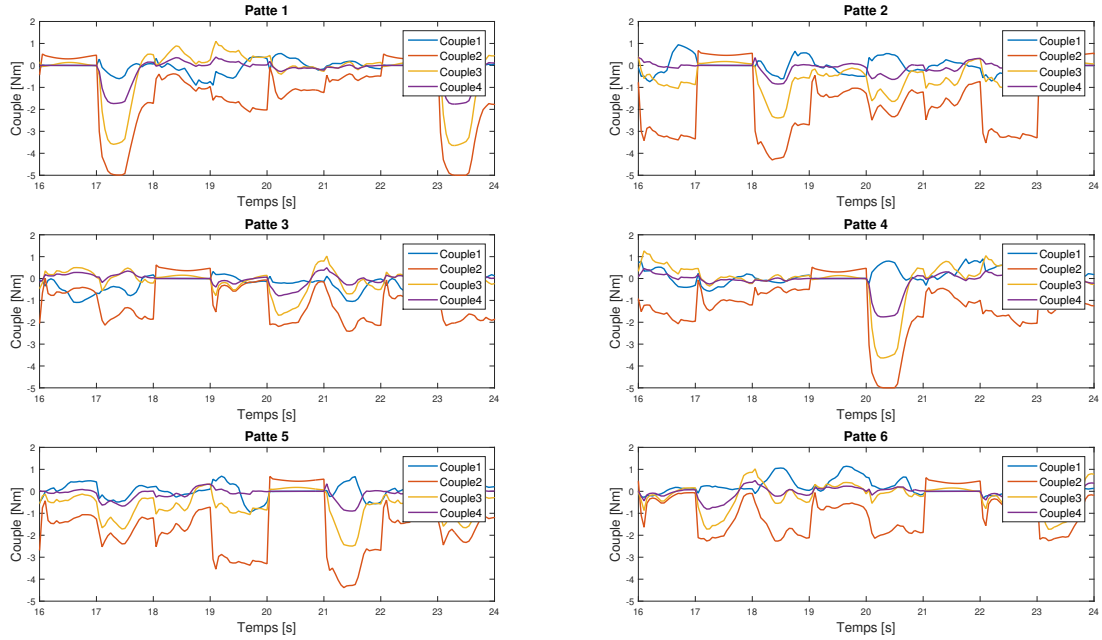


FIGURE 6.5 – Couples mesurés lors d'un wave gait

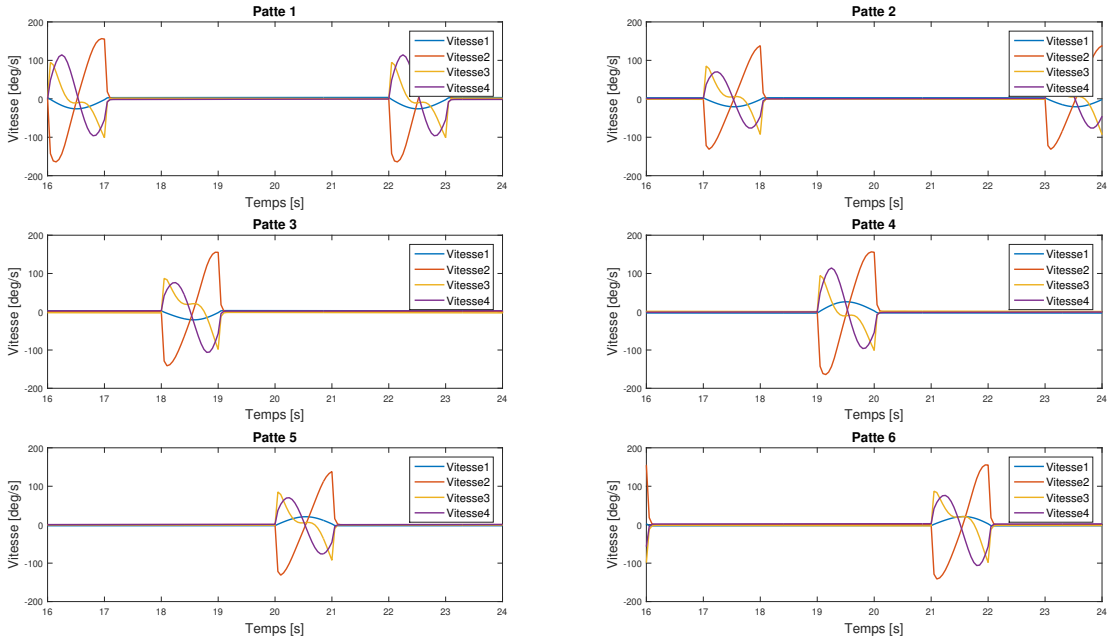


FIGURE 6.6 – Vitesses mesurées lors d'un wave gait

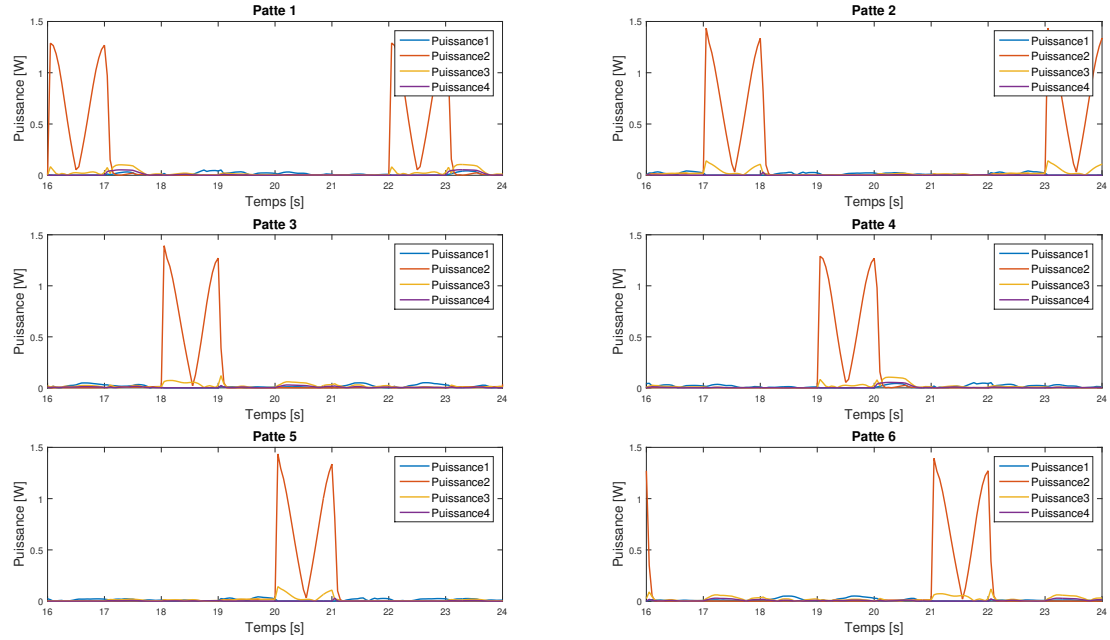


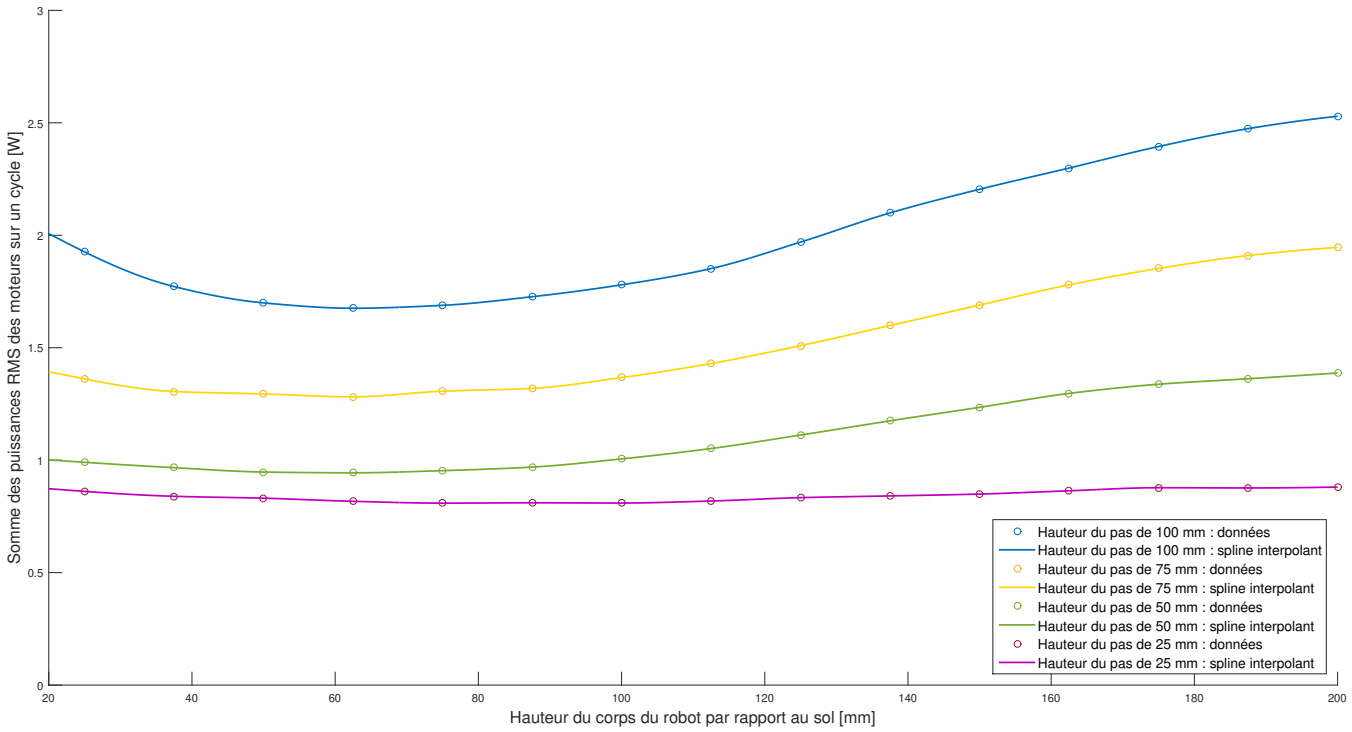
FIGURE 6.7 – Puissances mesurées lors d'un wave gait

6.3 Hauteur optimale du corps de l'hexapode

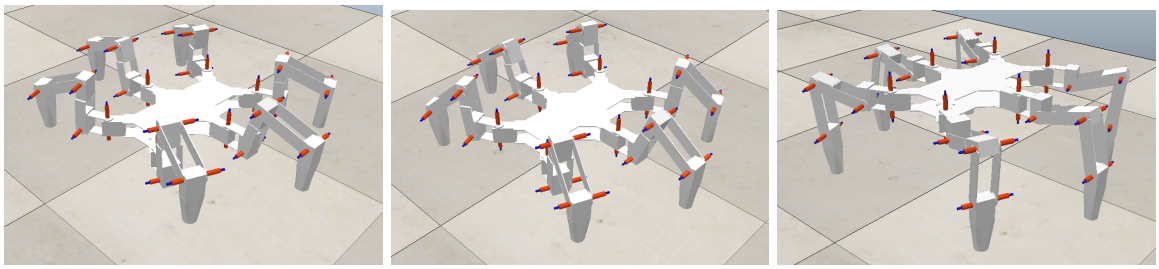
Plusieurs paramètres peuvent être choisis lors d'une démarche : la hauteur du pas, la hauteur du corps de l'hexapode par rapport au sol et les positions initiales des pattes. Est étudiée ici la puissance mécanique nécessaire sur un cycle en fonction de la hauteur du corps de l'hexapode. Un wave gait est pour cela effectué avec les mêmes paramètres qu'à la section précédente en variant la hauteur du corps de l'hexapode (entre 25 et 200 mm) pour plusieurs hauteurs de pas (entre 25 et 100 mm). Cela nous a permis d'interpoler des courbes entre les points obtenus (figure 6.8) et de déterminer la hauteur du corps minimisant la puissance nécessaire. La puissance représentée sur le graphe est la somme des puissance RMS de tous les moteurs calculée sur un cycle durant six secondes. La valeur de cette dernière peut paraître beaucoup trop petite. Cela s'explique par le fait que notre simulation ne prend pas en compte les frottements entre les liaisons qui sont normalement élevés car nos moteurs ont un rapport de réduction élevé. Les puissances mesurées sont donc les "puissances de sortie" de nos moteurs, il faudra évidemment fournir beaucoup plus de puissance à ceux-ci que ce qui est indiqué sur nos graphes. Une autre explication des puissances faibles apparaissant sur le graphe est le fait que l'on a pris les puissances RMS sur un cycle. Un cycle durant six secondes, l'hexapode aura ainsi parcouru 60 mm sur celui-ci : cela ne nécessite pas de puissances trop élevées. De plus, comme on l'a vu à la section précédente, les puissances les plus élevées sont développées lors de la phase de transfert où les vitesses sont importantes, ce qui ne représente qu'une seule seconde sur les six constituant un cycle. Tout cela explique les valeurs à priori faibles vues sur notre graphe, mais celles-ci nous permettent quand même de tirer des informations intéressantes d'un point de vue qualitatif et de comparer les paramètres de l'hexapode.

Le graphe reprend ainsi l'évolution de la puissance totale en fonction de la hauteur du corps et de la hauteur du pas. La puissance diminue en même temps que la hauteur du pas. Cela est logique : si la hauteur du pas diminue, la trajectoire des pattes pendant la phase de transfert montera moins haut et sera donc réduite. Le mouvement à exercer sera plus petit et les couples et vitesses en question seront réduits, d'où une réduction de la puissance totale.

Chaque courbe présente un minimum où la puissance est minimale. Nous pouvons déduire de cela la hauteur optimale du corps de l'hexapode par rapport au sol d'un point de vue énergétique. Ce minimum est assez marqué sur la première courbe, mais plus on réduit la hauteur du pas, moins la hauteur du corps a une influence sur la puissance, comme en témoigne la courbe à 25 mm de hauteur de pas qui est quasiment constante. La hauteur minimisant la puissance est donc de 62,5 mm.



(a)



(b) (c) (d)

FIGURE 6.8 – (a) Variation de la puissance totale sur un cycle en fonction de la hauteur du corps de l'hexapode (b) Configuration du robot avec une hauteur de 25 mm (c) Configuration du robot avec une hauteur de 62,5 mm (d) Configuration du robot avec une hauteur de 200 mm

Nous pouvons nous demander pourquoi les courbes présentent un minimum. Cela est dû à la configuration de notre robot. En changeant la hauteur du corps de ce dernier, la configuration des pattes change et certaines liaisons se retrouvent plus sollicitées que d'autres lors des mouvements. Cela est observé aux figures 6.9, 6.10 et 6.11 représentant l'évolution de la puissance dans les 3 configurations représentées à la figure 6.8. Dans les deux situations extrêmes (hauteurs de 25 et 200 mm), une seule liaison est fortement sollicitée (respectivement la troisième et la deuxième) lors de la phase de transfert. Alors qu'au minimum (hauteur de 62,5 mm), on remarque que les puissances générées sont mieux réparties au niveau de ces deux liaisons. Une meilleure répartition des efforts permettrait ainsi de minimiser la puissance nécessaire requise pour se mouvoir sur un cycle.

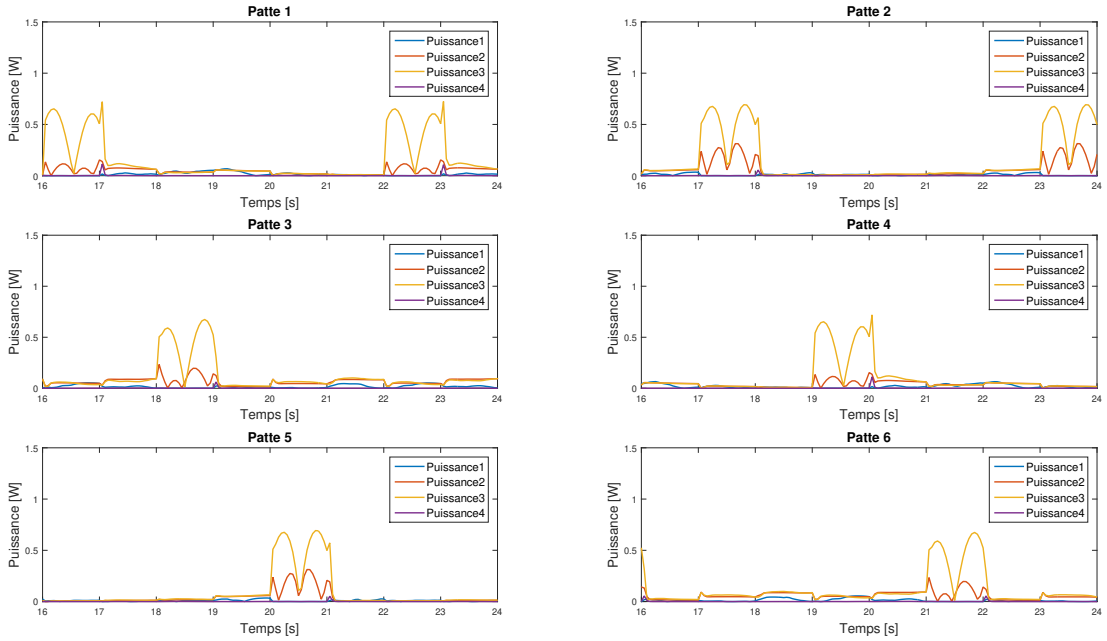


FIGURE 6.9 – Puissances mesurées lors d'un wave gait avec une hauteur du corps de 25 mm

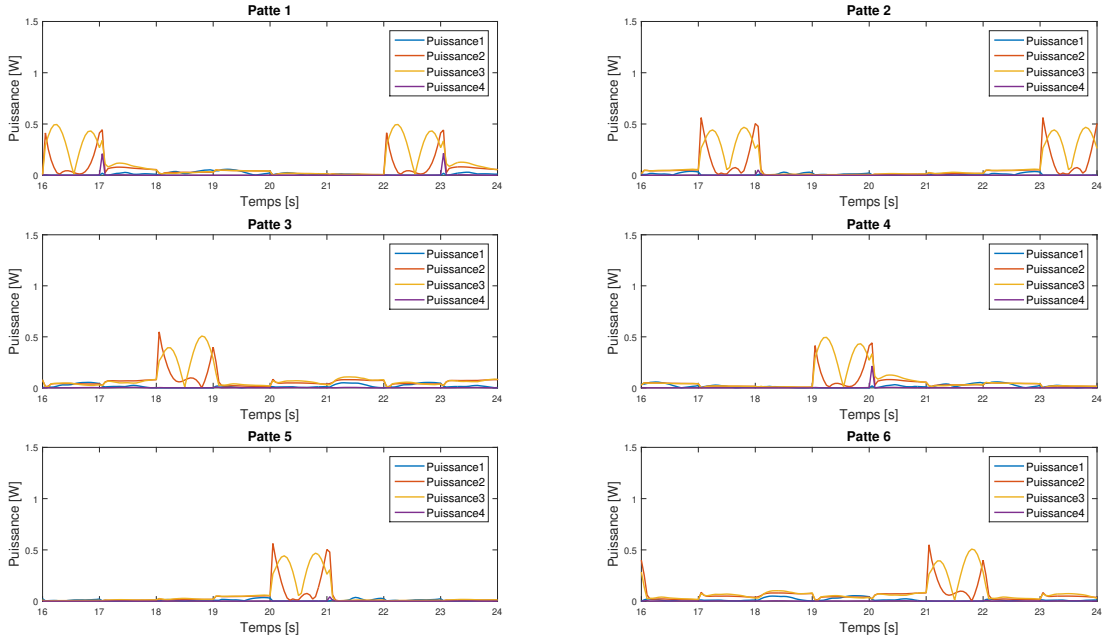


FIGURE 6.10 – Puissances mesurées lors d'un wave gait avec une hauteur du corps de 62,5 mm

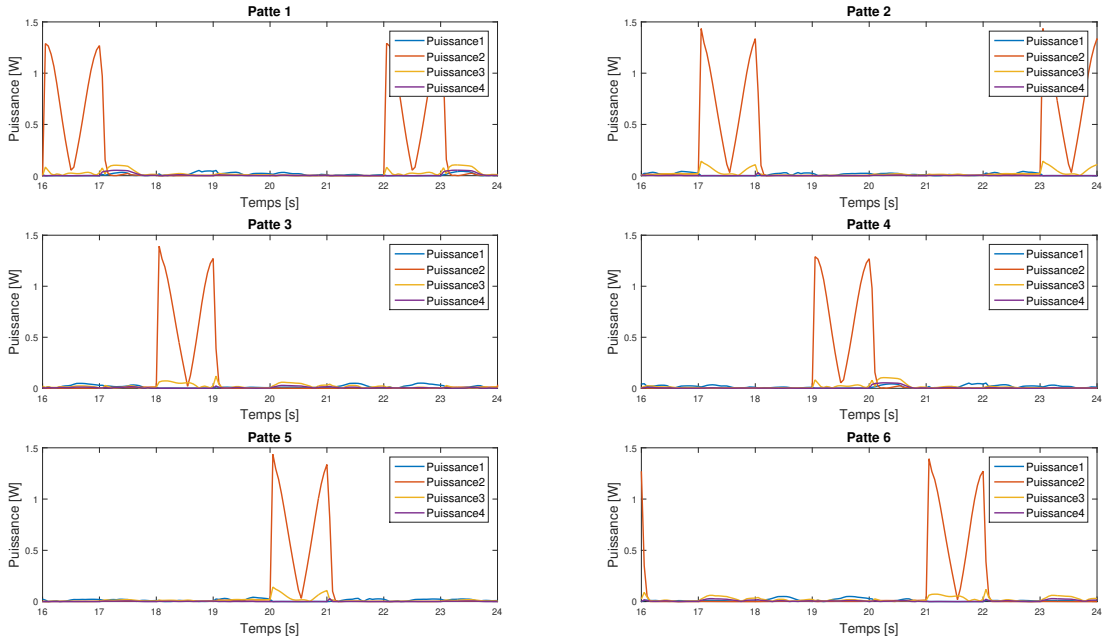


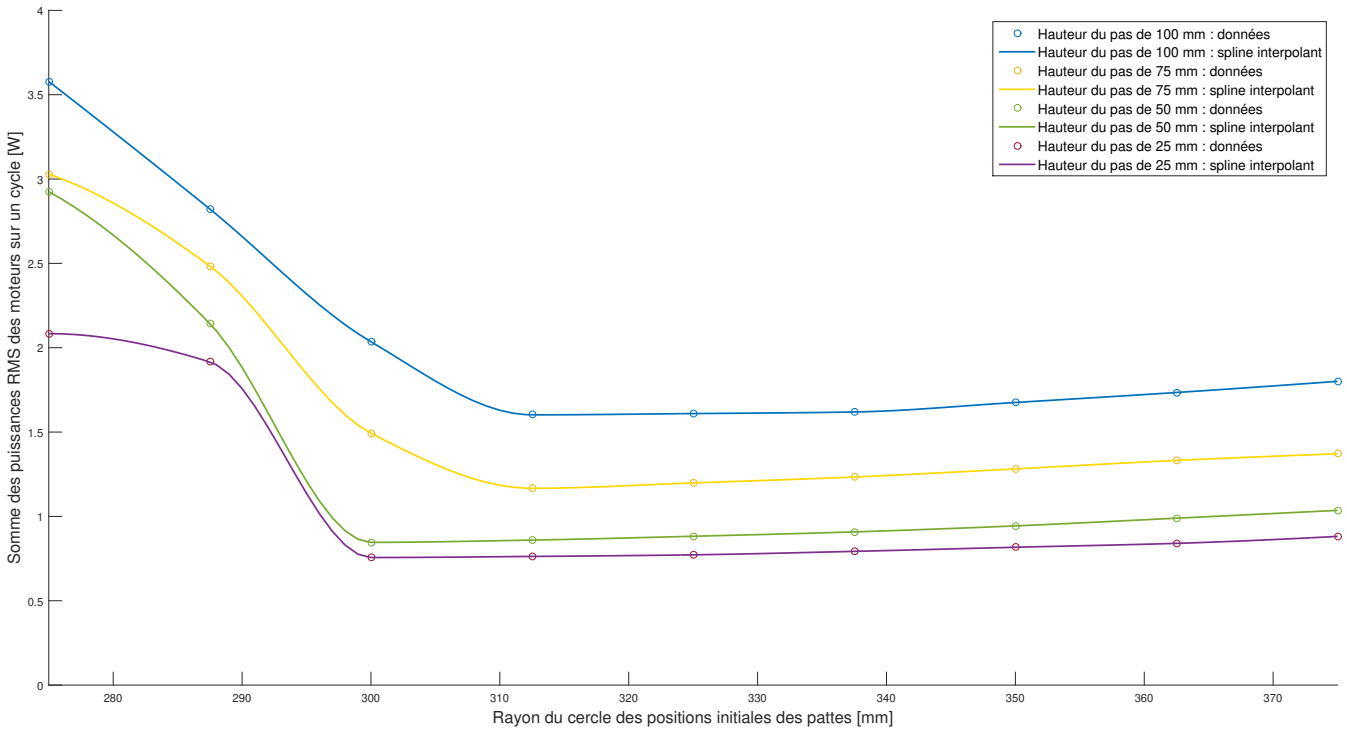
FIGURE 6.11 – Puissances mesurées lors d'un wave gait avec une hauteur du corps de 200 mm

6.4 Placement optimal des pattes de l'hexapode

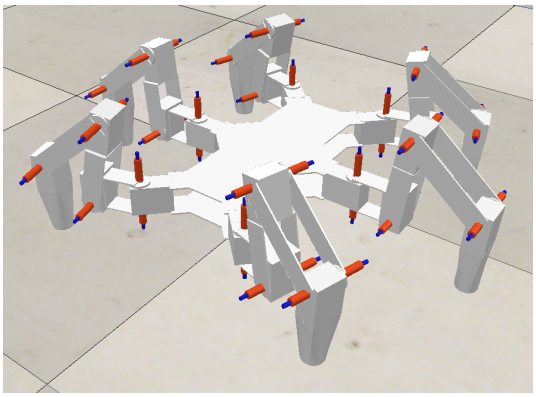
Comme décrit au chapitre précédent, le parti a été pris de positionner les extrémités des pattes du robot selon un cercle dont le centre est le centre de gravité du corps du robot. A la fin de chaque pas, chaque patte revient ainsi à sa position initiale inscrite sur ce cercle. Le rayon de ce cercle a été modifié selon plusieurs valeurs afin de voir l'évolution de la puissance totale (figure 6.12).

La première chose à observer sur ce graphe est l'abrupte remontée des courbes vers la gauche lorsque le rayon diminue. Cela s'explique par un problème d'interférences entre les pièces constituant les pattes et le corps de l'hexapode. Pendant la phase de transfert, lorsque certaines pattes remontent vers le haut, celles-ci viennent toucher le corps de l'hexapode, produisant un couple plus élevé afin d'atteindre une position impossible. Nous nous devons d'éviter ces situations anormales, il faut donc limiter le rayon du cercle à partir d'un certain point. Cela se traduit par une limite du rayon à 312,5 mm pour une hauteur de pas de 100 ou 75 mm et un rayon de 300 mm pour une hauteur de 50 ou 25 mm. Le rayon peut être plus petit dans le cas d'une hauteur de pas plus petite car les pattes remontent moins et ne rentrent plus en contact avec le corps de l'hexapode.

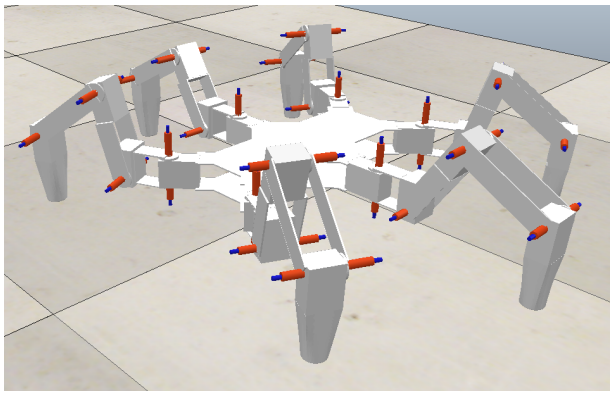
En dehors de cela, une légère réduction de la puissance est observée lorsque le rayon diminue. Dans le but de minimiser la puissance, il faut ainsi choisir le rayon limite définit plus haut. Comme à la section précédente, la puissance se réduit lors de la réduction de la hauteur du pas.



(a)



(b)



(c)

FIGURE 6.12 – (a) Variation de la puissance totale sur un cycle en fonction du rayon du cercle des positions initiales des pattes (b) Configuration du robot avec un rayon de 312,5 mm (c) Configuration du robot avec un rayon de 375 mm

6.5 Simulation de la Gecko Skin sur l'extrémité des pattes du robot

Afin de comprendre comment simuler de la Gecko Skin, il faut d'abord décrire les modèles usités dans la littérature pour décrire les mécanismes de contact intervenant avec celle-ci. [56] expose les deux principaux modèles utilisés, et permet de se faire une bonne idée de ce qui se fait dans le domaine. La propriété principale de la Gecko Skin est évidemment son adhésion importante. Il est donc nécessaire de disposer d'un modèle d'adhésion : le modèle Johnson-Kendall-Roberts (ou JKR, voir figure 6.13). Ce dernier décrit la force d'adhésion résultant d'un contact entre une demi-sphère et un plan. Comme décrit dans l'état de l'art, la peau de gecko est constituée de setae se divisant elles-mêmes en une multitude de spatulae. Celles-ci sont approximées par des cylindres présentant une demi-sphère à leur extrémité pour l'utilisation du modèle JKR. Ce modèle ne permet malheureusement pas d'expliquer l'augmentation de la force d'adhésion observée lors d'une déformation des setae. Des modèles d'adhésion fibrillaires ([57], [58], [59] et [60]) ont ainsi été développés afin de tenir compte des propriétés mécaniques des setae et de leur structure. Ces modèles permettent de comprendre les forces d'adhésion très importantes des setae présentes sur la Gecko Skin.

Le principal désavantage des modèles précédents est que la force critique capable de séparer la Gecko Skin de la surface et la force d'adhésion créée par le contact ont le même ordre de grandeur ce qui ne permet pas d'expliquer comment un gecko est capable de se détacher rapidement et facilement d'une surface : selon ces modèles, il devrait exercer une force beaucoup plus grande pour se détacher que ce que l'on observe en pratique. De plus, ces modèles prédisent une réduction de la force d'adhésion quand la rugosité de la surface présente augmente. En réalité, cette force diminue d'abord puis augmente en fonction de la rugosité.

Afin de mieux modéliser le fait que la force de détachement est beaucoup plus petite que la force d'adhésion présente, un modèle de peeling est choisi, le modèle Kendall (figure 6.13). Celui-ci décrit la force de peeling en fonction de l'angle de détachement, comme lorsque l'on détache du papier collant d'une surface. Ceci s'applique alors à une spatula au niveau nanométrique. Ce modèle permet de décrire le fait que l'on observe des forces de frottement et d'adhésion élevées selon un certain angle (lors de la phase d'accrochage du contact) et des forces de détachement très petites selon un autre angle (phase de détachement du contact). Il n'a pour l'instant été appliqué que sur une seule spatula et les futurs travaux dans ce domaine devront porter sur les comportement plus complexes des structures hiérarchiques.

Même si ces modèles permettent de mieux comprendre le comportement de la Gecko Skin, des zones d'ombre subsistent : influence des structures hiérarchiques pas encore totalement comprise, non existence de modèles permettant de décrire les déformations de ces structures et le peeling lors d'un glissement,...

Nous avons donc pu constater que le comportement de ce type de surface doit se modéliser au niveau microscopique afin de coller le plus possible à la réalité. La recherche dans ce domaine est active mais il n'existe pas encore de modèles permettant de décrire exactement la réalité. Il n'est donc à priori pas possible de simuler exactement ce type de comportement dans CoppeliaSim, à savoir une forte force d'adhésion couplée à une force de détachement bien plus faible, tout cela en fonction de l'angle du détachement ainsi que la disposition des structures hiérarchiques présentes sur la surface influençant le frottement d'un point de vue anisotropique. Nous avons quand même essayé de jouer sur certains paramètres présents dans le logiciel afin de voir si le comportement n'était pas réplicable, en modifiant le coefficient de frottement et l'adhésion du matériau. Il a été possible d'imposer un frottement élevé afin d'éviter le glissement de l'hexapode lors de la montée d'une pente, mais la force d'adhésion que l'on imposait n'était pas adaptée à notre application. Cette dernière est égale à la force nécessaire à appliquer pour détacher le pied de l'hexapode de la surface, comme expliqué plus haut. Cela ne collait donc pas au comportement de la Gecko Skin : si une force d'adhésion basse était imposée, l'hexapode arrivait à se détacher de la surface pour avancer mais la force d'adhésion n'était pas assez grande pour l'empêcher de tomber ; si la force d'adhésion était haute, ce dernier se retrouvait collé à la surface sans pouvoir se déplacer. Un comportement idéal serait d'avoir une force d'adhésion élevée lors du contact avec un détachement à la fois aisé, ce qui n'a pas su être réalisé.

Il faudra donc très certainement recourir à l'expérience comme prévu au début du projet afin de pouvoir tirer des conclusions sur l'utilisation de la Gecko Skin sur notre hexapode ou bien essayer de simuler le comportement du matériau autrement : CoppeliaSim permet l'utilisation et la création de plugins permettant

d'utiliser d'autres programmes comme Matlab en parallèle d'une simulation. Cela serait donc à priori possible à réaliser mais cela n'a pas été fait lors de ce travail car l'on a préféré se concentrer sur l'hexapode en lui-même.

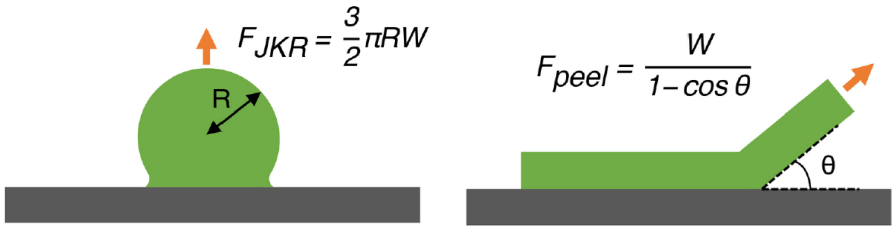


FIGURE 6.13 – (a) Modèle JKR (b) Modèle Kendall [61]

6.6 Grimper une pente

Nous avons malgré tout effectué des tests pour voir l'évolution des puissances en fonction du degré d'inclinaison de la pente que l'hexapode grimpe et de se rendre compte à partir de quel angle grimper cette pente n'est plus possible. Les paramètres utilisés sont une hauteur de corps de 62,5 mm, une hauteur de pas de 25 mm et un rayon du cercle des positions initiales de 300 mm. Un wave gait a ici aussi été utilisé. Il est également bon d'indiquer que le coefficient de frottement des extrémités des pattes a été fixé à 1, ce qui représente un contact avec un frottement élevé. Une représentation de la puissance RMS totale générée sur un cycle en fonction de la pente grimpée est présente à la figure 6.14.

Sur celle-ci, la puissance augmente en fonction de la pente grimpée, ce qui est normal vu que l'hexapode doit générer de plus en plus d'efforts s'opposant à la force de gravité plus la pente s'accentue. La puissance nécessaire diminue pour 40 et 45 °. Cela est dû au fait que l'hexapode commence à glisser légèrement vers l'arrière pour ces inclinaisons. La diminution de puissance en est ainsi une conséquence directe. Passé 45 °, le robot n'arrive plus à gravir la pente et glisse complètement sur celle-ci. On peut donc en conclure que 45 ° est la limite de la pente que l'hexapode peut grimper, sur sol plat avec un matériau présentant un coefficient de frottement de 1.

Une autre observation intéressante se trouve sur les figures 6.15 et 6.16 représentant les différentes puissances en fonction du temps. Le premier moteur se retrouve de plus en plus sollicité plus l'angle de la pente augmente. Lorsque l'angle de la pente est nulle, cela revient à une démarche normale sur un sol plat et cette première liaison permet la rotation d'une patte dans un plan perpendiculaire à la force de gravité. L'effort nécessaire est ainsi moindre que lorsque l'angle de la pente augmente et que l'influence de la gravité devient progressivement de plus en plus prépondérante pour mouvoir une patte avec cette liaison.

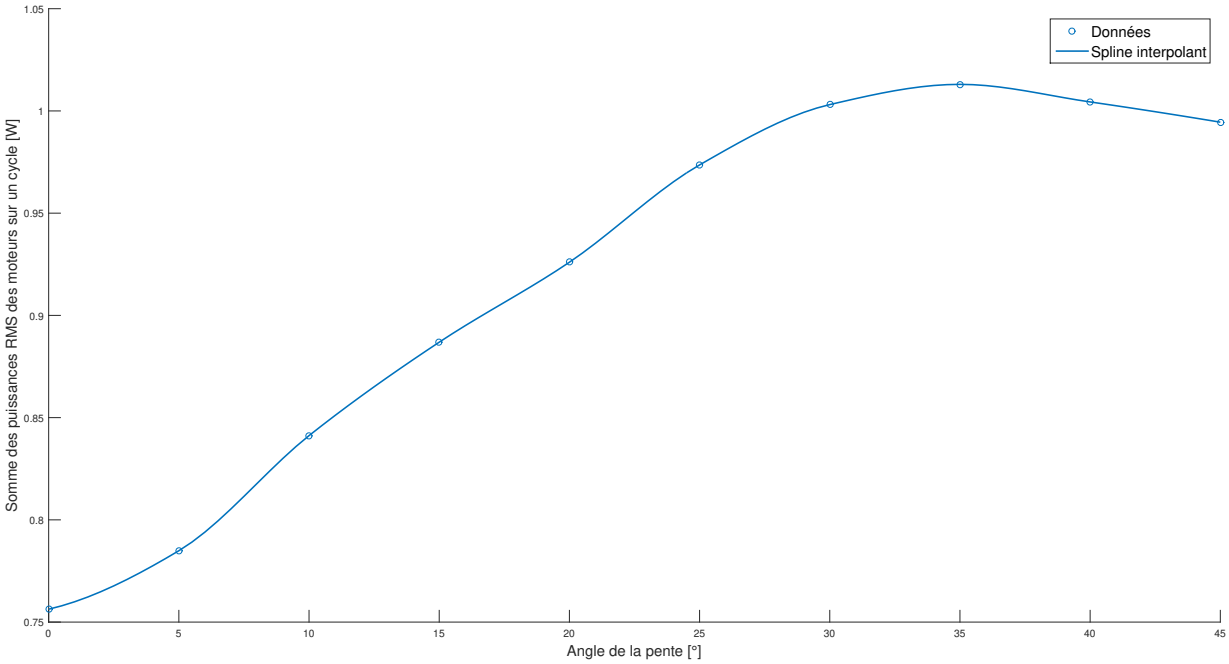


FIGURE 6.14 – Variation de la puissance totale sur un cycle en fonction de la pente grimpée

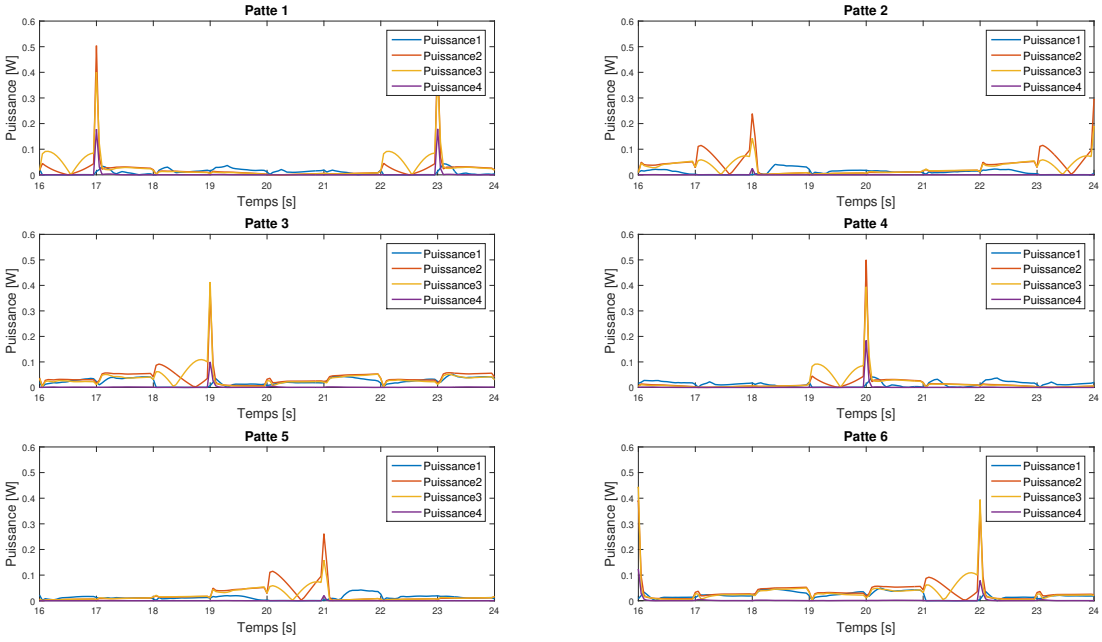


FIGURE 6.15 – Puissances mesurées lors d'un wave gait avec une pente de 0 °

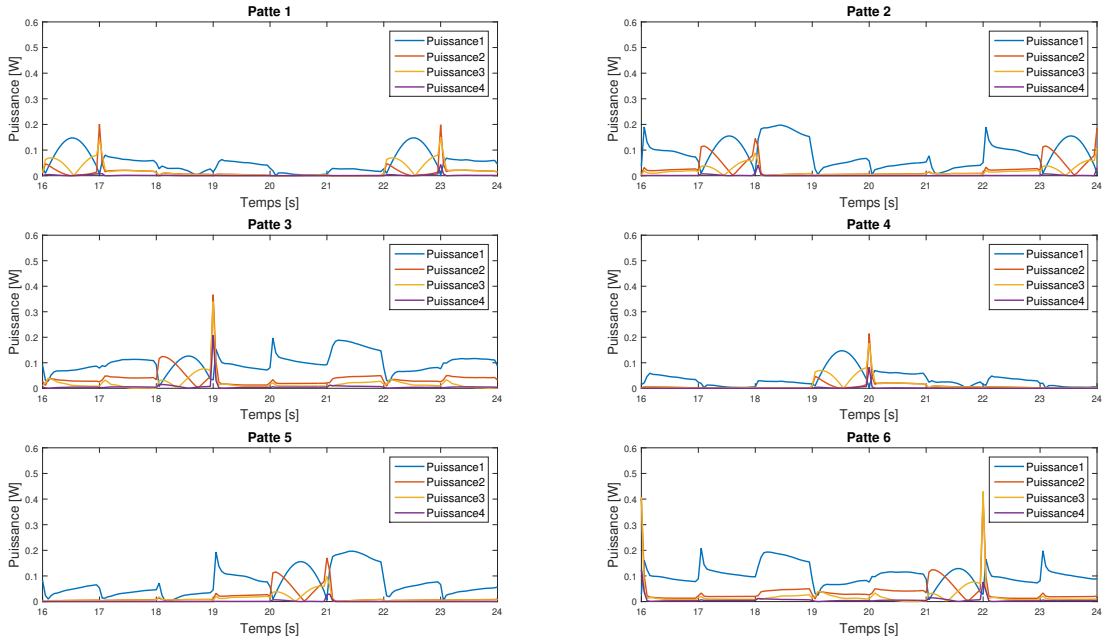


FIGURE 6.16 – Puissances mesurées lors d'un wave gait avec une pente de 35 °

6.7 Démarche sur terrain accidenté

Nous avons également testé l'hexapode et son algorithme sur un terrain accidenté généré aléatoirement. Le but était de voir jusqu'où notre algorithme qui génère une démarche fixe était applicable et si les résultats sur ce type de terrain était satisfaisants ou pas.

CoppeliaSim permet l'importation de terrains selon leur points d'élévation, il a ainsi été possible de générer sous Matlab une matrice reprenant l'altitude d'une grille de points afin d'obtenir un terrain accidenté. L'altitude de chacun de ces points a été générée aléatoirement selon une distribution uniforme entre 0 et l'altitude maximum désirée (25 mm, 50 mm, 75 mm et 100 mm). Un exemple de ce type de terrain est visible à la figure 6.17 et une trajectoire suivie par l'hexapode est aussi visible à la figure 6.18.

Nous avons donc effectué des tests en gardant les paramètre optimaux décrits dans les sections précédentes et nous avons fait varier l'altitude maximale du terrain. L'hexapode marche vers l'avant avec une vitesse d'avance de 10 mm/s sur une longueur totale de 2 m du terrain accidenté. La hauteur du pas de ce dernier varie. Un graphe a été établi sur base de ces tests montrant l'évolution de la somme des puissances RMS de toutes les liaisons pendant tout le trajet de l'hexapode en fonction de la hauteur du pas (figure 6.19).

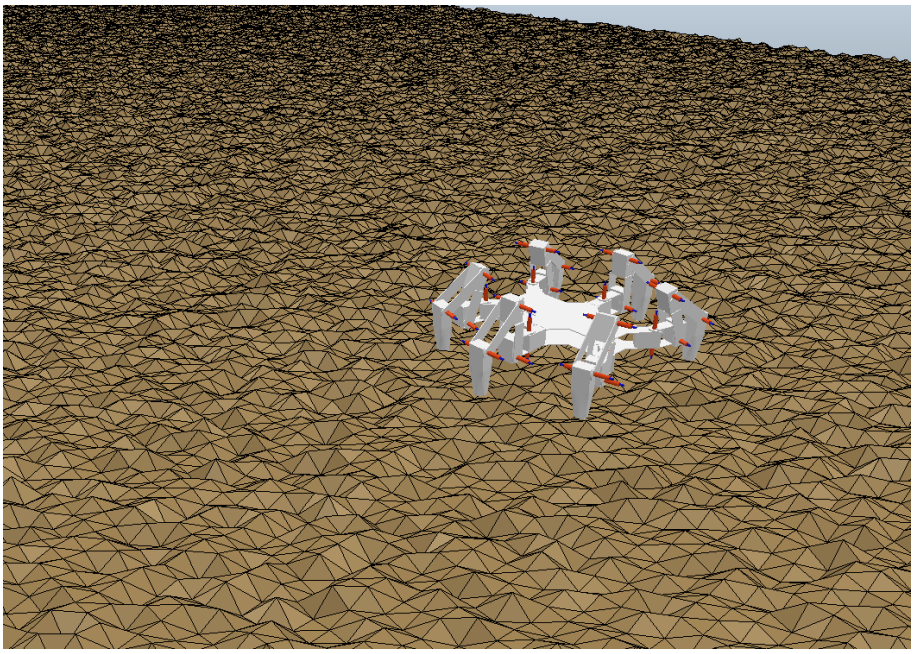


FIGURE 6.17 – Terrain accidenté avec une altitude maximum de 50 mm

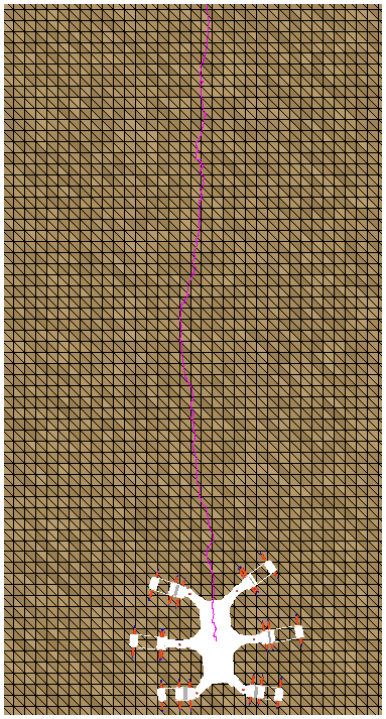


FIGURE 6.18 – Trajectoire sur un terrain accidenté avec une altitude maximum de 25 mm et une hauteur de pas de 75 mm

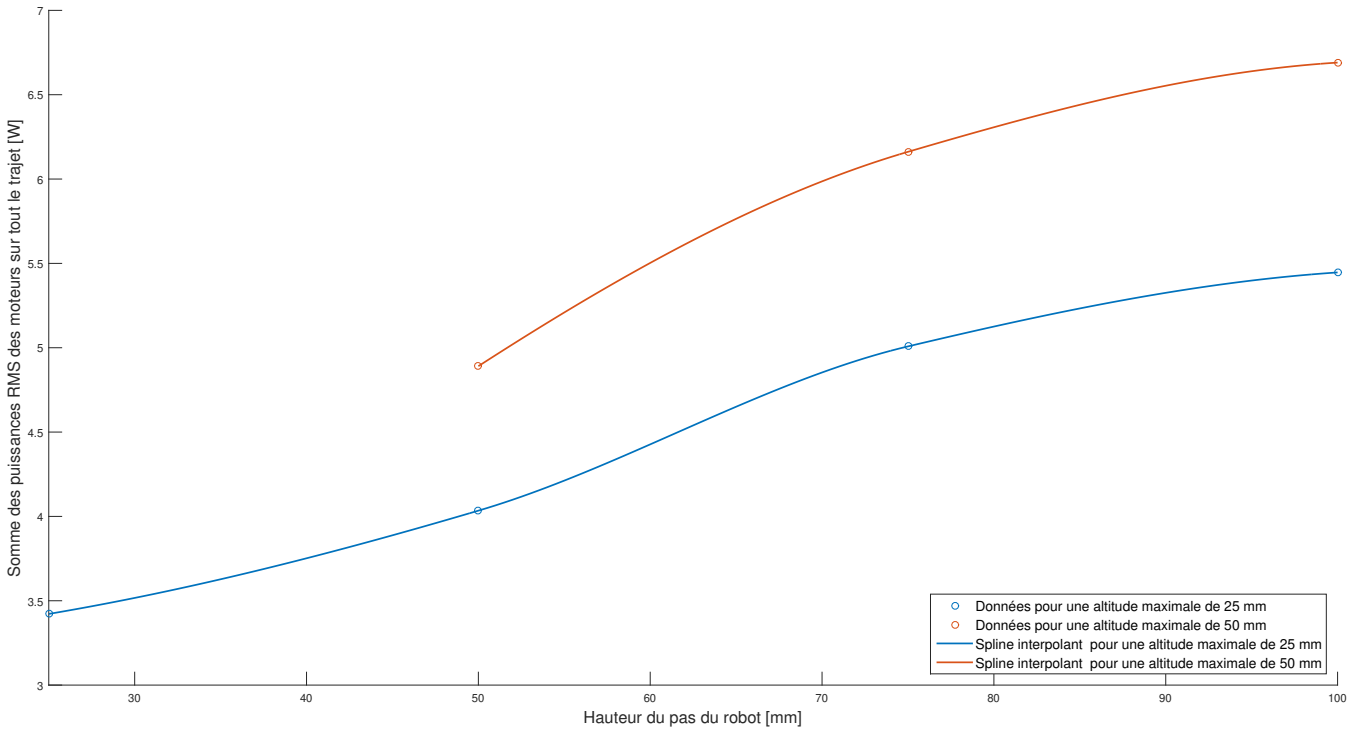


FIGURE 6.19 – Graphe de la somme des puissances RMS sur un trajet en fonction de la hauteur du pas du robot

La puissance totale augmente ainsi en fonction de la hauteur du pas, comme cela a déjà été observé précédemment. Elle augmente aussi en fonction de l'altitude maximale de notre terrain. Cela peut s'expliquer : plus l'altitude maximale est élevée et plus notre robot a du mal à avoir un contact plat avec le terrain, ce qui se traduit par des glissements plus importants lors du déplacement et une augmentation des efforts à générer. Il est bon de noter que nous avons aussi testé des altitudes maximales de 75 et 100 mm mais elles ne sont pas reprises sur le graphe car dans ces cas, l'hexapode n'arrivait pas à atteindre le bout du terrain et se retrouvait bloqué. La même raison est à l'origine d'un point manquant sur notre graphe pour une hauteur de pas de 25 mm et une altitude maximale de 50 mm, le robot n'arrivait pas non plus à naviguer sur le terrain. Cela est dû au type d'algorithme développé pour notre robot. On a choisi un algorithme avec une démarche fixe qui fonctionne très bien sur un sol plat mais donne de moins bons résultats sur un sol accidenté. Cela s'explique par le fait qu'il n'y a pas de feedback sur l'environnement extérieur à l'hexapode, celui-ci se contente de positionner ses pattes toujours dans le même plan lors de la phase de support. Dès lors, avec un terrain qui n'est pas parfaitement plat, les pattes essayent de rester dans ce plan mais n'y parviennent pas, ce qui entraîne des imperfections dans la démarche, imperfections s'accentuant lorsque le terrain devient de plus en plus accidenté.

Si l'on veut palier à ce problème, il faudra certainement adapter notre algorithme et rajouter certains types de capteurs à notre robot. Plusieurs stratégies existent, on pourrait par exemple essayer de stabiliser le corps de notre robot en utilisant un IMU ("inertial measurement unit") ce qui permettrait de réaliser un feedback sur l'inclinaison du corps du robot. On pourrait aussi rajouter des capteurs de force aux extrémités des pattes du robot afin d'avoir de l'information sur le toucher avec le sol et ainsi utiliser un algorithme de démarche libre avec d'autres règles afin de savoir quand et quelle patte on lève. On a déjà évoqué des robots utilisant ce type de contrôle de démarche dans l'état de l'art, de bons exemples sont [1] et [24].

Chapitre 7

Conclusion

Au travers de ce travail, un état de l'art sur les principales références existant dans la recherche sur les robots marcheurs a été dressé. Nous avons ainsi pu nous rendre compte de la diversité présente dans ce type de robot et s'inspirer des concepts déjà existants touchant à leur façon de se déplacer dans leur environnement. Un algorithme de démarche basique a été développé permettant à notre hexapode de se déplacer sur un sol plat selon plusieurs paramètres. Ce dernier permet de choisir un wave gait ou un tripod gait, régler les différentes vitesses de progression et de rotation de l'hexapode, choisir la durée de la phase de support et de transfert des différentes pattes. Cela nous a amené à pouvoir réaliser des trajectoires telles que suivre une ligne droite ou courbe, s'arrêter et effectuer une rotation sur soi-même ou encore pouvoir rebrousser chemin sans difficultés.

Une attention particulière a été portée sur l'architecture de notre robot et la cinématique inverse qui a été développée afin de pouvoir contrôler ce type de pattes à quatre degrés de liberté. Une solution exacte a ainsi été choisie afin de minimiser le temps de calcul et l'on a fait le choix d'utiliser notre quatrième degré de liberté disponible afin de garder l'extrémité des pattes du robot parallèles au sol afin d'obtenir un meilleur contact.

L'algorithme n'a pas pu être implémenté sur le vrai robot suite à la pandémie de COVID-19, ce qui nous a poussé à continuer le travail par le biais d'un logiciel de simulation robotique : CoppeliaSim. Grâce à celui-ci et un moteur physique performant, Vortex Studio, le test de notre algorithme a pu être réalisé de manière efficace sur un modèle du robot que nous avons tenté de rendre le plus proche possible de la réalité. Cette simulation a bien évidemment des limites, notamment le fait de ne pas avoir reproduit exactement le comportement qu'aurait eu nos moteurs MX106T sur le robot et aussi une limite due aux pièces créées sur Solidworks pour le modèle du robot, qui ne collent pas parfaitement à la réalité vu que nous ne pouvions pas mesurer le robot et que nous n'avions pas de plans de celui-ci disponibles.

Un autre aspect de ce travail était d'évaluer si il était possible d'utiliser la Gecko Skin sur notre robot afin que celui-ci puisse se déplacer dans le meilleur des cas verticalement sur un mur. Un bref état de l'art sur ce matériau et ses propriétés a été réalisé. Nous avons pu voir que sa principale méthode de fabrication était liée à l'utilisation de polymères et de moules réalisés grâce à de la lithographie. D'autres solutions low-cost existent également comme l'utilisation de feuilles de réseau de diffraction et de silicone : nous devions normalement tester cette dernière méthode au laboratoire mais cela n'a là encore pas été possible. Une discussion des modèles utilisés pour simuler le comportement de la Gecko Skin a été effectuée mais l'on s'est rendu compte que l'implémentation de ceux-ci avec les fonctionnalités de base de CoppeliaSim n'était pas possible. Il faudra ainsi sûrement recourir à l'expérimentation pratique dans le futur afin de pouvoir tirer plus de conclusions quant à l'utilisation de ce type de matériau et son application sur notre robot ou adopter un autre logiciel permettant de simuler son comportement.

Nous avons finalement réalisé une série de tests grâce à la simulation dont plusieurs informations sur les paramètres optimaux de notre robot d'un point de vue énergétique lors de son déplacement sur sol plat ont pu être tirées. La hauteur choisie du pas du robot a une influence sur l'énergie nécessaire à son déplacement : plus le pas est petit, plus la puissance nécessaire est basse. Une autre conséquence de ces tests était que la

hauteur optimale du corps de l'hexapode est de 62,5 mm. Les positions initiales des pattes minimisant la puissance ont aussi été observées et ces dernières doivent se placer sur un cercle de rayon de 300 mm ou 312,5 mm en fonction de la hauteur du pas choisie. Un test a aussi été effectué pour voir quelle était la pente maximale que l'hexapode pouvait grimper. Celui-ci ressentait un léger glissement entre 35 et 45 °et se retrouvait incapable de monter la pente après 45 °. Nous avons aussi pu remarquer au travers de ces tests que les liaisons 2 et 3 des pattes de l'hexapode étaient les plus sollicitées, sauf dans le cas d'une pente où la première liaison se retrouvait la plus sollicitée. La quatrième liaison était quant à elle très peu sollicitée, cela étant dû aux très faibles mouvements qu'elle devait réaliser pour garder le tarsus perpendiculaire par rapport au sol.

Un dernier test sur terrain accidenté a été réalisé et il en ressort que notre algorithme s'accommode de légères perturbations mais dès que celles-ci deviennent trop importantes, notre algorithme n'est plus performant. Cela est expliqué par le fait que nous avons développé un algorithme de démarche fixe et non libre, qui n'utilise aucune information sur le monde extérieur.

Nous avons donc pu dans ce travail développer un algorithme servant de base pour le robot hexapode du laboratoire de robotique de la Polytechnique de Montréal. Le travail réalisé servira de fondations pour des projets futurs à développer sur ce robot. Il faudra évidemment dans un premier temps valider l'algorithme écrit pour la simulation sur le vrai robot, ce qui permettra par la suite de continuer le travail commencé sur la Gecko Skin et d'en tirer des conclusions sur l'implémentation de celle-ci sur le robot. La possibilité de retravailler l'algorithme n'est pas exclue afin que le robot puisse tirer des informations du monde extérieur à travers des capteurs de force ou autres types de capteurs dans le but d'effectuer une démarche plus robuste sur tout type de terrain.

Bibliographie

[1] Alexandre P. "Le contrôle hiérachisé d'un robot marcheur hexapode". Université libre de Bruxelles, 1996.

[2] B. Robertson. "A preview of the european commission teleman programme for telerobotics research." Dans : *IEEE Robotics and Automation Magazine* (Décembre 1997).

[3] *Chebyshev Walking platform*. URL : http://cyberneticzoo.com/walking-machines/1892-mechanical-horse-l-a-rygg-american/.

[4] *Mechanical Horse*. URL : http://cyberneticzoo.com/walking-machines/1850-chebyshev-walking-platform-russian/.

[5] *GE Walking Truck*. URL : http://cyberneticzoo.com/walking-machines/1969-ge-walking-truck-ralph-mosher-american/.

[6] Robert B. Mcghee et Geoffrey I. Iswandhi. "Adaptive Locomotion of a Multilegged Robot over Rough Terrain". Dans : *IEEE Transactions on Systems, Man, and Cybernetics* (Avril 1979).

[7] Charles A. Klein et Randal L. Briggs. "Use of Active Compliance in the Control of Legged Vehicles". Dans : *IEEE Transactions on Systems, Man, and Cybernetics* (Juillet 1980).

[8] *Masha hexapod*. URL : http://cyberneticzoo.com/walking-machines/1976-masha-hexapod-gurfinkel-et-al-russian/.

[9] Shin-Min Song et Kenneth J. Waldron. *Machines That Walk : The Adaptive Suspension Vehicle*. 1989. ISBN : 9780262192743.

[10] R.A Brooks. "A Robot That Walks ; Emergent Behaviors from a Carefully Evolved Network ?" Dans : *MIT AI Lab Memo* 1091 (Février 1989).

[11] Ferrell Cynthia. "Robust Agent Control of an Autonomous Robot with Many Sensors and Actuators". MIT, 1993.

[12] *Adaptive Suspension Vehicle (ASV) Walker*. URL : http://www.theoldrobots.com/OSU-ASV_Walker.html.

[13] *Genghis*. URL : https://en.wikipedia.org/wiki/Genghis_Robot#/media/File:Genghis_Robot.jpg.

[14] *Hannibal and Attila*. URL : http://www.ai.mit.edu/projects/hannibal/hannibal.html.

[15] Suren N. Dwivedi et Swaminathan Mahalingam. "Periodic gaits for the CMU Ambler". Dans : *Journal of Robotic Systems* (Février 1992).

[16] Richard Seltzer. "Robot Dante II probes inside Alaskan volcano". Dans : *Chemical Engineering News* 72 (Août 1994).

[17] *Ambler*. URL : https://www.ri.cmu.edu/robot/ambler/.

[18] *Boston Dynamics robots*. URL : https://www.bostondynamics.com/robots.

[19] *PhantomX AX Metal Hexapod Mark III Kit*. URL : https://www.trossenrobotics.com/phantomx-ax-hexapod.aspx.

[20] Hirose S. et al. “Quadruped walking robots at Tokyo Institute of Technology”. Dans : *IEEE Robotics Automation Magazine* 16 (2009).

[21] A. Roennau ; G. Heppner ; M. Nowicki ; R. Dillmann. “LAURON V : A versatile six-legged walking robot with advanced maneuverability”. Dans : *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics* (2014).

[22] *SCORPION : An Eight-Legged Robot for Hazardous Outdoor-Terrain*. URL : https://robotik.dfki-bremen.de/en/research/projects/scorpion.html.

[23] Siciliano B. et Khatib O. *Springer Handbook of Robotics*. 2008. ISBN : 978-3-540-23957-4.

[24] J-C Habumuremyi, P Kool et Y Baudoin. “Adaptive Neuro-fuzzy Control of AMRU-5, a six-legged walking robot”. Jan. 2004.

[25] Aarne Halme, K. Hartikainen et K. Karkkainen. “Terrain Adaptive Motion and Free Gait of a Six Legged Walking Machine”. Dans : *Control Engineering Practice* 2 (avr. 1994).

[26] Quentin Bombled. “Modelling and Control of six-legged robots : application to AMRU5”. Université de Mons, 2011.

[27] Eduard Arzt Michael T. Northen Christian Greiner et Kimberly L. Turner. “A Gecko-Inspired Reversible Adhesive”. Dans : *Advanced Materials* 20 (oct. 2008).

[28] Daniel R. King et al. “Creating Gecko-Like Adhesives for “Real World” Surfaces”. Dans : *Advanced Materials* 26.25 (2014), p. 4345-4351.

[29] John Tamelier Sathya Chary et Kimberly Turner. “A microfabricated gecko-inspired controllable and reusable dry adhesive”. Dans : *Smart materials and structures* 22 (jan. 2013).

[30] Alborz Mahdavi et al. “A biodegradable and biocompatible gecko-inspired tissue adhesive”. Dans : *Proceedings of the National Academy of Sciences of the United States of America* 105 (fév. 2008), p. 2307-2312.

[31] Verena Tinnemann, Eduard Arzt et René Hensel. “Switchable double-sided micropatterned adhesives for selective fixation and detachment”. Dans : *Journal of the Mechanics and Physics of Solids* 123 (sept. 2018).

[32] Ping Gu Zhengzhi Wang et Xiaoping Wu. “A gecko-inspired double-sided adhesive”. Dans : *Physical Chemistry Chemical Physics* 47 (oct. 2013).

[33] Sangbae Kim et al. “Smooth Vertical Surface Climbing With Directional Adhesion”. Dans : *Robotics, IEEE Transactions on* 24 (mar. 2008), p. 65-74.

[34] Simon Kalouche et al. “Inchworm Style Gecko Adhesive Climbing Robot”. Dans : *IEEE International Conference on Intelligent Robots and Systems* (sept. 2014).

[35] URL : https://www.buygeckskin.com/.

[36] URL : https://www.innocise.com/en-en/gecomer.html.

[37] URL : https://www.youtube.com/watch?v=vpTX32KdVBQ.

[38] *Eye On Juliet*. URL : https://www.imdb.com/title/tt4702752/.

[39] *Odroid C1*. URL : https://wiki.odroid.com/odroid-c1/odroid-c1.

[40] *Arbotix Pro Robocontroller*. URL : https://www.trossenrobotics.com/arbotix-pro.

[41] *Site web de Robotis*. URL : http://en.robotis.com/.

[42] *MX106T e-manual*. URL : http://emanual.robotis.com/docs/en/dxl/mx/mx-106/.

[43] *MX64T e-manual*. URL : http://emanual.robotis.com/docs/en/dxl/mx/mx-64/.

[44] *350mm 3 Pin DYNAMIXEL Compatible Cable - Single Cable*. URL : https://www.trossenrobotics.com/350mm-3-Pin-DYNAMIXEL-Compatible-Cable.

[45] *6 Port AX/MX Power Hub*. URL : https://www.trossenrobotics.com/6-port-ax-mx-power-hub.

[46] *Goldat Batterie LiPo 3S 11,1 V 2200 mAh 35C 3S avec Deans et prise XT60 pour voiture, bateau, camion, bateau RC UAV Drone FPV*. URL : https://www.amazon.ca/-/fr/Goldat-Batterie-voiture-bateau-camion/dp/B07NZNGPBV.

[47] *Robotics simulator*. URL : https://en.wikipedia.org/wiki/Robotics_simulator.

[48] *Hexapod tutorial*. URL : https://www.coppeliarobotics.com/helpFiles/en/hexapodTutorial.htm.

[49] *Dynamics*. URL : https://www.coppeliarobotics.com/helpFiles/en/dynamicsModule.htm.

[50] Dan Thilderkvist et Sebastian Svensson. "Motion Control of Hexapod Robot Using Model-Based Design". Lund university : Department of Automatic Control, 2015.

[51] Olivier VERLINDEN. *Computer-Aided Analysis of Mechanical Systems*. Fév. 2016.

[52] M. Nitulescu S. Manoiu-Olaru et V. Stoian. "Hexapod Robot. Mathematical Support for Modeling and Control". Dans : *15th International Conference on System Theory, Control, and Computing (ICSTCC)* (oct. 2011), p. 1-6.

[53] Javier Ollervides et al. "Navigation Control System of Walking Hexapod Robot". Dans : *2012 Ninth Electronics, Robotics and Automotive Mechanics Conference* (nov. 2012), p. 60-65.

[54] Jianbo Sun et al. "Hexapod robot kinematics modeling and tripod gait design based on the foot end trajectory". Dans : *2017 IEEE International Conference on Robotics and Biomimetics* (déc. 2017), p. 2611-2616.

[55] Shibendu Shekhar Roy et Dilip Pratihar. "Kinematics, Dynamics and Power Consumption Analyses for Turning Motion of a Six-Legged Robot". Dans : *Journal of Intelligent and Robotic Systems* 74 (juin 2013).

[56] Ming Zhou et al. "Recent advances in gecko adhesion and friction mechanisms and development of gecko-inspired dry adhesive surfaces". Dans : *Friction* 1 (juin 2013).

[57] Tian Tang, Chung-Yuen Hui et Nicholas J Glassmaker. "Can a fibrillar interface be stronger and tougher than a non-fibrillar one ?" Dans : *Journal of The Royal Society Interface* 2.5 (2005), p. 505-516.

[58] Domenico Campolo, S. Jones et R.S. Fearing. "Fabrication of gecko foot-hair like nano structures and adhesion to random rough surfaces". Dans : t. 2. Sept. 2003, 856-859 vol. 2.

[59] M. P. Reyes et R. S. Fearing. "Macromodel for the mechanics of gecko hair adhesion". Dans : *2008 IEEE International Conference on Robotics and Automation*. 2008, p. 1602-1607.

[60] Burak Aksak, Michael P. Murphy et Metin Sitti. "Adhesion of Biologically Inspired Vertical and Angled Polymer Microfiber Arrays". Dans : *Langmuir* 23.6 (2007). PMID : 17284057, p. 3322-3332.

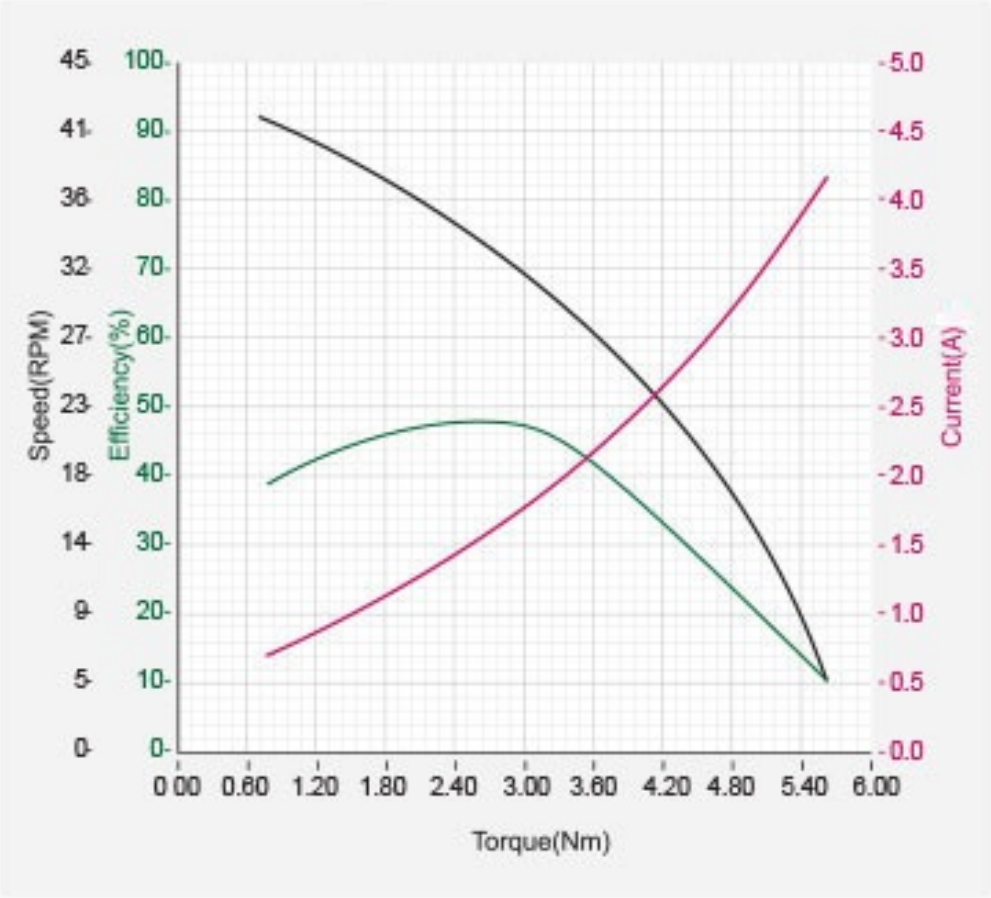
[61] Austin Garner et al. "Going Out on a Limb : How Investigation of the Anoline Adhesive System Can Enhance Our Understanding of Fibrillar Adhesion". Dans : *Integrative and Comparative Biology* 59 (avr. 2019), p. 1-9.

Annexe A

Servomoteurs MX106T

Item	Specifications
MCU	ARM CORTEX-M3 (72 [MHz], 32Bit)
Position Sensor	Contactless absolute encoder (12Bit, 360 [°]) Maker : ams(www.ams.com), Part No : AS5045
Motor	Coreless(Maxon)
Baud Rate	8,000 [bps] ∼ 4.5 [Mbps]
Control Algorithm	PID control
Resolution	4096 [pulse/rev]
Backlash	20 [arcmin] (0.33 [°])
Operating Mode	Joint Mode (0 ∼ 360 [°]) Wheel Mode (Endless Turn)
Weight	165 [g]
Dimensions (W x H x D)	40.2 x 65.1 x 46 [mm]
Gear Ratio	225 : 1
Stall Torque	8.0 [Nm] (at 11.1 [V], 4.8 [A]) 8.4 [Nm] (at 12[V], 5.2 [A]) 10.0 [Nm] (at 14.8 [V], 6.3 [A])
No Load Speed	41 [rev/min] (at 11.1 [V]) 45 [rev/min] (at 12 [V]) 55 [rev/min] (at 14.8 [V])
Radial Load	40 [N] (10 [mm] away from the horn)
Axial Load	20 [N]
Operating Temperature	-5 ∼ +80 [°C]
Input Voltage	10.0 ∼ 14.8 [V] (Recommended : 12.0 [V])
Command Signal	Digital Packet
Protocol Type	TTL Half Duplex Asynchronous Serial Communication with 8bit, 1stop, No Parity RS485 Asynchronous Serial Communication with 8bit, 1stop, No Parity
Physical Connection	RS485 / TTL Multidrop Bus
ID	254 ID (0 ∼ 253)
Feedback	Position, Temperature, Load, Input Voltage, etc
Material	Full Metal Gear Engineering Plastic(Front, Middle, Back) Metal(Front)
Standby Current	100 [mA]

Performance Graph

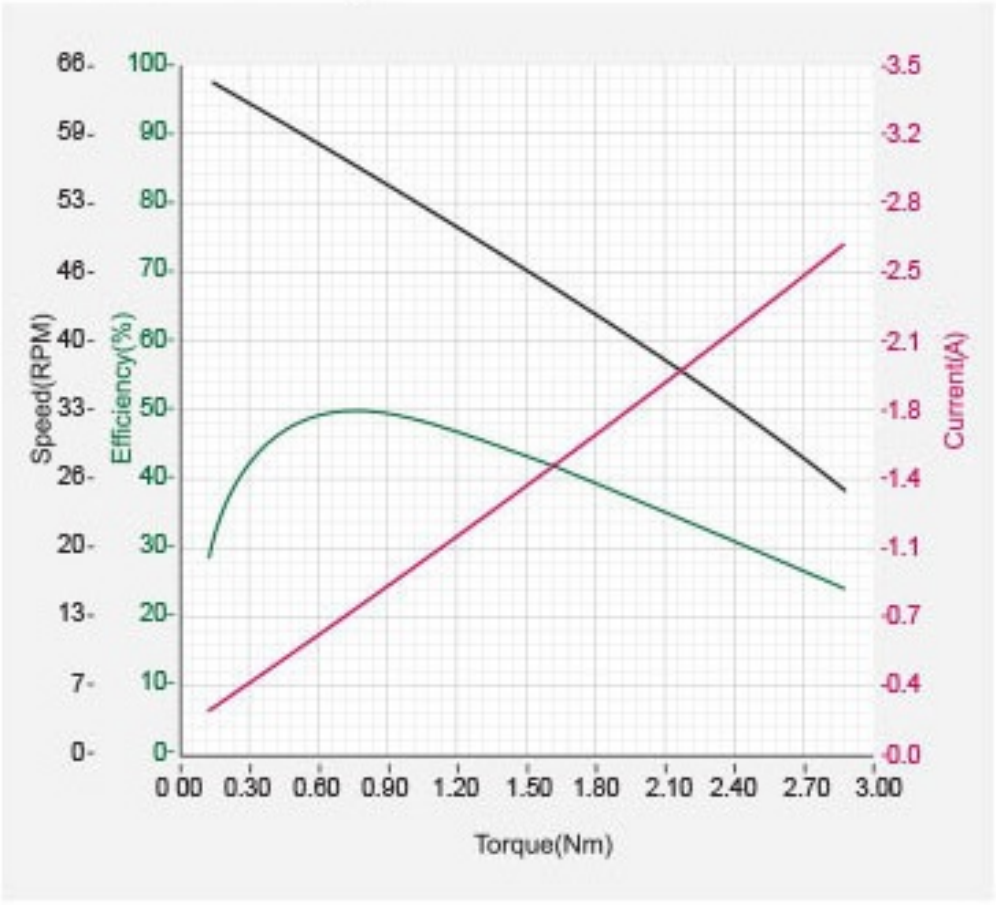


Annexe B

Servomoteurs MX64T

Item	Specifications
MCU	ARM CORTEX-M3 (72 [MHz], 32Bit)
Position Sensor	Contactless absolute encoder (12Bit, 360 [°]) Maker : ams(www.ams.com), Part No : AS5045
Motor	Coreless(Maxon)
Baud Rate	8,000 [bps] ∼ 4.5 [Mbps]
Control Algorithm	PID control
Resolution	4096 [pulse/rev]
Backlash	20 [arcmin] (0.33 [°])
Operating Mode	Joint Mode (0 ∼ 360 [°]) Wheel Mode (Endless Turn)
Weight	165 [g]
Dimensions (W x H x D)	40.2 x 61.1 x 41 [mm]
Gear Ratio	200 : 1
Stall Torque	5.5 [Nm] (at 11.1 [V], 3.9 [A]) 6.0 [Nm] (at 12 [V], 4.1 [A]) 7.3 [Nm] (at 14.8 [V], 5.2 [A])
No Load Speed	58 [rev/min] (at 11.1 [V]) 63 [rev/min] (at 12 [V]) 78 [rev/min] (at 14.8 [V])
Radial Load	40 [N] (10 [mm] away from the horn)
Axial Load	20 [N]
Operating Temperature	-5 ∼ +80 [°C]
Input Voltage	10.0 ∼ 14.8 [V] (Recommended : 12.0 [V])
Command Signal	Digital Packet
Protocol Type	TTL Half Duplex Asynchronous Serial Communication with 8bit, 1stop, No Parity RS485 Asynchronous Serial Communication with 8bit, 1stop, No Parity
Physical Connection	RS485 / TTL Multidrop Bus
ID	254 ID (0 ∼ 253)
Feedback	Position, Temperature, Load, Input Voltage, etc
Material	Full Metal Gear Engineering Plastic(Front, Middle, Back) Metal(Front)
Standby Current	100 [mA]

Performance Graph

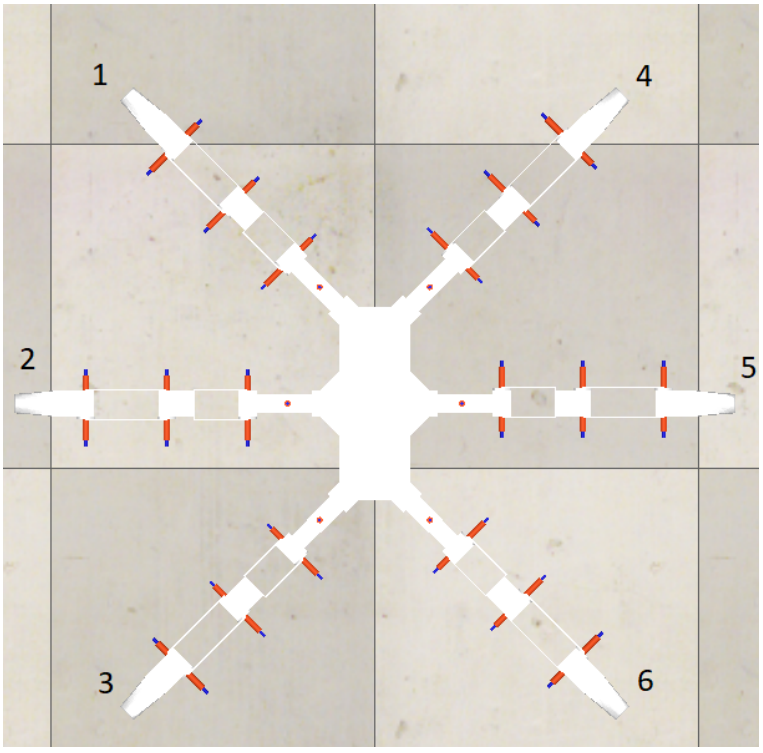


Annexe C

Paramètres géométriques du robot hexapode

Tous les angles sont exprimés en radians et les distances en mm.

C.1 Numérotation des jambes du robot



C.2 Matrices de transformations homogènes pour chacune des pattes

$$T_{0,1} = \begin{bmatrix} \cos(\frac{3\pi}{4}) & -\sin(\frac{3\pi}{4}) & 0 & -85 \\ \sin(\frac{3\pi}{4}) & \cos(\frac{3\pi}{4}) & 0 & 180 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{0,2} = \begin{bmatrix} \cos(\pi) & -\sin(\pi) & 0 & -135 \\ \sin(\pi) & \cos(\pi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{0,3} = \begin{bmatrix} \cos(\frac{5\pi}{4}) & -\sin(\frac{5\pi}{4}) & 0 & -85 \\ \sin(\frac{5\pi}{4}) & \cos(\frac{5\pi}{4}) & 0 & -180 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{0,4} = \begin{bmatrix} \cos(\frac{\pi}{4}) & -\sin(\frac{\pi}{4}) & 0 & 85 \\ \sin(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) & 0 & 180 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{0,5} = \begin{bmatrix} 1 & 0 & 0 & 135 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{0,6} = \begin{bmatrix} \cos(\frac{-\pi}{4}) & -\sin(\frac{-\pi}{4}) & 0 & 85 \\ \sin(\frac{-\pi}{4}) & \cos(\frac{-\pi}{4}) & 0 & -180 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

C.3 Longueurs des pièces constituant une jambe

$L_{coxa} = 62$, $L_{fémur} = 125$, $L_{tibia} = 125$, $L_{tarsus} = 110$