Conception et contrôle de robots à géométrie variable : applications au franchissement d'obstacles autonome

Jean-Luc Paillat

université angers

Année : 2010
N° d'ordre : 1107

Conception et contrôle de robots à géométrie variable : applications au franchissement d'obstacles autonome.

Thèse de doctorat

Spécialité : Automatique

École doctorale : STIM

Présentée et soutenue publiquement

le : 15.11.2010

à : Angers

par : Jean-Luc Paillat

Devant le jury ci-dessous :

Rapporteurs :	Luc Jaulin	-	Professeur
	Philippe Bonnifait	-	Professeur
Examinateurs :	Philippe Wenger	-	Professeur
	Olivier Simonin	-	Maître de conférences
Directeur :	Laurent Hardouin	-	Professeur
Co-encadrant :	Philippe Lucidarme	-	Maître de conférences

Laboratoire : LISA, 62 Avenue Notre Dame du Lac - 4900 Angers

ED (N°) : 503

Remerciements

Table des matières

Table des figures

Chapitre 1

Introduction

Depuis plusieurs décennies, les robots occupent une place importante dans de nombreux procédés de fabrication industriels. Ils évoluent alors dans un environnement connu et sont programmés pour répondre à des besoins précis et répétitifs. Les travaux menés depuis une dizaine d'années démontrent cependant une volonté de les voir s'intégrer dans notre quotidien et d'évoluer désormais dans un milieu inconnu et imprévisible. Les enjeux diffèrent alors suivant le type de plate-forme considérée, en allant de la reconnaissance et du mime d'émotions humaines, au franchissement de divers obstacles pour explorer des décombres ou encore des terrains minés. Cette évolution fait aujourd'hui de la robotique, un domaine très vaste au carrefour de nombreuses disciplines. Bien que très diversifiés, les systèmes robotiques peuvent être classifiés selon 5 catégories majeures généralement admises dans la littérature :

- La robotique humanoïde : on parle aussi souvent de robots marcheurs pour ces machines construites à notre image ; c'est un domaine dans lequel les japonais sont pionniers au travers d'entreprises comme HONDA (avec le robot ASIMO), FUJITSU (avec HOAP-3) ou KAWADA (avec HRP2). Aujourd'hui, certains prototypes sont même capables de courir et de sauter. Notons que certains modèles de robots humanoïdes sont désormais accessibles au grand public comme le robot français NAO développé par la société Aldébaran Robotics.

- Les robots manipulateurs : sans doute les robots les plus utilisés dans l'industrie, ces bras mécaniques fixés au sol ou montés sur des rails permettent une manipulation rapide, précise et répétable de pièces d'assemblage. On les retrouve même aujourd'hui au parc du Futuroscope où leur fiabilité peut être évaluée lorsque l'on prend place dans une nacelle qui fait office d'organe terminal.

- Les robots aériens ou UAV (Unmanned Aerial Vehicles) : à l'origine, il s'agit de drones dédiés à l'exploration militaire. Ces avions pilotés à distance permettent alors de prendre des clichés en survolant des zones dangereuses. Récemment la recherche dans ce domaine s'est concentrée sur des robots à hélices capables de vol stationnaire dans des conditions de vent important. Dans le domaine public, on peut noter les prouesses réalisées par le Parrot AR. Drone, hélicoptère à 4 hélices entièrement pilotable grâce aux accéléromètres

d'un terminal mobile comme l'iPhone.

- Les robots sous-marins ou UUV (Unmanned Undersea Vehicles) : utilisés pour la détection de mines sous-marines ou l'exploration dans des milieux où l'homme ne peut pas aller sans courir de risques importants. Les enjeux de la recherche dans ce domaine consistent à s'orienter efficacement pour suivre une trajectoire définie à l'aide de capteurs inertiels et autres outils de mesure, tout en positionnant précisément les obstacles vus lors de la traversé. Un concours étudiant baptisé SAUCE pour l'anglais "Student Autonomous Underwater Chalenge Europe" existe depuis 2006 et permet aux universitaires d'éprouver bon nombre de prototypes. Notons qu'une équipe française y participe depuis 2007 avec le robot Saucisse de l'ENSIETA.

- Les robots mobiles terrestres ou UGV (Unmanned Ground Vehicles) : tout d'abord, rappelons que l'appellation : "robot mobile", par opposition aux robots manipulateurs, désigne l'ensemble des robots à base mobile. Dans la plupart des cas, les robots mobiles seront désignés par leur type de locomotion (aérien, marcheur, sous-marin...). Cependant, les robots mobiles à roues où à chenilles ne seront jamais qualifiés de robots "roulants", ainsi, par abus de langage, le terme "Robotique mobile" sera très fréquemment associé aux plate-formes mobiles terrestres. Ces engins peuvent être utilisés pour des missions de déminage ou de recherche de victimes. Les enjeux de cette discipline reposent sur une orientation efficace et un capacité à franchir des obstacles.

Les travaux présentés dans cette thèse s'articulent autour de l'adaptabilité et des performances liées au franchissement d'obstacles à l'aide de robots mobiles terrestres. A l'inverse des robots aériens ou sous-marins, les robots mobiles terrestres évoluent dans un milieu très accidenté et inconnu, leur déplacement est donc très souvent contraint. Des critères de stabilité doivent alors être pris en compte pour éviter une chute pouvant rendre le robot inutilisable.

Il existe de nombreux systèmes mécaniques qui augmentent les performances de ce type de robots ainsi qu'un nombre important de plate-formes différentes. Le premier chapitre de cette thèse se concentre sur la classification et la présentation de l'intérêt en terme de franchissement d'obstacle de chaque catégorie de véhicule rassemblée sous l'appellation "robotique mobile".

Un concept de robot déformable muni d'articulations actives étudié dans le cadre de cette thèse est ensuite présenté dans un second chapitre. S'en suivent alors une étude concernant les critères de stabilité adaptés à cette plate-forme puis une décomposition du comportement du robot lors de franchissements télé-opérés effectués dans le cadre de manifestations comme le concours ETAS organisé par la DGA ou l "European Robot Trial" (ELROB) entre 2006 et 2008. Notons que l'acro-

nyme B2P2 est associé à cette structure tout au long de ces travaux en référence à sa capacité de déformation semblable à celle des personnages animés "BarBaPaPa"

Le troisième chapitre propose une méthode garantie pour contrôler la déformation d'un robot à chenilles équipé de n articulations. Les contraintes relatives à ce genre de problème y sont définies, puis une solution utilisant des outils empruntés à l'analyse par intervalles est proposée.

Enfin, la dernière partie sera consacrée à la commande d'un robot mobile de type B2P2 en vue du franchissement autonome d'un escalier. Une validation expérimentale avec notre prototype y est également présentée.

Notons que les deux derniers chapitres peuvent être lus sans considération de l'ordre dans lequel ils apparaissent. Il s'agit de deux pistes que nous avons explorées durant ces trois années couvrant deux aspects de la robotique mobile telle qu'elle est présentée dans le début de ce manuscrit : "la déformation d'un robot à l'intérieur d'une chenille" et "le franchissement autonome avec un robot mobile déformable".

L'ensemble de ces travaux est finalement discuté dans une conclusion générale.

Chapitre 2

Revue sur les robots mobiles terrestres

Sommaire

2.1 Historique

On peut prétendre que la robotique mobile telle qu'on la connaît aujourd'hui est née dans les années cinquante. Le Dr William Grey Walter, pionnier de la cybernétique développait alors ses "tortues" (figure 2.1a)) [Grey 1963], petits engins capables de réagir à la lumière en fonction de son intensité. Au cours des années soixante, les techniques et les composants se développent pour permettre le lancement le 10 novembre 1970 du robot "Lunokhod 1" (figure 2.1b)), premier robot d'exploration lunaire qui restera en état de fonctionnement sur le satellite de la terre une dizaine de jours.

A partir de ce moment, les robots mobiles doivent s'acquitter d'une nouvelle mission : être capable de franchir divers obstacles, aptitude que l'on qualifiera de "capacité de franchissement". C'est dans cette optique qu'au milieu des années 80, la conception de plate-formes mobiles sera revisitée par Iwamoto et Yamamoto [Iwamoto 1983] pour donner naissance à un nouveau type de robot à chenilles capable de changer de forme pour s'adapter à son environnement (figure 2.1c)). Au début des années 90, alors que seuls les robots manipulateurs sont conçus de manière industrielle, quelques entreprises dédiées à la conception de robots mobiles font leur apparition comme IRobot, fondée en 1990, a qui on doit les robots, "Urbie" (figure 2.2a)) et "PackBot" (figure 2.2b)), ou encore Inuktun, fondée en 1989 qui développera le "microVGTV" (figure 2.2c)).

La fin des années 90 est marquée par la création du CRASAR, ou Center for Robot Assisted Search and Rescue, organisme chargé de prévoir et d'analyser le développement des robots mobiles en vue de leur utilisation dans le cadre de missions de recherche de victimes après certaines catastrophes (Search and Rescue).

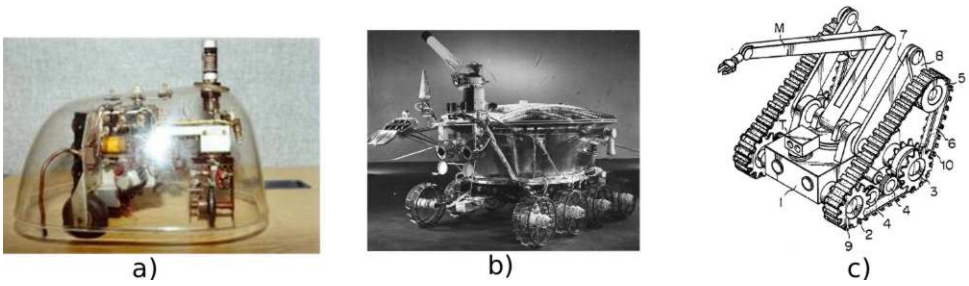


FIG. 2.1 – a) Une "tortue" du Dr Grey Walter. b) Lunokhod 1. c) Extrait du brevet de Iyamoto et Yamamoto.

Depuis une dizaine d'années, les applications liées à la robotique mobile ne cessent d'augmenter, tout comme le nombre d'entreprises dédiées à la construction de robots mobiles. N'oublions pas que la robotique domestique est en passe de devenir un marché très lucratif (en 2009, Roomba, le robot aspirateur était le robot le plus vendu au monde).

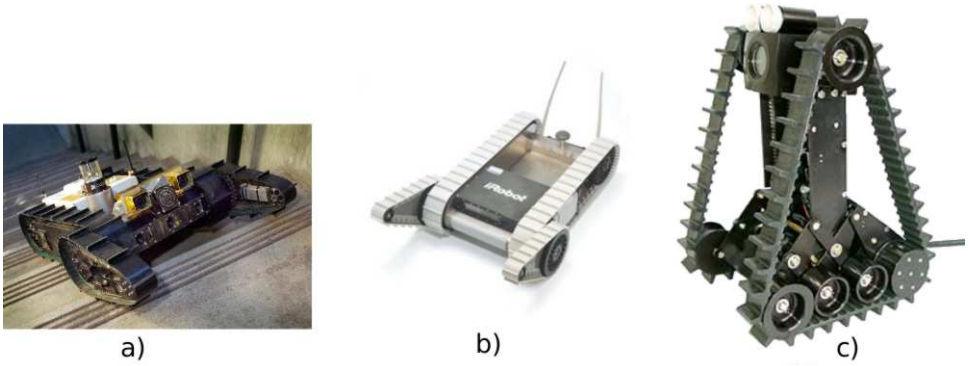


FIG. 2.2 – a) Urbie de IRobot. b) Packbot de IRobot. c) Le MicroVGTV de Inuktum.

2.2 Robotique mobile et franchissement d'obstacles

2.2.1 Les robots sur le terrain

Lorsque l'on parle de robotique mobile, il est souvent question de capacité de franchissement. Les applications pour lesquelles ces plate-formes sont utilisées, aussi bien dans les domaines militaires que de la domotique demandent souvent une grande adaptabilité et une capacité à franchir les obstacles qui l'entourent. Sur le terrain, ces robots qui peuvent être autonomes, semi-autonomes ou complètement télé-opérés doivent la plupart du temps interagir et s'adapter à leur environnement. On s'imagine très bien qu'un robot de reconnaissance militaire comme peut l'être le PackBot (figure 2.3a))) ou encore le Talon (figure 2.3b)) peut être amené à franchir certains types d'obstacles comme des trottoirs ou des escaliers pour permettre

une exploration efficace. On peut aussi citer le MicroVGTV, spécialement conçu pour effectuer des opérations de maintenance dans des tuyaux de petites tailles, qui accuse une capacité de franchissement importante. Bien sûr, la question se pose aussi pour les robots dits domestiques ; un robot aspirateur serait encore plus pratique s'il pouvait aller faire son office à l'étage de lui-même. L'adaptabilité des robots mobiles à leur environnement constitue donc un axe de recherche fondamental pour leur devenir.

En dehors du cadre des opérations militaires, certaines plate-formes ont pu être mises à l'épreuve pour la première fois, lors des évènements survenus au World Trade Centre en Septembre 2001 [Casper 2003]. R. R. Murphy et d'autres membres du CRASAR ont pu ainsi dresser les avantages et les développements à prévoir sur ce type de robots pour les rendre encore plus efficaces.

En outre, certaines applications de reconnaissance nécessitent l'utilisation de plusieurs véhicules (terrestres ou aériens) en même temps. Il est alors question de télé-opérer un maximum d'engins avec un minimum de personnes, des architectures adaptables et semi-autonomes semblent alors être un atout majeur.

Par conséquent, cette utilisation des véhicules comme "explorateurs tout terrains" pousse les concepteurs à innover pour obtenir de plus en plus d'adaptabilité, ce qui explique la diversification des architectures et des prototypes. Il convient donc de s'efforcer de trier les plate-formes existantes de manière à situer les différents concepts et techniques mises en jeu par différents choix de conception.

FIG. 2.3 – a) Packbot de IRobot. b) Version "SWAT" du Talon de Foster-Miller.

2.2.2 Classification

Deux grandes familles de robots terrestres, ou UGV de l'anglais "Unmanned Ground Vehicles", peuvent être identifiées :

- les véhicules non déformables,
- les véhicules déformables, dits à géométrie variable.

Robot	Vitesse Max ($m.s^{-1}$)	Poids (K_g)	Taille LxWxH (cm)	Plus haute marche (cm)	Ratio
ATRV-jr	1	50	77x55x62	30	0.48
Talon Hazmat	1.8	22 to 64	86x57x30	15	0.5
Packbot	1.8	15	88.4x51.5x16.7	30	1.7
Micro VGTV	0.075	1.6	31.7x16.5x6.5	25	3.8
RobuROC6	3.6	160	150x78x50	25	0.5
B2P2	1.5	8	55x37x12	35	2.91

TAB. 2.1 – Spécifications de plusieurs robots associées à leur ratio de franchissement

La capacité de franchissement d'un robot peut être estimée par le rapport entre sa taille et la taille du plus important obstacle franchissable. Ce dimensionnement, que l'on nommera ratio de franchissement, permet de qualifier la capacité de franchissement de robots différents en favorisant les structures capables de s'insérer dans des endroits étroits (inspection de conduite d'eau...) et de franchir des obstacles importants. Pour une comparaison claire, les ratios de franchissement des robots présentés dans cette étude figurent dans le tableau 2.1. Nous allons par la suite passer en revue les différents robots cités dans ce tableau.

2.2.2.1 Véhicules non déformables

Les figures 2.4(a) 2.4(b) présentent deux véhicules non déformables (terme très souvent associé à l'acronyme UGV) équipés de châssis à roues ou à chenilles. Ce genre de construction robuste sera théoriquement limité en terme de franchissement par la taille de ses roues. Ainsi, si D représente le diamètre des roues (ou des poulies pour le cas d'un robot à chenilles), la hauteur de la plus grande marche que pourra franchir le véhicule (en avançant droit sur l'obstacle) est donnée par $\frac{D}{2}$. Ainsi, quelle que soit la taille de l'engin, son ratio de franchissement n'excédera jamais 0.5.

Ce type de robots, bien que présentant une grande fiabilité [Carlson 2003], ne disposent pas d'une capacité d'adaptation suffisante pour mener à bien des missions où le franchissement d'obstacle devient primordial.

(a) Robot ATRV-Jr. (b) robot Talon-Hazmat (Constructeur : Foster-Miller)

FIG. 2.4 – Deux modèles d'UGV, robots nons déformables.

2.2.2.2 Robot à géométrie variable (VGTV)

Pour accroître les capacités de franchissement des UGVs non déformables, une alternative efficace consiste à développer des robots à géométrie variable. Munis plus souvent de chenilles que de roues, ces UGVs peuvent être distingués par l'acronyme anglais VGTV pour "Variable Geometry Tracked Vehicles". Ils sont constitués habituellement d'un châssis principal auquel sont associés des flippers (extensions de la chenille présents notamment sur le Packbot visible sur la figure 2.3a) ou des bras supplémentaires qui permettent au robot de s'accrocher sur le nez d'une marche par exemple.

Citons d'abord un véhicule à roues, le RobuRoc (figure 2.5a)) qui modifie l'inclinaison d'un de ses essieux pour s'adapter à des environnements accidentés (de plus amples informations peuvent être trouvées dans [Agency 2005]). Cela lui permet d'évoluer dans des environnements accidentés ; cependant il ne possède pas un ratio de franchissement élevé (0.5) en raison de sa taille importante.

Dans le cadre de missions de reconnaissance ou de sauvetage, les chenilles seront privilégiés car elles garantissent la transmission de l'effort même en cas de contact ponctuel avec le sol. Le PackBot distribué par la société IRobot est aujourd'hui le robot déformable à chenilles le plus populaire. Ce véhicule d'une conception très simple possède deux flippers (à l'avant) lui offrant une mobilité accrue (notons qu'il existe des versions munies d'une paire de flippers de chaque coté). L'augmentation de cette mobilité est illustré par un ratio de franchissement de 1.7. Dans le domaine de la robotique mobile, ce véhicule constitue une référence maintes fois utilisée dans le cadre d'expérimentations liées au franchissement d'obstacles [Tom Frost 2002].

En plus des robots de type "PackBot" (robots munis de flippers) d'autres prototypes semblent émerger depuis quelques années, à l'image du IRS-Soryu (figure 2.5b)), un robot "serpent" formé de plusieurs modules mobiles les uns par rapport aux autres dont l'efficacité à déjà été démontrée dans [K. Osuka 2006]. Certains robots "serpents" sont capables de déconnecter leur corps permettant une reconnaissance plus rapide d'un environnement donné. Les figures 2.6 and 2.7 présentent cette démarche qui fournit une capacité de franchissement importante au prix d'une commande et d'une mécanique plus complexe et plus lourde qu'un robot type "Packbot"[J. Liu 2005].

2.2.2.3 Robots à géométrie variable munis d'une seule chenille (VGSTV)

Il est possible de différencier certains modèles de VGTV des autres. Il s'agit des robots à géométrie variable munis d'une seule chenille classiquement désignés par l'acronyme VGSTV de l'anglais "Variable Geometry Single Tracked Vehicles". Leur particularité réside dans leur capacité à se déformer à l'intérieur de la chenille. Ces VGSTV se distinguent en deux sous-groupes :

- Robots à chenilles non déformables,
- Robots à chenilles déformables.

FIG. 2.5 – a) : RobuROC 6 (Constructeur : Robosoft) b) : IRS Soryu (robot serpent)

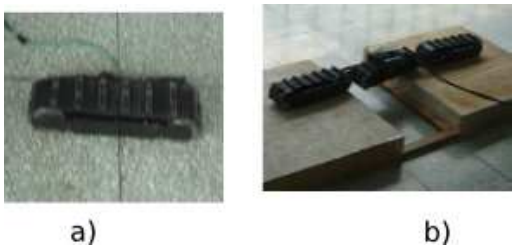


FIG. 2.6 – a) : Système modulaire : un module, b) : Système modulaire : trois modules

a - VGSTV à chenilles non déformables L'intérêt de l'utilisation de robots déformables réside dans leur capacité à s'adapter le plus possible à l'obstacle. L'utilisation d'une seule chenille à la place de flippers permet théoriquement une meilleure adaptabilité du fait des propriétés élastiques des chenilles qui permettent une meilleure transmission de l'effort en cas de contacts ponctuels avec le sol [L. Sung Kyun 2005].

Les premières études concernant ce genre de mécanisme ont été menées dans les années quatre vingts par Iwamoto et Yamamoto [Iwamoto 1983] et ont abouti au développement du MicroVGTV (figure 2.8a)). Il s'agit d'un robot capable d'élever la partie avant de son châssis tout en maintenant la chenille dans les poulies. Un système passif permet de tendre suffisamment la chenille pour une transmission de l'effort des moteurs. Ce robot très connu et très utilisé aujourd'hui, est capable de franchir une marche d'une trentaine de centimètres alors que sa hauteur n'excède pas les douze cm ; il dispose évidemment d'un ratio de franchissement important (3.8).

Une autre méthode pour déformer le robot à l'intérieur d'une chenille est de coupler deux articulations actives comme l'illustre le "VGSTV mechanism" présenté sur la figure 2.8 b). Ce robot peut adopter différentes configurations symétriques (trapèze, carré...) et franchir un escalier.

Bien entendu, plusieurs autres architectures intéressantes peuvent être trouvées dans [G. Clement 1987], [Misawa 1997], [Vincent 2007], et [M. Guarnieri 2004].

FIG. 2.7 – Un système modulaire pendant le franchissement d'un trottoir

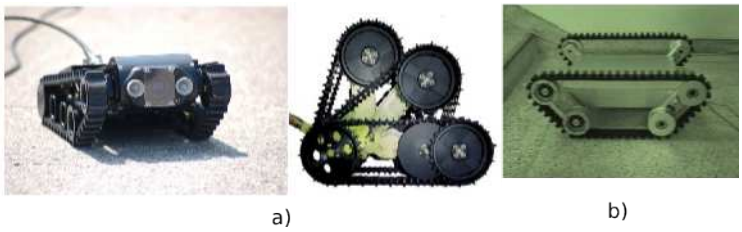


FIG. 2.8 – a) : Micro VGTV (Constructeur : Inuktun Ltd) b) : VGSTV mechanism

b - Robots à chenilles déformables Pour simplifier la commande et augmenter la fiabilité de ces VGSTV, certains véhicules comme le WORMY [Kinugasa 2008] (figure 2.9b)) ou le Viper (figure 2.9a)) développé par la société Galileo possèdent des chenilles déformables qui s'adaptent à la forme du châssis. Il semble alors possible de profiter des avantage des roues et des chenilles avec la même plate-forme.

FIG. 2.9 – a) : Viper robot (Constructeur : Galileo) b) : Chenille mobile : WORMY

2.2.2.4 Robots reconfigurables

Les robots reconfigurables sont le plus souvent des petits modules capables de se connecter les uns aux autres pour prendre des formes très différentes ; quelques exemples populaires sont illustrés par la figure 2.10. Leur utilisation n'est pas encore répandue pour des missions de reconnaissance nécessitant une bonne capacité de franchissement, mais leur évolution depuis ces dernières années laisse présager un avenir pour ce genre de structures capables aujourd'hui de franchir bon nombre

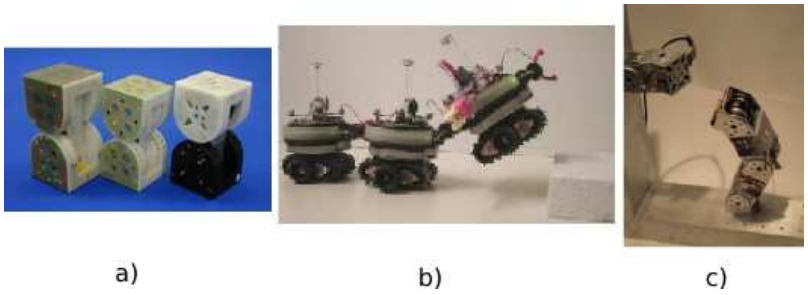


FIG. 2.10 – a) : M-TRAN modules (AIST) b) : SWARM-BOT (EPFL) c) : Polybot (PARC)

d'obstacles [F. Mondada 2004].

2.2.2.5 Conceptions alternatives

Pour finir cette classification, il est important de faire référence à quelques conceptions alternatives qui peuvent présenter un intérêt, mais qui restent très difficiles à classer. On pourra ainsi noter la souplesse d'Helios IV (figure 2.11a)) qui se hisse au dessus des obstacles en utilisant son bras, la rapidité du RHex (figure 2.11b)), un robot hexapode [U. Saranli 2001], capable de franchir sans problème des marches de tailles diverses ou des gravats, et enfin la discrétion du Terminatorbot (figure 2.11c)), sorte de main qui se déplace en utilisant ses doigts [Voyles 2000].

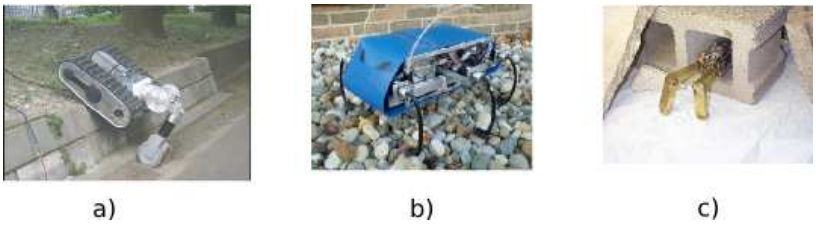


FIG. 2.11 – a) : Helios VII b) : le robot hexapode RHex c) : Terminatorbot

2.3 Conclusion

Depuis ses débuts, la robotique mobile a beaucoup évolué, et constitue encore au regard de la diversité des prototypes existants un domaine en plein essor. Ce chapitre qui avait pour vocation de présenter les principales catégories de robots ainsi que leur utilisation a dressé un classement non-exhaustif qui ne répertorie pas tous les véhicules existants, mais s'efforce de trier les grandes catégories de robots mobiles pour mettre en évidence les avantages et inconvénients de chacun.

Basé sur cette classification, l'équipe robotique du LISA a choisi d'étudier et d'approfondir les travaux sur les VGSTV qui semblent représenter un compromis

efficace entre la fiabilité, la capacité de franchissement et la télé-opération. Effectivement, des robots reconfigurables (figure 2.10) ou des systèmes modulaires (figure 2.7) présentent des avantages certains en terme d'exploration, mais demandent une grande concentration au pilote (il s'agit de gérer 2 ou 3 modules en même temps). En outre, il semble qu'un VGTV classique comme le packbot n'utilise pas au mieux les propriétés de ses chenilles. Les VGSTV représentent donc alors une alternative intéressante alliant une télé-opération aisée à une capacité de franchissement étonnante.

Le robot étudié dans la suite de ces travaux, peut donc être qualifié de VGSTV ; il ne possède en effet une seule chenille sur chacun de ses flancs. Notons que l'architecture de ce dernier apporte des nouveautés aux prototypes existants qui seront expliquées et analysées dans le chapitre suivant.

Chapitre 3

Le prototype B2P2 : modèles et études de stabilité

Sommaire

Dans ce chapitre, il est question de présenter et d'étudier le VGSTV nommé B2P2 développé en 2006 par le LISA dans le cadre d'un concours organisé par la DGA à l'ETAS (Angers). Il s'agissait de répondre à un cahier des charges et de réaliser un robot aux dimensions limitées capable de franchir divers obstacles. Différents prototypes ont ainsi été éprouvés et notés lors d'une journée de démonstration. En 2007, B2P2 a également été présent au concours ELROB qui s'est déroulé en suisse, puis, en 2008 lors de la deuxième édition du concours ETAS.
Dans un premier temps, ce robot à l'architecture originale sera décrit, puis ses performances en matière de capacité de franchissement seront analysées et comparées aux robots existants. Ensuite, les modèles géométriques et dynamiques seront explicités de manière à conduire une étude complète concernant la stabilité de l'engin lors du franchissement d'un obstacle.

FIG. 3.1 – Le prototype B2P2

3.1 Le Robot B2P2

3.1.1 Description matérielle

3.1.1.1 Description mécanique

B2P2 est un robot mobile à chenilles télé-opéré et propulsé par deux moteurs à courant continu couplés à un étage de réduction. La figure 3.1 présente le robot, composé de six poulies (trois de chaque coté), d'une partie fixe que l'on nommera par la suite le châssis, et d'une partie mobile désignée comme "partie avant" sur la figure 3.2. Ce prototype présente des roues de 12 cm de diamètre, un poids de 8 Kg pour 45 cm de largeur et 80cm de longueur.
En plus des moteurs destinés à la propulsion, B2P2 dispose de deux moteurs à courant continu supplémentaires dédiés à la mobilité de la partie avant. Comme le présente la figure 3.2, un moteur pilote la montée/descente de la partie avant, tandis qu'un second moteur actionne une vis mère pour modifier la distance entre le second et le troisième essieu. L'utilisation conjointe de ces deux moteurs pour garantir la tension de la chenille est expliquée en section 3.1.1.3.

Enfin, les compartiments blancs sur les flancs du véhicule, renferment 4 (2 de chaque côté) batteries polymères 12 Volts 3200 mAh qui pourvoient un peu plus d'une heure d'autonomie à ses 4 moteurs et à l'informatique de contrôle.

3.1.1.2 Infomatique embarquée

a - Commande L'architecture informatique s'articule autour d'un PC 104 équipé d'un processeur AMD Geode 500 MHz et d'un disque dur compact flash de 4 Go. Le bus de donnée PC104 qui fournit un grand nombre d'entrée/sorties permet aussi de connecter un grand nombre de cartes en cascade. Ainsi, on retrouvera aussi dans le prototype une carte d'extension fournissant 8 ports séries additionnels, une interface ISA/I2C pour communiquer avec d'éventuels capteurs puis une carte

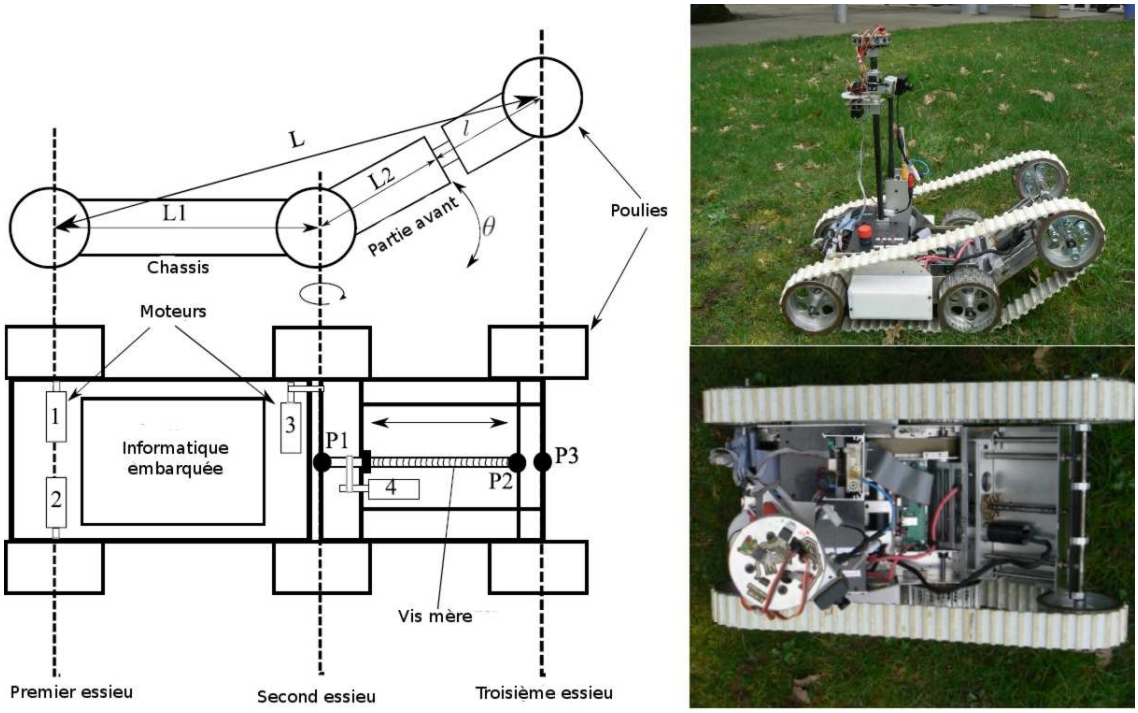


FIG. 3.2 – Décomposition des éléments présents dans le prototype

de gestion d'alimentation. Les moteurs sont actionnés par des cartes de puissance intégrées et pilotées par commande AT via une liaison série RS232.

b - Capteurs Une caméra analogique située sur le mat et montée sur deux servomoteurs permet à l'opérateur de visionner l'environnement dans lequel le robot évolue. La liaison sans fil est garantie par un transmetteur vidéo 2,4 GHz, tandis que du coté PC de commande, un récepteur HF et un convertisseur analogique USB se chargent de récupérer le flux vidéo.
Des capteurs de distance infrarouges SHARP montés sur un servomoteur renvoient la distance de l'obstacle le plus proche, et un inclinomètre 2 axes fixé sur le châssis informe sur le roulis et le tangage du robot.

c - Liaison sans fil et télé-opération Un couple d'émetteur/récepteur HF 380 MHz se charge de transmettre les commandes de l'opérateur obtenues via l'acquisition d'un joystick USB (fig 3.3). Lors de la conception du robot, plusieurs protocoles et matériels ont été testés (WiFi, Emeteur Radiometrix etc...) ; un bilan de ces tests est disponible dans [J. L. Paillat 2008b].

d - Logiciel Pour disposer d'un système fiable, temps réel démarrant rapidement, et sans limitations matérielles (couche réseau, Bus PCI, ports USB etc...),

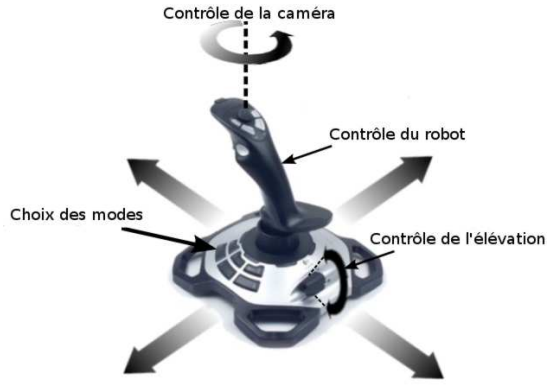


FIG. 3.3 – Contrôle du robot

B2P2 a été équipé d'une distribution Linux From Scratch associée à un noyau 2.6 patché temps réel par l'API Xenomai. Rappelons qu'une distribution Linux From Scratch est une distribution compilée de toute pièce par l'utilisateur en suivant les démarches explicités dans [Beekmans 2007]. On obtient alors le système le plus léger possible. L'OS ainsi conçu démarre en une trentaine de secondes sur le PC104 utilisé.

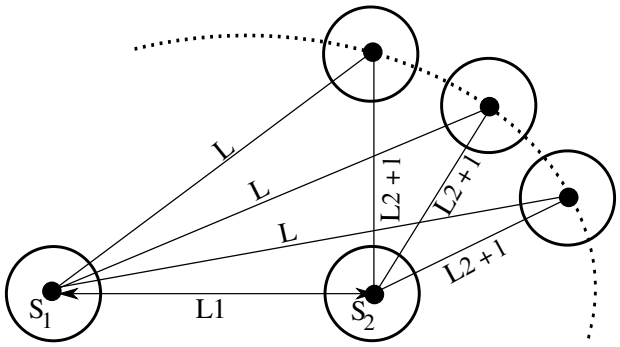


FIG. 3.4 – Une ellipse centrée sur les deux premiers essieux

3.1.1.3 La commande des moteurs : Tension de la chenille

Rappelons que B2P2 est équipé de 4 moteurs, 2 pour la propulsion, et 2 autres pour le système de déformation. Ainsi, contrairement aux VGSTVs classiques où la tension des chenilles est assurée par un système mécanique passif, B2P2 est équipé d'un système actif pour maintenir les chenilles dans les poulies [J. L. Paillat 2008a]. Rappelons qu'une élévation de la partie avant du robot (changement de l'angle θ, figure 3.2) modifie la taille de l'enveloppe convexe formée par les trois roues du robot. Ainsi, les chenilles peuvent ne plus être en contact avec les trois poulies,

rendant la transmission de l'effort des moteurs de propulsion difficile.

Pour maintenir ce contact entre les poulies et la chenille, il est nécessaire de maintenir la taille de l'enveloppe convexe constante durant l'élévation de la partie avant. Cela est facilement réalisé en modifiant la position de la 3^{me} roue en fonction de l'angle θ en garantissant l'équation suivante :

$$L_1 + (L_2 + l) + L + D = K \tag{3.1}$$

où L_1, $L_2 + l$ et L sont référencés sur les figures 3.2 et 3.4, K représente la longueur nominale de la chenille et D est égal au périmètre d'une roue. Notons que L_2 constitue la longueur minimale entre le deuxième et le troisième essieu, et que l représente l'offset dû au mouvement du moteur (de 0cm à 20cm). Cette équation décrit la trajectoire d'une ellipse définie par deux foyers (S_1 et S_2) confondus avec les deux centres des premières roues du robot (figure 3.4). Rappelons que la position de la troisième roue (e. g. l'élongation l) peut être modifiée grâce à un moteur couplé à une vis mère (cf section 3.1.1.1). Ainsi, à partir de 3.1, l peut être exprimée en fonction de θ :

$$l = f(\theta) = \frac{L_1^2 - K^2}{2(L_1 \cos \pi - \mid \theta \mid - K)} - L_2 \tag{3.2}$$

Les commandes reçues de l'opérateur notées $C\theta$ (commande de rotation) et Cl (commande d'élongation), sont soumises à la boucle d'asservissement présentée en figure 3.5. En raison de plusieurs contraintes mécaniques, une saturation de la commande θ est nécessaire pour garantir la dynamique du mouvement. Enfin, un commutateur logiciel noté S est mis en place pour offrir deux modes de fonctionnement :

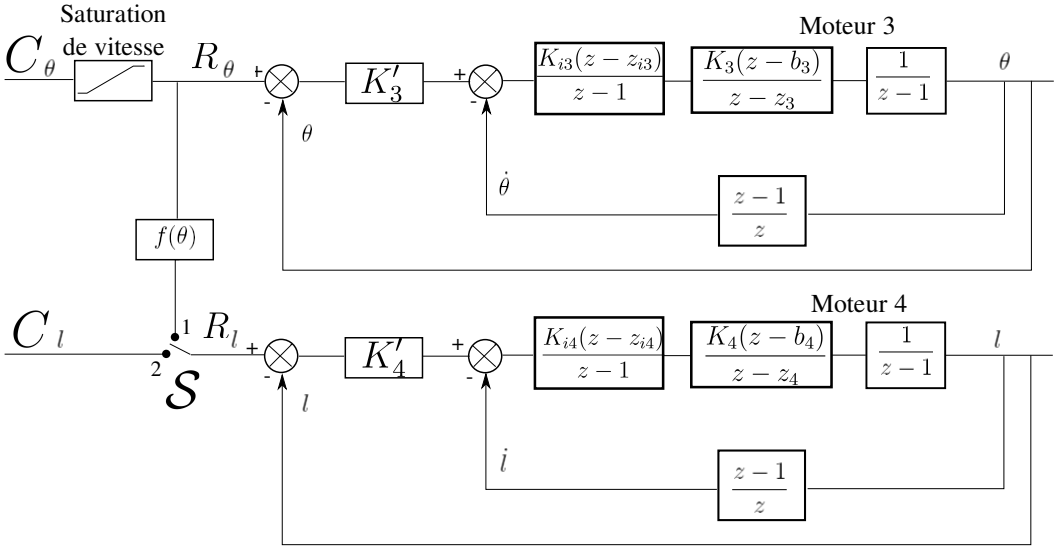


FIG. 3.5 – Asservissement des moteurs chargés de la déformation du robot.

– L'opérateur choisit une valeur pour θ, puis la commande d'élongation l est définie à partir de 3.2 (notée $f(\theta)$ sur la figure 3.5) (mode tendu),
– L'opérateur choisit indépendamment les valeurs des paramètres θ et l, (mode détendu).

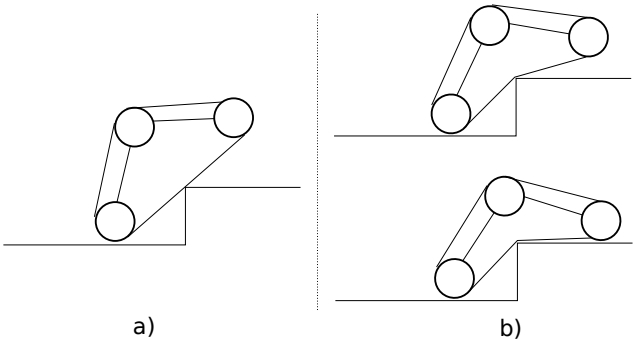


FIG. 3.6 – a) Mode tendu : une seule configuration possible. b) Mode détendu : une souplesse réglable.

Ainsi, l'ajout d'une articulation active à la place d'un couplage mécanique offre certaines capacités supplémentaires schématisées par la figure 3.6. En effet, dans le cas du franchissement d'un trottoir par exemple, le mode tendu n'offre qu'une seule configuration possible (figure 3.6a)), alors que le mode détendu permet de jouer sur la tension de la chenille pour obtenir différentes positions (figure 3.6b)). Cet aspect novateur du prototype est analysé et comparé à un VGTV classique en section 3.1.2.2.

3.1.2 Performances et modes de fonctionnement

3.1.2.1 Performances

Cette section présente les performances de B2P2 lors du franchissement d'obstacles (trottoir, bumper et escalier). Elles ont été collectées lors de manifestations nationales et internationales (ETAS 07, ELROB 07, ETAS 08). L'attrait du système de tension active y est ainsi démontré puis comparé à un système classique n'offrant qu'un mode de fonctionnement équivalent au mode tendu.

Il est important de rappeler que le mode détendu n'intervient que lors de certaines phases du franchissement et est commandé directement par l'opérateur. Dans tout les cas, lorsque l'opérateur demande un relâchement des chenilles, il faudra toujours s'assurer que l'obstacle fournit suffisamment de contacts pour empêcher une perte de la chenille.

a - Trottoir Le franchissement d'un trottoir est composé de trois phases représentées sur le figure 3.7, l'approche de la marche, la montée, et le franchissement. Lors des deux premières phases, B2P2 se pilote comme n'importe quel autre VGTV, il suffit de s'approcher de la marche en élevant la partie avant suffisamment pour

accrocher le nez du trottoir. Ensuite, le robot monte naturellement l'obstacle grâce notamment aux propriétés inhérentes à ses chenilles.
La plupart du temps, la position alors obtenue représentée par la figure 3.7 b) comporte un risque de voir le véhicule basculer. En effet, il est nécessaire d'abaisser la partie avant pour augmenter la surface de contact entre la marche et les chenilles ce qui augmente le risque de déséquilibre (figure 3.7 c)). La réduction de la tension de la chenille obtenue grâce au couplage des moteurs permet alors au châssis d'épouser parfaitement la forme de l'obstacle, augmentant de manière importante la surface de contact (figure 3.7 d)) ; en outre, cela permet aussi d'abaisser le centre de gravité. Dans le cas du trottoir, ce deuxième mode de fonctionnement permet donc un franchissement plus rapide, plus facile (pour l'opérateur), et plus souple qu'en conservant la chenille tendue pendant tout le franchissement.

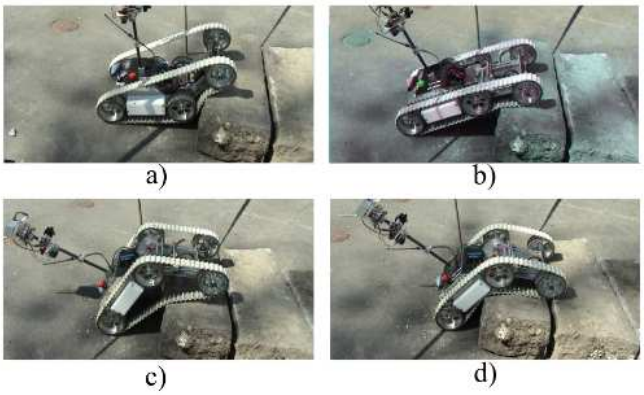


FIG. 3.7 – B2P2 : franchissement d'un trottoir

b - Escalier Classiquement, le franchissement d'escaliers se fait de manière assez naturelle. Une fois le nez de la première marche franchi (figure 3.8 b)), il est indispensable de rabaisser la partie avant de manière à contre balancer le poids du véhicule vers l'avant. Ensuite, en avançant doucement, le robot monte une à une les marches en opérant un transfert de masse à chaque fois. Pour des marches relativement hautes, cette opération nécessite une grande connaissance du robot par l'opérateur qui élèvera, ou baissera la partie mobile pour modifier la position du centre de gravité et garantir ainsi la montée de l'escalier. Enfin, lors du franchissement de la dernière marche, la procédure est la même que pour franchir un trottoir, en détendant les chenilles, le robot obtient une surface de contact suffisamment importante pour arriver en haut de l'escalier d'une manière très souple et fluide.

c - Bumper Le bumper (un demi cylindre en tôle) est sans aucun doute le meilleur exemple de l'intérêt d'un robot muni d'une seule chenille (figure 3.9). En

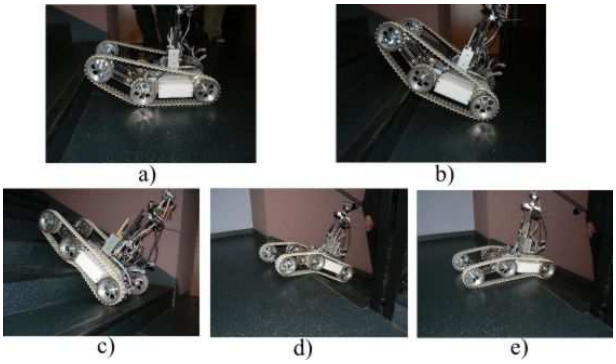


FIG. 3.8 – B2P2 : franchissement d'un escalier

effet, il est très difficile d'épouser parfaitement ce genre d'obstacle avec un robot muni de flippers (type Packbot). Des plate-formes à chenilles comme B2P2 ou le MicroVGTV vont quant à elles se révéler efficaces et offrir une adhérence supérieure. Là encore, la possibilité de détendre les chenilles offre à B2P2 la possibilité d'épouser parfaitement l'obstacle pour un franchissement plus aisé (figure 3.9b)). Des vidéos

FIG. 3.9 – B2P2 : franchissement d'un bumper

de démonstration présentant le franchissement des obstacles en mode détendu et/ou tendu sont hébergées à l'adresse `http://www.istia.univ-angers.fr/LISA/B2P2/b2p2.html`.

3.1.2.2 De l'intérêt du mode détendu

Dans la mesure où le choix de la tension des chenilles constitue l'attrait majeur de la plate-forme étudiée, les opportunités qu'elle offre se doivent d'être analysées. Notons que lorsque B2P2 fonctionne en mode tendu (la commande d'élongation l est automatiquement calculée en fonction de θ) il se comporte exactement comme un VGSTV classique du type de MicroVGTV.
Les expérimentations menées avec ce véhicule décrites en section 3.1.2.1, démontrent que le transfert de masse semble plus souple ; ceci offre un pilotage plus aisé. Rappelons que le transfert de masse représente le mouvement effectué par le

robot lorsque son centre de gravité passe au delà des points de contacts sol/robot ; typiquement, lorsque B2P2 bascule en équilibre sur le sommet d'une marche afin que la 3^{me} roue se pose sur la marche suivante. A ce moment, le pilote n'a aucun moyen de contrôler le robot, il convient donc que cette phase du franchissement soit la moins brusque possible.
Pour quantifier les avantages offerts par cette plate-forme, une expérience simple consistant à analyser le franchissement de la dernière marche d'un escalier en considérant l'un après l'autre les modes tendus et détendus (à vitesse égale) a été conduite (figure 3.10 et 3.11). Notons que la première différence entre ces deux manipulations réside dans les points de contacts sol/robot. En effet, en mode tendu (figure 3.10) lors du transfert de masse (entre figure c et d) il n'y a qu'un point de contact entre la chenille et le sol (en rouge), ce qui rend l'opération très délicate. Le mode détendu (figure 3.11) quant à lui, permet de garder au moins deux points de contact durant le franchissement, ce qui évite de mettre le robot dans une position d'équilibre précaire et dangereux.

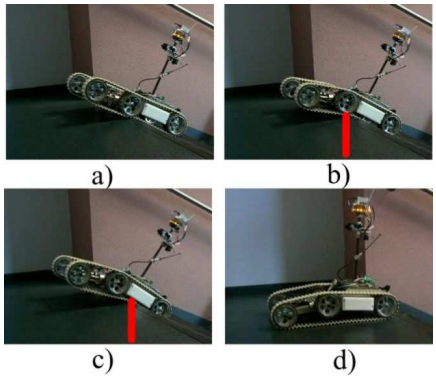


FIG. 3.10 – Chenilles tendues (Mode de fonctionnement proche de celui d'un Micro VGTV)

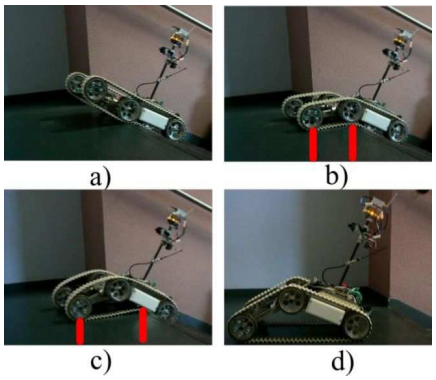


FIG. 3.11 – Chenilles détendues (Mode de fonctionnement innovant)

Lors de ces manipulations, l'inclinaison du châssis a été mesurée de manière à quantifier les mouvements lors du franchissement. Les courbes de la figure 3.12 présentent les résultats. La première chose remarquable réside dans la limite supérieure de l'inclinaison du châssis. Le mode détendu réduit cette limite qui est directement liée à la stabilité du robot (plus l'angle entre le châssis et le sol est important, plus le risque de voir le robot se renverser augmente). En plus de ce gain de stabilité, les mesures ont mis en évidence la souplesse du transfert de masse. Les vitesses angulaires ont été calculées pour la durée du transfert de masse (zone répertoriée MT sur le figure 3.12) et sont présentées sur le tableau 3.1. L'accélération et la vitesse angulaire sont considérablement réduites par l'utilisation du mode détendu.

Le relâchement de la tension des chenilles a donc un effet important sur le

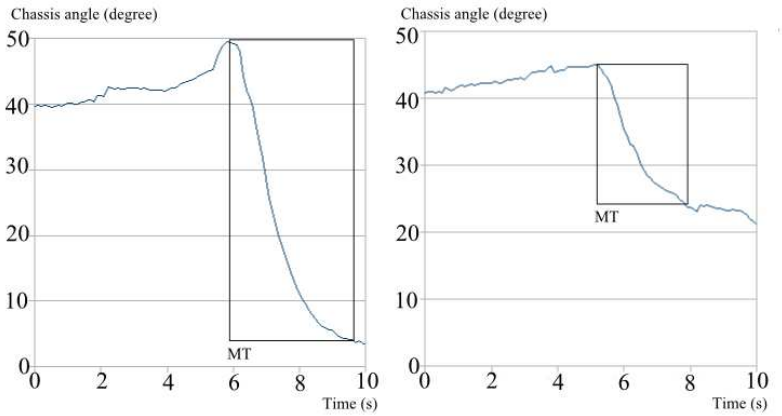


FIG. 3.12 – Comparaison des deux modes de fonctionnement. L'évolution de l'angle du châssis lors du transfert de masse est représentée à gauche pour le mode tendu, et à droite pour le mode détendu.

	mode tendu	mode détendu
Accélération angulaire moyenne	0.2 $deg.s^{-2}$	0.02 $deg.s^{-2}$
Vitesse angulaire moyenne	16.2 $deg.s^{-1}$	6.96 $deg.s^{-1}$
Vitesse angulaire maximale	34.94 $deg.s^{-1}$	12.54 $deg.s^{-1}$

TAB. 3.1 – Comparaison des deux modes de fonctionnement.

comportement du robot, sur sa stabilité et facilite l'opération pour le pilote. On imagine alors qu'il devient possible de libérer l'opérateur de certaines tâches pour optimiser le nombre de véhicules (terrestres, aériens, sous-marins..) qu'il peut piloter simultanément.

3.2 Modèles géométriques et dynamiques

3.2.1 Modèle géométrique

La modélisation de la plate forme est indispensable si l'on désire contrôler la stabilité du robot. Le modèle géométrique direct permet d'obtenir les coordonnées x, y et z d'un élément du robot (centre de gravité, organe terminal) en fonction des coordonnées articulaires $\mathbf{q}$.

Classiquement, un robot de type bras manipulateur par exemple, est représenté par une succession de solides mobiles les uns par rapport aux autres. Ces solides sont liés par des articulations prismatiques ou rotoïdes. A chaque articulation j correspond une valeur articulaire notée q_j ; ainsi, le vecteur $\mathbf{q}$ dit vecteur de

coordonnées articulaires permet de repérer la position ainsi que la configuration d'un robot dans l'espace dit de coordonnées articulaires. Un exemple simple de la modélisation d'un robot manipulateur SCARA est présenté figure 3.13

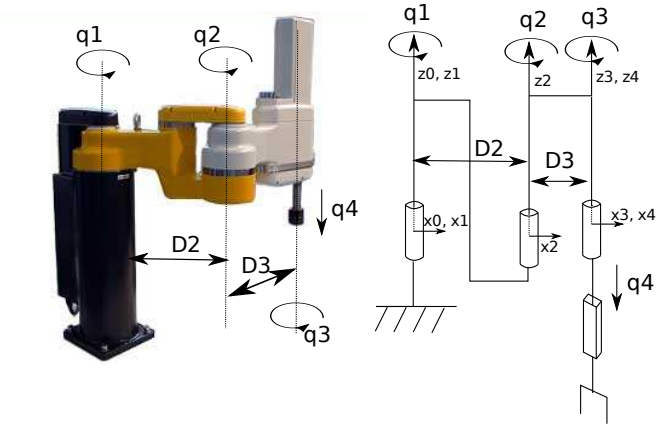


FIG. 3.13 – Modèle géométrique d'un robot manipulateur

Le modèle géométrique va illustrer les relations qui existent entre la position d'un robot dans l'espace de coordonnées articulaires et la position d'un de ses éléments (l'organe terminal, le Centre de Gravité...) dans l'espace défini par un repère orthonormé x, y, z. Plusieurs méthodes existent pour mettre en oeuvre ces relations, la plus populaire étant sans doute la méthode de Denavitt-Hartenberg [Craig 1989]. Cette méthode décrite en Annexe A fournit plusieurs matrices de transformation entre les différents solides telles que :

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix}^T = R_{0j} \begin{bmatrix} x_j & y_j & z_j & 1 \end{bmatrix}^T \tag{3.3}$$

où R_{0j} permet d'identifier les coordonnés x, y, z d'un point dans le repère R_0. Notons que coordonnées x_j, y_j, z_j de ce point dans le repère R_j sont connues. L'orientation des repères $O_j, x_j, y_j z_j$ attachés aux articulations déterminent les paramètres des matrices.

Dans le cas de B2P2, l'approche consiste à le représenter comme un robot manipulateur (figure 3.14). Sa position dans l'espace sera évaluée par trois articulations prismatiques (q_1, q_2 et q_3), le roulis, le lacet et le tangage seront représentés par trois articulations rotoïdes (q_4, q_5 et q_6), puis, un couple d'articulation rotoïde et prismatique symbolisera le couplage actif décrit en section 3.1.1.3. Ainsi, le vecteur de coordonnées articulaires est donné par :

$$\mathbf{q} = [q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8]^T \tag{3.4}$$

Notons que le robot sera aussi décomposé en 3 solides, notés S_i, mobiles les uns par rapport aux autres. S_1 (le châssis) se retrouve ainsi associé à l'articulation

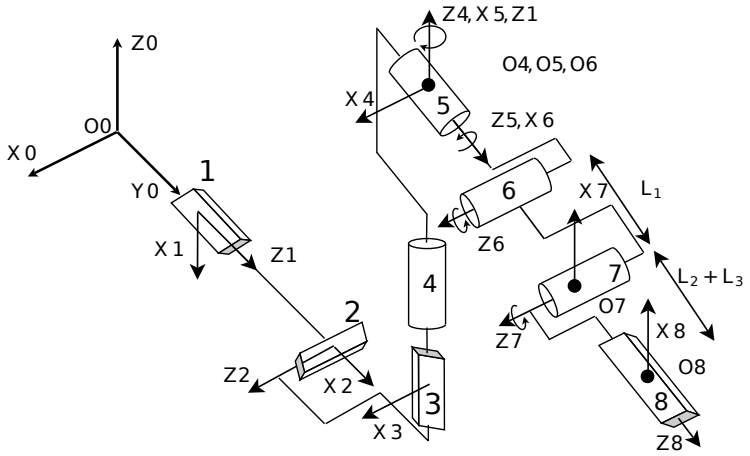


FIG. 3.14 – Modèle géometrique du robot B2P2.

6, puis la partie avant est décomposée en deux solides, S_2 et S_3, respectivement associés aux articulations 7 et 8 de la figure 3.14.

Cependant, pour plus de lisibilité, q_1, q_2 et q_3 seront notés x, y et z, q_4, q_5 et q_6 deviendront respectivement α, γ et β, et finalement, q_7 et q_8 seront notés θ et l. Le modèle géométrique ainsi écrit est représenté par la figure 3.15. Le vecteur $\vec{q}$ s'écrira :

$$\mathbf{q} = [x, y, z, \alpha, \gamma, \beta, \theta, l]^T \tag{3.5}$$

A partir de cette représentation, les paramètres de Denavitt-Hartenberg sont listés en tableau 3.2 dans lequel L_i désigne la longueur du solide i.

j	σ_j	α_j	d_j	θ_j	r_j
1	1	$\frac{-\pi}{2}$	0	$\frac{\pi}{2}$	x
2	1	$\frac{\pi}{2}$	0	$\frac{\pi}{2}$	y
3	1	$\frac{-\pi}{2}$	0	$\frac{-\pi}{2}$	z
4	0	0	0	α	0
5	0	$\frac{-\pi}{2}$	0	$\gamma - \frac{\pi}{2}$	0
6	0	$\frac{-\pi}{2}$	0	$\beta - \frac{\pi}{2}$	0
7	0	0	L_1	$\theta + \frac{\pi}{2}$	0
8	1	$\frac{\pi}{2}$	0	0	$L_2 + l$

TAB. 3.2 – Paramètres de Denavitt-Hartenberg

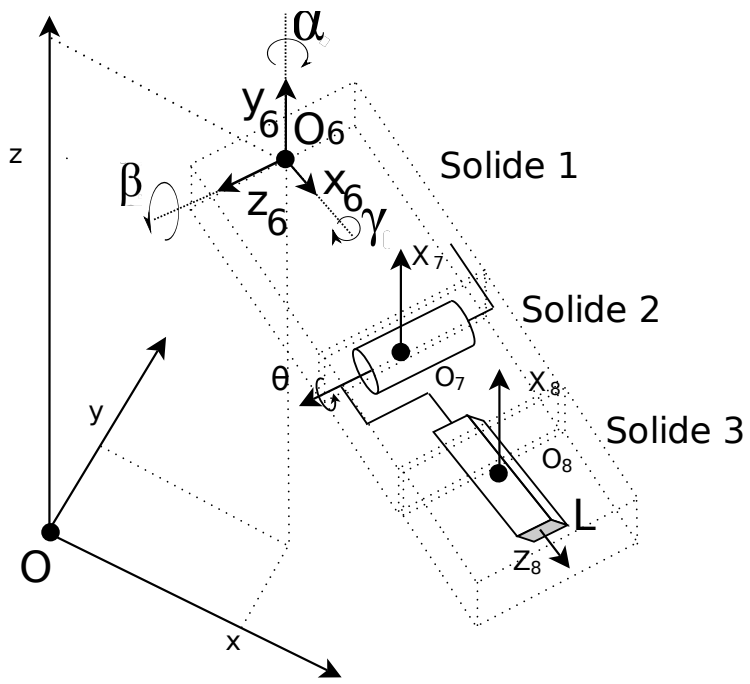


FIG. 3.15 – Modèle géometrique final du robot B2P2.

3.2.2 Modèle dynamique

Pour étudier le comportement du robot lors des phases de franchissement, il est nécessaire d'étudier les équations dynamiques du système, c'est à dire le mouvement d'ensemble du robot lors de sa déformation (mouvement brusque d'un des paramètres du vecteur $\mathbf{q}$). Ces équations sont souvent exprimées à partir du formalisme de Lagrange [Khalil 2004] qui prend en compte les couples et vitesses de chaque moteur, l'effet de la gravité et d'autres effets non linéaires tels que la force de Coriolis.

Les équations de Lagrange donnent la dynamique du système directement dans l'espace articulaire $\mathbf{q}$ sous la forme vectorielle suivante :

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_j} - \frac{\partial L}{\partial q_j} = \Gamma_j \tag{3.6}$$

avec :
$\left\{ \begin{array}{l} L \text{ le Lagrangien du système,} \\ q_j, \text{ la } j^{ème} \text{ articulation du système,} \\ T_j \text{ le couple dû aux forces extérieures appliqué à la } j^{ème} \text{ articulation.} \end{array} \right.$

Le Lagrangien L est défini par la différence entre l'énergie cinétique du système et l'énergie potentielle des forces conservatrices, ainsi :

$$L(\mathbf{q}, \dot{\mathbf{q}}) = K(\mathbf{q}, \dot{\mathbf{q}}) - E(\mathbf{q}) \tag{3.7}$$

où :

$$\begin{cases} \mathbf{q}, \dot{\mathbf{q}} \text{ : respectivement les positions et vitesses des articulations,} \\ K(\mathbf{q}, \dot{\mathbf{q}}), \text{ : l'énergie cinétique,} \\ E(\mathbf{q}, \dot{\mathbf{q}}) \text{ : l'énergie potentielle.} \end{cases}$$

L'energie cinétique s'écrit en fonction des masses et vitesses des éléments (corps) du robot :

$$K = \sum_{i=1}^{n} \frac{1}{2} m_i v_i^T v_i + \frac{1}{2} w_i^T I_i w_i. \tag{3.8}$$

$$avec : \begin{cases} m_i \text{ la masse du } i^{ème} \text{ corps,} \\ v_i \text{ la vitesse linéaire du centre de gravité du } i^{ème} \text{ corps,} \\ w_i \text{ la vitesse angulaire du centre de gravité du } i^{ème} \text{ corps,} \\ I_i \text{ est le tenseur d'inertie du } i^{ème} \text{ corps.} \end{cases}$$

Bien entendu, pour avoir des équations homogènes, w_i est définie dans le même repère que le tenseur d'inertie I_i. Les vitesses angulaires et linéaires de chaque corps peuvent aussi être mises sous la forme d'une fonction dépendant de $\mathbf{q}$:

$$v_i = J_{v_i}(\mathbf{q})\dot{\mathbf{q}} \tag{3.9}$$

$$w_i = R_{0j}^T J_{w_i}(\mathbf{q})\dot{\mathbf{q}} \tag{3.10}$$

J_{v_i} and J_{w_i} définissent deux matrices qui lient la vitesse d'un élément du robot au vecteur de vitesse articulaire $\mathbf{q}$. R_{0j} représente la matrice de transport entre le repère R_0 et le repère j attaché au corps i. Ces repères sont définis par le modèle géométrique illustré en figure 3.14.

A partir de 3.8, 3.9 et 3.10, l'énergie cinétique peut être mise sous forme quadratique en fonction de $\mathbf{q}$

$$K = \frac{1}{2}\dot{\mathbf{q}}^T \sum_i [m_i J_{v_i}(\mathbf{q})^T J_{v_i}(\mathbf{q}) + J_{w_i}^T(\mathbf{q}) R_{0j} I_i R_{0j}^T J_{w_i}(\mathbf{q})]\dot{\mathbf{q}} \tag{3.11}$$

qui peut finalement s'écrire :

$$K = \frac{1}{2}\dot{\mathbf{q}}^T D(\mathbf{q})\dot{\mathbf{q}}. \tag{3.12}$$

Ensuite, l'énergie potentielle s'écrit :

$$E(\mathbf{q}) = \sum_i g m_i G_{zi}^0 \tag{3.13}$$

$$avec : \begin{cases} G_{zi}^0 \text{ est la coordonnée } z \text{ du centre de gravité du corps } i \\ \text{ prise dans le repère de base } R_0 \text{ (cf figure 3.15),} \\ g \text{ est l' accélération de la gravité.} \end{cases}$$

D'après 3.7 et 3.6, les équations du mouvement en fonction des énergies cinétique et potentielle deviennent :

$$\frac{d}{dt}(\frac{\partial K}{\partial \dot{q}_j}) - \frac{\partial K}{\partial q_j} + \frac{\partial E}{\partial q_j} = \Gamma_j \tag{3.14}$$

Ainsi, le modèle dynamique s'écrira :

$$\sum_m d_{jm}(\mathbf{q})\ddot{q}_m + \sum_{n,m} c_{nmj}(\mathbf{q})\dot{q}_n\dot{q}_m + G(\mathbf{q}) = \Gamma_j \tag{3.15}$$

$$c_{nmj} = \frac{1}{2}[\frac{\partial d_{jm}}{\partial q_n} + \frac{\partial d_{jn}}{\partial q_m} - \frac{\partial d_{nm}}{\partial q_j}] \tag{3.16}$$

ou bien, sous une forme classique :

$$D(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}) = \Gamma \tag{3.17}$$

$D(\mathbf{q})$ issue de 3.12 est appelée matrice d'inertie du système et est dépendante des tenseurs d'inertie et des positions articulaires, tandis que $G(\mathbf{q})$ dépend directement de l'énergie potentielle evaluée en 3.13 :

$$G(\mathbf{q}) = \frac{\partial E(\mathbf{q})}{\partial \mathbf{q}} = \sum_i g m_i \frac{\partial G^0_{zi}}{\partial q_j}. \tag{3.18}$$

Enfin, $C(\mathbf{q}, \dot{\mathbf{q}})$ dite matrice centrifuge-coriolis est construite telle que :

$$X_{jm} = \sum_n c_{nmj}\dot{q}_n.$$

3.2.3 Moment cinétique

Le moment cinétique est aussi un point important de la dynamique d'un robot. En effet, le moment cinétique joue un rôle analogue à la quantité de mouvement dans le cas d'un solide en rotation. Il est indispensable en cas de contrôle dynamique du centre de masse qui sera étudié en section 3.3.2.

3.2.3.1 Moment cinétique d'un solide

Classiquement, le moment cinétique autour d'un point fixe A est l'intégrale pour l'ensemble des points P d'un système de $AP \wedge V(P)$, où $V(P)$ désigne le vecteur vitesse du point P. Ainsi, le moment cinétique du point A lié au solide S défini dans le repère R est donné par :

$$\sigma_A(S/R) = \int_{P \in S} AP \wedge V(P/R) dm \tag{3.19}$$

Dans [Djoudi 2007], σ est redéfini comme :

$$\sigma_A(S/R) = mAG \wedge V(A/R) + I_A(S)\Omega(S/R) \tag{3.20}$$

Où $\Omega(S/R)$ représente le vecteur de rotation du solide S autour du repère R, et G est le centre de gravité du solide.

3.2.3.2 Moment cinétique d'un système multicorps

Par définition, le moment cinétique total d'un système multicorps (composé de plusieurs corps rigides S_i) est égal à la somme vectorielle des moments cinétiques qui le constituent. Ainsi, pour un point P n'appartenant pas au système :

$$\sigma_P = \sum_{i=1}^{n} \sigma_{iP} \tag{3.21}$$

où n est le nombre de corps, et σ_{iP}, le moment cinétique du corps i calculé au point P.
La relation de transport du moment cinétique entre deux points A et B appartenant au solide S donne :

$$\sigma_B(S/R) = \sigma_A(S/R) + mV(G/R) \wedge AB \tag{3.22}$$

Ainsi, à partir des équations 3.26 et 3.21 le moment cinétique d'un point P pour un système S composé de n corps rigides est tel que :

$$\sigma_P(S/R) = \sum_{i=1}^{n} I_{Gi}\Omega_i + \sum_{i=1}^{n} miV_{Gi} \wedge G_iP \tag{3.23}$$

3.3 Etude de la stabilité de la plateforme

Les expérimentations menées sur le prototype B2P2 ont souligné l'importance du contrôle de la stabilité du robot. Ce contrôle nécéssite de quantifier cette stabilité à travers un critère. Dans la littérature, deux critères sont souvent utilisés :

- un critère statique : la position du centre de Gravité (CoG) dans le polygone de sustentation,
- un critère dynamique : la position du point de moment zéro (ZMP) dans le polygone de sustentation.

Notons que le polygone de sustentation est formé par la surface de contact entre un objet et le sol. Avant de mettre en place une stratégie de commande, il semble judicieux de choisir le critère adéquat. L'étude qui suit propose donc de présenter ces deux critères, puis de les comparer lors du franchissement d'un escalier par B2P2 afin de quantifier les effets de la dynamique sur le prototype.

3.3.1 Centre de Gravité (CoG)

Le centre de gravité d'un solide est le point d'application de la résultante des forces de gravité, il sera assimilé au barycentre des masses si l'on considère que le champ gravitationnel est le même pour tout le corps. D'un point de vue pratique, les logiciels de CAO actuels sont capables d'évaluer sa position en fonction du poids de tous les éléments qui composent le corps.

Ainsi, pour un système multicorps, le centre de gravité G de l'ensemble est donné par le barycentre des centres de gravité de chaque solide G_i :

$$G = \frac{\sum_{i=1}^{n} m_i Gi}{\sum_{i=1}^{n} m_i} \tag{3.24}$$

L'équilibre d'un objet est directement lié à la position du centre de gravité. Si le projeté au sol du centre de gravité se trouve en dehors du polygone de sustentation, l'objet bascule autour de son point d'appui (figure 3.16).

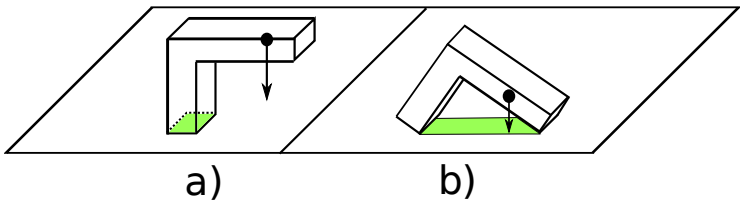


FIG. 3.16 – a) objet en déséquilibre : projeté du centre de gravité en dehors du polygone de sustentation (zone verte). b) objet en équilibre : projeté du centre de gravité dans le polygone de sustentation.

Dans notre cas, considérant le modèle géométrique décrit en section 3.2.1, le centre de gravité G de la plate-forme est donné par :

$$\begin{bmatrix} x_g \\ y_g \\ z_g \\ 1 \end{bmatrix}_{R_0} = \frac{m_1 T_6 \begin{bmatrix} z_1 \\ x_1 \\ y_1 \\ 1 \end{bmatrix}_{R_6} + m_2 T_7 \begin{bmatrix} z_2 \\ -y_2 \\ x_2 \\ 1 \end{bmatrix}_{R_7} + m_3 T_8 \begin{bmatrix} -y_3 \\ z_3 \\ x_3 \\ 1 \end{bmatrix}_{R_8}}{m_1 + m_2 + m_3} \tag{3.25}$$

où T_j correspond à la matrice de transport qui lie le repère j au repère 0 (les repères utilisés ici sont directement liés à ceux présentés dans le modèle géométrique).

3.3.2 ZMP

L'équilibre dynamique est très souvent étudié dans le domaine de la robotique humanoïde. En effet, dans le cadre du contrôle de la marche de ces robots munis d'une vingtaine de degrés de liberté, et d'autant de solides en mouvement, il est pertinent d'assurer l'équilibre en considérant la dynamique du système. Le point largement utilisé dans ce genre de contrôle est le ZMP (Zero Moment Point). La stabilité du système est assurée si ce point est à l'intérieur du polygone de sustentation. Il s'agit de l'équivalent dynamique du projeté du centre de gravité. Ce critère de stabilité fut proposé par M. Vukobratovic en 1968 et a depuis quelques années été efficacement utilisé [Vukobratovic 2004] [J. Kim 2002], [S. Kajita 2003].

3.3.2.1 Interprétation physique du ZMP

Prenons un solide quelconque en contact avec le sol. Pour simplifier, posons $\vec{F_A}$ et M_A respectivement une force et un moment appliqués au point A représentant l'ensemble des actions qui agissent sur le solide (figure 3.17). Dans le cas de l'étude d'un robot bipède, le solide serait un pied, et $\vec{F_A}$, l'influence du corps sur le pied. Dans le cas de B2P2, le solide représenterait le châssis, et $\vec{F_A}$ l'influence de la partie mobile sur celui ci. D'après [Vukobratovic 2004], le moment exercé par la force $\vec{F_A}$ peut être compensé en modifiant le point d'action de la force de réaction du sol $\vec{R}$. Ainsi, l'équilibre est assuré si $\vec{R}$ peut être évaluée en un point du sol P pour lequel M_x et M_y (composantes horizontales du moment généré par R) sont nulles. Bien entendu, si ce point P est en dehors du polygone de sustentation, il n'a plus d'existence réelle car le sol n'exerce aucune réaction à cet endroit, on parlera alors de FZMP (Frictitious Zero Moment Point) [Vukobratovic 2004] ou de FRI (Foot Rotation Indicator)[Goswami 1999].

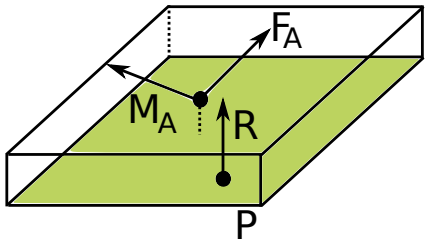


FIG. 3.17 – Forces en action sur le solide

3.3.2.2 Calcul du ZMP

Les coordonnées du ZMP sont décrites à partir des équations de la dynamique de Newton. Rappelons tout d'abord que le moment généré par une force $\vec{R}$ en un point z noté M_z et en un point 0 noté M_0, sont soumis à la relation suivante :

$$M_0 = M_Z + \vec{OZ} \wedge R \tag{3.26}$$

L'application des lois de Newton au robot soumis à la force de pesanteur ainsi qu'à la force de réaction du sol s'écrit :

$$\begin{cases} m\ddot{\vec{G}} = R - P \\ \delta_0 = M_O + \vec{OG} \wedge \vec{P} \end{cases} \tag{3.27}$$

avec :
m la masse totale du robot,
G, le centre de gravité,
R les forces de réaction du sol,
P la force de pesanteur (égale à $mg\vec{z}$),
δ_0 le moment dynamique du robot évalué au point O,
M_0 le moment généré par R au point 0.

A partir de 3.26, ces relations peuvent être réécrites :

$$\begin{cases} R = mg\vec{z} + m\ddot{G} \\ \delta_0 = M_Z + \vec{OZ} \wedge \vec{R} + \vec{OG} \wedge \vec{P} \end{cases} \tag{3.28}$$

En considérant Z comme étant le ZMP, M_{Z_x} et M_{Z_y} sont nuls. On obtient alors la relation suivante :

$$\begin{cases} \delta_{0x} = Z_y R_z + G_y P_z - G_z P_y \\ \delta_{0y} = Z_y R_z + G_y P_z - G_z P_y \end{cases} \tag{3.29}$$

Ainsi, les coordonnées du ZMP (notées Z_y et Z_x) sont calculées directement en fonction du vecteur de coordonnées articulaires $\vec{q}$ par la relation :

$$\begin{cases} Z_y = \frac{\delta_{0x}(\ddot{q}) - G_y(q)P_z + G_z(q)P_y}{R_z(\ddot{q})} \\ Z_x = \frac{-\delta_{0y}(\ddot{q}) + G_z(q)P_x - G_x(q)P_z}{R_z(\ddot{q})}. \end{cases} \tag{3.30}$$

Notons que si la vitesse et l'accélération du robot sont nulles, alors le ZMP est naturellement confondu avec le projeté du centre de gravité. L'expression détaillée du ZMP pour le robot B2P2 est disponible dans l'annexe B.

3.3.3 Comparaison

Les critères d'équilibre cités ci-avant pourraient être utilisés pour calculer les commandes du robot lors de franchissement d'obstacles. Afin de pouvoir choisir le plus adéquat en ce qui concerne notre plate-forme, ils ont été analysés lors du franchissement d'un escalier. L'escalier considéré dans cette manipulation possède des marches de 15 cm de haut et de 28 cm de profondeur. B2P2 l'a franchi en étant télé-opéré avec une vitesse moyenne de 0.13 $m.s^{-1}$ tout en enregistrant les données relatives aux capteurs (inclinomètre et encodeurs rotatifs fixés sur l'axe des moteurs). A l'aide de ces rapports la trajectoire de plusieurs points de référence du robot au cours du franchissement a été relevée. Ces points de références notés P1, P2, P3, CoG (centre de gravité) et ZMP sont indiqués sur la figure 3.18 représentant le robot. P1, P2 et P3 permettent de suivre le mouvement des solides 1, 2 et 3.

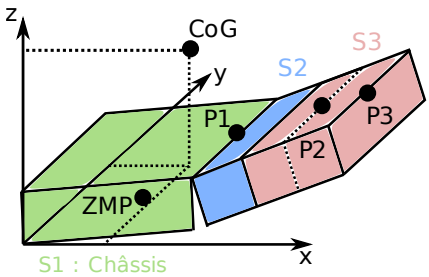


FIG. 3.18 – Points de référence lors du franchissement

Dans la mesure où l'escalier est franchi de manière directe (le robot ne se retrouve jamais en travers sur le nez d'une marche), il n'y a pas de risque de chute dûe

au roulis. En outre, il existe de nombreuses méthodes pour garantir le franchissement correct (en ligne droite) d'un escalier, elles seront d'ailleurs étudiées en section 5.8. Finalement, les influences du tangage sont bien plus importantes et difficiles à éviter, c'est pourquoi nous nous sommes concentrés sur ces dernières en analysant les coordonnées des points de référence sur l'axe O_x du repère fixé au robot présenté sur la figure 3.18. Ces résultats sont illustrés par la figure 3.19.

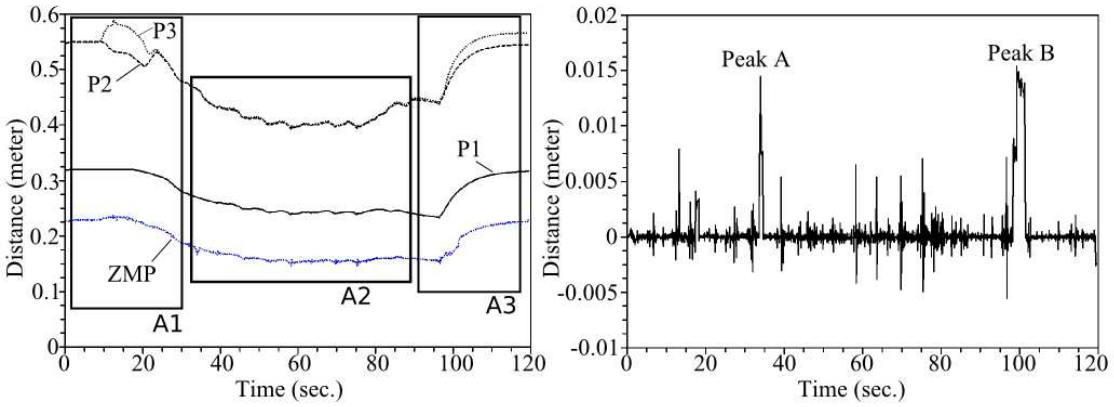


FIG. 3.19 – Résultats expérimentaux ; le graphique de gauche représente l'évolution du ZMP durant le franchissement d'un escalier (calculé à partir de l'équation 3.30). La différence entre le CoG et le ZMP est illustrée par le graphique de droite. L'axe des abscisses représente le temps en secondes ; l'axe des ordonnées reporte la composante O_x du repère présenté en figure 3.18.

Les zones A1, A2 et A3 qui y sont référencées correspondent aux phases du franchissement, respectivement l'approche de la marche, les marche intermédiaires puis la dernière marche. Notons que le robot est en mode tendu au début de l'expérience (P2 et P3 ne sont pas confondus lors de l'étape A1). La zone A2 illustre les différents transferts de masse qui ont lieu à chaque marche par les oscillations visibles sur chacun des points de référence. Ces oscillations, augmentent bien entendu avec la taille des marches. La partie droite de la figure 3.19 présente la différence instantanée entre le centre de gravité et le ZMP.

L'écart moyen entre la position du ZMP et la position du projeté du centre de gravité lors du franchissement d'un escalier est d'environ 0.21%. A la vue de cet écart et de la complexité du calcul du ZMP comparé au critère statique, il semble donc qu'avec une plate-forme telle que B2P2 dans le cas du franchissement d'un escalier à vitesse moyenne, le centre de gravité est un critère de stabilité suffisant.

Notons que les deux pics répertoriés sur la courbe correspondent en réalité à un léger dérapage du robot. En effet, comme la vitesse des articulations des modèles est

calculée à partir des relevés de codeurs reliés à l'axe du moteur, le moindre dérapage des chenilles fausse la mesure. L'ensemble de ces travaux concernant l'étude de l'équilibre de notre prototype à fait l'objet d'une publication dans [J. L. Paillat 2010]

3.4 Conclusion

La plate-forme analysée dans ce chapitre introduit un concept novateur permettant d'adapter un VGSTV à son environnement en utilisant des articulations actives afin d'augmenter le ratio de franchissement du robot. La conception de ce prototype a été ponctuée de plusieurs validations expérimentales. Les retours d'expérience ont été utilisés pour quantifier l'apport de cette particularité et de définir des séquences de pilotage propres à divers types d'obstacles. Le but de tout cela étant évidemment d'arriver à un contrôle autonome, ou semi-autonome de B2P2. Dans cette optique, une étude plus théorique sur le comportement dynamique du robot a été menée afin de définir si le centre de gravité constituait un critère suffisant pour contrôler l'équilibre du véhicule pendant un franchissement, ou bien, si l'apport du ZMP emprunté à la robotique humanoïde était significatif. A la vue des résultats expérimentaux (0.21% de différence), il semble inutile d'implémenter un contrôle du ZMP. Par conséquent, dans la suite du manuscrit et lors de la mise en place du simulateur présenté au chapitre 5, nous nous baserons sur l'analyse statique du comportement de B2P2. La projection du centre de gravité sera alors considérée comme un critère de stabilité suffisant.

Chapitre 4

Contrôle de la déformation des VGSTV via le calcul ensembliste

Sommaire

Dans ce chapitre, nous introduisons une méthode de contrôle de la déformation des VGSTV à articulations actives via l'analyse par intervalles et le calcul ensembliste. Le contrôle du robot B2P2 (voir chapitre 3) est effectué grâce à une fonction $f : \theta \mapsto l$ qui lie les différents éléments du vecteur de coordonnées articulaires à la longueur de la chenille. Il s'agit d'un cas particulier, car pour un VGSTV muni d'un nombre plus important d'articulations il deviendra impossible d'obtenir cette fonction. Ainsi, ce chapitre présente une généralisation du contrôle de ce genre de robot dont plusieurs exemples sont fournis par la figure 4.2.
Les outils présentés ici proposent de caractériser l'ensemble des coordonnées articulaires admissibles pour déformer le VGSTV à l'intérieur des chenilles. Après

avoir expliqué le problème en détails, nous proposons des algorithmes de résolution basés sur les techniques d'analyse par intervalles. Ces techniques dont le principe général est introduit en début de chapitre font l'objet d'une étude plus détaillée dans l'annexe C ; un nombre important de notions présentes dans ce chapitre se rapportent à cette annexe.

4.1 Principes généraux de l'analyse par intervalles

Cette introduction constitue une présentation succinte de ce qu'est l'arithmétique par intervalles, on y définira quelques notions indispensables utilisées dans la résolution du problème. Pour une définition plus détaillée, le lecteur pourra se référer à l'annexe C.

4.1.1 Arithmétique par intervalles

Un intervalle $[x]$ est un sous-ensemble fermé et connexe de $\mathbb{R}$. On pourra noter :

$$[x] = \{x \in \mathbb{R} | x^- \leq x \leq x^+, x^- \in \mathbb{R}, x^+ \in \mathbb{R}\} \tag{4.1}$$

où x^- et x^+ représentent respectivement les bornes inférieures et supérieures de l'intervalle $[x]$. Un intervalle sera ainsi noté : $[x] = [x^-; x^+]$.

L'arithmétique par intervalles regroupe un certain nombre d'opérateurs et d'outils liés à la manipulation de ce type de données :

– Intersection :

$$[x] \cap [y] = \{x \in [x] \text{ et } y \in [y]\}, \tag{4.2}$$

– Union :

$$[x] \cup [y] = \{x \in [x] \text{ ou } y \in [y]\}, \tag{4.3}$$

– Deprivation :

$$[x] \setminus [y] = \{x | [x] \text{ et } x \notin [y]\}, \tag{4.4}$$

– Produit scalaire :

$$[x] \times [y] = \{(x, y) | x \in \text{ [x] et } y \in [y]\}, \tag{4.5}$$

– Projection :

$$Proj_x(Z) = \{x \in [x] | \exists y \in \text{ [y] tel que } (x, y) \in Z\}. \tag{4.6}$$

Les opérations élémentaires classiques sur les réels sont aussi adaptées aux intervalles :

– L'addition :

$$[x] + [y] = [x^- + y^-; x^+ + y^+], \tag{4.7}$$

– La soustraction :

$$-[x] = [-x^+; -x^-] \tag{4.8}$$

$$[x] - [y] = [x] + (-[y]), \tag{4.9}$$

– La multiplication :

$$[x] * [y] = [min(x^-y^-, x^+y^-, x^-y^+, x^+y^+); max(x^-y^-, x^+y^-, x^-y^+, x^+y^+)], \tag{4.10}$$

– La division :

$$\frac{1}{[y]} = [\frac{1}{y^+}; \frac{1}{y^-}] \text{ si } 0 \notin [y],]-\infty; +\infty[\text{ sinon.} \tag{4.11}$$

4.1.2 Encadrement d'ensemble

L'analyse par intervalles se trouve être un outil puissant pour caractériser un ensemble de manière garantie via des outils d'inversions ensemblistes. Ces outils font appel à des notions simples qu'il semble toutefois nécessaire de définir ici.

4.1.2.1 Pavé

La notion de pavé (ou vecteur d'intervalles) de $\mathbb{R}^n$ permet d'encadrer un ensemble, voir d'approximer des vecteurs incertains. Il s'agit du produit cartésien de n intervalles écrit sous la forme suivante :

$$[\mathbf{x}] = [x_1^-; x_1^+] \times [x_2^-; x_2^+] \times ... \times [x_n^-; x_n^+] = \begin{pmatrix} [x_1^-; x_1^+] \\ [x_2^-; x_2^+] \\ ... \\ [x_n^-; x_n^+] \end{pmatrix} \tag{4.12}$$

Cette représentation induit un certain pessimisme comme le présente la figure 4.1. Ainsi, un ensemble $\mathbb{S}$ de forme quelconque sera encadré par un pavé $[\mathbf{x}]$ contenant tout les points de $\mathbb{S}$, mais aussi d'autres points n'appartenant pas à $\mathbb{S}$.

4.1.2.2 Bissection

Une opération de bissection permet de scinder un pavé en deux selon son plan principal ; plan de symétrie perpendiculaire au côté le plus large. Par exemple, la bissection du pavé $[\mathbf{x}] = [1; 2] \times [-1; 3]$ va générer deux nouveaux pavés : $[\mathbf{x}_1] = [1; 2] \times [-1; 1]$ et $[\mathbf{x}_2] = [1; 2] \times [1; 3]$.

Remarque 1. *Un pavé $[\boldsymbol{x}]$ pourra être qualifié d'inclus dans un ensemble défini par une fonction f de $\mathbb{R}^n$ dans $\mathbb{R}^m$ si $f([\boldsymbol{x}]) \subset \mathbb{S}$. Au contraire, il sera dit exclu si $f([\boldsymbol{x}]) \cap \mathbb{S} = \varnothing$. On le qualifiera d'incertain si $f([\boldsymbol{x}]) \cap \mathbb{S} \neq \varnothing$ et $f([\boldsymbol{x}]) \cup \mathbb{S} \neq \mathbb{S}$. La fonction f est quant à elle appelée fonction d'inclusion.*

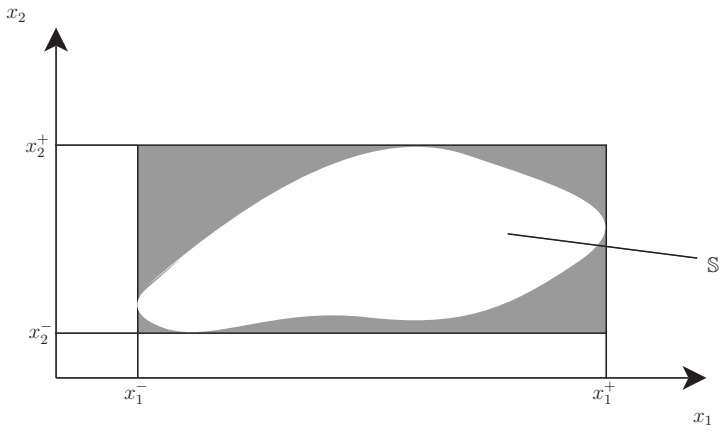


FIG. 4.1 – Encadrement d'un ensemble $\mathbb{S}$ par une boîte. Les points en gris n'appartiennent pas à $\mathbb{S}$

4.2 Un problème de satisfaction de contraintes

4.2.1 VGSTV étudiés

Les robots étudiés dans ce chapitre sont des VGSTV à articulations actives à l'instar du prototype B2P2. Ce dernier peut, par ailleurs, être qualifié de robot à un *étage* selon notre dénomination. Un *étage* constitue ainsi un couple d'articulations rotoïde / prismatique ; des exemples de robots à un ou plusieurs *étages* sont représentés sur la Figure 4.2. Cette représentation permet en outre de généraliser le comportement de n'importe quelle structure (RRR, RRP, RPR, etc...). On notera que la première articulation rotoïde relie toujours la seconde et la troisième roue ; la longueur entre la première et la seconde roue, notée B, constitue la première articulation prismatique, elle n'appartient cependant pas au premier étage et constitue un paramètre qui sera souvent considéré fixe. Le vecteur de coordonnées articulaires

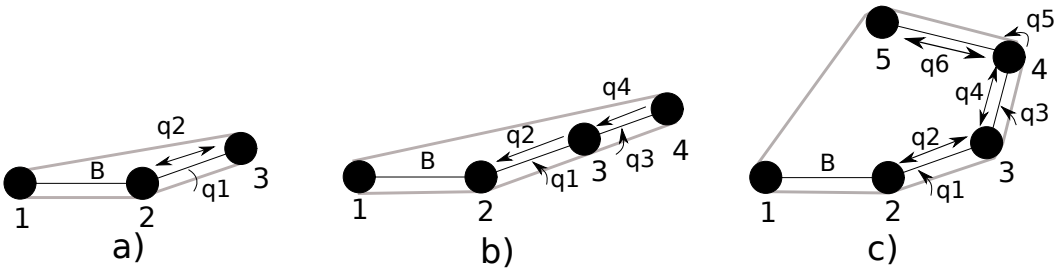


FIG. 4.2 – a) VGSTV à un étage. b) VGSTV à deux étages. c) VGSTV à trois étages.

sera noté **q** et défini tel que :

$$\mathbf{q} = \begin{bmatrix} B \\ q_1 \\ q_2 \\ \dots \\ q_{nb_{Etages} \times 2} \end{bmatrix} \tag{4.13}$$

4.2.2 Contrainte due à la chenille

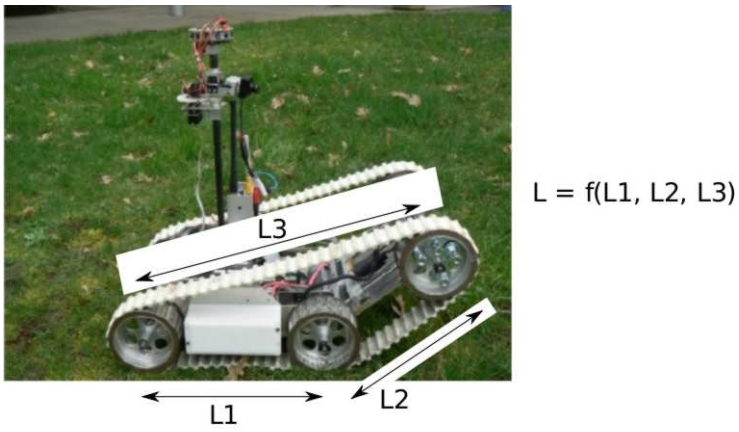


FIG. 4.3 – L'enveloppe convexe L est une fonction de L_1, L_2 et L_3.

Le contrôle de la déformation d'un VGSTV à n étages à l'intérieur d'une chenille est contraint par la longueur de celle-ci. En effet, lors du mouvement, le périmètre de l'enveloppe convexe formée par les roues en contact avec la chenille ne doit pas dépasser sa taille nominale (notée L_{max}) pour ne pas risquer de dommages mécaniques. Cependant, il est aussi indispensable de maintenir la taille de cette enveloppe au dessus d'une limite minimale (L_{min}) pour éviter de décheniller. Ainsi, si on note L la taille de cette enveloppe convexe (cf Figure 4.3), la contrainte qui doit être respectée lors de la déformation est donnée par :

$$L_{min} < L < L_{max} \tag{4.14}$$

Remarque 2. *Suivant le nombre d'étages du robot et la position de chacune des articulations, l'enveloppe convexe n'est pas toujours formée grâce aux mêmes roues, ainsi, la formulation de la fonction L ne sera pas toujours identique (figure 4.5)*

Notons que l'expérience acquise par l'utilisation du robot B2P2 démontre que la contrainte $l < L_{max}$ constitue un paramètre très critique. En effet, l'élasticité de la chenille ne permet pas de la tendre au delà de sa longueur nominale sous peine d'empêcher les moteurs de propulsion du robot de fonctionner correctement. Dans certains cas, cela peut aussi engendrer d'importants dommages mécaniques. Il est

donc primordial de garantir la contrainte 4.14.

Des méthodes d'optimisation locales sous contraintes, peuvent être utilisées pour contrôler la trajectoire du vecteur **q** à l'instar de la commande des robots manipulateurs. Ces méthodes ne sont cependant pas garanties. En effet, la trajectoire à suivre est alors discrétisée, puis l'erreur entre la trajectoire voulue et la trajectoire réelle est minimisée à chaque pas. Le respect des contraintes n'est donc absolument pas garanti de manière continue.
Afin d'illustrer ce phénomène, prenons un robot équipé d'un étage unique (typiquement B2P2). La trajectoire désirée est représentée par la figure 4.4a). Or, la longueur minimale de l'articulation prismatique notée q_2 sur la figure 4.4b) rend impossible le positionnement du robot à plat en respectant la contrainte lié à la longueur de la chenille (on obtient alors $L > L_{max}$). Afin de déformer le robot sans risquer de casse, le pas de discrétisation doit alors être très faible, ce qui peut entraîner un temps de traitement important, sans pour autant fournir un résultat garanti.

Les travaux de Lengagne concernant la planification de mouvements sûrs ont également soulevé ces problèmes. En effet, il a été démontré dans [Sébastien Lengagne 2007] et [Sébastien Lengagne 2008] que l'utilisation de méthodes non garanties peut s'avérer très dangereux pour l'intégrité du robot. Une méthode d'optimisation hybride utilisant une discrétisation garantie en encadrant les contraintes par des intervalles a alors été utilisée.

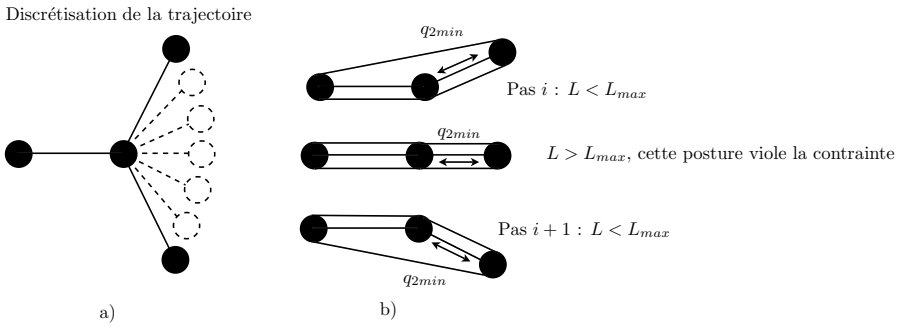


FIG. 4.4 – a) : Exemple de discrétisation de la trajectoire du robot ; à chaque pas, q_2 est déterminé de manière à minimiser $L - L_{max}$. b) La posture "robot à plat" viole la contrainte, cependant, comme les postures étudiées au pas i et $i+1$ acceptent une solution, la trajectoire se retrouve validée.

Dans notre cas, la garantie des résultats dans l'encadrement $[L_{min}; L_{max}]$ revêt un caractère indispensable au fonctionnement du robot. Les méthodes d'optimisa-

tion classique ne pouvant garantir ce résultat nous nous sommes efforcés de résoudre le problème de planification de trajectoire dans l'espace de travail avec les outils issus de l'analyse par intervalles.

4.2.3 Notation et énumération des enveloppes convexes

L'espace articulaire d'un robot est un ensemble de points accessibles dans le repère des coordonnées articulaires. Afin de contrôler la déformation du VGSTV, il est nécessaire de déterminer un sous-ensemble de l'espace articulaire noté espace articulaire contraint. Ce sous-ensemble regroupe les coordonnées articulaires accessibles par le robot pour lesquelles le périmètre de l'enveloppe convexe formée par les roues du robot est compris dans $[L_{min}; L_{max}]$.

Cette contrainte simple ne peut cependant pas être formulée directement. En effet, un VGSTV à n étages est associé à un nombre fini d'enveloppes convexes, chacune identifiée par une fonction L évaluant son périmètre. Or, il existe autant de formulation de L différente que d'enveloppes convexes.
Nous proposons alors, dans un premier temps de lister toutes ces enveloppes afin d'être en mesure de choisir la formulation de L qui convient étant donné un jeu de paramètres $\mathbf{q}$ pour déterminer si ce dernier affecte ou non l'intégrité de l'ensemble robot / chenille.

4.2.3.1 Définition d'une enveloppe convexe

Posons un numéro (de 1 à n) pour chaque roue d'un VGSTV. Ainsi, une enveloppe convexe H est associée à une suite de chiffres correspondant à l'ordre dans lequel les roues du robot touchent la chenille (pris dans le sens trigonométrique). La Figure 4.5 présente ainsi deux configurations d'un robot correspondant respectivement aux enveloppes convexes H_{1234} et H_{1243}. A ces deux enveloppes sont associées des fonctions L_i telles que :

$$\begin{aligned} L_{1234} &= \| \vec{V_{12}} \| + \| \vec{V_{23}} \| + \| \vec{V_{34}} \| + \| \vec{V_{41}} \| + 2\pi R \\ &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} + \sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2} \\ &\quad + \sqrt{(x_4 - x_3)^2 + (y_4 - y_3)^2} + \sqrt{(x_1 - x_4)^2 + (y_1 - y_4)^2} + 2\pi R, \end{aligned}$$

$$\begin{aligned} L_{1243} &= \| \vec{V_{12}} \| + \| \vec{V_{24}} \| + \| \vec{V_{43}} \| + \| \vec{V_{31}} \| + 2\pi R \\ &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} + \sqrt{(x_4 - x_2)^2 + (y_4 - y_2)^2} \\ &\quad + \sqrt{(x_3 - x_4)^2 + (y_3 - y_4)^2} + \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2} + 2\pi R. \end{aligned}$$

où R représente le rayon d'une roue, comme il s'agit d'un paramètre constant, le terme $2\pi R$ n'apparaitra plus dans la suite de ces travaux. $\| \vec{V_{xy}} \|$ sont des fonctions du vecteur de coordonnées articulaires (ici $\mathbf{q} = [q_1, q_2, q_3, q_4]^T$).

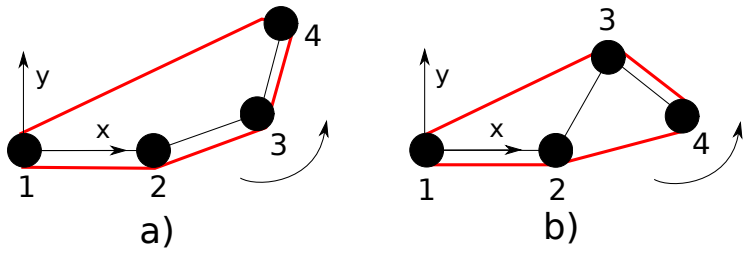


FIG. 4.5 – a) enveloppe notée : H_{1234} b) enveloppe notée : H_{1243}

4.2.3.2 Un nombre fini d'enveloppes

L'ensemble des configurations ou enveloppes qui peuvent être prises par le robot est défini grâce à quelques opérations d'arithmétique combinatoire réunies dans un algorithme dépendant du nombre de roues du robot. Cet algorithme a pour but de fournir toutes les enveloppes convexes possibles, chaque enveloppe appartient à une classe liée au nombre de roues en contact avec la chenille (de 3 à nb_{roues}).
La figure 4.6 qui illustre toutes les permutations possibles de l'ordre des roues pour un robot à quatre roues montre que certaines enveloppes sont identiques au sens de la formulation de la fonction L_i qui évalue leur périmètre ; notamment les enveloppes H_{1342} et H_{2134} en rouge sur la figure. Les permutations cycliques d'indices doivent alors être évitées pour distinguer uniquement les enveloppes possédant une formulation du périmètre L_i distincte.

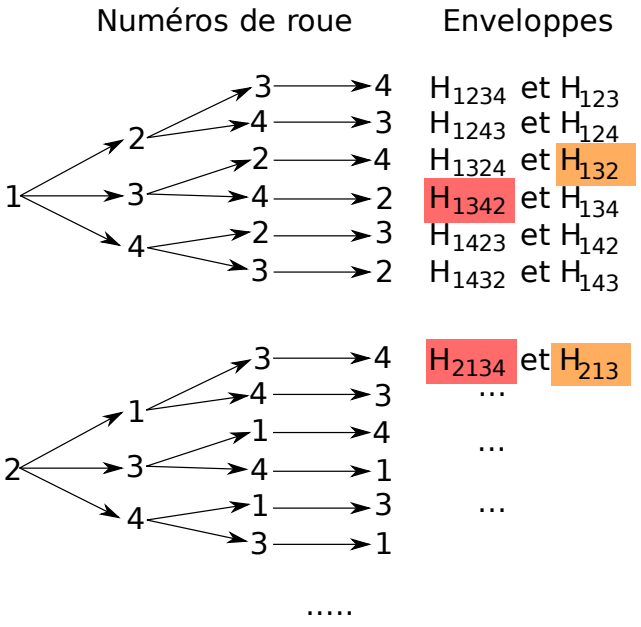


FIG. 4.6 – Permutations possibles pour un VGSTV à 4 roues ; les enveloppes colorées sont identiques au sens de la formulation de leur périmètre.

Classe	3	4
Combinaisons	123	1234
	124	
	234	

TAB. 4.1 – Combinaisons pour un VGSTV à quatre roues

D'une manière plus générale, notons nb_{roues} le nombre de roues du VGSTV considéré et $Classe$ le nombre de roues composant l'enveloppe (de 3 à nb_{roues}). L'ensemble des combinaisons de nb_{roues} parmi $Classe$ forme autant d'enveloppes possibles car, par définition, la combinaison ne tient pas compte de l'ordre des éléments. Ainsi des enveloppes comme H_{12345} et H_{23451} seront considérées équivalentes.

Considérons un VGSTV à quatre roues pour lequel deux classes d'enveloppes peuvent être obtenues, le nombre de combinaisons possibles est donné par : $C_4^4+C_4^3 = 4$; ces combinaisons sont listées dans le tableau 4.1. Évidemment, cela ne constitue pas la totalité des enveloppes ; $(Classe-1)$ permutations pour chaque combinaison doivent être effectuées afin de déterminer l'ensemble des enveloppes accessibles en évitant les permutations cycliques d'indices. Formellement, le nombre d'enveloppes d'un VGSTV sera donc donné par :

$$nb_{enveloppes} = C_{nb_{roues}}^{nb_{roues}} \times !(nb_{roues}-1) + C_{nb_{roues}}^{nb_{roues}-1} \times !(nb_{roues}-2) + ... + C_{nb_{roues}}^{3} \times !2 \tag{4.15}$$

Le tableau 4.2 répertorie ainsi la totalité des enveloppes obtenues pour un robot à quatre roues dont le nombre est de : $C_4^4 \times !3 + C_4^3 \times !2 = 1 \times 6 + 3 \times 2 = 12$. La liste des enveloppes convexes s'obtient de manière générale grâce à l'algorithme 1.

Algorithme 1 Algorithme de détection d'enveloppes

Précondition : Int : nb_{roues}

1: **pour** $i = nb_{roues}$ à 2 pas=-1 **faire**
2: PointsDeDepart=combinaisons(nb_{roues},i)
3: **pour** $i = 1$ à taille de PointsDeDepart pas=$+1$ **faire**
4: Enveloppe[i]=permutations(PointsDeDepart[i])

Remarque 3. *Par convention, lors du calcul des enveloppes possibles, chaque articulation rotoïde possède un domaine de travail équivalent à* $[-\pi, \pi]$ *; les articulations prismatiques possèdent un domaine de travail équivalent à* $[0, +\infty]$

4.2.4 Caractérisation d'une enveloppe convexe

Pour un vecteur **q** donné, le robot est dans une position appartenant à l'enveloppe i. Il convient alors d'évaluer si la contrainte $L_{min} < L_i < L_{max}$ est respectée. Cependant, il est indispensable de caractériser chacune des enveloppes via une

Classe	3	4
Enveloppes	**123**	**1234**
	132	1243
	124	1423
	142	1432
	234	1342
	243	1324

TAB. 4.2 – Enveloppes pour un VGSTV à quatre roues

contrainte nécessaire et suffisante de manière à identifier facilement la formulation L_i à utiliser telle que :

$$^{nb_{roues}}E_i(\mathbf{q}) \geq 0 \Leftrightarrow L_i(\mathbf{q}) \text{ correspond au périmètre de l'enveloppe convexe} \quad (4.16)$$

où nb_Roues correspond au nombre de roues du robot étudié, et i l'enveloppe concernée.

En effet, pour deux enveloppes différentes i et j, et un vecteur $\mathbf{q}$ donné, il est tout à fait envisageable d'obtenir $L_{min} < L_i(\mathbf{q}) < L_{max}$ et $L_j(\mathbf{q}) > L_{max}$. En guise d'illustration, la figure 4.7 présente deux positions différentes d'un robot à quatre roues associées respectivement aux enveloppes H_{124} et H_{1243}. Notons $\vec{v}$ et $\vec{w}$ leurs vecteurs de coordonnées articulaires respectifs. On peut constater que :

- $L_{min} < L_{124}(\vec{v}) < L_{max}$
- $L_{1243}(\vec{w}) > L_{max}$

ainsi, la position induite par $\vec{v}$ respecte la contrainte alors que $\vec{w}$ ne fait pas partie de l'espace articulaire contraint.

Cependant, notons aussi que la position des roues 1, 2 et 4 ne change pas entre les deux configurations. Rappelons alors la formulation de L_{124} :

$$L_{124} = \sqrt{(x_2 - x_1)^2 - (y_2 - y_1)^2} + \sqrt{(x_4 - x_2)^2 - (y_4 - y_2)^2} + \sqrt{(x_1 - x_4)^2 - (y_1 - y_4)^2} \quad (4.17)$$

Cette formulation ne prend bien évidement pas en compte la position des roues qui ne font pas partie de l'enveloppe (ici, la roue 3). Ainsi :

- $L_{min} < L_{124}(\vec{w}) < L_{max}$

or $\vec{w}$ n'appartient pas au CWA. Il est donc très important de déterminer a priori la formulation $L_i(\mathbf{q})$ à utiliser. Autrement dit, une contrainte qui indique que $\vec{w}$ n'est pas compatible avec l'enveloppe H_{124} est donc ici indispensable pour ne pas considérer le résultat donné par $L_{124}(\vec{w})$.

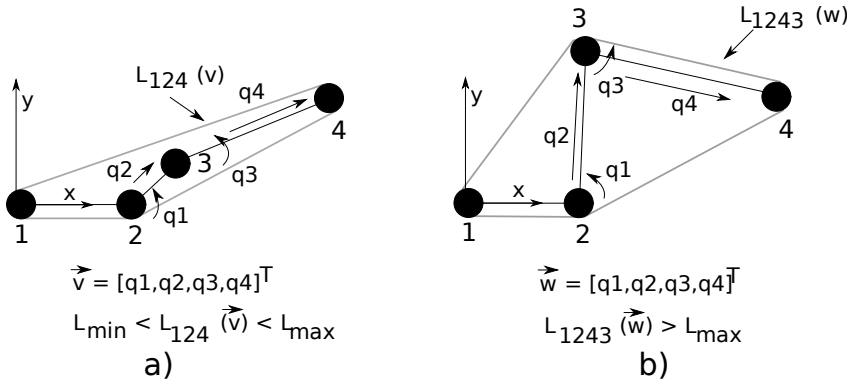


FIG. 4.7 – a) : L'enveloppe formée par les roues du robot est de type H_{124}, et son périmètre respecte la contrainte b) : H_{1234}, seule la roue 3 a bougé, la position des roues 1, 2 et 4 dans le plan xy est restée la même, $L_{1234}(\vec{w}) > l_{max}$ donc $\vec{w}$ n'appartient pas à l'espace articulaire contraint.

4.2.4.1 Une contrainte nécessaire pour garantir l'ordre des roues en contact avec la chenille

L'enveloppe convexe d'un nuage de points dans le plan peut aisément être déterminée par l'algorithme dit parcours de Graham [Graham 1972] dont le détail est présenté en Annexe D. La démarche consiste à parcourir tous les points en déterminant si trois points consécutifs appartiennent à l'enveloppe grâce au signe de la composante en z du produit vectoriel formé par ces éléments. Il renseigne sur le sens du chemin (vers la gauche où vers la droite). Une illustration est donnée par la figure 4.8.

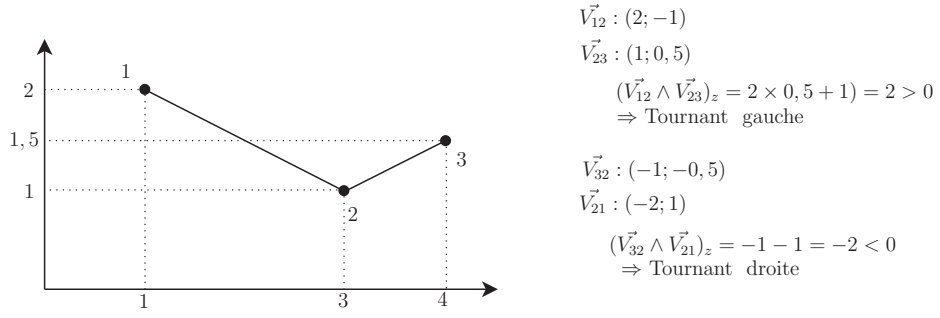


FIG. 4.8 – Le signe de la composante z du produit vectoriel (déterminent entre les deux vecteurs) renseigne sur le sens du chemin.

La généralisation de cet algorithme définit un jeu de contraintes qui garantit l'ordre dans lequel les roues sont en contact avec la chenille. En effet, un ensemble de points constitue une enveloppe convexe si tous les triplets de points consécutifs forment un tournant dans le même sens.

Ainsi, une contrainte garantissant l'ordre dans lequel les roues touchent la chenille est déterminée directement à partir de la désignation de l'enveloppe. Finalement, pour une enveloppe telle que H_{12345}, les points 1, 2, 3, 4 et 5 forment une enveloppe convexe si et seulement si les vecteurs $\vec{V_{12}}$ et $\vec{V_{23}}$ forment un tournant à gauche, tout comme $\vec{V_{23}}$ et $\vec{V_{34}}$, $\vec{V_{34}}$ et $\vec{V_{45}}$ puis $\vec{V_{45}}$ et $\vec{V_{51}}$. Cette contrainte notée G_{12345} s'écrit sous la forme suivante :

$$G_{12345}(\mathbf{q}) = \begin{pmatrix} (\vec{V_{12}} \wedge \vec{V_{23}})_z \\ (\vec{V_{23}} \wedge \vec{V_{34}})_z \\ (\vec{V_{34}} \wedge \vec{V_{45}})_z \\ (\vec{V_{45}} \wedge \vec{V_{51}})_z \\ (\vec{V_{51}} \wedge \vec{V_{12}})_z \end{pmatrix} \geq 0 \tag{4.18}$$

La figure 4.9 illustre ceci pour différentes configurations du robot. Notons que les vecteurs $\vec{V_{xy}}$ sont obtenus grâce aux coordonnées des roues du robot dans le repère xy présenté sur la figure 4.5 qui dépendent directement de $\mathbf{q}$.

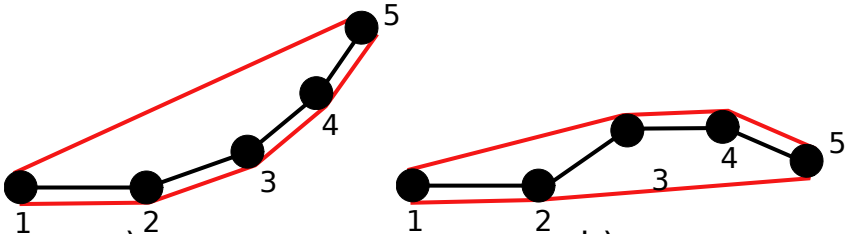


FIG. 4.9 – a) : Enveloppe H_{12345}, par définition, on obtient $G_{12345}(\mathbf{q}) \geq 0$. b) : Enveloppe H_{12543}, on constate que $(\vec{V_{23}} \wedge \vec{V_{34}})_z < 0$, ainsi la contrainte $G_{12345}(\mathbf{q}) \geq 0$ n'est plus respectée.

Remarque 4. *La contrainte $G_i(\boldsymbol{q})$ prend en compte les roues en contact avec la chenille et garantit qu'elles sont correctement ordonnées. Elle ne prend cependant pas en compte les roues absentes de l'enveloppe convexe. Prenons une enveloppe notée H_{1234} pour un robot à 5 roues, naturellement, la position de la roue 5 ne sera pas considérée lors de l'évaluation de G_{1234}. Or il semble indispensable de s'assurer que cette roue est bien à l'intérieur de l'enveloppe formée par les roues 1, 2, 3 et 4 pour garantir que le robot est bel et bien dans une position du type H_{1234}.*

4.2.4.2 Une contrainte nécessaire pour garantir la position des roues non comprises dans l'enveloppe

Les remarques concernant le formalisme des contraintes précédemment cités montrent que les roues qui ne font pas partie de l'enveloppe convexe doivent aussi

être prises en compte afin de garantir la posture prise par le robot. En effet, la condition G_i nécessaire afin de s'assurer de l'ordre des roues de l'enveloppe convexe ne garantit en aucun cas la position des roues hors-enveloppe. En d'autres termes, un vecteur de coordonnées articulaires $\vec{q}$ positionne le robot dans une configuration impliquant l'enveloppe convexe i si et seulement si les roues de l'enveloppe convexe sont correctement ordonnées ($G_i(\mathbf{q}) > 0$) et si les roues hors-enveloppe sont bien à l'intérieur de cette dernière comme illustré par la figure 4.10.

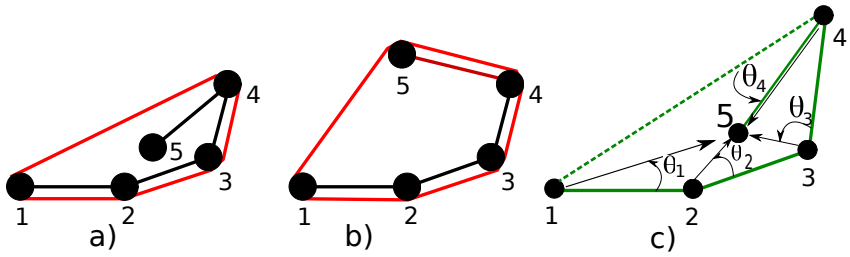


FIG. 4.10 – a) : Enveloppe H_{1234}, ainsi $G_{1234} \geq 0$. b) : Enveloppe H_{12345}, or on constate $G_{1234} \geq 0$ et $G_{12345} \geq 0$ c) : Pour différencier ces deux cas, il est nécessaire d'ajouter une seconde contrainte telle que les angles notés θ_x forment un tournant à gauche.

Formellement, cela est aussi résolu via l'utilisation du projeté en z du produit vectoriel ; ainsi, la contrainte J^5_{1234} suivante :

$$J^5_{1234}(\mathbf{q}) = \begin{pmatrix} (\vec{V_{12}} \wedge \vec{V_{15}})_z \\ (\vec{V_{23}} \wedge \vec{V_{25}})_z \\ (\vec{V_{34}} \wedge \vec{V_{35}})_z \\ (\vec{V_{41}} \wedge \vec{V_{45}})_z \end{pmatrix} \geq 0 \tag{4.19}$$

permet de s'assurer que la $5^{ème}$ roue d'un VGSTV à 5 roues est bien à l'intérieur de l'enveloppe convexe H_{1234}.

Remarque 5. *Par convention les contraintes de type J seront associées à un indice i représentant l'enveloppe considérée, et à un exposant $l_n \setminus i$ désignant une roue l_n non présente dans l'enveloppe i.*

4.2.4.3 Une contrainte nécessaire pour éviter les croisements de segments du robot

Lorsque le nombre de roues du robot est supérieur à 4, la position d'une roue hors enveloppe, même garantie à l'intérieur de l'enveloppe, peut engendrer certains problèmes. En effet, comme l'illustre la figure 4.11, il arrive qu'un des solides qui composent le robot ne fasse pas partie de l'enveloppe convexe. Il semble alors indispensable de s'assurer que les solides associés aux roues hors enveloppe ne se retrouvent pas en contact avec d'autres solides qui composent le robot.

Considérons les deux segments notés V_{23} et V_{45} et quatre angles notés θ_j sur la figure 4.11. Typiquement, ces deux segments se croisent si et seulement si $signe(\theta_1) \neq signe(\theta_2)$ et $signe(\theta_3) \neq signe(\theta_4)$. Le signe de ces angles est déterminé en utilisant encore une fois la composante z du produit vectoriel. Ainsi, la contrainte K qui permet de s'assurer que la roue 5 ne vient pas en contact avec le segment 23 notée K^5_{23} est donné par :

$$K^5_{23}(\mathbf{q}) = \begin{pmatrix} (\vec{V_{42}} \wedge \vec{V_{45}})_z \times (\vec{V_{43}} \wedge \vec{V_{45}})_z \\ (\vec{V_{24}} \wedge \vec{V_{23}})_z \times (\vec{V_{25}} \wedge \vec{V_{23}})_z \end{pmatrix} \geq 0 \tag{4.20}$$

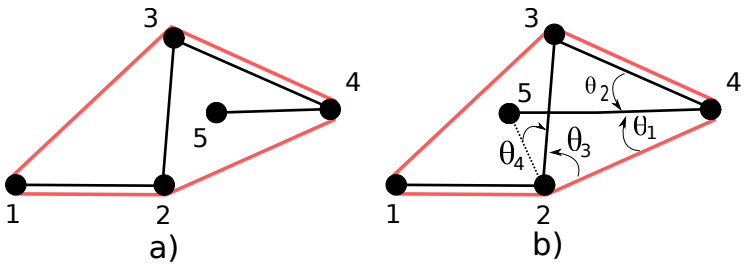


FIG. 4.11 – Exemple de croisement de deux solides du robot

Remarque 6. *Par convention les contraintes de type K seront associées à un indice $s_p \setminus i$ désignant un solide non présent dans l'enveloppe i, et à un exposant $l_m \setminus i$ désignant une roue l_m non présente dans l'enveloppe i.*

Remarque 7. *Les segments hors enveloppe notés $s_p \setminus i$ sont facile à répertorier directement grâce à la notation de l'enveloppe. En effet, si deux indices de roue tels que n et $n+1$ entourent un ou plusieurs indices $n+x$, alors le segment $V_{n,n+1}$ est un segment hors enveloppe. Notons que si le nombre de roues hors enveloppe est supérieur ou égal à trois, les segments associés seront assimilés à des segments hors enveloppe car ils présentent alors un risque comme l'illustre la figure 4.12.*

Remarque 8. *Une enveloppe i appartient à une classe comme défini en section 4.2.3.2. $Classe[i]$ représente donc la classe de l'enveloppe, autrement dit le nombre de roues appartenant à cette enveloppe.*

Nous proposons alors une notation générale pour formuler une contrainte E nécessaire et suffisante pour être certain de l'enveloppe convexe formée pour une position donnée du robot. A l'instar des contraintes illustrées précédemment, cette notation est composée d'un indice i désignant l'enveloppe considérée et d'un exposant nb_{roues} illustrant le nombre de roues du robot. Elle est donc composée d'une contrainte G_i, de $nb_l = n - Classe[i]$ contraintes J^k_i et $nb_s \times nb_s$ contraintes K. Notons que nb_l correspond au nombre de roues hors enveloppes et que nb_s représente

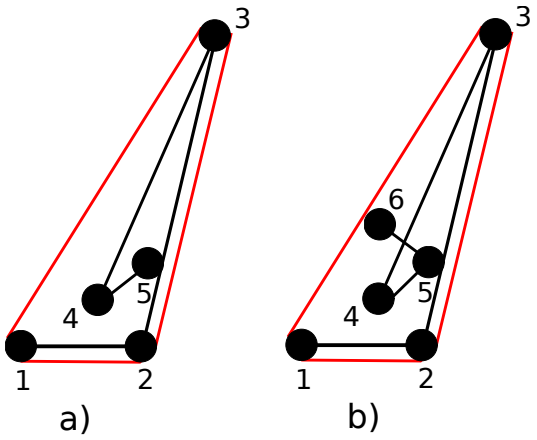


FIG. 4.12 – a) enveloppe comportant 2 roues hors enveloppe, il n'y a pas de risque de croisement. b) à partir de 3 roues hors enveloppe, l'un des segments (ici V_{34}) constitue un segment dit hors enveloppe.

le nombre de solides hors enveloppe. On obtient alors :

$$ {}^{nb_{roues}}E_i(\mathbf{q}) = \begin{pmatrix} G_i(\mathbf{q}) \\ J_i^{l_1 \setminus i}(\mathbf{q}) \\ \dots \\ J_i^{l_n \setminus i}(\mathbf{q}) \\ K_{s_1 \setminus i}^{l_1 \setminus i}(\mathbf{q}) \\ \dots \\ K_{s_p \setminus i}^{l_1 \setminus i}(\mathbf{q}) \\ \dots \\ K_{s_p \setminus i}^{l_m \setminus i}(\mathbf{q}) \end{pmatrix} \geq 0 \tag{4.21} $$

Remarque 9. *$l_j \setminus i$ indique ici une roue j du robot non présente dans l'enveloppe i considérée ; $s_j \setminus i$ indique un solide j non présent sur l'enveloppe i.*

4.2.4.4 Formalisme complet d'une contrainte nécessaire et suffisante

Prenons l'exemple illustré par la figure 4.13. Ce robot à 5 roues est dans une posture qui implique une enveloppe convexe H_{1234}, la taille de sa chenille est donc obtenue via la fonction $L_{1234}(\mathbf{q})$. Les roues 1, 2, 3, et 4 forment une enveloppe

convexe admissible si $E^5_{1234} \geq 0$ donc, si :

$$ {}^5E_{1234}(\mathbf{q}) = \begin{pmatrix} G_{1234}(\mathbf{q}) \\ J^5_{1234}(\mathbf{q}) \end{pmatrix} = \begin{pmatrix} (\vec{V_{12}} \wedge \vec{V_{23}})_z \\ (\vec{V_{23}} \wedge \vec{V_{34}})_z \\ (\vec{V_{34}} \wedge \vec{V_{41}})_z \\ (\vec{V_{41}} \wedge \vec{V_{12}})_z \\ (\vec{V_{12}} \wedge \vec{V_{15}})_z \\ (\vec{V_{23}} \wedge \vec{V_{25}})_z \\ (\vec{V_{34}} \wedge \vec{V_{35}})_z \\ (\vec{V_{41}} \wedge \vec{V_{45}})_z \end{pmatrix} \geq 0 \tag{4.22} $$

Dans ces conditions, $\vec{x}$, $\vec{y}$ et $L_{1234}(\vec{q})$ sont évalués comme :

$$ \begin{cases} L_{1234}(\mathbf{q}) &= \sqrt{(x_2-x_1)^2+(y_2-y_1)^2} + \sqrt{(x_3-x_2)^2+(y_3-y_2)^2} \\ & \quad + \sqrt{(x_4-x_3)^2+(y_4-y_3)^2} + \sqrt{(x_4-x_1)^2+(y_4-y_1)^2} \\ x_1(\mathbf{q}) &= 0 \\ y_1(\mathbf{q}) &= 0 \\ x_2(\mathbf{q}) &= B \\ y_2(\mathbf{q}) &= 0 \\ x_3(\mathbf{q}) &= x2 + q_2\cos(q_1) \\ y_3(\mathbf{q}) &= y2 + q_2\sin(q_1) \\ x_4(\mathbf{q}) &= x3 + q_4\cos(q_1+q_3) \\ y_4(\mathbf{q}) &= y3 + q_4\sin(q_1+q_3) \end{cases} \tag{4.23} $$

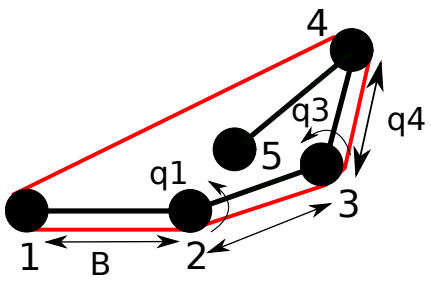


FIG. 4.13 – H_{1234} pour un robot à cinq roues.

Bien entendu, il est possible de considérer une enveloppe convexe constituée de plusieurs roues hors enveloppes comme c'est le cas de H_{123} pour un robot à 5 roues

(figure 4.14) :

$$ {}^5E_{123}(\mathbf{q}) = \begin{pmatrix} G_{123}(\mathbf{q}) \\ J^4_{123}(\mathbf{q}) \\ J^5_{123}(\mathbf{q}) \end{pmatrix} = \begin{pmatrix} (\vec{V_{12}} \wedge \vec{V_{23}})_z \\ (\vec{V_{23}} \wedge \vec{V_{31}})_z \\ (\vec{V_{31}} \wedge \vec{V_{12}})_z \\ (\vec{V_{12}} \wedge \vec{V_{14}})_z \\ (\vec{V_{23}} \wedge \vec{V_{24}})_z \\ (\vec{V_{31}} \wedge \vec{V_{34}})_z \\ (\vec{V_{12}} \wedge \vec{V_{15}})_z \\ (\vec{V_{23}} \wedge \vec{V_{25}})_z \\ (\vec{V_{31}} \wedge \vec{V_{35}})_z \end{pmatrix} \geq 0 \tag{4.24} $$

Dans ces conditions, $\vec{x}$, $\vec{y}$ sont évalués comme dans l'exemple précédent. $L_{123}(\vec{q})$ est alors donné par la formule suivante :

$$ L_{123}(\mathbf{q}) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} + \sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2} + \sqrt{(x_3 - x_1)^2 + (y_3 - y_1)^2} \tag{4.25} $$

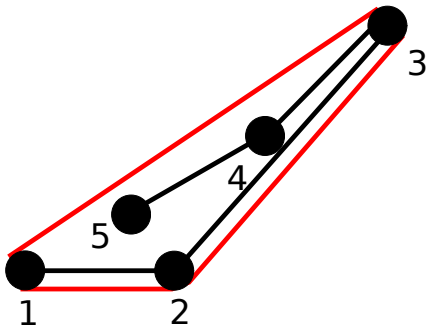


FIG. 4.14 – H_{123} pour un robot à cinq roues.

Enfin, lorsqu'il est question d'une enveloppe composée d'un solide traversant, comme par exemple H_{1243} (figure 4.15), E^5_{1243} est alors donné par :

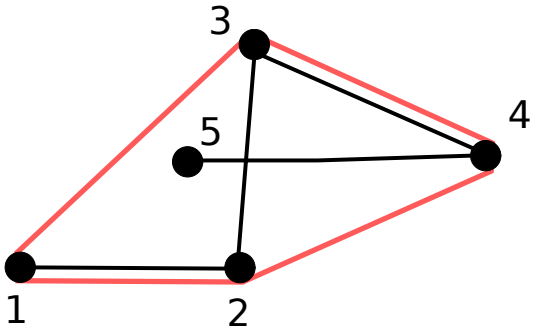


FIG. 4.15 – H_{1243} pour un robot à cinq roues.

$$
{}^{5}E_{1243}(\mathbf{q}) = \begin{pmatrix} G_{1243}(\mathbf{q}) \\ J^{5}_{1243}(\mathbf{q}) \\ K^{5}_{23}(\mathbf{q}) \end{pmatrix} = \begin{pmatrix} (\vec{V_{12}} \wedge \vec{V_{24}})_z \\ (\vec{V_{24}} \wedge \vec{V_{43}})_z \\ (\vec{V_{43}} \wedge \vec{V_{31}})_z \\ (\vec{V_{31}} \wedge \vec{V_{12}})_z \\ (\vec{V_{12}} \wedge \vec{V_{15}})_z \\ (\vec{V_{24}} \wedge \vec{V_{25}})_z \\ (\vec{V_{43}} \wedge \vec{V_{45}})_z \\ (\vec{V_{31}} \wedge \vec{V_{35}})_z \\ (\vec{V_{42}} \wedge \vec{V_{45}})_z \times (\vec{V_{43}} \wedge \vec{V_{45}})_z \\ (\vec{V_{24}} \wedge \vec{V_{23}})_z \times (\vec{V_{25}} \wedge \vec{V_{23}})_z \end{pmatrix} \geq 0 \qquad (4.26)
$$

4.2.5 Un problème de satisfaction de contraintes par enveloppe

Chaque enveloppe convexe possède un espace articulaire contraint particulier. Dans certains cas, il est même envisageable que l'intersection entre deux espaces associés à deux enveloppes différentes ne soit pas vide. Deux méthodes de résolution ont été développées pour caractériser cet espace de travail pour un VGSTV donné :

- Résolution à l'aide de SIVIA pour l'ensemble des enveloppes,
- Résolution à l'aide de contracteurs sous forme de problèmes de satisfaction de contraintes (CSP) qui traitent chaque enveloppe individuellement afin de déterminer un espace articulaire contraint par enveloppe.

La première solution possède l'avantage de ne fournir qu'un seul et unique pavage, cependant un nombre important de bissections est nécessaire pour y arriver. A l'inverse, la seconde solution fournit plusieurs pavages qu'il faut ensuite réunir, mais la contraction de l'ensemble réduit le nombre de bissections.

Ces deux méthodes d'inversion ensembliste consistent à évaluer $\mathbb{S}$, l'ensemble des pavés $[\mathbf{x}_1]$, $[\mathbf{x}_2]$ ... $[\mathbf{x}_n]$ qui garantissent l'équation $y = f(x_1, x_2...x_n)$. Pour chaque pavé $[\mathbf{x}]$, il existe un pavé $[\mathbf{y}]$ image de $[\mathbf{x}]$ par la fonction f. En connaissant le domaine de définition de y noté $\mathbb{Y}$, on peut déterminer si $[\mathbf{x}]$

- est une solution de $y = f(x)$, si $[\mathbf{y}] = [\mathbf{f}]\ (\ [\mathbf{x}]\ ) \subset \mathbb{Y}$,
- n'est pas une solution de $y = f(x)$, si $[\mathbf{y}] = [\mathbf{f}]\ (\ [\mathbf{x}]\ ) \cap \mathbb{Y} = \varnothing$
- contient des solutions et des non-solutions (qualifié d'ensemble incertain), si $[\mathbf{y}] = [\mathbf{f}]\ (\ [\mathbf{x}]\ ) \cap \mathbb{Y} \neq \varnothing$ et $[\mathbf{y}] = [\mathbf{f}]\ (\ [\mathbf{x}]\ ) \cap \mathbb{Y} \neq [\mathbf{f}]\ (\ [\mathbf{x}]\ )$.

La figure 4.16 résume ainsi le principe de l'inversion ensembliste. Les pavés rouges représentent les pavés solutions, les bleus, les pavés non-solutions, et les jaunes, les pavés incertains.

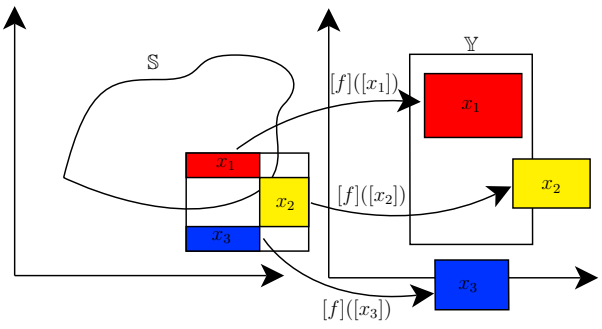


FIG. 4.16 – Principe de l'inversion ensembliste

Les algorithmes d'inversion ensembliste sont des algorithmes récursifs chargés d'évaluer un pavé, puis :

- de le bissecter s'il s'avère incertain (les pavés résultants seront alors ré-évalués successivement),
- de le conserver comme partie de la solution s'il s'avère être solution afin de déterminer une approximation intérieure de l'ensemble $\mathbb{S}$,
- de le rejeter si il s'avère être un pavé non solution.

Les deux méthodes de résolution citées ci-dessus possèdent cependant une différence fondamentale. La méthode SIVIA pour Set Inversion Via Interval Analysis a été introduite pour la première fois dans [L. Jaulin 1993] pour résoudre des problèmes d'estimation d'erreurs. Notons que la convergence de l'algorithme est prouvée dans cette référence. Il s'agit d'un algorithme récursif chargé de tester les pavés de manière globale et de bissecter petit à petit tout les pavés incertains. Il a notamment été utilisé pour la localisation de véhicule garantie (prenant en compte les erreurs dues à la mesure) dans [O. Lévêque 2000].

L'utilisation de contracteurs, ou du moins de la propagation de contraintes est introduite dans [Jaulin 2000]. Les auteurs estiment que certains problèmes peuvent voir réduire leur temps de convergence par un facteur 30. La taille des pavés à évaluer est alors réduite grâce aux contraintes liées au problème de manière à enlever les valeurs inconsistantes. Le nombre de bissection s'en trouve réduit, tout comme le temps de convergence. Ces deux méthodes sont détaillées dans les annexes C.1.3 et C.2.

4.2.5.1 Résolution par SIVIA

Cette approche propose de tester l'espace articulaire du robot afin de trier les pavés appartenant à l'espace articulaire contraint (quelle que soit l'enveloppe concernée) de manière à construire un pavage. L'algorithme SIVIA a donc été adapté à notre problème pour aboutir à l'algorithme 2.

A partir d'un pavé $[\mathbf{q}]$, la contrainte ${}^{nb_{Roues}}E_i(\mathbf{q})([\mathbf{q}])$ est évaluée pour chaque enveloppe i. Si une des enveloppes est compatible avec ce pavé $[\mathbf{q}]$, alors l'intersection de $L_i([\mathbf{q}])$ par rapport à l'ensemble $[L_{min}; L_{max}]$ est évaluée. Il y a alors deux possibilités, soit $[\mathbf{q}]$ est reconnu comme solution appartenant au CWA, soit il reste incertain car ne contenant pas uniquement des solutions. Les pavés incertains supérieurs à une taille notée ε sont bissectés puis chacun des pavés ainsi obtenus est évalué par SIVIA.
Le temps de calcul de cette procédure est donc assez long (limité par le paramètre ε), mais le pavage obtenu est unique pour toutes les enveloppes possibles.

4.2.5.2 Résolution à l'aide des contracteurs

a - Un CSP par enveloppe

Algorithme 2 Résolution avec la méthode SIVIA (D_i représente le domaine de définition des articulations)

```
1: SIVIA( [q] ∈ D_i)
2: pour tout i faire
3:   si E_i^{nb_roues}( [q] ) ≥ 0 alors
4:     si L_i([q]) ∩ [L_min; L_max] ≠ ∅ alors
5:       si L_i([q]) ∩ [L_min; L_max] = L_i([q]) alors
6:         [q] est une solution, on l'ajoute au CWA
7:         retour
8:       sinon
9:         si size([q])<ε alors
10:          [q] est trop petit et toujours incertain
11:        sinon
12:          [q] est incertain
13:          une bissection est nécessaire
14:          bisect( [q] , [q1] , [q2] )
15:          SIVIA( [q1] )
16:          SIVIA( [q2] )
```

Remarque 10. *Un contracteur noté $C_S([\boldsymbol{x}])$ permet de réduire la taille d'un pavé $[\boldsymbol{x}]$ sans perdre d'élément appartenant à l'ensemble S comme l'illustre la figure 4.17. On notera $\bar{C_S}([\boldsymbol{x}])$ un contracteur capable de réduire la taille d'un pavé $[\boldsymbol{x}]$ sans perdre d'élément n'appartenant pas à S (figure 4.18).*

Le formalisme CSP pour l'anglais "Constraint Satisfaction Problem" propose une formulation efficace pour caractériser un ensemble défini par un jeu de contraintes (cf Annexe C). Néanmoins, l'utilisation d'un outil tel qu'un contracteur oblige dans le cas considéré ici à écrire un CSP par enveloppe convexe. L'effet de consistance locale (étudié en Annexe C.2.4) implique que l'appel successif de contracteurs pose des problèmes d'optimalité. Ainsi, le pavé résultant de la contraction successive d'un ensemble **[q]** par les contracteurs relatifs aux enveloppes i, $j...n$ est différente de la contraction via un contracteur global, autrement dit :

$$C_i \circ C_j \circ .... \circ C_n([\mathbf{q}]) \neq C_{(i,j,...n)}([\mathbf{q}]) \tag{4.27}$$

où $C_{(i,j,...n)}$ désigne un contracteur global pour les enveloppes $i, j, ..., n$. Dans notre problème, comme la fonction $L(\mathbf{q})$ est formulée différemment pour chaque enveloppe i, il n'existe pas de contracteur global. Ainsi, le CSP ci-dessous sera résolu pour chaque enveloppe convexe i afin de déterminer un espace articulaire contraint par enveloppe noté $CArtS_i$ (Constraint Articular Space) :

$$\left\{ \begin{array}{rcl} V_i & : & q_1, q_2, ..., q_n \\ D_i & : & [q_1], [q_2], ..., [q_n] \\ C_i & : & min(^{nb_{roues}}E_i, -L_{min} + L_i, L_{max} - L_i)) \geq 0 \end{array} \right\} \tag{4.28}$$

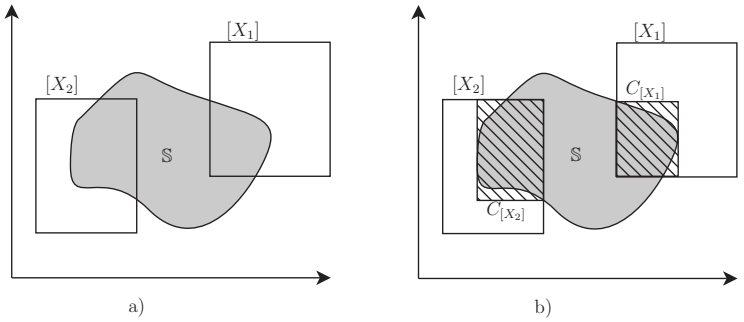


FIG. 4.17 – Effet d'un contracteur sur un pavé donné

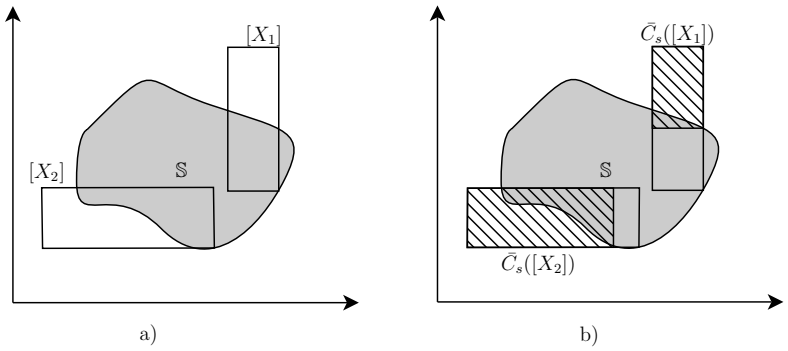


FIG. 4.18 – Effet d'un contracteur de non solution sur un pavé donné

b - Résolution du CSP Pour chaque enveloppe i, posons deux ensembles S_i (ensemble solution) et $\bar{S}_i$ (ensemble non solution) tels que :

$$\begin{array}{rcl} S_i & = & \{q \in D_i | min(E_i^{nb_{roues}}, -L_{min} + L_i, L_{max} - L_i)) \geq 0\} \text{ et} \\ \bar{S}_i & = & \{q \in D_i | min(E_i^{nb_{roues}}, -L_{min} + L_i, L_{max} - L_i)) < 0\} \end{array} \quad (4.29)$$

Ainsi, le contracteur de solution noté C_i réduira un pavé $[\mathbf{q}]$ sans perdre de solution, c'est-à-dire de pavés appartenant à l'ensemble S_i. Le contracteur de non solution noté $\bar{C}_i$ réduira, quant à lui, un pavé $[\mathbf{q}]$ sans perdre de non solutions, c'est à dire de pavés appartenant à l'ensemble $\bar{S}_i$ (ou n'appartenant pas à l'ensemble S_i).

L'algorithme 3 utilise ces deux contracteurs pour obtenir une approximation intérieure de l'ensemble S_i. Un pavé initial noté $[\mathbf{q}_{in}]$ est contracté pour fournir un

pavé réduit noté $[\mathbf{q}_{S_i}]$ contenant autant de solutions que le pavé $[\mathbf{q}_{in}]$. Ce pavé est ensuite contracté avec l'opérateur $\bar{C}_i$. Si un ensemble non solution est contenu dans $[\mathbf{q}_{S_i}]$, alors $[\mathbf{q}_{S_i}] \setminus \bar{C}_i([\mathbf{q}_{S_i}])$ est considéré comme faisant partie de l'approximation intérieure de S_i (figure 4.19). Ensuite, $[\mathbf{q}_{\bar{S}_i}]$ est bissecté et les pavés résultants sont évalués de façon récursive.

Algorithme 3 CArtSSolve (entrée : $[\mathbf{q}_{in}]$, sortie : $CArtS_i(\text{Pile}([Q]))$)

1: $[\mathbf{q}_{S_i}] = C_{S_i}([\mathbf{q}_{in}])$
2: **si** $[\mathbf{q}_{S_i}] \neq \varnothing$ **alors**
3: $\quad [\mathbf{q}_{\bar{S}_i}] = \bar{C}_{s_i}([\mathbf{q}_{S_i}])$
4: $\quad$ mettre $[\mathbf{q}_{S_i}] \setminus [\mathbf{q}_{\bar{S}_i}]$ dans la pile $CArtS_i$.
5: $\quad$ **si** $size([\mathbf{q}_{\bar{S}_i}]) > \varepsilon$ **alors**
6: $\quad\quad [\mathbf{q}_{in_1}]$, $[\mathbf{q}_{in_2}] =$ bissecter$([\mathbf{q}_{\bar{S}_i}])$
7: $\quad\quad$ CArtSSolve$([\mathbf{q}_{in_1}])$
8: $\quad\quad$ CArtSSolve$([\mathbf{q}_{in_2}])$

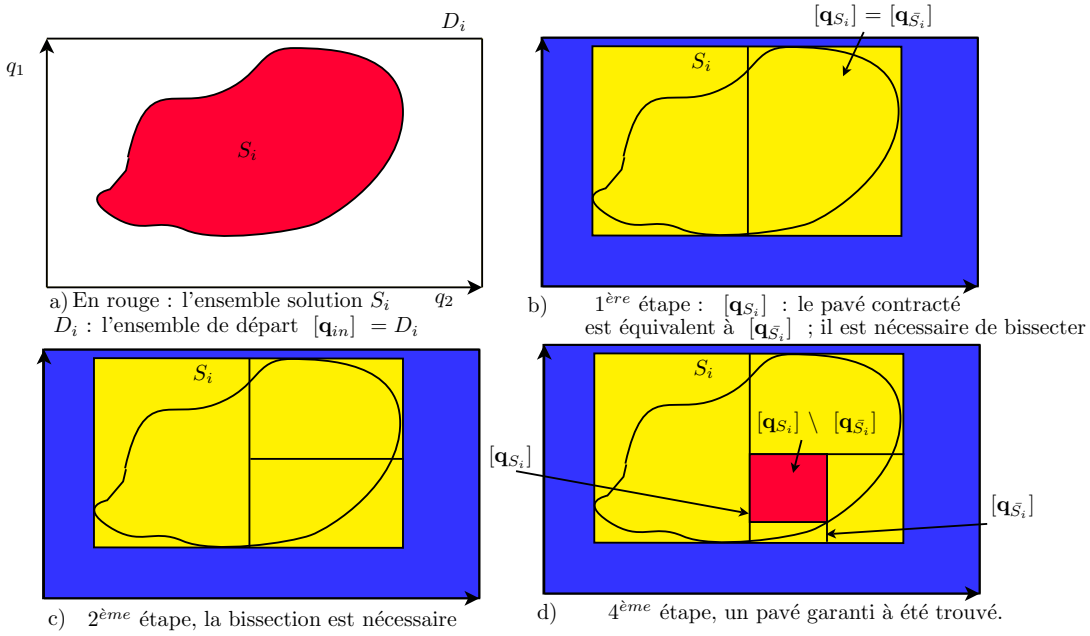


FIG. 4.19 – Contraction d'un ensemble

Remarque 11. *Notons, que lorsqu'il est impossible de positionner le robot dans une configuration impliquant une enveloppe i donnée, à cause d'un domaine de définition D_i trop restreint, le contracteur de non solution associé $\bar{C}_i$ ne sera pas en mesure*

de contracter un ensemble d'où :

$$\bar{C}_i([\boldsymbol{q}]) = [\boldsymbol{q}] \forall [\boldsymbol{q}] \in D_i \tag{4.30}$$

Bien entendu, l'algorithme 3 teste avant tout le contracteur de solution, et cherche à extraire les ensembles non solutions d'un ensemble contenant des solutions. Ainsi dans notre cas, on aura : $C_i([\boldsymbol{q}]) = \varnothing$ et $\bar{C}_i$ ne sera pas évalué.

Cependant, si on décide de caractériser tous les espaces articulaires contraints en appelant successivement les contracteurs des enveloppes i, j, ... n et que l'enveloppe j est impossible à obtenir l'algorithme va alors bissecter de manière infinie car $C_i \circ C_j \circ .. \circ C_n([\boldsymbol{q}]) = [\boldsymbol{q}']$, mais comme $\bar{C}_j([\boldsymbol{q}]) = [\boldsymbol{q}] \forall [\boldsymbol{q}] \in D_j$, alors $\bar{C}_i \circ \bar{C}_j \circ .. \circ \bar{C}_n([\boldsymbol{q}']) = [\boldsymbol{q}'] \forall [\boldsymbol{q}'] \in D_j$. Il ne sera alors jamais possible d'obtenir un ensemble solution différent d'un ensemble non solution, ce qui justifie l'utilisation de plusieurs CSP dans cette application.

4.3 Des contracteurs génériques

Comme tous les CSPs traités sont constitués d'un même jeu de contraintes, l'algorithme de contraction reste le même quelle que soit l'enveloppe i considérée. La structure générale est présentée par l'algorithme 4. Le pavé noté $[\mathbf{q}]$ est tout d'abord projeté dans le plan x, y, puis chaque produit vectoriel est utilisé pour contracter ce pavé en deux dimensions ; la contrainte relative à la longueur de la chenille est ensuite utilisée pour une dernière contraction. Les vecteurs d'intervalles notés $\vec{X}$ et $\vec{Y}$ sont finalement projetés dans le plan des coordonnées articulaires pour revenir au pavé $[\mathbf{q}]$.

Remarque 12. *Les vecteurs notés $\vec{X}$ et $\vec{Y}$ sont en réalité des vecteurs d'intervalles tels que $\vec{X} = [[X_1].....[X_n]]^T$ où chaque couple [X_j], [Y_j] illustre la zone dans laquelle la roue j est censée évoluer (voir figure 4.20).*

Algorithme 4 Contracteur d'enveloppe : $C_s([\vec{q}])$

1: $[\vec{X},\vec{Y}]$=projection($[\vec{q}]$)
2: **pour tout** les produits vectoriels **faire**
3: $[\vec{X}, \vec{Y}] = C_{Vect_i}([\vec{X}, \vec{Y}])$
4: $[\vec{X}, \vec{Y}] = C_{length}([\vec{X}, \vec{Y}])$
5: $[\vec{q}]$ =inverse($[\vec{X}, \vec{Y}]$)

4.3.1 Contracteur de produit vectoriel

Chaque produit vectoriel est utilisé pour contracter les intervalles de $\vec{X}$, et $\vec{Y}$. Pour cela, la propagation suivante est prise en compte ; il s'agit de contracter une inéquation de type $(V_{AB} \wedge V_{BC})_z \geq 0$:

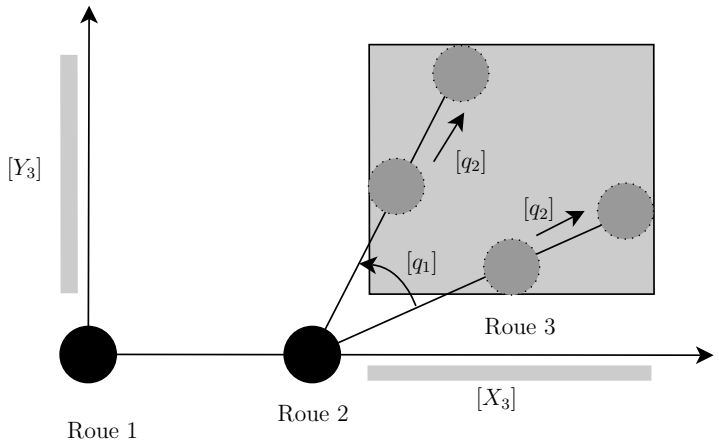


FIG. 4.20 – Illustration de la projection de l'espace des paramètres dans le plan xy.

1. $CVect_i : in : [X_A], [X_B], [X_C], [Y_A], [Y_B], [Y_C]$
2. $[a] = [X_B] - [X_A]$
3. $[b] = [Y_C] - [Y_B]$
4. $[c] = [Y_B] - [Y_A]$
5. $[d] = [X_C] - [X_B]$
6. $[e] = [a] * [b]$
7. $[f] = [c] * [d]$
8. $[z] = [0; +\inf] \cap ([e] - [f])$
9. $[e] = [e] \cap ([z] + [f])$
10. $[f] = [f] \cap (-[z] + [e])$
11. $[c] = [c] \cap ([f]/[d])$
12. $[d] = [d] \cap ([f]/[c])$
13. $[a] = [a] \cap ([e]/[b])$
14. $[b] = [b] \cap ([e]/[a])$
15. $[X_C] = [X_C] \cap ([d] + [X_B])$
16. $[X_B] = [X_B] \cap (-[d] + [X_C])$
17. $[Y_B] = [Y_B] \cap ([c] + [Y_A])$
18. $[Y_A] = [Y_A] \cap (-[c] + [Y_B])$
19. $[Y_C] = [Y_C] \cap ([b] + [Y_B])$
20. $[Y_B] = [Y_B] \cap (-[b] + [Y_C])$
21. $[X_B] = [X_B] \cap ([a] + [X_A])$
22. $[X_A] = [X_A] \cap (-[a] + [X_B])$

Cette procédure est généralisée par l'utilisation de l'algorithme 17 présent dans l'annexe E qui a pour entrées deux vecteurs d'intervalles : $\vec{X}$ et $\vec{Y}$, qui représentent le déplacement (dans le plan x, y) des trois points (où roues) considérés ainsi qu'un intervalle noté *domaine*. Cet intervalle représente le domaine pour lequel le produit vectoriel doit être contracté (typiquement, $[0; +\infty]$ ou $[-\infty; 0]$).

4.3.2 Contracteur dédié à la taille de la chenille

Pour un vecteur de coordonnées articulaires $\mathbf{q}$ donné, qui place le robot dans une position impliquant l'enveloppe convexe H_{123}, la contraction du domaine de travail des roues 1, 2 et 3 est obtenue de la manière suivante :

1. $C_{L_{123}}$:
2. $[a_0] = [X_2] - [X_1]$
3. $[a_1] = [X_3] - [X_2]$
4. $[a_2] = [X_1] - [X_3]$
5. $[b_0] = [Y_2] - [Y_1]$
6. $[b_1] = [Y_3] - [Y_2]$
7. $[b_2] = [Y_1] - [Y_3]$
8. $[c_0] = [a_0]^2$
9. $[c_1] = [a_1]^2$
10. $[c_2] = [a_2]^2$
11. $[d_0] = [b_0]^2$
12. $[d_1] = [b_1]^2$
13. $[d_2] = [b_2]^2$
14. $[e_0] = [c_0] + [d_0]$
15. $[e_1] = [c_1] + [d_1]$
16. $[e_2] = [c_2] + [d_2]$
17. $[f_0] = \sqrt{[e_0]}$
18. $[f_1] = \sqrt{[e_1]}$
19. $[f_2] = \sqrt{[e_2]}$
20. $[l] = [f_0] + [f_1] + [f_2]$
21. $[g] = [g] \cap [-L_{min} + [l]]$
22. $[h] = [h] \cap [L_{max} - [l]]$
23. $[g] = [g] \cap [0; +\inf]$
24. $[h] = [h] \cap [0; +\inf]$
25. $[l] = [l] \cap (-[h] + L_{max})$
26. $[l] = [l] \cap ([g] + L_{min})$
27. $[f_0] = [f_0] \cap ([l] - [f_1] - [f_2])$
28. $[f_1] = [f_1] \cap ([l] - [f_0] - [f_2])$
29. $[f_2] = [f_2] \cap ([l] - [f_1] - [f_0])$
30. $[e_0] = [e_0] \cap ([f_0]^2)$
31. $[e_1] = [e_1] \cap ([f_1]^2)$
32. $[e_2] = [e_2] \cap ([f_2]^2)$
33. $[c_0] = [c_0] \cap ([e_0] - [d_0])$
34. $[c_1] = [c_1] \cap ([e_1] - [d_1])$
35. $[c_2] = [c_2] \cap ([e_2] - [d_2])$
36. $[d_0] = [d_0] \cap ([e_0] - [c_0])$
37. $[d_1] = [d_1] \cap ([e_1] - [c_1])$
38. $[d_2] = [d_2] \cap ([e_2] - [c_2])$
39. $[b_0] = [b_0] \cap \sqrt{[d_0]} \cup [b_0] \cap -\sqrt{[d_0]}$
40. $[b_1] = [b_1] \cap \sqrt{[d_1]} \cup [b_1] \cap -\sqrt{[d_1]}$
41. $[b_2] = [b_2] \cap \sqrt{[d_2]} \cup [b_2] \cap -\sqrt{[d_2]}$
42. $[a_0] = [a_0] \cap \sqrt{[d_0]} \cup [a_0] \cap -\sqrt{[d_0]}$
43. $[a_1] = [a_1] \cap \sqrt{[d_1]} \cup [a_1] \cap -\sqrt{[d_1]}$
44. $[a_2] = [a_2] \cap \sqrt{[d_2]} \cup [a_2] \cap -\sqrt{[d_2]}$
45. $[X_2] = [X_2] \cap ([a_0] + [X_1])$
46. $[X_1] = [X_1] \cap (-[a_0] + [X_2])$
47. $[X_3] = [X_3] \cap ([a_1] + [X_2])$
48. $[X_2] = [X_2] \cap (-[a_1] + [X_3])$
49. $[X_1] = [X_1] \cap ([a_2] + [X_3])$
50. $[X_3] = [X_3] \cap (-[a_2] + [X_1])$
51. $[Y_2] = [Y_2] \cap ([b_0] + [Y_1])$
52. $[Y_1] = [Y_1] \cap (-[b_0] + [Y_2])$
53. $[Y_3] = [Y_3] \cap ([b_1] + [Y_2])$
54. $[Y_2] = [Y_2] \cap (-[b_1] + [Y_3])$
55. $[Y_1] = [Y_1] \cap ([b_2] + [Y_3])$
56. $[Y_3] = [Y_3] \cap (-[b_2] + [Y_1])$

L'algorithme 18 (Annexe E), qui généralise cette propagation de contrainte prend en entrée la liste ordonnée des roues qui composent l'enveloppe convexe (*ordre*), les vecteurs $\vec{X}$ et $\vec{Y}$ représentant les domaines de travail initiaux de chaque roue du robot, les bornes L_{min} et L_{max} ainsi que le sens dans lequel on désire contracter (C_S ou $C_{\bar{S}}$).

4.3.3 Fonction project et inverse

La projection de **q** dans un espace en deux dimensions censé représenter l'espace dans lequel les roues sont autorisées à se déplacer est assez simple. Il suffit en effet de calculer les cordonnées des roues directement à partir des éléments du vecteur **q** comme illustré par l'algorithme 16 (Annexe E).

Une fois cette représentation en deux dimensions contractée, il faut désormais retrouver les contractions que cela implique sur **q**. D'une manière générale, la relation

entre l'intervalle lié à une roue $[X_n]$ et les éléments du vecteur de coordonnées articulaire est donnée par :

$$[X_n] = [X_{n-1}] + [q_i]cos([q_1] + [q_3] + ... + [q_j]) \tag{4.31}$$

Une opération de rétro-propagation permet alors simplement de définir les intervalles liés à $[q_i]$ et $cos([q_1] + [q_3] + ... + [q_j])$, l'opérateur *acos* est ensuite utilisé pour déterminer l'intervalle global assigné à $[q_1] + [q_3] + ... + [q_j]$. Cependant, la périodicité de ce dernier pourra rendre impossible l'extraction de l'intervalle $[q_j]$. En effet, posons :

$$[q_1] = [0, \frac{\pi}{2}] \text{ et } [q_3] = [\frac{\pi}{2}, \pi] \tag{4.32}$$

ainsi, $cos([q_1] + [q_3]) = [-1, 1]$. Or, quel que soit le domaine d'évolution d'une cinquième articulation l'intervalle de l'ensemble sera toujours donné par : $cos([q_1] + [q_3] + [q_5]) = [-1, 1]$. L'évaluation de ce dernier intervalle sera donc impossible via l'opérateur *acos*. Afin de s'affranchir de ce problème, un algorithme de contraction des formules de trigonométrie présentées par l'équation 4.33 est utilisé pour déterminer simplement le vecteur **q** contracté.

$$\begin{array}{l} cos(a+b) = cos(a)cos(b) - sin(a)sin(b) \\ sin(a+b) = sin(a)cos(b) + cos(a)sin(b) \end{array} \tag{4.33}$$

4.4 Un graphe représentant l'espace articulaire contraint

Une fois l'espace articulaire contraint défini, l'étude du graphe qui en résulte donne des informations sur les capacités du robot. En effet, en fonction de l'espace de travail initial du robot (noté D_i) dans la formulation du CSP, il ne sera pas toujours possible de relier toutes les configurations entre elles, et donc de naviguer d'un point de l'espace de configuration du robot à un autre. Pour illustrer ces propos, prenons l'exemple simple, mais facile à représenter dans le plan, d'un robot à deux degrés de liberté (typiquement, le prototype B2P2). La figure 4.21 représente les espaces de travail contraints du même robot pour deux domaines d'évolution des articulations différents ($q_1 \in [-\pi/2, \pi/2]$ et $q_2 \in [16, 20]$ pour le premier, et $q_1 \in [-\pi/2, \pi/2]$, $q_2 \in [15, 20]$ pour le second) obtenus en résolvant deux CSP (respectivement pour les enveloppes H_{123} et H_{132}). Il y apparaît très clairement, deux composantes connexes disjointes. Cette information obtenue grâce à la caractérisation d'ensemble est capitale pour déterminer le domaine d'évolution des articulations du robot et évaluer sa capacité de déformation.

4.4.1 Naviguer de manière garantie entre deux pavés

Lors de la navigation dans le graphe, il est important de garantir la tension des chenilles. En effet, comme l'illustre la figure 4.22 pour passer du centre d'un pavé à un autre, le chemin direct n'est pas toujours un chemin garanti. Or, puisque nous

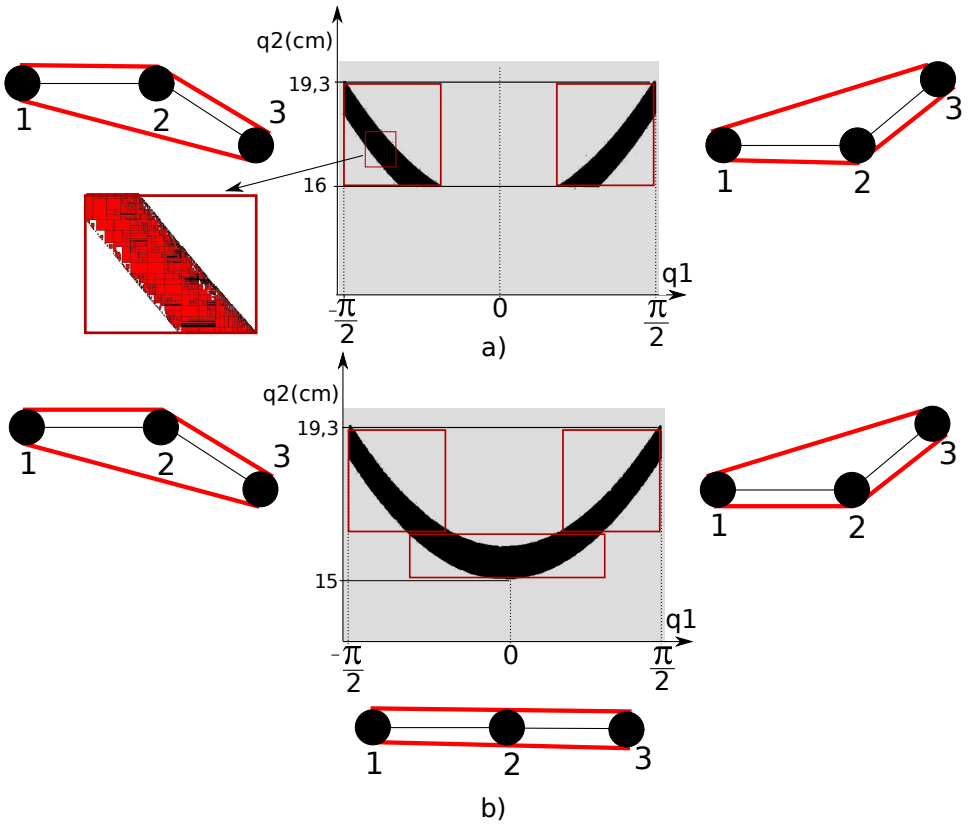


FIG. 4.21 – a) Premier exemple : $q_1 \in [-\pi/2, \pi/2]$, $q_2 \in [16, 20]$, les zones noires représentent les pavages pour lesquels le périmètre de l'enveloppe convexe est tel que $L \in [49; 50]$, b) Second exemple :$q_1 \in [-\pi/2, \pi/2]$, $q_2 \in [15, 20]$, la zone noire représente le pavage pour lequel le périmètre de l'enveloppe convexe est tel que $L \in [49; 50]$.

disposons d'un ensemble de pavés, chaque élément du graphe est un pavé de forme régulière (rectangle en deux dimensions, parallépipède rectangle en trois dimensions etc...). Une droite entre le centre du pavé considéré et n'importe quel élément contenu dans le pavé est alors incluse dans celui-ci. Cette définition implique une méthode simple pour garantir la trajectoire en considérant deux lignes (cf figure 4.22 b)) :

- Une ligne partant du centre du pavé de départ, jusqu'au centre de l'intersection des deux pavés,
- Une ligne partant du centre de l'intersection des deux pavés jusqu'au centre du pavé d'arrivée.

4.4.2 Algorithmes de chemin le plus court

Pour déformer le robot en naviguant d'un élément du graphe à un autre, il est important de déterminer un chemin efficace. A cet effet, différents algorithmes dit

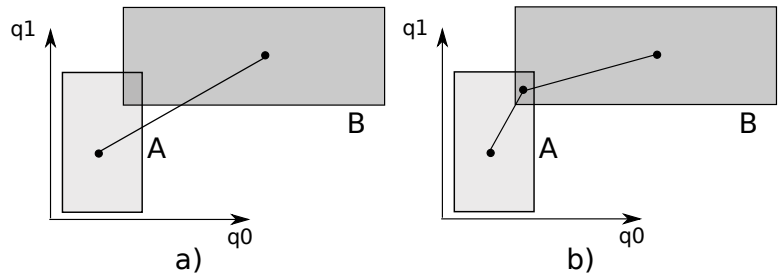


FIG. 4.22 – La trajectoire garantie pour naviguer entre deux pavés passe par l'intersection des pavés.

du "plus court chemin" ont été évalués :

- L'algorithme de Dijkstra,
- L'algorithme A-star.

Notons que le détail de ces procédures est illustré en annexe F. A la différence de Dijkstra qui détermine le chemin le plus court par rapport à un critère donné, l'algorithme A-star ne permet pas d'obtenir à coup sûr le chemin optimal. En effet, au lieu de tester tous les chemins possibles, on se contente ici de déterminer, à chaque itération, la direction à prendre grâce à une fonction coût qui minimise la distance entre la position actuelle et la position d'arrivée (une distance euclidienne qui ne prend donc pas en compte les éventuels obstacles).

Dans la mesure où il n'y a pas d'obstacles dans l'espace de travail contraint, le chemin trouvé via l'algorithme A-star sera toujours assez proche du chemin optimal que nous fournit Dijkstra. La solution sera cependant plus rapide à obtenir, car tout le graphe ne sera pas exploré à la recherche du meilleur chemin.

Remarque 13. *Dans le cas de l'analyse du graphe de l'espace articulaire contraint, la distance entre deux pavés est obtenue via la distance euclidienne entre leurs centres respectifs.*

4.5 Illustration sur un robot à 2 étages

L'exemple illustré ici représente un robot à deux étages ; les espace articulaires des enveloppes H_{1234} et H_{123} ont été caractérisés pour un domaine d'évolution donné par :

$$\mathbf{q} = \begin{bmatrix} [-\pi, \pi] \\ [5, 20] \\ [-\pi, \pi] \\ [5, 20] \end{bmatrix} \tag{4.34}$$

Notons que la longueur B entre la première et la seconde roue est considérée fixe dans cet exemple.

La longueur nominale de la chenille a été fixée à 62 cm, on obtient alors :

$$95\%L_{max} < L_i < L_{max} \Leftrightarrow 59 < L_i < 62 \tag{4.35}$$

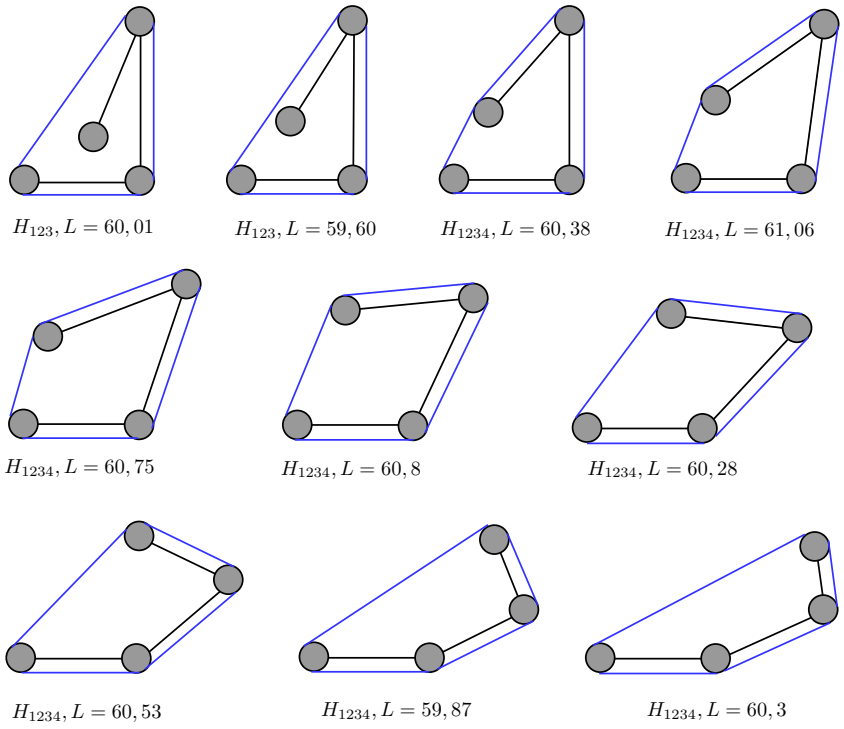


FIG. 4.23 – Illustration de la déformation d'un robot à 2 étages d'une configuration appartenant à H_{123} vers une posture appartenant à H_{1234}

Lors de la la simulation de la déformation via l'exploration du graphe, le simulateur calcule un chemin depuis une configuration appartenant à l'enveloppe convexe de l'espace articulaire étudiée à une autre. Afin de déterminer ces positions, l'espace articulaire est projeté dans deux plans (un pour chaque étage du robot) ; en sélectionnant un pavé dans les différentes projections de chaque étage, il est alors possible d'obtenir un jeu de positions garanties pour le robot étudié.

Une fois les positions de départ et d'arrivée choisies, le parcours est simulé et le robot se déforme en respectant la contrainte liée à la longueur de la chenille bien que la formulation de cette contrainte ne soit pas tout le temps identique.

La figure 4.23 illustre le mouvement d'un VGSTV d'un point appartenant à l'enveloppe H_{123} à un point de l'enveloppe H_{1234}. Une vidéo de cette simulation est disponible sur Internet à l'adresse suivante : `http://www.istia.univ-angers.fr/`

~jeanluc.paillat/Recherche/Recherche_index.html dans la rubrique "Simulations".

Un autre exemple intéressant est illustré par la figure 4.24. En effet, ici pour passer d'une posture de H_{1234} à une posture de H_{1243} le robot doit passer par une posture appartenant à une enveloppe intermédiaire (ici H_{124}). Là encore, une vidéo de la déformation est présente à l'adresse http://www.istia.univ-angers.fr/~jeanluc.paillat/Recherche/Recherche_index.html.

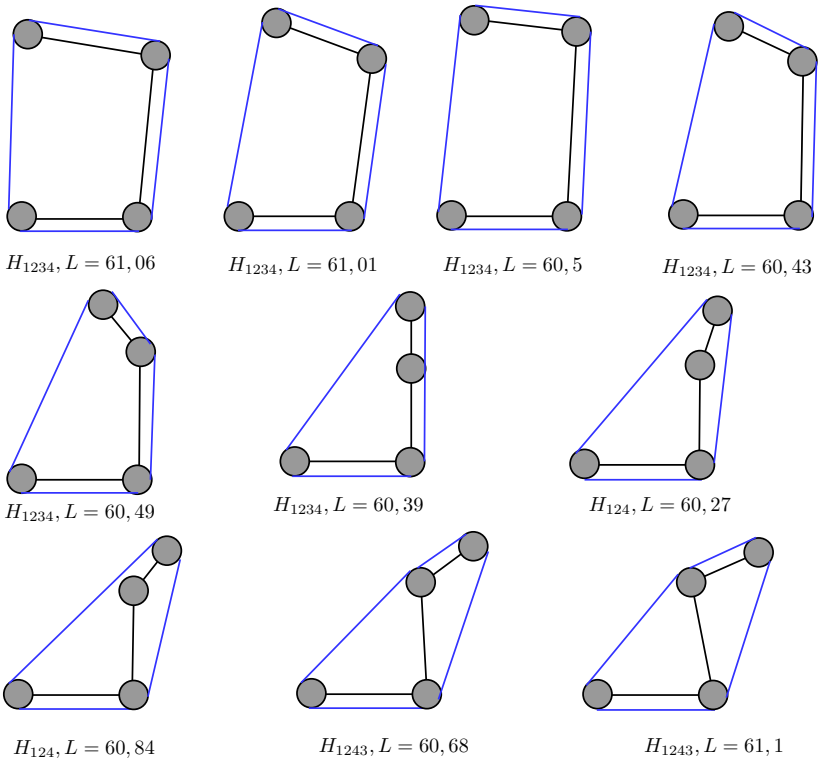


FIG. 4.24 – Illustration de la déformation d'un robot à 2 étages d'une configuration appartenant à H_{1234} vers une posture appartenant à H_{1243} en passant par H_{124}

4.6 Discussions

Nous venons de présenter une méthode permettant de résoudre de manière générale les problèmes liés à la déformation d'un VGSTV à l'intérieur d'une chenille. Il s'agit de déterminer un espace de travail fortement contraint. De ce fait, il existe néanmoins quelques limitations pratiques à la résolution via le calcul ensembliste. En effet, bien que les algorithmes utilisés soient convergeants comme cela a été prouvé dans [L. Jaulin 1993] le temps de calcul ainsi que le nombre de pavés contenus dans le graphe s'avère assez important.

4.6.1 Temps de convergence des méthodes utilisées

Dans l'ensemble, la convergence de SIVIA pour un robot à 2 étages est obtenue après deux heures de calcul sur le PC de test (Pentium 4 dual Core cadencé à 2GHz), et ceci pour l'intégralité des enveloppes. En utilisant des contractions, et donc en traitant les enveloppes une par une, il faut une dizaine de minutes par enveloppe, ce qui globalement revient au même qu'avec SIVIA si on désire les traiter toutes (rappelons qu'un robot à 2 étages présente 12 enveloppes). On peut cependant noter que la diminution du facteur de précision (le paramètre ε utilisé dans les algorithmes 2 et 3) n'augmente de manière exponentielle que la durée de convergence de SIVIA, si bien que la précision maximale obtenue à l'aide des contractions reste meilleure au regard du temps de calcul. En effet, traiter toutes les enveloppes d'un coup implique de déterminer un ensemble où les "frontières" sont nombreuses, et où l'algorithme perd un temps précieux.

4.6.2 Taille du graphe et calcul du chemin le plus court

La taille des graphes obtenus est aussi un élément limitant. Un graphe important implique évidemment un temps de création long, et une navigation plus lente dans le cadre de la recherche de chemins. A titre d'exemple, les graphes utilisés dans les simulations illustrées par les figures 4.23 et 4.24 possèdent respectivement une composante connexe principale de 200000 et 400000 éléments pour des fichiers pesant 55 et 100 Mo. Dans ce contexte, la recherche d'un chemin prend alors en moyenne une dizaine de secondes.

4.6.3 Pistes d'amélioration

Il semble tout de même prématuré de parler d'intégration du graphe dans un système embarqué pour réaliser le contrôle garanti d'un VGSTV. En effet, la taille du graphe et le temps de calcul d'un chemin entre deux postures au sein de celui ci ne fourniraient pas une commande intuitive et suffisamment réactive.

Cependant, le temps de création du graphe pourrait être facilement réduit en le construisant dans le même temps que l'espace articulaire comme c'est le cas dans [Jaulin 2001] où le graphe est actualisé à chaque bissection de manière à toujours

connaître les voisins d'un pavé garanti.
Les arbres d'intervalles, ou "interval trees" peuvent aussi représenter une piste pour diminuer le temps de création du graphe. En effet, il s'agit d'une structure de données introduite dans [de Berg 2000] où le tri et le rangement d'un ensemble d'intervalles sous la forme d'un arbre réduit le temps mis pour trouver les intersections entre pavés.
La mise en forme du graphe pour le rendre facilement exploitable semble donc constituer une perspective majeure pour mener à bien l'intégration de cette méthode de commande dans une plate-forme expérimentale.

Chapitre 5

Commande pour le franchissement autonome d'escalier

Sommaire

Les nombreuses expérimentations menées avec le prototype du LISA ont montré l'importance de la dextérité du pilote lors du mouvement de la partie mobile en cours de franchissement [J. L. Paillat 2009]. Ce retour d'expérience souligne la nécessité d'implémenter un contrôleur de cette partie mobile dans le cadre du franchissement d'escaliers. Cela offrirait une précieuse aide au pilotage et constitue une étape indispensable vers l'autonomie de ce type de robots.
Le contrôleur envisagé dans ce chapitre est basé sur l'apprentissage d'un réseau de neurones et a pour objectif d'automatiser la déformation de la plate-forme sur l'obstacle. Après avoir présenté les méthodes de franchissement autonome développées dans la littérature lors de la dernière décennie, nous illustrerons les principes liés aux réseaux de neurones. La définition d'un contrôleur propre aux VTGVs et son

entraînement en simulation seront ensuite explicités. Enfin, l'évolution du comportement obtenu à différentes étapes de l'apprentissage sera proposée. Les résultats expérimentaux seront finalement illustrés et discutés dans une conclusion.

5.1 Franchissement autonome

Le franchissement d'obstacles tels qu'un escalier à l'aide de véhicules à géométrie variable soulève un certain nombre de problématiques étudiées depuis quelques années. En 2000, Xiong et Matthies [Yalin Xiong 2000] proposaient une méthode de franchissement basée sur un système de vision pour le robot Urban II de IS Robotics (un proche cousin du performant Packbot). Deux paramètres dits critiques y étaient observés et contrôlés :

- L'angle entre l'orientation de l'escalier et le cap du véhicule (noté α sur la figure 5.1),
- La position du véhicule par rapport au centre de l'escalier (rapport $\frac{d_r}{d_l}$ sur la figure 5.1).

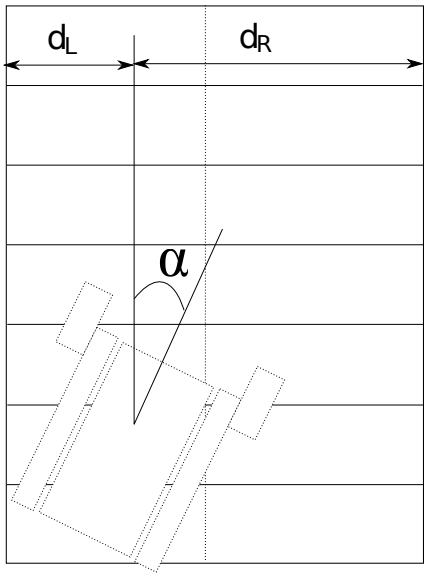


FIG. 5.1 – Contôle de la montée d'un escalier

En effet, le contrôle de ces deux critères permet théoriquement d'éviter au robot de heurter les murs ou encore de se retrouver en position de déséquilibre après avoir pivoté sur le coin d'une marche. A partir de la détection des coins des marches via un dispositif de vision monoculaire, la position et l'orientation du robot sont alors estimées dans l'espace (en fonction de l'inclinaison des marches sur l'image) et l'offset entre les critères α et $\frac{d_r}{d_l}$ estimés et les valeurs désirées est utilisé pour la commande des moteurs du robot.

Cela nécessite cependant une bonne connaissance à priori de l'obstacle à franchir. Des algorithmes de détection de marche sont alors associés à des systèmes de stéréo vision comme c'est le cas dans [Xiaoye Lu 2005]. L'algorithme qui y est présenté basé sur la corrélation entre les informations géométriques et de luminosité issues d'images en 3 dimensions détecte efficacement les marches de l'escalier. La fréquence de détection s'elève à $4H_z$ pour un système Linux embarqué dans un PC cadencé à $1GH_z$. La position relative du robot est alors estimée en fonction du retour vidéo. En 2002, Helnick et al. proposent dans [Daniel M. Helnick Stergios I. Roumeliotis 2002] une technique de détection de marches basée sur la fusion de données visuelles avec les mouvements du robot (typiquement acquis avec un gyroscope 3 axes) via un filtre de Kalman afin d'augmenter la fiabilité et les performances de l'algorithme de franchissement. Il en résulte une méthode éprouvée en 2007 par Mourikis et al. dans [Anastasios I. Mourikis 2007] où une étude complète démontre la fiabilité de cette dernière technique en s'appuyant sur un nombre important de validations expérimentales (environ 300 tests) dans des conditions météorologiques diverses sur des escaliers différents.

Les modèles de robots utilisés dans ces expérimentations sont des robots de type "packbot" équipés de deux flippers. Le mouvement de ces flippers très légers n'influence quasiment pas la position du centre de gravité de ces plate formes. Le prototype B2P2 présenté dans le chapitre 3 possède une différence remarquable avec ce type de robot. En effet, le mouvement de la partie mobile influe de manière importante sur le déplacement du centre de gravité comme le présente la figure 5.2. Cela comporte des risques, car le CoG peut alors facilement se retrouver en dehors du polygone de sustentation et provoquer la chute du robot. D'un autre côté, cela facilite grandement les transferts de masse permettant de passer d'une marche à l'autre. Lors du franchissement d'un escalier, il est donc primordial de se concentrer sur le mouvement de cette partie mobile en plus du contrôle du "cap" (critères α et $\frac{d_r}{d_l}$) du robot. C'est dans cette optique que nous avons développé une méthode basée sur un réseau de neurones afin d'adapter la morphologie de la plate-forme à l'escalier.

Ce chapitre est donc composé de trois parties ; tout d'abord, nous rappelons les définitions relatives aux réseaux de neurones. Ensuite, les détails concernant le réseau utilisé comme contrôleur autonome d'un VGSTV sont illustrés, puis les résultats obtenus qui ont fait l'objet d'une validation expérimentale sont discutés.

5.2 Réseaux de neurones et méthodes d'apprentissage

5.2.1 Les réseaux de neurones

Le premier modèle de neurones à été proposé en 1943 par Mac Cullogh et Pitts, deux bio-physiciens de l'université de Chicago [Cullogh 1943]. Ce premier neurone, connu aussi sous l'appellation de neurone à seuil fonctionne comme un automate, une fonction de transfert lie ses entrées à ses sorties pour comparer ce résultat à une valeur de seuil qui définit la réponse (binaire, 0 ou 1 dans le cas des neurones

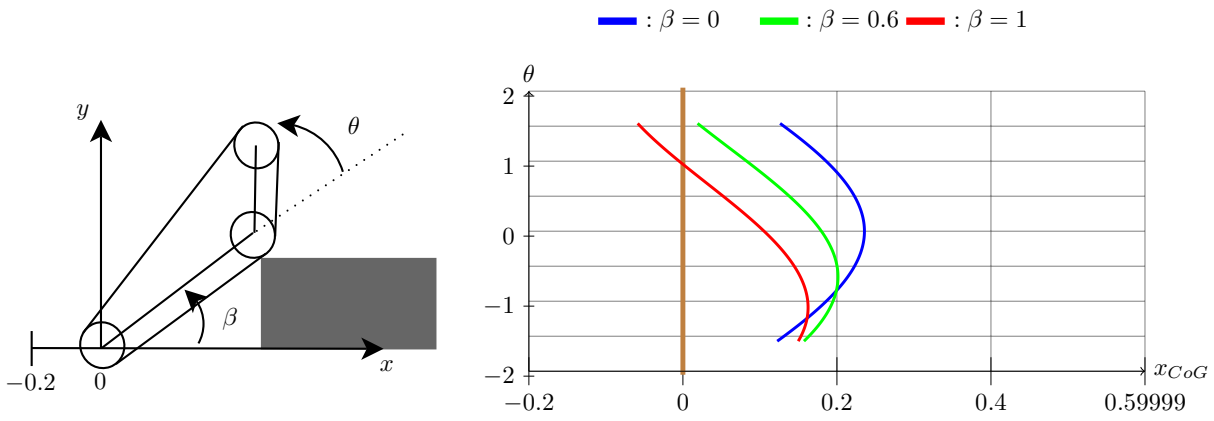


FIG. 5.2 – Représentation de la trajectoire du CoG sur l'axe x pour différentes valeurs de α. On remarque que le projeté au sol du centre de gravité peut dans certaines positions sortir du polygone de sustentation.

formels). L'organisation de ces neurones en réseaux de topologies variées permet alors de réaliser des fonctions logiques et arithmétiques complexes.

Au sein d'un réseau, chaque élément est relié à d'autres par l'intermédiaire de liaisons synaptiques auxquelles sont associées un poids conditionnant l'influence de la sortie d'un neurone sur l'entrée d'un autre. Les travaux du psychologue Donald Hebb à la fin des années quarante introduisent un certain nombre de règles pour adapter la valeur de ces poids synaptiques dans un réseau [Hebb 1949] ; notamment la fameuse règle de Hebb qui stipule que deux neurones dont l'activité est corrélée (actifs en même temps, où excités par le même stimulus) vont voir le poids synaptique qui les relie augmenter.

Après une dizaine d'années de recherche dans ce domaine, la fin des années cinquante voit l'apparition d'un nouveau modèle appelé perceptron par le psychologue Frank Rosenbatt [Rosenblatt 1958]. Ce réseau alors capable d'apprendre à différencier des formes simples, est acceuilli avec enthousiasme, mais, à la fin des années soixante dix, ses limites sont publiées par deux américains, Minsky et Papert [Minsky 1969], notamment son incapacité à résoudre les problèmes non linéaires.

La recherche sur les réseaux de neurones paraissant alors dans une impasse, elle est peu à peu abandonnée. Il faut attendre le début des années quatre vingts pour que Hopfield, un physicien reconnu publie un modèle de réseau de neurones

entièrement connectés et en démontre l'utilité. Dans le même temps, la désillusion engendrée par les approches algorithmiques de l'intelligence artificielle pousse la communauté scientifique à relancer son intérêt pour les réseaux de neurones et à développer des méthodes d'optimisation des réseaux telles que la rétro-propagation du gradient.

Depuis lors, les applications n'ont pas cessées de croître, et il a d'ailleurs été démontré qu'un réseau multi-couches à deux couches basé sur un apprentissage via rétro-propagation introduit par Werbos et popularisé dans [Rumelhart 1986] est capable d'approximer n'importe quelle fonction de $\mathbb{R}^n$ dans $\mathbb{R}^m$.

5.2.2 Notation

Un modèle de neurone artificiel classique est composé de différents éléments comme illustré par la figure 5.3 :

- Une ou plusieurs entrées (x_i) pondérées par des poids synaptiques (w_{ij}),
- un sommateur ($\sum$),
- une fonction de transfert ($\sigma_j()$),
- une sortie (s_j).

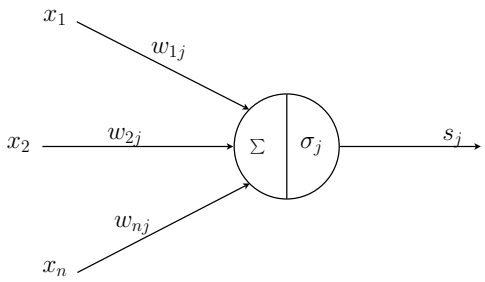


FIG. 5.3 – Un neurone simple

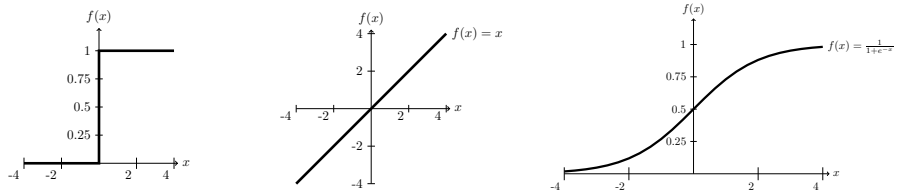
La sortie d'un neurone est ainsi liée à ses entrées par la relation suivante :

$$s = \sigma_j(\sum_{j=1}^{n} x_i.w_{ij}) \tag{5.1}$$

Dans certains cas, un seuil θ est inclus dans la fonction de transfert pour contrôler l'activation du neurone. Typiquement, trois types de fonctions sont utilisées :

- Fonction de type seuil (figure 5.4a)),
- fonction de type linéaire, bornée ou non (figure 5.4b)),
- fonction sigmoïde (figure 5.4c)).

Notons que la fonction sigmoïde est souvent remplacée par une tangente hyperbolique notée $\tanh(kx)$ qui est bornée entre -1 et 1.



(a) Foncion "step" (b) Fonction "kx" (c) Fonction "sigmoïde"

FIG. 5.4 – Les trois types de fonctions classiques utilisées pour un neurone.

Un réseau de neurones est organisé en couches ; on parlera de couches d'entrées (directement reliées à l'environnement), de couches de sorties (directement reliées aux sorties du réseau) et de couches cachées (entre les deux). Les liaisons entre ces couches sont différentes suivant le type de réseau étudié. En effet, dans une structure dite "feedforward", le signal part de la couche d'entrée vers la couche de sortie, à l'inverse, dans un réseau dit "récurent", il existe des liaisons entre les couches supérieures et inférieures (voir figure 5.5).

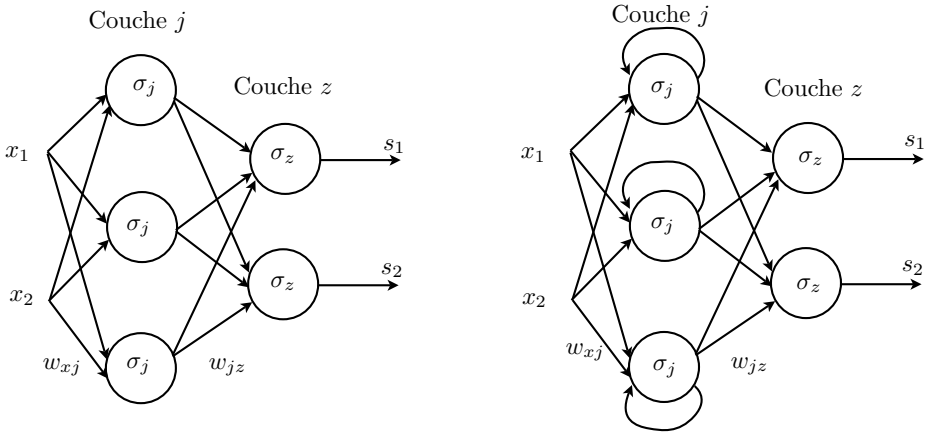


FIG. 5.5 – a) : réseau de neurone "feedforward". b) : réseau récurent

Un réseau de neurones est une méthode efficace pour représenter une fonction mathématique. Bien entendu, le comportement du réseau est déterminé par la valeur des poids synaptiques qui lui sont associés. Quelques méthodes d'apprentissage simples basées sur l'analyse de l'erreur entre la sortie obtenue et la réponse désirée

pour un jeu de paramètres donnés sont utilisées pour ajuster les valeurs des poids synaptiques.

5.2.3 Les procédures d'apprentissage de réseaux neuronaux

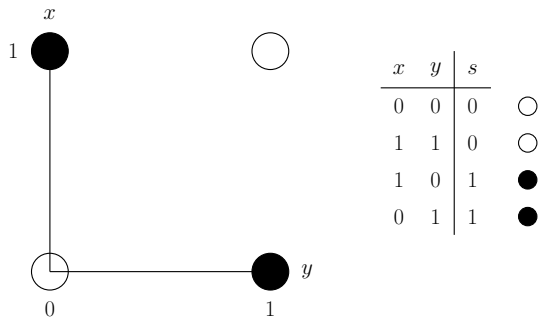


FIG. 5.6 – Résolution de la fonction XOR : il est impossible de trouver une ligne (fonction linéaire) capable de séparer les deux solutions possibles (d'un coté les points noirs, de l'autre les points blancs). L'ajout d'une couche cachée dans le réseau de neurone, est identifié ici à la possibilité d'utiliser plusieurs lignes pour séparer les deux ensembles.

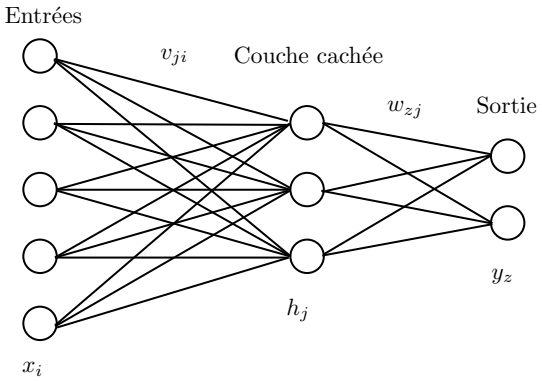


FIG. 5.7 – Notation utilisée dans L'algorithme 6

5.2.3.1 Réseaux sans couche cachée

Ce type de réseau qui ne peut modéliser que des fonctions linéaires est composé d'autant de neurones que de sorties. A partir d'un jeu de données exemple noté

x_i, s_j, respectivement entrées et sorties correspondant à la fonction que l'on veut approximer, on calcule l'erreur de chaque sortie à partir de l'équation suivante :

$$\varepsilon_{ij} = \eta.(s_j - \sigma(\sum_{j=1}^{n} w_{ij}.x_j)).x_i \tag{5.2}$$

où η constitue un paramètre d'apprentissage relatif à la précision voulue. Ensuite, les poids synaptiques sont ajustés en fonction de cette erreur comme l'illustre l'algorithme 5.

Algorithme 5 Apprentissage des réseaux sans couche cachée

Précondition : Initialisation des poids synaptiques avec des nombres aléatoirement tirés

répéter
  pour tout sortie j du réseau **faire**
    Calcul de l'erreur à partir d'un jeu de données exemple x_i, s_j
    $\varepsilon_{ij} = \eta.(s_j - \sigma(\sum_{j=1}^{n} w_{ij}.x_j)).x_i$
  pour tout poids synaptique **faire**
    $w_{ij} = w_{ij} + \varepsilon_{ij}$
jusqu'à ce que l'erreur soit suffisamment faible

Algorithme 6 Apprentissage des réseaux avec couche cachée

Précondition : Initialisation des poids synaptiques avec des nombres aléatoirement tirés

répéter
  pour tout neurone de la couche cachée **faire**
    Calcul de la valeur des neurones pour un jeu de données exemple x_i, s_z
    $h_j = \sigma_j(\sum_{i=1}^{n} v_{ji}.x_i)$
  pour tout neurone de la couche de sortie **faire**
    Calcul de la valeur des sorties
    $y_z = \sigma_z(\sum_{j=1}^{n} w_{zj}.h_j)$
    Puis l'erreur résultante
    $\varepsilon_z = (s_z - y_z).\sigma_z^{'}(\sum_{j=1}^{n} w_{zj}.h_j)$
  pour tout neurone de la couche cachée **faire**
    Calcul de l'erreur résultante
    $\varepsilon_j = \sigma_j^{'}(\sum_{i=1}^{n} v_{ji}.x_i) * \sum_z w_{zj}.\varepsilon_z$
  pour tout poids synaptique **faire**
    $w_{zj} = w_{zj} + \eta\varepsilon_z$
    $v_{ji} = v_{ji} + \eta\varepsilon_j$
jusqu'à ce que l'erreur soit suffisamment faible

Ce type de réseau est cependant incapable de traiter les problèmes non linéairement séparables comme c'est le cas de la fonction logique XOR illustrée par la

figure 5.6. L'ajout d'une couche supplémentaire de neurones est alors indispensable pour que le réseau puisse apprendre ce genre de fonctions non linéaires.

Remarque 14. *Les fonctions de transferts des neurones de la couche cachée doivent cependant être non linéaires. En effet, comme expliqué dans [Nolfi 2000], un réseau multi-couche composé de neurones dont les fonctions de transferts sont linéaires peut toujours être réduit à un réseau sans couches cachées, et donc incapable d'apprendre à résoudre des fonctions non linéaires.*

5.2.3.2 Réseaux avec couche cachée

Une méthode largement utilisée pour entraîner un réseau de neurones à couches cachées à partir d'un jeu d'exemples est basée sur la rétro-propagation du gradient. Cette méthode est presque équivalente à celle étudiée précédemment, à la différence que l'erreur en sortie est propagée à la couche cachée afin de modifier tous les poids synaptiques comme l'illustre l'algorithme 6 (les notations prises pour cette algorithme sont illustrées par la figure 5.7).

Les différentes techniques explicitées dans cette section démontrent l'intérêt et les limites des réseaux de neurones. Elles soulignent aussi l'importance de déterminer le type du réseau à utiliser ainsi que certains paramètres (entrées, fonctions synaptiques) afin d'approcher au mieux la fonction que l'on désire approximer. Le contrôleur utilisé pour le franchissement d'obstacles a donc bien entendu fait l'objet d'une étude présentée dans la section suivante.

5.3 Le contrôleur pour B2P2

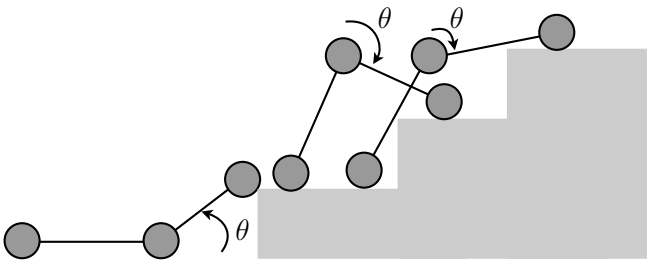


FIG. 5.8 – Illustration du contrôleur désiré : l'angle θ doit être adapté en fonction de la position du robot sur l'escalier

Rappelons tout d'abord la finalité de cette étude ; nous désirons mettre en place une procédure autonome permettant à un VGTV d'adapter sa forme à un obstacle. Les réseaux de neurones semblent capables de résoudre ce genre de problèmes dans le sens où ils permettent d'approximer n'importe quelle fonction de $\mathbb{R}^n$ dans $\mathbb{R}^m$.
La première étape consiste à définir les entrées et sorties de notre système. En l'occurrence, la sortie correspond naturellement à la commande de l'angle d'élévation de la partie mobile noté θ. Les entrées du système doivent cependant être choisies afin d'obtenir un maximum d'informations concernant l'environnement dans lequel évolue le robot. Leur choix fait donc l'objet d'une réflexion particulière.

Notons que les figures qui vont suivre ainsi que le contrôleur qui sera défini ne prennent pas en compte la tension des chenilles. Afin de mettre au point une méthode générique, adaptable à n'importe quel VGTV nous utiliserons un modèle de simulation qui ne prend pas en compte la particularité d'un VGSTV comme B2P2. Les résultats pourront cependant être validés sur notre prototype en maintenant les chenilles détendues ; en d'autres termes, l'interrupteur S présenté dans la chaîne d'asservissement du robot en figure 3.5 dans le chapitre 3 sera maintenu en position 2. En effet, les premières expérimentations ont montré qu'une fois la première marche franchie, les chenilles peuvent être détendue, et les contacts robots/chenille suffisent à éviter tout problème. B2P2 peut alors avoir un comportement proche de celui d'un VGTV classique comme le Packbot.

5.3.1 Les entrées du régulateur

5.3.1.1 La distance obstacle/robot

Comme expliqué dans le chapitre 3, la montée d'un escalier est décomposée en plusieurs étapes :

- l'approche de l'obstacle,
- la montée des marches,
- le franchissement de la dernière marche.

Le contrôleur doit être en mesure de différencier ces différentes étapes grâce aux données qu'il reçoit. La première hypothèse consiste à considérer la distance entre le bout du robot et l'obstacle se trouvant en face de lui. En effet, comme l'illustre la figure 5.9, élever la partie avant lorsqu'un obstacle est présent en face permet de grimper dessus naturellement. A l'inverse, si aucun obstacle n'est présent, il peut convenir d'abaisser l'avant du robot afin de faciliter le transfert de masse. Cette mesure semble donc à première vue fournir des informations relatives au comportement à adopter.

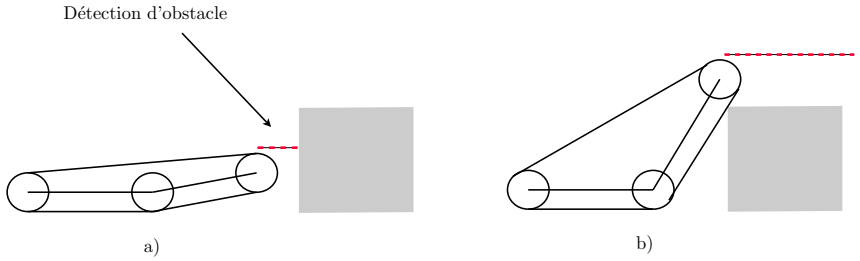


FIG. 5.9 – Intérêt de la mesure de la distance, en présence d'obstacle, la partie avant s'élève. A l'inverse, si rien n'est présent, il peut convenir de l'abaisser pour faciliter le transfert de masse. Notons que sur le robot, ces mesures sont acquises via plusieurs capteurs infrarouges.

5.3.1.2 L'angle du chassis

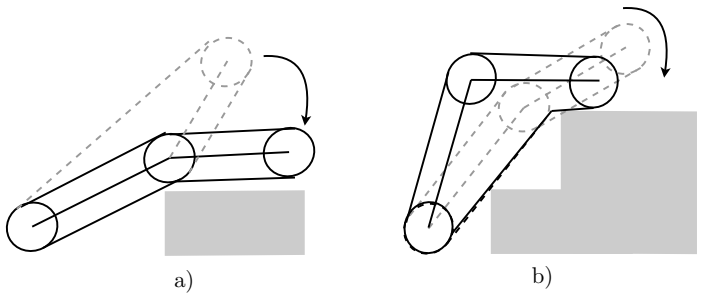


FIG. 5.10 – Intérêt de la mesure de l'angle. L'ajout de cette mesure permet de déterminer à priori si la descente de la partie avant est possible, ou bien si elle compromet l'équilibre du robot.

Notre seconde hypothèse se rapporte à l'angle d'inclinaison du châssis. En effet, suivant les points de contacts entre le robot et le sol, à priori inconnus, descendre la partie avant du robot ne facilitera pas toujours le transfert de masse et peut même amener le véhicule dans une position très dangereuse comme l'illustre la figure 5.10. Il existe donc une limite de cet angle au delà de laquelle il devient difficile de continuer à déformer le robot.
Finalement, ces deux entrées offrent théoriquement la possibilité de différencier chacune des étapes du franchissement d'un escalier avec un VGSTV classique (figure 5.11).

5.3.2 Le réseau de neurones utilisé

A partir des précédentes remarques, nous utilisons un réseau à deux entrées notées l et β (respectivement la longueur entre le robot et la marche et l'angle d'inclinaison du châssis). Ce réseau multi-couches possède une couche cachée composée de plusieurs neurones (cf figure 5.12). Les premières expérimentations que nous avons menées ont permis de fixer le nombre de ces neurones à 5. L'augmentation de la diversité des solutions de franchissement induite par un nombre supérieur de neurones n'étant pas pertinente.

Chaque neurone de la couche cachée possède une fonction de transfert de type $\sigma(x) = tanh(kx)$. Le neurone relié à la sortie est quant à lui muni d'une fonction de transfert de type $\sigma(x) = kx$ seuillée entre $\frac{-\pi}{2}$ et $\frac{+\pi}{2}$.

Le choix d'une fonction linéaire pour le dernier neurone est motivé par la réalité du système. En effet, l'angle de la partie avant du robot est commandé par un

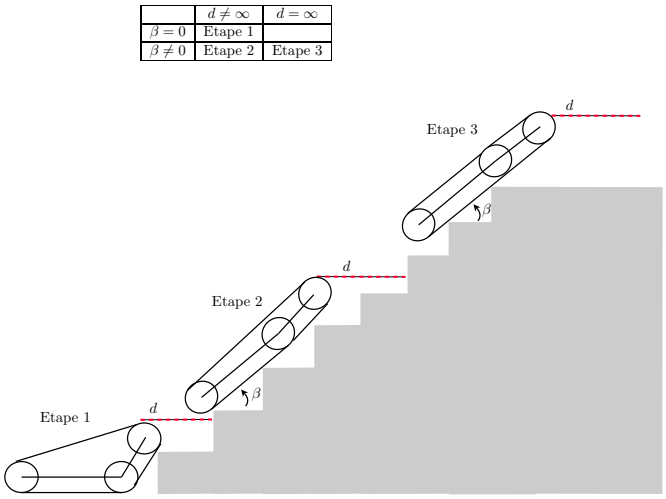


FIG. 5.11 – Les mesures de la distance et de l'angle du châssis semble suffisantes pour distinguer les trois phases du franchissement.

moteur à courant continu, asservi en position. Or, les mouvements rapides de la partie mobile du robot ne sont pas toujours réalisables sur le prototype. En utilisant une fonction de transfert de type $\sigma(x) = kx$ nous nous assurons donc que l'élévation de la partie avant ne se fera pas brusquement même pour une importante variation des entrées.

Les approches classiques d'apprentissage de réseaux de neurones se basent sur un jeu de données (entrées et sorties) qui déterminent le comportement désiré pour le réseau. Dans ce cas précis, il semble cependant assez difficile de mettre en place ce genre d'apprentissage. En effet, le réseau doit juste être capable de déformer le robot au bon moment pour assurer le franchissement de l'escalier. On ne peut pas alors se contenter de réunir un jeu de règles du type :

- Si pas de marche devant, monter la partie avant,
- Si pas de marche mais l'angle du châssis important, diminuer β
- ...

En effet, à aucun moment dans la mise en place des règles nous ne prenons en compte la position des points de contact entre le robot et l'escalier, c'est à dire la taille du polygone de sustentation, ou encore la position du CoG. Une méthode judicieuse pour entraîner notre réseau de manière à ce que ces critères soient intrinsèquement pris en compte consiste à utiliser un algorithme génétique. Cet apprentissage est donc effectué par l'intermédiaire d'un simulateur intégrant les éléments mécaniques de notre plate-forme.

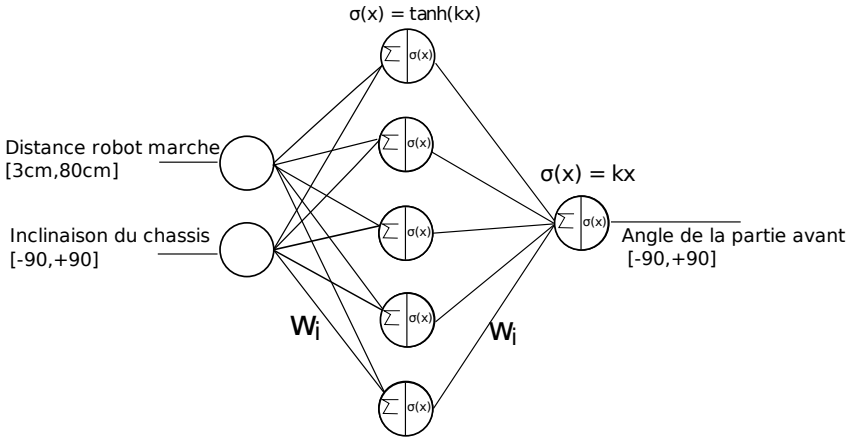


FIG. 5.12 – Réseau de neurones utilisé

5.4 Les algorithmes évolutionnistes

Les algorithmes évolutionnistes sont inspirés de la théorie de l'évolution Darwinienne qui favorise la survie du plus adapté. Les gènes de ce dernier vont ainsi devenir de plus en plus présents dans les générations suivantes et aider l'adaptation d'une espèce entière. Ces techniques dont le principe général est décrit dans [Mitchell 1997] proposent d'évaluer une génération d'individus possédant des caractéristiques génétiques différentes (la première génération étant définie aléatoirement). Au fur et à mesure du temps, les générations suivantes sont définies à partir d'une sélection basée sur les résultats des générations précédentes afin de converger vers l'individu le plus adapté à résoudre le problème.

Dans cette optique, il est souvent nécessaire de formuler le problème afin que chaque solution soit prise en compte comme un individu. Pour illustrer cela, prenons l'exemple de la recherche du juste chiffre. A partir de 4 nombres et des opérateurs numériques $*$, $+$, $-$ et $\div$ il faut trouver la séquence qui s'approche le plus d'un chiffre donné. Un individu sera donc identifié par une séquence de 4 nombres et trois opérateurs. Nous proposons donc le codage binaire illustré par le tableau 5.1.

Ainsi un individu sera caractérisé par 7 gènes composés chacun de 4 bits comme le souligne l'exemple du tableau 5.2.

L'algorithme 7 illustre l'utilisation classique d'un algorithme génétique. L'évaluation d'un ensemble d'individus, ou génération, est contrôlée par une fonction coût f qui détermine l'adaptabilité de l'individu. Dans le problème considéré ici, la différence entre le résultat donné par l'évaluation d'un individu et le chiffre cible confère à chacun une note. Cette note est utilisée pour définir la prochaine génération à partir d'une reproduction de chacun des gènes basée sur un système semblable à une roulette de casino (figure 5.13). Les gènes des individus les plus

Caractère	Codage
0	0000
1	0001
...	...
$*$	1010
$+$	1011
$-$	1100
$\div$	1101

TAB. 5.1 – Codage des différents gènes constituant une solution

1	+	9	*	8	-	4
0001	1011	1001	1010	1000	1100	0100

TAB. 5.2 – Codage d'un individu/solution

adaptés occupent une surface plus grande sur la roulette et ont donc plus de chances d'être reproduit. Avant d'évaluer cette nouvelle génération, des croisements et des mutations sont opérés sur un pourcentage d'individus (les figures 5.14 et 5.15 illustrent ces phénomènes). Cela a pour objectif d'introduire de nouvelles solutions possibles et d'éviter de converger vers un minimum local.

5.5 Entrainement du réseau de neurone

5.5.1 Notation

Dans le cadre du contrôle de B2P2, chaque individu doit représenter un nouveau comportement pour le réseau de neurones, les caractéristiques génétiques de ce dernier sont donc liées avec les poids synaptiques du réseau. Par convention, ces poids sont identifiés par la lettre w. Notons que la première génération est constituée de flottants tirés aléatoirement entre -1 et 1. Chaque individu est donc identifié par une suite de 15 gènes comme l'illustre le tableau 5.3.

5.5.2 Évaluation par simulation

Génération après génération, chaque individu est évalué par l'intermédiaire d'un simulateur de franchissement d'escalier. Il s'agit d'un simulateur quasi-statique

w_1	w_2	w_3	...	w_{14}	w_{15}

TAB. 5.3 – Codage d'un individu pour l'entraînement du contrôleur de B2P2.

Algorithme 7 Principe des algorithmes évolutionistes

1: Initialisation de la population première : chaque gène h_i est tiré aléatoirement pour chaque individu p_j
2: **répéter**
3: **pour tout** individu p_j **faire**
4: évaluation de l'individu :$f_j = f(p_j)$
5: Créer la génération suivante en sélectionnant statistiquement les gènes à reproduire ; la probabilité qu'un individu soit utilisé est donnée par
6: $\frac{f_j}{\sum_j^n f_j}$
7: Réaliser des croisements
8: Réaliser des mutations
9: **jusqu'à** Nombre de génération trop important où résultat trouvé

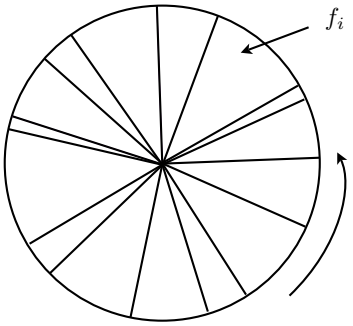


FIG. 5.13 – Reproduction sélective. On associe ce type de reproduction au fonctionnement d'une roulette de casino. On crée pour chaque gène, une roulette où la surface de chaque portion du disque est liée à la note obtenue par chaque individu. On tourne chaque roulette une fois pour déterminer les gènes qui vont composer le nouvel individu.

prenant en compte le modèle géométrique de B2P2 basé sur la prise en compte du centre de gravité comme critère d'équilibre. Les premières expérimentations menées avec notre plate forme ont démontré que le critère de stabilité statique était suffisant pour contrôler l'équilibre du robot (chapitre 3) ; le simulateur utilisé ne prend donc pas en compte les aspects dynamiques du système et se contente d'étudier à chaque pas la position du robot sur l'escalier dans le plan, puis la position du centre de gravité pour déterminer si B2P2 est capable de continuer le franchissement ou non.

Un individu est évalué grâce à une fonction "performance" notée f (pour l'anglais "fitness") qui répond à plusieurs critères. Au fil des différents expérimentations effectuées, il s'est avéré que des fonctions simples comme :

$$f = k * n \tag{5.3}$$

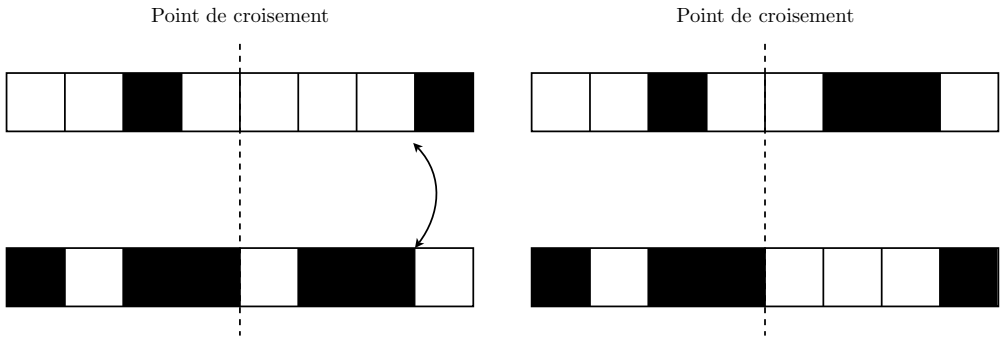


FIG. 5.14 – Croisements. On choisit tout d'abord un point de croisement, puis deux portions de gènes de deux individus sont interverties.

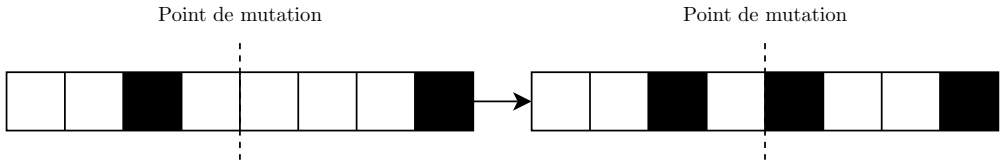


FIG. 5.15 – Mutations. On choisit un point de mutation, et le gène situé à cet emplacement est tout simplement modifié.

où n représente le nombre de marches franchies et k, un facteur d'échelle convergeaient vers une solution. Cependant, le réseau ainsi obtenu offre un comportement très brusque où les mouvements de la partie mobile sont très importants. Il en résulte alors un manque de souplesse flagrant qui rend la méthode difficile à exploiter sur notre plate-forme expérimentale. La fonction "fitness" que nous présentons ci-après n'est donc pas une solution unique, mais correspond à ce qui a donné les meilleurs résultats lors des différentes phases de l'apprentissage.

Tout d'abord, nous avons défini plusieurs critères ; autrement dit, la meilleure fonction "fitness" doit :

- minimiser l'énergie dépensée,
- minimiser les positions dangereuses,
- donner les transferts de masse les plus souples possibles,
- franchir le plus de marches possibles.

Notons β_{max} l'angle β maximum pris par le châssis lors du franchissement, $\dot{\beta}_{max}$ la vitesse maximale de ce mouvement (représentative de la souplesse des transferts de masse), $\bar{\dot{\theta}}$ le déplacement moyen de la partie avant, et x la position du robot à la fin de l'évaluation. Ainsi, f est donné par la relation suivante :

$$f = \overbrace{\frac{x \times 100}{x_{max}}}^{\text{nombre de marches franchies}} + \overbrace{\frac{(Seuil_\beta - \beta_{max}) \times 100}{Seuil_\beta}}^{\text{bonus si } \beta_{max} < Seuil_\beta\text{, malus sinon}} + \underbrace{\frac{(Seuil_{\dot\beta} - \dot\beta_{max}) \times 100}{Seuil_{\dot\beta}}}_{\text{bonus si } \dot\beta_{max} < Seuil_{\dot\beta}\text{, malus sinon}} - \underbrace{k \times \bar{\dot\theta}}_{\text{minimise l'énergie}} \tag{5.4}$$

Bien entendu, les termes $Seuil_\beta$, $Seuil_{\dot\beta}$ et k sont fixés au début de l'entraînement et dépendent du nombre et de la taille des marches.

5.5.3 Reproduction sélective

A l'issue de l'évaluation d'une génération complète de 100 individus, un tableau est créé pour chaque gène. Ces 15 tableaux sont remplis avec les valeurs testées au travers de la génération précédente ; cependant, elles sont répétées un certain nombre de fois en fonction de la note qu'a reçu l'individu à qui appartient le gène en question. Ainsi, en sélectionnant aléatoirement 100 15-uplets de gènes dans ces tableaux, on créé une nouvelle génération censée être plus performante que la précédente car issue de la sélection du meilleur patrimoine génétique. L'algorithme 8 est alors utilisé à cet effet, "population" y représente une matrice contenant chaque génotype de chaque individu, et f_z, la note obtenue par l'individu z. Notons que le meilleur individu de chaque génération est conservé tel quel pour s'assurer de sa pérennité.

Algorithme 8 Reproduction séléctive

```
1: pour tout individu z faire
2:   pour tout gène j faire
3:     pour i = 1 à f_z faire
4:       tabGenes[j].ajouter(Population[z][j])
5: pour tout individu z faire
6:   pour tout gène j faire
7:     On choisit aléatoirement un élément du tableau tabGenes[j]
8:     NouvellePopulation[z][j]=tabGenes[j][rand()]
```

5.5.4 Mutations

Si on se contente d'une reproduction sélective, l'algorithme converge assurément vers une solution. Cependant, cette solution ne sera qu'une combinaison des poids synaptiques tirés aléatoirement à l'initialisation. En effet, il se peut que l'individu le plus adapté possède des gènes différents non tirés à l'initialisation. C'est dans cette optique que des opérations de mutation sont effectués sur 10% de la nouvelle

population. A l'instar de ce qui a été présenté en section 5.4, l'élément situé sur le point de mutation choisi arbitrairement est muté en modifiant simplement sa valeur par un nouveau tirage aléatoire.

5.6 Résultats

L'apprentissage a été effectué pour un escalier contenant 3 types de marches différentes afin de définir un réseau qui puisse s'adapter à plusieurs situations. L'escalier de test présenté en figure 5.16 est ainsi composé de 3 "petites" marches (10 cm de haut pour 30 de long), suivies de 3 "grandes" marches (20 cm de haut pour 30 de long) et se termine avec 3 marches "moyennes" (15 cm de haut pour 25 de long).

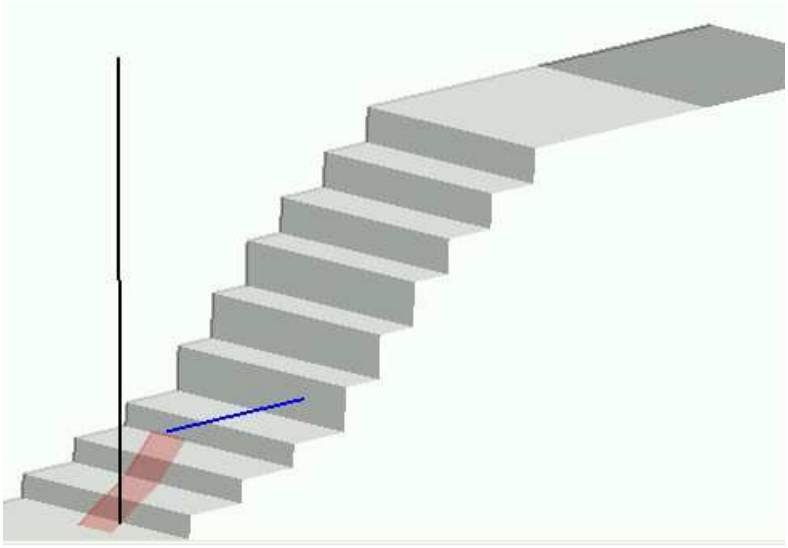


FIG. 5.16 – Escalier utilisé pour l'apprentissage.

5.6.1 Début de l'apprentissage

Dans les premiers temps, le robot adopte une stratégie qui consiste à élever la partie avant en présence d'un obstacle, puis à la rabaisser une fois la première marche franchie comme le présente les figures 5.17a) et b). Cependant, les mouvements effectués alors par la partie avant restent très brusques et la troisième roue se retrouve trop basse pour permettre la montée de la seconde marche (figures 5.17c) et d)).

A partir de quelques générations les mouvements se font plus souples, et leur amplitude se trouve aussi réduite. Ainsi, le robot passe facilement les premières marches à la manière de ce qui est représenté par la figure 5.18a). Cependant, ces premiers résultats concluants ne sont pas pour autant définitifs. En effet, le robot se

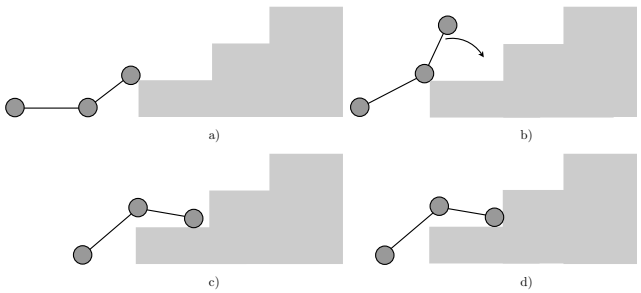


FIG. 5.17 – Comportement du robot lors des premières générations

trouve dans une impasse dès que la taille des marches augmente. A ce moment là, la troisième roue devrait être remontée pour accrocher le nez de la prochaine marche, or il reste coincé comme illustré par la figure 5.18b).

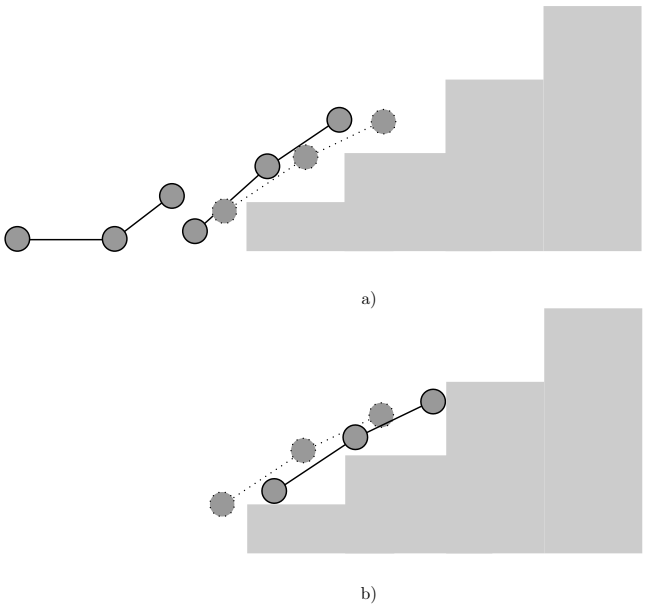


FIG. 5.18 – Au bout de quelques générations, le comportement est plus souple

5.6.2 Convergence de l'apprentissage

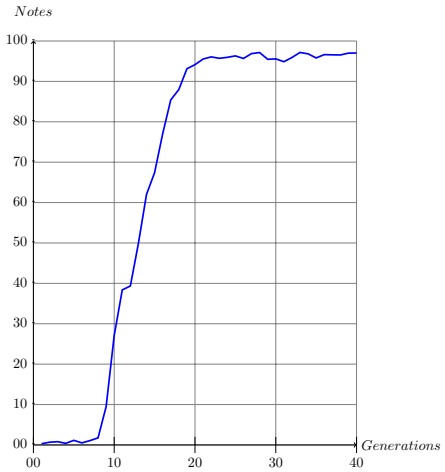


FIG. 5.19 – Moyenne des évaluations (via la fonction "performance" f) obtenues à chaque génération.

L'algorithme finit tout de même par converger au bout d'une vingtaine de générations. L'évolution de la moyenne des notes obtenues à chaque génération est présentée par la figure 5.19. Le comportement du robot est alors suffisamment souple pour ne pas se retrouver dans des situations dangereuses, et l'amplitude des mouvements s'adapte à la taille des marches pour franchir l'intégralité de l'escalier (figure 5.20). Une vidéo de la simulation du meilleur individu obtenu à l'issue des quarante générations est disponible à l'adresse : `http://www.istia.univ-angers.fr/~jeanluc.paillat/Recherche/Franchissement.html`. La figure 5.19 résume les résultats obtenus.

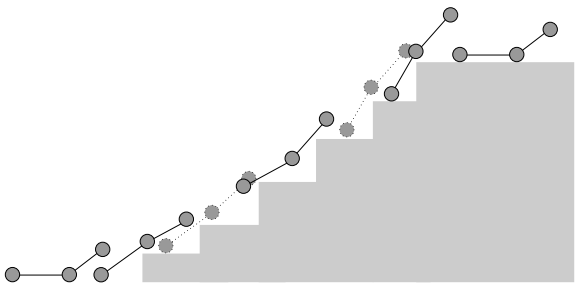


FIG. 5.20 – Comportement définitif du robot.

5.6.3 Résultats de simulation

Le simulateur a ensuite été utilisé pour évaluer l'efficacité du réseau de neurones sur différents types d'escaliers. En effet, l'apprentissage a été réalisé sur un escalier contenant trois types de marches différents, il est donc théoriquement capable d'aider au franchissement d'une gamme d'obstacles divers. La figure 5.21 présente le résultat des tests effectués à travers la simulation ; tous les escaliers que B2P2 peut franchir lorsqu'il est télé-opéré par un pilote aguerri ont été testés. Notons que des vidéos relatives à différents franchissements (les ecaliers encadrés en bleus sur la figure 5.21) sont disponibles à l'adresse : `http://www.istia.univ-angers.fr/~jeanluc.paillat/Recherche/Franchissement.html`. L'analyse de ces conclusions laisse à penser que les paramètres synaptiques obtenus offrent un contrôleur efficace capable de s'adapter à presque tous les types de marche que la taille de B2P2 lui permet de franchir.

Hauteur de marche / Longueur de marche	10 cm	15 cm	20 cm	25 cm
15 cm	*OK*	*OK*	*OK*	*NOK*
20 cm	*OK*	*OK*	*OK*	*OK*
25 cm	*OK*	*OK*	*OK*	*OK*
30 cm	*OK*	*OK*	*OK*	*OK*

FIG. 5.21 – Simulation du réseau sur différents escaliers ; les zones encadrées en bleu indiquent un type de marche présent sur l'escalier d'entraînement.

5.7 Validation Expérimentale

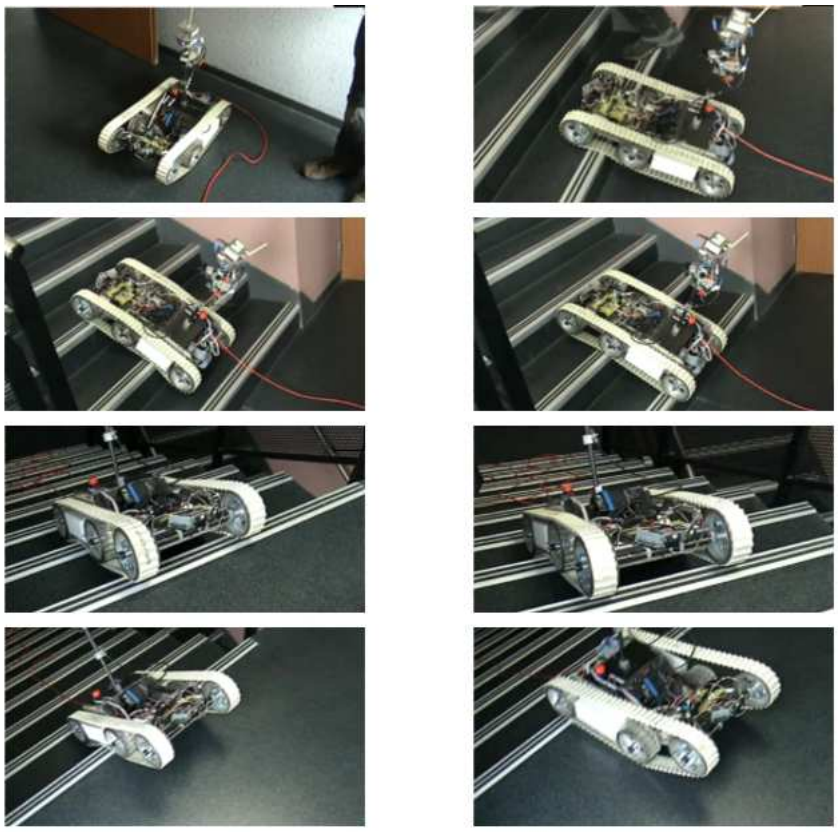


FIG. 5.22 – Validation expérimentale sur notre plate-forme.

Le contrôleur obtenu a bien entendu fait l'objet d'une validation expérimentale disponible en vidéo à l'adresse `http://www.istia.univ-angers.fr/~jeanluc.paillat/Recherche/Franchissement.html` et illustrée par la figure 5.22. Sur le prototype, la distance entre le nez du robot et la marche est obtenue à l'aide de quatre télémètres infra rouges SHARP possédant des caractéristiques différentes montés sur un support mobile (mesure comprise entre 3 et 80 cm). La position du support est asservie par la valeur de l'inclinomètre γ monté sur la partie avant du robot. L'angle d'élévation de la partie avant θ est obtenu par l'intermédiaire des codeurs du moteur dédié à sa rotation. Ainsi, l'angle du châssis β est donné par la relation suivante :

$$\beta = \gamma - \theta \tag{5.5}$$

Il s'agissait ici de franchir un escalier intérieur proposant 10 marches de 20 cm de haut et de 25 cm de long. Lors de ces tests, B2P2 est complètement autonome ; afin de conserver le "cap" lors du franchissement, deux capteurs de distance latéraux sont

utilisés et garantissent le parallélisme entre les marches et le prototype. Finalement, le contrôle de la partie mobile du VGSTV offre une adaptabilité suffisante pour permettre un franchissement efficace au regard du peu d'informations disponibles sur l'environnement.

5.8 Conclusion

Dans ce chapitre une méthode simple pour adapter la forme d'un VGSTV à un escalier a été présentée et validée. Dans la littérature, il existe de nombreuses méthodes de franchissement d'escalier à l'aide d'un VGSTV. Elle proposent de contrôler les moteurs dédiés à la propulsion de manière à garantir le "cap" du robot sur l'obstacle. Ici nous nous sommes plutôt intéressés à ce qui fait l'intérêt de ces plateformes, autrement dit, leur faculté à se déformer. Ainsi, le réseau de neurones issu de ces recherches ne constitue sûrement pas une alternative à ces méthodes, mais leur utilisation conjointe pourrait augmenter la capacité de franchissement de certains prototypes. De plus, au regard du peu d'informations nécessaires (β et l) au franchissement, il semble qu'un réseau de neurones comme celui présenté ici constitue une approche simple pour offrir une aide au pilotage efficace lors de franchissement d'escaliers avec un VGSTV tel que B2P2. En effet, dans ce cadre, le pilote s'affranchit alors de la commande de la partie mobile du robot et se contente de commander rigoureusement les moteurs de propulsion.

Chapitre 6

Conclusion générale

Ces travaux de thèse portent sur l'étude généralisée de la robotique mobile, plus particulièrement les véhicules à géométrie variable qui offrent des performances étonnamment efficaces lors de missions où le franchissement d'obstacles est primordial (déminage, opérations de sauvetage). La robotique mobile est illustrée dans un premier chapitre, puis un prototype innovant développé au LISA depuis 4 ans est présenté et analysé dans le cadre de franchissement d'obstacles divers. Ensuite, deux axes de travail sont développés, respectivement, le contrôle de la déformation d'un robot à n articulations dans une chenille de taille fixe, puis le développement d'une méthode de franchissement autonome d'escalier basée sur un réseau de neurones. Avant de conclure sur les perspectives de l'ensemble de ces travaux, nous proposons de dresser un rapide bilan de chacun des chapitres.

Le premier chapitre présente la robotique mobile de ses débuts jusqu'à nos jours. Les problèmes auxquels les robots mobiles doivent faire face aujourd'hui y sont présentés, ainsi qu'une classification des grandes familles de robots mobiles que l'on retrouve dans la littérature. Ces véhicules doivent être capables de franchir un certain nombre d'obstacles de toute nature, on retrouve donc différents types d'architectures aux propriétés diverses. Cette introduction permet notamment de définir le concept de VGSTV (Véhicules à Géométrie Variable munis d'une seule Chenille) et de comparer les performances théoriques avec d'autres types d'engins. Il semble d'ailleurs qu'aujourd'hui, il s'agisse du compromis idéal entre capacité de franchissement, fiabilité et télé-opération.

Le second chapitre est consacré à la présentation détaillée du B2P2, un prototype de VGSTV développé dans le cadre de cette thèse qui intègre un dispositif original de contrôle de tension de chenille actif. Ce dispositif y est présenté en détails, puis le comportement de B2P2 est analysé lors de plusieurs franchissements d'obstacles afin de quantifier l'intérêt du système. Les premières conclusions indiquent que l'utilisation de ces nouvelles techniques réduit par dix l'accélération angulaire du châssis du robot lors des transferts de masse. En pratique, cela augmente considérablement la souplesse du franchissement. Les modèles géométriques et dynamiques de la plate-forme sont aussi détaillés afin de mener une expérimentation concernant l'importance des effets de la dynamique sur B2P2. Il semble par ailleurs que ces effets soient négligeables, ce qui pousse à utiliser le centre de gravité du robot (autrement dit, un critère purement statique) pour assurer la stabilité lors du franchissement d'obstacles. Ce chapitre soulève cependant un problème

fondamental important. En effet, B2P2 disposant de 2 degrés de liberté, il existe une fonction qui lie leur position à la taille de la chenille, c'est d'ailleurs ce qui est utilisé comme base du contrôleur. Cependant, B2P2 constitue un cas particulier car dès lors que l'on ajoute des articulations au robot, une méthode plus générique doit être utilisée pour assurer son contrôle.

Le troisième chapitre propose un développement de cette problématique en introduisant une généralisation du contrôle d'un VGSTV à n degrés de liberté. Dans un premier temps les problèmes liés à la multitude de configurations qu'un robot de ce type peut adopter sont présentés. En effet, suivant le nombre de roues en contact avec la chenille, la formulation de la fonction qui lie le périmètre de l'enveloppe convexe à la position des articulations est différente. Il existe ainsi un ensemble de contraintes qui doivent être respectées pour maintenir la tension de la chenille lors de la déformation du robot correspondant à chaque posture envisageable. Les équations qui les composent sont directement inspirés de l'algorithme dit "parcours de Graham" utilisé pour identifier l'enveloppe convexe d'un ensemble de points (les détails concernant cette procédure sont présentés dans l'annexe D). Ensuite, une méthode de résolution de ces problèmes de satisfaction de contraintes basée sur l'analyse par intervalles et l'inversion ensembliste est présentée. Les résultats obtenus grâce à cette méthode nous permettent aujourd'hui de simuler le comportement d'une plate-forme mobile munie de deux étages, ou quatre degrés de liberté. L'intégration de ces résultats dans une plate-forme mobile constitue une perspective majeure pour la continuité de ces travaux. Il faudra cependant travailler autour de la mise en forme des données obtenues pour permettre une navigation rapide dans le graphe et un contrôle efficace.

Enfin, le dernier chapitre illustre un second axe de de ces travaux de thèse. Il s'agit d'un approfondissement des techniques existantes en matière de franchissement autonome. En effet, depuis le début de cette thèse, nous nous sommes efforcés de considérer la mobilité accrue apportée par les véhicules à géométrie variable. Il nous semblait donc naturel que ces propriétés soient prises en compte lors de manoeuvres autonomes. Les précédentes recherches en la matière se concentrent sur des méthodes contrôlant le cap du véhicule sur l'escalier. Ici, nous désirions développer un contrôleur de la partie mobile de notre prototype afin qu'en plus de conserver son cap sur l'escalier, il s'adapte aux marches en modifiant la position de son centre de gravité. Finalement, il s'avère que peu d'informations sur l'environnement sont nécessaires (ici l'angle d'inclinaison du châssis et la distance robot/obstacle) pour réaliser ce contrôle via un réseau de neurones (entraîné par un algorithme génétique). La validation expérimentale des résultats de simulation a permis de vérifier l'efficacité de cette solution pouvant s'adapter à beaucoup de plate-formes et être utilisée de concert avec les techniques existantes. Il serait cependant intéressant de quantifier son impact sur différentes plate-formes et escaliers.

En parallèle à ces aspects fondamentaux, un nouveau prototype de robot ap-

FIG. 6.1 – A gauche, un module seul. A droite, un exemple de configuration.

partenant à la famille des robots reconfigurables a été développé (figure 6.1). Un système de fixation inspiré de ce que l'on peut trouver dans la littérature (dans [Ion Lungu 2008] notamment) est actuellement en test afin de fournir au laboratoire (figure 6.2) une nouvelle plate-forme d'expérimentation constituée de plusieurs modules mobiles les uns par rapport aux autres. Ce nouveau prototype pourra permettre de réaliser un véhicule à géométrie variable à plusieurs étages sur lequel les résultats du chapitre 4 pourraient être embarqués. En outre, à la vue de l'efficacité du réseau de neurones pour adapter la forme d'un VGTV à l'obstacle dans le chapitre 5, une généralisation de la méthode pour un robot à n degrés de liberté pourrait être envisagée. Cela constituerait un axe de recherche intéressant pour profiter au mieux des possibilités offertes par cette nouvelle plate-forme.

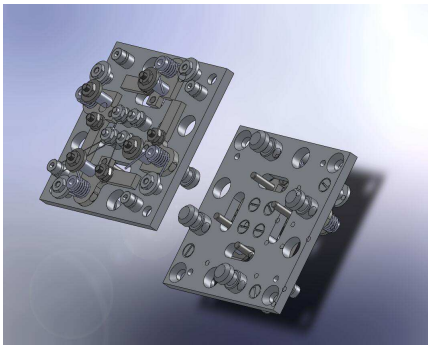


FIG. 6.2 – Système de fixation permettant aux robots de se reconfigurer.

Annexe A

Modélisation géométrique via la méthode de Denavitt-Hartenberg

A.1 Paramètres de Denavitt-Hartenberg

La définition du modèle géométrique par la convention de Denavitt-Hartenberg offre la possibilité de déterminer un jeu de matrices de transformations entre les différentes articulations d'un robot à partir de quelques paramètres précis. Notons R_i le repère attaché à l'articulation i, et $T_{i-1,i}$ une matrice de transport entre les repères $i-1$ et i. La position de l'organe terminal d'un robot peut alors être exprimée dans n'importe quel repère, et notamment dans le repère général par l'équation :

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} = T_{0,1} \times T_{1,2} \times ... \times T_{n-1,n} \times \begin{bmatrix} x_n \\ y_n \\ z_n \\ 1 \end{bmatrix} \tag{A.1}$$

Avant de déterminer les paramètres relatifs aux modèles, quelques règles de notation doivent être respectées. Ainsi, les repères R_i sont déterminés de la manière suivante :

- L'axe $\vec{z_i}$ est porté par l'axe de rotation ou de translation de l'articulation i,
- L'axe $\vec{x_i}$ est porté par la perpendiculaire commune aux axes $\vec{z_i}$ et $\vec{z_{i+1}}$.

Grâce à cette convention, le passage entre le repère $i-1$ et le repère i est conditionné par quatre déplacements :

- Une rotation notée α_i entre les axes $\vec{z_{i-1}}$ et $\vec{z_i}$ autour de l'axe $\vec{x_{j-1}}$,
- une translation notée d_i entre $\vec{z_{i-1}}$ et $\vec{z_i}$ mesurée le long de l'axe $\vec{x_{i-1}}$,
- une rotation notée θ_i entre les axes $\vec{x_{i-1}}$ et $\vec{x_i}$ autour de l'axe $\vec{z_i}$,
- une translation notée r_i entre les axes $\vec{x_{i-1}}$ et $\vec{x_i}$ mesurée le long de l'axe $\vec{z_i}$.

La figure A.1 illustre le placement des différents repère. On y distingue aussi les quatre déplacements relatifs au changement de repère.

Ainsi, quelle que soit l'architecture du robot, le transfert entre le repère $i-1$ et

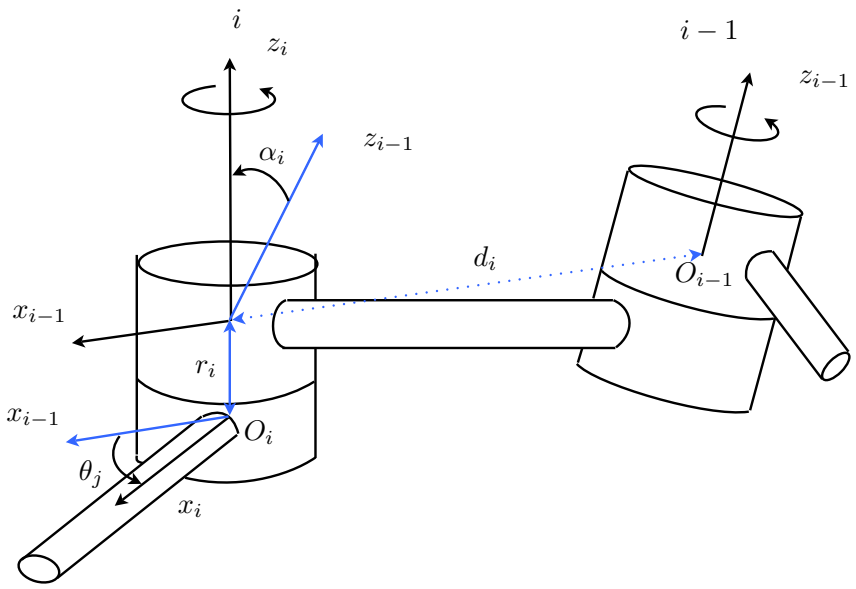


FIG. A.1 – Illustration des paramètres de Denavitt-Hartenberg sur une chaîne cinématique donnée

i sera toujours donné par :

$$
\begin{aligned}
T_{i-1,i} &= Rot(\vec{x_{i-1}}, \alpha_i) \times Trans(\vec{x_{j-1}}, d_j) \times Rot(\vec{z_i}, \theta_i) \times Trans(\vec{z_i}, r_i) \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha_i) & -sin(\alpha_i) & 0 \\ 0 & sin(\alpha_i) & cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & d_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) & 0 & 0 \\ sin(\theta_i) & cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) & 0 & d_i \\ cos(\alpha_i)sin(\theta_i) & cos(\alpha_i)cos(\theta_i) & -sin(\alpha_i) & -r_j sin(\alpha_i) \\ sin(\alpha_i)sin(\theta_i) & sin(\alpha_i)cos(\theta_i) & cos(\alpha_i) & r_j cos(\alpha_i) \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{A.2}
$$

A.1.1 Exemple pour un robot manipulateur à 4 degrés de liberté

Prenons le robot manipulateur présenté par la figure A.2 sur laquelle les repères ont été placés suivant la convention de Denavitt-Hartenberg. Les paramètres de transfert sont alors faciles à déterminer et ont été listés dans le tableau A.1.

Le modèle géométrique de la plate-forme est alors obtenu par la formulation de

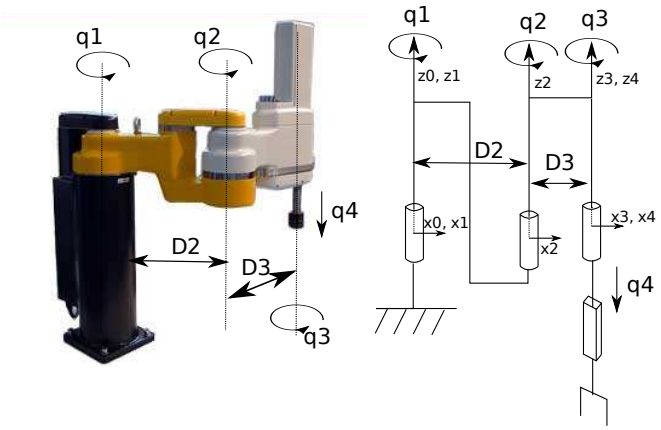


FIG. A.2 – Exemple pour un robot manipulateur

i	α_i	d_i	θ_i	r_j
1	0	0	q_1	0
2	0	D_2	q_2	0
3	0	D_3	q_3	0
4	0	0	0	q_4

TAB. A.1 – Paramètres de Denavitt-Hartenberg pour un robot manipulateur SCARA

quatre matrices de transfert telles que :

$$T_{0,1} = \begin{bmatrix} cos(q_1) & -sin(q_1) & 0 & 0 \\ sin(q_1) & cos(q_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.3}$$

$$T_{1,2} = \begin{bmatrix} cos(q_2) & -sin(q_2) & 0 & D_2 \\ sin(q_2) & cos(q_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.4}$$

$$T_{2,3} = \begin{bmatrix} cos(q_3) & -sin(q_3) & 0 & D_3 \\ sin(q_3) & cos(q_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.5}$$

$$T_{3,4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.6}$$

Annexe B

Expression détaillée du ZMP pour la plate-forme B2P2

B.1 Moment dynamique de B2P2

Dans [Djoudi 2007], le moment dynamique est défini comme étant la dérivée du moment cinétique pour tout point fixe dans le repère considéré. Dans le cadre de la formulation du ZMP, le moment cinétique est pris en compte à l'origine du repère R_6 du modèle géométrique présenté par la figure B.1, ainsi, on peut noter :

$$\delta_0 = \frac{d}{dt}\sigma_O(S/R_6) \tag{B.1}$$

Appliquons la définition du moment cinétique d'un système multi-corps défini dans la section 3.2.3.2 au robot B2P2 :

$$\sigma_O(S/R_6) = \sum_{i=1}^{3} I_{Gi}\Omega_i + \sum_{i=1}^{3} miV_{Gi} \wedge G_i 0 \tag{B.2}$$

Ainsi, le moment dynamique du robot B2P2 calculé à l'origine du repère R_6 est obtenu par l'équation suivante :

$$\delta_0 = \sum_{i=1}^{3} I_{Gi}\dot{\Omega}_i + \sum_{i=1}^{3} mi \times \begin{bmatrix} y_{G_i}\ddot{z_{G_i}} - z_{G_i}\ddot{y_{G_i}} \\ x_{G_i}\ddot{z_{G_i}} - z_{G_i}\ddot{x_{G_i}} \\ x_{G_i}\ddot{y_{G_i}} - y_{G_i}\ddot{x_{G_i}} \end{bmatrix} \tag{B.3}$$

B.2 Expression analytique du zmp pour B2P2

Rappelons tout d'abord l'expression formelle du ZMP telle que définie en section 3.3.2 :

$$\begin{cases} Z_y = \frac{\delta_{0x}(\ddot{q}) - G_y(q)P_z + G_z(q)P_y}{R_z(\ddot{q})} \\ Z_x = \frac{-\delta_{0y}(\ddot{q}) + G_z(q)P_x - G_x(q)P_z}{R_z(\ddot{q})}. \end{cases} \tag{B.4}$$

Comme $R = mg\vec{z} + m\ddot{G}$, on obtient alors :

$$\begin{cases} Z_y = \frac{\delta_{0x}(\ddot{q}) + y_G mg}{m(g + \ddot{z_g})} \\ Z_x = \frac{-\delta_{0y}(\ddot{q}) + x_G mg}{m(g + \ddot{z_g})}. \end{cases} \tag{B.5}$$

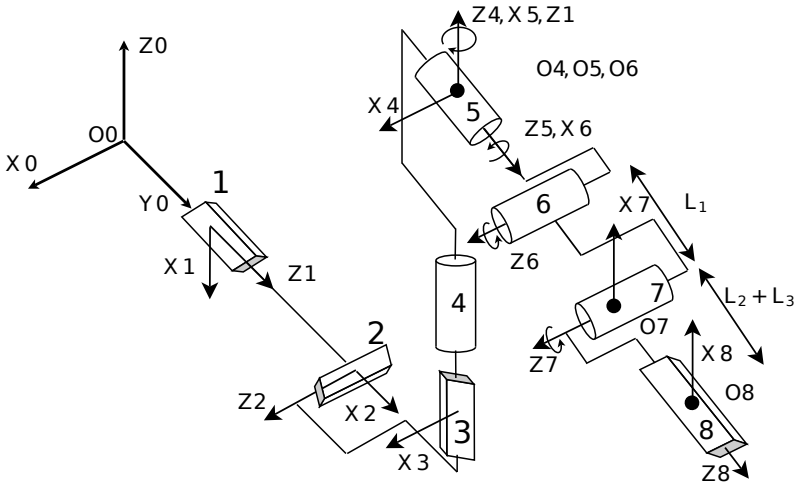


FIG. B.1 – Modèle géométrique du robot B2P2

Ainsi, l'expression complète de la position du ZMP dans le repère R_6 de la plate-forme B2P2 devient :

$$Z_x = \frac{-(\sum_{i=1}^{3} I_{Gi}\dot{\Omega}_i + \sum_{i=1}^{3} mi \times (x_{G_i}\ddot{z_{G_i}} - z_{G_i}\ddot{x_{G_i}})) + x_G mg}{m(g + \ddot{z_g})}. \tag{B.6}$$

$$Z_y = \frac{\sum_{i=1}^{3} I_{G_i}\dot{\Omega}_i + \sum_{i=1}^{3} mi \times (y_{G_i}\ddot{z_{G_i}} - z_{G_i}\ddot{y_{G_i}}) + y_G mg}{m(g + \ddot{z_g})} \tag{B.7}$$

Les tenseurs d'inertie notés I_{G_1}, I_{G_2} et I_{G_3} ont été déterminés pour chacun des solides du robot à l'aide du logiciel utilisé pour la CAO du véhicule (SolidWorks 2007).

B.3 Constantes mécaniques du robot

- Poids :
  - Solide 1 : 6.356067 Kg
  - Solide 2 : 0.897066 Kg
  - Solide 3 : 1.06215 Kg

- Densité :
  - Solide 1 : 2080.745009 $Kg.m^{-3}$
  - Solide 2 : 1739.217853 $Kg.m^{-3}$
  - Solide 3 : 2423.47357 $Kg.m^{-3}$

B.3.0.1 Dimensions

- Longueurs :
  - Solide 1 : 0.32 m
  - Solide 2 : 0.23 m
  - Solide 3 : 0.226 m
- Hauteur : 0.60 m
- Largeur : 0.37 m
- Volume :
  - Solide 1 : 0.003055 m^3
  - Solide 2 : 0.000516 m^3
  - Solide 3 : 0.00044 m^3

B.3.0.2 Tenseurs d'inertie, centre de gravité

- Tenseurs d'inertie ($Kg.m^2$) :
  - I_{G_1} :
$$\begin{pmatrix} 0.128728 & -0.033872 & 0.002259 \\ -0.033872 & 0.363749 & 0.001158 \\ 0.002259 & 0.001158 & 0.283586 \end{pmatrix}$$
  - I_{G_2} :
$$\begin{pmatrix} 0.016485 & -0.000107 & 0.000002 \\ -0.000107 & 0.003941 & 0.001773 \\ 0.000002 & 0.001773 & 0.012863 \end{pmatrix}$$
  - I_{G_3} :
$$\begin{pmatrix} 0.05229 & 0 & -0.00002 \\ 0 & 0.02964 & 0 \\ -0.00002 & 0 & 0.02523 \end{pmatrix}$$
- Coordonnées du centre de gravité :
  - Solide 1 (pris dans le repère R_6) :
$$\begin{pmatrix} 0.160894 \\ 0.034997 \\ -0.002784 \end{pmatrix}$$

– Solide 2 (pris dans le repère R_7) :

$$\begin{pmatrix} -0.000723 \\ -0.104083 \\ 0.014127 \end{pmatrix}$$

– Solide 3 (pris dans le repère R_8) :

$$\begin{pmatrix} 0.00009 \\ -0.00001 \\ 0.15268 \end{pmatrix}$$

Annexe C

Analyse par intervalles et projection de contraintes

Sommaire

Dans cette annexe, nous présentons les outils liés à l'arithmétique par intervalles utilisés dans l'étude des VGSTVs vu au chapitre 4.
Cette annexe est organisée en deux parties ; après avoir détaillé l'arithmétique par intervalles et l'inversion ensembliste, nous présenterons les outils de propagation de contraintes et de contraction.

C.1 Analyse par intervalles

C.1.1 Historique

L'analyse par intervalles est apparue dans les premiers travaux de Young [Young 1932] ; il faudra attendre les progrès informatiques des années soixante pour voir apparaître l'implémentation en machine de ce type de calcul notamment dans les travaux de Moore [Moore 1966]. Ce nouveau formalisme permet d'évaluer des opérations sans dépendre du codage machine des données qui induit dans certains cas des erreurs non négligeables comme souligné dans le livre de Hansen [Hansen 1992]. Bien entendu, de la taille de l'intervalle étudié dépend le temps de calcul et la pertinence du résultat. Il sera alors toujours question de compromis entre ces deux paramètres

pour déterminer une taille raisonnable. La dimension des problèmes traités reste aussi une limitation majeure de cet outil et est aujourd'hui un aspect fondamental de la recherche dans ce domaine.

C.1.2 Arithmétique par intervalles

Un intervalle $[x]$ est un sous-ensemble fermé et connexe de $\mathbb{R}$. On pourra noter :

$$[x] = \{x \in \mathbb{R} | x^- \leq x \leq x^+, x^- \in \mathbb{R}, x^+ \in \mathbb{R}\} \tag{C.1}$$

où x^- et x^+ représentent respectivement les bornes inférieures et supérieures de l'intervalle $[x]$. Une intervalle sera ainsi notée : $[x] = [x^-; x^+]$.

Remarque 15. *Un réel sera représenté par un intervalle avec deux bornes égales. Il conviendra alors de le représenter à l'aide d'une seule de ses bornes. Par exemple,* 3 *sera représenté par l'intervalle* [3].

L'utilisation d'intervalles permet de garantir les résultats obtenus. En effet, une machine traite les réels avec une précision donnée selon le type de codage. Certains arrondis peuvent alors amener à des dérives importantes [Hansen 1992]. Encadrer les réels par un intervalle réduit, ou du moins encadre cette imprécision.

C.1.2.1 Opérations ensemblistes sur les intervalles

L'utilisation d'intervalles introduit de nouveaux opérateurs liés aux propriétés de ce type de données :

– Intersection :

$$[x] \cap [y] = \{x \in [x] \text{ et } y \in [y]\}, \tag{C.2}$$

– Union :

$$[x] \cup [y] = \{x \in [x] \text{ ou } y \in [y]\}, \tag{C.3}$$

– Deprivation :

$$[x] \setminus [y] = \{x | [x] \text{ et } x \notin [y]\}, \tag{C.4}$$

– Produit cartésien :

$$[x] \times [y] = \{(x, y) | x \in \text{ [x] et } x \in [y]\}, \tag{C.5}$$

– Projection :

$$Proj_x(Z) = \{x \in [x] | \exists y \in \text{ [y] tel que } (x, y) \in Z\}. \tag{C.6}$$

Une application graphique de ces opérations est présenté par la figure C.1.

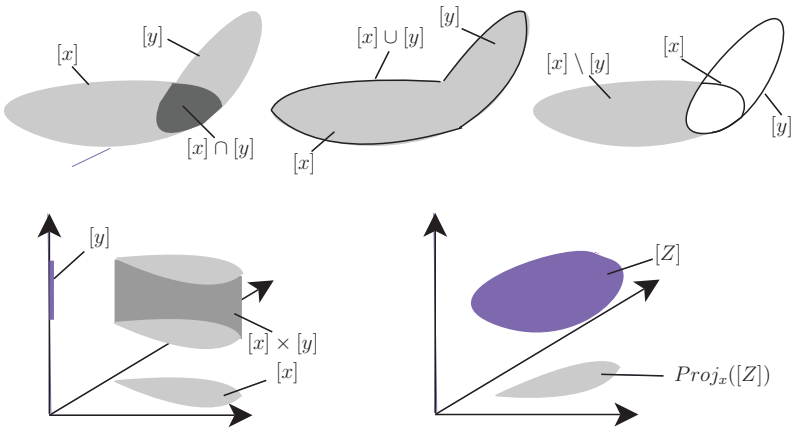


FIG. C.1 – Opérations ensemblistes sur les intervalles

C.1.2.2 Opérations algébriques

Les opérations élémentaires sur les réels peuvent être adaptées aux intervalles. Classiquement, elles sont définies de la manière suivante :

– L'addition :

$$[x] + [y] = [x^- + y^-; x^+ + y^+], \tag{C.7}$$

– La soustraction :

$$-[x] = [-x^+; -x^-] \tag{C.8}$$

$$[x] - [y] = [x] + (-[y]), \tag{C.9}$$

– La multiplication :

$$[x] * [y] = [min(x^-y^-, x^+y^-, x^-y^+, x^+y^+); max(x^-y^-, x^+y^-, x^-y^+, x^+y^+)], \tag{C.10}$$

– La division :

$$\frac{1}{[y]} = [\frac{1}{y^+}; \frac{1}{y^-}] \text{ si } 0 \notin [y],]-\infty; +\infty[\text{ sinon.} \tag{C.11}$$

C.1.2.3 Pavés, bissections et encadrement d'ensembles

a - Pavé La notion de pavé (ou vecteur d'intervalles) de $\mathbb{R}^n$ induit d'encadrer un ensemble, voir d'approximer des vecteurs incertains. Il s'agit du produit cartésien

de n intervalles écrit sous la forme suivante :

$$[\mathbf{x}] = [x_1^-; x_1^+] \times [x_2^-; x_2^+] \times ... \times [x_n^-; x_n^+] = \begin{pmatrix} [x_1^-; x_1^+] \\ [x_2^-; x_2^+] \\ ... \\ [x_n^-; x_n^+] \end{pmatrix} \tag{C.12}$$

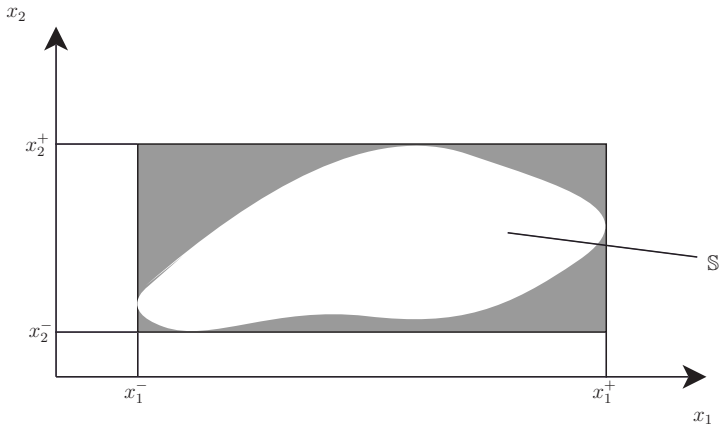


FIG. C.2 – Encadrement d'un ensemble $\mathbb{S}$ par une boîte. Les points en gris n'appartiennent pas à $\mathbb{S}$

Cette représentation induit un certain pessimisme comme le démontre la figure C.2. Ainsi, un ensemble $\mathbb{S}$ de forme quelconque sera encadré par un pavé $[\mathbf{x}]$ contenant tout les points de $\mathbb{S}$, mais aussi d'autres points n'appartenant pas à $\mathbb{S}$.

b - Bissection Une opération de bissection permet de scinder un pavé en deux selon son plan principal ; plan de symétrie perpendiculaire au côté le plus large. Par exemple, la bissection du pavé $[\mathbf{x}] = [1; 2] \times [-1; 3]$ va générer deux nouveaux pavés : $[\mathbf{x}_1] = [1; 2] \times [-1; 1]$ et $[\mathbf{x}_2] = [1; 2] \times [1; 3]$.

Remarque 16. *Un pavé $[\boldsymbol{x}]$ pourra être qualifié d'inclus dans un ensemble défini par une fonction f de $\mathbb{R}^n$ dans $\mathbb{R}^m$ si $f([\boldsymbol{x}]) \subset \mathbb{S}$. Au contraire, il sera dit exclu si $f([\boldsymbol{x}]) \cap \mathbb{S} = \varnothing$. On le qualifiera d'incertain si $f([\boldsymbol{x}]) \cap \mathbb{S} \neq \varnothing$ et $f([\boldsymbol{x}]) \cup \mathbb{S} \neq \mathbb{S}$. La fonction f est quand à elle appelée fonction d'inclusion.*

C.1.2.4 Fonctions d'inclusion

Definition 1. *Soit f une fonction de $\mathbb{R}^n$ à valeurs dans $\mathbb{R}^m$. La fonction intervalle $[f] : \mathbb{R}^n \to \mathbb{R}^m$ est une fonction d'inclusion pour f si :*

$$\forall x \in \mathbb{R}^n, f([x]) \subset [f]([x]) \tag{C.13}$$

Remarque 17. *Si $f([x]) = [f]([x])$, alors la fonction d'inclusion est dite minimale.*

La figure C.3 illustre l'évaluation d'un intervalle $[x]$ par une fonction f (fonction d'inclusion minimale) et par une fonction [f] d'inclusion quelconque.

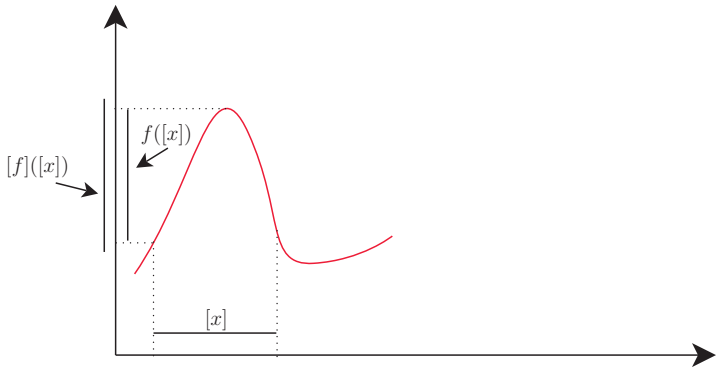


FIG. C.3 – Évaluation de l'intervalle [x] via une fonction [f] d'inclusion quelconque.

Exemple 1. *La fonction constante [f] : $\mathbb{R} \to \mathbb{R}$, définie par $[f] : [x] \mapsto [-1; 1]$ est une fonction d'inclusion pour la fonction sin.*

Par conséquent, les opérations algébriques présentées en section C.1.2.2 constituent les fonctions d'inclusions des fonctions usuelles (addition, soustractions etc...) et c'est cette famille de fonctions d'inclusions qui forme ce qui est appelé l'arithmétique par intervalles.

C.1.2.5 Fonctions d'extension naturelle

Un moyen de calculer rapidement un intervalle réalisant une fonction d'inclusion, sans aucune garantie d'obtenir un encadrement minimal consiste à remplacer chacune des variables par leur domaine intervalle associé. Cette opération est appelée fonction d'extension naturelle.

Exemple 2. *Soit la fonction suivante :*

$$f_1(x) = x^2 - x - 1. \tag{C.14}$$

Son évaluation naturelle pour trois intervalles distincts [0 ;1], [1 ;2], et [2 ;3] donne respectivement [-2 ;0], [-4 ;2] et [0 ;6]. La figure C.4 présente l'évolution de cette fonction ainsi que les intervalles d'évaluation calculés. On peut noter un important pessimisme dû à ce que l'on appelle le problème de dépendance.

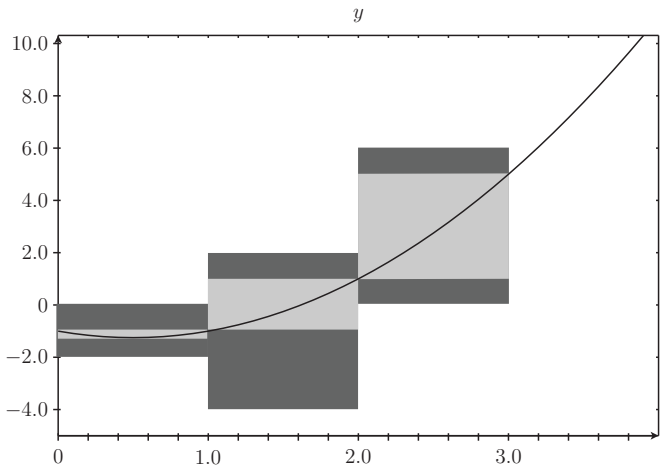


FIG. C.4 – Évaluation naturelle de $f_1([x])$ (foncé) et $f_2([x])$ (clair).

a - Le problème de dépendance Dans le cadre de la manipulation d'intervalles, la taille de l'intervalle solution d'une fonction peut dépendre du nombre d'occurrence de chaque variable. Pour illustrer ce phénomène, prenons l'évaluation sur $[x] = [-1; 1]$ de deux expressions équivalentes :

$$\begin{array}{rcl} [x] + [x] - [x] & = & [-1;1] + [-1;1] - [-1;1] = [-3;3] \\ [x] & = & [-1;1] \end{array} \tag{C.15}$$

Il convient donc de veiller à réduire le nombre d'occurrences des variables pour obtenir l'évaluation la plus petite possible.

Exemple 3. *Ainsi, la fonction f_1 évaluée dans l'exemple précèdent peut être écrite :*

$$f_2(x) = (x - \frac{1}{2})^2 - \frac{5}{4} \tag{C.16}$$

Son évaluation naturelle pour trois intervalles distincts [0 ;1], [1 ;2], et [2 ;3] devient alors [-1.25 ;-1], [-1 ;1] et [1 ;5]. Les éléments clairs qui apparaissent sur la figure C.4 représentent ces nouvelles évaluations.

Ces exemples illustrent de façon claire que l'évaluation d'une fonction sur un intervalle donné n'est pas toujours pertinente. On cherchera alors à réduire l'ensemble de façon optimale dans le but de toujours obtenir le plus petit encadrement possible. Cette opération passe la plupart du temps par une décomposition de la fonction à évaluer.

C.1.3 Inversion ensembliste

Prenons le problème suivant :

$$y = f(x_1, x_2 ... x_n) \tag{C.17}$$

En connaissant les domaines de définition de x et de y notés

$$x_1 \in [x_1], x_2 \in [x_2], ..., x_n \in [x_n], y \in \mathbb{Y} \tag{C.18}$$

L'analyse par intervalles permet de définir l'ensemble $\mathbb{S}$ tel que :

$$\mathbb{S} = \{x_1 \in [x_1], x_2 \in [x_2], ..., x_n \in [x_n], f(\ [\mathbf{x}]\ ) \in \mathbb{Y}\} \tag{C.19}$$

Cette opération est appelée inversion ensembliste et permet de déterminer un ensemble de pavés $[\mathbf{x}_1]$, $[\mathbf{x}_2]$,..., $[\mathbf{x}_n]$ pour lesquels la fonction f est vérifiée comme illustré sur la figure C.5.

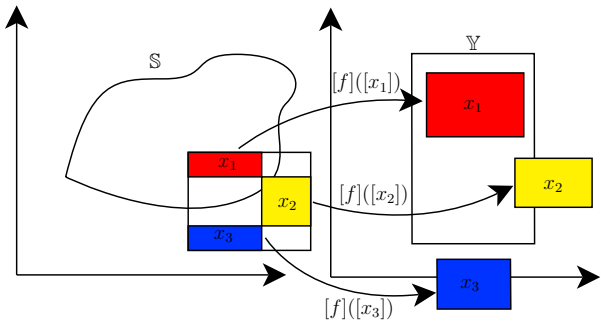


FIG. C.5 – Principe de l'inversion ensembliste

L'algorithme 9 dit SIVIA (Set Inversion Via Interval Analysis) [L. Jaulin 1993] constitue un outil efficace pour déterminer $\mathbb{S}$ dans le cadre de problèmes de faible dimension. Le pavé $[\mathbf{x}] = \begin{pmatrix} [x_1^-; x_1^+] \\ [x_2^-; x_2^+] \\ ... \\ [x_n^-; x_n^+] \end{pmatrix}$ est évalué (évaluation naturelle) via la fonction $[f]$. Si $[f](\ [\mathbf{x}]\ )$ est un sous ensemble de l'ensemble solution $\mathbb{Y}$, alors $[\mathbf{x}]$ constitue un ensemble garanti et appartient à l'approximation intérieure de $\mathbb{S}$ (pavé inclus). A l'inverse, si $[f](\ [\mathbf{x}]\ ) \cap \mathbb{Y} = \varnothing$ le pavé $[\mathbf{x}]$ ne contient alors aucune solution et n'appartient donc pas à $\mathbb{S}$ (pavé exclu). Lorsqu'un pavé est dit incertain, il contient des points solutions, et non solutions. On bissecte alors l'ensemble $[\mathbf{x}]$ testé pour relancer l'algorithme pour les deux nouveaux pavés obtenus. Cet algorithme récursif s'arrête lorsque le pavé testé atteint une certaine taille ε. Cela induit l'obtention de deux ensembles, solutions et non solutions, séparés par un ensemble de pavés incertains trop petits pour être réévalués (voir figure C.6). En raison du caractère exponentiel de cet algorithme, il existe une limitation de la dimension du problème de l'ordre de 5. Cependant, des outils de contraction permettant de réduire l'espace

de définition des variables $x_1, x_2 ... x_n$ peuvent être utilisés de manière à réduire le nombre de bissections. Ces outils seront analysés dans la section C.2.

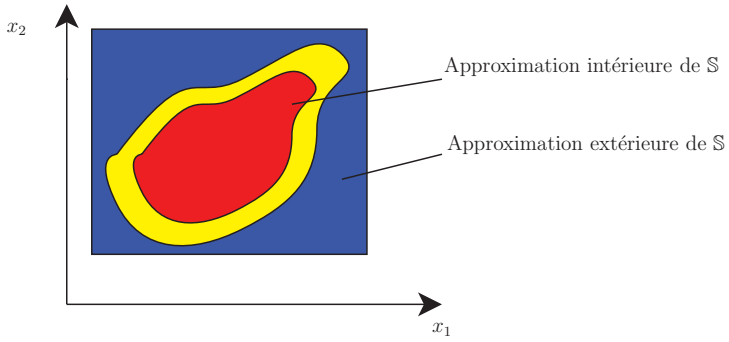


FIG. C.6 – Approximation d'ensemble via SIVIA

Remarque 18. *Par convention, les pavés solutions, se verront représenter en rouge ; les pavés non solutions en bleu et les pavés incertains en jaune.*

Algorithme 9 Algorithme d'inversion ensembliste : SIVIA

```
SIVIA( [x] )
si [f]( [x] ) ⊂ [y] alors
   [x] représente un pavé solution
sinon
   si [f]( [x] ) ∩ [y] = ∅ alors
      [x] ne représente pas un pavé solution
   sinon
      si Taille( [x] )<ε alors
         [x] est trop petit pour être réévalué et toujours incertain
      sinon
         [x] est un pavé incertain
         il va être bissecté, et les deux pavés résultants vont être
         évalué tour à tour.
         bisect( [q] , [q1] , [q2] )
         SIVIA( [q1] )
         SIVIA( [q2] )
```

C.1.3.1 Problèmes de Satisfaction de contraintes

Le formalisme des problèmes de satisfaction de contraintes, ou CSP pour "Contraint Satisfaction problems" est défini par trois ensembles :

$$\left\{\begin{array}{lcl} V & : & x_1, x_2, ..., x_n \\ & & y_1, x_2, ..., y_n \\ D & : & [x_1], [x_2], ..., [x_n] \\ & & [y_1], [y_2], ..., [y_n] \\ C & : & f_1(x) = y_1 \\ & & ... \\ & & f_n(x) = y_n \end{array}\right\} \tag{C.20}$$

L'ensemble V représente l'ensemble des variables du problème, D associe à chaque variable un domaine de définition, et enfin C contient l'ensemble des contraintes liées au problème. Ainsi, si on définit l'ensemble $\mathbb{S}$ tel que :

$$\mathbb{S} = (x_1 \in [x_1], x_2 \in [x_2], y \in [y] | f(x_1, x_2) > 0) \tag{C.21}$$

Le CSP équivalent est alors donné par :

$$\left\{\begin{array}{lcl} V & : & x_1, x_2, y \\ D & : & [x_1] =]-\infty; +\infty[, [x_2] =]-\infty; +\infty[, y = [0; +\infty[\\ C & : & f : y_1 = x_1 + x_2 \end{array}\right\} \tag{C.22}$$

Pour illustrer l'intérêt de ce formalisme dans le cadre de l'analyse par intervalles, prenons un exemple simple :

$$\left\{\begin{array}{lcl} V & : & x_1, x_2, y_1, y_2 \\ D & : & [x_1] =]-\infty; +\infty[, [x_2] =]-\infty; +\infty[, y_1 = [0; 1], y_2 = [0; 1] \\ C & : & f_1 : y_1 = x_1 + 2 - x_2 \\ & & f_2 : y_2 = x_1^2 - x_2 \end{array}\right\} \tag{C.23}$$

L'ensemble $\mathbb{S}$ associé à ce CSP est présenté sur la figure C.7. Il correspond à l'intersection des ensembles vérifiant respectivement les fonctions f_1 et f_2.

Les algorithmes d'inversion ensemblistes tels que SIVIA permettent de définir un ensemble $\mathbb{S}$ de pavés $[\mathbf{x}]$ pour lesquels l'équation $f(\ [\mathbf{x}]\ ) = [\mathbf{y}]$ est garantie. C'est pourquoi, l'association d'outils d'inversion ensemblistes au formalisme CSP est un moyen efficace d'obtenir un ensemble contraint que l'on pourra qualifier d'espace de travail contraint dans le cadre de problèmes de robotique étudiés au chapitre 4.

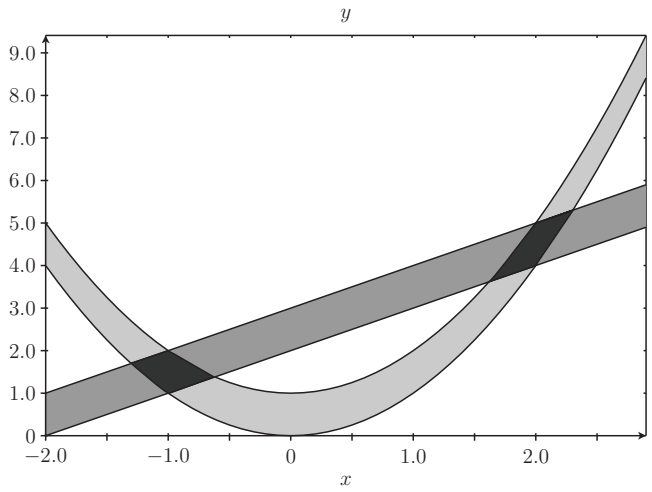


FIG. C.7 – Représentation du CSP

C.2 Propagation de contraintes et contracteurs

Les algorithmes d'inversion ensemblistes qui évaluent une fonction tels que SIVIA sont aussi très utiles dans le cas de résolution de problèmes contraints. En effet, les fonctions à évaluer peuvent être vues comme un ensemble de contraintes. Cependant, ces algorithmes récursifs ont un temps d'exécution exponentiel ; les outils de propagation de contraintes permettent alors de réduire le domaine de définition des variables à évaluer. L'utilisation conjointe de ces outils aux algorithmes d'inversions ensemblistes offre donc la possibilité de réduire le nombre d'appels successifs, e. g. de bissections, en réduisant l'espace à évaluer.
Dans cette partie, les principes de la propagation de contraintes sont expliqués et illustrés, puis un algorithme de résolution de CSP utilisant ces outils pour réduire le temps de traitement est présenté.

C.2.1 Projection de contraintes : principe

Prenons un ensemble de contraintes :

$$\begin{array}{lcl} c_1 & : & f_1(x) = 0 \\ c_2 & : & f_2(x) = 0 \\ & \ldots & \\ c_n & : & f_n(x) = 0 \end{array} \tag{C.24}$$

Un ensemble $\mathbb{S}$ contient toutes les valeurs de x_i qui satisfont les contraintes c_1, c_2 ... c_n. La projection de ces contraintes sur chacun des x_i, consiste à réévaluer chaque domaine $[x_i]$ en fonction des contraintes c_i.

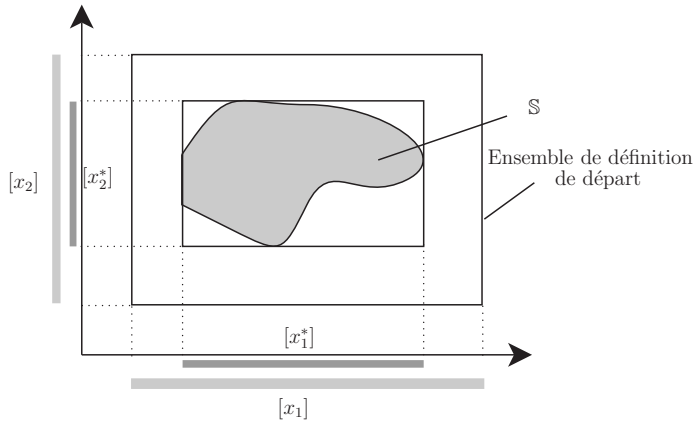


FIG. C.8 – Projection d'un ensemble sur deux axes x_1 et x_2

Ainsi, comme le montre la figure C.8, les opérations de projection de contraintes vont réduire un domaine de définition défini par un pavé $[x_1; x_2]$ en un pavé $[x_1^*; x_2^*]$ optimal, e. g. un plus petit pavé encadrant l'ensemble solution.

Prenons par exemple trois variables x, y et z et une contrainte telles que :

$$\begin{array}{rcl} x & \in & [0;6] \\ y & \in & [3;9] \\ z & \in & [0;10] \\ z & = & x-y \end{array} \tag{C.25}$$

Projeter la contrainte $z = x - y$ consiste à définir le plus petit ensemble $\mathbb{S}$ tel que :

$$\mathbb{S} = \{(x, y, z) \in [0;6] \times [3;9] \times [0;10] | z = x - y\} \tag{C.26}$$

Ainsi, dans notre exemple, la projection de la contrainte est donnée par :

$$\begin{array}{rcll} z = x - y & \Rightarrow & z \in & [0;10] \cap [0;6] - [3;9] = [0;3] \\ x = z + y & \Rightarrow & x \in & [0;6] \cap [0;3] + [3;9] = [3;6] \\ y = x - z & \Rightarrow & y \in & [3;9] \cap [3;6] - [0;3] = [3;6] \end{array} \tag{C.27}$$

Cette opération simple réduit facilement le domaine de définition d'un problème. Ici, le domaine de définition de la variable z est passé de [0 ;10] à [0 ;3] à l'aide d'une seule intersection d'intervalle. A titre de comparaison, un algorithme classique de type SIVIA aurait tout d'abord évalué la fonction $[z] = [x] - [y]$. L'intervalle alors obtenu : $[-9; 3]$ démontre que le pavé x, y testé est incertain ($[-9; 3] \cap [0; 10] \neq \varnothing$ et $[-9; 3] \cup [0; 10] \neq [0; 10]$) ; il va alors être bissecté puis réévalué et ainsi de suite. L'efficacité de la projection de la contrainte réduit, ici, considérablement le temps

d'approximation d'un ensemble en diminuant rapidement le domaine d'exploration initial aux valeurs strictement consistantes.

Remarque 19. *Les contraintes citées dans cet exemple constituent ce que l'on appelle des contraintes primitives, c'est à dire ne contenant qu'une addition, soustraction ou fonction élémentaire. Dans le cadre de la projection de contraintes plus complexes, une décomposition en contraintes primitives sera indispensable.*

C.2.2 Propagation de contraintes

Considérons le CSP suivant :

$$\left\{\begin{array}{lcl} V & : & y, x \\ D & : &]-\infty;+\infty[,]-\infty;+\infty[\\ C & : & (C_1) : y = x^2 \\ & & (C_2) : y^2 + x^2 = 2 \\ & & (C_3) : -x + 3 - y = 0 \end{array}\right\} \tag{C.28}$$

Propager les contraintes de ce problème consiste à projeter chacune des contraintes de manière itérative jusqu'à obtenir l'équilibre. Autrement dit jusqu'à ce que les projections ne réduisent plus le domaine de définition. Ainsi, les contraintes du CSP présenté par l'équation C.28 seront projetées de la manière suivante (la figure C.9 propose une illustration graphique à ces opérations) :

$$\begin{array}{lllll} (C1) & y = x^2 & \Rightarrow & y \in]-\infty;+\infty[^2 = [0;+\infty[\\ (C2) & y^2 = 2 - x^2 & \Rightarrow & y^2 \in [0;+\infty[\cap 2 - [0;+\infty[=]-\infty;2[\cap [0;+\infty[= [0;2] \\ & & \Rightarrow & y \in [0;+\infty[\cap [-\sqrt{2};\sqrt{2}] = [0;\sqrt{2}] \\ (C2) & x^2 = 2 - y^2 & \Rightarrow & x^2 \in [0;+\infty[\cap 2 - [0;\sqrt{2}] =]2-\sqrt{2};2[\cap [0;+\infty[= [2-\sqrt{2};2] \\ & & \Rightarrow & x \in]-\infty;+\infty[\cap [-\sqrt{2};\sqrt{2}] = [-\sqrt{2};\sqrt{2}] \\ (C3) & x = -y + 3 & \Rightarrow & x \in [-\sqrt{2};\sqrt{2}] \cap [-\sqrt{2};0] + 3 = [-\sqrt{2};\sqrt{2}] \cap [3-\sqrt{2};3] = \varnothing \\ (C1) & y = x^2 & \Rightarrow & y \in [0;\sqrt{2}] \cap \varnothing = \varnothing \end{array} \tag{C.29}$$

Remarque 20. *Dans certains cas, la propagation de contraintes ne permet pas de réduire efficacement le domaine d'évaluation. Il conviendra alors de modifier la contrainte elle même en ajoutant par exemple des contraintes redondantes.*

C.2.3 Les contracteurs

La notion de contracteurs regroupe plusieurs techniques de réductions de domaines. Seul le contracteur par projection de contraintes sera abordé dans cette partie car il possède l'avantage de ne pas avoir de restrictions sur la forme du CSP.

a - Principe Un contracteur, noté C_x est un opérateur de $\mathbb{R}^n$ dans $\mathbb{R}^n$ chargé d'éliminer d'un pavé les valeurs inconsistantes (non comprises dans l'ensemble solution). Ainsi, pour un pavé **[x]** donné, un contracteur $C($ **[x]** $)$ retournera un pavé **[x']** diminué ou non (si **[x]** est déjà suffisamment contracté) par rapport à **[x]** .

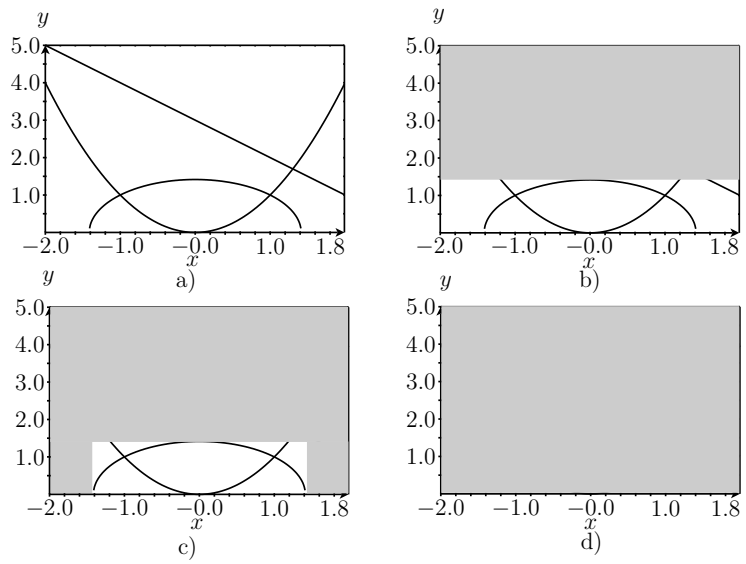


FIG. C.9 – Propagation de contraintes. b) : La propagation de l'équation C2 sur y réduit son domaine de définition. c) : C2 est ensuite projetée sur x. d) : finalement, la projection de C3 indique une absence de solutions ; il n'y a en effet pas d'intersection entre les trois courbes.

b - Contractance Le pavé contracté sera toujours inclus dans le pavé de recherche :

$$\forall\ [\mathbf{x}]\ \in \mathbb{R}^n, C(\ [\mathbf{x}]\ ) \subset\ [\mathbf{x}]\ . \tag{C.30}$$

c - Complétude L'ensemble des points solutions du pavé de recherche appartient au pavé contracté :

$$\forall\ [\mathbf{x}]\ \in \mathbb{R}^n,\ [\mathbf{x}]\ \cap \mathbb{S} \subset C(\ [\mathbf{x}]\ ). \tag{C.31}$$

d - Équilibre N'oublions pas que le but recherché est de réduire au maximum le domaine d'évaluation d'un problème. Il convient alors de s'intéresser au pavé minimal. Ainsi, tant que le résultat du contracteur n'est pas un pavé d'équilibre, i. e. tant que $C(\ [\mathbf{x}]\ ) \neq [\mathbf{x}]$, le contracteur C est appliqué au dernier pavé contracté. Ces appels successifs peuvent être notés :

$$C \circ C \circ C(\ [\mathbf{x}]\ ) = C(C(C(\ [\mathbf{x}]\ ))). \tag{C.32}$$

Remarque 21. *Le nombre d'appels du contracteur étant lié au problème, il est difficile à déterminer à priori.*

C.2.4 La notion de consistance

Prenons deux contracteurs $C_{\mathbb{S}_1}$ et $C_{\mathbb{S}_2}$, deux contracteurs minimaux pour les ensembles $\mathbb{S}_1$ et $\mathbb{S}_2$. Notons $C_{\mathbb{S}}$ le contracteur de $\mathbb{S} = \mathbb{S}_1 \cap \mathbb{S}_2$ tel que $C_{\mathbb{S}} = C_{\mathbb{S}_1} \circ C_{\mathbb{S}_2} \circ C_{\mathbb{S}_1} \circ C_{\mathbb{S}_2} \circ .....$

Le contracteur ainsi obtenu ne sera malheureusement pas toujours optimal. Ce phénomène est appelé l'effet de consistance locale.

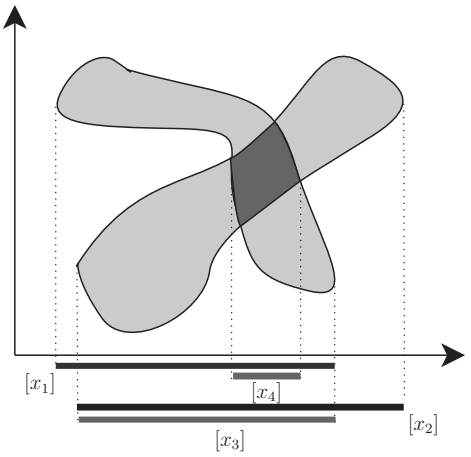


FIG. C.10 – Illustration de la consistance globale et locale

La figure C.10 représente bien le problème. $[x_1]$ et $[x_2]$ sont les plus petits encadrements des ensembles $\mathbb{S}_1$ et $\mathbb{S}_2$. Or, il apparaît clairement que l'intersection de $[x_1]$ et $[x_2]$ (notée $[x_3]$) n'est pas l'encadrement optimal de $\mathbb{S}_1 \cap \mathbb{S}_2$. On dira que $[x_3]$ est consistant localement, à l'inverse de $[x_4]$ qui sera consistant globalement.

Remarque 22. *L'efficacité d'un contracteur, et donc sa consistance est aussi liée à la formulation de la contrainte. En effet, les multi-occurences de certaines variables ont un impact direct sur l'efficacité de l'évaluation, et donc de la contraction de l'ensemble. Il conviendra donc de décomposer correctement les contraintes afin d'obtenir un contracteur optimal.*

C.2.5 Contraintes redondantes

Du fait des problèmes de consistance locale, il sera parfois impossible de contracter un ensemble de définition. Prenons le CSP suivant :

$$\left\{ \begin{array}{lcl} V & : & y, x \\ D & : & [-10;10], [-10;10] \\ C & : & (C_1) : y + x = 0 \\ & & (C_2) : y - x = 0 \end{array} \right\} \tag{C.33}$$

Une propagation de contraintes simple ne permet pas de contracter efficacement l'ensemble des $[x], [y]$:

$$\begin{array}{lll} (C1) & \Rightarrow & y \in [-10;10] \cap [-10;10] = [-10;10] \\ (C2) & \Rightarrow & x \in [-10;10] \cap [-10;10] = [-10;10] \end{array} \tag{C.34}$$

Ajoutons à ce CSP une nouvelle contrainte notée C_3 déduite directement de C_1 et C_2 telle que $C_3 : C_1 + C_2 = 0 \Rightarrow 2y = 0$. La propagation de contrainte du nouveau CSP devient alors :

$$\begin{array}{lll} (C1) & \Rightarrow & y \in [-10;10] \cap [-10;10] = [-10;10] \\ (C2) & \Rightarrow & x \in [-10;10] \cap [-10;10] = [-10;10] \\ (C3) & \Rightarrow & y \in [-10;10] \cap [0;0] = [0;0] \end{array} \tag{C.35}$$

C.2.6 Algorithmes de propagation de contraintes

Pour être efficaces, les algorithmes de contraction/propagation dépendent de la formulation des contraintes. En introduisant la notion d'opérateur de projection, nous nous proposons de définir les algorithmes de projection optimaux de contraintes à une, deux ou trois variables (opérateurs arithmétiques, sin, cos,...)

C.2.6.1 Exemple d'opérateur de projection

Prenons l'opérateur arithmétique de l'addition, la contrainte étudiée étant :

$$x_1 = x_2 + x_3 \tag{C.36}$$

L'algorithme 10 présente l'opérateur de projection associé. Cet algorithme propose une décomposition de l'addition suivant chacune des variables de la contrainte dans un sens (propagation) puis dans les sens inverse (retro-propagation). Il en résulte une contraction optimale de la contrainte. Une analyse plus détaillée ainsi qu'une présentation des plusieurs opérateurs de contraction est disponible dans la référence [Baguenard 2005].

Algorithme 10 Opérateur de projection de l'addition

```
1: Projection Addition(in : sens, in out : [x1], [x2], [x3])
2: si sens==1 alors
3:     [x1] := [x1] ∩ ([x2] + [x3])
4: si sens==-1 alors
5:     [x2] := [x2] ∩ ([x1] − [x3])
6:     [x3] := [x3] ∩ ([x1] − [x2])
```

Remarque 23. *L'opérateur de projection ne sera pas toujours aussi simple a trouver. Prenons par exemple une contrainte de multiplication :*

$$x_1 = x_2 * x_3 \tag{C.37}$$

Si on se contente d'un opérateur de projection basique comme présenté par l'algorithme 11, il n'en résultera pas une contraction optimale. En effet, si 0 appartient aux domaines $[x_2]$ *et* $[x_3]$*, les opérations* $[x_1] := [x_1] \cap ([x_1]/[x_2])$ *et* $[x_1] := [x_1] \cap ([x_1]/[x_3])$ *ne permettent pas de réduire l'ensemble des solution. Dans certains cas, où la contraction optimale est primordiale, il conviendra de recourir à des algorithmes plus complexes reposant sur un découpage du domaine de variation de la fonction en partie simples [Baguenard 2005].*

Algorithme 11 Opérateur de projection simple de la multiplication

```
1: Projection Multiplication(in : sens, in out : [x1], [x2], [x3])
2: si sens==1 alors
3:     [x1] := [x1] ∩ ([x2] * [x3])
4: si sens==-1 alors
5:     [x2] := [x2] ∩ ([x1]/[x3])
6:     [x3] := [x3] ∩ ([x1]/[x2])
```

C.2.6.2 Propagation de contraintes : Forward-Backward

Il existe de nombreux algorithmes de propagation de contraintes. Cependant, les outils utilisés dans ces travaux reposent sur l'utilisation de contracteurs obtenus par propagation puis retro-propagation. Ainsi, une contrainte est décomposée en contraintes primitives, puis chacune de ces contraintes primitives est projetée (forward), puis retro-projetée (backward). Dans la littérature internationale cette opération est appelée "Forward-Backward Propagation". Afin d'illustrer ceci, prenons la contrainte suivante :

$$f(x) \in [y] f(x) = x_1 * (x_2 + \sin(x_3)) \tag{C.38}$$

La décomposition en contraintes primitives donne :

$$\begin{array}{rcl} a_1 & := & sin(x_3) \\ a_2 & := & x_2 + a_1 \\ y & := & x_1 * a_2 \end{array} \tag{C.39}$$

Ce qui donne la propagation suivante :

$$\begin{array}{rcl} [a_1] & := & sin([x_3]) \\ [a_2] & := & [x_2] + [a_1] \\ [y] & := & [y] \cap [x_1] * [a_2] \end{array} \tag{C.40}$$

Finalement, la rétro propagation associée est donnée par :

$$\begin{array}{rcl} [a_2] & := & [a_2] \cap [y]/[x_1] \\ [x_1] & := & [x_1] \cap [y]/[a_2] \\ [x_2] & := & [x_2] \cap [a_2] - [a_1] \\ [a_1] & := & [a_1] \cap [a_2] - [x_2] \end{array} \quad \begin{array}{rcl} [x_3] & := & [x_3] \cap \sin^{-1}([a_1]) \end{array} \tag{C.41}$$

Enfin, le contracteur ainsi évalué est défini par l'algorithme 12. Rappelons que ce contracteur va être utilisé de manière itérative jusqu'à obtenir le pavé d'équilibre.

Algorithme 12 Contracteur de f(x)

1: Contracteur(in out : $[x_1], [x_2], [x_3], [y]$)
2: $[a_1] := sin([x_3])$
3: $[a_2] := [x_2] + [a_1]$
4: $[y] := [y] \cap [x_1] * [a_2]$
5: $[a_2] := [a_2] \cap [y]/[x_1]$
6: $[x_1] := [x_1] \cap [y]/[a_2]$
7: $[x_2] := [x_2] \cap [a_2] - [a_1]$
8: $[a_1] := [a_1] \cap [a_2] - [x_2]$
9: $[x_3] := [x_3] \cap \sin^{-1}([a_1])$

C.2.7 Inversion ensembliste via les contracteurs

Algorithme 13 Algorithme d'inversion ensembliste utilisant les contracteurs

Précondition : Pile : EnsembleDeDéfinition
Précondition : Pile : EnsembleDeDéfinitionContraint
Précondition : Pavé : **[xDépart]**

1: EnsembleDeDéfinition.empiler(**[xDépart]**)
2: **si** EnsembleDeDéfinition $\neq \varnothing$ **alors**
3: **[x]** = Premier élément de la pile EnsembleDeDéfinition
4: $\mathbf{[x]} = C_s[\ \mathbf{[x]}\ ]$
5: **si** $\mathbf{[x]} = \varnothing$ **alors**
6: **[x]** ne contient pas de solution
7: retourner à l'étape 2
8: $\mathbf{[a]} = \bar{C_s}[\ \mathbf{[x]}\ ]$
9: **si** $\mathbf{[a]} \neq [\ \mathbf{[x]}\ ]$ **alors**
10: **[x]** contient des solutions
11: mettre **[x]** \ **[a]** dans la pile EnsembleDeDéfinitionContraint.
12: **si** taille(**[a]**)$< \varepsilon$ **alors**
13: retourner à l'étape 2
14: **sinon**
15: bissecter **[a]** et empiler les deux pavé restants dans la pile EnsembleDeDéfinition.
16: retourner à l'étape 2

Une alternative à l'algorithme SIVIA est utilisée de manière à réduire le nombre de bissections, et ainsi le temps de résolution. Prenons un ensemble de contraintes inégalités non linéaires :

$$\begin{pmatrix} f_1(x_1, ...x_n) & \leq & 0 \\ & ... & \\ f_m(x_1, ...x_n) & \leq & 0 \end{pmatrix} \tag{C.42}$$

Notons $\mathbb{S}$ l'ensemble de tous les vecteurs $\mathbf{x}$ satisfaisant ces contraintes, et $\bar{\mathbb{S}}$ l'ensemble de tous les vecteurs $\mathbf{x}$ ne satisfaisant pas ces contraintes tels que :

$$\begin{array}{rcl} \mathbb{S} & = & \{x \in \mathbb{R}^n | max(f_1, ..., f_m) \leq 0\} \text{ et} \\ \bar{\mathbb{S}} & = & \{x \in \mathbb{R}^n | max(f_1, ..., f_m) > 0\} \end{array} \tag{C.43}$$

En notant $C_{\mathbb{S}}$ et $\bar{C}_{\mathbb{S}}$ les contracteurs associés aux deux ensembles définis dans C.43, l'inversion ensembliste du problème peut être effectuée en utilisant l'agorithme 13.

C.3 Conclusion

Cette annexe regroupe un ensemble d'outils liés à l'analyse par intervalles et à l'inversion ensembliste. Les outils de contraction et deux approches algorithmiques d'inversion ensembliste y ont été rappelées afin de fournir les outils algébriques nécessaires à la compréhension du chapitre 4 de cette thèse.

Annexe D

Parcours de Graham

L'algorithme de Graham est couramment utilisé pour déterminer l'enveloppe convexe d'un nuage de points. Dans un premier temps on recherche le point de plus petite ordonnée. En cas d'égalité, on choisira celui qui a la plus petite abscisse. Ce point noté P servira de point de départ. L'ensemble des points est ensuite trié en fonction de l'angle entre chacun d'eux et l'axe des abscisses. Un tableau T est alors obtenu avec les points ainsi triés.

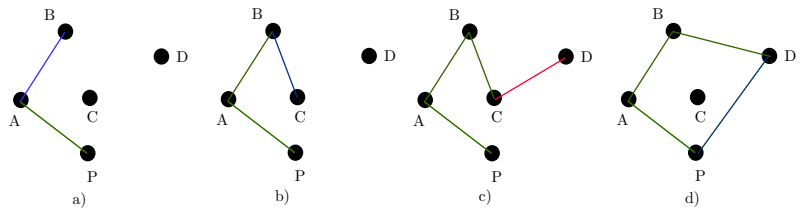


Fig. D.1 – Illustration du fonctionnement du parcours de Graham

Algorithme 14 Algorithme de Graham, le produit vectoriel est utilisé pour définir un "tournant à gauche" ou un "tournant à droite". L'élément Pile contient les points de l'enveloppe convexe à la fin de la procédure.

1: Trouver le point de départ P
2: Trier les points par angle (les points d'angle égaux sont triés en fonction de leur ascisse)
3: T[1] est le point de départ (P)
4: Pile.empiler(T[1])
5: Pile.empiler(T[2])
6: **pour** $i = 3$ à taille du tableau T **faire**
7: **tant que** Taille de la pile $\leq$ 2 et ProduitVectoriel(Pile.AvantDernier, Pile.Dernier,T[i])$\leq$ 0 **faire**
8: Pile.dépiler
9: Pile.empiler(T[i])

On considère ensuite chacune des séquences de trois points successifs dans le tableau T, vus comme deux couples. L'angle formé par ces paires de couples est

ensuite considéré (on parlera de tournant à droite et de tournant à gauche). Ainsi, si il s'agit d'un tournant à droite, l'avant dernier point considéré ne fait pas partie de l'enveloppe convexe et doit être enlevé du tableau T. A l'inverse, un tournant à gauche ne modifie pas la structure du tableau, et on passe simplement à la séquence de points suivante. Par conséquent, une fois toutes les séquences considérées, le tableau contiendra seulement les points qui appartiennent à l'enveloppe convexe pour une complexité algorithmique dépendant de l'algorithme de tri utilisé. La figure D.1 propose une illustration du fonctionnement de cette procédure qui est détaillée par l'algorithme 14.

Annexe E

Algorithmes utilisés pour la contraction

Afin de contracter efficacement un pavé pour une enveloppe convexe H_i donnée, plusieurs procédures sont utilisés successivement par l'algorithme 15. Tout d'abord, un appel de l'algorithme "Project" présenté en 16 est effectué pour projeter un pavé $[\mathbf{q}]$ dans le plan x, y de chacune des roues. A la suite de quoi, on obtient un intervalle sur X et sur Y, autrement dit un pavé en deux dimensions pour chaque roue du robot.

Ensuite, l'algorithme "C_{Vect_i}" est appelé successivement pour chaque produit vectoriel de la contrainte E associée à l'enveloppe i afin de contracter les pavés associés aux limites possibles de déplacement des roues. Il est présenté en 17

L'utilisation de l'algorithme "$C_{longueur}$", présenté en 18 contracte une fois de plus les pavés X, Y des roues du robot en fonction de l'encadrement désiré pour le périmètre de l'enveloppe convexe H_i.

Enfin, une fois les intervalles X et Y contractées par l'intermédiaire des contraintes liées à l'enveloppe étudiée, l'algorithme "Inverse" présenté en 19 permet de retrouver un pavé contracté $[\mathbf{q}]$ dans l'espace des coordonnées articulaires.

Notons, que ces algorithmes ont été implémentés en machine grâce à la librairie de calcul par intervalles Filib ++ disponible gratuitement à l'adresse `http://www2.math.uni-wuppertal.de/~xsc/software/filib.html`.

Algorithme 15 Contracteur d'enveloppe : $C_s([\vec{q}])$

1: $[\vec{X},\vec{Y}]$=projection($[\vec{q}]$)
2: **pour tout** les produit vectoriels **faire**
3: $[\vec{X},\vec{Y}] = C_{Vect_i}([\vec{X},\vec{Y}])$
4: $[\vec{X},\vec{Y}] = C_{length}([\vec{X},\vec{Y}])$
5: $[\vec{q}]$ =inverse($[\vec{X},\vec{Y}]$)

Algorithme 16 Project q

X[0]=[0,0]
Y[0]=[0,0]
X[1]=[B,B]
Y[1]=[0,0]
Par définition, les indices impairs de $\vec{q}$ représentent des angles
et les indices pairs, des longueurs.
pour $j = 0$ à taille($\vec{q}$)/2 pas de 1 **faire**
  pour $i = 0$ à j pas de 1 **faire**
    On remplit le tableau Q avec les valeurs d'angles correspondants
    $Q[0] = q[1], Q[2] = q[1] + q[3], Q[3] = q[1] + q[3] + q[5]$ etc...
    $Q[j] = Q[j] + q[i \times 2 + 1]$
On commence à projeter l'espace de définition pour les roues 2 à n
j=2
pour $i = 0$ à taille($\vec{q}$) pas de 2 **faire**
  $a[j] = cos(Q[j-2])$
  $a'[j] = cos(Q[j-2])$
  $b[j] = q[i] * a[j]$
  $b'[j] = q[i] * a'[j]$
  $X[j] = X[j-1] + b[j]$
  $Y[j] = Y[j-1] + b'[j]$
  $j++$

Algorithme 17 Contracteur de produit vectoriel : C_{Vect_i}

Précondition : $\vec{X}$, $\vec{Y}$, *domaine*.

1: **répéter**
2: %*Propagation "Forward"*
3: a=X[1]-X[0] ;
4: b=Y[2]-Y[1] ;
5: c=Y[1]-Y[0] ;
6: d=X[2]-X[1] ;
7: e=a*b ;
8: f=c*d ;
9: z=intersect(*domaine*,e-f) ;
10: %*Propagation "Backward"*
11: e=intersect(e,z+f) ;
12: f=intersect(f,-z+e) ;
13: c=intersect(c,f/d) ;
14: d=intersect(d,f/c) ;
15: a=intersect(a,e/b) ;
16: b=intersect(b,e/a) ;
17: X[2]=intersect(X[2],d+X[1]) ;
18: X[1]=intersect(X[1],-d+X[2]) ;
19: Y[2]=intersect(Y[1],c+Y[0]) ;
20: Y[0]=intersect(Y[0],-c+Y[1]) ;
21: Y[2]=intersect(Y[2],b+Y[1]) ;
22: Y[1]=intersect(Y[1],-b+Y[2]) ;
23: X[1]=intersect(X[1],a+X[0]) ;
24: X[0]=intersect(X[0],-a+X[1]) ;
25: **jusqu'à** ce que $\vec{X}$ et $\vec{Y}$ ne se contractent plus

Algorithme 18 Contracteur dédié à la taille de la chenille : $C_{longueur}$

Précondition : *ordre*, $\vec{X}$, $\vec{Y}$, L_{min} et L_{max}.

pour $i = 0$ à taille de *ordre* -1 **faire**
  a[i]=X[*ordre*[i+1]]-X[*ordre*[i]] ;
  b[i]=Y[*ordre*[i+1]]-Y[*ordre*[i]] ;
  c[i]=a[i]*a[i] ;
  d[i]=b[i]*b[i] ;
  e[i]=c[i]+d[i] ;
  f[i]=sqrt(e[i]) ;
a[i+1]=X[*ordre*[0]]-X[*ordre*[i+1]] ;
b[i+1]=Y[*ordre*[0]]-Y[*ordre*[i+1]] ;
c[i+1]=a[i+1]*a[i+1] ;
d[i+11]=b[i+1]*b[i+1] ;
e[i+1]=c[i+1]+d[i+1] ;
f[i+1]=sqrt(e[i+1]) ;
pour $i = 0$ à taille de *ordre* **faire**
  l=l+f[i] ;
%*Contraction de* $L_{min} < L_i < L_{max}$
si sens==0 **alors**
  %$g = -L_{min} + l > 0$ *et* $h = L_{max}$ $l > 0$
  g=-lmin+l ;
  h=lmax-l ;
  g=$[0; +\infty] \cap g$;
  h=$[0; +\infty] \cap h$;
  %*Propagation "Backward"*
  $l = l \cap -h + lmax$;
  $l = l \cap g + lmin$;
sinon
  si sens==1 **alors**
    %*Contraction de* $L_i > L_{max}$
    h=-lmax+l ;
    h=$[0; +\infty] \cap h$;
    %*Propagation "Backward"*
    $l = l \cap h + lmax$;
  sinon
    %*Contraction de* $L_i < L_{min}$
    g=lmin-l ;
    g=$[0; +\infty] \cap g$;
    %*Propagation "Backward"*
    $l = l \cap -g + lmin$;
pour $i =$ taille de *ordre* à 0 **faire**
  temp=0 ;
  pour $j =$ taille de *ordre* à 0 **faire**
    si j!=i **alors**
      temp=temp+f[j] ;
  $f[i] = f[i] \cap l - temp$;
pour $i =$ taille de *ordre* à 0 **faire**
  $e[i] = e[i] \cap sqr(f[i])$;
  $c[i] = c[i] \cap e[i] - d[i]$;
  $d[i] = d[i] \cap e[i] - c[i]$;
  $bprim[i] = b[i]$;
  $aprim[i] = a[i]$;
  $bprim[i] = bprim[i] \cap -sqrt(d[i])$;
  $aprim[i] = aprim[i] \cap -sqrt(c[i])$;
  $b[i] = b[i] \cap sqrt(d[i])$;
  $a[i] = a[i] \cap sqrt(c[i])$;
  $b[i] = b[i] \cup bprim[i]$;
  $a[i] = a[i] \cup aprim[i]$;
  si i==taille de *ordre* **alors**
    X[*ordre*[0]]=X[*ordre*[0]]$\cap$ a[i]+X[*ordre*[i]] ;
    X[*ordre*[i]]=X[*ordre*[i]]$\cap$-a[i]+X[*ordre*[0]] ;
    Y[*ordre*[0]]=Y[*ordre*[0]]$\cap$b[i]+Y[*ordre*[i]] ;
    Y[*ordre*[i]]=Y[*ordre*[i]]$\cap$-b[i]+Y[*ordre*[0]] ;
  sinon
    X[*ordre*[i+1]]=X[*ordre*[i+1]]$\cap$a[i]+X[*ordre*[i]] ;
    X[*ordre*[i]]=X[*ordre*[i]]$\cap$-a[i]+X[*ordre*[i+1]] ;
    Y[*ordre*[i+1]]=Y[*ordre*[i+1]]$\cap$b[i]+Y[*ordre*[i]] ;
    Y[*ordre*[i]]=Y[*ordre*[i]]$\cap$-b[i]+Y[*ordre*[i+1]] ;

Algorithme 19 Inverse X et Y

pour $i = nb_{roues}$ à 1 pas de -1 **faire**
  $b'[i] = b'[i] \cap (Y[i] - Y[i-1])$
  $Y[i-1] = Y[i-1] \cap (Y[i] - b'[i])$
  $a'[i] = a'[i] \cap (\frac{c[i]}{q[j-1]})$
  $q[j-1] = q[j-1] \cap (\frac{c[i]}{a'[i]})$
  $b[i] = b[i] \cap (X[i] - X[i-1])$
  $X[i-1] = X[i-1] \cap (X[i] - b[i])$
  $a[i] = a[i] \cap (\frac{b[i]}{q[j-1]})$
  $q[j-1] = q[j-1] \cap (\frac{b[i]}{a[i]})$
pour $i = 1$ à taille($\vec{q}$) pas de 2 **faire**
  $q[i] = ForwardBackwardCosinus()$

Annexe F

Algorithmes du plus court chemin

F.1 Algorithme de Dijkstra

Cet algorithme n'est valable que pour les graphes à validation positive (ne contenant pas de circuit négatif). Deux étapes principales peuvent être relevées. Tout d'abord, on définit la distance entre le somment de départ et tous ses sommets voisins. Ensuite, le sommet voisin associé à la plus petite valeur noté v (le plus proche) est sélectionné et mis de côté. On s'assure alors que les distances associées à chacun des voisins de v sont plus faibles que celles nouvellement trouvées en passant par le sommet v, si tel n'est pas le cas, les poids sont alors mis à jour. Cette boucle principale est alors ré-itérée jusqu'à épuisement des sommets.

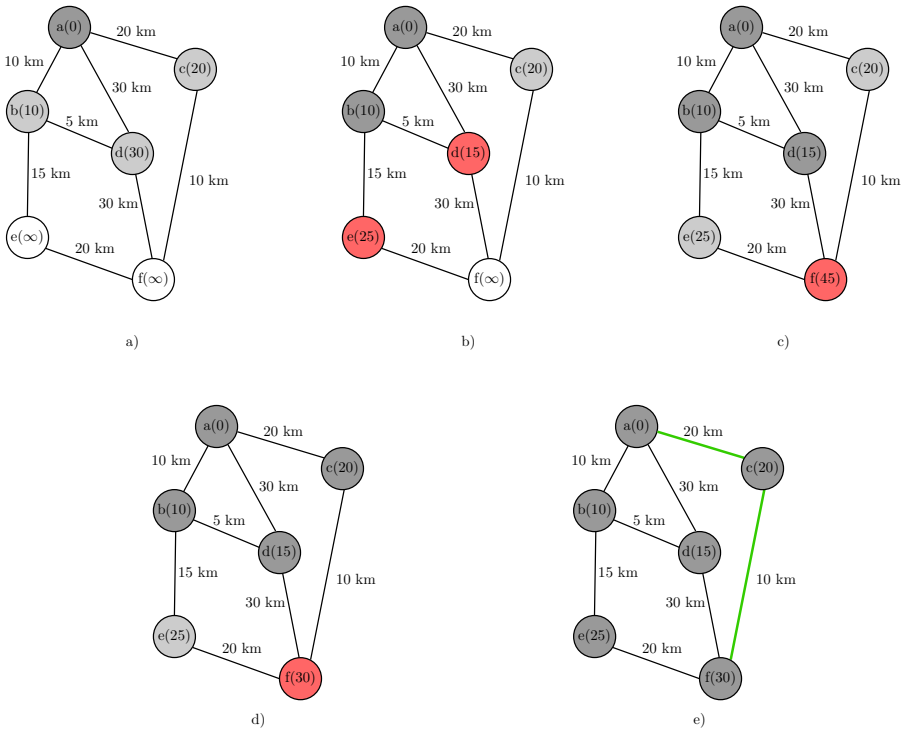


Fig. F.1 – Exemple de résolution via Dijkstra.

La figure F.1 présente une illustration du déroulement de cette procédure. La

légende est la suivante :

- a) : la ville "a" constitue le point de départ. Le coût relatif au voyage vers les sommets voisins est mis à jour,
- b) : la distance la plus petite est celle qui mène à la ville "b". On la sélectionne et on met à jour les distances de ses voisins (ici, "d" et "e"),
- c) : la plus petite distance suivante se retrouve être associée à la ville "d". Les voisins sont mis à jour, ce qui donne un chemin vers la ville d'arrivée ("f") d'une distance de 45 km,
- d) : la ville "c" est associée à une distance encore plus petite, sa sélection ouvre un chemin plus court vers la ville d'arrivée,
- e) : la ville "e" ne permet pas de réduire la taille du chemin passant par "c", le plus court chemin a donc été trouvé.

L'algorithme 20 propose une implémentation de ce comportement.

Algorithme 20 Algorithme de Dijkstra. Le tableau T contient la liste de tous les sommets du graphe, et le tableau coût, la distance depuis le point de départ associé à chaque sommet. A la fin de l'algorithme, le tableau prédécesseurs est utilisé pour retrouver le plus court chemin.

```
1: Trouver le point de départ P
2: tant que T n'est pas vide faire
3:   v=ExtraireMinimum(T,coût)
4:   pour tout les voisins u de v faire
5:     si coût(u)>coût(v)+distance(v,u) alors
6:       coût(u)=coût(v)+distance(v,u)
7:       Prédécesseur(u)=v
```

F.2 Algotithme A-star

L'algorithme A-star, ou A-étoile en français, fonctionne d'une manière équivalente. Il subsiste cependant 2 différences principales entre ces deux procédures. Tout d'abord en ce qui concerne l'évaluation de la distance. Alors que Dijkstra se concentre sur la distance entre le sommet considéré et ses voisins pour associer un coût à chacun des sommets, l'utilisation de A-Star suggère de prendre aussi en compte la distance relative entre les voisins et le point d'arrivée. Cela permet notamment de s'efforcer d'aller en ligne droite pour minimiser la taille du chemin. La seconde différence fondamentale réside dans le fait qu'une fois le sommet final atteint, la boucle principale n'est pas ré-itérée, ainsi, le chemin trouvé ne peut être qualifié d'optimal. Cependant, cette modification de l'algorithme de Dijkstra présente des résultats supérieurs en terme de temps de calcul lorsque l'on travaille avec des graphes relativement denses.

Bibliographie

[Agency 2005] State Agency. *Improvement of the Emergency Risk Management through Secure Mobile Mechatronic Support to Bomb Disposal and Rescue Operations*. Specific Targeted Research Project 511492, Internal report of the State Agency for Civil Protection, 2005. 9

[Anastasios I. Mourikis 2007] Stergios I. Roumeliotis Daniel M. Helmick et Larry Matthies Anastasios I. Mourikis Nikolas Trawny. *Autonomous Stair Climbing for Tracked Vehicles*. The International Journal of Robotics Research, vol. 26, no. 737, 2007. 73

[Baguenard 2005] X. Baguenard. *Propagation de contraintes sur les intervalles Application à l'étalonnage des robots*. PhD thesis, Université d'Angers, 2005. 121, 122

[Beekmans 2007] Gerard Beekmans. Linux from scratch v 6.6. M. Burgess and B. Dubbs, 2007. 18

[Carlson 2003] J. Carlson et R. R. Murphy. *Reliability Analysis of Mobile Robots*. International Conferenee on Robotics and Automation, pages 274–281, September 2003. Taiwan - Taipei. 8

[Casper 2003] J. Casper et R. R. Murphy. *Human-Robot Interactions During the Robot-Assisted Urban Search and Rescue Response at the World Trade Center*. IEEE Transactions on systems, man, and cybernetics, vol. 33, no. 3, pages 367–384, June 2003. 7

[Craig 1989] J. J. Craig. Introduction to robotics mechanics and control. Silma, 2 édition, 1989. 25

[Cullogh 1943] W.S. Mc Cullogh et W. Pitts. *A logical calculus of the ideas immanent in nervous activity*. Bulletin of Mathematical Biophysics 5, 1943. 73

[Daniel M. Helnick Stergios I. Roumeliotis 2002] Michael C. McHenry et Larry Matthies Daniel M. Helnick Stergios I. Roumeliotis. *Multi sensor, High speed Autonomous Stair Climbing*. IEEE International conference on Robots and Systems, 2002. 73

[de Berg 2000] Mark de Berg, Marc van Kreveld, Mark Overmars et Otfried Schwarzkopf. Computational geometry. Springer-Verlag, 2000. 69

[Djoudi 2007] Dalila Djoudi. *Contribution à la commande d'un robot bipède*. PhD thesis, Ecole centrale de Nantes, 2007. 29, 103

[F. Mondada 2004] G. C. Pettinaro A. Guignard I. W. Kwee D. Floreano J.-L. Deneubourg S. Nolfi L. Maria Ambardella F. Mondada et M. Dorigo. *SwarmBot : A New Distributed Robotic Concept*. Autonomous Robots, pages 193–221, 2004. 12

[G. Clement 1987] E. Villedieu G. Clement. *Variable geometry track vehicle.* Us patent, 1987. 10

[Goswami 1999] A. Goswami. *Postural stability of biped robots and foot rotation indocator fri point.* The Internationnal journal of Robotics Reserach, vol. 18, no. 6, 1999. 32

[Graham 1972] R.L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. Information Processing Letters, 1972. 47

[Grey 1963] Walter W. Grey. The living brain. W. W. Norton, 1963. 5

[Hansen 1992] E. Hansen. Global optimization using interval analysis. Marcal Dekker, Inc., 1992. 107, 108

[Hebb 1949] D.O. Hebb. *The Organisation of Behaviour.* Wiley, New York, 1949. 74

[Ion Lungu 2008] Simona Noveanu Olimpio Tatar Ion Lungu Dan Mândru. *Docking Mechanism Actuated by Shape Memory Alloy Actuator.* Mecatronics, 2008. 97

[Iwamoto 1983] T. Iwamoto et H. Yamamoto. *Variable configuration track laying vehicle (US Patent)*, March 1983. 5, 10

[J. Kim 2002] W. K. Chung Y. Youm J. Kim et B. H. Lee. *Real time ZMP Compensation Method using Null Motion for Mobile Manipulators.* pages 1967–1971. Proceedings of the 2002 IEEE Internationnal Conference on Robotics ans Automation, May 2002. Washington, DC. 31

[J. L. Paillat 2008a] L. Hardouin J. L. Paillat P. Lucidarme. *Original design of Unmanned Grounded Vehicle (UGV) for exploration in rough terrain.* Mecatronics, 2008. 18

[J. L. Paillat 2008b] P. Lucidarme L. Hardouin J. L. Paillat. *Variable Geometry Tracked Vehicle (VGTV) prototype : conception, capability and problems.* pages 115–126. Humans Operating Unmanned Systems (HUMOUS) conference, September 2008. France - Brest. 17

[J. L. Paillat 2009] L. Hardouin J. L. Paillat P. Lucidarme. *Variable Geometry Tracked Unmanned Grounded Vehicle : model, stability and experiments.* ICINCO, 2009. 71

[J. L. Paillat 2010] L. Hardouin J. L. Paillat P. Lucidarme. *Original design of Unmanned Grounded Vehicle (UGV) for exploration in rough terrain.* Advanced Journal of Robotics, 2010. 35

[J. Liu 2005] Y. Wang S. Ma J. Liu et B. Li. *Analysis of stairs-climbing ability for a tracked reconfigurable modular robot.* pages 36–41. International Workshop on Safety, Security and Rescue Robotics, June 2005. Japan-Kobe. 9

[Jaulin 2000] L. Jaulin. *Interval constraint propagation with application to bounded-error estimation.* Automatica, vol. 36, 2000. 56

[Jaulin 2001] L. Jaulin. *Path planning using intervals and graphs.* Reliable computing, vol. 7, no. 1, 2001. 68

[K. Osuka 2006] T. Doi K. Osuka et Y. Yokokouji. *In rubble Robot system for USAR under debris.* pages 3439–3442. SICE-ICASE Internationnal Joint Conference, October 2006. Korea - Bexco. 9

[Khalil 2004] W. Khalil et E. Dombre. Modeling, identification and control of robots. Kogan Page Science, 2 édition, 2004. 27

[Kinugasa 2008] T. Kinugasa, Y. Otani, T. Haji, K. Yoshida, K. Osuka et H. Amano. *A proposal of Flexible Mono-tread Mobile Track.* pages 1642–1647. Internationnal Conference on Intelligent Robots and Systems, Sept 2008. France - Nice. 11

[L. Jaulin 1993] E. Walter L. Jaulin. *Set Inversion Via Interval Analysis for Non Linear bounded error estimation.* Automatica, vol. 29, no. 4, 1993. 56, 68, 113

[L. Sung Kyun 2005] P. Dong Il K. Yoon Keun K. Byung-Soo J. Sang-Won L. Sung Kyun. *Variable geometry single-tracked mechanism for a rescue robot.* IEEE International Workshop on Safety, Security and Rescue Robotics, pages 111–115, 2005. 10

[M. Guarnieri 2004] P. Debenest T. Inoh E. Fukushima S. Hirose M. Guarnieri. *Development of Helios VII : an arm-equipped tracked vehicle for search and rescue operations.* pages 39–45, 2004. Japan - Sendai. 10

[Minsky 1969] M. Minsky et S. Papert. *Perceptron.* The MIT Press, 1969. 74

[Misawa 1997] R. Misawa. *Stair-climbing crawler transporter.* Us patent, 1997. 10

[Mitchell 1997] T. M. Mitchell. Machine learning. Mc Graw-Hill International Editions, 1997. 84

[Moore 1966] R. E. Moore. Interval analysis. Prentice-Hall, Englewood Cliffs, 1966. 107

[Nolfi 2000] S. Nolfi et D. Floreano. Evolutionary robotics. The MIT Press, 2000. 79

[O. Lévêque 2000] L. Jaulin E. Walter O. Lévêque D. Meizel. *Set inversion for chi-algorithms, with application to guaranteed robot localization.* Math. Comput. Simulation, vol. 52, 2000. 56

[Rosenblatt 1958] F. Rosenblatt. *The Perceptron : a Probabilistic Model for Information Storage and Organisation in the Brain.* Psychological Review, pages 386–408, 1958. 74

[Rumelhart 1986] D.E. Rumelhart et J.L. Mc Clelland. *Parallel Distributed Processing.* The MIT Press, 1986. 75

[S. Kajita 2003] F. Kanehiro K. Kaneko K. Fujiwara K. Harada K. Yokoi S. Kajita et H. Hirukawa. *Biped walking pattern generation by using preview control of zero moment point.* pages 1620–1626. Proceedings of the 2003 IEEE Internationnal Conference on Robotics ans Automation, September 2003. Taiwan - Taipei. 31

[Sébastien Lengagne 2007] Nacim Ramdani et Philippe Fraisse. Sébastien Lengagne. *Guaranteed computation of constraints for safe path planning.* IEEE-RAS 7th International Conference on Humanoid Robots, 2007. 42

[Sébastien Lengagne 2008] Nacim Ramdani et Philippe Fraisse. Sébastien Lengagne. *A new method for generating safe motions for humanoid robots.* IEEE-RAS 7th International Conference on Humanoid Robots, 2008. 42

[Tom Frost 2002] Christopher Norman Scott Pratt Tom Frost et Brian Yamauchi. *Derived Performance Metrics and Measurements Compared to Field Experience for the PackBot.* 2002. USA - Gaithersburg. 9

[U. Saranli 2001] M. Buehler U. Saranli et D. E. Koditschek. *RHex - a simple and highly mobile hexapod robot.* Journal of Robotics Research, pages 616–631, July 2001. 12

[Vincent 2007] I. Vincent et M. Trentini. *Shape-shifting Tracked Robotic Vehicle for complex terrain navigation : Characteristics and architecture.* Technical memorandum, Defence R and D Canada, December 2007. 10

[Voyles 2000] R. M. Voyles. *Terminatorbot : A robot with dual use arms for manipulation and locomotion.* pages 61–66. Internationnal Conference on Robotics and Automation, April 2000. United States - San Francisco. 12

[Vukobratovic 2004] M. Vukobratovic et B. Borovac. *Zero-momment point - thirty five years of its life.* Internationnal journal of humanoid Robotics, vol. 1, no. 1, pages 157–173, 2004. 31, 32

[Xiaoye Lu 2005] Roberto Manduchi Xiaoye Lu. *Detection and Localization of Curbs and Stairways Using Stereo Vision.* IEEE International conference on Robotics and Automation, 2005. 73

[Yalin Xiong 2000] Larry Matthies Yalin Xiong. *Vision guided Autonomous Stair Climbing.* IEEE International conference on Robotics and Automation, 2000. 72

[Young 1932] R. Young. The algebra of many-valued quantities. Mathematische Annalen, 1932. 107

Conception et contrôle de robots à géométrie variable : applications au franchissement d'obstacles autonome.

Résumé :

Les travaux de cette thèse se placent dans le cadre de la robotique mobile terrestre. Un prototype innovant de robot à géométrie variable capable de franchir divers obstacles (escaliers, trottoirs...) est présenté et analysé. Le retour d'expérience nous donne des pistes pour évoluer vers un robot autonome.

L'étude de la déformation d'un robot muni de n degrés de liberté est présentée et formalisée comme un problème de satisfaction de contraintes. L'objectif est d'actionner les articulations du robot tout en conservant la tension de la chenille qui transmet l'effort des moteurs de propulsion. Des outils d'analyse par intervalles sont utilisés pour proposer une solution.

Un autre aspect du contrôle d'un robot à géométrie variable est ensuite étudié pour fournir une méthode autonome de déformation lors de phases de franchissement d'obstacles. Un réseau de neurones est entraîné via un algorithme génétique dans le but de franchir un escalier. Le prototype existant nous permet de valider expérimentalement ces résultats.

Mots clés : robotique mobile, VGSTV, capacité de franchissement, CSP, inversion ensembliste, contracteurs, franchissement autonome, réseau de neurones.

Conception and control of variable geometry robots : autonomous obstacles clearing applications.

Abstract :

This work focuses on mobile robotics. An innovative variable geometry vehicle able to pass over several obstacles (staircase, curb...) is presented and analysed. Experiment feedbacks provide some clues to evolve into a fully autonomous robot.

A study about the deformation of a robot equipped with n degrees of freedom is presented and formalised as a constraints satisfaction problem. The aim is to provide a way to change the shape of the robot while keeping the tracks tightened. Interval analysis tools are used to propose a solution.

An other variable geometry robot control method is then studied to lead to autonomous staircase climbing. A neural network is trained through a genetic algorithm in order to climb a staircase autonomously. Our prototype is used to validate those results.

Keywords : mobile robotics, VGSTV, clearing capability, CSP, contractors, autonomous clearing, neural network, genetic algorithms.