Génération de mouvement en robotique mobile et humanoïde

Guilhem Saurel

Université de Toulouse

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue le 3 octobre 2017 par :

Guilhem Saurel

Génération de mouvement en robotique mobile et humanoïde

JURY

Brigitte D'Andrea-Novel	Professeur	Rapporteur
Christine Chevallereau	Directeur de Recherche	Rapporteur
Guy Caverot	Ingénieur, Ph.D.	Examinateur
Philippe Souères	Directeur de Recherche	Examinateur
Michel Taïx	Maître de Conférences	Examinateur
Jean-Paul Laumond	Directeur de Recherche	Directeur de thèse

École doctorale et spécialité :
EDSYS : Robotique 4200046
Unité de Recherche :
Laboratoire d'Analyse et d'Architecture des Systèmes LAAS-CNRS
Directeur de Thèse :
Jean-Paul Laumond
Rapporteurs :
Brigitte D'Andrea-Novel et Christine Chevallereau

La simplicité est l'ultime sophistication.
— Leonardo da Vinci

Remerciements

> If I have seen further, it is by standing on the shoulders of giants.
>
> — Isaac Newton, 1675

Cette formule, reprise depuis le XII^{e} siècle par maints scientifiques, n'a jamais cessé d'évoluer dans mon esprit. À mon entrée au collège, je pensais qu'elle parlait de Thalès et Pythagore, et me demandais où nous en serions aujourd'hui sans ces grands personnages du VI^{e} siècle avant J.-C.

Évidemment, cette liste s'est rapidement agrandie au fil de mes études : Galilée, Descartes, Newton, Tesla, Einstein, Turing Mais en débutant cette thèse, je me suis rendu compte qu'au sein de l'équipe Gepetto du LAAS-CNRS, ces géants qui nous permettent de voir plus loin peuvent également être présents à nos côtés, et même passer du temps et de l'énergie à nous hisser eux-même au-dessus de leurs épaules.

Je tiens donc à remercier grandement Jean-Paul, Florent, Michel, Nicolas, Olivier et Philippe pour ces trois années de discussions, de bonne humeur, de sciences, de conseils avisés, de solutions techniques, de projets fous, voire même d'aventures que vous avez dirigées.

Je dois aussi remercier Mmes. Chevallereau et D'Andrea-Novel pour avoir accepté de rapporter cette thèse, et la société BA Systèmes pour avoir travaillé avec nous sur les projets présentés dans les chapitres C et D, et particulièrement à M. Caverot qui a accepté d'être membre du jury.

La bonne ambiance au sein de l'équipe m'a permis de travailler dans d'excellentes conditions, et m'a par ailleurs motivé à essayer d'y rester pour la suite. Je souhaite donc remercier pour cela tous les doctorants, post-doctorants et stagiaires que j'y ai rencontré, et notamment Alexis, Alexis, Andrea, Bernard, Céline, Diane, Dinesh, Florenç, François, Gabriele, Joseph, Justin, Kévin, Mathieu, Maximilien, Mehdi, Mylène, Naoko, Nassime, Nirmal, Pierre, Rohan, Steve et Thomas.

Il me faut également remercier du fond du cœur ma famille et mes amis, qui m'ont aidé, motivé, et soutenu, parfois sans même s'en rendre compte. Clément, Daniel, Luc, et vos familles, merci pour tout pour ces 20 dernières années.

Merci de même au club informatique de l'INP-ENSEEIHT, net7, et en particulier à lionel, pierref, storm, YGA, xouillet, zempashi, francor, benoit, seb, meloiso, bok, fabien, antoine, huhuh, vinduv, vins, tysebap, djanos, ethelward, ken, nug, linkid, nouph, maxima, viod, palkeo, carlm, mak.aʁtɔ, CHA, gofish, kordump, patate, toffan, jayjader, sligoo, pibou, kaname, bitchos, zadig, et zil0, pour toutes les discussions, l'entraide, les débats techniques, tout ce que j'ai pu apprendre, ainsi que la vie au club durant ces 7 années.

Et enfin, évidemment, un immense merci à Delphine, qui m'a énormément aidé tout au long de cette thèse, à un point que je n'imaginais pas possible, et dont je croyais être largement capable me passer. En fait, on l'a bien vu, non.

Table des matières

Introduction 3

A Introduction 3
A.1 Introduction Générale . . . 3
A.2 Précautions d'usage . . . 7
A.3 Plan . . . 8
A.4 Contributions . . . 9

I Étude de la robotique mobile 11

Introduction : Les robots à roues 13

B Robots mobiles différentiels 15
B.1 Introduction du projet offroad . . . 15
B.1.1 Céleste Boursier-Mougenot . . . 15
B.1.2 Perturbations . . . 16
B.1.3 Offroad . . . 18
B.2 Contrôle . . . 18
B.2.1 Perception . . . 18
B.2.2 Action . . . 21
B.3 Planification . . . 22
B.3.1 Champs de potentiel . . . 23
B.3.2 Entrée du système . . . 24
B.4 Architecture Matérielle et Logicielle . . . 25
B.5 Résultats . . . 28
B.5.1 Qualité du mouvement . . . 28
B.5.2 Suites du projet . . . 28

C Robots mobiles à tourelle 29
C.1 Introduction du projet LEMON . . . 29
C.2 Création de la carte . . . 30
C.3 Trajectoires de nettoyage des bordures . . . 31
C.3.1 Détection des segments de droites . . . 32
C.3.2 Détection des arcs de cercle . . . 35
C.4 Trajectoires de nettoyage des surfaces . . . 35
C.5 Génération de la trajectoire finale . . . 36
C.6 Optimisation . . . 36

C.7 Résultats . . . 39
C.7.1 Limitations . . . 39
C.8 Travaux futurs . . . 40
C.8.1 Suivi des murs . . . 40
C.8.2 Direction du nettoyage des surfaces et découpage de la zone principale 40
C.8.3 Ordre de parcours des portions de trajectoires . . . 40

D Robots mobiles tri-tourelles 43
D.1 Introduction . . . 43
D.2 Spécifications et solutions techniques . . . 44
D.2.1 Spécifications . . . 45
D.2.2 Solutions technologiques . . . 47
D.3 Architecture logicielle . . . 50
D.4 Planification de mouvement et contrôle . . . 52
D.4.1 Modélisation de la plate-forme . . . 52
D.4.2 Génération de mouvement . . . 54
D.4.3 Lissage . . . 54
D.4.4 Sélection du but . . . 55
D.5 Résultats . . . 57
D.5.1 Interface utilisateur . . . 57
D.5.2 Simulateur . . . 57
D.5.3 Résultats expérimentaux . . . 58
D.6 Suite du projet . . . 61

II Étude de la robotique humanoïde 63

Introduction : La locomotion bipède 65

E Robots bipèdes 69
E.1 Introduction . . . 69
E.1.1 Travaux apparentés . . . 70
E.1.2 Définition du problème . . . 71
E.1.3 Contributions . . . 71
E.1.4 Plan . . . 72
E.2 Simulation Dynamique . . . 72
E.2.1 Notations . . . 72
E.2.2 Modèle . . . 73
E.2.3 Contrôleur . . . 73
E.2.4 Contacts . . . 73
E.2.5 Impacts . . . 74
E.2.6 Calcul Dynamique . . . 74
E.3 Contrôle optimal pour le design et le contrôle . . . 75
E.3.1 Notations . . . 75
E.3.2 Formulation du problème de contrôle optimal . . . 75
E.3.3 Résoudre le problème de contrôle optimal . . . 76
E.3.4 Solveur pour contrôle optimal . . . 77
E.4 Premières expériences de test . . . 77
E.4.1 Entrées et sorties . . . 78

E.4.2 Fonction de coût . . . 78
E.4.3 Contraintes . . . 79
E.4.4 Actionnement . . . 79
E.4.5 Paramètres des corps rigides . . . 80
E.5 Résultats des premiers tests . . . 80
E.5.1 Influence des genoux . . . 80
E.5.2 Influence du buste . . . 82
E.5.3 Influence du cou . . . 82
E.5.4 Comparaison des types d'actionnement . . . 82
E.5.5 Passage à la troisième dimension . . . 83
E.6 Étude d'actionneurs . . . 83
E.6.1 État du système . . . 83
E.6.2 Fonctions de coût . . . 84
E.6.3 Expériences . . . 84
E.6.4 Résultats . . . 85
E.7 Étude de la stabilisation de la tête . . . 86
E.8 Fabrication d'un prototype . . . 87
E.9 Perspectives . . . 89
E.9.1 Actionneurs . . . 89
E.9.2 Pieds ellipsoïdaux . . . 89
E.9.3 Stabilité . . . 90
E.9.4 Fonctions de coût . . . 90

Conclusion 93

F Conclusion 93
F.1 Robotique mobile . . . 93
F.1.1 Contributions . . . 94
F.1.2 Perspectives . . . 94
F.2 Robotique Humanoïde . . . 95
F.2.1 Contributions . . . 95
F.2.2 Perspectives . . . 95

Annexes 99

1 Algorithmes complémentaires pour le projet LEMON 99

2 Implémentation logicielle pour le projet transhumus 101
2.1 Briques élémentaires des transferts de données . . . 102
2.2 Composants de base . . . 108
2.3 Composants finaux . . . 112
2.4 Code source complet . . . 117

Références 121

x

Table des figures

A-1 Robots humanoïdes japonais, dans la recherche à gauche, et dans la fiction à droite. . . . 4
A-2 Exemples de robots mobiles dans la recherche . . . 5
A-3 Exemples de robots mobiles dans l'industrie. . . . 6
A-4 Robots d'Aldebaran Robotics prévus pour le grand public. . . . 6
A-5 Exemples de robots mobiles dans la science-fiction. . . . 7
I-1 Trois classes de robots mobiles étudiés dans cette partie. Dans ces schémas, les flèches représentent les degrés de liberté des roues, parmi lesquels on retrouve ceux qui sont actionnés en rouge et gras. . . . 14

B-1 Céleste Boursier-Mougenot à la biennale de Venise en 2015. . . . 16
B-2 Œuvres classiques de Céleste Boursier-Mougenot. . . . 16
B-3 Œuvres de Céleste Boursier-Mougenot lors de l'exposition perturbations du musée des Abattoirs de Toulouse en 2014. . . . 17
B-4 Images des caméras au plafond superposées au niveau de l'altitude des pianos. . . . 19
B-5 Image des caméras à gauche, sortie de l'algorithme d'extraction de contours pour cette image à droite. Les contours des machines sont bien visibles, mais ceux des pianos sont estompés par endroits, y compris pour l'œil humain. . . . 21
B-6 Les pianos sont désormais des robots mobiles différentiel (2, 0). . . . 22
B-7 Champs de potentiels . . . 24
B-8 Architecture matérielle de l'œuvre offroad. . . . 25
B-9 Composants matériels utilisés pour offroad. . . . 27

C-1 Illustrations du prototype du robot LEMON. . . . 29
C-2 Exemple de carte créée avec la classe `Bitmap`. . . . 31
C-3 Illustration graphique de la transformée de Hough. Dans l'image de droite, on voit trois points blancs, dont les coordonnées correspondent aux paramètres (ρ, θ) des trois droites de l'image de gauche. . . . 33
C-4 Illustration de la détection des segments de droites à balayer pour nettoyer le long des murs . . . 33
C-5 Détection de cercles : transformées de Hough en (ρ, θ) à gauche et (x, y, r) à droite. . . . 35
C-6 Exemple de trajectoire de suivi des murs générée par l'algorithme de Reeds et Sheep étonnamment longue. . . . 37

C-7 Illustration du raccourci utilisé. Cette stratégie est surtout utile dans les angles droits fermés, mais elle réduit inutilement les segments de balayage des bordures dans d'autres cas. 38
C-8 Amélioration de l'exemple de la figure C-6. Dans certains cas, la méthode de Dubins raccourcit grandement les trajectoires, mais elle n'est pas toujours applicable. 38
C-9 Exemple d'un environnement vaste (de l'ordre de quelques centaines de mètres) et non-convexe. Relier deux trajectoires de nettoyage des surfaces dans cet exemple peut s'avérer complexe, puisque l'algorithme doit comprendre qu'il doit aller chercher la suite derrière un mur. 39
C-10 Exemple d'une salle où la bonne direction de balayage serait en diagonale, alors que le mur le plus long est horizontal. 41
C-11 Exemple d'algorithme de planification de trajectoires pour engin agricole dans des polygones non convexes. 41
C-12 Dans cet exemple, il serait intéressant de pouvoir balayer les bords des piliers ronds en s'arrêtant à mi-chemin sur les trajectoires de balayage des murs verticaux. 42

D-1 Des spécifications à la réalisation. Neuf mois séparent les deux images. . 44
D-2 Vue aérienne de la partie des Giardini qui nous intéresse. Le pavillon français est le bâtiment sur la gauche. Un arbre se déplace dans la salle principale de ce pavillon, et les deux autres se partagent l'esplanade commune aux pavillons anglais, canadien et allemand. 46
D-3 AGV dans les locaux de BA Systèmes. 47
D-4 Mesures du flux de sève : ces expériences nous ont permis de vérifier cette solution technique par rapport à nos spécifications. Nous avons constaté que la réaction d'un arbre passant de l'ombre à la lumière était mesurable et donc exploitable en tant qu'entrée de notre système, puisque le métabolisme des arbres peut influer sur leur déplacement. 48
D-5 Implantation des antennes UWB Ubisense dans les Giardini. 49
D-6 Architecture logicielle : chaque arbre a trois sondes Granier qui sont utilisées par le planificateur de trajectoire. Le planificateur de trajectoire récupère également la position et l'orientation actuelle de chaque robot grâce au système de géolocalisation, puis calcule les vitesses de traction et d'orientation de chaque tourelle de chaque AGV. Un utilisateur peut aussi directement donner des consignes au planificateur de trajectoire lorsque c'est nécessaire. Les variables (s_1, s_2, s_3), (x, y, α) et (v_i, θ_i) sont explicitées dans la section D.4. 50
D-7 Exemple d'utilisation de l'interface utilisateur web, affichant la situation actuelle : deux AGVs se déplacent sur l'esplanade, et on peut voir leur position et orientation, ainsi que l'orientation et la vitesse de traction des tourelles. 51
D-8 (v_i, θ_i) en fonction de (v, θ, ω) et (R_i, α_i). 53
D-9 Centre Instantané de Rotation (CIR). 53
D-10 Exemple de couverture d'espace en simulation. Sur site, un tel test aurait demandé plusieurs jours entre la fin de l'installation matérielle et l'ouverture de la Biennale. 57

D-11 Capture d'écran de l'interface utilisateur, dans sa version avancée, en production le 9 septembre 2015. On peut y voir les AGVs ainsi que leurs traces de 10 :30 (heure à laquelle l'interface a été lancée) à 10 :45. Il y a également une table indiquant le statut de chaque AGV ainsi que certains contrôles. . 58
D-12 Exemple de simulation avec en haut les trajectoires des centres des AGV et en bas les trajectoires des roues des AGVs. En temps réel, cela correspondrait à peu près à un voyage de 45 minutes. On remarque que les roues peuvent parfois sortir des bordures, mais pas le tronc. 59
D-13 Essais de suivi de figures simples par un AGV, pendant la phase d'installation matérielle de la biennale. La figure de gauche correspond à une réussite totale sur plusieurs tours, tandis que sur la figure de droite, on comprend que le système de géolocalisation est perdu. 60
II-1 Captures d'écrans de vidéos de Boston Dynamics et Agility Robotics. Ces robots bipèdes semblent être livrés à eux-mêmes en pleine nature. 65
II-2 Robots bipèdes passifs. 66
II-3 Robots bipèdes actifs. 67

E-1 Vue d'ensemble de l'implémentation de notre méthode de simulation et d'optimisation. Le simulateur est décrit dans la section E.2 et le solveur numérique dans la section E.3. 70
E-2 Marcheur bipède à quatre segments, articulé par deux hanches et un cou. La fonction de coût incorpore un objectif de stabilisation verticale de l'estimateur du système vestibulaire. Le marcheur adopte ici une stratégie consistant à lancer sa jambe en arrière pour mieux profiter de son inertie. 87
E-3 Modèle CAD du prototype. 88

F-1 Une roue omnidirectionnelle. Ce mécanisme est rarement fabriqué pour des robots de plusieurs tonnes, mais il aurait en théorie grandement facilité le contrôle bas niveau des robots du projet transhumus. 94

2-1 Modules de l'architecture logicielle du projet transhumus 101

Table des Algorithmes

C-1 Transformée de Hough 32
C-2 Détection des segments de droite à balayer pour nettoyer le long des murs 34
D-3 Génération de mouvement 56
1-1 Liste des configurations $\bar{\boldsymbol{q}}$ suivant la droite (ρ, θ) orientée en σ 99
1-2 Recherche des extrémités S, E de la droite (ρ, θ) sur le `Bitmap` 100

Liste des tableaux

|I-1| Cinq classes de robots mobiles, d'après Campion, Bastin, et D'Andrea-Novel (1996). 14

|E-1| Six marcheurs bipèdes sont comparés afin de déterminer l'influence des genous, du buste, du cou, des bras, et du type d'actionnement. Pour chaque exemple, l'algorithme nous fourni la démarche, le coût final de transport optimisé et la longueur d'un pas. Les deux dernières lignes donnent les performances de l'algorithme. Dans tous ces exemples, la durée d'un pas est fixée à 0.8 secondes et la pente du sol à 0.05 radians. 81
|E-2| Expériences utilisant une fonction de coût $c_{\boldsymbol{\tau}}$. 85
|E-3| Expériences utilisant une fonction de coût c_{CoT}. 85

Table des Codes sources

2-1 Valeurs initiales des données utilisées par chaque AGV. 102
2-2 `vmq/vmq.py` : composant abstrait servant de base à tous les modules. . . 103
2-3 `vmq/publisher.py` : composant de base permettant de publier des données. 104
2-4 `vmq/subscriber.py` : composant de base permettant de souscrire aux données envoyées par le publieur. 105
2-5 `vmq/puller.py` : composant de base permettant de tirer des donnnées. 106
2-6 `vmq/pusher.py` : composant de base permettant de pousser des donnnées vers le tireur. 107
2-7 `inputs/input.py` : base servant à créer des modules envoyant des données dans le système. 108
2-8 `processors/processor.py` : base servant à créer des modules capables de recevoir des données, de les traiter, et d'en mettre à jour d'autres.. . . . 109
2-9 `trajectories/base_trajectory.py` : base servant à créer le module principal générant le mouvement des AGVs. 110
2-10 `inputs/probe.py` : base servant à créer des modules correspondant à des sondes tout en vérifiant les valeurs renvoyées. 111
2-11 `output/print.py` : module de sortie affichant régulièrement les données dans le terminal. 112
2-12 `inputs/granier_serial.py` : module d'entrée permettant de lire les données renvoyées par les sondes Granier. 113
2-13 `inputs/granier_random.py` : module d'entrée permettant de simuler des valeurs de sondes granier. On peut également utiliser d'anciennes données en lisant les fichiers de logs vus dans le code source `2-12`. 114
2-14 `processors/granier.py` : module processeur de données permettant de normaliser les valeurs lues par les sondes Granier. 114
2-15 `processors/is_up.py` : module processeur vérifiant que l'AGV est correctement connecté au module principal, permettant d'alerter les utilisateurs au besoin. Un système similaire fonctionne dans l'autre sens, permettant de stopper l'AGV en cas d'anomalie sur le flux de données. Naturellement, il est nécessaire que les machines du réseau soient synchronisées en NTP. 115
2-16 `processors/websockets.py` : module processeur convertissant les données transportées par ZeroMQ en Websockets et vice-versa. 116

Introduction

Chapitre A

Introduction

Ce chapitre présente dans un premier temps la robotique dans une brève perspective historique puis par ses applications et sa place dans la société dans la section A.1, et donne des précautions d'usage dans la section A.2.

Il décrit enfin le plan de la suite de cette thèse et ses contributions respectivement dans les sections A.3 et A.4.

A.1 Introduction Générale

La robotique consiste à implémenter des facultés artificielles de perception, de décision et d'action dans une machine, afin de lui permettre de réaliser une plus grande variété de tâches.

Grâce à un savant mélange d'électronique, d'informatique, de mécanique, de mathématiques, d'intelligence artificielle, et d'automatique, la robotique a aujourd'hui bien dépassé le stade de science-fiction, où elle était encore confinée il y a quelques décennies à peine.

En effet, la culture robotique est antérieure à ses applications industrielles et à la recherche qui y est associée. Le terme robot lui-même trouve son origine en 1920 dans une pièce de théâtre tchécoslovaque de Karel Čapek. Il y désigne un automate travailleur universel, dans le sens où il pourrait exécuter n'importe quelle tâche.

La création d'un tel automate universel n'aurait pas été possible avant les travaux d'Alan Turing sur la conception d'ordinateurs pendant la seconde guerre mondiale. En effet, le premier automate industriel qui ait vocation à être universel et puisse donc être qualifié de robot est un bras manipulateur d'assemblage pour les chaînes de production de la General Motors nommé Unimate, et date des années 1960.

Entre temps, Isaac Asimov a eu le temps d'introduire les « Trois lois de la robotique » dans ses romans de science-fiction dès les années 1940. Puis, dans les années 1950, les enfants japonais ont à leur tour pu découvrir la robotique avec le manga « Astro, le petit robot » d'Osamu Tezuka.

Ce manga a initié un nouveau genre, celui des « Mecha », mettant en scène des armures robotisées humanoïdes. De ce genre est par exemple issu l'univers de Gundam (ガンダム, figure A-1b), une franchise créée dans les années 1980 comportant entre autres des films, des romans, des mangas, des animés et des jouets, et générant de nos jours annuellement environ 50 milliards de yens, soit 500 millions d'euros.

Parmi les artistes qui ont travaillé pour cette franchise, on retrouve le designer Yutaka Izubuchi, qui a également dessiné la première véritable plateforme de recherche en robotique humanoïde, HRP-2, financée par le programme japonais « Humanoid Robotics Project ».

Suite à la création du JRL (CNRS/AIST), un laboratoire franco-japonais de robotique, le LAAS-CNRS est le seul laboratoire à disposer d'une telle plateforme en dehors du Japon, et ce depuis 2006 : il s'agit d'HRP-2 14 (figure A-1a).

(a) HRP-2 14, au LAAS-CNRS, en 2015.

(b) Statue du Gundam RX-78-2 de taille réelle (18 mètres de haut), exposée de 2009 à 2017 sur l'île artificielle d'Odaiba à Tokyo.

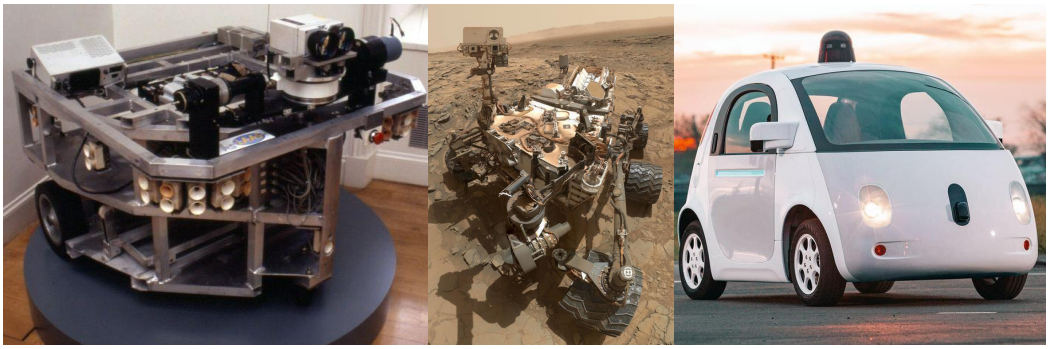
Figure A-1 : Robots humanoïdes japonais, dans la recherche à gauche, et dans la fiction à droite.

Les robots humanoïdes illustrent bien la robotique dans l'imaginaire. Cependant, dans la recherche et l'industrie, on trouve des robots à roues dans bien plus d'applications et depuis bien plus longtemps que des robots humanoïdes.

Le robot mobile HILARE (Giralt, Sobek, et Chatila 1979) (figure A-2a) a par exemple été conçu à des fins de recherche au LAAS-CNRS, et a eu trois version, en 1977, 1990 et 1999.

La recherche en robotique mobile a notamment abouti à la création de divers rovers lunaires et martiens (figure A-2b), utilisés par des scientifiques issus de tous domaines pour réaliser des expériences dans des conditions inédites, et mieux comprendre notre système solaire.

Aujourd'hui, de nombreuses entreprises et laboratoires poursuivent encore la recherche en robotique mobile, entre autres afin de robotiser nos voitures (figure A-2c), et de résoudre ainsi de nombreux problèmes engendrés par nos moyens de transport actuels.



(a) HILARE au musée des Arts et Métiers en 2009.

(b) Autoportrait de Curiosity sur Mars en 2015.

(c) Google Car, une voiture robot.

Figure A-2 : Exemples de robots mobiles dans la recherche

L'industrie est également friande de robotique mobile, surtout grâce aux Automatic Guided Vehicle (AGV, figure A-3a) qui permettent d'améliorer la gestion et les performances d'un entrepôt, et aux bras manipulateurs pouvant manier des pièces et des outils sur les chaînes d'assemblage, mais aussi grâce à de plus modestes robots qui peuvent être vendus en grandes séries directement au grand public, comme des aspirateurs (figure A-3b) ou des tondeuses.

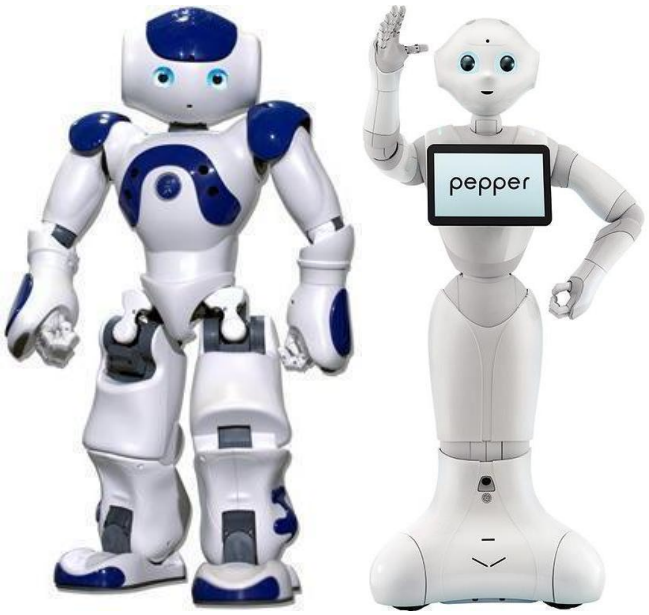
De nos jours, on trouve des applications à la robotique dans tous les domaines de l'industrie, que ce soit pour la fabrication, la manutention, ou le contrôle qualité. On la retrouve également dans un nombre croissant d'autres secteurs, comme la médecine, l'agriculture, les transports, ou encore le spatial.

De plus en plus, on retrouve également des robots dans notre quotidien, comme le Roomba (figure A-3b) présenté précédemment, ou encore Pepper (figure A-4b), un robot français de forme humanoïde (mais qui se déplace grâce à trois roues omnidirectionnelles), qui peut servir d'hôte d'accueil, et aussi son petit frère Nao (figure A-4a), utilisable comme plate-forme didactique.

(a) Transpalette robotique conçu par l'entreprise BA Systèmes.

(b) Aspirateur robotique grand public Roomba.

Figure A-3 : Exemples de robots mobiles dans l'industrie.



(a) Nao, 58cm, 2005

(b) Pepper, 121cm, 2014

Figure A-4 : Robots d'Aldebaran Robotics prévus pour le grand public.

Enfin, la robotique mobile n'est pas non plus oubliée dans la science-fiction, comme le montrent les exemples de robots mobiles bien connus donnés dans la figure A-5.

(a) Dalek, Dr Who, 1963 (b) R2-D2, Star Wars, 1977 (c) Wall-E, 2008

Figure A-5 : Exemples de robots mobiles dans la science-fiction.

En remplaçant ainsi l'homme dans un nombre croissant de tâches difficiles, répétitives, fastidieuses, voire dangereuses, la robotique démontre à la fois son impact sur la société et son intérêt économique. Son impact sociétal n'est cependant pas à prendre à la légère, comme nous allons le voir dans la section A.2.

A.2 Précautions d'usage

Les applications possibles de la robotique sont innombrables. Aujourd'hui, bon nombre d'entre elles n'attendent plus que du temps de travail de roboticiens avant de pouvoir arriver dans nos vies, et ainsi faire évoluer la société.

Cependant, cette évolution nécessite une réflexion. Comme le disait Rabelais :

> Science sans conscience n'est que ruine de l'âme.
>
> — Pantagruel, 1542

Il appartient donc au chercheur de se positionner en citoyen et de réfléchir avant de se lancer dans la science et la technique.

Dans la majeure partie de cette thèse, nous parlons d'œuvres artistiques et de science fondamentale, mais dans le chapitre C, nous verrons un projet consistant à robotiser une tâche habituellement effectuée par des êtres humains dans le but de recevoir une rémunération.

Au XV^{ème} siècle, quand Gutenberg invente l'imprimerie, les moines copistes perdent leur source de revenu, et sont remplacés par des presses mécanisées. Les bénéfices pour l'humanité dans les décennies et siècles à venir sont évidents.

Mais à plus court terme et à plus petite échelle, remplacer un être humain par une machine n'est pas une mince affaire. Par exemple, on ne veut pas voir des enfants travailler dans des usines de chaussures, mais on ne veut pas non plus qu'ils meurent de faim à cause d'un manque d'argent.

S'il ne faut pas oublier que la robotique crée à la fois des emplois et de la croissance, il est également nécessaire de constater que ces emplois sont généralement hautement qualifiés, et ne peuvent donc pas convenir à tout le monde.

De plus, pour qu'une application robotique soit économiquement viable alors que les emplois qu'elle génère sont généralement mieux rémunérés que ceux qu'elle remplace, il est probable qu'elle en crée dans des quantités moindres.

Pour l'instant, nous n'avons pas de solution miracle. Refuser la robotisation pour protéger des emplois déjà existants ne résout que temporairement certains problèmes pour en poser d'autres ensuite, comme le montre le retard industriel que la France a pris sur l'Allemagne (Laumond 2016).

Il faut donc accompagner humainement au mieux les changements apportés par ces nouvelles machines, voire repenser plus globalement la manière dont sont distribuées les richesses.

Les chercheurs et industriels en robotique et en intelligence artificielle sont au premier plan de ces réflexions, notamment avec la création de groupes de travail internationaux tels que « The IEEE Global Initiative for Ethical Considerations in Artificial Intelligence and Autonomous Systems [1] », le « future of life institute [2] », ou encore le « Machine Intelligence Research Institute [3] ».

À l'échelle locale, les universités et les laboratoires forment aussi des comités d'éthique, et organisent également des opérations de vulgarisation afin de communiquer avec le grand public et de lancer les débats sur les enjeux qui apparaissent pour la société de demain.

A.3 Plan

La robotique est intimement liée à la notion de mouvement. Le mouvement caractérise généralement le monde vivant, et plus particulièrement le règne animal. Il peut notamment servir à la manipulation, la locomotion, ou à la communication. La robotique vise donc à doter des systèmes de cette faculté, afin de leur permettre d'accomplir ce type de d'actions.

Dans cette thèse, nous nous intéresserons plus en détail à la locomotion. La locomotion terrestre est généralement réalisée à l'aide de roues ou de chenilles, ou par un système bipède, ou encore multipède.

Dans la partie I, nous étudierons la locomotion en robotique mobile, c'est-à-dire sur des robots à roues, à travers deux projets artistiques dans les chapitres B et D et un projet industriel dans le chapitre C.

1. https://standards.ieee.org/develop/indconn/ec/autonomous_systems.html
2. https://futureoflife.org/ai-open-letter
3. https://intelligence.org/

Puis dans la partie II nous parlerons de robotique humanoïde, et donc de locomotion bipède. Nous présenterons un cadre logiciel destiné à concevoir un robot bipède, en optimisant simultanément son design mécanique et son contrôle, dans le chapitre E.

A.4 Contributions

Dans cette section, nous exposons les contributions de cette thèse.

En premier lieu, nous avons dirigé la réalisation technique de l'œuvre transhumus représentant la France lors de la Biennale de Venise 2015. Cette contribution est décrite dans le chapitre D, et a donné lieu aux publications suivantes :

— Guilhem Saurel, Michel Taïx, et Jean-Paul Laumond. 2016. « transhumus : A poetic experience in mobile robotics ». Dans 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm. doi : 10.1109/ICRA.2016.7487455, hal : hal-01206067.
— Springer, à paraître

Ensuite, nous avons créé un cadre logiciel de codesign de marcheurs bipèdes optimisant l'utilisation de leur dynamique passive intrinsèque. Cette contribution est décrite dans le chapitre E, et a donné lieu aux publications suivantes :

— Guilhem Saurel, Justin Carpentier, Nicolas Mansard, et Jean-Paul Laumond. 2016. « A Simulation Framework for Simultaneous Design and Control of Passivity Based Walkers ». Dans 2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots SIMPAR, San Francisco. doi : 10.1109/SIMPAR.2016.7862383, hal : hal-01360450.
— Gabriele Buondonno, Justin Carpentier, Guilhem Saurel, Nicolas Mansard, Alessandro De Luca, et Jean-Paul Laumond. 2017. « Actuator Design of Compliant Walkers via Optimal Control », 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver. doi : 10.1109/IROS.2017.8202228, hal : hal-01483567.

Enfin, nous avons conçu une bibliothèque logicielle de planification de trajectoire pour un robot autolaveur industriel. Cette contribution est décrite dans le chapitre C.

Première partie

Étude de la robotique mobile

Introduction : Les robots à roues

Comme nous l'avons vu dans l'introduction générale, la robotique est déjà bien présente dans notre quotidien.

Pour autant, la recherche en robotique est loin d'être terminée. Citons par exemple les efforts humains et financiers actuellement fournis par des entreprises comme Tesla, Google ou Uber, ainsi que de plus en plus de constructeurs automobiles plus conventionnels, qui œuvrent à robotiser nos moyens de transports.

Afin de mieux comprendre comment la robotique permet à des systèmes de se mouvoir, nous étudierons dans cette partie la locomotion en robotique mobile, et plus particulièrement les robots à roues.

La roue est le premier et le plus simple des systèmes créés par l'homme pour assurer des fonctions de déplacement. Elle est caractérisée par un contact de roulement sans glissement, ce qui implique une contrainte dite de non-holonomie sur les déplacements du robot qu'elle supporte.

Avec différents types de roues, positionnées suivant diverses combinaisons, un robot peut se déplacer de différentes manières dans le plan. Pour étudier ces différents types de robots, nous reprendrons la classification de Campion, Bastin, et D'Andrea-Novel (1996).

Cette classification repose sur l'étude des différents types de roues, puis celle de la structure des modèles cinématiques et dynamiques de robots actionnés par ces roues. En introduisant les concepts de degré de mobilité et de degré de dirigeabilité d'un robot mobile, elle démontre que les robots mobiles peuvent être répartis en cinq classes.

Un robot mobile a donc un degré de mobilité δ_m, compris entre 1 et 3, correspondant au nombre de degrés de liberté pouvant être directement actionnés. On lui attribue également un degré de dirigeabilité δ_s, compris entre 0 et 2, indiquant le nombre de roues pouvant être indépendamment réorientées pour diriger le robot.

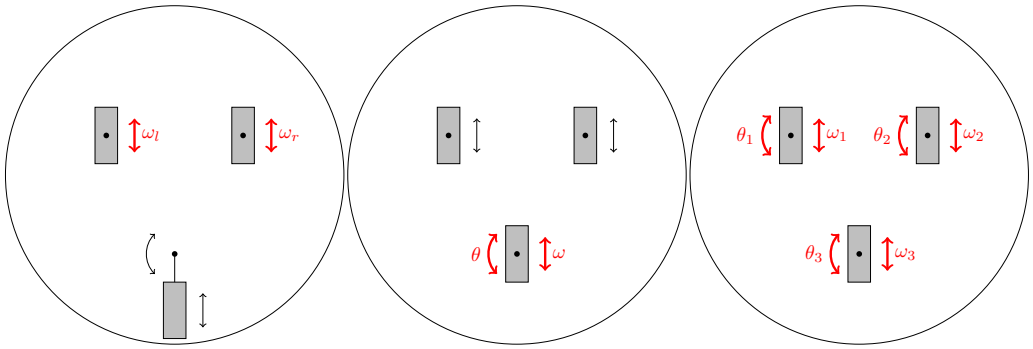
La somme de ces deux nombres correspond au degré de manœuvrabilité du robot $\delta_M = \delta_m + \delta_s$, compris entre 2 et 3, indiquant le nombre total de degrés de liberté dont il dispose dans son mouvement dans le plan.

On a alors cinq classes de robots mobiles, notées (δ_m, δ_s), présentées dans la table |I-1|.

Table |I-1| : Cinq classes de robots mobiles, d'après Campion, Bastin, et D'Andrea-Novel (1996).

δ_M	3	2	3	2	3
δ_m	3	2	2	1	1
δ_s	0	0	1	1	2

Dans la suite de cette partie, nous montrerons dans le chapitre B un exemple d'application pour des robots différentiels, c'est-à-dire munis principalement de deux roues motorisées, fixes, et sur le même axe (figure I-1a). Puis, nous étudierons dans le chapitre C, un exemple de robots munis de deux roues fixes et d'une tourelle, qui est une roue dont le plan dans lequel elle tourne est orientable autour d'un axe passant par son centre (figure I-1b). Enfin, dans le chapitre D, nous terminerons cette partie avec un exemple de robots munis de trois tourelles (figure I-1c).



(a) Différentiel (2, 0) (b) Car-like (1, 1) (c) Omnidirectionnel (1, 2)

Figure I-1 : Trois classes de robots mobiles étudiés dans cette partie. Dans ces schémas, les flèches représentent les degrés de liberté des roues, parmi lesquels on retrouve ceux qui sont actionnés en rouge et gras.

Le robot omnidirectionnel ayant plus d'actionneurs que de degré de manœuvrabilité, il est bien sûr nécessaire d'asservir certains de ces actionneurs par rapport aux autres. En d'autres termes, en pratique, trois actionneurs définissent le mouvement, et les autres se contentent de suivre, afin d'éviter des problèmes de stabilité.

Ces robots sont respectivement de type $(2, 0)$, $(1, 1)$ et $(1, 2)$. Nous verrons alors l'impact de la répartition des degrés de mobilité et de dirigeabilité lorsque le degré de manœuvrabilité est constant, puis l'impact de l'ajout d'un degré de dirigeabilité lorsqu'on ne change pas le degré de mobilité sur la planification de mouvement d'un robot.

Chapitre B

Robots mobiles différentiels

Dans ce chapitre préambule, nous établissons le rapport d'un projet de robotique réalisé préalablement à cette thèse. Il a consisté à implémenter la génération de mouvements de robots mobiles différentiels. Ces robots sont en réalité des pianos à queue se déplaçant dans un musée.

B.1 Introduction du projet offroad

Offroad est un projet robotique atypique, puisqu'il s'agit de la réalisation d'une œuvre artistique.

L'artiste, l'exposition et l'œuvre sont présentés dans la suite de cette section.

B.1.1 Céleste Boursier-Mougenot

Céleste Boursier-Mougenot (figure B-1) est un artiste plasticien, musicien et installationniste français.

Il est notamment connu pour des œuvres comme from here to ear (figure B-2a), où le public voit un musée transformé en volière abritant des dizaines de petits oiseaux, qui ont pour perchoir des guitares électriques amplifiées disposées horizontalement pour les accueillir.

C'est donc en se promenant que le visiteur joue de la musique, puisqu'il fait s'envoler les oiseaux des cordes des guitares. L'artiste qualifie alors sa musique de « vivante ».

Parmi ses œuvres, on retrouve également clinamen (figure B-2b), dans laquelle des bols blancs flottent dans une piscine circulaire bleutée, au gré d'un léger courant.

En s'entrechoquant, les bols en porcelaine forment une mélodie dont la partition est finement réglée grâce au nombre et à la taille des bols, ainsi que la force et la direction du courant.

Figure B-1 : Céleste Boursier-Mougenot à la biennale de Venise en 2015.

On voit également sur la figure B-2b des bancs autour de la piscine, invitant le visiteur à prendre le temps de s'asseoir et à profiter de l'œuvre, ce qui, paradoxalement, est plutôt rare dans un musée. Cette idée, chère à l'artiste, se retrouve dans d'autres de ses projets, comme zombiedrones ou rêvolutions.

(a) from here to ear : des oiseaux se perchent sur une guitare électrique amplifiée.

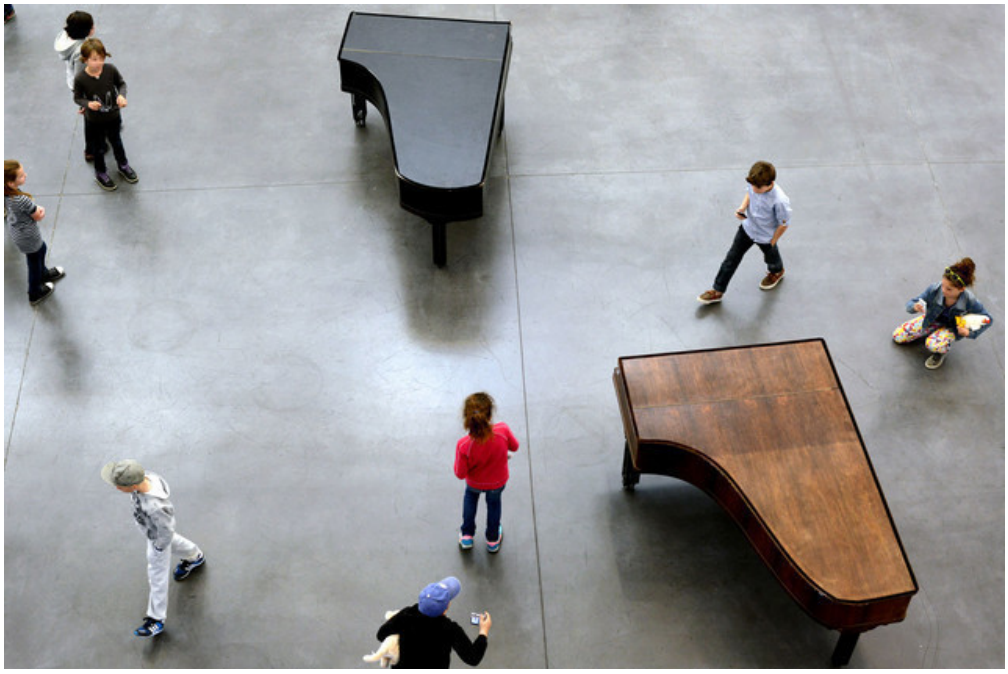
(b) clinamen : des bols de porcelaine s'entrechoquent dans une piscine suivant un courant artificiel.

Figure B-2 : Œuvres classiques de Céleste Boursier-Mougenot.

B.1.2 Perturbations

Du 31 janvier au 4 mai 2014, le musée des Abattoirs de Toulouse a organisé l'exposition perturbations (figure B-3), qui a principalement présenté cinq œuvres de Céleste Boursier-Mougenot, dont deux inédites.

L'une de ces deux œuvres réalisées pour l'occasion s'intitulait offroad (figure B-3a), et consistait à doter du pouvoir de locomotion trois pianos à queue.



(a) offroad : trois pianos à queue évoluent parmi le public.

(b) scanner : un ballon sonde muni d'un micro erre grâce à un ventilateur parmi des hauts-parleurs, créant des effets Larsen modulés.

(c) averses : un détecteur de particules cosmiques déclenche l'envoi d'une salve d'eau sur une batterie depuis le plafond.

(d) zombiedrone : une télévision où chaque image est soustraite à la précédente. La bande son est générée à partir de l'image. Les visiteurs peuvent s'assoir et changer de chaînes.

(e) U43 : un téléphone en bakélite noir de type U43 sonne lorsque le mot « fantôme » apparaît sur Google News.

Figure B-3 : Œuvres de Céleste Boursier-Mougenot lors de l'exposition perturbations du musée des Abattoirs de Toulouse en 2014.

B.1.3 Offroad

Dans cette œuvre, trois pianos à queue errent dans le musée.

Comme pour clinamen, les pianos s'entrechoquent de temps en temps ou heurtent les murs (généralement à faible vitesse), faisant ainsi résonner leur table d'harmonie.

Et comme pour from here to ear, le public fait partie de l'œuvre, puisqu'il est invité, s'il l'ose, à errer parmi les pianos. Ce faisant, il ignore que ces derniers peuvent décider de le fuir ou de le poursuivre, comme nous le verrons par la suite, section B.3.1.

Dans la suite de ce chapitre, nous détaillons la réalisation technique de cette œuvre, financée par le musée des Abattoirs de Toulouse, dirigée par l'artiste, et réalisée, en trois mois seulement, par Vincent Angladon pour la gestion de la vision par ordinateur, Guilhem de Gramont pour la mécanique, Korantin Auguste, Ken Hasselmann et moi-même pour la robotique.

B.2 Contrôle

À son plus bas niveau, le contrôle du déplacement d'un robot nécessite à minima des capteurs lui permettant de savoir quelle est sa position à un instant donné, et des actionneurs lui permettant de se déplacer. Ces deux composants doivent ensuite être reliés par un mécanisme de décisionnel.

Dans cette section, nous verrons quelles solutions techniques ont été mises en œuvre dans le cadre de ce projet pour ces besoins de perception et d'action dans les sections B.2.1 et B.2.2, puis, dans la section suivante, B.3, quel mécanisme décisionnel a été utilisé.

B.2.1 Perception

Lorsqu'on évoque la perception, l'être humain pense à ses capteurs internes qui constituent ses cinq sens. Pourtant, en robotique, il est souvent plus aisé d'utiliser des capteurs qui ne sont pas embarqués dans le robot.

Ainsi, pour de la géolocalisation de pianos à queue en intérieur pour offroad, la plupart des solutions techniques que nous avons envisagées, comprenant celle que nous avons retenue, consistaient à équiper l'aire d'évolution des pianos plutôt que les pianos eux-mêmes.

En effet, dans une pièce connue, l'une des meilleures solutions pour déterminer où un robot se situe et dans quelle orientation il est par rapport à son environnement est d'utiliser un télémètre laser balayant un plan horizontal. En ne gardant que les points les plus éloignés, on trouve la position des murs, et détermine donc la position et l'orientation du laser, et donc du robot, aux éventuelles symétries de la salle près. Mais cette solution était largement en dehors de nos moyens financiers.

Une seconde solution, au coût financier négligeable, et à la simplicité et rapidité de mise en place appréciable, est l'odométrie. Elle consiste à ajouter un capteur sur l'axe des roues afin de déterminer incrémentalement la position à chaque instant. Cependant, la précision de cette méthode s'amenuise au cours du temps, et n'est donc pas adaptée à un système devant pouvoir fonctionner pendant une journée complète sans intervention humaine. De plus, cette solution ne fonctionne pas si la roue dérape ou saute. Dans notre cas, un choc entre deux pianos semble suffisamment important pour justifier que l'on n'utilise pas cette technique.

Parmi les solutions externes de géolocalisation, il existe également la triangulation et/ou trilatération à base d'ondes (dans le domaine visible, auditif, WiFi, Bluetooth, etc.). Cette solution, bien qu'efficace (cf. section D.2.2.3), n'est pas simple à mettre en place, et faute de temps et d'argent nous avons du l'abandonner.

Notre choix s'est donc porté sur l'installation de caméras au plafond de la pièce, et l'utilisation du traitement de ces images pour détecter la position et l'orientation des trois pianos. Cette solution a entre autres l'avantage d'être plutôt discrète, et également de nous permettre de détecter les visiteurs si besoin (cf. section B.3.1).

La première étape est alors de fusionner les images des différentes caméras, comme le montre la figure B-4.

Figure B-4 : Images des caméras au plafond superposées au niveau de l'altitude des pianos.

Malheureusement, la mise en œuvre de cette solution ne s'est pas révélée aussi simple que prévu. En effet, dans notre cas, la texture du sol était similaire à celle des pianos, et le contraste entre leurs teintes n'était pas suffisant. Donc, sous un éclairage uniforme, les images présentaient un grain similaire pour les pianos et le sol, comme en témoigne la figure B-5.

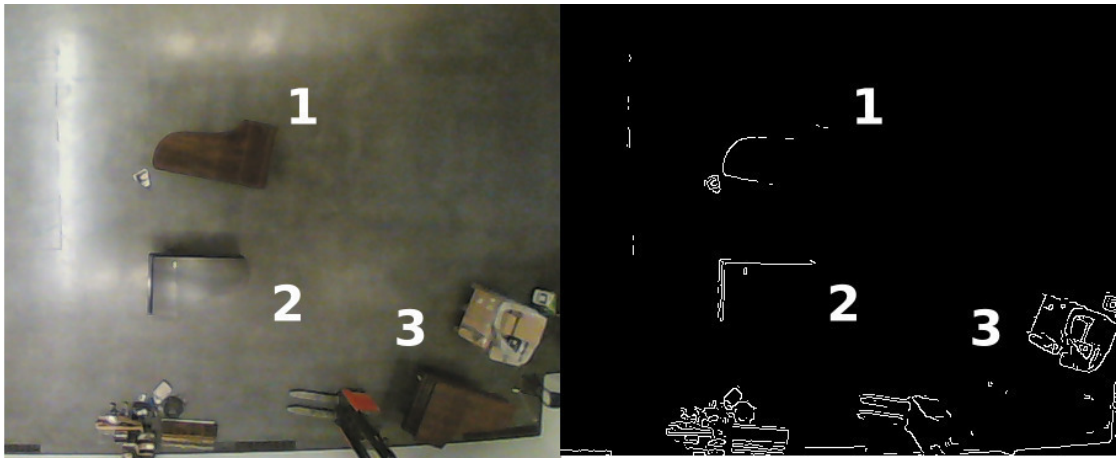


Figure B-5 : Image des caméras à gauche, sortie de l'algorithme d'extraction de contours pour cette image à droite. Les contours des machines sont bien visibles, mais ceux des pianos sont estompés par endroits, y compris pour l'œil humain.

Heureusement, en connaissant la position des pianos à un instant t et leur vitesse approximative, il est possible de forcer l'algorithme d'extraction des contours à chercher un contour particulier (connu) dans un zone réduite à l'instant $t + \delta t$.

En pratique, cela a fonctionné, mais a nécessité l'ajout d'une interface utilisateur pour que les opérateurs (les guides et vigiles du musée) positionnent correctement des masques sur les pianos le matin en démarrant l'installation.

L'algorithme de vision doit alors continuellement chercher à garder les masques au dessus des pianos, afin de n'avoir à trouver leurs contours que dans une zone réduite.

L'inconvénient de cette méthode est que lorsque l'algorithme se trompe pendant quelques itérations, il est fort probable qu'il arrête définitivement de chercher le piano à l'endroit où il est réellement. On parle alors de perte de masque, et les résultat pour le contrôle des pianos est particulièrement mauvais, si l'équipe du musée ne remarque pas sur l'écran de contrôle qu'un masque ne suit pas du tout son piano.

B.2.2 Action

Pour faire bouger ces pianos, deux moteurs ont été ajoutés et couplés via une chaîne aux roues qui sont de part et d'autre du clavier. La troisième roue, au bout de la queue, de type caster, n'est pas modifiée. Le piano devient ainsi un robot mobile différentiel de type (2, 0) (figure B-6).

Sa vitesse linéaire v est donc proportionnelle à la moyenne des vitesses des moteurs, et sa vitesse angulaire ω est proportionnelle à la différence des vitesses de ses moteurs, comme le montre l'équation (B-1).

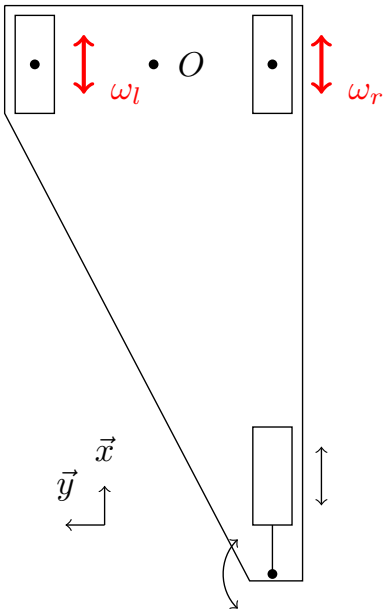


Figure B-6 : Les pianos sont désormais des robots mobiles différentiel (2, 0).

$$\begin{aligned} v &= \frac{\omega_r + \omega_l}{2} \cdot r \\ \omega &= \frac{\omega_r - \omega_l}{l} \cdot r \end{aligned} \tag{B-1}$$

où r est le rayon des roues du piano, et ω_r et ω_l respectivement les vitesses appliquées aux roues droite et gauche.

L'équation (B-1) est vraie pour le point O du piano se trouvant au milieu du clavier, donc tout point P du piano a une vitesse $\|v_P\| = v + \omega \cdot \|OP\|$.

On comprend donc que, lorsque le piano tourne, l'extrémité de sa queue atteint rapidement une vitesse conséquente. Ce fait n'est pas une problématique à négliger lorsque la tête d'un enfant visiteur pourrait se trouver au point d'intersection des trajectoires de deux pianos.

B.3 Planification

Une fois que l'on connait à un instant donné la position (x, y, α) d'un piano et qu'on est capable de le déplacer en utilisant (v, ω), le robot est contrôlable ; mais il reste à planifier, à plus haut niveau, ce que l'on veut que le robot fasse.

Naturellement, c'est à Céleste Boursier-Mougenot qu'il revient de décider ce que les robots doivent faire. Mais bien sûr, c'était à nous, roboticiens, d'implémenter techniquement le comportement choisi.

L'un des principaux challenges de ce projet a donc été d'arriver à nous comprendre, artiste et roboticiens, sur les spécifications. Cela a donc également été l'un des éléments les plus riches de cette collaboration.

Dans un premier temps, l'artiste nous a expliqué qu'il ne souhaitait pas voir de mouvements « robotiques ». Nous avons donc évité de donner aux pianos des suites de consignes simples, comme avancer et reculer en ligne droite, et tourner sur place, en suivant une machine à états classique.

Nous avons donc tenté d'implémenter des trajectoires plus « douces », telles que des splines. Cependant, des problèmes de nécessité de prédiction ainsi que de deadlocks non triviaux se sont rapidement posés, et les délais semblaient bien trop court pour que nous puissions finaliser l'implémentation d'une telle solution à temps pour le vernissage.

De plus, dans tous les cas, Céleste Boursier-Mougenot n'était pas satisfait par le rendu artistique de nos premières itérations.

B.3.1 Champs de potentiel

La solution aux problèmes évoqués ci-dessus a été d'utiliser la méthode des champs de potentiel. Dans ce paradigme, on considère que l'aire d'évolution des pianos est parsemée de potentiels P_i caractérisés par une localisation, une norme $\|P_i\|$, et un signe s_i, suivant si l'on désire un potentiel attractif ou un potentiel répulsif.

On calcule alors en un point p l'action de ce champ de potentiels $C(p)$ suivant l'équation (B-2).

$$C(p) = \sum_i \frac{s_i \|P_i\|}{\text{dist}(P_i, p) + 1} \tag{B-2}$$

La figure B-7 montre un exemple de ce qu'il se passe si l'on trace sur un graphe 3D l'allure de cette fonction dans l'aire d'évolution des pianos. Il suffit alors d'imaginer une telle surface comme un relief dans lequel le piano se baladerait en se déplaçant suivant les pentes.

Il n'est donc plus nécessaire de prédire la trajectoire des pianos, et en cas de deadlock il suffit d'ajouter un fort potentiel répulsif sur le piano. De plus, le mouvement produit semble « naturel » et non « robotique » pour un artiste, ce qui n'est pas une contrainte simple à remplir en utilisant d'autres méthodes.

Par différences finies, on peut donc déterminer la forme de la « pente » sur laquelle roule un piano, et donc ajuster sa vitesse en conséquence, comme le montre l'équation (B-3).

$$\begin{aligned} v &= \frac{C(O + \varepsilon\vec{x}) - C(O - \varepsilon\vec{x})}{2\varepsilon} K_v \\ \omega &= \frac{C(O + \varepsilon\vec{y}) - C(O - \varepsilon\vec{y})}{2\varepsilon} K_\omega \end{aligned} \tag{B-3}$$

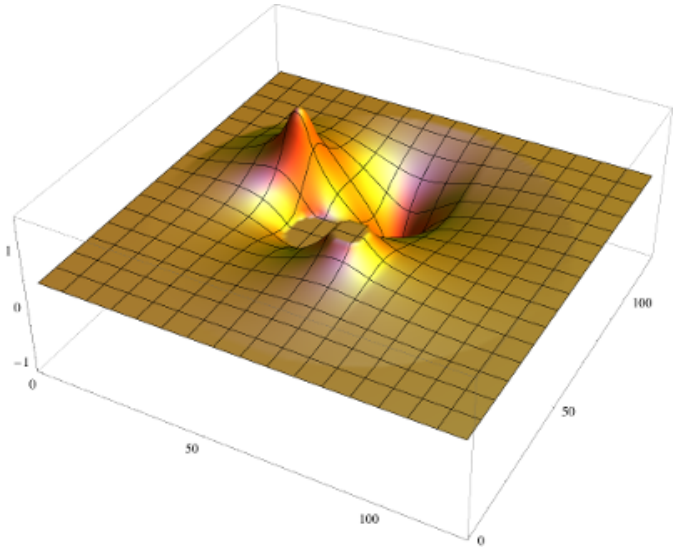


Figure B-7 : Champs de potentiels

Dans notre cas, nous avons considéré les murs et d'autres zones interdites comme des potentiels répulsifs constants. En jouant sur la norme du potentiel des murs, on peut modifier la fréquence à laquelle les pianos vont s'y cogner. On peut alors contenter à la fois l'artiste qui désire que cela puisse arriver, et l'équipe du musée qui doit maintenir les murs dans un état correct [1].

Ensuite, les pianos sont vus les uns par les autres comme des potentiels, tantôt attractifs tantôt répulsifs, jusqu'à ce que l'on détecte un choc. À ce moment-là, les pianos deviennent de forts potentiels répulsifs l'un pour l'autre pour une durée limitée.

Enfin, selon les circonstances, le système de planification peut repérer un visiteur (toujours grâce aux caméras présentes au plafond), et le considérer comme un potentiel attractif ou répulsif.

B.3.2 Entrée du système

Comme nous l'avons vu dans la section B.3.1, il suffit d'ajouter des potentiels pour que les pianos bougent. Cependant, Céleste Boursier-Mougenot souhaite que le comportement de ses œuvres ne soit ni prédictible, ni dicté par un générateur de nombres aléatoires.

Ainsi, dans les œuvres décrites dans la figure B-3, des éléments extérieurs comme les particules cosmiques, les résultats en temps réel de Google News ou encore les chaînes de télévision sont introduits et dirigent l'expérience du visiteur.

Dans offroad, l'artiste a choisi d'utiliser le vent. Nous avons donc installé une girouette et un anémomètre sur un mur extérieur du musée, de sorte qu'ils soient visibles à travers des fenêtres lorsqu'on est à côté de l'œuvre.

1. Un rideau de scène de 8.30 × 13.25m réalisé par Pablo Picasso se trouve en permanence derrière l'une de ces cloisons. On comprendra donc que l'équipe du musée tienne à ce que la cloison ne s'effondre pas.

La vitesse du vent, son accélération, et sa direction sont donc dans cette œuvre les principaux facteurs qui créent des potentiels, soit directement, soit indirectement en désignant un visiteur. Celui-ci peut ainsi, sans le savoir, faire partie de la performance.

B.4 Architecture Matérielle et Logicielle

Dans un premier temps, l'artiste voulait que les pianos soient « maîtres d'eux-mêmes », et donc que nous embarquions tous nos algorithmes dans de petits ordinateurs à leur bord.

Sur le plan théorique, cela ne change pas grand-chose. Les batteries de voitures déjà embarquées sur les pianos suffisent largement à alimenter en plus un mini ordinateur de type Raspberry Pi ou NUC/Brix suivant la puissance de calcul requise.

Cependant, en termes de complexité de déploiement, le coût en temps était trop élevé. De plus, les choix présentés en section B.2.1 imposaient la présence d'une interface utilisateur dans le musée.

Nous avons donc utilisé une architecture centralisée, où un ordinateur de bureau, muni d'un écran, d'un clavier, d'une souris et d'une manette de jeu, restait à disposition de l'équipe du musée, et servait à orchestrer les déplacements des pianos ainsi qu'a afficher une interface graphique pour des utilsateurs peu formés.

La figure B-8 explique l'architecture matérielle utilisée pour cette œuvre.

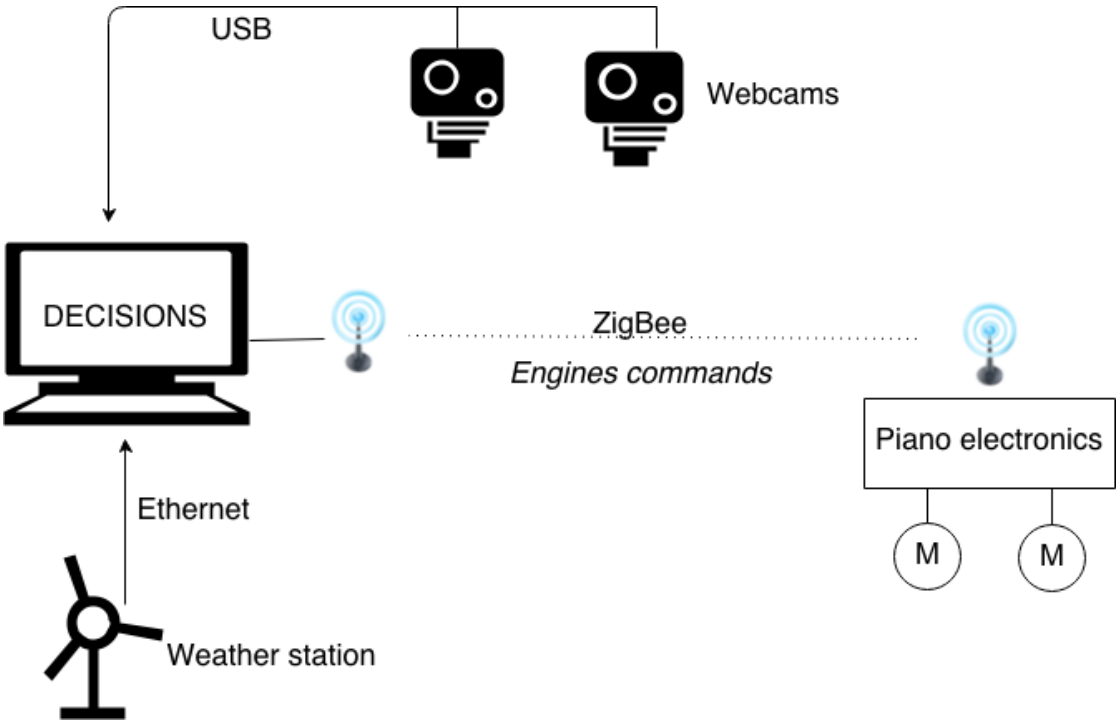


Figure B-8 : Architecture matérielle de l'œuvre offroad.

Dans la [figure B-8, on retrouve deux caméras décrites à la section B.2.1, attachées au plafond, reliées par USB directement sur l'ordinateur principal.

Un module XBEE est également connecté à cet ordinateur, et envoie des ordres aux pianos grâce au protocole ZigBee.

Sur les pianos, on retrouve ces différents composants :

— Une batterie de voiture 12V 120Ah ;
— Deux moteurs ;
— Un module XBEE pour recevoir les ordres de l'ordinateur principal (figure B-9c) ;
— Un accéléromètre pour détecter les chocs ;
— Un capteur de courant par sécurité ;
— Une carte de contrôle moteur 60A (figure B-9d) ;
— Un Arduino (figure B-9b) qui implémente l'équation (B-1), gère les autres composants électroniques, et remonte des données sur l'état courant à l'ordinateur principal ;
— Un « shield » Arduino sur mesure pour connecter tous ces composants électroniques (figure B-9e).

Enfin, la station météo (figure B-9f) est connectée à un Arduino qui interprète les données analogiques, et les envoie à une Raspberry Pi (figure B-9a) en USB, qui à son tour les transmet à l'ordinateur principal en ethernet grâce à la librairie ZeroMQ.

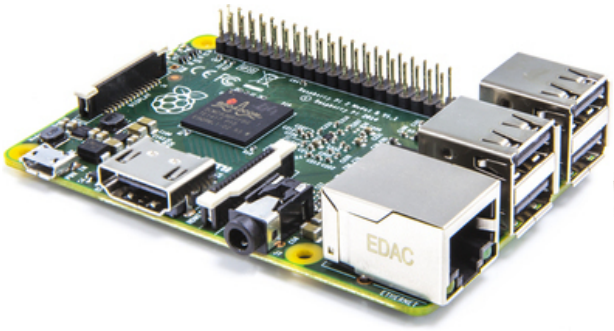
Cette solution a été choisie puisque d'une part nous avions déjà tous les composants nécessaires (et nous n'avions que très peu de temps pour des commandes de matériel supplémentaire), et d'autre part il y a besoin de plusieurs dizaines de mètres de câbles. L'ethernet est donc l'une des seules liaisons physiques viables.

Tous ces composants sont illustrés sur la figure B-9.

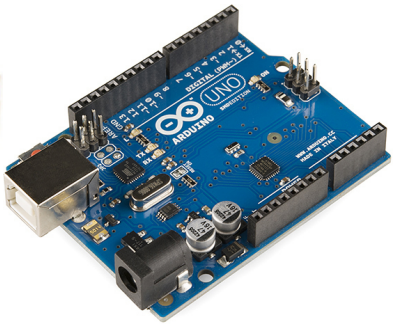
Une manette de jeu était également fournie et branchée sur l'ordinateur principal, afin de pouvoir déplacer manuellement les pianos.

Du point de vue logiciel, tous nos développements ont été effectués en C pour les microcontrôleurs et en Python pour les autres éléments.

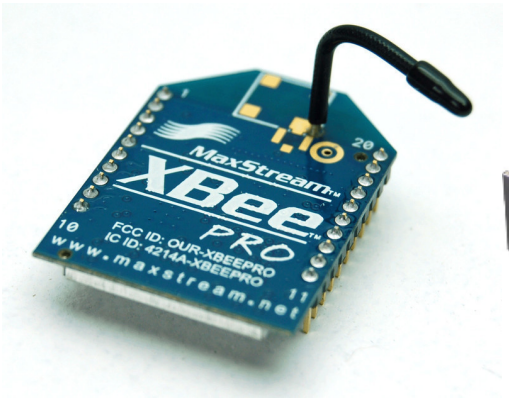
Le protocole de communication entre les arduinos des pianos et l'ordinateur principal a été créé sur mesure.



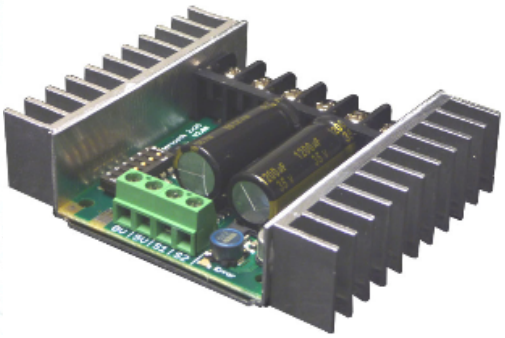
(a) Raspberry Pi, un ordinateur avec un processeur ARM de la taille d'une carte de crédit.



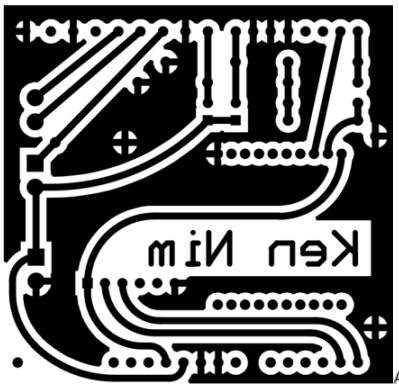
(b) Arduino, un microcontrôleur simple à utiliser.



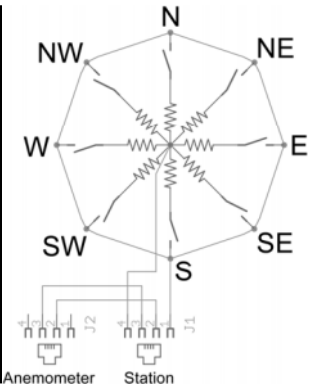
(c) Module XBEE, pour transmettre des données sans fil selon le protocole ZigBee.



(d) Carte de contrôle moteurs.

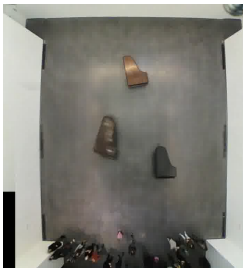


(e) Layout du shield Arduino conçu sur mesure. L'effet miroir est inhérent à la technologie utilisée pour graver le circuit imprimé.



(f) Schéma électrique de la girouette et de l'anémomètre utilisés.

Figure B-9 : Composants matériels utilisés pour offroad.



B.5 Résultats

Avec seulement trois mois pour la réalisation complète du projet, nous n'avons pas eu suffisament de temps pour faire des tests en conditions réelles. L'œuvre a été achevée le jour du vernissage de l'exposition, le 29 janvier 2014, et a tenu jusqu'à sa fin, le 4 mai suivant.

Le seul problème technique qui s'est posé à quelques reprises était une perte de masque de la vision, généralement due à des changements d'éclairage (eg. grillage d'une ampoule dans la salle).

Malgré la présence de bouton d'arrêt d'urgence, et de nos tentatives d'explications techniques à l'équipe du musée, cela a quand même causé la destruction partielle de l'un des pianos, lorsqu'il a percuté un escalier à pleine vitesse.

B.5.1 Qualité du mouvement

Il est difficile de décrire le mouvement final à l'écrit, mais plusieurs vidéos [2] [3] [4] ont été réalisées et montrent plus précisément le résultat final.

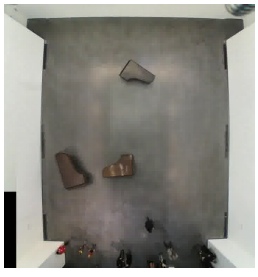
Le lecteur peut également voir une partie de la seconde vidéo en utilisant les folioscopes dans les angles inférieurs de ce document, et faire défiler les pages impaires dans l'ordre croissant puis les pages paires dans l'ordre décroissant pour mieux apprécier la qualité de ce mouvement.

B.5.2 Suites du projet

Nous avons par la suite été contactés par une équipe d'Airbus qui avait initialement jugé le projet impossible à réaliser compte tenu du budget et des délais. Ils ont souhaité que nous leur présentions nos méthodes et nos solutions techniques.

Jean-Paul Laumond, intrigué par cette application inattendue du problème classique du déménageur de pianos de (Schwartz, Sharir, et Hopcroft 1987), ainsi que par notre utilisation de la méthode des champs de potentiel, a également voulu nous rencontrer suite au vernissage de cette œuvre. Ceci a donc participé à la réalisation de cette thèse.

2. http://dai.ly/x1di66l?start=314
3. https://vimeo.com/87218362
4. https://youtu.be/cF4aS3LsHcg



Chapitre C

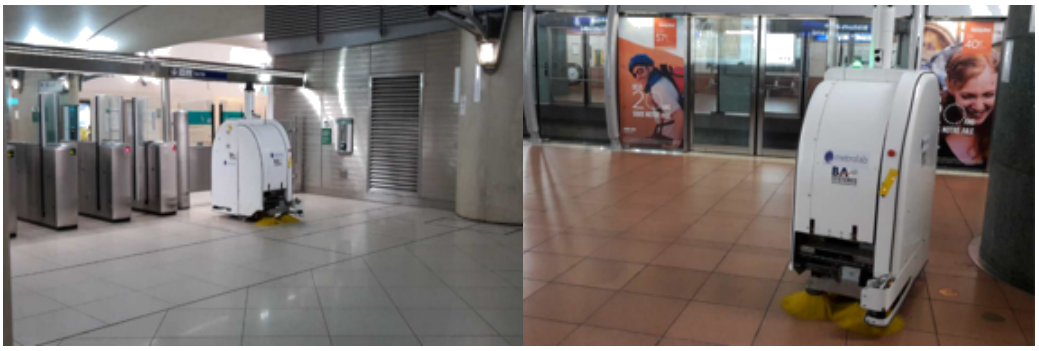
Robots mobiles à tourelle

Dans ce chapitre, nous faisons le rapport d'un projet de robotique réalisé lors de cette thèse en coopération avec BA Robotic Systems dans le cadre d'un projet avec Metrolab.

C.1 Introduction du projet LEMON

Le projet LEMON, bien plus prosaïque que les autres projets de cette partie, porte sur la conception d'un robot qui remplacerait une autolaveuse autoportée et son opérateur, dans des endroits tels qu'une station de métro ou un hôpital.

Le prototype d'un tel robot a été réalisé par la société BA Robotic Systems (figure C-1), et un accord a été conclu avec le LAAS-CNRS pour que nous concevions l'algorithme de planification de trajectoire.



(a) (b)

Figure C-1 : Illustrations du prototype du robot LEMON.

Ce projet a été réalisé au sein de la plateforme logicielle libre HPP [1], développée au sein de l'équipe Gepetto par Mirabel et al. (2016).

1. Humanoid Path Planner

Dans la suite de ce chapitre, nous verrons en premier lieu de quelle manière nous cartographions une zone afin de faciliter la planification de trajectoire dans la section C.2.

Le robot doit ensuite utiliser une brosse latérale rétractable afin de nettoyer les bords des murs, ce qui est détaillé dans la section C.3. Après cela, le robot doit nettoyer les aires libres d'obstacles grâce à sa brosse principale, comme nous le verrons dans la section C.4.

Enfin, nous expliquerons comment nous relions au mieux toutes ces portions de trajectoires dans la section C.5, puis quelles optimisations nous avons ajoutées dans la section C.6.

Pour conclure, nous exposerons nos résultats et leurs limitations dans la section C.7 et les perspectives dans la section C.8.

C.2 Création de la carte

Dans cette section, nous partons d'un plan généré par les capteurs laser du robot, constitué d'une liste d'obstacles.

Afin de réaliser des tests de performances dans un maximum de cas, ce plan peut provenir de différentes sources :

- un fichier texte généré par le prototype de BA Robotic Systems composé de deux nombres par lignes indiquant les coordonnées de chaque point obstacle trouvé par les lasers ;
- une image (par exemple un plan d'architecte) et une échelle ;
- en utilisant directement l'API de la librairie implémentée pour l'occasion, par exemple dans un contexte de production sur le robot.

L'objectif est d'extraire la position des murs de ce nuage de point, afin de générer des trajectoires de balayage des bords de ces murs. On a aussi besoin de pouvoir savoir où le robot peut passer ou non.

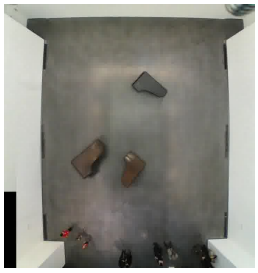
Pour cela, nous créons une classe `Bitmap` pour discrétiser la zone d'évolution. Ses bords sont ceux d'un rectangle dont la largeur, la longueur et l'orientation sont calculées automatiquement pour englober la surface à nettoyer et limiter la consommation en mémoire, et donc ainsi optimiser la vitesse d'exécution des algorithmes suivants.

Cette classe `Bitmap` est composée de `Pixels`, qui peuvent avoir plusieurs étiquettes :

- `OBSTACLE` si au moins un point obstacle est à l'intérieur ;
- `FREE` sinon ;
- `BOUNDARY` si c'est un `FREE` directement à côté d'un `OBSTACLE` ;
- `REACHABLE` si c'est un `FREE` accessible au robot ;
- `CLEANED` s'il est sur le chemin de balayage final.

D'autres `OBSTACLES` peuvent être ajoutés par un utilisateur pour définir des zones interdites circulaires ou polygonales. On en ajoute également tout autour de l'aire définie par le `Bitmap`.

Les zones `REACHABLE` sont calculées à partir de la position de départ du robot, ainsi que de ses dimensions physiques.



Une zone de tests a été réalisée dans les locaux de BA Robotic Systems, et nous en avons créé une carte (figure C-2).

Sur cette image, les `Pixels` en rouge contiennent des `OBSTACLES`, ceux en bleu sont des `BOUNDARY`, et ceux en verts représentent les `FREE` dont l'intensité varie avec la distance de Manhattan (Black 2006) aux `Pixels` de type `OBSTACLE`. Ce diagramme de Voronoı (1908) nous permet de savoir où le robot peut passer ou non.

Figure C-2 : Exemple de carte créée avec la classe `Bitmap`.

C.3 Trajectoires de nettoyage des bordures

Dans un premier temps, une fois que la carte de la zone est disponible, nous souhaitons utiliser la brosse latérale rétractable du robot afin de balayer le long des murs.

Dans cette section, nous expliquons comment nous détectons les segments de droites et les arcs de cercle dans cette carte, afin de donner des consignes simples au robot.

L'objectif est de fournir la première partie de la `Roadmap` destinée au robot. Cette `Roadmap` sera dans un premier temps composée d'une liste de segments définis par des paires (q_s, q_e) indiquant les positions initiales et finales nécessaires pour nettoyer chaque bordure.

C.3.1 Détection des segments de droites

L'obstacle le plus courant est un mur rectiligne. Il nous parait donc logique de commencer par chercher ce type d'obstacle. Pour trouver la liste des `Pixels` de type `BOUNDARY` qui sont sur des segments de droite, nous utilisons une transformée de Hough (Duda et Hart 1972), décrite dans l'algorithme C-1 et illustrée dans la figure C-3.

Algorithme C-1 Transformée de Hough

```
procedure HoughTransform(BOUNDARY, N_ρ, N_θ)
    H ← 0_{N_ρ, N_θ}                          ▷ Initialisation de la matrice H
    θ̄ ← linspace(−π, π, N_θ)                  et des ensembles discretisés
    ρ̄ ← linspace(0, sqrt(x_max^2 + y_max^2), N_ρ)   de coordonées polaires.
    for (x, y) ∈ BOUNDARY do
        for θ ∈ θ̄ do
            ρ ← arg min_{ρ∈ρ̄} (|x cos(θ) + y sin(θ) − ρ|)
            H_{ρ,θ} ← H_{ρ,θ} + 1             ▷ Incrémentation du coefficient obtenu.
        end for
    end for
    return H
end procedure
```

Cette transformée consiste à créer une matrice dite de Hough, $\mathcal{H}$, dont les dimensions sont la discretisation souhaitée de l'espace en coordonnées polaires (ρ, θ) : (N_ρ, N_θ).

Ensuite, chaque point (x, y) de l'ensemble `BOUNDARY` « vote » pour la liste des droites (ρ, θ) dont il pourrait faire partie.

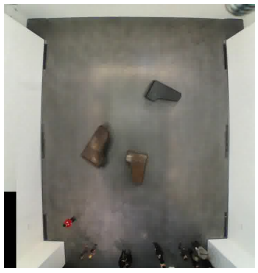
Le résultat de cette transformée de Hough est alors la matrice de Hough $\mathcal{H}$, dont les coefficients les plus importants indiquent les paramètres des droites les plus probables.

Afin de détecter les trajectoires qui balaient les bordures, on utilise ensuite le cadre logiciel HPP pour simuler un passage de ce robot suivant les droites fournies par la matrice de Hough dans les deux sens.

On garde alors les portions de ces trajectoires qui nettoient effectivement des `Pixels` de type `BOUNDARY` et qui ne présentent pas de collisions entre le robot et son environnement.

Enfin, on enlève les `Pixels` ainsi nettoyés de l'ensemble de ceux de type `BOUNDARY` et on recommence autant de fois que nécessaire.

L'extraction complète de ces trajectoires est explicitée dans l'algorithme C-2, et dans l'annexe 1, et illustrée dans la figure C-4.



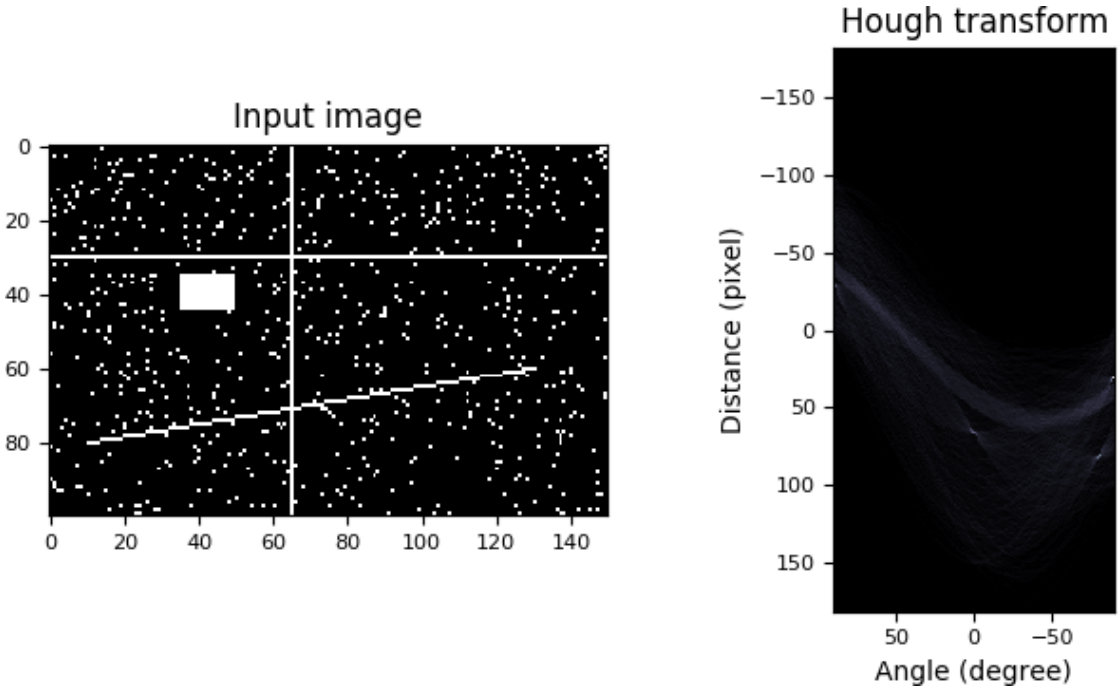
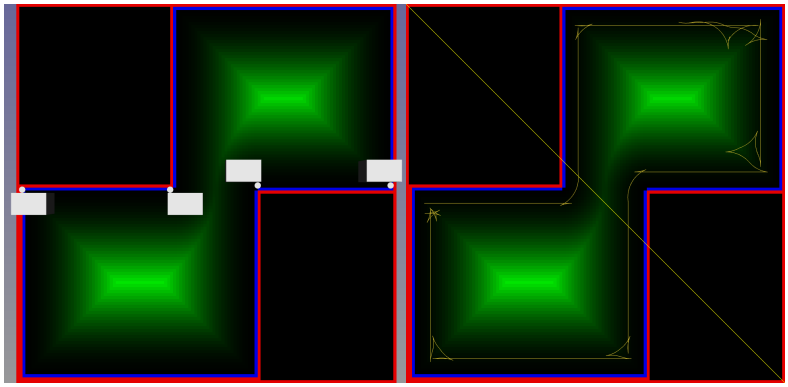


Figure C-3 : Illustration graphique de la transformée de Hough. Dans l'image de droite, on voit trois points blancs, dont les coordonnées correspondent aux paramètres (ρ, θ) des trois droites de l'image de gauche.



(a) Détection des segments à balayer suivants une droite déterminée par transformée de Hough. On simule un balayage de cette droite dans les deux sens, puis on ne garde que les portions qui nettoient des bordures et ne sont pas en collision.

(b) Trajectoire complète de balayage des murs. Les segments des droites sont reliés entre eux par des trajectoires de Reeds et Shepp, comme expliqué dans la section C.5

Figure C-4 : Illustration de la détection des segments de droites à balayer pour nettoyer le long des murs

Algorithme C-2 Détection des segments de droite à balayer pour nettoyer le long des murs

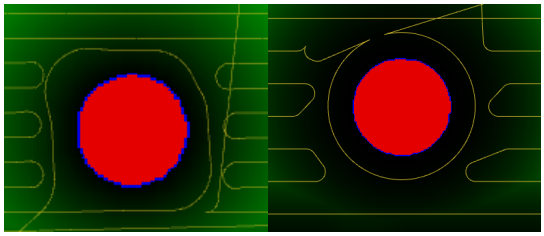
```
1 : while BOUNDARY ≠ ∅ do
2 :     for (ρ, θ) ∈ arg max(HoughTransform(BOUNDARY, N_ρ, N_θ)) do
3 :         followLine(ρ, θ, +1)
4 :         followLine(ρ, θ, −1)
5 :     end for
6 : end while
7 : procedure followLine(ρ, θ, σ)
8 :     start ← false
9 :     for q ∈ line(ρ, θ, σ) do                ▷ σ est le sens de parcours à suivre. line est explicité en annexe 1.
10 :        valid ← chkPos(q)                   ▷ Place le robot pour que la brosse soit positionnée en q et vérifie les collisions et qu'on nettoie bien des BOUNDARY
11 :        if valid and ¬start then
12 :            start ← true
13 :            q_s ← q                         ▷ Début d'une trajectoire
14 :        else if start and ¬valid then
15 :            start ← false
16 :            addLane(q_s, q)                 ▷ Ajoute la trajectoire à la Roadmap et supprime les BOUNDARY nettoyées
17 :        end if
18 :    end for
19 : end procedure
```

C.3.2 Détection des arcs de cercle

Les environnements dans lesquels le robot LEMON est destiné à évoluer peuvent aussi comporter des obstacles circulaires. Par exemple, une station de métro peut suivre une voie ferrée courbée. On peut également voir des piliers ronds.

Ces arcs de cercle pourraient dans certains cas être approximés par des suites de segments, mais en pratique les résultats n'étaient pas satisfaisants, comme le montre la figure C-5a. Nous avons donc ajouté à l'algorithme présenté dans la section précédente une phase de transformée de Hough circulaire.

Un opérateur doit alors entrer la liste des rayons de cercles qui sont présent dans un environnement, et, pour chacun de ces rayons, nous construisons une matrice de Hough dont les coefficients correspondent aux coordonnées (x, y) du centre d'un cercle d'un tel rayon à la place des coordonnées (ρ, θ) d'une droite.



(a) Suivi d'un cercle à partir de segments de droites.

(b) Suivi d'un cercle par transformée de Hough circulaire.

Figure C-5 : Détection de cercles : transformées de Hough en (ρ, θ) à gauche et (x, y, r) à droite.

C.4 Trajectoires de nettoyage des surfaces

Dans cette section, nous présentons la stratégie de balayage des surfaces retenue. Ce balayage de surface intervient après le balayage des bordures.

De la première transformée de Hough pour la détections des segments de droites (cf. section C.3.1), nous extrayons la droite de la transformée de Hough ayant le plus de `Pixels`.

Depuis cette droite principale, nous traçons des segments parallèles qui couvrent la surface, tout en prenant toujours soin d'éviter les collisions avec l'environnement grâce à la procédure followLine décrite dans l'algorithme C-2.

Pour chaque segment ainsi généré, nous ajoutons deux trajectoires dites « symétriques » à la `Roadmap`, qui correspondent aux deux sens de parcours du segment par la brosse principale du robot.

Il suffit alors que l'une des deux trajectoires de la paire soit suivie par le robot pour considérer que la paire est traitée.

C.5 Génération de la trajectoire finale

Une fois que les trajectoires de balayage des bordures droites (section C.3.1) et courbes (section C.3.2) ainsi que les trajectoires symétriques de nettoyage des surfaces (section C.4) sont entrées dans la `Roadmap` sous forme de paires de configuration (q_s, q_e), il reste à générer la trajectoire finale.

Pour cela, il faut commencer par déterminer l'ordre de parcours des trajectoires de balayage des bordures, puis celui des trajectoires de nettoyage des surfaces. La trajectoire finale consiste alors à relier la suite de configurations obtenue par des trajectoires dites de Reeds et Shepp (1990), composées de segments de droites et d'arcs de cercles.

Dans le but de déterminer l'ordre de parcours des trajectoires, nous avons utilisé un algorithme glouton. Ainsi, à partir de la position initiale du robot, nous recherchons la configuration de départ de la trajectoire de suivi balayage des bordures la plus proche, au sens de la distance de Reeds and Shepp, et suivant la méthode de tir aléatoire RRT-Connect de Kuffner et LaValle (2000).

Ensuite, nous repartons de la configuration finale associée, et recommençons l'opération jusqu'à ce que toutes les trajectoires de balayage des bordures soient effectuées. Le même procédé est alors répété à partir de la configuration finale de la dernière trajectoire de balayage des bordures pour les trajectoires de nettoyage des surfaces.

Les principaux algorithmes de cette section, qui sont la planification d'une trajectoire de Reeds and Shepp ansi que la méthode RRT-Connect, sont directement implémentés dans HPP.

C.6 Optimisation

Les premiers résultats de la méthode présentée dans les sections précédentes ont permis de remplir le cahier des charges. Cependant, il est facile pour un observateur humain d'imaginer de meilleures options que la trajectoire finale générée. Regarder le robot suivre sa trajectoire est donc un peu frustrant.

Les trajectoires générées par l'algorithme de Reeds et Shepp peuvent parfois surprendre, comme le montre l'exemple de trajectoire généré de la figure C-6, et l'algorithme de tir aléatoire peut facilement rater une transition qui semble simple à un humain, malgré notre utilisation d'une méthode de raccourcis aléatoires.

Notre objectif est alors de raccourcir au maximum la longueur de la trajectoire finale, quitte à relaxer un peu la contrainte de couverture de la surface.

Pour cela, nous remarquons que dans le cas classique d'un angle droit entre deux murs, le robot balaye le premier mur jusqu'à la collision avec le second, puis fait marche arrière, va se placer parallèlement au second, refait marche arrière jusqu'à la collision avec le premier, puis commence le balayage du second.

Cette manœuvre parait naturelle si l'on cherche à tout nettoyer au mieux, mais demande beaucoup de temps. On nous demande alors de tronquer la fin de la première trajectoire de balayage des bordures et le début de la seconde.

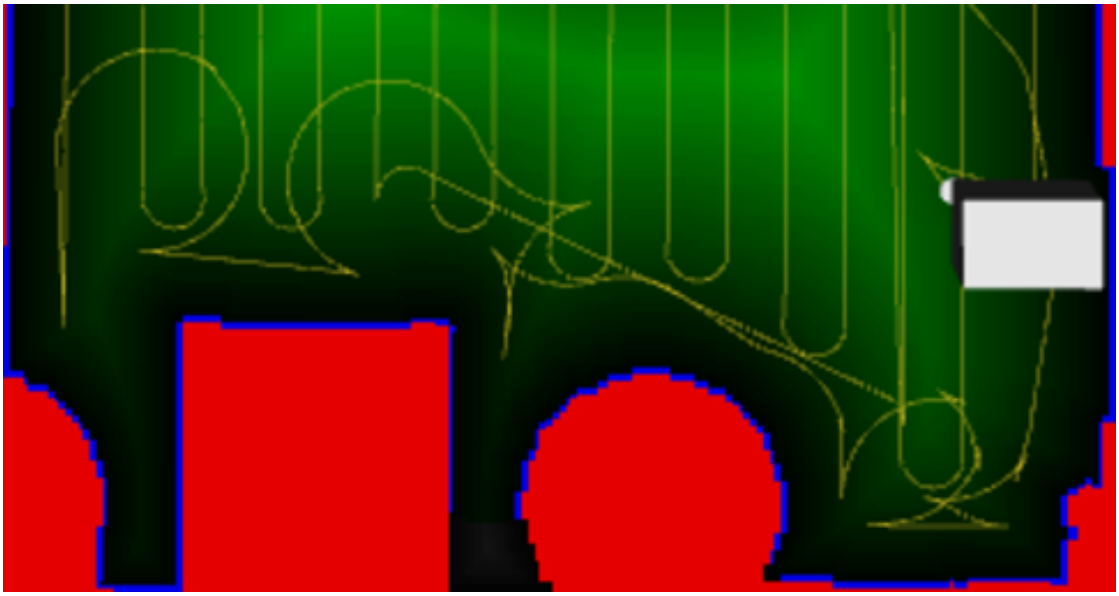


Figure C-6 : Exemple de trajectoire de suivi des murs générée par l'algorithme de Reeds et Sheep étonnamment longue.

Nous relions ensuite ces deux configurations par une trajectoire dite de Dubins (1957), qui est optimale pour un robot mobile à tourelle ayant un rayon de giration borné et ne se déplaçant qu'en marche avant [2], évitant ainsi les manœuvres.

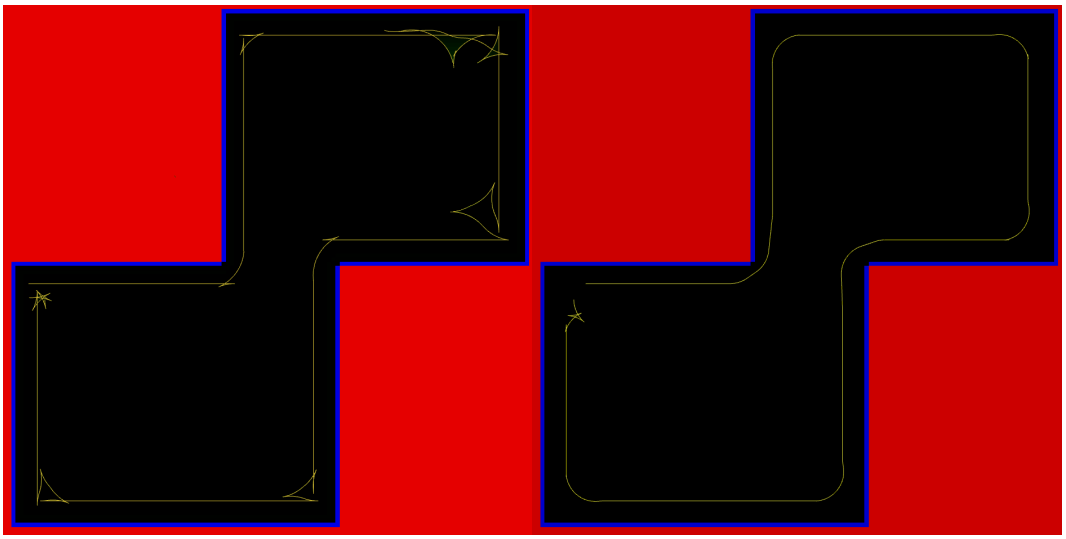
Cette stratégie est illustrée sur la figure C-7

Dans le cas d'un angle droit, cette solution donne les bons résultats auxquels nous nous attendions. Mais en pratique, il faut garder à l'esprit que cela ajoute un paramètre supplémentaire à régler, à savoir la longueur tronquée sur chaque trajectoire (voire deux si on veut des longueurs tronquées différentes).

Or la valeur optimale de ce paramètre dépend grandement d'autres paramètres, et notamment de l'angle entre deux murs successifs, de la longueur initiale des trajectoires avant troncage, et de la distance nécessaire entre le robot et le mur pour pouvoir tourner directement sans engendrer de collision.

Enfin, cette technique ne fonctionne pas sur toutes les transitions, comme le montre la figure C-8.

2. En pratique, vu que le robot balaye en marche arrière, nous parlons plutôt d'une trajectoire de Snibud.



(a) Avant : Trajectoires complètes de balayage des murs, reliées par une trajectoire de Reeds et Sheep.

(b) Après : Trajectoires tronquées de balayage des murs dans l'angle droit, reliées par une trajectoire de Dubins. On remarque que certaines trajectoires sont inutilement tronquées, puisqu'elles n'aboutissent pas sur un obstacle.

Figure C-7 : Illustration du raccourci utilisé. Cette stratégie est surtout utile dans les angles droits fermés, mais elle réduit inutilement les segments de balayage des bordures dans d'autres cas.

Figure C-8 : Amélioration de l'exemple de la figure C-6. Dans certains cas, la méthode de Dubins raccourcit grandement les trajectoires, mais elle n'est pas toujours applicable.

C.7 Résultats

Dans le cadre de ce projet, nous avons développé des classes et méthodes spécifiques au problème posé, nous les avons intégrées avec les classes et méthodes déjà disponibles dans HPP, puis nous avons réglé empiriquement les différents paramètres afin d'obtenir des trajectoires le plus satisfaisantes possible sur les exemples fournis ainsi que sur certains cas d'école.

Le résultat est un programme qui, dans la plupart des cas, produit une trajectoire couvrant bien la surface à nettoyer, mais qui souvent produit des trajectoires plus longues que nécessaire. L'optimalité de ces trajectoires reste un problème ouvert.

C.7.1 Limitations

Comme nous l'avons vu dans la section C.3, nous recherchons pour l'instant uniquement des segments de droite, voire des arcs de cercles dont le rayon doit être connu a priori.

Cependant, dans certains cas où plusieurs petits segments sont proches les uns des autres, les transformées de Hough peuvent rater des segments de droites pourtant évidents pour un humain.

Aussi, cette transformée de Hough implique un choix des paramètres de discrétisation (N_ρ, N_θ). Pour pouvoir bien construire une carte, il est donc souvent nécessaire de bien comprendre la théorie de la transformée afin d'adapter ces paramètres à une zone donnée.

Sans cela, en utilisant des paramètres par défaut, des points alignés peuvent se retrouver sur des droites différentes pour la transformée. Cet effet peut être amplifié par le bruit inhérent à toute carte construite à l'aide de capteurs.

En conséquence, cela implique des situations difficilement compréhensibles pour un observateur extérieur où le robot balaye plusieurs fois la même bordure.

Un autre problème vient de notre utilisation du RRT-Connect, qui peut poser des problèmes dans le cas où la meilleure trajectoire suivante peut éventuellement être à une grande distance. Cette situation s'est notamment présentée dans l'un des exemples fournis, illustré dans la figure C-9.

Figure C-9 : Exemple d'un environnement vaste (de l'ordre de quelques centaines de mètres) et non-convexe. Relier deux trajectoires de nettoyage des surfaces dans cet exemple peut s'avérer complexe, puisque l'algorithme doit comprendre qu'il doit aller chercher la suite derrière un mur.

C.8 Travaux futurs

Lors de ce projet LEMON réalisé en coopération avec BA Robotic Systems, nous avons tenté de répondre à un cahier des charges dans un temps limité par le client final. Nous avons donc livré un logiciel fonctionnel, mais sous-optimal.

Dans cette section, nous discutons des possibilités d'extension de ce travail afin de rendre les trajectoires générées plus proches de celles qui seraient réalisées par un être humain, voire meilleures.

C.8.1 Suivi des murs

Dans un premier temps, nous avons choisi de détecter des primitives géométriques afin de suivre les bordures.

Une autre approche possible consisterait à extraire une trajectoire directement de la forme des contours formés par les `Pixels` de type `BOUNDARY`, en les reliant avec une trajectoire de Dubins. Certains `Pixels` ne seraient pas atteignables, mais comme nous l'avons vu dans la section C.6, ce n'est finalement pas un problème.

Pour cela, l'idéal serait de pouvoir profiter du débattement de la brosse latérale pour mieux gérer la distance entre le robot et le mur, alors que jusqu'ici nous n'avons considéré qu'elle n'avait que deux états possibles : sortie pour le balayage des bordures et rentrée pour le nettoyage des surfaces.

C.8.2 Direction du nettoyage des surfaces et découpage de la zone principale

Dans cette première version, nous avons choisi de nettoyer toutes les surfaces selon la même direction, et en utilisant la direction donnée par le plus grand coefficient de la première transformée de Hough. Dans certains cas, cette direction n'est pas la meilleure, comme le montre le contre-exemple sur la figure C-10.

Par ailleurs, il peut être bon de découper la zone principale en sous-zones présentant des directions générales différentes. Un algorithme de découpage d'une zone non-convexe a déjà été développé au sein de l'équipe Gepetto dans le cadre du travail d'engins agricoles dans (Tran, Taïx, et Souères 2005), dont est extrait la figure C-11.

C.8.3 Ordre de parcours des portions de trajectoires

Un dernier axe d'amélioration serait d'optimiser l'ordre de parcours des portions de trajectoires de balayage des bordures et de nettoyage des surfaces.

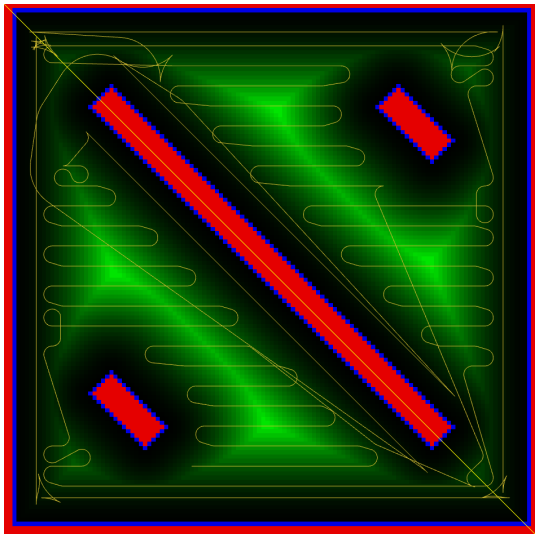


Figure C-10 : Exemple d'une salle où la bonne direction de balayage serait en diagonale, alors que le mur le plus long est horizontal.

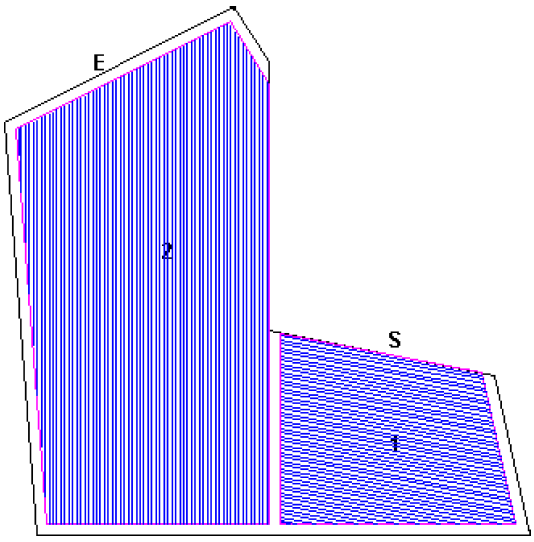


Figure C-11 : Exemple d'algorithme de planification de trajectoires pour engin agricole dans des polygones non convexes.

Dans un premier temps, une modification d'algorithmes classiques de recherche opérationnelle pourrait permettre de mieux prendre en compte la possibilité de parcourir les trajectoires de nettoyage des surfaces dans un sens ou dans l'autre, ainsi que de pallier le problème de la recherche de la trajectoire suivante si sa configuration initiale est trop loin pour notre RRT-Connect.

Dans un second temps, l'idéal serait de pouvoir découper certaines trajectoires de balayage des bordures pour aller balayer un obstacle proche d'un long mur, comme dans la figure C-12.

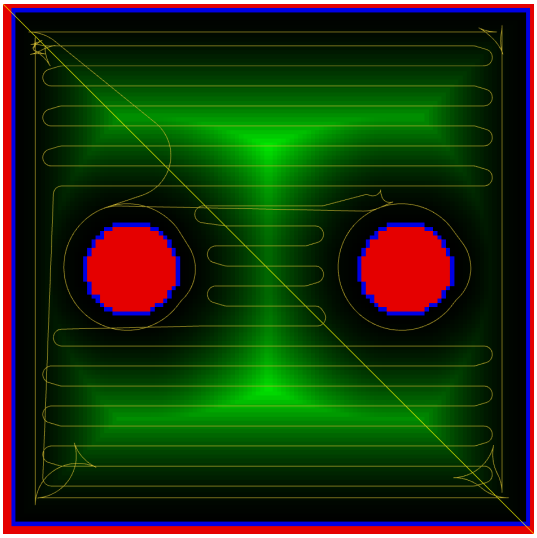


Figure C-12 : Dans cet exemple, il serait intéressant de pouvoir balayer les bords des piliers ronds en s'arrêtant à mi-chemin sur les trajectoires de balayage des murs verticaux.

Chapitre D

Robots mobiles tri-tourelles

Dans ce dernier chapitre sur la robotique mobile, nous faisons le rapport d'un projet réalisé pour l'Institut Français, en coopération avec les entreprises BA Systèmes et Ubisense. Il a consisté à implémenter le contrôle et la planification de robots mobiles omnidirectionnels.

Comme pour le projet offroad présenté dans le chapitre B, les robots constituent une œuvre d'art de l'artiste Céleste Boursier-Mougenot. Cette fois-ci, nous dotons de moyens de locomotion trois pins sylvestres, de plusieurs mètres de haut et plusieurs tonnes.

D.1 Introduction

La Biennale de Venise (Di Martino 2005) est la plus importante exposition d'art contemporain au monde. Elle a lieu une fois tous les deux ans depuis 1895. L'exposition est répartie dans différents lieux à Venise, parmi lesquels les Giardini comportant 90 pavillons, étant chacun dirigés par une nation.

Pour la 56ème édition en 2015, l'Institut Français, en charge du pavillon français, a choisi le projet Rêvolutions de l'artiste Céleste Boursier-Mougenot et la commissaire Emma Lavigne. Ce projet incluait l'œuvre transhumus dont nous parlons dans ce chapitre.

Dans transhumus, l'artiste a voulu créer une chorégraphie de trois arbres mobiles se déplaçant en fonction de leur métabolisme, et a choisi la variation du flux de leur sève lorsqu'ils passent de l'ombre à la lumière. Ce spectacle a eu lieu de mai à novembre, six jours par semaine, huit heures par jour.

La figure D-1 présente respectivement l'idée initiale et la réalisation finale du projet. Neuf mois séparent les deux images. L'objectif de ce chapitre est de faire le rapport sur la recherche et le développement qui ont été conduits pendant cette période, depuis notre première réunion avec l'artiste jusqu'au vernissage de l'exposition le 5 mai 2015.

(a) Vue d'artiste. (b) Résultat final.

Figure D-1 : Des spécifications à la réalisation. Neuf mois séparent les deux images.

Comment transcrire la dimension poétique de ce projet en des termes technologiques ? La question s'adresse à tous les composants d'un système robotique, de la conception de machines capables de déplacer trois tonnes (l'arbre, ses racines, et la terre nécessaire), l'architecture de perception qui doit combiner des capteurs égocentriques et allocentriques, et le système de génération de mouvement qui doit retranscrire le critère de qualité du mouvement défini par l'artiste en termes de lissage de trajectoire et de vitesse.

Notre contribution dans ce projet a été de sélectionner les fournisseurs (section D.2), développer un architecture logicielle comprenant une interface utilisateur pour l'équipe du pavillon (section D.3) et s'occuper de la gestion du projet (section D.5).

Notre seconde contribution a été de proposer à l'artiste des stratégies de génération et de planification du mouvement donnant l'impression que les arbres errent en totale autonomie (section D.4).

D.2 Spécifications et solutions techniques

L'ambition poétique de ce projet est de libérer les arbres de leur immobilité et de les laisser explorer le monde par eux-mêmes. Cette ambition impose la création de connections originales entre des éléments naturels et technologiques.

Cette inspiration continue le travail de Céleste Boursier-Mougenot de ces dernières années (MacAdam 2015).

La première étape de ce projet a été d'ouvrir un processus de dialogue avec l'artiste. Nous avons ainsi dû définir le type d'arbres utilisés, leur occupation de l'espace dans les Giardini, ce que signifie pour un arbre de bouger « de lui-même », et quelles sont les conditions de développement et d'exploitation.

D.2.1 Spécifications

Les spécifications qui suivent dans cette section sont issues de ce processus de dialogue avec l'artiste.

D.2.1.1 Arbres

Les arbres retenus pour ce projet sont des pins sylvestres, dont la floraison en mai coïncide avec l'ouverture de la Biennale. Nous avons donc pris trois de ces arbres pour l'œuvre à Venise, plus un pour nos tests au LAAS-CNRS.

Ils font environ cinq mètres de haut. Ils sont vivants, et se déplacent avec leur motte de terre pour un total d'environ trois tonnes.

D.2.1.2 Zones

Deux de ces arbres évoluent dans l'espace devant les pavillons français, anglais, canadien et allemand. Ils doivent rester à l'intérieur d'une zone de 300 mètres carrés bien définie (figure D-2). Le sol de cette zone est composée de terre battue et de graviers, donc la dureté dépend des conditions météorologiques.

Le troisième arbre se déplace lui dans le pavillon français, dans une zone de 50 mètres carrés pourvue d'un sol en béton.

Ces deux zones ne comportent pas d'obstacles permanents, mais sont des aires de passage de visiteurs sans contraintes spécifiques de sécurité. Ces visiteurs sont autorisés à approcher et toucher les arbres.

D.2.1.3 Origine du mouvement

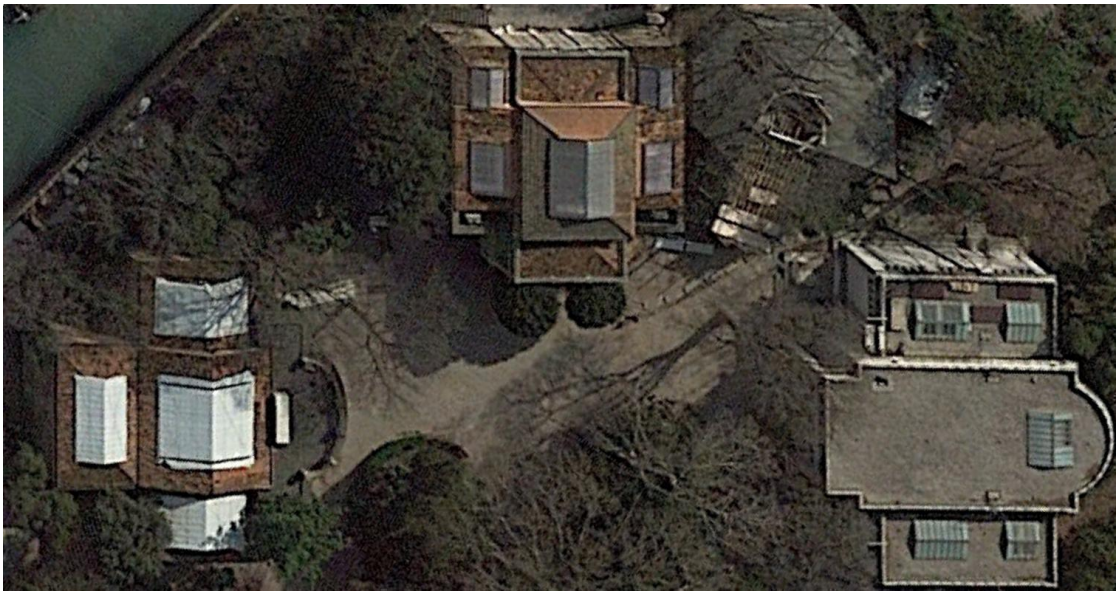
Les arbres sont des êtres vivants, dont le métabolisme dépend des conditions environnementales et météorologiques. Leur mouvement doit dépendre des variations de leur état interne. Ceci est l'une des principales problématiques de ce projet : cette œuvre cherche à révéler l'invisible état interne des arbres.

Il est également nécessaire de restreindre le mouvement des arbres aux zones définies dans la section précédente, et d'éviter les collisions entre les deux arbres qui partagent l'esplanade.

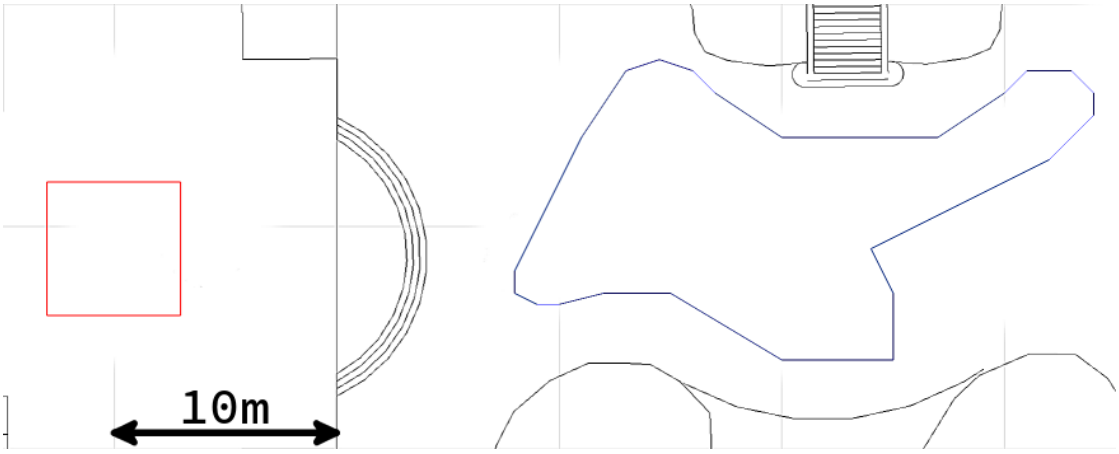
D.2.1.4 Qualité du mouvement

L'artiste tient à ce que les arbres bougent extrêmement lentement. Avec une vitesse inférieure au mètre par minute, leurs mouvements doivent être à peine perceptibles. Les arbres doivent bouger sans avoir une direction privilégiée : ils devraient être holonomes.

Aucun bruit ne doit être audible pour les visiteurs, les arbres doivent donner l'impression de se mouvoir par lévitation. La base de la motte de terre doit être à moins de 5cm du sol.

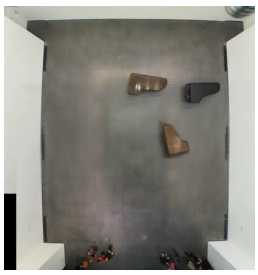


(a) Vue aérienne des pavillons français (sur la gauche), anglais (en haut au centre), canadien (en haut à droite) et allemand (sur la droite) autour d'une esplanade dans les Giardini.



(b) Modèle géométrique des zones d'évolution, avec une grille dont les carreaux font 10 mètres pour l'échelle. Cette image est issue de l'interface utilisateur servant à monitorer le déplacement des arbres.

Figure D-2 : Vue aérienne de la partie des Giardini qui nous intéresse. Le pavillon français est le bâtiment sur la gauche. Un arbre se déplace dans la salle principale de ce pavillon, et les deux autres se partagent l'esplanade commune aux pavillons anglais, canadien et allemand.



D.2.1.5 Contraintes opérationnelles

Le projet doit être réalisé en six mois sans aucune extension possible de date limite, fixée à l'ouverture de la biennale, le 5 mai 2015, avec seulement deux semaines de tests sur site.

L'œuvre doit fonctionner jusqu'au 22 novembre 2015, à raison de 8 heures par jour et 6 jours par semaine. Elle doit être utilisable par l'équipe de non-spécialistes du pavillon français en une journée de formation.

D.2.2 Solutions technologiques

Ces spécifications ont conduit aux solutions technologiques suivantes.

D.2.2.1 Conception de la plate-forme

Des plate-formes robotiques sur mesure ont été conçues par BA Système, une société spécialisée dans la conception et la production d'AGV (Automatic Guided Vehicles) pour la logistique.

Chaque plate-forme est composée d'un bac octogonal supporté par trois tourelles figure D-3, composées d'une roue motrice électrique, orientable autour de son axe central. Les roues peuvent donc tourner sur place.

Figure D-3 : AGV dans les locaux de BA Systèmes.

Le bruit des moteurs électriques de traction et d'orientation sont inaudibles.

Cela ne rend pas la plate-forme holonome, mais bien omnidirectionnelle, de type $(1, 2)$ dans la classification de Campion, Bastin, et D'Andrea-Novel (1996). On peut donc contrôler la plate-forme dans chacune des trois directions de $\mathbb{R}^2 \times S^1$. Le modèle de contrôle utilisé est présenté dans la section D.4.1.

Les mottes des arbres sont insérées dans les bacs, et des coquilles synthétiques imitant la terre et les racines encapsulent les AGVs. Les coques synthétiques contribuent également à la suppression totale de légers bruits qui pourraient provenir des moteurs.

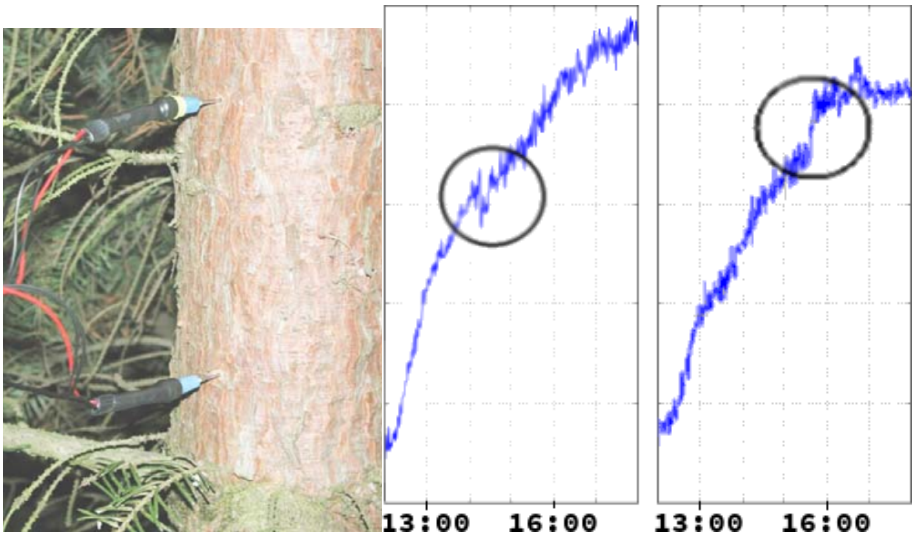
D.2.2.2 Capteur du métabolisme des arbres

La sonde Granier (Granier 1987) (figure D-4a) est l'une des techniques les plus courantes pour mesurer le métabolisme d'un arbre (Lu, Urban, et Ping 2004). La sonde est fondée sur un principe de dissipation thermique, et est constituée de deux aiguilles.

Une aiguille chauffée est placée dans l'aubier au-dessus d'une aiguille neutre. Quand la vitesse de la sève est faible, la chaleur de l'aiguille chauffée est peu dissipée, et la différence de température entre les deux aiguilles est donc élevée. Cette différence diminue avec l'augmentation de la vitesse de la sève.

Sur le pin sylvestre installé au LAAS-CNRS, nous avons vérifié que la sensibilité de ces sondes était suffisante pour observer une différence de luminosité (figure D-4b).

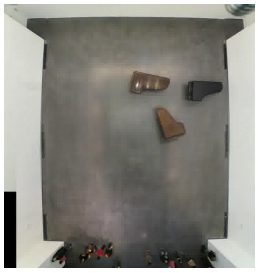
Cette luminosité change avec les conditions atmosphériques, ainsi que lorsque que l'arbre se déplace entre l'ombre et la lumière. Nous installons alors dans chaque arbre trois sondes dans différentes directions. Les mesures données par les sondes sont ensuite utilisées comme des entrées pour la génération de mouvement.



(a) Capteurs de flux de sève : les sondes Granier sont composées de deux aiguilles thermocouples.

(b) Déplacement d'un arbre de l'ombre à la lumière et vice-versa : mesures d'une sonde Granier sur l'arbre de test au LAAS-CNRS les 8 et 9 avril 2015, entre 12 :00 et 18 :00, lorsque l'on déplace l'arbre aux alentours de 15 :00.

Figure D-4 : Mesures du flux de sève : ces expériences nous ont permis de vérifier cette solution technique par rapport à nos spécifications. Nous avons constaté que la réaction d'un arbre passant de l'ombre à la lumière était mesurable et donc exploitable en tant qu'entrée de notre système, puisque le métabolisme des arbres peut influer sur leur déplacement.



D.2.2.3 Localisation

Les aires d'évolution sont définies à l'intérieur et à l'extérieur du pavillon français. Pour localiser les AGV, nous avons donc opté pour une technologie UWB (Ultra Wide Band) développée par la société Ubisense. Cette technologie nous apporte un bon compromis entre la précision et la discrétion.

Les bâtiments autour des zones d'évolution des arbres sont équipés d'antennes assurant une couverture complète (figure D-5). Les arbres sont équipés de plusieurs petits récepteurs cachés dans les branches.

Une antenne ayant un rôle de maître interroge régulièrement les récepteurs un à un. Lorsque celui-ci répond, chaque antenne mesure l'angle d'arrivée du signal ainsi que la durée de la transmission.

Les données de triangulation, de trilatération et les informations connues sur la position relatives de plusieurs récepteurs installés dans un même arbre (considéré rigide) sont ensuites filtrées.

Une précision de l'ordre de 15 centimètres est obtenue avec un bon niveau de qualité.

Figure D-5 : Implantation des antennes UWB Ubisense dans les Giardini.

D.3 Architecture logicielle

> Designing a robot architecture is much more of an art than a science.
> — Handbook of Robotics, Siciliano et Khatib (2016)

Dans la section précédente, nous avons détaillé notre choix de fournisseurs de solutions matérielles. Nous étions également en charge de la gestion des principaux développements logiciels. La figure D-6 récapitule les différents composants du système et les flux de données entre eux.

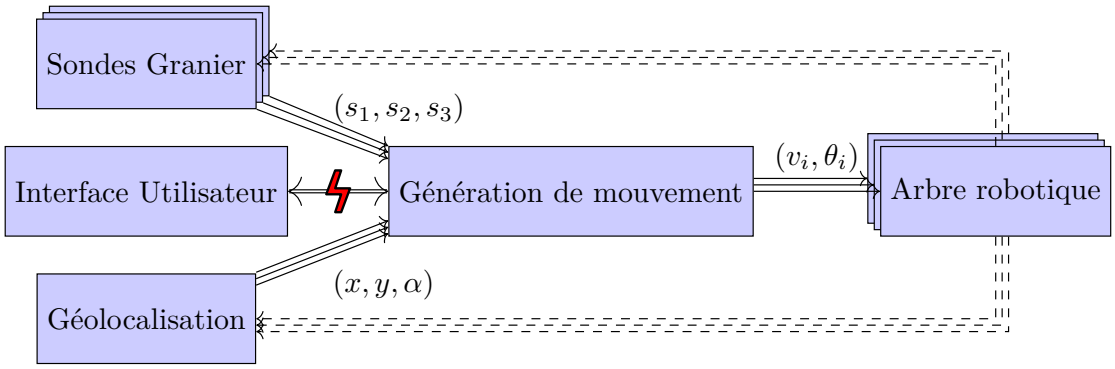


Figure D-6 : Architecture logicielle : chaque arbre a trois sondes Granier qui sont utilisées par le planificateur de trajectoire. Le planificateur de trajectoire récupère également la position et l'orientation actuelle de chaque robot grâce au système de géolocalisation, puis calcule les vitesses de traction et d'orientation de chaque tourelle de chaque AGV. Un utilisateur peut aussi directement donner des consignes au planificateur de trajectoire lorsque c'est nécessaire. Les variables (s_1, s_2, s_3), (x, y, α) et (v_i, θ_i) sont explicitées dans la section D.4.

Les logiciels ont été développés pour être le plus modulaire possible. Il est donc facile de passer des cas de tests en simulation aux cas de production. Cette modularité permet également de surmonter la diversité de technologies utilisées par nos fournisseurs, puisque des convertisseurs sont simples et rapide à coder.

En effet, les sondes Granier fournissent des valeurs toutes les 30 secondes sur une liaison série, le logiciel des AGV attend des commandes ASCII sur un socket TCP, et la suite logicielle de géolocalisation est bâtie sur une technologie .NET qui ne peut être étendue que via des modules d'un cadre logiciel propriétaire dont la fréquence d'actualisation peut aller de 1 à 100 hertz.

Par conséquent, nous avons utilisé une architecture logicielle fondée sur la librairie de messagerie ZeroMQ (Hintjens 2013), qui peut être utile pour tous nos canaux de communication. Cette librairie est disponible dans plusieurs langages de programmation, et fournit une abstraction aux problématiques de connections / déconnections des sockets sous-jacentes. Enfin, elle implémente des schémas de communication classiques tels que Client/Server, Publish/Subscribe (PUB/SUB) et PUSH/PULL.

L'idée générale de notre architecture est d'avoir un planificateur de trajectoire qui est un composant central, et qui est capable d'effectuer des Pull depuis les entrées et des Publish vers les sorties. Ainsi les autres composants peuvent se mettre à récupérer des données via des Subscribe et en envoyer grâce à des Push à tout moment, sans perturber le processus principal. Certaines fonctionnalités périphériques peuvent également être ajoutées à la volée en étant à la fois un Subscriber et un Pusher. Cette architecture est plus précisément détaillée dans l'annexe 2.

Une interface utilisateur graphique fondée sur des technologies web a également été développée pour aider l'équipe du pavillon à voir la situation actuelle, et gérer les éventuels problèmes qui pourraient survenir (figure D-7).

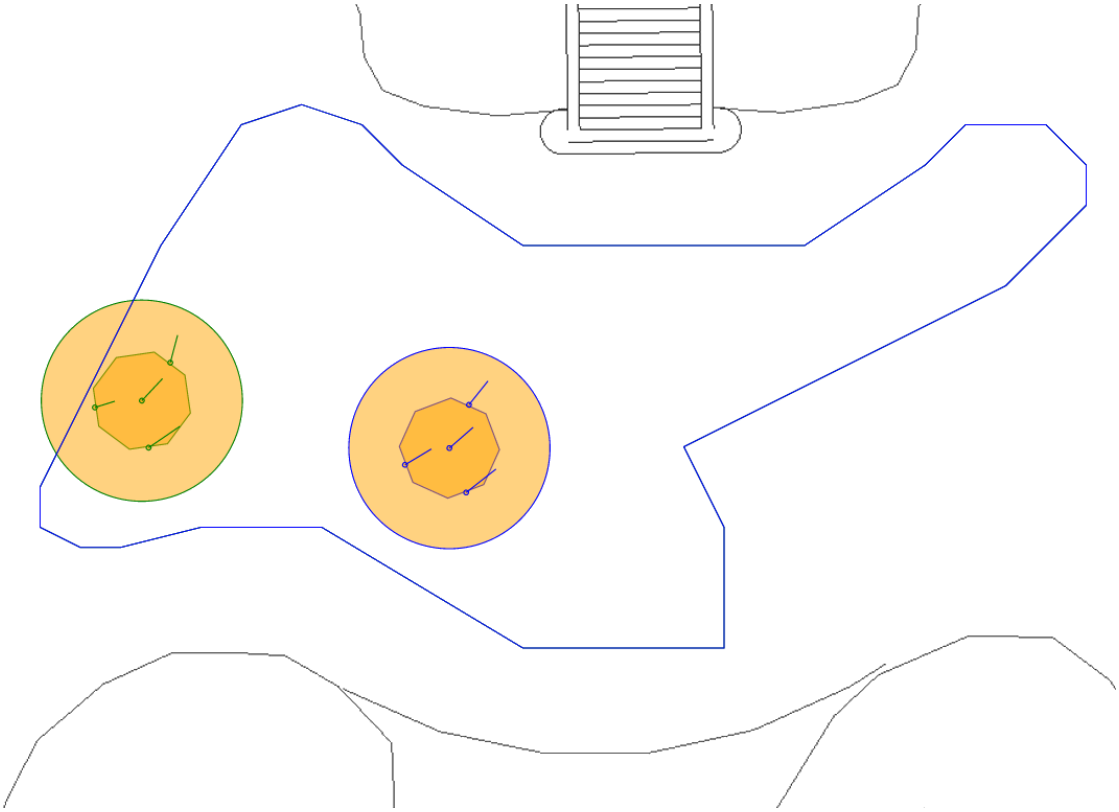


Figure D-7 : Exemple d'utilisation de l'interface utilisateur web, affichant la situation actuelle : deux AGVs se déplacent sur l'esplanade, et on peut voir leur position et orientation, ainsi que l'orientation et la vitesse de traction des tourelles.

Dans des cas plus complexes, ceci nous permet également de travailler sur l'installation à distance, pour éviter d'avoir besoin d'ingénieurs sur place. Enfin, pendant la période de développement, cette interface a également permis à l'artiste de se faire une idée du mouvement des arbres au fur et à mesure de notre avancement.

NB : En temps normal, cette interface utilisateur est inutile, vu que le système est entièrement autonome.

D.4 Planification de mouvement et contrôle

Dans cette section, nous explicitons la manière dont est généré le mouvement des plate-formes robotiques supportant les arbres.

Pour cela, nous commençons par la modélisation mathématique de la gestion des orientations et vitesses de traction des tourelles, dans la section D.4.1, puis nous voyons comment nous générons le mouvement à partir des sondes Granier dans la section D.4.2 et nous ajoutons certaines fonctions de lissage dans la section D.4.3. Pour finir, nous expliquons comment nous faisons en sorte que l'arbre « choisisse » sa destination, dans la section D.4.4.

D.4.1 Modélisation de la plate-forme

Comme nous l'avons vu dans la section D.2, l'artiste désire un mouvement omnidirectionnel. Nous avons donc besoin de trois variables d'entrée, ce qui correspond à la classe de robot mobile $(1, 2)$.

Puisque l'artiste désire que le robot n'ait pas une direction privilégiée du mouvement, nous choisissons un système de coordonnées polaires en (v, θ, ω), où :

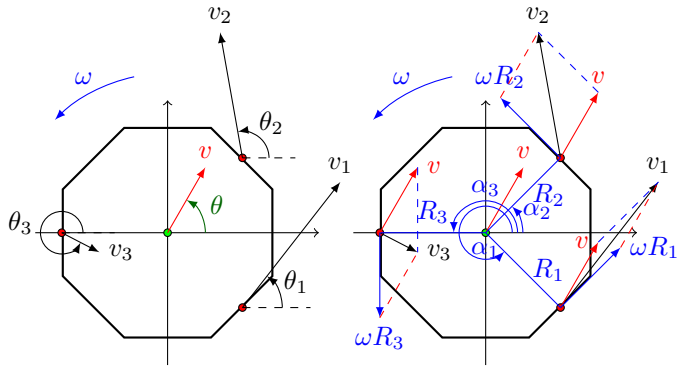
- $\theta \in [-\pi, \pi[$ est la direction dans laquelle se déplace le centre de l'AGV ;
- $v \in [0, 1]$ est sa vitesse linéaire dans la direction θ ;
- $\omega \in [-1, 1]$ est sa vitesse angulaire.

Avec ce système de coordonnées, on peut facilement calculer la sortie demandée par l'AGV, qui est la direction θ_i et la vitesse de traction v_i de chaque roue i, comme le montre l'équation (D-1) :

$$\begin{aligned} v_{x_i} &= v\cos(\theta) - \omega R_i \sin(\alpha_i) \\ v_{y_i} &= v\sin(\theta) + \omega R_i \cos(\alpha_i) \\ v_i &= \sqrt{v_{x_i}^2 + v_{y_i}^2} \\ \theta_i &= \text{atan2}(v_{y_i}, v_{x_i}) \end{aligned} \tag{D-1}$$

Dans l'équation (D-1), (R_i, α_i) est la position angulaire de la roue i, comme on peut le voir sur la figure D-8

Avec cette construction, on peut assurer l'unicité du centre instantané de rotation, et que donc on ne risque pas de se retrouver dans une situation ou les tourelles risquent d'écarteler un AGV, comme on peut le voir sur la figure D-9



(a) De (v, θ, ω) à (v_i, θ_i). (b) Construction géométrique.

Figure D-8 : (v_i, θ_i) en fonction de (v, θ, ω) et (R_i, α_i).

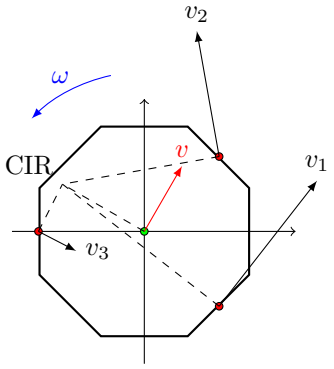


Figure D-9 : Centre Instantané de Rotation (CIR).

D.4.2 Génération de mouvement

D'après les spécifications établies avec l'artiste, la génération du mouvement doit provenir du métabolisme de l'arbre. Et comme nous venons de le voir, nous avons besoin de trois variables indépendantes d'entrée. Nous choisissons donc d'utiliser les signaux de trois sondes Granier implantées à différents endroits de l'arbre, que nous notons (s_1, s_2, s_3) et normalisons dans $[0, 1]$.

Deux de ces sondes peuvent être directement connectées aux vitesses linéaires et angulaires du centre de l'AGV, comme on peut le voir dans l'équation (D-2) :

$$\begin{aligned} v &= s_1 \\ \omega &= 2s_2 - 1 \end{aligned} \tag{D-2}$$

Les unités de ces vitesses sont ensuite déduites pour satisfaire les spécifications sur la vitesse générale de l'arbre. Dans ce cas, la vitesse maximale du tronc de l'arbre est d'un mètre par minute (soit environ 17 millimètres par seconde), mais les roues de l'AGV peuvent aller jusqu'à deux mètres par minute. Par conséquent, nous n'avons qu'à multiplier chaque v_i par 17mm/s avant de les envoyer à l'AGV.

Une stratégie globale de génération de mouvement a été établie avec l'artiste, et implique que l'arbre se « souvienne » à quel endroit l'une des variables de son métabolisme était à son maximum ou son minimum. Un arbre ira donc alternativement chercher le soleil et l'ombre, en fonction de ses sondes Granier, dans un lieu où il se souvient de l'état de son flux de sève, ou dans un lieu où il n'est pas encore allé.

La principale composante de la génération de trajectoire est donc le paramètre θ. La géolocalisation donne la position et l'orientation absolue (x, y, α) de l'AGV, donc une fois que la destination (x_{goal}, y_{goal}) est déterminée grâce à la dernière sonde Granier, on peut obtenir ce paramètre θ suivant l'équation (D-3).

$$\theta = \left(\text{atan2}\left(y - y_{goal}(s_3), x - x_{goal}(s_3)\right) - \alpha\right) \%2\pi - \pi \tag{D-3}$$

Ce principe règle le problème de couverture de l'espace et est développé dans la section D.4.4.

D.4.3 Lissage

Au-dessus de ce principe basique de génération de mouvement, nous implémentons un composant de lissage qui contrôle la vitesse angulaire maximale de chaque roue. En effet, les deux AGVs qui évoluent sur l'esplanade peuvent facilement s'embourber, et notamment en tournant leurs roues trop brutalement, creusant ainsi le sol sous leurs roues.

De son côté, l'AGV à l'intérieur du pavillon français peut produire un bruit strident fort désagréable lorsque son pneu crisse sur le béton, toujours s'il tourne trop vite.

Si ce composant détecte que la différence entre la direction d'un AGV et la direction du $goal$ est supérieure à $2\pi/3$, il peut également inverser le sens de rotation des roues pour n'avoir à tourner que de moins de $\pi/3$. Ceci améliore notamment le comportement aux points de rebroussement, lorsque l'AGV change de $goal$.

Un composant similaire est implémenté sur l'AGV au niveau de chaque tourelle, tout en s'assurant que le centre instantané de rotation reste unique.

D.4.4 Sélection du but

La génération de mouvement définit comment un AGV évoluerait seul sur un plan infini. Nous avons ajouté certaines règles pour nous assurer que la sélection du $goal$ ne fera pas aller un AGV dans un mur ou un autre AGV. Ces règles sont détaillées dans l'algorithme D-3, mais l'idée générale est de mettre à jour en continu deux cartes de l'aire d'évolution avec le $timestamp$ et s_3 dans la case correspondant aux coordonnées actuelles.

De cette manière, nous obtenons des cartes $map_{sapflow}$ et $map_{timestamp}$ indiquant les zones où la sève a coulée rapidement ou non (qui coïncideront probablement avec les zones d'ombre et de soleil), et les zones où un AGV n'est pas allé depuis longtemps.

Ensuite, une machine d'états finis alterne le $goal$ entre des endroits où s_3 était maximum, puis minimum, puis un endroit où l'arbre n'est pas allé depuis longtemps.

Dans certaines circonstances, il est possible qu'aucune de ces zones ne soit atteignable en ligne droite, sans traverser de bordure, ou de trajectoire d'un autre AGV. Dans un tel cas de deadlock, l'AGV doit simplement attendre dans sa position courante.

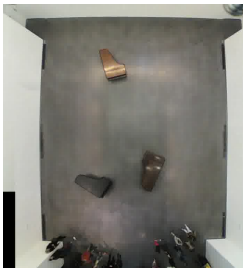
Les traces générées par cet algorithme en simulation sont données dans la figure D-10.

Un des objectifs de la génération de trajectoire est d'assurer une couverture raisonnable des zones d'évolution intérieure et extérieure. De multiples essais en simulation (section D.5.2) nous ont permis de régler précisément les contrôles à travers la normalisation des paramètres (s_1, s_2, s_3).

Dans la zone extérieure, la synchronisation des arbres est effectuée de manière centralisée, à partir de leur localisation respective. Du point de vue formel, ce système ne garantit pas l'absence de minima locaux. Pour autant, la complétion de cet algorithme n'est pas un problème du point de vue pratique.

En effet, deux personnes de l'équipe du pavillon sont tout le temps présentes sur le site pour éviter tout problème. Ces personnes ont la possibilité de déverrouiller un deadlock en donnant artificiellement aux arbres un nouveau $goal$.

En pratique, les interventions de l'équipe arrivent rarement (moins d'une fois par semaine, pendant la période nominale). Par ailleurs, un tel deadlock est impossible pour l'arbre dans le pavillon, puisqu'il évolue seul et dans une zone convexe.



Algorithme D-3 Génération de mouvement

```
loop
    start_loop :                                                    ▷ Label du début
    map_sapflow[position] ← s_3                                     ▷ Mise à jour de cartes
    map_timestamp[position] ← timestamp
    if ‖goal − position‖ < 1 then                                   ▷ Besoin d'un nouveau goal
        for try ← 1, N do
            if state = 1 then
                goal ← arg min(map_sapflow)
            else if state = 2 then
                goal ← arg max(map_sapflow)
            else
                goal ← arg min(map_timestamp)
            end if
            state ← (state + 1)%3
            if (goal − position) ∩ (borders ∪ trajectory_otherAGV) = ∅ then
                goto continue                                       ▷ Pas d'obstacle entre position et goal
            end if
        end for
        goal ← position                                             ▷ Échec de recherche d'un nouveau goal
        v ← 0
        ω ← s_2                                                     ▷ L'arbre peut continuer de tourner sur place
        sleep 10s
        goto start_loop                                             ▷ Nouvelle tentative
    end if
    continue :                                                      ▷ Label de succès
    v ← s_1
    ω ← s_2                                                         ▷ Voir D.4.2
    θ_goal ← (atan2 (y − y_goal(s_3), x − x_goal(s_3)) − α) %2π − π
    sleep 1s
end loop
```

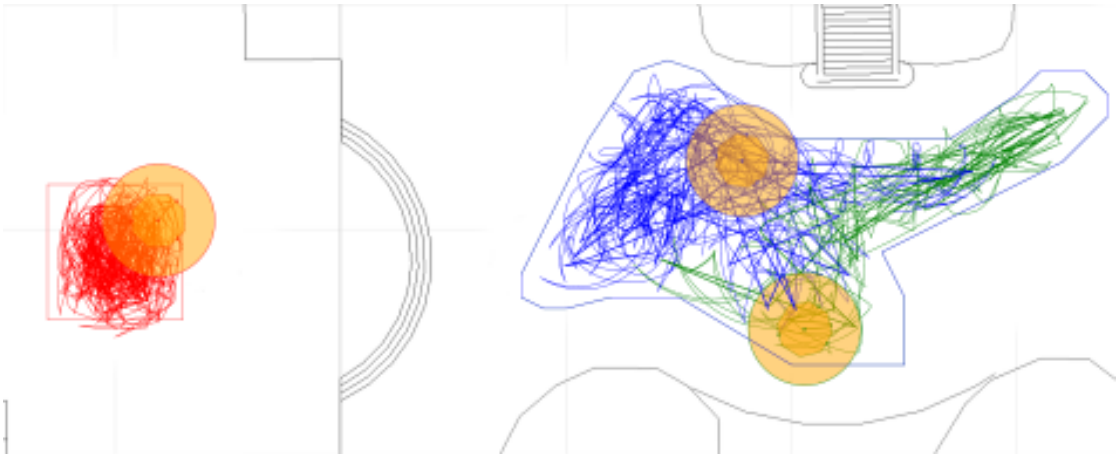


Figure D-10 : Exemple de couverture d'espace en simulation. Sur site, un tel test aurait demandé plusieurs jours entre la fin de l'installation matérielle et l'ouverture de la Biennale.

D.5 Résultats

Dans cette section, nous exposons les résultats opérationnels de ce projet, qui s'est terminé le 22 novembre 2015.

D.5.1 Interface utilisateur

L'interface utilisateur est composée d'une carte du pavillon et de l'esplanade des Giardini. Les AGVs sont présentés dans leur position et orientation actuelle, d'après le système de géolocalisation vu dans la section D.2.2.3, comme on peut le voir dans la figure D-11.

Un panneau de contrôle est également disponible sous cette carte, donnant divers indicateurs numériques ainsi qu'une série de contrôles simplifiant les opérations de maintenance que l'équipe du pavillon pourrait avoir à faire.

Ceci évite la plupart du temps le besoin de brancher un joystick directement sur l'AGV pendant l'exploitation, notamment lors de la présence d'un obstacle inattendu, ou de problèmes dans la stabilité du sol à certains endroits.

Cette interface utilisateur est également disponible à distance, ce qui nous a permis de pouvoir comprendre diverses situations depuis Toulouse et intervenir sans avoir à faire le voyage jusqu'à Venise [1].

D.5.2 Simulateur

La courte période de test sur place combinée à l'extrême lenteur attendue des AGV nous a imposé de régler le système à l'aide d'un simulateur.

1. Bien que, naturellement, cela ne nous a jamais posé de problèmes personnels de faire le déplacement.

Figure D-11 : Capture d'écran de l'interface utilisateur, dans sa version avancée, en production le 9 septembre 2015. On peut y voir les AGVs ainsi que leurs traces de 10 :30 (heure à laquelle l'interface a été lancée) à 10 :45. Il y a également une table indiquant le statut de chaque AGV ainsi que certains contrôles.

Cela a permis de rapidement vérifier la manière dont réagirait le système de génération de trajectoire dans différentes situations, lorsque cela aurait pris des semaines à vérifier en temps réel.

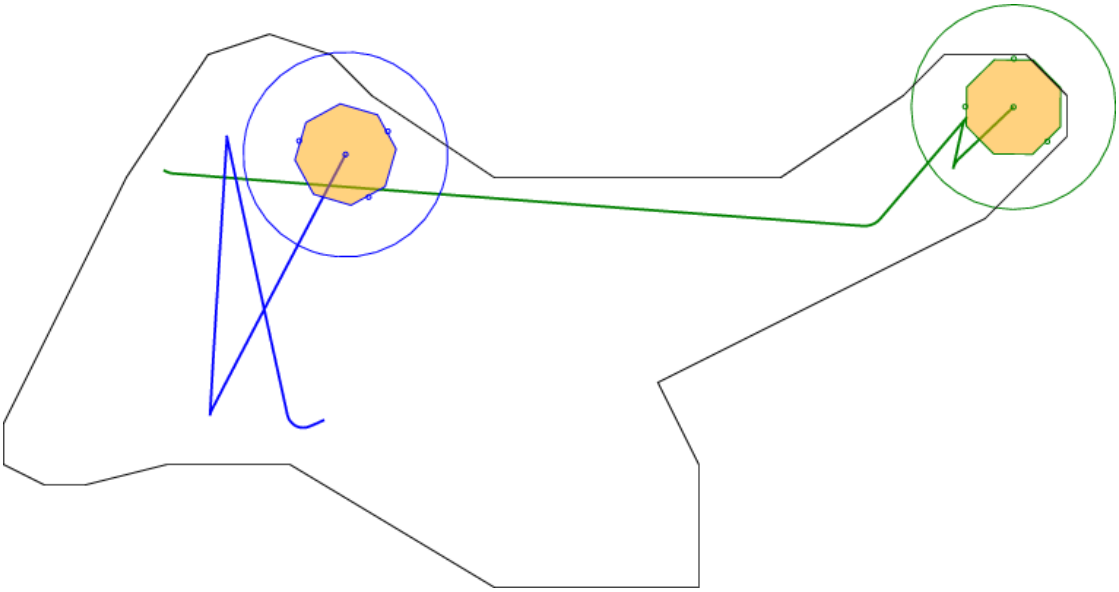
Les blocks logiciels de ce simulateur sont les mêmes que ceux décrits dans la figure D-6, à l'exception des blocks AGV et géolocalisation qui sont remplacés par un seul module de simulation. Les données des sondes Granier utilisées en simulation provenaient de plusieurs jours d'enregistrements sur l'arbre de test au LAAS-CNRS.

Sur la figure D-12, nous montrons un exemple des trajectoires qui peuvent être exécutées par les arbres, ainsi que les marques qui seraient laissées au sol par les roues pour de telles trajectoires.

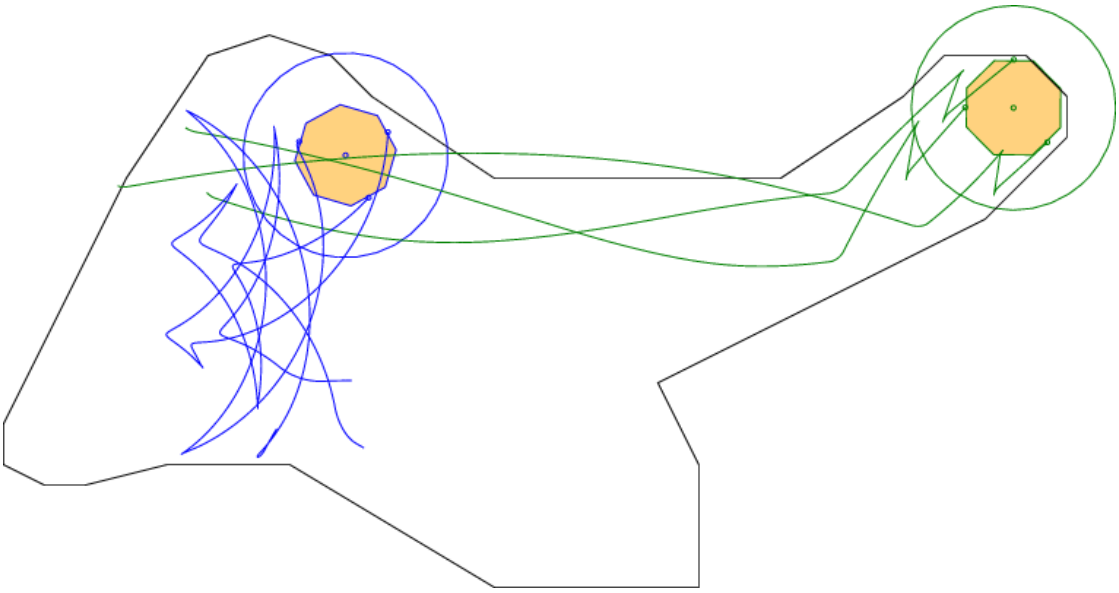
D.5.3 Résultats expérimentaux

Pendant les premiers jours de l'installation matérielle, nous avons remarqué que les roues des AGVs peuvent être très rapidement réorientées. Cette rapidité de réponse est bénéfique aux premiers tests de fonctionnement, mais pose tout de même certains problèmes par la suite :

— Une réorientation rapide rend les moteurs audibles ;
— Un ample et trop rapide mouvement angulaire du pneu par rapport à la traction laisse une marque permanente sur le sol en béton à l'intérieur du pavillon français ;
— Un tel mouvement produit également un désagréable bruit de crissement ;



(a) Trajectoire du centre des AGVs.



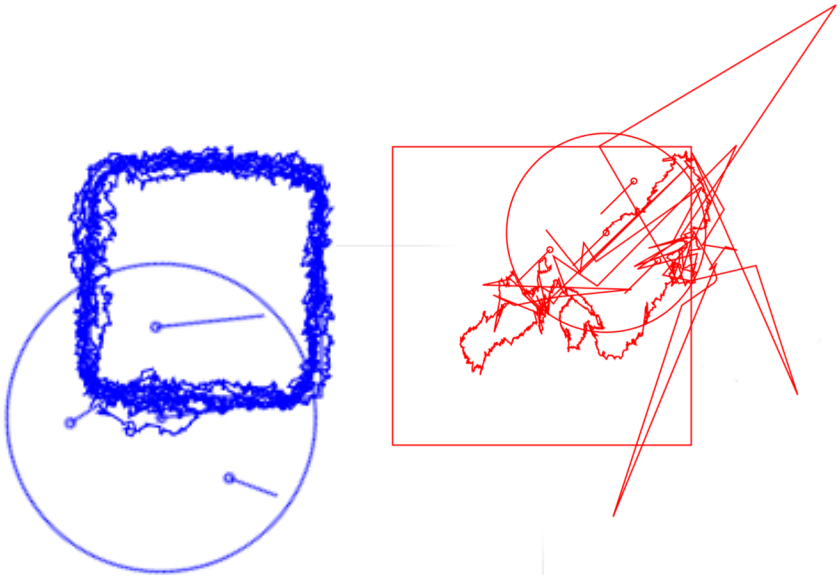
(b) Trajectoire des roues des AGVs.

Figure D-12 : Exemple de simulation avec en haut les trajectoires des centres des AGV et en bas les trajectoires des roues des AGVs. En temps réel, cela correspondrait à peu près à un voyage de 45 minutes. On remarque que les roues peuvent parfois sortir des bordures, mais pas le tronc.

— En fonction de l'humidité du sol de l'esplanade, un changement brusque dans l'orientation d'une tourelle risque de creuser une petite tranchée. Ensuite, si une tranchée est trop profonde ou si les trois roues se retrouvent dans des tranchées, l'AGV court un fort risque d'embourbement.

Dans ces circonstances, notre réaction a été d'implémenter des composants de lissage des trajectoires, vus en section D.4.3.

Par ailleurs, la configuration logicielle des paramètres de géolocalisation n'était pas parfaite au premier essai, comme on peut le voir sur la figure D-13. Dans la majorité des situations, le système de géolocalisation fonctionnait bien mais dans certains cas nous avions des perturbations entraînant des comportements inadmissibles pour les AGVs.



(a) Réussite (b) Échec

Figure D-13 : Essais de suivi de figures simples par un AGV, pendant la phase d'installation matérielle de la biennale. La figure de gauche correspond à une réussite totale sur plusieurs tours, tandis que sur la figure de droite, on comprend que le système de géolocalisation est perdu.

Ces problèmes ont été résolus en réglant au mieux les paramètres des algorithmes d'Ubisense, ce qui n'aurait pas été possible sans l'aide d'ingénieurs de cette entreprise.

Vu la vitesse des robots (un mètre par minute au maximum), la sécurité n'a jamais été un problème majeur. L'équipe du pavillon avait accès à l'interface web grâce à une tablette qu'ils pouvaient garder avec eux, et ils avaient également une télécommande d'arrêt d'urgence qui coupait directement la puissance en cas de risque majeur.

Au fur et à mesure de l'exposition, le mode nominal de fonctionnement de cette œuvre a été de plus en plus utilisé, de quelques heures par jour lors de l'ouverture, jusqu'à des semaines entières au cours de l'été, lorsque les problèmes liés à la configuration du système de géolocalisation ont été définitivement réglés.

La pluie restait le principal problème de l'exploitation de cette installation. Venise est connue pour ses fortes pluies, qui pouvaient régulièrement rendre la zone extérieure totalement impraticable pour les robots. Heureusement, cela n'empêchait pas le public de voir l'œuvre fonctionner dans le pavillon, même si la verrière de la salle principale avait été retirée pour laisser l'arbre respirer. Les salles périphériques pouvaient alors offrir un refuge aux visiteurs.

Les spécifications sur le bruit ont été respectées, si bien que le public ne se rendait pas forcément compte que les arbres bougeaient. Et lorsqu'un visiteur le remarquait, il se demandait comment des arbres pouvaient bouger.

D.6 Suite du projet

Cette œuvre doit à nouveau être exposée au MONA de Tasmanie, à partir de l'automne 2017. Dans cette section, nous exposons les différences sur les spécifications, et les modifications apportées au système.

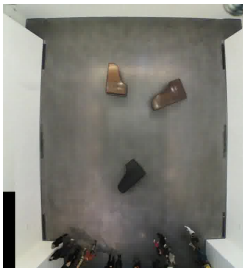
Pour la Biennale de Venise, il était envisageable de faire le trajet en urgence depuis Toulouse en cas de problème majeur, même si dans la plupart des cas, un accès distant à l'environnement logiciel suffisait. Pour la Tasmanie, cela n'est plus possible.

Nous avons donc formé une équipe de techniciens qualifiés à l'entretien mécanique des AGVs, ainsi qu'au déploiement logiciel. Les logiciels utilisés ont alors été traduits, mis à jour, et adaptées à certaines nouvelles contraintes :

— Un seul AGV sera présent, ce qui rend inutile l'évitement des collisions entre deux arbres ;
— Une partie de la zone d'évolution de l'AGV sera un terrain de tennis, sur lequel seules les translations sont autorisées pour ne pas endommager le court ;
— Divers obstacles permanents sont au centre de la zone d'évolution, il faut donc s'assurer de leur évitement.

Deuxième partie

Étude de la robotique humanoïde



Introduction : La locomotion bipède

Après avoir exploré la robotique mobile avec les robots à roues, nous passerons dans cette partie à un autre mode de déplacement en examinant les bipèdes.

L'étude de la marche bipède a trois intérêts majeurs :

— Mieux comprendre la locomotion humaine (Arechavaleta et al. 2008),
— Créer des prothèses pour aider des personnes à se déplacer (Mombaur et Hoang 2017).
— Créer des robots capables de se déplacer et d'agir dans un environnement façonné par et pour l'homme,

Ces dernières années, la conception et la fabrication de robots bipèdes a largement évolué. On trouve désormais régulièrement des vidéos, réalisées à destination du grand public, où l'on pourrait croire que des robots bipèdes sont prêts à être utilisés de façon commerciale. C'est notamment le cas avec les vidéos de sociétés comme Boston Dynamics (figure II-1a) ou Agility Robotics (figure II-1b), où l'on voit des robots bipèdes, de taille humaine, qui semblent suffisamment à l'aise pour se déplacer seuls sur des types de sols allant du béton plat à une pente en forêt recouverte de neige, ou même pour sauter des escaliers.

(a) Atlas, Boston Dynamics (b) Cassie, Agility Robotics

Figure II-1 : Captures d'écrans de vidéos de Boston Dynamics et Agility Robotics. Ces robots bipèdes semblent être livrés à eux-mêmes en pleine nature.

Cependant, comme l'a remarqué Moravec (1988), le mouvement est un problème complexe. Plus récement, le DARPA Robotics Challenge nous a montré ce que l'état de l'art permettait de réellement accomplir dans un contexte de tests réalistes (Atkeson et al. 2016). Sans surprise, même les meilleures équipes du monde sont encore loin de pouvoir créer un robot accomplissant une série de tâches pourtant triviales pour un enfant.

La locomotion bipède n'est pas un problème simple. L'être humain a besoin d'une à trois années pour la contrôler, alors que d'autres moyens de locomotion sont maîtrisés dès la naissance chez les animaux.

Pour étudier la marche, de nombreuses machines ont été créées.

(a) Ruina, 2001 (b) Ikemata, 2006

Figure II-2 : Robots bipèdes passifs.

Certaines sont purement mécaniques (figure II-2), et étaient initialement des jouets bipèdes qui descendaient passivement une pente. Elles ont commencé à être étudiées par McGeer (1990) dans les années 1990, puis ont conduit à des travaux de complexité croissante, notamment à Delft (Wisse et Van der Linde 2007a).

À l'opposé, d'autres machines ont dès le début été dotées d'un nombre important de capteurs et d'actionneurs puissants, et constituent donc de véritables robots (figure II-3). Celles-ci ont été d'abord étudiées et développées au Japon (Sakagami et al. 2002 ; Kaneko et al. 2002) dans les années 2000.

(a) Asimo, Honda, 2000 (b) Valkyrie, NASA, 2013 (c) Talos, PAL Robotics, 2017

Figure II-3 : Robots bipèdes actifs.

Dans un premier temps, ces machines actives ont utilisé des mouvements quasi-statiques. Autrement dit, à tout instant, la projection de leur centre de masse sur le sol restait dans le polygone support. Or, si une locomotion constituée d'une série de poses à l'équilibre statique est simple et a donc de bonnes chances de fonctionner, elle présente certains inconvénients. Parmi ces inconvénients, on citera notamment une faible vitesse, une grande consommation énergétique, et une démarche peu pertinente. Ainsi, pour Grizzle et al. (2010), en locomotion, le régime permanent correspond à des solutions périodiques et non des équilibres statiques.

En utilisant des moteurs plus puissants et des contrôleurs plus complexes, on est aujourd'hui en mesure de générer des mouvements de locomotion dynamique bien plus convaincants, mais qui restent loin de ce que l'on retrouve chez les êtres vivants.

Dans le chapitre E, nous verrons une méthode ayant pour objectif de concevoir de manière optimale des robots tirant parti de leur inertie comme un marcheur passif, tout en étant actionné par des moteurs, et donc contrôlables.

Chapitre E

Robots bipèdes

Dans ce chapitre, nous présentons une méthode de codesign de marcheurs bipèdes, adaptée à l'étude et la fabrication de robots dont la répartition des masses et le contrôle sont simultanément optimisés.

Nous générons donc un mouvement de locomotion par l'optimisation simultanée d'une conception mécanique et d'un contrôleur associé.

E.1 Introduction

Les marcheurs passifs sont des robots bipèdes actionnés principalement par la gravité. Ils utilisent leur dynamique naturelle pour avancer, mais ne peuvent pas rester dans un équilibre quasi-statique stable.

De tels systèmes mécaniques ont un excellent coût de transport (CoT). Ils permettent à la fois au bio-mécaniciens de mieux comprendre le fonctionnement de la marche bipède, et aux créateurs de robots de fabriquer des robots humanoïdes plus efficaces.

Dans ce chapitre, nous proposons une méthode dynamique générique pour optimiser à la fois le design mécanique d'un robot et de son contrôleur en fonction d'un coût donné. Elle est décrite dans la figure E-1.

Cette méthode cherche à tirer parti de la dynamique d'un marcheur passif, tout en permettant de créer un robot pouvant, au choix, être actionné activement ou passivement.

Cette méthode consiste à donner principalement une structure cinématique et une fonction de coût à un solveur numérique. Celui-ci doit alors optimiser les paramètres mécaniques du robot et de son contrôleur. Pour cela, il utilise une librairie de calculs dynamiques pour simuler la réponse du système en fonction des paramètres et contrôles qu'il lui donne.

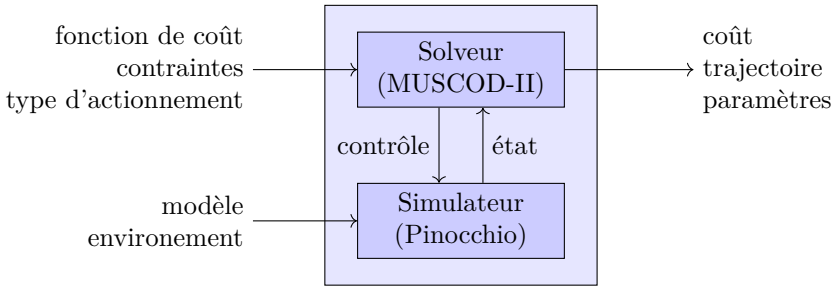


Figure E-1 : Vue d'ensemble de l'implémentation de notre méthode de simulation et d'optimisation. Le simulateur est décrit dans la section E.2 et le solveur numérique dans la section E.3.

E.1.1 Travaux apparentés

Depuis McGeer (1990), de nombreux marcheurs passifs ont été pensés et fabriqués. Une bonne introduction à ce domaine est donnée par Collins et al. (2005).

Une méthodologie complète de fabrication de marcheurs passifs de complexités croissantes et décrite par Wisse et Van der Linde (2007a). Elle commence avec un simple modèle de compas (c'est à dire seulement un segment rigide pour chaque jambe, reliés par une liaison pivot au niveau du bassin) et va jusqu'à la réalisation de Denise, un marcheur dynamique 3D, avec un système d'équilibre du torse utilisant la bissectrice de l'angle des jambes, deux bras mécaniquement couplés aux angles des hanches, des genoux que se déverrouillent pendant la phase de balancier, et des chevilles conçues pour diriger le robot dans la direction qui empêche la chute.

Cette approche incrémentale révèle les principaux problèmes de la création de marcheurs anthropomorphes. Le premier de ces problèmes est la transition du compas évoluant uniquement dans le plan sagittal (c'est à dire le plan de symétrie pour un humanoïde) à un système 3D (Collins, Wisse, et Ruina 2001 ; Tedrake et al. 2004 ; Hobbelen, De Boer, et Wisse 2008).

Le second problème est l'extension du modèle du compas à l'utilisation de jambes articulées (McGeer 1990 ; Collins, Wisse, et Ruina 2001 ; Chevallereau et al. 2003 ; Ikemata, Sano, et Fujimoto 2006 ; Hobbelen, De Boer, et Wisse 2008).

Ensuite, l'ajout des chevilles est abordé dans (Wisse et al. 2006 ; Hobbelen, De Boer, et Wisse 2008), celui des bras dans (Tedrake et al. 2004 ; Hobbelen, De Boer, et Wisse 2008), et celui du cou et de la tête dans (Benallegue, Laumond, et Berthoz 2013 ; Falotico et al. 2016).

La distribution des masses pour une structure cinématique donnée est également un problème important, et il est adressé dans (Hass, Herrmann, et Geisel 2006 ; Remy, Buffinton, et Siegwart 2011).

Outre ces contributions mécatroniques, la recherche s'intéresse également à la minimisation de la consommation énergétique (Collins et al. 2005 ; Bhounsule et al. 2014) et à la stabilité de marcheurs passifs (Mombaur 2001 ; Grizzle, Abba, et Plestan 2001 ; Ikemata, Sano, et Fujimoto 2006 ; Byl et Tedrake 2009 ; Benallegue, Laumond, et Berthoz 2013).

Par opposition aux approches de contrôle prédictif non-passif (Kajita et al. 2003 ; Pratt et al. 2006) qui exigent une planification des pas à l'avance, le contrôle de marcheurs passifs cherche à maintenir au mieux la dynamique périodique naturelle provenant du design mécanique.

Le design mécanique et le contrôle sont profondément liés, et le défi est de les gérer tous les deux simultanément, en considérant que le cycle limite provenant du design mécanique est à l'origine d'une démarche stable.

Le rôle du contrôleur est alors de maintenir au mieux le système proche de son cycle limite intrinsèque en dépit des perturbations.

Dans le cas de l'optimisation fondée sur un modèle, la recherche d'un bon contrôleur est exprimé comme un problème d'optimisation numérique contraint bénéficiant d'une solide analyse théorique de la stabilité (Diehl, Mombaur, et Noll 2009). Cette méthode numérique générale a été appliquée à la distribution des masses de marcheurs passifs (Hass, Herrmann, et Geisel 2006), puis à l'étude de quadrupèdes passifs (Remy, Buffinton, et Siegwart 2010), et enfin aux marcheurs passifs dans le cadre de la création et de l'analyse de démarches efficaces (Remy, Buffinton, et Siegwart 2011).

Dans ce dernier article, les auteurs proposent un système de simulation dynamique dont la portée est illustrée dans divers exemples dont des marcheurs passifs et des robots qui courent. Notre travail utilise la même idée générale.

E.1.2 Définition du problème

Comme nous l'avons vu, notre but est de résoudre simultanément les deux problèmes suivants :

1. Pour une chaîne cinématique donnée, nous cherchons à optimiser les paramètres d'un robot tels que la distribution de masse, les longueurs des différents segments, la taille et la durée d'un pas, etc. ;
2. Pour un robot donné, nous voulons le contrôleur minimisant au mieux une fonction de coût donnée, telle que le coût de transport, ou le temps minimal.

En bref, pour une structure cinématique donnée, nous proposons une méthode générique pour trouver tous les paramètres d'un robot ainsi que le contrôleur qui optimisent une fonction de coût donnée.

E.1.3 Contributions

La première nouveauté de cette approche est sa capacité à gérer une architecture complexe telle qu'un marcheur humanoïde, tant en 2D qu'en 3D. Il n'est pas nécessaire que le mouvement se fasse uniquement dans le plan sagittal.

De plus, et contrairement aux travaux similaires sur le sujet, notre système calcule automatiquement la dynamique complète d'un robot donné. Il n'est donc plus nécessaire de décrire entièrement un système polyarticulé à travers ses équations dynamiques. De ce fait de nombreux marcheurs passifs peuvent être efficacement créés, optimisés et comparés.

Deuxièmement, le contrôle peut être actif ou passif. Tout type d'actionneur peut être modélisé et géré par cette méthode. Nous pouvons par exemple appliquer un couple idéal suivant une spline, déterminer les coefficients d'un ressort ou d'un amortisseur, ou encore utiliser des moteurs en série ou en parrallèle avec des ressorts.

Enfin, nous pouvons optimiser différents paramètres d'un marcheur donné (pente, longueurs, masses, vitesses, etc.) par rapport à une fonction de coût donnée.

E.1.4 Plan

Dans la section E.2, nous commençons par introduire le simulateur de contacts dynamiques au cœur du système. Nous montrons comment les différents paramètres du problème sont répartis entre le modèle mécanique et le contrôleur. Puis, dans la section E.3, nous établissons la formulation générique de contrôle optimal qui permet l'optimisation de ces paramètres.

Ensuite, nous présentons un premier protocole expérimental visant à tester et illustrer notre méthode dans la section E.4, dont les résultats sont donnés dans la section E.5. Puis, dans la section E.6, nous appliquons notre méthode à l'étude de différents actioneurs rigides ou munis de ressorts.

Enfin, nous exposons les travaux en cours sur l'étude de la stabilisation de la tête (section E.7) et la conception d'un prototype associé à cette étude (section E.8), puis les perspectives futures (section E.9).

E.2 Simulation Dynamique

Les marcheurs bipèdes sont des systèmes intrinsèquement hybrides (Grizzle et al. 2010). Ils sont soumis à une dynamique continue lorsque la jambe d'appui est en contact avec le sol, puis ils sont ensuite soumis à un impact lorsque l'autre jambe heurte le sol.

Outre cette dynamique hybride, certaines contraintes de contact doivent être vérifiées pour assurer la faisabilité du mouvement.

Dans cette section, nous décrivons les notations, le modèle paramétrique des marcheurs, et la formulation des contacts utilisé dans notre système.

E.2.1 Notations

Nous assimilons le marcheur bipède à une chaîne polyarticulée dont la base flotte librement. On note son vecteur de configurations par $\boldsymbol{q} \in SE(3) \times \mathbb{R}^n$, où $SE(3)$ est le groupe spécial Euclidien de dimension 3 exprimant la position de la base du robot, et n le nombre de degrés de liberté (DoF). Les vitesses et accélérations de ce vecteur de configurations sont notées respectivement $\dot{\boldsymbol{q}}$ et $\ddot{\boldsymbol{q}}$, et évoluent dans $\mathbb{R}^{6+n}$. Enfin, le couple appliqué à chaque articulation est noté $\boldsymbol{\tau} \in \mathbb{R}^n$.

E.2.2 Modèle

Un marcheur bipède est principalement un arbre cinématique, c'est-à-dire un arbre d'articulations où chaque articulation a sa propre topologie (pivot, glissière, etc.) et un placement par rapport à l'articulation parente. Les articulations sont les nœuds de l'arbre.

De plus, chaque articulation porte un corps, qui est défini par sa masse, la position de son centre de masse, et sa matrice d'inertie. L'ensemble de ces corps définit la distribution des masses du modèle.

Cette structure en arbre et la distribution des masses correspondent aux paramètres structurels du système. Le modèle du marcheur bipède est donc paramétré par ces deux ensembles de paramètres :

$$\text{modèle}(\text{arbre}, \text{distribution\_masses})$$

E.2.3 Contrôleur

Une démarche est caractérisée par son contrôleur qui est représenté par un ensemble de paramètres réels. Par exemple, un contrôleur peut être un ensemble de splines qui encodent les trajectoires du couple, ou simplement les gains d'un PID dans le cas d'un contrôleur purement passif.

$$\text{contrôleur}(\text{paramètres\_contrôle})$$

E.2.4 Contacts

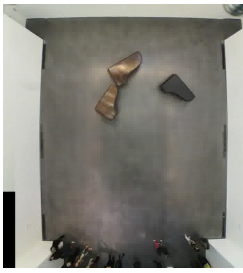
En ce qui concerne la dynamique continue, nous faisons l'hypothèse d'un contact ponctuel rigide avec des cônes de frottement suivant les lois de Coulomb.

L'équation dynamique du système polyarticulé sous contraintes peut être définie comme dans l'équation (E-1).

$$\begin{aligned} M(\boldsymbol{q})\ddot{\boldsymbol{q}} + b(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= S^\top \boldsymbol{\tau} + J_c(\boldsymbol{q})^\top \boldsymbol{f}_c \\ J_c(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{J}_c(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} &= 0 \end{aligned} \tag{E-1}$$

Dans l'équation (E-1), $M(\boldsymbol{q})$ est la matrice d'inertie exprimée dans l'espace des articulations, $b(\boldsymbol{q}, \dot{\boldsymbol{q}})$ correspond aux effets de Coriolis, centrifuge et gravitationnel, S est une matrice de sélection encodant la sous-actuation, $J_c(\boldsymbol{q})$ est la matrice Jacobienne des contacts avec $\boldsymbol{f}_c$ les forces aux contacts, et $.^\top$ est l'opérateur de transposition.

Une condition nécessaire et suffisante de contact sans glissement est que $\boldsymbol{f}_c$ reste à l'intérieur du cône de frottements $\mathcal{K}_c$. Cette condition implique que la composante normale de cette force de contact reste positive (le sol ne peut pas tirer), et que la norme de sa composante tangentielle et la norme du couple normal sont limités par la composante normale.



Nous résolvons les deux équations de équation (E-1) ensemble, et ajoutons la contrainte du cône de frottements directement dans le problème de contrôle optimal pour assurer le modèle de contact de Coulomb (Brogliato 1999).

D'autres travaux ont essayé d'éliminer cet hypothèse supplémentaire (Stewart et Trinkle 2000), ce qui a amené des problèmes d'optimisation de trajectoire (Tassa, Erez, et Todorov 2012 ; Posa, Cantu, et Tedrake 2014).

Dans le contexte de nos travaux, sélectionner à l'avance les phases de contacts n'est pas une limitation.

L'accélération des articulations et les forces aux contacts sont donc données dans l'équation (E-2)

$$\begin{aligned} \ddot{\boldsymbol{q}} &= M^{-1}(I_n - J_c^t \Lambda_c^{-1} J_c M^{-1})(S^\top \boldsymbol{\tau} - \boldsymbol{b}) - M^{-1} J_c^\top \Lambda_c^{-1} \dot{J}_c \dot{\boldsymbol{q}} \\ \boldsymbol{f}_c &= \Lambda_c^{-1} \left(J_c M^{-1} (\boldsymbol{b} - S^t \boldsymbol{\tau}) - \dot{J}_c \dot{\boldsymbol{q}} \right) \end{aligned} \tag{E-2}$$

Dans l'équation (E-2), $\Lambda_c \triangleq J_c M^{-1} J_c^\top$ est la matrice d'inertie et I_n la matrice identité de dimension n. Les dépendances à $\boldsymbol{q}$ et $\dot{\boldsymbol{q}}$ ont été omises pour simplifier les notations.

E.2.5 Impacts

Les marcheurs bipèdes sont également soumis à des impacts. Dans ce cas, nous faisons l'hypothèse d'un impact instantané et inélastique, avec un coefficient de restitution de zéro. En d'autres termes, la vitesse du point de contact après l'impact est nulle.

La dynamique de l'impact nous conduit alors à une discontinuité dans l'espace des vitesses des articulations, ce qui est décrit dans l'équation (E-3).

$$\begin{aligned} \dot{\boldsymbol{q}}^+ &= (I_n - M^{-1} J_c^\top \Lambda_c^{-1} J_c) \dot{\boldsymbol{q}}^- \\ \boldsymbol{\lambda}_c &= \Lambda_c^{-1} J_c \dot{\boldsymbol{q}}^- \end{aligned} \tag{E-3}$$

Dans l'équation (E-3), $\dot{\boldsymbol{q}}^+$ et $\dot{\boldsymbol{q}}^-$ sont les vitesses généralisées pré-impact et post-impact, et $\boldsymbol{\lambda}_c$ est l'impulsion résultante de l'impact (Brogliato 1999).

D'autres modèles d'impact (e.g. élastique) pourraient également être introduits.

Ce modèle est fréquemment utilisé dans la littérature (Schultz et Mombaur 2010), même si sa consistance physique est contestée (Chatterjee et Ruina 1998).

E.2.6 Calcul Dynamique

Le solveur de contrôle optimal doit calculer la dynamique du corps complet des milliers de fois lors de la procédure d'intégration numérique. Dans ce but, nous avons utilisé Pinocchio (Carpentier et al., s. d.), une librairie C++ efficace qui sert à modéliser et calculer les dynamiques directe et inverse d'un système polyarticulé en contact.

Pinocchio utilise la librairie C++ d'algèbre linéaire Eigen (Guennebaud, Jacob, et al. 2010).

Pinocchio est basé sur les algorithmes de Featherstone (2008), mais ils ont été implémentés de manière à bénéficier de la prédiction de branche et des mécanismes de cache des processeurs modernes (Naveau et al. 2014).

E.3 Contrôle optimal pour le design et le contrôle

Le contrôle optimal est un puissant outil mathématique générique qui permet de trouver une solution particulière parmi une infinité de possibilités. Ces dernières années, des cadres fiables et efficaces de contrôle optimal sont apparus, permettant de contrôler des systèmes complexes de grandes dimensions (Leineweber et al. 2003; Houska, Ferreau, et Diehl 2011).

Dans notre cas, la recherche simultanée des paramètres du modèle et du contrôle est posée comme un problème de contrôle optimal pour une fonction de coût donnée. Cette fonction de coût représente l'objectif de la démarche et peut être n'importe quelle fonction dont les valeurs sont des réels.

Comme exemples de fonction de coût pour des marcheurs bipèdes, on retrouve le coût de transport ou le temps minimal. De plus, nous ajoutons la possibilité de laisser la durée du mouvement comme étant l'un des paramètres libres du problème.

D'autres paramètres libres, comme la longueur d'un pas ou la pente du sol peuvent être ajoutés dans la liste des paramètres.

E.3.1 Notations

Nous notons $\boldsymbol{x} \triangleq (\boldsymbol{q}, \dot{\boldsymbol{q}})$ l'état du système, $\boldsymbol{u}$ le vecteur de contrôle et $\boldsymbol{p}$ le vecteur des paramètres, composé à la fois des paramètres du modèle et des paramètres libres susmentionnés.

Les trajectoires d'état et de contrôle sont respectivement notées $\underline{\boldsymbol{x}}$ et $\underline{\boldsymbol{u}}$

La fonction de coût et la dynamique du système sont respectivement notées $\ell(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p})$ et $\frac{d\boldsymbol{x}}{dt} = f(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p})$. Nous utilisons ici les mêmes notations pour noter à la fois la dynamique continue et la dynamique d'impact.

Enfin, $g(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p})$ correspond aux contraintes d'inégalités qui doivent être vérifiées tout au long de la trajectoire. Des contraintes d'égalité pourraient également être ajoutées sans nuire à la généralité de notre démarche.

Comme nous l'avons vu dans la section E.2.4, la fonction g est essentiellement composée des contraintes sur le cône de frottement.

E.3.2 Formulation du problème de contrôle optimal

La dynamique hybride des marcheurs bipèdes peut être considérée comme un système multi-phases, comprenant des phases de simple et double support et d'impact. Dans la suite, l'entier s correspond à l'index de la $s^{\text{ème}}$ phase.

Le problème générique de contrôle optimal pour déterminer simultanément les paramètres du modèle et du contrôleur peut donc être écrit comme dans l'équation (E-4)

$$\begin{aligned} \min_{\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\boldsymbol{p}} \quad & \sum_{s=1}^{S} \int_{t_s}^{t_s+\Delta t_s} \ell_s(t,\boldsymbol{x},\boldsymbol{u},\boldsymbol{p})\, dt \\ \text{s.t.} \quad & \dot{\boldsymbol{x}} = f_s(t,\boldsymbol{x},\boldsymbol{u},\boldsymbol{p}), \forall t \in [t_s, t_s+\Delta t_s] \\ & g_s(t,\boldsymbol{x},\boldsymbol{u},\boldsymbol{p}) \geq \mathbf{0}, \forall t \in [t_s, t_s+\Delta t_s] \\ & \pi(\underline{\boldsymbol{x}},\underline{\boldsymbol{u}},\boldsymbol{p}) \geq \mathbf{0} \end{aligned} \tag{E-4}$$

Dans l'équation (E-4), π est une fonction qui agit à a fois sur les trajectoires d'état et de contrôle pour garantir les contraintes de périodicité. Δt_s est la durée de la phase s, et $T = \sum \Delta t_s$ est la durée totale du mouvement.

Dans le cas d'un impact, la durée de la phase est réduite à 0.

E.3.3 Résoudre le problème de contrôle optimal

Deux principales méthodes existent pour résoudre le problème de dimension infinie équation (E-4).

La première méthode et dite méthode indirecte. Elle consiste à exploiter les conditions nécessaires d'optimalité, à savoir le principe du maximum de Pontryagin (Liberzon 2012), qui transforme le problème équation (E-4) en un problème à valeurs bornées d'équations différentielles classiques. Cependant, cette méthode n'est pas capable de gérer les contraintes de trajectoire.

La seconde méthode est dite directe. Cette méthode commence par discrétiser le problème initial en un problème d'optimisation non linéaire (NLP) de dimensions finies, qui est alors résolu avec des stratégies standard pour des NLP. Parmi ces stratégies, on en retrouve trois principales : le single shooting, la collocation et le multiple shooting.

Dans la suite de cette section, nous présentons succinctement ces trois méthodes. Pour plus de détails, on peut se reporter à (Diehl et al. 2006).

E.3.3.1 Single Shooting

Cette méthode discrétise le contrôle et les contraintes suivant une grille temporelle. La trajectoire de l'état est retrouvée par intégration de la trajectoire discrète du contrôle suivant cette grille.

Cette méthode réduit le NLP à la recherche d'une trajectoire de contrôle, donc le problème d'optimisation est de faible dimension. Cependant, le solveur est difficile à initialiser si l'on dispose seulement d'une estimation initiale sur la trajectoire de l'état, et peut ne pas converger du tout dans le contexte de systèmes non stables.

E.3.3.2 Collocation

Cette méthode discrétise à la fois les trajectoires de contrôle et d'état suivant une grille temporelle. Pour cette méthode, la trajectoire de l'état assure la partie dynamique de l'équation (E-4) à chaque nœud de la grille. Le problème peut donc alors facilement être initialisé à partir d'une trajectoire d'état donnée, et la collocation gère bien les dynamiques instables.

Cependant, une grille très fine est nécessaire pour rapprocher la trajectoire d'état de la dynamique réelle du système.

E.3.3.3 Multiple Shooting

Cette méthode profite des deux précédentes. Elle utilise une grille temporelle plus grossière. Sur chaque intervalle, le contrôle et l'état initial sont discrétisés. L'état final de l'intervalle est alors obtenu par l'intégration de la dynamique du système.

De cette manière, chaque intervalle est indépendant de ses voisins. Les dépendances entre les intervalles successifs sont transférées en tant que contraintes d'égalité du NLP.

Le NLP reste à faible dimension et peut facilement être initialisé à partir d'une trajectoire d'état.

De plus, le multiple shooting est particulièrement adapté aux systèmes multi-phases, puisque les phases sont indépendantes les unes des autres.

Enfin, le multiple shooting a déjà été utilisé avec succès pour la modélisation de la course humaine (Schultz et Mombaur 2010).

D'après ces avantages, nous choisissons cette stratégie pour résoudre le problème équation (E-4).

E.3.4 Solveur pour contrôle optimal

Notre méthode repose sur l'utilisation du logiciel MUSCOD-II (Leineweber et al. 2003), un solveur utilisant la méthode du multiple shooting prévu pour les systèmes hautement non-linéaires soumis à des contraintes d'égalité et d'inégalités sur les trajectoires, développé dans le groupe d'Optimisation et de Simulation de l'université d'Heidelberg.

MUSCOD-II gère efficacement les systèmes multi-phases avec des dynamiques discontinues et des contraintes de périodicité.

Cependant, MUSCOD-II n'est pas un logiciel libre. Il reste possible de le remplacer par ACADO (Houska, Ferreau, et Diehl 2011), qui implémente des fonctionnalités similaires.

E.4 Premières expériences de test

Dans cette section, nous décrivons la première expérience de test de notre méthode. Les résultats de cette expérience sont énoncés dans la section suivante, section E.5

La généralité de notre méthode est illustrée dans les divers exemples fournis dans la table |E-1|. Ces exemples incluent différentes structures cinématiques et différents types de contrôle.

De cette manière, nous pouvons comparer différents topologies de chaînes polyarticulées, et, pour une topologie donnée, comparer différentes méthodes de contrôle, utilisant des actionneurs actifs ou bien passifs. Une fois qu'une topologie est choisie, nous montrons qu'il est possible de remplacer un actionnement actif par un simple amortisseur.

Le mouvement des cinq premiers robots donnés dans la table |E-1| est contraint au plan sagittal. Cette restriction est supprimée pour le sixième robot, qui évolue dans l'espace 3D.

E.4.1 Entrées et sorties

La figure E-1 montre les entrées et sorties générales de notre système. Dans les expériences données dans le table |E-1|, les entrées sont :

- la structure cinématique du robot ;
- des paramètres anthropométriques pour les corps attachés aux articulations de cette structure ;
- une fonction de coût définie dans la section E.4.2 ;
- des contraintes définies dans la section E.4.3 ;
- une méthode d'actionnement définie dans la section E.4.4.

Dans ces expériences, nous choisissons de fixer à la fois la durée d'un pas à 0,8 secondes et une pente descendante inclinée de 0,05 radians pour chaque scénario, et laissons le solveur optimiser la longueur d'un pas, tout en la limitant à l'intervalle $[0.4, 1]$ mètres.

Les sorties de notre système sont le coût optimal de transport, ainsi que la longueur du pas et les trajectoires d'état et de contrôle associées.

Nous étudions également le nombre d'itérations nécessaire au solveur pour converger ainsi que la durée de cette résolution.

E.4.2 Fonction de coût

Nous utilisons la même fonction de coût dans chaque scénario. Cette fonction de coût correspond au classique coût de transport (CoT). Le CoT est une quantité sans dimensions que reflète l'efficacité énergétique d'une méthode de locomotion.

Par définition, le CoT est le ratio entre l'énergie E consommée par le système et sa masse mg multipliée par la distance parcourue d, comme le montre l'équation (E-5).

$$\text{CoT} \triangleq \frac{E}{mgd} \tag{E-5}$$

Dans l'équation (E-5), m est la masse du système. L'énergie E qu'il consomme est la somme des énergies potentielle mgh (où h est la variation de l'altitude du centre de masse) et de l'intégrale de la puissance consommée par les actionneurs, donnée dans l'équation (E-6).

$$E_A = \int_{t=0}^{T} \|\dot{\boldsymbol{q}}^+ - \dot{\boldsymbol{q}}^-\|_M \delta + \boldsymbol{\tau} \cdot \dot{\boldsymbol{q}} dt \tag{E-6}$$

Dans cette équation (E-6), δ est l'impulsion de Dirac correspondant à l'impact, $\cdot$ est l'opérateur de produit scalaire, et $\|\boldsymbol{x}\| \triangleq \sqrt{\boldsymbol{x}^\top M \boldsymbol{x}}$.

Sur une pente d'angle α, nous pouvons donc écrire le CoT suivant l'équation (E-7).

$$\text{CoT} = \sin\alpha + \int_{t=0}^{T} \frac{\|\dot{\boldsymbol{q}}^+ - \dot{\boldsymbol{q}}^-\|_M \delta + \boldsymbol{\tau} \cdot \dot{\boldsymbol{q}}}{mgd} dt \tag{E-7}$$

Cette fonction de coût est ici utilisée comme un exemple pour les différents cas vus dans la section E.5 ; dans des cas réels, le choix d'une meilleure fonction de coût reste un problème ouvert.

De plus, les CoT que nous obtenons ne sont pas destinés à être comparés avec d'autres exemples, dans la mesure où nous ne considérons que des systèmes parfaits, sans frottements ni pertes lors de conversion d'énergies.

E.4.3 Contraintes

Pour réduire les dimensions du NLP, nous calculons la solution du problème pour un demi-pas, grâce à la périodicité et la symétrie entre les segments gauches et droits.

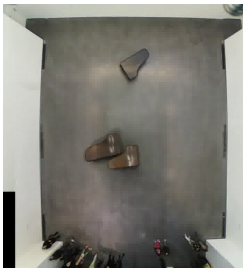
Nous contraignons alors la jambe qui se balance à être en contact avec le sol uniquement au début et à la fin de la simulation. Ensuite, la cyclicité et la symétrie du mouvement est assurée par des contraintes périodiques sur les configurations, vitesses et couples initiaux et finaux.

E.4.4 Actionnement

Dans une première étape, nous utilisons un actionnement actif. La trajectoire du couple appliqué à chaque articulation lors de la marche est alors modélisée par des splines d'ordre trois.

Une implémentation matérielle aurait alors besoin d'une source d'énergie, comme une batterie ou une bonbonne d'air comprimé, ainsi que d'un contrôleur pouvant délivrer le couple désiré.

Dans une seconde étape, nous comparons cet actionnement actif à un actionnement passif, qui consiste simplement en un contrôleur Proportionnel Dérivé (PD) réalisé à partir d'un ressort et d'un amortisseur. Dans ce cas, le couple τ_j appliqué à l'articulation j est donné par l'équation (E-8).



$$\tau_j = -P_j(q_j - Q_j) - D_j\dot{q}_j \tag{E-8}$$

Dans l'équation (E-8), q_j est la configuration de l'articulation j, et $\dot{q}_j$ sa vitesse. P_j est la raideur du ressort associé à l'articulation j, Q_j sa longueur libre, et D_j le coefficient d'amortissement. Ces trois paramètres sont optimisés par le solveur.

À partir de ces paramètres, il est théoriquement possible de fabriquer un marcheur purement passif. Dans ce cas, la gravité est la seule source d'énergie.

Naturellement, nous contraignons le solveur à utiliser les mêmes coefficients pour les articulations symétriques.

E.4.5 Paramètres des corps rigides

Tous les robots présentés dans ces résultats utilisent des paramètres fixés et anthropométriques pour ce qui est des tailles des segments, de leurs masses, de la position de leur centre de masse et de leurs matrices d'inertie, calculés à partir d'une taille et d'un poids de référence gardé constant entre les différents tests. Ces paramètres suivent les tables anthropométriques données par Dumas, Cheze, et Verriest (2007).

Notre modèle minimal M_A est composé d'un bassin, de deux cuisses et deux jambes, où seules les hanches sont actionnées. Dans le modèle M_B, les genoux sont également actionnés.

Au-dessus de cette base, nous ajoutons dans les modèles M_C, M_D et M_E un torse et une tête. Le cou n'est actionné que dans le modèle M_D.

Enfin, nous ajoutons deux bras et deux avant-bras dans le modèle M_E avec des épaules actionnées.

Toutes les articulations actionnées sont des liaisons pivot d'axes parallèles.

E.5 Résultats des premiers tests

Dans cette section, nous examinons les résultats des six expériences présentées dans la table |E-1| sous forme de cinq comparaisons. Ces résultats sont également disponibles en vidéo [1].

Ces comparaisons servent à illustrer le potentiel et l'utilité de la méthode, et ne doivent pas être sorties de ce contexte d'illustration.

E.5.1 Influence des genoux

L'influence de l'actionnement des genoux pour un marcheur bipède est mis en évidence en comparant les modèles M_A et M_B de la table |E-1|.

1. https://hal.archives-ouvertes.fr/hal-01360450v2/file/3D.mp4

Entrées						
Modèle	M_A	M_B	M_C		M_D	M_E
Description	Compas	M_A avec des genous	M_A avec un buste		M_C avec un cou	M_C avec des bras
Masse totale	33.1 kg	33.1 kg	60.3 kg	60.3 kg	60.3 kg	65.9 kg
Actionnement	actif	actif	actif	passif	actif	actif
Dimension	2D	2D	2D	2D	2D	3D
Sorties						
CoT	0.1007	0.0544	0.0618	0.0499	0.0621	0.0651
Longueur	0.56 m	0.85 m	0.40 m	0.40 m	0.40 m	0.41 m
Vitesse	0.70 m/s	1.06 m/s	0.50 m/s	0.50 m/s	0.50 m/s	0.51 m/s
Performances						
Itérations	56	40	76	9	66	108
Durée	2.8 s	2.9 s	2.6 s	1.2 s	4.1 s	7.6 s

Table |E-1| : Six marcheurs bipèdes sont comparés afin de déterminer l'influence des genous, du buste, du cou, des bras, et du type d'actionnement. Pour chaque exemple, l'algorithme nous fourni la démarche, le coût final de transport optimisé et la longueur d'un pas. Les deux dernières lignes donnent les performances de l'algorithme. Dans tous ces exemples, la durée d'un pas est fixée à 0.8 secondes et la pente du sol à 0.05 radians.

Les résultats expérimentaux montrent que, dans notre cas, ajouter des genoux à un simple compas divise environ le coût de transport par deux, tout en augmentant la longueur optimale d'un pas. On note également que le coût additionnel en termes de complexité et de temps de calcul est négligeable.

NB : Dans le cas d'un marcheur bipède réel, sans genoux, on ne peut marcher que sur un terrain parfaitement prévu pour. Sans cela, la jambe qui se balance heurterait le sol au moment où l'angle entre les deux jambes s'annule.

E.5.2 Influence du buste

Dans ce cas, nous étudions l'impact de l'ajout d'un segment supérieur, correspondant au torse et à la tête rigidement fixés l'un à l'autre et au bassin. Cette situation correspond aux modèles respectivement M_A et M_C de la table |E-1|, en ne considérant que l'actionnement actif.

On remarque que la modification d'un corps sans l'ajout de degré de liberté supplémentaire permet de réduire dans ce cas le coût de transport de 40 %. Cela réduit également la longueur optimale du pas à 40 cm, ce qui est la borne inférieure que nous avons autorisé.

Par ailleurs, le temps nécessaire à la convergence reste identique.

E.5.3 Influence du cou

Ici, nous considérons l'influence de l'actionnement du cou, en comparant les modèles M_C et M_D pour un même actionnement.

Ces expériences nous montrent que l'accroissement du coût de transport entre ces deux cas est inférieur au pourcent, alors que nous avons ajouté un actionneur et dépensons donc plus d'énergie. Cependant, le solveur a besoin d'un peu plus de temps pour converger.

E.5.4 Comparaison des types d'actionnement

Cette comparaison considère seulement le modèle M_C, et étudie les différences entre un contrôleur idéal actif et un contrôleur passif.

Si l'on compare les résultats des troisième et quatrième colonnes de la table |E-1|, il apparaît que le coût de transport, tel que nous le calculons à travers l'équation (E-7), est supérieur pour le marcheur passif. Le temps de calcul est lui inférieur dans le cas d'un actionnement de type Proportionnel Dérivé.

Cela peut être expliqué par la dimensionnalité du problème : dans le premier cas, le contrôleur est composé du nombre de nœuds de multiple shooting de splines cubiques, tandis que dans le second cas il n'a que trois paramètres scalaires. Il est donc plus complexe du point de vue du solveur, mais permet d'atteindre un meilleur résultat.

E.5.5 Passage à la troisième dimension

Notre méthode est générale et permet de considérer aussi bien des modèles 3D que des modèles 2D. Cependant, afin de permettre au robot de garder son équilibre sur le lacet, nous lui avons ajouté des bras dans le modèle M_E.

En comparaison avec le modèle 2D sans bras M_C, le coût de transport est légèrement supérieur, mais reste inférieur à celui du premier compas M_A.

On remarque également que le temps de calcul et le nombre d'itérations du solveur sont supérieurs, mais même dans ce cas complexe, l'efficacité de MUSCOD-II et celle de Pinocchio nous permettent de rester sous la barre des dix secondes.

E.6 Étude d'actionneurs

Dans cette section, nous adaptons notre méthode aux actionneurs composés d'un moteur et d'un ressort, montés en série ou en parallèle.

Les Series Elastic Actuators (SEA) (Pratt et Williamson 1995) représentent une nouvelle génération d'actionneurs en ajoutant un élément d'une élasticité non négligeable entre un moteur et le segment qui lui est associé.

Ces actionneurs ont déjà été utilisés sur des robots humanoïdes (Tsagarakis et al. 2013). Ils peuvent en améliorer les performances, notamment au niveau de la gestion des impacts, de l'efficacité énergétique, et de la sécurité à la fois pour le robot et les humains.

Les Parallel Elastic Actuators (PEA) (Grimmer et al. 2012) sont quant à eux utiles pour emmagasiner de l'énergie, et ainsi réduire les pics de puissance consommée.

Nous comparons donc les cas suivants :

- Actionneurs :
  - moteurs rigides ;
  - moteurs et ressorts en série ;
  - moteurs et ressorts en parallèle ;
- Modèles :
  - Compas 2D M_A ;
  - Corps complet 3D M_E ;
- Fonction de coût :
  - Coût de transport ;
  - Norme au carré du couple.

E.6.1 État du système

Dans le cas de moteurs rigides ou de moteurs et ressorts montés en parallèle, l'état du système est défini par les positions et vitesses angulaires des articulations, comme le montre l'équation (E-9)

$$\boldsymbol{x} = \left(\boldsymbol{q}^\top \dot{\boldsymbol{q}}^\top\right)^\top \tag{E-9}$$

Cependant, dans le cas d'actionneurs série élastiques, la position angulaire d'un moteur ne suffit pas pour avoir la position angulaire de l'articulation. On a donc besoin de considérer à la fois la position angulaire du moteur (après un éventuel réducteur), que l'on note θ et celle de l'articulation, qui reste notée q. Cela nous donne donc l'équation (E-10).

$$\boldsymbol{x} = \left(\boldsymbol{q}^\top \dot{\boldsymbol{q}}^\top \boldsymbol{\theta}^\top \dot{\boldsymbol{\theta}}^\top\right)^\top \tag{E-10}$$

Naturellement, comme précédemment, nous utilisons un seul jeu de paramètres qualifiant un ressort donné pour une paire d'articulations symétriques.

E.6.2 Fonctions de coût

L'expression du coût de transport est légèrement modifiée, comme le montre l'équation (E-11).

$$c_{\text{CoT}} = \int_{t=0}^{T} \frac{|\boldsymbol{\tau}|^\top |\dot{\boldsymbol{\theta}}|}{d} dt \tag{E-11}$$

On notera que, contrairement aux expériences précédentes, ce coût de transport n'est pas le coût de transport sans unités classique. Celui-ci s'exprime en Watts par mètre.

Dans un second lot d'expériences, nous utilisons plutôt le carré de la norme du couple, normalisé par la longueur d'un pas, selon l'équation (E-12). Cette fonction est utile pour limiter l'énergie dépensée par les moteurs, et Schultz et Mombaur (2010) ont montré que cela pouvait produire des comportements semblant plus naturels.

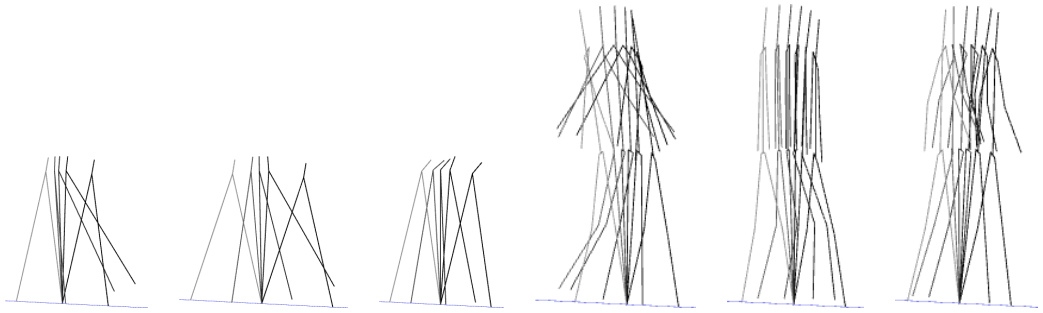
$$c_{\boldsymbol{\tau}} = \int_{t=0}^{T} \frac{\|\boldsymbol{\tau}\|^2}{d} dt \tag{E-12}$$

E.6.3 Expériences

Dans cette section, nous présentons les expériences menées sur l'optimisation de marcheurs de deux modèles, utilisant trois types d'actionneurs, séparées en deux tables selon la fonction de coût utilisée. Ces résultats sont également disponibles en vidéo [2].

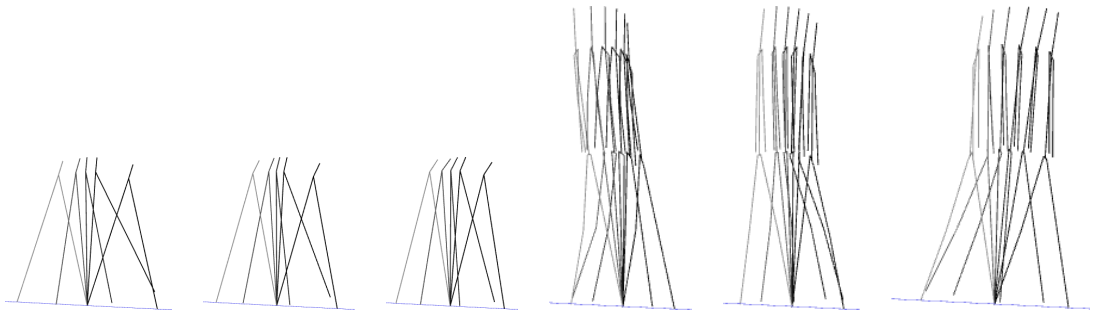
2. http://homepages.laas.fr/gsaurel/iros.mp4

Table |E-2| : Expériences utilisant une fonction de coût c_{τ}.



Modèle	M_A	M_A	M_A	M_E	M_E	M_E
Actionneurs	Rigide	Série	Parallèle	Rigide	Série	Parallèle
Coût [N^{2}m]	0.09563	0.00004	0.00060	0.62844	0.12800	0.36654
Longueur [m]	0.6	0.92726	0.65017	0.6	0.6	0.6
Durée [s]	0.74361	0.90204	0.73318	0.64572	0.81010	0.51664
Vitesse [m/s]	0.80687	1.02795	0.88679	0.92919	0.74065	1.16135

Table |E-3| : Expériences utilisant une fonction de coût c_{CoT}.



Modèle	M_A	M_A	M_A	M_E	M_E	M_E
Actionneurs	Rigide	Série	Parallèle	Rigide	Série	Parallèle
Coût [W/m]	0.45729	0.04651	0.03863	0.51831	0.00844	0.04792
Longueur [m]	0.91416	0.78682	0.69466	0.6	0.6	0.85526
Durée [s]	0.80094	0.87833	0.61705	0.59480	0.82184	0.47053
Vitesse [m/s]	1.14136	0.89581	1.12577	1.00875	0.73007	1.81764

E.6.4 Résultats

Les douze expériences présentées dans les tables |E-2| et |E-3| illustrent une nouvelle fois le potentiel et l'utilité de notre méthode.

Naturellement, ces résultats comparatifs ne sont pas encore suffisants pour affirmer qu'un critère est plus important à optimiser qu'un autre, ou qu'un type d'actionneur est meilleur qu'un autre en général.

Cependant, ces expériences nous permettent de faire plusieurs remarques en les comparant.

Dans un premier temps, on observe effectivement que les marcheurs corps complet ont des démarches semblant plus naturelles pour $c_{\boldsymbol{\tau}}$ que pour c_{CoT}, et notamment sur la vidéo.

Ensuite, on constate systématiquement que le coût est inférieur pour les actionneurs aidés de ressorts que pour les actionneurs rigides.

Aussi, si l'on compare les actionneurs en série ou en parallèle avec des ressorts, on observe des oscillations rapides dans le premier cas et pas le second. En d'autres termes, il semble que dans les cas étudiés ici, les SEA stressent plus fortement les arbres des moteurs, ce qui pourrait éventuellement s'avérer néfaste. Ce comportement n'est pourtant pas étonnant, dans la mesure où la solution optimale n'est pas forcément lisse. De plus, un comportement oscillatoire est typique des SEAs.

Enfin, on note contre-intuitivement qu'il ne semble pas y avoir de corrélation dans ces expériences entre la vitesse de déplacement et le coût. Ceci pourrait par exemple nous permettre de concevoir des robots à la fois rapides et économes en énergie, en prenant en compte la vitesse dans la fonction de coût (Chevallereau et Aoustin 2001).

E.7 Étude de la stabilisation de la tête

Dans cette section, nous ajoutons à notre méthode un mécanisme de simulation d'une centrale inertielle incorporée au robot lui permettant de mieux se stabiliser.

Dans le cadre du projet Yoyo-Man (Laumond et al. 2017), il est intéressant d'étudier l'effet de la stabilisation de la tête sur la locomotion bipède.

Dans ce but, nous avons implémenté dans notre cadre logiciel un estimateur de position de la tête imitant le système vestibulaire humain suivant Farkhatdinov, Hayward, et Berthoz (2011).

Il suffit ensuite d'ajouter un terme dans la fonction de coût visant à minimiser l'angle entre cet estimateur et la verticale, suivant l'équation (E-13).

$$c_\beta = c_{\text{CoT}} + W\beta \tag{E-13}$$

Dans l'équation (E-13), β est l'angle mesuré entre le pendule inversé reproduisant le système vestibulaire humain et la verticale, et W un poids indiquant l'importance de l'objectif de la stabilisation de tête par rapport à l'objectif c_{CoT}.

La figure E-2 présente l'un des premiers résultats de démarche que l'on obtient dans ce cas.

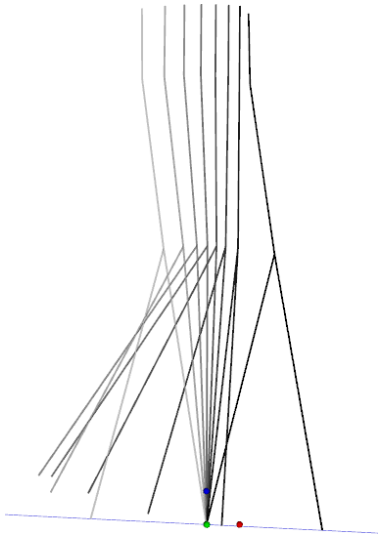


Figure E-2 : Marcheur bipède à quatre segments, articulé par deux hanches et un cou. La fonction de coût incorpore un objectif de stabilisation verticale de l'estimateur du système vestibulaire. Le marcheur adopte ici une stratégie consistant à lancer sa jambe en arrière pour mieux profiter de son inertie.

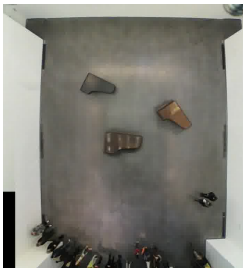
E.8 Fabrication d'un prototype

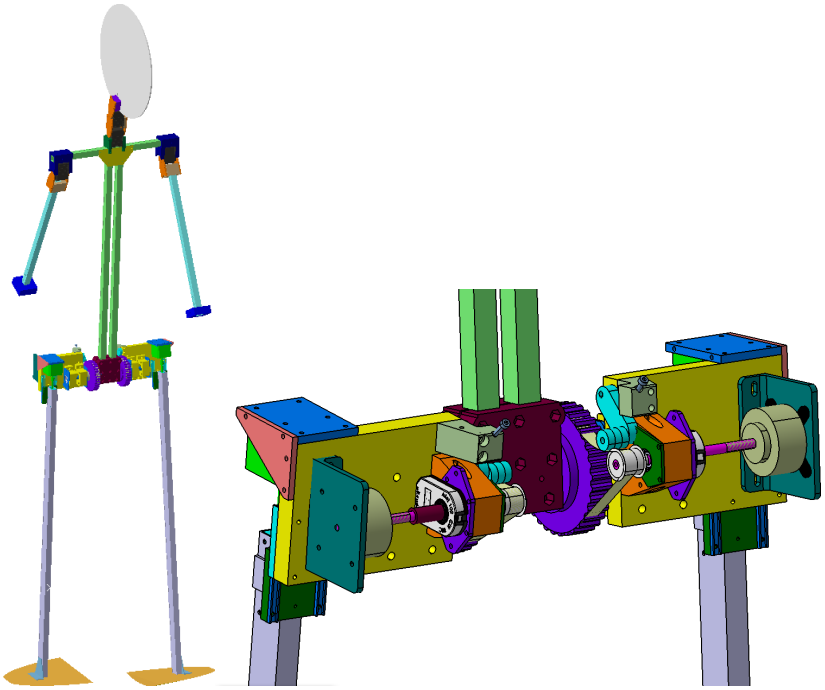
L'étape suivant la simulation de robots est naturellement le prototypage physique. Dans cette section, nous décrivons les travaux visant à fabriquer un robot réel, utilisant notre méthode d'optimisation.

Ce premier prototype a pour objectif d'étudier en situation réelle l'effet de la stabilisation de la tête par rapport à la verticale, en suivant les concepts de la section E.7. Ce premier bipède doit rester le plus simple possible. Nous utilisons donc un modèle composé de deux jambes, attachées au buste par des hanches à deux degrés de liberté. Le premier permet à la jambe de se balancer d'avant en arrière, et le second permet de lever la jambe pour éviter qu'elle heurte le sol au mauvais moment, de la même manière que ce qui a été réalisé par Bhounsule et al. (2014), mais pour un bipède.

Deux bras sont également présents, et servent à contrecarrer le moment de lacet créé par le balancier des jambes. Et bien sûr, un cou articulé vise à stabiliser la tête verticalement.

Les pieds sont arrondis, de manière à orienter la marche dans la direction qui empêche une chute latérale (Wisse et Van der Linde 2007b), ainsi qu'à entretenir un mouvement de balancier.





(a) Corps Complet (b) Détail des hanches. Un degré de liberté en rotation est visible à l'avant, et un en translation à l'arrière.

Figure E-3 : Modèle CAD du prototype.

E.9 Perspectives

Dans les sections E.7 et E.8, nous avons brièvement mentionné les principaux travaux en cours au moment de la rédaction de cette thèse. Dans cette section, nous énumérons les différents points encore en suspens.

E.9.1 Actionneurs

L'étude des actionneurs comprenant des ressorts réalisée dans la section E.6 ouvre diverses pistes pour des travaux futurs. Par exemple, on pourrait envisager d'utiliser des actionneurs combinant un ressort en série et un autre en parallèle (SE+PEA), ou encore des actionneurs à impédance variable (VSA).

On pourrait aussi chercher à optimiser le type d'actionneur à utiliser en fonction de chaque articulation, étant donné les différences entre leurs besoins. Ainsi, en pratique, il n'est pas forcément utile d'avoir un actionneur complexe, donc lourd et onéreux au niveau des épaules, si notre seul objectif est la locomotion. Ces paramètres peuvent aisément être ajoutés dans la fonction de coût.

La marche n'est pas non plus le seul mode de locomotion bipède [3], il serait donc possible de chercher à étudier dans notre modèle l'intérêt de ces actionneurs pour la course ou le saut (Grimmer et al. 2012).

Par ailleurs, il pourrait être intéressant de prendre en compte des modèles plus réaliste d'actionneurs, notamment dans le cas où ceux-ci sont capables de fonctionner également en générateur de courant. Par exemple les actionneurs « proprioceptifs » du robot Cheetah (Wensing et al. 2017) sont particulièrement performants pour absorber les chocs, notamment grâce à leur rétroactivité.

Cela nous permettrait ainsi de pouvoir utiliser une fonction de coût représentant plus fidèlement la consommation énergétique réelle du robot.

E.9.2 Pieds ellipsoïdaux

Carpentier et al. (2015) ont proposé une formulation analytique de contacts de roulement sans glissement qui est compatible avec la formulation dynamique présentée en section E.2.4. L'ajout de ce modèle de contacts à notre méthode pourrait grandement améliorer la qualité des mouvements générés ainsi que nous permettre de gérer des scenario plus complexes.

Comme nous l'avons vu dans la section E.8, nous avons pour objectif d'utiliser des pieds ronds pour notre premier prototype, en suivant Wisse et Van der Linde (2007b). L'ajout de cette formulation dans notre programme est donc l'une de nos priorités.

3. Sans compter les nombreux modes de locomotion multipèdes, que nous n'avons pour l'instant pas l'intention de traiter.

E.9.3 Stabilité

Pour l'instant, notre méthode produit une trajectoire de référence cyclique encodant un régime permanent de marche. Rien n'est prévu au cours de l'optimisation pour garantir la stabilité, et nous ne pouvons donc que la vérifier a posteriori.

L'étude de la stabilité d'un marcheur oscillant autour d'un cycle limite est généralement effectuée par une observation de la section de Poincaré (Grizzle et al. 2010). Benallegue et Laumond (2013) reprennent cette méthode et proposent comme mesure l'espérance du nombre de pas qu'un robot en boucle ouverte peut faire avant de tomber (MFPT). Cette métrique pourrait être directement inclue dans notre fonction de coût.

Outre l'étude d'un terrain ayant une texture présentée par Benallegue et Laumond (2013), il est aussi possible de rechercher la hauteur maximale d'une marche inattendue que le robot pourrait descendre sans tomber (Wisse et Van der Linde 2007a). Cette métrique a l'avantage d'être bien plus parlante, mais son calcul est plus complexe à mettre en œuvre dans le cadre de notre méthode.

Une autre méthode classique utilise une analyse de Monte-Carlo. Elle est bien plus simple à mettre en place, mais risque de faire exploser les temps de calculs au vu des dimensions du problème que l'on traite.

Il est également envisageable de poser un second problème d'optimisation au-dessus de l'actuel, visant à optimiser la stabilité du contrôle en boucle ouverte (Mombaur 2001).

Enfin, une solution plus complexe sur le plan théorique et encore peu explorée serait d'utiliser des techniques de contrôle optimal afin de générer un contrôleur en boucle ouverte de référence, puis d'extraire ce contrôleur pour le réutiliser dans un contexte de contrôle rétroactif. Le contrôle rétroactif est déjà largement répandu dans le cadre de la locomotion bipède (Westervelt et al. 2007), et permet d'assurer la stabilité de la marche dans des conditions bien plus vastes.

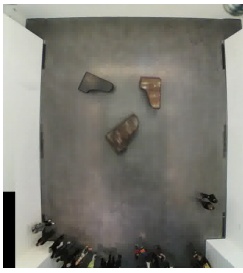
E.9.4 Fonctions de coût

Le choix d'une fonction de coût reste un problème ouvert, alors même qu'il est l'une des questions les plus importantes en contrôle optimal.

Comme nous l'avons fait dans les travaux présentés dans la section E.6, on peut trouver dans la littérature différents résultats en fonction de différentes fonctions de coût (Chevallereau et Aoustin 2001), ou des indications sur le choix de ces fonctions de coût (Tassa, Erez, et Todorov 2012).

Tout au long de ce chapitre, nous avons évoqué diverses métriques qu'il serait judicieux d'ajouter dans cette fonction de coût. Cependant, il n'est pas aisé d'arriver à faire convenablement cohabiter plusieurs termes au sein d'une telle fonction. Pour cela, un paramètre de poids peut être utilisé comme dans l'équation (E-13), mais il parait plus raisonnable d'établir une hiérarchie dans les objectifs à la manière de Geisert et al. (2017).

Conclusion



Chapitre F

Conclusion

Au cours de cette thèse, nous avons étudié la génération de mouvement ainsi que la planification de trajectoire pour la locomotion de différents types de robots mobiles et bipèdes.

Ce chapitre de conclusion récapitule nos contributions à la maîtrise de ces modes de locomotion, et présente diverses perspectives.

F.1 Robotique mobile

Dans la partie I, nous avons utilisé trois classes de robots mobiles :

- des robots différentiels de classe $(2, 0)$, dans le chapitre B ;
- des robots à tourelle de classe $(1, 1)$, dans le chapitre C ;
- des robots tri-tourelles de classe $(1, 2)$, dans le chapitre D.

Il en ressort que des robots différentiels sont bien plus facile à maîtriser que des robots car-like. En effet, ces deux classes de robots ont le même nombre de degrés de manœuvrabilité, mais la première a deux degrés mobilité alors que la seconde n'en a qu'un, compensé par un dégré de dirigeabilité.

Cela a notamment engendré des complications lors de la mise en œuvre des trajectoires que nous générons sur le prototype développé par BA Robotics Systems, et particulièrement au niveau des points de rebroussement. Pour un robot de classe $(1, 1)$, il est nécessaire de s'arrêter à chaque point de rebroussement pendant la durée nécessaire à la modification du degré de dirigeabilité.

Ensuite, en comparant les robots mobiles omnidirectionnels avec des robots ne pouvant pas se déplacer latéralement, il apparaît que ce gain implique un grand coût de complexité, tant sur le design mécanique du robot que sur les algorithmes de contrôle. En effet, à moins d'utiliser des roues omnidirectionnelles (figure F-1), le robot a forcément plus d'actionneurs que de degrés de liberté. Les algorithmes de contrôle doivent donc s'assurer que les degrés de liberté supplémentaires soient correctement asservis par rapport aux actionneurs principaux.

Figure F-1 : Une roue omnidirectionnelle. Ce mécanisme est rarement fabriqué pour des robots de plusieurs tonnes, mais il aurait en théorie grandement facilité le contrôle bas niveau des robots du projet transhumus.

F.1.1 Contributions

Pour les projets offroad et transhumus, vus dans les chapitres B et D, nous avons créé des mouvements répondant principalement à des contraintes esthétiques. Ces mouvements sont générés par des variables extérieures, telles que la force et la direction du vent ou la vitesse de flux de sève. Ces générations de mouvement assurent également des contraintes de couverture de surface et d'évitement d'obstacles connus a priori.

Pour le projet LEMON vu dans le chapitre C, nous avons généré des trajectoires de suivi des murs et de balayage des surfaces, utilisant une carte créée à l'aide de techniques de SLAM, et essayant de minimiser le nombre de manœuvres à effectuer.

F.1.2 Perspectives

Les trois projets vus dans cette première partie ont tous été réalisés en un temps extrêmement court suivant les contraintes opérationnelles.

Les deux projets artistiques sont suffisament aboutis sur le plan théorique, et ont prouvé qu'ils ont pu fonctionner pendant la durée de leurs expositions respectives. Cependant, une réécriture de certaines parties du code utilisé ainsi qu'un plus grand nombre de tests et de gestion d'erreurs rendraient ces œuvres plus faciles à installer, plus maintenables, et plus pérennes.

Aussi, pour ces deux projets, il serait particulièrement utile d'améliorer le système de géolocalisation, par exemple en recoupant les données issues de technologies différentes (vision, laser, ondes UWB, odométrie, ultrasons, etc.).

Concernant le projet LEMON, des perspectives ont déjà été évoquées dans la section C.8. Cependant, à la lumière de notre comparaison entre différentes classes de robots, il apparaît que pour cette application, utiliser un robot différentiel permettrait de résoudre les problèmes de suivi des trajectoires que nous générons, de même que de les parcourir en un temps réduit, puisqu'il ne serait plus nécessaire de s'arrêter pour manœuvrer à chaque point de rebroussement.

F.2 Robotique Humanoïde

Dans la partie II, nous avons étudié la génération de mouvements pour différents marcheurs bipèdes.

F.2.1 Contributions

Nous avons pour cela créé et prototypé une méthode de codesign de marcheurs bipèdes utilisant la librairie de calculs dynamique Pinocchio et le solveur de contrôle optimal MUSCOD-II.

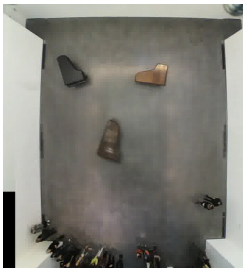
Nous avons ensuite testé ce prototype logiciel et montré qu'il peut servir à comparer différentes architectures de chaînes polyarticulées, différentes méthodes d'actionnement, et optimiser différents critères.

F.2.2 Perspectives

Nous avons vu le potentiel de notre méthode, et il reste donc naturellement à l'utiliser dans des cas concrets, par exemple pour choisir le type d'actionnement d'un robot particulier, ou de générer différentes trajectoires de marche suivant une liste pré-définie de critères à optimiser, ou encore d'optimiser le design mécanique lors de la création d'un nouveau robot.

Outre les questions relatives à la gestion d'autres types d'actionneurs, de pieds ellipsoïdaux, de la stabilité, ainsi qu'au choix d'une fonction de coût vues dans la section E.9, il serait également intéressant d'essayer d'utiliser un solveur de contrôle optimal libre. En effet, MUSCOD-II est un excellent logiciel, mais sa licence fermée nous empêche actuellement de mieux diffuser notre méthode et de la rendre accessible au plus grand nombre.

Annexes

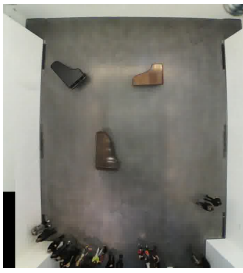


Annexe 1

Algorithmes complémentaires pour le projet LEMON

Algorithme 1-1 Liste des configurations $\bar{\boldsymbol{q}}$ suivant la droite (ρ, θ) orientée en σ

```
1 : procedure Line(ρ, θ, σ)
2 :     S, E ← findExtremities(ρ, θ)        ▷ Début et fin de l'intersection de la droite (ρ, θ) et des bords du Bitmap
3 :     if σ < 0 then
4 :         swap(S, E)
5 :     end if
6 :     q̄ ← [(x, y, σθ)∀(x, y) ∈ linspace(S, E)]
7 :     return q̄
8 : end procedure
```



Algorithme 1-2 Recherche des extrémités S, E de la droite (ρ, θ) sur le `Bitmap`

1 : procedure findExtremities(ρ, θ)
2 : if $\sin(\theta) = 0$ then
3 : $S \leftarrow \begin{pmatrix} x_{min} + \rho \\ y_{min} \end{pmatrix}$
4 : $E \leftarrow \begin{pmatrix} x_{min} + \rho \\ y_{max} \end{pmatrix}$
5 : else if $\cos(\theta) = 0$ then
6 : $S \leftarrow \begin{pmatrix} x_{min} \\ y_{min} + \rho \end{pmatrix}$
7 : $E \leftarrow \begin{pmatrix} x_{max} \\ y_{min} + \rho \end{pmatrix}$
8 : else
9 : $A \leftarrow \begin{pmatrix} x_{min} \\ y_{min} + \frac{\rho}{\sin(\theta)} \end{pmatrix}$
10 : $B \leftarrow \begin{pmatrix} x_{min} + \frac{\rho}{\cos(\theta)} \\ y_{min} \end{pmatrix}$
11 : $C \leftarrow \begin{pmatrix} x_{max} \\ y_{min} + (\rho - (y_{max} - y_{min})\,\mathrm{cotan}(\theta)) \end{pmatrix}$
12 : $D \leftarrow \begin{pmatrix} x_{min} + (\rho - (x_{max} - x_{min})\tan(\theta)) \\ y_{max} \end{pmatrix}$
13 : $start \leftarrow false$
14 : if $y_{min} \leqslant y_A \leqslant y_{max}$ then
15 : $start \leftarrow true$
16 : $S \leftarrow A$
17 : end if
18 : if $x_{min} \leqslant x_B \leqslant x_{max}$ then
19 : if $start$ then
20 : $E \leftarrow B$
21 : else
22 : $start \leftarrow true$
23 : $S \leftarrow B$
24 : end if
25 : end if
26 : if $y_{min} \leqslant y_C \leqslant y_{max}$ then
27 : if $start$ then
28 : $E \leftarrow C$
29 : else
30 : $S \leftarrow C$
31 : end if
32 : end if
33 : if $x_{min} \leqslant x_D \leqslant x_{max}$ then
34 : $E \leftarrow D$
35 : end if
36 : end if
37 : return S, E
38 : end procedure

Annexe 2

Implémentation logicielle pour le projet transhumus

Comme nous l'avons vu dans la section D.3, nous avons utilisé la bibliothèque logicielle ZeroMQ comme base de notre architecture logicielle.

Cette architecture modulaire peut être schématisée suivant la figure 2-1 :

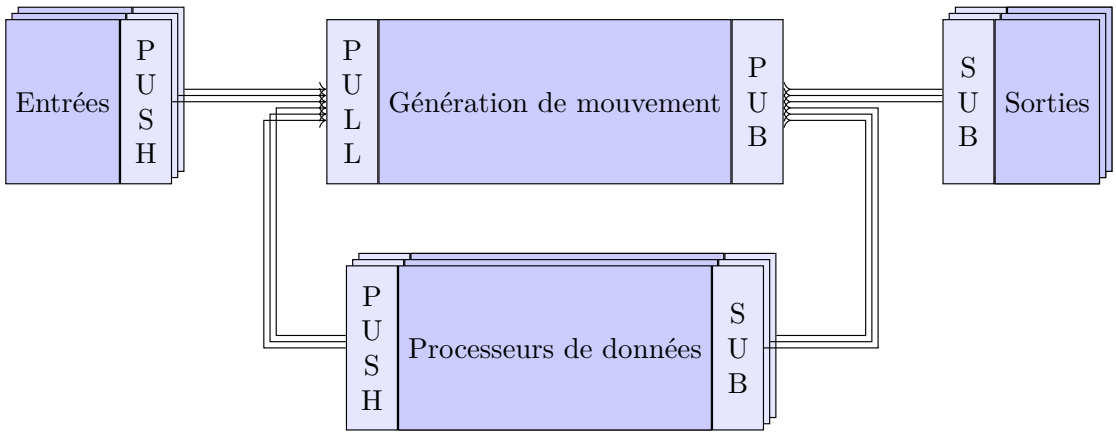


Figure 2-1 : Modules de l'architecture logicielle du projet transhumus

Le composant principal de cet architecture est donc le générateur de trajectoires. Il ouvre à a fois un socket de type Pull pour que d'autres modules puissent lui envoyer des données, et un socket de type Pub pour publier les données aux modules qui en ont besoin.

Ainsi, tous les autres composants se connectent grâce à un composant de type PUSH au socket PULL et/ou un composant de type SUB au socket PUB, et peuvent donc être ajoutés, enlevés et rechargés à la volée. Chaque composant peut ainsi traiter chaque information à son rythme, être codé dans n'importe quel langage et être éxécuté sur n'importe quel ordinateur.

Pour simplifier les choses nous n'utilisons que des données sérialisables au format JSON. Par exemple, dans notre cas, le dictionnaire des données à traiter pour chaque AGV est donné dans le code source 2-1.

Code source **2-1** Valeurs initiales des données utilisées par chaque AGV.

```
DATA = {
    # Status:
    'status': 'Not connected', 'errors': 'Not connected', 'anomaly': False,
    'is_up': False, 'inside': False, 'last_seen_agv': str(datetime(1970, 1, 1
    # Position:
    'x': 0, 'y': 0, 'a': 0,
    # Vitesse:
    'v': 0, 'w': 0, 't': 0,
    # Vitesse à atteindre
    'vg': 0, 'wg': 0, 'tg': 0,
    # Vitesses et orientations des tourelles à atteindre:
    'vt': [0, 0, 0], 'tt': [0, 0, 0],
    # Vitesses et orientations des tourelles de consigne pour l'AGV:
    'vc': [0, 0, 0], 'tc': [0, 0, 0],
    # Vitesses et orientations des tourelles réelles lues sur l'AGV:
    'vm': [0, 0, 0], 'tm': [0, 0, 0],
    # Nombre de tour des tourelles
    'nt': [0, 0, 0],
    # Valeurs absolues des sondes graniers et normalisation
    'granier': [0] * N_PROBES, 'gm': [10] * N_PROBES,
    'gmi': [-10] * N_PROBES, 'gma': [0] * N_PROBES,
    # Commandes
    'stop': False, 'smoothe': False, 'smoothe_speed': True, 'boost': False,
    # Destination et état (cf. l'algorithme |D-3|):
    'destination': [0, 0], 'state': -1,
}
```

2.1 Briques élémentaires des transferts de données

Dans cette section nous détaillons les briques élémentaires qui permettent de créer les bases des modules. Ceux-ci seront présentés dans les sections suivantes.

Code source **2-2** `vmq/vmq.py` : composant abstrait servant de base à tous les modules.

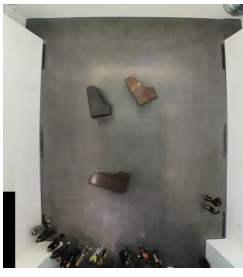
```
from argparse import ArgumentParser
from pprint import pprint

from zmq import Context

from ..settings import MAIN_HOST, MAIN_HOSTS, Host


class VMQ(object):
    def __init__(self, main, host, verbosity, *args, **kwargs):
        self.main, self.host, self.verbosity = main, Host[host], verbosity
        self.hosts = [Host.ame]
        self.log(self.hosts)
        self.context = Context()
        self.data = {h: {} for h in self.hosts}
        self.ended = False

    def run(self):
        try:
            while not self.ended:
                self.loop()
        except:
            pass
        finally:
            self.end()
            print()

    def loop(self):
        # Abstract Class
        raise NotImplementedError

    def end(self):
        print('terminating…')

    def log(self, data=None):
        if data is None:
            data = self.data
        if self.verbosity > 1:
            pprint(data)
        elif self.verbosity > 0:
            print(data)


parser = ArgumentParser(conflict_handler='resolve')
parser.add_argument('-M', '--main', default=MAIN_HOST.name, choices=MAIN_HOSTS,
                    help="main host")
parser.add_argument('-H', '--host', default='ame', choices=[h.name for h in Host],
                    help="source host")
parser.add_argument('-V', '--verbosity', action='count', default=0,
                    help="set verbosity")
```

Code source **2-3** `vmq/publisher.py` : composant de base permettant de publier des données.

```
from zmq import PUB

from ..settings import PORT_PUB
from .vmq import VMQ


class Publisher(VMQ):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

        self.publisher = self.context.socket(PUB)
        self.publisher.bind("tcp://*:%i" % PORT_PUB)

    def pub(self):
        self.publisher.send_json(self.data)
        self.log()
```

Code source **2-4** `vmq/subscriber.py` : composant de base permettant de souscrire aux données envoyées par le publieur.

```
from datetime import datetime

from zmq import NOBLOCK, SUB, SUBSCRIBE
from zmq.error import Again

from ..settings import AGV_HOST, PORT_PUB
from .vmq import VMQ


class Subscriber(VMQ):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.subscriber = self.context.socket(SUB)
        url = "tcp://%s:%i" % (self.main, PORT_PUB)
        self.log(url)
        self.subscriber.connect(url)
        self.subscriber.setsockopt_string(SUBSCRIBE, '')
        self.last_seen = datetime(1970, 1, 1)
        self.sub(block=0)

    def sub(self, block=NOBLOCK):
        while True:
            try:
                data = self.subscriber.recv_json(block)
                for h in self.hosts:
                    if str(h.value) in data:
                        self.data[h].update(**data[str(h.value)])
                self.last_seen = datetime.now()
                self.log([self.host, data[str(self.host.value)]
                          if AGV_HOST else data])
            except Again:
                break
            if not block:
                break
```



Code source **2-5** `vmq/puller.py` : composant de base permettant de tirer des donnnées.

```
from datetime import datetime

from zmq import NOBLOCK, PULL
from zmq.error import Again

from ..settings import PORT_PUSH
from .vmq import VMQ


class Puller(VMQ):
    def __init__(self, wait=True, *args, **kwargs):
        super().__init__(*args, **kwargs)

        self.puller = self.context.socket(PULL)
        self.puller.bind("tcp://*:%i" % PORT_PUSH)

        self.last_seen = datetime(1970, 1, 1)
        if wait:
            print('Waiting for a connexion')
            self.pull(block=0)
            print('Connected')

    def pull(self, block=NOBLOCK):
        while True:
            try:
                num, data = self.puller.recv_json(block)
                if num in self.hosts:
                    self.data[num].update(**data)
                    self.last_seen = datetime.now()
            except Again:
                break
            if block != NOBLOCK:
                break
```

Code source **2-6** `vmq/pusher.py` : composant de base permettant de pousser des donnnées vers le tireur.

```
from zmq import PUSH

from ..settings import PORT_PUSH
from .vmq import VMQ


class Pusher(VMQ):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

        self.push = self.context.socket(PUSH)
        self.push.connect("tcp://%s:%i" % (self.main, PORT_PUSH))

    def send(self):
        for h in self.hosts:
            self.push.send_json([h, self.data[h]])
            self.log([h, self.data[h]])
```

2.2 Composants de base

Dans cette section, nous présentons les bases servant à créer les modules vus dans la figure 2-1, elles-même créees à partir des composants de gestion des transferts de données vus dans la section précédente.

Code source **2-7** `inputs/input.py` : base servant à créer des modules envoyant des données dans le système.

```
from argparse import ArgumentParser
from time import sleep

from ..settings import PERIOD
from ..vmq import Pusher, parser


class Input(Pusher):
    def __init__(self, period, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.period = period

    def loop(self):
        self.iteration()
        self.send()
        if self.period:
            sleep(self.period)
        else:
            self.ended = True

    def iteration(self):
        self.process(**self.data[self.host])

    def process(self, **kwargs):
        pass


input_parser = ArgumentParser(parents=[parser], conflict_handler='resolve')
input_parser.add_argument('-T', '--period', type=float, default=PERIOD,
                          help="period for sending data (0: one shot)")
```

Code source **2-8** `processors/processor.py` : base servant à créer des modules capables de recevoir des données, de les traiter, et d'en mettre à jour d'autres..

```
from argparse import ArgumentParser
from time import sleep

from ..settings import PERIOD
from ..vmq import Pusher, Subscriber, parser


class Processor(Subscriber, Pusher):
    def __init__(self, period, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.period = period

    def loop(self):
        self.sub()
        self.send(self.process(**self.data[self.host]))
        if self.period:
            sleep(self.period)
        else:
            self.ended = True

    def send(self, data):
        self.log(data)
        self.push.send_json([self.host, data])


processor_parser = ArgumentParser(parents=[parser], conflict_handler='resolve')
processor_parser.add_argument('-T', '--period', type=float, default=PERIOD,
                              help="period for processing data (0: one shot)")
```

Code source **2-9** `trajectories/base_trajectory.py` : base servant à créer le module principal générant le mouvement des AGVs.

```
from argparse import ArgumentParser
from time import sleep

from ..settings import DATA, PERIOD
from ..vmq import Publisher, Puller, parser


class BaseTrajectory(Puller, Publisher):
    def __init__(self, period, *args, **kwargs):
        super().__init__(wait=False, *args, **kwargs)
        self.period = period
        self.data = {h: DATA.copy() for h in self.hosts}
        for h in self.hosts:
            self.data[h]['host'] = h
        self.data['trajectory'] = self.__class__.__name__

    def send(self):
        self.pub()

    def loop(self):
        self.pull()
        self.update()
        self.send()
        sleep(self.period)

    def end(self):
        return

    def update(self):
        for host in self.hosts:
            self.data[host].update(**self.inside(**self.data[host]))
            self.data[host].update(**self.process_speed(**self.data[host]))
            self.data[host].update(**self.smooth_speed(**self.data[host]))
            self.data[host].update(**self.process_turrets(**self.data[host]))

    def process_speed(self, **kwargs):
        raise NotImplementedError


t_parser = ArgumentParser(parents=[parser], conflict_handler='resolve')
t_parser.add_argument('-T', '--period', type=float, default=PERIOD,
                      help="main loop period")
```

Code source **2-10** `inputs/probe.py` : base servant à créer des modules correspondant à des sondes tout en vérifiant les valeurs renvoyées.

```
from argparse import ArgumentParser

from .input import Input, input_parser


class Probe(Input):
    def __init__(self, name, mini, maxi, n_values, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.name, self.mini, self.maxi, self.n_values = name, mini, maxi, n_value
        self.data[self.host][name] = [(maxi + mini) / 2] * n_values

    def iteration(self):
        self.check_value(self.process(self.data[self.host][self.name]))

    def check_value(self, value):
        if len(value) != self.n_values:
            raise ValueError(f'len({value}) != {self.n_values}')
        for i, v in enumerate(value):
            if not self.mini <= v <= self.maxi:
                raise ValueError(f'{v} not beetwen {self.mini} and {self.maxi}')
        self.data[self.host][self.name] = value


p_parser = ArgumentParser(parents=[input_parser], conflict_handler='resolve')
p_parser.add_argument('-n', '--name', choices=['granier', 'sick', 'brightness'])
p_parser.add_argument('-m', '--mini', type=float, default=-1)
p_parser.add_argument('-M', '--maxi', type=float, default=1)
p_parser.add_argument('-N', '--n_values', type=int, default=1)
```

2.3 Composants finaux

Dans cette section, nous exposons une partie des composants finaux créés sur les bases vues dans les deux sections précédentes

Code source **2-11** `output/print.py` : module de sortie affichant régulièrement les données dans le terminal.

```
from pprint import pprint
from time import sleep

from ..settings import PERIOD
from ..vmq import Subscriber, parser


class PrintOutput(Subscriber):
    def loop(self):
        self.sub()
        pprint(self.data)
        sleep(PERIOD)


if __name__ == '__main__':
    PrintOutput(**vars(parser.parse_args())).run()
```

Code source **2-12** `inputs/granier_serial.py` : module d'entrée permettant de lire les données renvoyées par les sondes Granier.

```
from argparse import ArgumentParser
from datetime import datetime
from os.path import expanduser

from serial import Serial

from .probe import Probe, p_parser


class GranierSerial(Probe):
    def __init__(self, port, filename, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.serial = Serial('/dev/ttyUSB%i' % port, 38400)
        self.filename = expanduser(filename % self.host)

    def process(self, value):
        l = self.serial.readline().decode('ascii')
        l = l.replace('\x00', '').replace('\x04', '').split()
        try:
            datetime.strptime(' '.join(l[:2]), '%d-%m-%y %H:%M:%S')
        except:
            print('fail:', l)
            return []
        with open(self.filename, 'a') as f:
            print(';'.join(l[:5]), file=f)
        return [round(float(l[2 + s]), 5) for s in range(3)]

    def end(self):
        self.serial.close()


gs_parser = ArgumentParser(parents=[p_parser], conflict_handler='resolve')
gs_parser.add_argument('-p', '--port', type=int, default=0)
gs_parser.add_argument('-f', '--filename', default='~/granier_%i.log',
                       help="log filename")
gs_parser.set_defaults(period=30)

if __name__ == '__main__':
    GranierSerial(**vars(gs_parser.parse_args())).run()
```

Code source **2-13** `inputs/granier_random.py` : module d'entrée permettant de simuler des valeurs de sondes granier. On peut également utiliser d'anciennes données en lisant les fichiers de logs vus dans le code source 2-12.

```
from argparse import ArgumentParser
from random import random

from .probe import Probe, p_parser


class GranierRandom(Probe):
    def process(self, value, **kwargs):
        return [round(min(self.maxi, max(self.mini, v + (random() - .5) / 10)
                for v in value]


gr_parser = ArgumentParser(parents=[p_parser], conflict_handler='resolve')
gr_parser.set_defaults(name='granier', period=25, n_values=3, maxi=5, mini=0)

if __name__ == '__main__':
    GranierRandom(**vars(gr_parser.parse_args())).run()
```

Code source **2-14** `processors/granier.py` : module processeur de données permettant de normaliser les valeurs lues par les sondes Granier.

```
from argparse import ArgumentParser

from ..settings import N_PROBES
from .processor import Processor, processor_parser


class Granier(Processor):
    def process(self, granier, gmi, gma, gm, **kwargs):
        for i in range(N_PROBES):
            gmi[i] = min(granier[i], gmi[i])
            gma[i] = max(granier[i], gma[i])
            diff = gma[i] - gmi[i]
            gm[i] = round((granier[i] - gmi[i]) / diff, 5) if diff > .1 else
        return {'gma': gma, 'gmi': gmi, 'gm': gm}


g_parser = ArgumentParser(parents=[processor_parser], conflict_handler='reso
g_parser.set_defaults(name='granier', period=25, n_values=3, maxi=5, mini=0)

if __name__ == '__main__':
    Granier(**vars(g_parser.parse_args())).run()
```

Code source **2-15** `processors/is_up.py` : module processeur vérifiant que l'AGV est correctement connecté au module principal, permettant d'alerter les utilisateurs au besoin. Un système similaire fonctionne dans l'autre sens, permettant de stopper l'AGV en cas d'anomalie sur le flux de données. Naturellement, il est nécessaire que les machines du réseau soient synchronisées en NTP.

```
from datetime import datetime, timedelta

from .processor import Processor, processor_parser

DT_FORMAT = '%Y-%m-%d %H:%M:%S.%f'


class IsUp(Processor):
    def process(self, last_seen_agv, **kwargs):
        try:
            dt = datetime.strptime(last_seen_agv, DT_FORMAT)
            is_up = (datetime.now() - dt) < timedelta(seconds=2)
        except:
            is_up = False
        return {'is_up': is_up}


if __name__ == '__main__':
    IsUp(**vars(processor_parser.parse_args())).run()
```

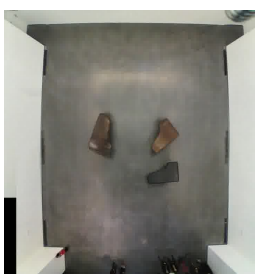
Code source **2-16** `processors/websockets.py` : module processeur convertissant les données transportées par ZeroMQ en Websockets et vice-versa.

```
import asyncio
from json import dumps, loads

from autobahn.asyncio import websocket

from ..settings import PERIOD, PORT_WS, CONSTS
from .processor import Processor, processor_parser


class MyServerProtocol(websocket.WebSocketServerProtocol):
    connections = []
    processor = Processor(**vars(processor_parser.parse_args()))

    def onConnect(self, request):
        self.connections.append(self)

    def onOpen(self):
        self.sendMessage(dumps({'consts': CONSTS}).encode())

    def onClose(self, wasClean, code, reason):
        self.connections.remove(self)

    def onMessage(self, payload, isBinary):
        self.processor.send(loads(payload.decode()))

    @classmethod
    @asyncio.coroutine
    def send_all(cls):
        while True:
            cls.processor.sub()
            for connection in cls.connections:
                connection.sendMessage(dumps({
                    'agv': cls.processor.data[cls.processor.host]
                }).encode())
            yield from asyncio.sleep(PERIOD)


if __name__ == '__main__':
    factory = websocket.WebSocketServerFactory(f"ws://0.0.0.0:{PORT_WS}")
    factory.protocol = MyServerProtocol

    loop = asyncio.get_event_loop()
    coro = loop.create_server(factory, '0.0.0.0', PORT_WS)
    send_all = asyncio.Task(factory.protocol.send_all())

    server = loop.run_until_complete(asyncio.gather(coro, send_all))

    try:
        loop.run_forever()
    except KeyboardInterrupt:
        pass
    finally:
        server.close()
        loop.close()
```

2.4 Code source complet

La totalité du code utilisé pour ce projet, ainsi que des instructions permettant de lancer rapidement un simulateur avec docker-compose, sont disponibles sur https://github.com/nim65s/venise.

Références



Arechavaleta, Gustavo, Jean-Paul Laumond, Halim Hicheur, et Alain Berthoz. 2008. « An optimality principle governing human walking ». IEEE Transactions on Robotics 24 (1). https://doi.org/10.1109/TRO.2008.915449.

Atkeson, Christopher G, BPW Babu, N Banerjee, D Berenson, CP Bove, X Cui, M DeDonato, et al. 2016. « What happened at the DARPA robotics challenge, and why ». submitted to the DRC Finals Special Issue of the Journal of Field Robotics.

Benallegue, Mehdi, et Jean-Paul Laumond. 2013. « Metastability for High-Dimensional Walking Systems on Stochastically Rough Terrain ». In Robotics : Science and Systems. Berlin, Germany. https://hal.archives-ouvertes.fr/hal-01079789.

Benallegue, Mehdi, Jean-Paul Laumond, et Alain Berthoz. 2013. « Contribution of actuated head and trunk to passive walkers stabilization ». In 2013 IEEE International Conference on Robotics and Automation. IEEE. https://doi.org/10.1109/ICRA.2013.6631387.

Bhounsule, Pranav A, Jason Cortell, Anoop Grewal, Bram Hendriksen, JG Daniël Karssen, Chandana Paul, et Andy Ruina. 2014. « Low-bandwidth reflex-based control for lower power walking : 65 km on a single battery charge ». The International Journal of Robotics Research 33 (10). https://doi.org/10.1177/0278364914527485.

Black, Paul E. 2006. « Manhattan distance ». Dictionary of Algorithms and Data Structures 18. https://xlinux.nist.gov/dads/HTML/manhattanDistance.html.

Brogliato, Bernard. 1999. Nonsmooth mechanics. Springer.

Buondonno, Gabriele, Justin Carpentier, Guilhem Saurel, Nicolas Mansard, Alessandro De Luca, et Jean-Paul Laumond. 2017. « Actuator Design of Compliant Walkers via Optimal Control ». In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2017), 7p. Vancouver, Canada. https://doi.org/10.1109/IROS.2017.8202228.

Byl, Katie, et Russ Tedrake. 2009. « Metastable walking machines ». The International Journal of Robotics Research 28 (8).

Campion, Guy, Georges Bastin, et Brigitte D'Andrea-Novel. 1996. « Structural properties and classification of kinematic and dynamic models of wheeled mobile robots ». IEEE transactions on robotics and automation 12 (1). https://doi.org/10.1109/70.481750.

Carpentier, Justin, Andrea Del Prete, Nicolas Mansard, et Jean-Paul Laumond. 2015. « An analytical model of rolling contact and its application to the modeling of bipedal locomotion ». In IMA Conference on Mathematics of Robotics. Oxford, United Kingdom. https://hal.laas.fr/hal-01182733.

Carpentier, Justin, Florian Valenza, Nicolas Mansard, et al. s. d. « Pinocchio : fast forward and inverse dynamics for poly-articulated systems ». https://stack-of-tasks.github.io/pinocchio.

Chatterjee, Anindya, et Andy Ruina. 1998. « A new algebraic rigid-body collision law based on impulse space considerations ». Journal of Applied Mechanics 65 (4).

Chevallereau, Christine, Gabriel Abba, Yannick Aoustin, Franck Plestan, Eric R Westervelt, Carlos Canudas de Wit, et Jessy W Grizzle. 2003. « Rabbit : A testbed for advanced control theory ». IEEE Control Systems Magazine 23 (5). https://doi.org/10.1109/MCS.2003.1234651.

Chevallereau, Christine, et Yannick Aoustin. 2001. « Optimal reference trajectories for walking and running of a biped robot ». Robotica 19 (5). https://doi.org/10.1017/S0263574701003307.

Collins, Steven Hartley, Andy Ruina, Russ Tedrake, et Martijn Wisse. 2005. « Efficient Bipedal Robots Based on Passive-Dynamic Walkers ». Science 307 (5712). https://doi.org/10.1126/science.1107799.

Collins, Steven Hartley, Martijn Wisse, et Andy Ruina. 2001. « A Three-Dimensional Passive-Dynamic Walking Robot with Two Legs and Knees ». The International Journal of Robotics Research 20 (7). https://doi.org/10.1177/02783640122067561.

Diehl, Moritz, Hans Georg Bock, Holger Diedam, et Pierre-Brice Wieber. 2006. « Fast Direct Multiple Shooting Algorithms for Optimal Robot Control ». In Fast Motions in Biomechanics and Robotics : Optimization and Feedback Control, édité par Moritz Diehl et Katja Mombaur. Berlin, Heidelberg : Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-36119-0_4.

Diehl, Moritz, Katja Mombaur, et Dominikus Noll. 2009. « Stability Optimization of Hybrid Periodic Systems via a Smooth Criterion ». IEEE Transactions on Automatic Control 54 (8).

Di Martino, Enzo. 2005. The history of the Venice Biennale. Papiro Arte.

Dubins, Lester E. 1957. « On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents ». American Journal of mathematics 79 (3).

Duda, Richard O, et Peter E Hart. 1972. « Use of the Hough transformation to detect lines and curves in pictures ». Communications of the ACM 15 (1).

Dumas, Raphael, Laurence Cheze, et Jean-Pierre Verriest. 2007. « Adjustments to McConville et al. and Young et al. body segment inertial parameters ». Journal of biomechanics 40 (3).

Falotico, Egidio, Nino Cauli, Przemyslaw Kryczka, Kenji Hashimoto, Alain Berthoz, Atsuo Takanishi, Paolo Dario, et Cecilia Laschi. 2016. « Head stabilization in a humanoid robot : models and implementations ». Autonomous Robots. https://doi.org/10.1007/s10514-016-9583-z.

Farkhatdinov, Ildar, Vincent Hayward, et Alain Berthoz. 2011. « On the benefits of head stabilization with a view to control balance and locomotion in humanoids ». In 2011 11th IEEE-RAS International Conference on Humanoid Robots. IEEE. https://doi.org/10.1109/Humanoids.2011.6100859.

Featherstone, Roy. 2008. Rigid body dynamics algorithms. Springer.

Geisert, Mathieu, Andrea Del Prete, Nicolas Mansard, Francesco Romano, et Francesco Nori. 2017. « Regularized Hierarchical Differential Dynamic Programming ». https://hal.archives-ouvertes.fr/hal-01356992.

Giralt, Georges, Ralph Sobek, et Raja Chatila. 1979. « A Multi-level Planning and Navigation System for a Mobile Robot : A First Approach to HILARE ». In Proceedings of the 6th International Joint Conference on Artificial Intelligence - Volume 1. IJCAI'79. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc. http://dl.acm.org/citation.cfm?id=1624861.1624936.

Granier, André. 1987. « Evaluation of transpiration in a Douglas-fir stand by means of sap flow measurements ». Tree Physiology 3 (4).

Grimmer, Martin, Mahdy Eslamy, Stefan Gliech, et André Seyfarth. 2012. « A comparison of parallel-and series elastic elements in an actuator for mimicking human ankle joint in walking and running ». In 2012 IEEE International Conference on Robotics and Automation. IEEE. https://doi.org/10.1109/ICRA.2012.6224967.

Grizzle, Jessy W, Gabriel Abba, et Franck Plestan. 2001. « Asymptotically stable walking for biped robots : Analysis via systems with impulse effects ». IEEE Transactions on automatic control 46 (1).

Grizzle, Jessy W, Christine Chevallereau, Aaron D Ames, et Ryan W Sinnet. 2010. « 3D bipedal robotic walking : models, feedback control, and open problems ». IFAC Proceedings Volumes 43 (14).

Guennebaud, Gaël, Benoît Jacob, et al. 2010. « Eigen v3 ». http ://eigen.tuxfamily.org.

Hass, Joachim, J Michael Herrmann, et Theo Geisel. 2006. « Optimal mass distribution for passivity-based bipedal robots ». The International Journal of Robotics Research 25 (11).

Hintjens, Pieter. 2013. ZeroMQ : messaging for many applications. O'Reilly Media, Inc.

Hobbelen, Daan GE, Tomas De Boer, et Martijn Wisse. 2008. « System overview of bipedal robots Flame and TUlip : Tailor-made for Limit Cycle Walking ». In Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on. IEEE. https://doi.org/10.1109/IROS.2008.4650728.

Houska, Boris, Hans Joachim Ferreau, et Moritz Diehl. 2011. « ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization ». Optimal Control Applications and Methods 32 (3).

Ikemata, Yoshito, Akihito Sano, et Hideo Fujimoto. 2006. « A physical principle of gait generation and its stabilization derived from mechanism of fixed point ». In Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on. IEEE. https://doi.org/10.1109/ROBOT.2006.1641813.

Kajita, Shuuji, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, et Hirohisa Hirukawa. 2003. « Biped walking pattern generation by using preview control of zero-moment point ». In Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on. Vol. 2. IEEE. https://doi.org/10.1109/ROBOT.2003.1241826.

Kaneko, Kenji, Fumio Kanehiro, Shuuji Kajita, Kazuhiko Yokoyama, Kazuhiko Akachi, Toshikazu Kawasaki, Shigehiko Ota, et Takakatsu Isozumi. 2002. « Design of prototype humanoid robotics platform for HRP ». In Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on. Vol. 3. IEEE.

Kuffner, James J, et Steven M LaValle. 2000. « RRT-connect : An efficient approach to single-query path planning ». In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings. Vol. 2. IEEE. https://doi.org/10.1109/ROBOT.2000.844730.

Laumond, Jean-Paul. 2016. « La robotique ». In Annales des Mines-Réalités industrielles. 4. FFE. http://www.annales.org/edit/ri/2016/resumes/novembre/09-ri-resum-FR-AN-AL-ES-novembre-2016.html.

Laumond, Jean-Paul, Mehdi Benallegue, Justin Carpentier, et Alain Berthoz. 2017. « The Yoyo-Man ». The International Journal of Robotics Research. https://doi.org/10.1177/0278364917693292.

Leineweber, Daniel B, Irene Bauer, Hans Georg Bock, et Johannes P Schlöder. 2003. « An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part 1 : theoretical aspects ». Computers & Chemical Engineering 27 (2).

Liberzon, Daniel. 2012. Calculus of variations and optimal control theory : a concise introduction. Princeton University Press.

Lu, Ping, Laurent Urban, et Zhao Ping. 2004. « Granier's thermal dissipation probe (TDP) method for measuring sap flow in trees : Theory and practice ». Acta Botanica Sinica 46 (6).

MacAdam, Barbara A. 2015. « French Pavilion Artist Céleste Boursier-Mougenot Teaches the World to Sing ». ARTnews.

McGeer, Tad. 1990. « Passive Dynamic Walking ». The international journal of robotics research 9 (2). https://doi.org/10.1177/027836499000900206.

Mirabel, Joseph, Steve Tonneau, Pierre Fernbach, Anna-Kaarina Seppälä, Mylène Campana, Nicolas Mansard, et Florent Lamiraux. 2016. « HPP : a new software for constrained motion planning ». In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Daejeon, South Korea. https://doi.org/10.1109/IROS.2016.7759083.

Mombaur, Katja. 2001. « Stability optimization of open-loop controlled walking robots ». Thèse de doctorat, University of Heidelberg.

Mombaur, Katja, et Khai-Long Ho Hoang. 2017. « How to best support sit to stand transfers of geriatric patients : Motion optimization under external forces for the design of physical assistive devices ». Journal of Biomechanics.

Moravec, Hans. 1988. Mind children : The future of robot and human intelligence. Harvard University Press.

Naveau, Maximilien, Justin Carpentier, Sébastien Barthelemy, Olivier Stasse, et Philippe Souères. 2014. « METAPOD Template META-programming applied to dynamics : CoP-CoM trajectories filtering ». In International Conference on Humanoid Robotics. Madrid, Spain. https://doi.org/10.1109/HUMANOIDS.2014.7041391.

Posa, Michael, Cecilia Cantu, et Russ Tedrake. 2014. « A Direct Method for Trajectory Optimization of Rigid Bodies Through Contact ». International Journal of Robotics Research 33 (1).

Pratt, Gill A, et Matthew M Williamson. 1995. « Series elastic actuators ». In Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots. Vol. 1. IEEE. https://doi.org/10.1109/IROS.1995.525827.

Pratt, Jerry, John Carff, Sergey Drakunov, et Ambarish Goswami. 2006. « Capture point : A step toward humanoid push recovery ». In 2006 6th IEEE-RAS international conference on humanoid robots. IEEE. https://doi.org/10.1109/ICHR.2006.321385.

Reeds, James, et Lawrence Shepp. 1990. « Optimal paths for a car that goes both forwards and backwards ». Pacific journal of mathematics 145 (2). https://doi.org/10.2140/pjm.1990.145.367.

Remy, C David, Keith W Buffinton, et Roland Siegwart. 2010. « Stability Analysis of Passive Dynamic Walking of Quadrupeds ». I. J. Robotic Res. 29.

——. 2011. « A matlab framework for efficient gait creation ». In 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE.

Sakagami, Yoshiaki, Ryujin Watanabe, Chiaki Aoyama, Shinichi Matsunaga, Nobuo Higaki, et Kikuo Fujimura. 2002. « The intelligent ASIMO : System overview and integration ». In Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on. Vol. 3. IEEE.

Saurel, Guilhem, Justin Carpentier, Nicolas Mansard, et Jean-Paul Laumond. 2016. « A Simulation Framework for Simultaneous Design and Control of Passivity Based Walkers ». In 2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots SIMPAR. San Francisco, United States. https://doi.org/10.1109/SIMPAR.2016.7862383.

Saurel, Guilhem, Michel Taïx, et Jean-Paul Laumond. 2016. « transhumus : A poetic experience in mobile robotics ». In 2016 IEEE International Conference on Robotics and Automation (ICRA). Stockholm, Sweden. https ://doi.org/ 10.1109/ICRA.2016.7487455.

Schultz, Gerrit, et Katja Mombaur. 2010. « Modeling and optimal control of human-like running ». IEEE/ASME Transactions on mechatronics 15 (5).

Schwartz, Jacob T, Micha Sharir, et John E Hopcroft. 1987. Planning, geometry, and complexity of robot motion. Vol. 4. Intellect Books.

Siciliano, Bruno, et Oussama Khatib. 2016. Springer handbook of robotics. Springer. https://doi.org/10.1007/978-3-319-32552-1.

Stewart, David, et Jeff C Trinkle. 2000. « An implicit time-stepping scheme for rigid body dynamics with Coulomb friction ». In Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on. Vol. 1. https://doi.org/10.1109/ROBOT.2000.844054.

Tassa, Yuval, Tom Erez, et Emanuel Todorov. 2012. « Synthesis and stabilization of complex behaviors through online trajectory optimization ». In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. https://doi.org/10.1109/IROS.2012.6386025.

Tedrake, Russ, Teresa Weirui Zhang, Ming-fai Fong, et H Sebastian Seung. 2004. « Actuating a simple 3D passive dynamic walker ». In Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on. Vol. 5. IEEE. https://doi.org/10.1109/ROBOT.2004.1302452.

Tran, Minh, Michel Taïx, et Philippe Souères. 2005. « Algorithmes de planification de trajectoires pour engin agricole dans des polygones non convexes ». In 3rd Conference Internationale en Informatique, Recherche Innovation et Vision du futur (RIVF'05). Hanoi, Vietnam.

Tsagarakis, Nikos G, Stephen Morfey, Gustavo Medrano Cerda, Li Zhibin, et Darwin G Caldwell. 2013. « COMpliant huMANoid COMAN : Optimal joint stiffness tuning for modal frequency control ». In 2013 IEEE International Conference on Robotics and Automation. IEEE. https://doi.org/10.1109/ICRA.2013.6630645.

Voronoı, Georges. 1908. « Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire. Recherches sur les parallélloèdres primitifs. » Journal für die reine und angewandte Mathematik 134. https://doi.org/10.1515/crll.1908.133.97.

Wensing, Patrick M, Albert Wang, Sangok Seok, David Otten, Jeffrey Lang, et Sangbae Kim. 2017. « Proprioceptive actuator design in the MIT cheetah : Impact mitigation and high-bandwidth physical interaction for dynamic legged robots ». IEEE Transactions on Robotics 33 (3). https://doi.org/10.1109/TRO.2016.2640183.

Westervelt, Eric R, Jessy W Grizzle, Christine Chevallereau, Jun Ho Choi, et Benjamin Morris. 2007. Feedback control of dynamic bipedal robot locomotion. Vol. 28. CRC press.

Wisse, Martijn, Daan GE Hobbelen, Remco JJ Rotteveel, Stuart O Anderson, et Garth J Zeglin. 2006. « Ankle springs instead of arc-shaped feet for passive dynamic walkers ». In Humanoid Robots, 2006 6th IEEE-RAS International Conference on. IEEE. https://doi.org/10.1109/ICHR.2006.321371.

Wisse, Martijn, et Richard Q Van der Linde. 2007a. Delft pneumatic bipeds. Springer.

———. 2007b. « Denise ; Sideways Stability ». Delft Pneumatic Bipeds 34.