République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université A/Mira de Bejaïa
Faculté de la technologie
Département De Génie Electrique

Mémoire de Master

En vue de l'obtention du diplôme de master en électronique

Spécialité : Automatique

Thème :

Planification de chemins pour un robot à segments polyédriques dans une scène 3D à obstacles polyédriques en utilisant une carte de route probabiliste

Présentés par :

Benbara Mabrouk

Brahami Sabiha

Soutenue le 13 / 09 / 2020 devant le jury composé de :

Mr	B.Mendil	Président
Mr	H. HADDAR	Promoteur
Mr	M.Sadji	Examinateur

Année Universitaire : 2019 / 2020

REMERCIEMENTS

La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes à qui nous voudrions témoigner toute notre reconnaissance.

Nous voudrions tout d'abord adresser toute notre gratitude à l'encadreur de ce mémoire, Mr.Hadder Hocine pour sa patience sa disponibilité et ses conseils qui ont contribué à et monter notre réflexion.

Nous sommes profondément reconnaissants aux membres de jury pour avoir participé à l'évaluation de ce travail.

Nous voudrions aussi remercier nos professeurs qui ont fourni les outils nécessaires à la réussite dans notre étude universitaire.

Enfin nous tenons à exprimer notre reconnaissance envers les amis et collègue qui nous ont apporté leur support moral et intellectuel tout au long de notre démarche.

Merci à vous tous.

DEDICACES

Du profond de mon cœur, je dédie ce travail à tous ceux qui me sont chers

À ma chère maman

Aucune dédicace ne saurait exprimer mon respect mon amour et ma considération pour les sacrifices que vous avez consenti à mon égard. Que ce modeste de travail soit l'exaucement de vous vœux tant formulés, le fruit de vos innombrables sacrifices. Puisse Dieu vous accorder santé et bonheur et longue vie.

À la mémoire de mon Père.

A mani Hnifa et toute ma famille avec tous mes sentiments de respect, d'amour, de gratitude et de reconnaissance pour leurs encouragements et leurs soutiens.

A toutes mes copines Basma et Sara et surtout ma chère amie Dida qui est toujours là à l'écoute je te remercie du fond du cœur.

Sabiha.

A mes chers parents, pour tous leurs sacrifices et leur soutien

A toute ma famille pour leur appui et leur encouragement

Que ce travail soit l'accomplissement de vos vœux tant allégués

Mabrouk

Table des matières

LISES DES FIGURES

INTRODUCTION GENERAL

INTRODUCTION GENERAL

Les développements récents de la robotique visent à accroitre l'autonomie de ces systèmes. Cette notion d'autonomie représente une composante importante caractérisant les robots de la nouvelle génération. L'objectif est de développer des méthodes algorithmiques à la fois performantes et effectives en pratique qui permettent le calcul automatique de trajectoires pour des systèmes mécaniques, malgré la forte combinatoire de ces problèmes. [5].Plusieurs méthodes de planification de mouvement existent, grâce à leur efficacité, ont permis d'étendre le champ d'application de la planification à une grande variété de domaines, allant de la robotique, à la CAO (conception assistée par ordinateur), la bio-informatique ou encore l'animation graphique.

Parmi ces méthodes le RPP (Randomized Path planner) qui est une méthode de champs de potentiel utilisant des pas aléatoires pour essayer d'éviter les minimums locaux. Quoique cette méthode travail bien, il y a des situations où elle a des performances médiocres. L'autre méthode c'est la carte de route probabiliste 'PRM' ; dont cette dernière fera l'objet de notre travail.

Cette approche utilise le choix aléatoire de configuration durant la phase prétraitement, dite phase d'apprentissage, pour construire un graphe dans l'espace de configuration. Dans ce graphe les nœuds représentent des configurations hors collision et un arc connecte deux nœuds si un planificateur local arrive à trouver un chemin hors collision entre les deux Configurations. [5]

Au début un ensemble de configuration candidates est échantillonné de façon aléatoire si une configuration et hors collision, elle est gardée sinon elle est ignorée, c'est le test de collision. Ensuite en utilisant des techniques spéciales pour définir la distance entre une configuration à une autre Configuration voisine. Un planificateur local permet de déterminer si un chemin entre ces deux Configurations existe ou non. Cette planification consiste à connecter les configurations initiale et finale à la carte de route, ensuite chercher dans le graphe un chemin entre elles.

L'objectif de ce mémoire et de réaliser sous MATLAB un planificateur de la trajectoire basée sur la méthode de carte de route probabiliste dans une scène en 3D, ou les obstacles et le robot ont des formes polyédriques. Nous allons choisir l'échantillonnage aléatoire selon une distribution uniforme c'est la technique la plus simple à implémenter, et pour le planificateur local nous utiliserons un planificateur basé sur le calcul des intervalles interdits des coordonnées articulaires, ceci permis la déduction de l'espace de configuration hors collision sous forme d'une collection de cellules. Pour effectuer la recherche de chemin dans le graph qu'est la PRM, nous allons utiliser l'algorithme de profondeur d'abord (Depth-first).

Ce mémoire comprend trois chapitres :

Dans le premier chapitre nous présenterons quelques généralités de la robotique et une introduction sur la modélisation des robots.

Le deuxième chapitre expose les stratégies de connexion PRM et d'échantillonnage ainsi l'intégration des planificateurs.

Enfin, le troisième chapitre présente notre travail. On y trouve un exposé pour chaque méthode utiliser et les résultats obtenus.

Nous terminerons par une conclusion générale.

I. CHAPITRE Généralité sur les bras de robots

I.1 Introduction

Au début, les robots sont faits pour remplacer les ouvriers dans une chaine industrielle. Actuellement, les applications sont diverses et la robotique est une discipline à part entière qui exploite plusieurs outils et fait appel à un ensemble de techniques pour la conception et la réalisation des machines automatiques de plus en plus intelligentes.

I.2 Histoire de robotique

• 1920-1941 : Apparition du mot robot et robotique :
L'origine du mot robot provient de la langue tchèque dans laquelle sont ancêtre "rabota" signifie travail forcé. Il a été introduit, en 1920, par l'écrivain tchèque Karel Capek dans la pièce de théâtre Rossums Universel Robots. Le terme robotique a été employé pour la première fois par Asimov en 1941.

• 1961 : Unimation, le 1er robot industriel :
Descendant direct des télémanipulateurs développés pour les besoins du nucléaire. Il est vendu à partir de 1961 par la société américaine Unimation (devenu Stäubli Unimation), créée par George Devol et Joseph Engelberger. Il est utilisé pour la première fois sur les lignes d'assemblage de General Motors. Ce robot, grâce à son bras articulé de 1,5 tonne, était capable de manipuler des pièces de fonderie pesant 150 kg.

En 1966, l'entreprise Unimation continue de développer des robots et élaborent notamment des robots permettant de faire d'autres tâches, comme des robots de manipulation matérielle ou encore des robots conçus pour la soudure ou pour d'autres applications de ce genre.

Figure I-1Robot Unimate.

• En 1972 : 1ere chaîne de production robotisée :
Nissan ouvre la première chaîne de production complètement robotisée, Selon une étude de l'IFR, 2142 millions de robots ont été fabriqués entre les années 60 et la fin 2010, Les analystes estiment qu'aujourd'hui, de 1 à 1,3 million de robots travaillent pour nous dans les usines dans le monde.

En 1978 un nouveau robot est conçu par Unimation Inc avec l'aide de General Motors. Ensemble ils conçurent le robot PUMA 500. Le robot PUMA (Programmable Universel Machine for Assembly) a été conçu par Vic Schienman et fut financé par General Motors. [1] [2]

Figure I-2 chaine de production robotisée.

Figure I-3 RobotPUMA.

• 1970-1980 : Premiers robots mobiles :
- 1960-64: Ouverture des laboratoires d'Intelligence Artificielle au M.I.T., Stanford Research Institute (SRI), Sanford Université, Université of Edinburgh.
- Fin des années 60: Mise en place de "Shakey" premier robot mobile Intégrant perception, planification et exécution.
- 1970 : Sandford Cart.
- 1977 : premier robot mobile français HILARE au LAAS (CNRS Toulouse).

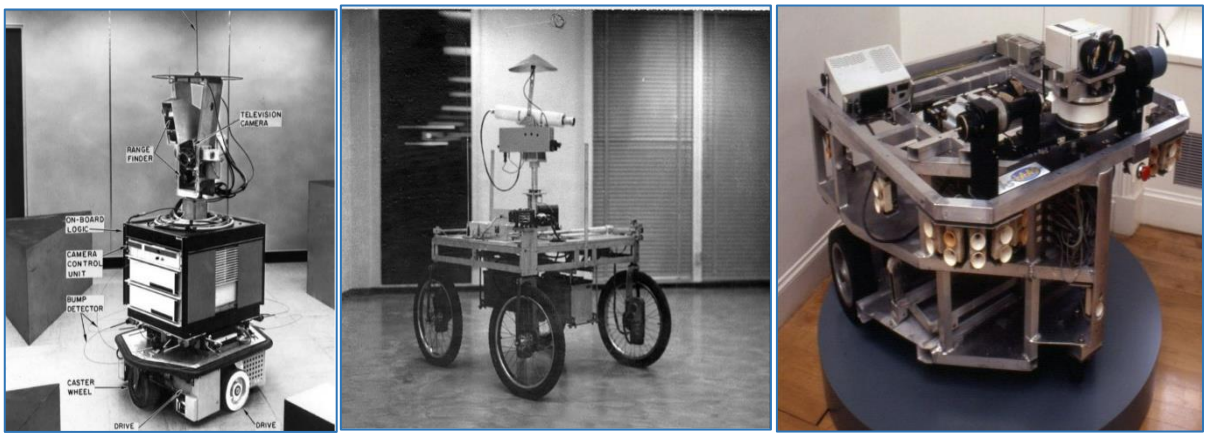


Figure I-4Premiers robots mobiles.

• 1980-1990 : l'intelligence Artificielle : L'intelligence artificielle et l'apprentissage automatique pour une vaste gamme d'applications, de la reconnaissance faciale à la détection des maladies dans les imageries médicales jusqu'aux compétitions mondiales dans des jeux tels que les échecs et Go. Les nations investissent d'ores et déjà d'énormes montants d'argent dans le secteur, qui doit être au cœur de l'avancée de la technologie et qui sera déterminant dans la croissance économique du monde entier.
• 1990-2000: Essor de la robotique mobile :
-1992 : Mise en place de la compétition annuelle AAAI sur la robotique mobile.
-1995 : Mise en place de la RoboCup (lien vidéo).
-1997 : premier robot mobile extraplanétaire sur Mars.
-1999 : Lancement d'Aibo

Figure I-5 Essor de la robotique.

• Depuis 2000 : Exploration :
- 2003 : Projet "Mars Exploration Rover" (Spirit & Opportunity).
- 2009 : projet "Mars Science Laboratory" succédant au projet Rover, envoi prévu de Curiosity fin 2011.

Figure I-6 Spirit & Opporunity.

Figure I-7Curiosity.

• Depuis 2000 : Démocratisation des robots: robots.
- 2000 : Lancement d'Asimo.
- Diversification des compétitions de robotique.
- Utilisation de drones en situation réelle (Irak...).
- 2006 : le projet Aibo n'est plus assez rentable, fin de la production.
-2009: robot Nao utilisé à la RobocupSoccer.

Figure I-8 Les robot Asimo et Aibo.

• Entre 2010 et 2019 :

Cette période est riche invention et on ne peut pas mentionner tous en raison des différents domaines et le nombre important, il y avait des inventions et un développement de nombreux types de robots et dans différent domaine, les plus récents sont des robots humanoïde basée sur l'intelligence artificielle telle que le robot kenshiro et kengoto, ce derniers est développer par l'université de Tokyo, capable de transpirer quand il fait du sport un moyen pour refroidir ces circuits. [1, 2]

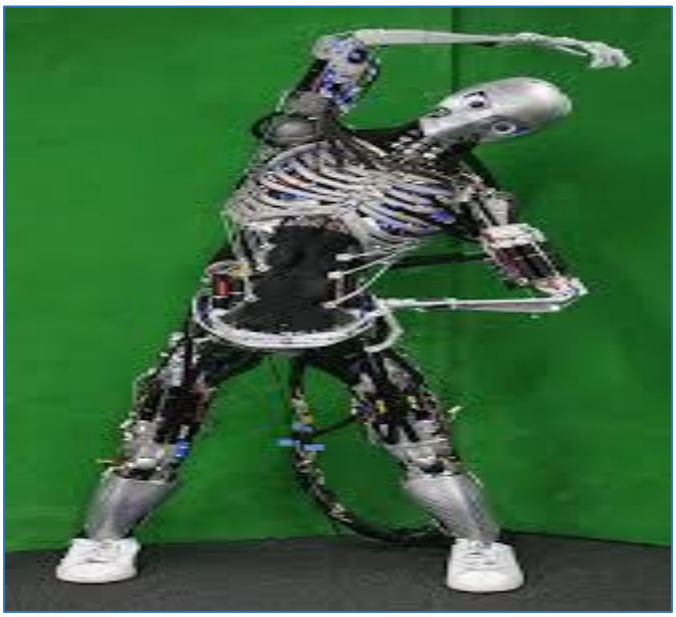


Figure I-9 Kengoro robot.

I.3 Définition générale :

I.3.1 Robots :

L'Association Française de Normalisation (A.F.N.O.R.) définit un robot comme étant un système mécanique de type manipulateur commandé en position, reprogrammable, polyvalent (i.e., à usages multiples), à plusieurs degrés de liberté, capable de manipuler des

matériaux, des pièces, des outils et des dispositifs spécialisés, au cours de mouvements variables et programmés pour l'exécution d'une variété de tâches. Il a souvent l'apparence d'un, ou plusieurs, bras se terminant par un poignet. Son unité de commande utilise, notamment, un dispositif de mémoire et éventuellement de perception et d'adaptation à l'environnement et aux circonstances. Ces machines polyvalentes sont généralement étudiées pour effectuer la même fonction de façon cyclique et peuvent être adaptées à d'autres fonctions sans modification permanente du matériel. [2]

Un robot est un dispositif mécatronique conçu pour accomplir automatiquement des tâches imitant ou reproduisant, dans un domaine précis, des actions humaines. La conception de ces systèmes est l'objet d'une discipline scientifique, branche de l'automatisme nommé robotique.

I.3.2 Robotique :

La robotique est une activité multidisciplinaire visant l'étude, la conception et la construction de robots ou plus simplement de machines automatiques. Sa pratique réunie des savoir-faire techniques et des connaissances scientifiques des domaines de l'électronique, de l'informatique et de la mécanique. [2]

I.3.3 Constituants de robot :

On distingue classiquement deux sous-ensembles : un système mécanique articulé et un ou plusieurs organes terminaux.

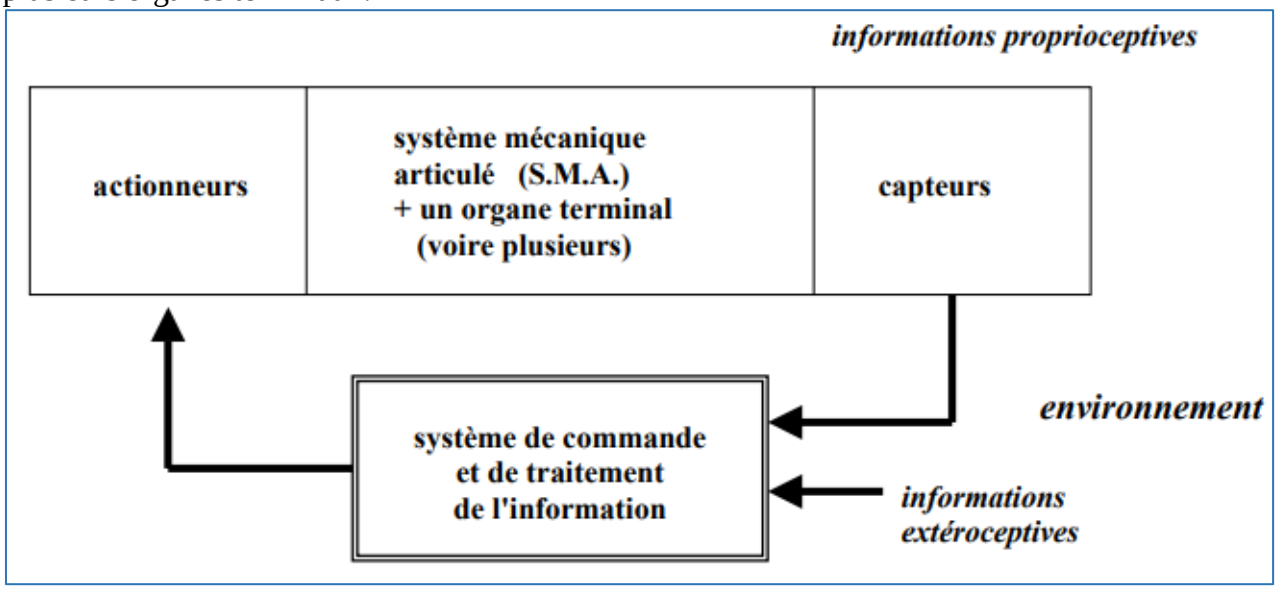


Figure I-10Structure fonctionnelle d'un robot.

- ***organe terminal***, on regroupe tout dispositif destiné à manipuler des objets (dispositifs de

serrage, dispositifs magnétiques, à dépression, …), ou à les transformer (outils, torche de soudage, pistolet de peinture, …). En d'autres termes, il s'agit d'une interface permettant au robot d'interagir avec son environnement. Un organe terminal peut être multifonctionnel, au sens où il peut être équipé de plusieurs dispositifs ayant des fonctionnalités différentes. Il peut aussi être monofonctionnel, mais interchangeable. Un robot, enfin, peut-être multi-bras, chacun des bras portant un organe terminal différent. On utilisera indifféremment le terme organe terminal, préhenseur, outil ou effecteur pour nommer le dispositif d'interaction fixé à

l'extrémité mobile de la structure mécanique.

➢ ***Le système mécanique articulé*** **(S.M.A.)** est un mécanisme ayant une structure plus ou moins proche de celle du bras humain. Il permet de remplacer, ou de prolonger, son action (le terme "manipulateur" exclut implicitement les robots mobiles autonomes) .Son rôle est d'amener l'organe terminal dans une situation (position et orientation) donnée, selon des caractéristiques de vitesse et d'accélération données. Son architecture est une chaîne cinématique de corps, généralement rigides (ou supposés comme tels), assemblés par des liaisons appelées articulations. Sa motorisation est réalisée par des actionneurs électriques, pneumatiques ou hydrauliques qui transmettent leurs mouvements aux articulations par des systèmes appropriés. [1]

I.3.4 Articulation :

Une articulation lie deux corps successifs en limitant le nombre de degré de liberté de l'un par rapport à l'autre. Soit m le nombre de degrés de liberté résultant, encore appelé mobilité de l'articulation. La mobilité d'une articulation est telle que :

$0 \leq m \leq 6$.

Lorsque m= 1, ce qui est fréquemment le cas en robotique, l'articulation est dite simple: soit rotoïde, soit prismatique.

I.3.4.1 Articulation rotoïde :

Il s'agit d'une articulation de type pivot, notée R, réduisant le mouvement entre deux corps à une rotation autour d'un axe qui leur est commun. La situation relative entre les deux corps est donnée par l'angle autour de cet axe (voir la figure suivante).

Figure I-11 Symbole de l'architecture rotoïde

I.3.4.2 Articulation prismatique :

Il s'agit d'une articulation de type glissière, notée P, réduisant le mouvement entre deux corps à une translation le long d'un axe commun. La situation relative entre les deux corps est mesurée par la distance le long de cet axe (voir la figure suivante).

Figure I-12 Symbole de l'architecture prismatique.

I.3.5 Degré de liberté (ddl) :

Le degré de liberté (ddl) d'un robot manipulateur est égal au nombre des paramètres

indépendants qui fixent la situation de l'organe terminal.

I.3.6 Espace de configuration :

L'emplacement d'un robot est représenté comme une configuration. La configuration d'un robot est un ensemble minimal de paramètres qui sont nécessaires pour décrire son emplacement exact. Cet ensemble de toutes les configurations possibles pour le robot est appelé espace de configuration, et c'est là que la planification de mouvement se produit. Il est d-dimensionnel. Dans chaque configuration q, le robot occupe un ensemble de points noté (q) dans un espace de travail W.Les n obstacles contenues dans W sont noté Oi ; $1 \leq i \leq n$; pour chaque Oi il y'a une contrepartie COi dans l'espace de configuration C. Cet obstacle de l'espace de configuration peut être défini comme :

$$COi = \{q \epsilon C \mid (q) \cap Oi \neq \emptyset\}, \qquad I\text{-}1$$

Ce qui signifie queCOi est un ensemble de toutes les configurations où le robot R serait en collision avec l'obstacle. Dans le cas d'un robot à corps rigide, l'espace de configuration libre peut être défini comme :

$$Cfree = C - COi\ n\ i=1 \qquad I\text{-}2$$

Et c'est un ensemble de configurations où le robot ne heurte aucun des obstacles. [6]

I.3.7 Espace articulaire :

On appelle espace de configuration articulaire d'un robot manipulateur l'état du robot représentant la situation de ses différents corps. Pour représenter celle-ci, la solution adoptée consiste à associer à chaque articulation une ou plusieurs variables (coordonnées articulaires). La dimension N de cet espace est égale au nombre de variables articulaires indépendantes et correspond au nombre de degré de liberté de la structure mécanique.

I.3.8 Espace opérationnel :

L'espace opérationnel est celui dans lequel est représentée la situation de l'organe terminal. La dimension de cet espace est définie par le nombre de degrés de liberté maximum que peut avoir l'organe terminal, est égale au nombre de paramètres indépendants nécessaires pour décrire la situation de l'organe terminal dans l'espace.

I.3.9 Espace de travail

L'espace de travail $W = \mathbb{R}N$ où$N = 2$ si le robot se déplace dans un plan, $N = 3$ s'il se déplace dans un espace tridimensionnel, est un environnement statique contenant des obstacles. Supposons que WOi est le ième obstacle tel que$1 \leq i \leq n$. L'espace de travail libre est l'ensemble de points exprimés par la fonction suivante :

$$Wfree = W \backslash UiWOi \qquad I\text{-}3$$

Où le \ est un opérateur de soustraction.
Le but est de trouver un chemin sans collision pour que le robot puisse passer d'une position et orientation initiale à une position et orientation de but. [6]

I.3.10 Notion de redondance

Un robot est redondant lorsque le nombre de degrés de liberté de l'organe terminal est inférieur au nombre d'articulations motorisées. Cette propriété permet d'augmenter le volume du domaine accessible et de préserver les capacités de déplacement de l'organe terminal en présence d'obstacles, ou les degrés de liberté supplémentaire autorisant leur contournement.

I.3.11 Le mécanisme

Ayant une structure plus ou moins proche de celle du bras humain, il permet de remplacer ou de prolonger son action. Sa motorisation est réalisée par des actionneurs électriques, pneumatiques ou hydrauliques qui transmettent leurs mouvements aux articulations par des systèmes appropriés.

I.3.12 La perception

Permet de gérer les relations entre le robot et son environnement. Les organes de perception sont des capteurs dits « proprioceptifs » lorsqu'ils mesurent l'état interne du robot (position et vitesses des articulations) ou « extéroceptifs »lorsqu'ils recueillent des informations sur l'environnement (détection de présence, mesure de distance, vision artificielle).

I.3.13 Commande

Synthétise les consignes des asservissements pilotant les actionneurs, à partir de la fonction de perception et des ordres de l'utilisateur.

I.3.14 L'interface homme-machine

C'est l'interface à travers laquelle l'utilisateur programme les tâches que le robot doit exécuter.

I.4 Anatomie d'un bras de robot

Lorsqu'on parle d'anatomie on pense à l'être humain. Mais, dans notre cas, on s'intéresse au bras de robot. Ce dernier est composée de : la base, les segments, les articulations, les actionneurs, l'organe terminal et l'effecteur.

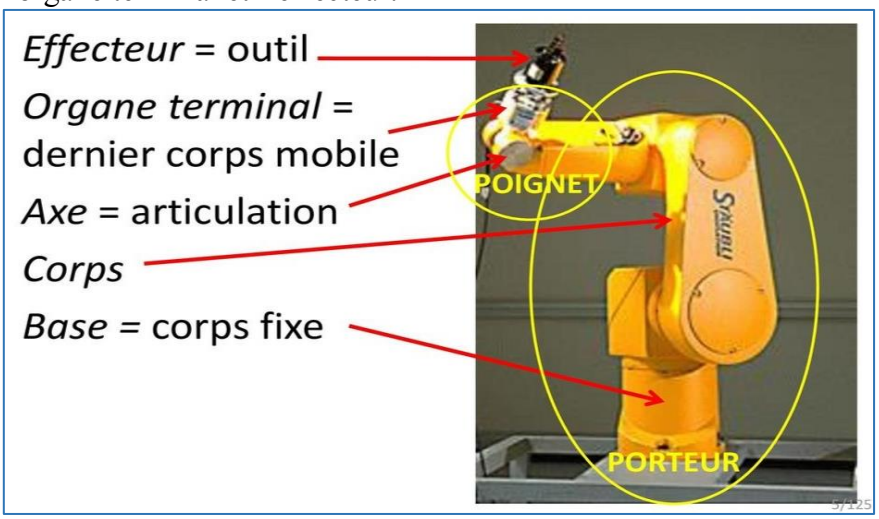


Figure I-13Anatomie d'un bras de robot.

I.4.1 Porteur et poignet :

On convient d'appeler les trois premiers degrés de liberté d'un robot (partant de la base du robot) « le Porteur du robot ». Les degrés de liberté résiduels forment le poignet, caractérisé par des dimensions beaucoup plus petites et une plus faible masse. Le porteur a pour rôle de fixer la position du point d'intersection, noté P, des axes des 3 dernières articulations (centre du poignet), cette position (P) ne dépend que de la configuration des solides (corps) 1, 2 et 3 (i.e., du porteur). Le poignet est destiné à l'orientation de l'organe terminal (pince, outil). [6]

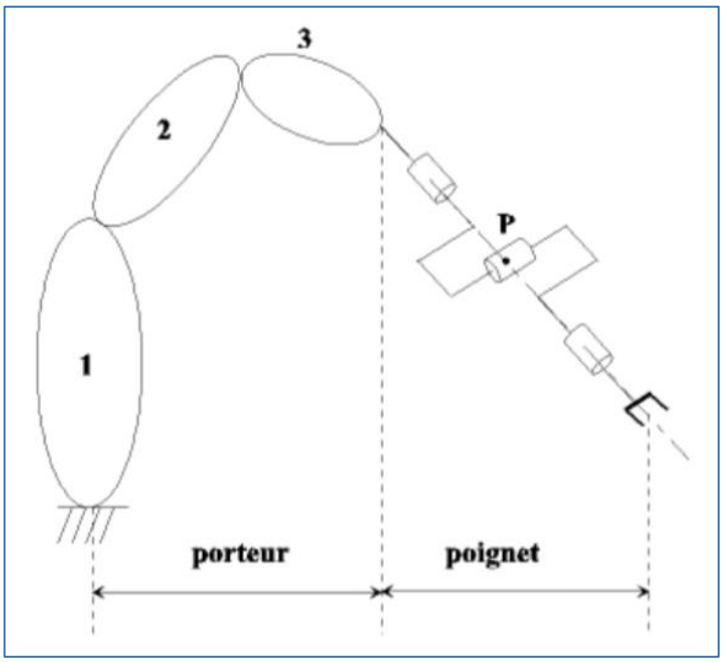


Figure I-14 Représentation du porteur et du poignet d'une structure articulée.

I.4.2 Différentes architectures des porteurs:

- Structure Cartésienne(PPP) : Très ancienne. Cette structure est constituée de trois liaisons prismatiques. Elle est inspirée de la conception traditionnelle d'une machine-outil à trois axes.

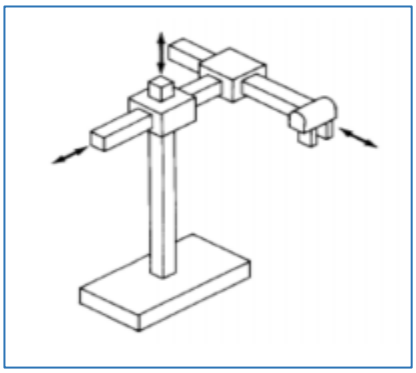


Figure I-15 structure Cartésienne.

- La structure cylindrique RPP (ou PRP) : Elle associe une rotation et deux translations. Elle présente l'inconvénient d'offrir un volume de travail faible devant un

encombrement total important.

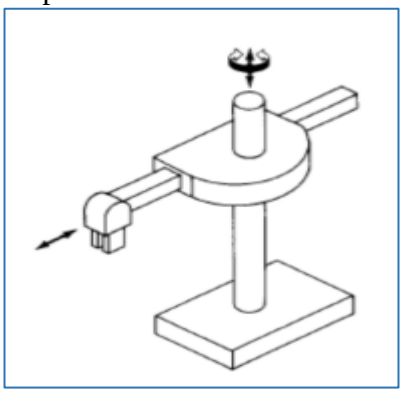


Figure I-16 structure cylindrique.

- La structure dite SCARA: La structure SCARA (Seletive Compilence Adaptive Robot Arm), possède des axes de rotation parallèles. Elle est l'une des plus utilisées, en particulier pour les tâches de manutention ou d'assemblages très fréquentes dans l'industrie.

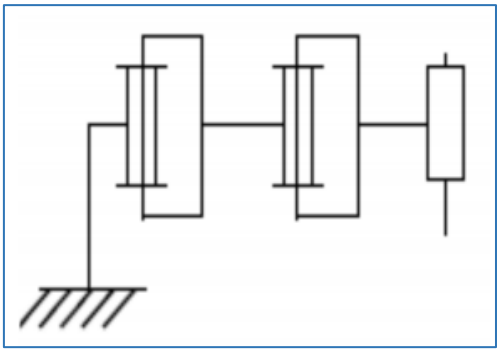


Figure I-17 Structure SCARA.

- Structure sphérique (RRP) : Il s'agit d'une structure qui est proche de la structure cylindrique et possède les mêmes inconvénients. Cette structure est abandonnée.

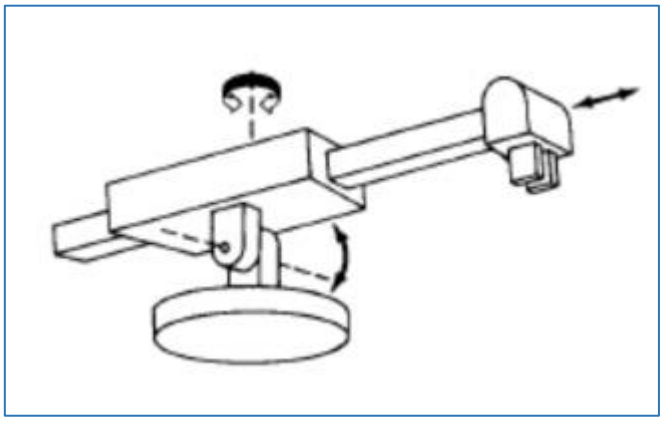


Figure I-18 Structure sphérique.

- Robot anthropomorphe (RRR) : Architecture plus généraliste. Elle reproduit le bras humain (3 rotations, une correspond à l'épaule, les deux autres aux coudes et le poignet). [7]

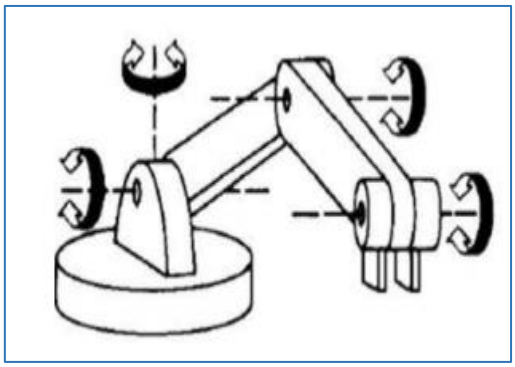


Figure I-19 Structure anthropomorphe.

I.4.3 Architecture des poignets:

L'organe terminal peut avoir 3 d.d.l pour assurer l'orientation souhaitée, selon les trois mouvements:

- Tangage (Pitch): (e.g, mouvement d'un avion autour d'un axe parallèle à l'envergure des ailes en passant par le centre de gravité)
- Lacet (Yaw) : (e.g, mouvement d'avion autour d'un axe vertical passant par son centre de gravité)
- Roulis (roll) : (e.g, mouvement d'avion autour de son axe longitudinal)

On peut aussi trouver les lettres pour désigner les axes : s (swivel, sliding), a (approach) et n (normal). [9]

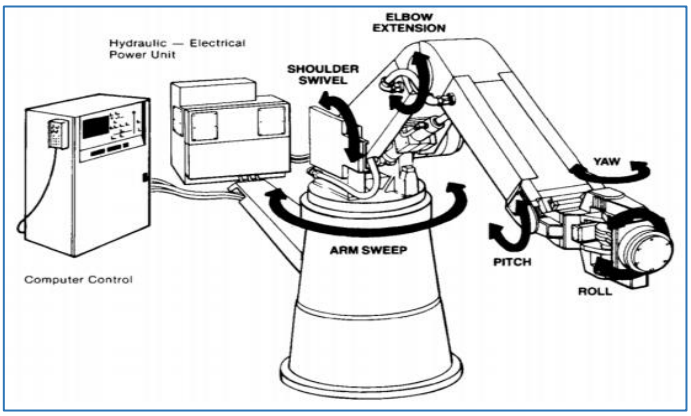


Figure I-20 Architecture des poignets.

I.4.4 Les actionneurs :

L'actionneur est un système dans le rôle principal est de convertir d'énergie, il est nécessaire de bien choisir l'actionneur adéquate car cela dépend de domaine d'application. La motorisation d'un actionneur comprend le moteur proprement dit, ainsi que le convertisseur qu'ils ont permis le contrôle on peut différencier les types de moteur en fonction de la source primaire. En général 3 sources d'énergie sont utilisées électrique, hydraulique, pneumatique :

- Robots électriques : Les actionneurs les plus fréquents utilisent des moteurs électriques .Les robots classiques ont une vitesse de régime élevée par rapport au niveau d'une articulation pivot, le moteur sera suivi d'un réducteur, ce qui permet d'avoir une amplification du couple moteur.
- Robots hydrauliques : Pour les robots devant manipuler de très lourdes charges, les actionneurs sont le plus souvent hydrauliques, agissant en translation (vérins hydrauliques) ou en rotation.

- Robots pneumatiques (manipulateurs à cycles): Les actionneurs pneumatiques sont d'un usage général pour les manipulateurs à cycles. Un manipulateur à cycle (automate ou bras transfert), est un SMA, permettant une succession de mouvements contrôlés uniquement par des capteurs de fin de course réglables manuellement à la course désirée. [7]

I.4.5 Les transmetteurs :

Ils sont des dispositifs mécaniques permettant de transmettre un mouvement d'une pièce à une autre (actionneur vers articulations).Ils assurent aussi l'adaptation du mouvement tel que la réduction de la vitesse, le changement de direction et la transformation du mouvement. Il existe beaucoup de modèles de transmettions, mais en robotique on s'intéresse aux transmissions à engrenages, à courroie et à roue de friction.

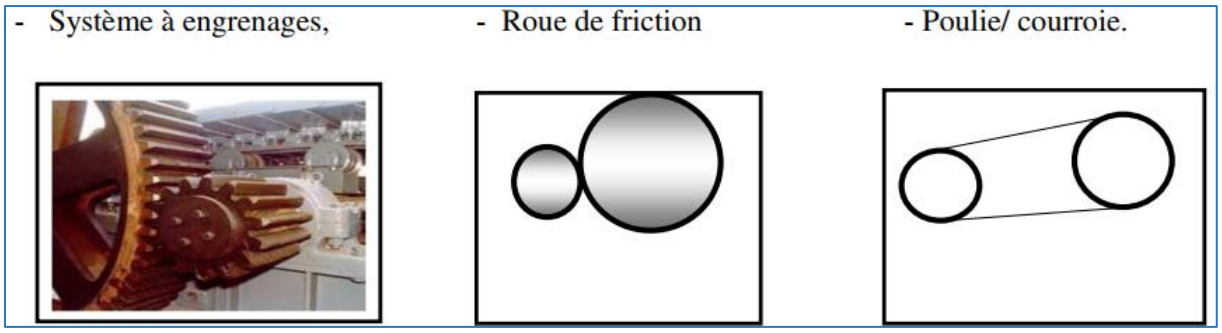


Figure I-21 Types de transmetteurs.

I.4.6 Les capteurs :

Plusieurs technologies de construction des capteurs existent. Un capteur est un dispositif transformant l'état d'une grandeur physique observée en une grandeur électrique. En robotique, on trouve deux types de capteurs :
Les capteurs proprioceptifs: Ils permettent d'identifier l'état du robot à un instant donné (e.g., positions, vitesses et l'accélérations des articulations).
Les capteurs extéroceptifs : Ils permettent d'identifier l'environnement du robot à un instant donné (caméras, télémètres,…).

I.5 Différentes générations de robots:

Les robots évoluent avec les progrès qui s'opèrent dans la mécanique, l'informatique, l'énergétique et les capteurs-actionneurs. Il existe trois classes de robots :

- Les robots passifs : Ils sont capables d'exécuter des tâches complexes, mais d'une manière répétitive et lourde. Ces robots n'adaptent avec les modifications dans l'environnement, mais leurs précision est faible.
- Les robots actifs : Ils sont capables de reconnaitre leur environnement et d'éviter des obstacles. Leur auto-adaptative est très élevée.
- Les robots intelligents Ils exploitent les avancées réalisées dans le domaine de l'intelligence artificielle pour développer la capacité d'établir une stratégie afin d'exécuter une tâche ou effectuer un déplacement.

I.6 Caractéristiques d'un robot :

On peut mentionner quelques-unes qui permettent de choisir un robot en fonction de l'application envisagée :

- L'espace de travail.
- Les vitesses et accélérations maximales, qui conditionnent les temps de cycle.
- Les performances (exactitude, répétitivité).
- L'architecture du système mécanique articulé : le choix est guidé par la tâche à réaliser
- La résolution : c'est la plus petite modification de la configuration du robot à la fois observable et contrôlable par le système
- La masse du robot.
- Le coût du robot.
- La maintenance.

I.7 Domaines d'application :

La robotique est un domaine en plein essor depuis quelques années. Les évolutions technologiques, dépassant sans cesse nos espérances, permettent maintenant de réaliser des solutions technologiques s'adaptant au moindre problème.
Par conséquent, la robotique est utilisée dans des domaines extrêmement rigoureux et exigeants. Nous allons explorer ces différents domaines.

I.7.1 L'industrie

Le but premier des robots est de remplacer l'homme dans des activités fastidieuses ou onéreuses pour l'employeur. Les robots ont donc commencé à être utilisés dans les chaînes d'assemblage industrielles. Dans ces chaînes d'assemblage, on retrouve des robots soudeurs, manipulateurs, peintres.

I.7.2 Le domaine militaire :

Les robots sont de plus en plus utilisés dans le domaine militaire. En effet, la miniaturisation permet aujourd'hui de créer des robots discrets mais dotés de nombreux capteurs, ce qui est idéal pour des missions d'espionnage ou d'éclairement.

I.7.3 La santé :

Les robots commencent à être de plus en plus dans le domaine médical, qu'il s'agisse de « simples » échographies ou d'opérations chirurgicales plus délicates. En fait ces robots ne sont pas complètement autonomes mais ils assistent les médecins ou chirurgiens, jusqu'à permettre des opérations médicales à distance (télémédecine). Cette pratique de « chirurgie assistée » est émergente donc bien que peu répandue, elle est en phase de devenir la chirurgie du futur.

I.7.4 L'usage domestique

La démocratisation de la robotique a conduit, ces dernières années, à voir de nombreux robots s'installer chez les particuliers pour effectuer des tâches à la place de leurs possesseurs. En effet, ceux-ci sont capables de faire le ménage, tondre la pelouse, nettoyer la piscine... Ce qui conduit certains clients (aisés) à se procurer ces domestiques contemporains.

I.8 Modélisation géométrique

I.8.1 Représentation d'un point dans l'espace:

Soit P un point de coordonnées cartésiennes (Px, Py, Pz) Figure I.22. Les coordonnées homogènes du point P sont les termes W*Px, W*Py et W*Pz ; où w est un facteur d'échelle souvent égal 1 en robotique.

On présente alors les coordonnées homogènes d'un point par le vecteur :

$$P=\begin{bmatrix} Px \\ Py \\ Pz \\ 1 \end{bmatrix}$$

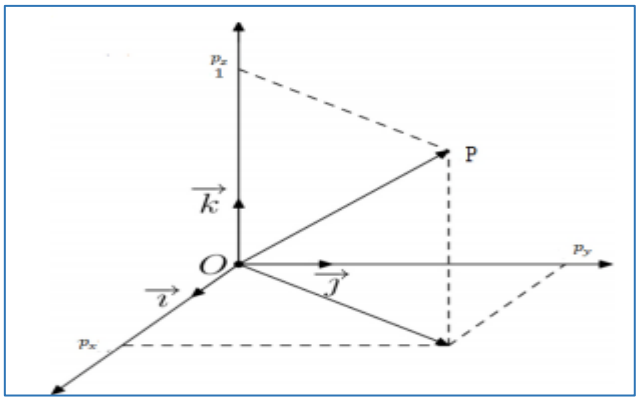


Figure I-22 Représentation d'un point dans l'espace.

I.8.2 Matrices de transformation homogènes

La représentation de la position et de l'orientation à la fois dans un repère peut se faire avec les matrices de transformations homogènes. Toutes les manipulations, telle que le passage d'un repère à un autre, peut se faire par une simple multiplication de ces matrices. Une matrice de transformation homogène a la forme :

$$T = \begin{bmatrix} R_{3x3} & P_{3x1} \\ N_{1x3} & 1 \end{bmatrix} \qquad I\text{-}4$$

Avec R3X3 est la matrice de rotation représentant l'orientation, P est le vecteur colonne de position et N = [0 0 0] est un vecteur ligne généralement nul, mais qui peut être utilisé pour la perspective en vision artificielle.

I.8.2.1 Matrices de rotation :

Soient deux repère A et B dont les origines coïncident (O_0 ; O_1). L'orientation du repère B par rapport au repère A peut être représentée par le produit de trois matrices élémentaires représentant les rotations autour des axes X, Y et Z :

- Rotation d'un angle $\boldsymbol{\theta}_1$ autour de X :

$$Rot(X,\ \theta 1)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & COS\theta 1 & -sin\theta 1 & 0\\ 0 & sin\theta 1 & COS\theta 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \qquad \text{I-5}$$

- Rotation d'un angle $\theta 2$ autour de Y :

$$Rot\ (Y,\ \theta_2\ )\ =\begin{bmatrix}COS\theta 2 & 0 & sin\theta 2 & 0\\ 0 & 1 & 0 & 0\\ -sin\theta 2 & 0 & COS\theta 2 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \qquad I\text{-}6$$

- Rotation d'un angle $\theta 3$ autour de Z:

$$Rot\ (Z,\ \theta_3)\ =\begin{bmatrix}COS\theta 3 & Sin\theta 3 & 0 & 0\\ Sin\theta 3 & COS\theta 3 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \qquad I\text{-}7$$

I.8.2.2 Matrice de translation :

Soient deux repère A et B et (Px, Py, Pz) les coordonnées de l'origine du repère B exprimées dans le repère A. Lorsque les axes des deux repères sont en parallèle (même orientation), alors la matrice A est appelée ‘’matrice de translation ‘’elle est donnée par :

$$T_{trans}=\begin{bmatrix}1 & 0 & 0 & Px\\ 0 & 1 & 0 & Py\\ 0 & 0 & 1 & Pz\\ 0 & 0 & 0 & 1\end{bmatrix} \qquad I\text{-}8$$

➢ Nécessité d'un modèle :

La conception et la commande des robots nécessitent le calcul de certains modèles mathématiques, tels que :
–les modèles de transformation entre l'espace opérationnel (dans lequel est définie la situation de l'organe terminal) et l'espace articulaire (dans lequel est définie la configuration du robot). On distingue :
* les modèles géométriques direct et inverse qui expriment la situation de l'organe terminal en fonction des variables articulaires du mécanisme et inversement ;
*les modèles cinématiques direct et inverse qui expriment la vitesse de l'organe terminal en fonction des vitesses articulaires et inversement ;
– les modèles dynamiques définissant les équations du mouvement du robot, qui permettent d'établir les relations entre les couples ou forces exercés par les actionneurs et les positions, vitesses et accélérations des articulations.

I.9 Modèle géométrique directe MGD :

La modélisation des robots exige une méthode adéquate pour la description de leur structure. Plusieurs méthodes ont été proposées, telle que la convention de Denavit-Hartenberg(DH) et DH modifiée, et celle de Khalil-Kleinfinger (KK).

I.9.1 Convention de Denavit-Hartenberg :

Une structure ouverte simple est composée de n+1 corps notés C0, …, Cn et de n articulations. Le corps C0 désigne la base du robot et le corps Cn le corps qui porte l'organe terminal. L'articulation j connecte le corps Cj au corps Cj-1. La méthode de description est fondée sur les règles et conventions suivantes :

- La variable de l'articulation j est notée qj
- Le corps j est noté Cj
- Les corps sont supposés parfaitement rigides, ils sont connectés par des articulations soient rotoïdes soient prismatiques.
- Le repère Rj est lié au corps Cj
- L'axe Zj du repère Rj est porté par l'axe de l'articulation j pour les robots à chaines ouvertes simple.
- Le repère Rj fixé au corps Cj est définit comme suit :
  - L'axe Zj est porté par l'axe de l'articulation j
  -L'axe Xj est porté par la perpendiculaire commune aux axes Zj et $Zj-1$

Le passage du repère Rj-1 au repère Rj s'exprime en fonction des quatre paramètres géométriques suivants (figure I-23) :
• αj : angle entre les axes zj-1 et zj correspondant à une rotation autour de xj-1 ;
• dj : distance entre zj-1 et zj le long de xj-1 ;
• θj : angle entre les axes xj-1 et xj correspondant à une rotation autour de zj ;
• rj : distance entre xj-1 et xj le long de zj

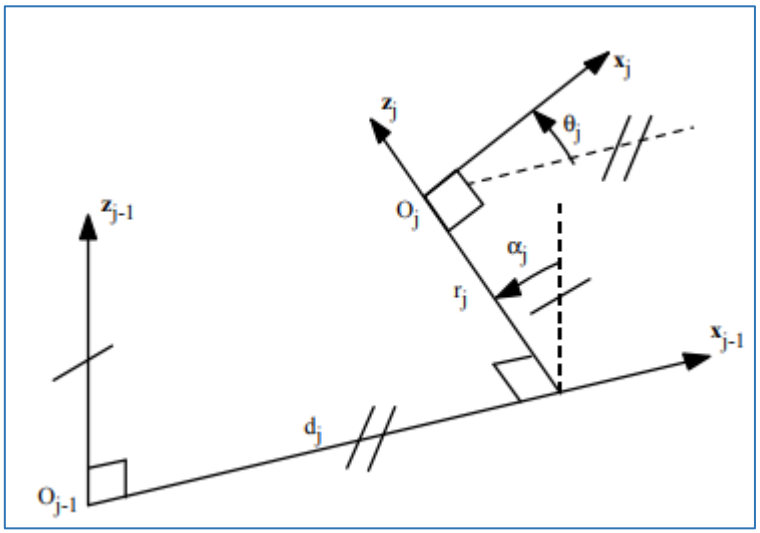


Figure I-23 Paramètres géométriques dans le cas d'une structure ouverte simple.

La matrice de transformation définissant le repère Rj dans le repère Rj-1 est donnée par (figure I-23) :

^{j-1}Tj = Rot(x, αj) Trans(x, dj) Rot(z, θj) Trans(z, rj) $^{j-1}$

$$^{j-1}T_j = \begin{bmatrix} C\theta j & -S\theta j & 0 & d \\ C\alpha jS\theta j & C\alpha jC\theta j & -S\alpha j & -rjS\alpha j \\ S\alpha jS\theta j & S\alpha jC\theta j & C\alpha j & rjC\alpha j \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{I-9}$$

Dans la convention DH modifiée (MDH), le repère Ri est placé au début du corps Ci. Il est associé à l'articulation i. La matrice de passage d'une articulation à la suivante est

^{j}Tj-1= Rot(x, αj-1) Trans(x, dj-1) Rot(z, θj) Trans(z, rj)

$$ {}^{j}_{Tj-1} = \begin{bmatrix} C\theta j & -S\theta j & 0 & d_{j-1} \\ -C\alpha_{j-1}s\theta j & C\alpha_{j-1}C\theta j & -S\alpha_{j-1} & -rjS\alpha_{j-1} \\ S\alpha_{j-1}S\theta j & S\alpha_{j-1}C\theta j & C\alpha_{j-1} & rjC\alpha_{j-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{I-10} $$

Le MGD d'un bras est obtenu par le produit des matrices : $T_0^n = T_0^1 . T_1^2 \ldots . T_{n-1}^n$

I.10 Modèle géométrique inverse (MGI):

Le modèle géométrique inverse (MGI) permet de calculer les coordonnées articulaires qui amènent l'organe terminal dans une situation désirée, spécifiée par ses coordonnées opérationnelles. Cette opération est souvent appelée transformation de coordonnées.
Le MGI s'écrit :

$$ \theta = f^{-1}(X) \qquad \text{I-11} $$

La détermination du modèle géométrique inverse (MGI) est un problème complexe. Il y a rarement unicité de solution. On doit inverser un système d'équations non linéaires. Ce qui n'est pas trivial.

I.11 Modélisation cinématique direct et inverse :

I.11.1 Modèle cinématique direct

Le modèle cinématique direct d'un robot manipulateur décrit les vitesses des coordonnées opérationnelles en fonction des vitesses articulaires. Il est noté :

$$ \dot{x} = J(q)\,\dot{q} \qquad I\text{-}12 $$

Où J(q) désigne la matrice jacobéenne de dimension (mxn) du mécanisme, égale à ∂X ∂q et la fonction de la configuration articulaire q. La même matrice jacobéenne intervient dans le calcul du modèle différentiel direct qui donne les variations élémentaires dX des coordonnées opérationnelles en fonction des variations élémentaires des coordonnées articulaires dq, soit :

$$ dX = J(q)\,dq \qquad I\text{-}13 $$

I.11.2 Modèle cinématique inverse :

L'objectif du modèle cinématique inverse (MCI) est de calculer, à partir d'une configuration q donnée, les vitesses articulaires $\dot{q}$ qui assurent au repère terminal une vitesse opérationnelle $\dot{x}$ imposée. On peut déterminer aussi la différentielle articulaire correspondant à une différentielle des coordonnées opérationnelles dq. Dans le cas régulier d'une matrice Jacobéenne carrée et non singulière, le modèle cinématique inverse s'écrit :

$$ \dot{q} = J^{-1}\,\dot{x} \qquad I\text{-}14 $$

Le modèle dynamique est la relation entre les couples (et/ou forces) appliqués aux actionneurs et les positions, vitesses et accélérations articulaires. On représente le modèle dynamique par une relation de la forme :

$$\Gamma = f(q, \dot{q}, \ddot{q}, fe) \qquad I\text{-}15$$

Avec :

- Γ : vecteur des couples/forces des actionneurs, selon que l'articulation est rotoïde ou prismatique.
- q : vecteur des positions articulaires ;
- $\dot{q}$: Vecteur des vitesses articulaires ;
- $\ddot{q}$: Vecteur des accélérations articulaires ;
- fe : vecteur représentant l'effort extérieur (forces et moments) qu'exerce le robot sur l'environnement.

On peut appeler la relation (I.11-4) le modèle dynamique inverse, ou tout simplement le modèle dynamique. Le modèle dynamique directe exprime les accélérations articulaires q en termes de positions q, vitesses $\dot{q}$ et des couples τ. Par conséquent, il est représenté par la relation :

$$q = (q, \ , \dot{q}, \tau, fe) \qquad I\text{-}16$$

Deux grandes approches existent pour calculer les équations dynamiques d'un robot. Celle de Newton-Euler, [6, 9], qui décrit le comportement dynamique d'un système en termes de forces et moments, et celle de Lagrange qui le décrit en termes de travail et d'énergie. Pour plus de détail consulter les références suivantes. [3]

Conclusion :

Dans ce chapitre, nous avons présenté des généralités sur les bras de robot, leur utilisation, leurs constituants et quelques définitions de certains termes. On a vu que l'utilisation des robots n'est pas limité seulement à l'industrie mais elle est étendue à d'autres domaines.

Par la suite, nous avons présenté la modélisation des bras de robots. Les trois modèles : géométrique, cinématique et dynamique (avec modèle direct et inverse pour tous les modèles).

II. Chapitre.2 Planification de trajectoires et les outils de la PRM

II.1 Introduction

Feuille de route probabiliste (prms), a montré un grand potentiel pour résoudre des problèmes complexes de grande dimension. Prm utilise la randomisation (généralement pendant le prétraitement) pour construire un graphe de chemins représentatifs dans l'espace C (une feuille de route) dont les sommets correspondent à des configurations sans collision du robot et dans lesquels deux sommets sont connectés par une arête si un chemin entre les deux, les configurations correspondantes peuvent être trouvées par une méthode de planification locale.

II.2 Planification de du mouvement

Certains concepts de base de la planification de mouvement sont brièvement décrits et le problème est formulé. Un concept utile est un espace de configuration qui est généralement utilisé de nos jours dans les planificateurs de mouvement [6]. Au fil des ans, de nombreux algorithmes ont été proposés pour résoudre le problème de planification de mouvement et certaines de ces méthodes de solution sont également passées en revue dans ce chapitre.

II.2.1 Formulation du problème

En planification de mouvement, le robot se déplace dans un espace de travail W qui est dans des cas pratiques soit R2 ou R3. Habituellement, l'emplacement d'un robot est représenté comme une configuration. Comme pour, un robot à corps rigide qui se déplace dans un espace de travail en trois dimensions et peut à la fois translater et tourner, un emplacement exact peut être exprimé sous la forme d'une configuration à six paramètres. Trois sont nécessaires pour le poste et trois pour l'orientation. Un ensemble de toutes les configurations possibles pour un robot est appelé un espace de configuration (défini dans chapitre1).Il est utile de décrire le robot comme une configuration car il nous permet de traiter Le robot comme un point dans un espace en d dimensions, ce qui est beaucoup pratique.
Un ensemble de configurations où le robot n'entre en collision avec aucun des obstacles. Cependant, cela est insuffisant dans le cas de robots composés de plusieurs corps. Dans ce cas, la collision avec des obstacles doit être vérifiée séparément pour chaque corps et en plus, les collisions entre différents corps doivent également être prises en considération.
La tâche dans la planification de mouvement peut être définie comme la recherche d'un chemin libre entre la configuration qstart et la configuration qgoal. Le chemin est une fonction continue

$$\tau: [0, 1] \rightarrow Cfree \qquad \text{II-1}$$

Où τ (0) = qstart et τ (1) = qgoal. Parce que le chemin se trouve totalement dans un espace de configuration libre, il est garanti que le robot ne heurte aucun obstacle sur l'espace de travail tout en se déplaçant le long de celui-ci. Il est très difficile de résoudre exactement ce problème de planification de mouvement, surtout si le l'espace de configuration est complexe et de grande dimension. Par conséquent, le problème est généralement résolu avec des méthodes approximatives dans la pratique. [6]

II.2.2 Résolution du problème de planification de mouvement

Différentes approches ont étaient utilisées pour résoudre le problème de planification de mouvement. On essaye de décrire quelques-unes de ces approches :

1. Recherche basé sur la grille : Il s'agit de diviser l'espace de configuration en une grille. Cellules de la grille elles ont généralement une forme uniforme, et dans le cas bidimensionnel elles peuvent être des carrés ou des hexagones, par exemple. Chaque cellule peut être libre ou bloquée. Cellules libres ceux qui n'ont pas d'obstacles et les cellules bloquées sont ceux avec un obstacle le robot prend une cellule à la fois et peut aller dans les cellules adjacentes. C'est possible de penser que les cellules libres forment un graphique, et donc le chemin entre deux cellules peut trouver en utilisant un algorithme de recherche graphique.

Un problème avec les méthodes basées sur la grille est qu'il est possible de se déplacer d'une cellule uniquement aux cellules adjacentes, ce qui rend le chemin final longue et peu naturelle.

2. Méthodes de décomposition des cellules: Les méthodes de décomposition cellulaire ressemblent à une recherche de grille de base dans certaines façons. Au lieu de cellules de forme uniforme, Cfree est divisé en cellules qui peuvent toutes être de forme et de taille différentes. Tout comme dans la recherche de grille de base, deux cellules sont adjacentes les uns aux autres s'ils ont une frontière commune. Parce que les cellules ne sont pas uniformément en forme, les informations sur la contiguïté des cellules sont explicitement stockées dans un graphique. Le robot peut se déplacer librement à l'intérieur d'une cellule et en fonction des informations le graphique d'adjacence, le robot peut également se déplacer vers d'autres cellules. Il est important est très facile de trouver un chemin pour un robot entre deux configurations à l'intérieur d'une cellule. Par conséquent, les cellules doivent de préférence être convexes.

Le problème avec les méthodes de décomposition des cellules, qu'elles soient exactes ou approximatives, est leur besoin de nombre de cellules dans des environnements complexes. Ces méthodes deviennent généralement rapidement peu pratiques lorsque la dimension de l'espace de configuration augmente. [7].

3. Planificateur de chemins aléatoires: Cette technique attribue à chaque point du C-espace une valeur de potentiel, somme des forces d'attraction propagées depuis la configuration cible, et des forces de répulsion résultantes des obstacles. Bien que simple, et dans certains cas efficace, son inconvénient majeur est qu'il existe un ensemble de minima locaux dans cette description (figure II-1). Il faut alors mettre en œuvre des mécanismes de sortie des minima (un tirage aléatoire par exemple), mais aussi avant tout un mécanisme de détection de l'entrée dans une telle zone. Cet inconvénient existera aussi dans un espace de configurations en N dimensions.

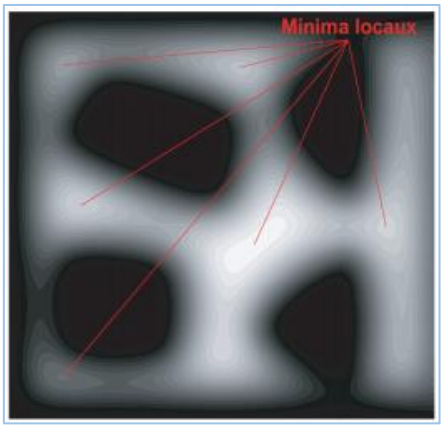


Figure II-1Carte de champs de potentiel. Les zones foncées représentent les obstacles (potentiel de répulsion), et les zones claires décrivent

4. Les RRT : Arbres probabilistes pour l'exploration rapide: Cette technique est souvent utilisée en planification de mouvement dans des espaces de grande dimension. Elle consiste à construire un arbre de recherche à la demande. La configuration initiale est prise comme racine de l'arbre. À chaque itération une configuration q est tirée de manière aléatoire. Le nœud de l'arbre (configuration) le plus proche de q est noté qnear. Puis une configuration qnew est sélectionnée, qui se situe entre q et qnear, à une distance maximale ϵ de qnear. Cette configuration sera ajoutée comme nœud de l'arbre s'il existe un lien possible (sans collision) entre qnear et qnew (figure II-2). Cette technique essaye d'atteindre, à chaque itération la configuration cible. Il est aussi possible d'augmenter cette méthode en construisant deux arbres, le premier ayant pour racine la configuration initiale, et le second ayant pour racine la configuration cible. Ce modèle essaye ensuite à chaque itération de relier les nœuds qui viennent d'être ajoutés.

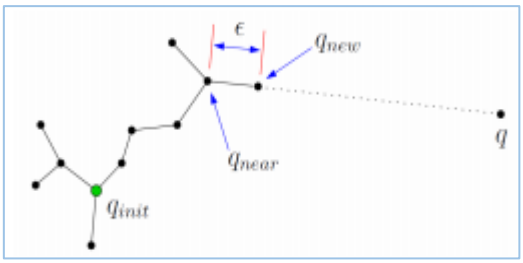


Figure II-2 Méthode de construction d'un RRT

Les méthodes mentionnées dans cette section créent une sorte de feuille de route. Tel que, le graphique formé par les méthodes de décomposition cellulaire afin de coder la contiguïté des cellules peut être considérée comme une feuille de route ainsi que des arbres construits avec les méthodes RRT. Une feuille de route est une approximation de Cfree, au lieu d'essayer d'obtenir une représentation exacte, il peut être très utile dans des problèmes de dimension supérieure. Les planificateurs probabilistes de la feuille de route essaient de faire exactement cela et ils sont étudiés plus en détail dans le chapitre suivant.

II.3 Planificateur local

Une autre décision de conception importante est liée à la vitesse à laquelle le planificateur local doit être. Il y a clairement un compromis entre le temps passé dans chaque appel individuel de ce planificateur et le nombre d'appels. Si un puissant planificateur local est utilisé, il réussit souvent à trouver un chemin quand il en existe un. Par conséquent, relativement peu de nœuds pourraient être nécessaires pour construire une feuille de route capturant suffisamment bien la connectivité de Qfree pour répondre de manière fiable aux requêtes de planification de chemin. Un tel planificateur local serait probablement assez lent, mais cela pourrait être quelque peu compensé par le petit nombre d'appels nécessaires. D'un autre côté, un planificateur très rapide aura probablement moins de succès. Il faudra plus de configurations pour être inclus dans la feuille de route et par conséquent, le planificateur local est appelé plus de fois pour les connexions entre les nœuds.

Cependant, chaque appel sera moins cher. Le choix du planificateur local affecte également la phase de requête. Il est important de pouvoir connecter n'importe quelle configuration qinit et qgoal donnée à la feuille de route ou de détecter très rapidement qu'aucune connexion de ce type n'est possible. Cela nécessite que la feuille de route soit suffisamment dense pour qu'elle contienne toujours au moins quelques nœuds auxquels il est facile de connecter qinit et qgoal. Il semble donc préférable d'utiliser un planificateur local très rapide, même s'il n'est pas trop puissant, et de construire de grandes feuilles de route avec des configurations largement réparties sur Qfree. De plus, si le planificateur local est très rapide, le même planificateur peut être utilisé pour connecter qinit et qgoal à la feuille de route au moment de la requête. Des discussions sur l'utilisation de différents planificateurs locaux peuvent être trouvées dans [13].

Un planificateur populaire, applicable à tous les robots holonomiques, connecte deux configurations données par un segment rectiligne dans Q et vérifie la collision de ce segment de ligne. Il faut veiller à interpoler séparément les composants de translation et de rotation (voir [21]). Il existe deux choix couramment utilisés pour la vérification de collision, les algorithmes de vérification de collision incrémentielle et de subdivision. Dans les deux cas, le segment de ligne, ou plus généralement, tout chemin généré par le planificateur local entre les configurations q' et q'', est discrétisé en un certain nombre de configurations $(q1, ..., ql)$, où $q' = q1$ et $q'' = ql$. La distance entre deux configurations consécutives quelconques qi et $qi+1$ est inférieure à une taille de pas constante positive. Cette valeur est spécifique au problème et est définie par l'utilisateur. Il est important de noter que l'échantillonnage est à nouveau utilisé pour déterminer si un chemin local est sans collision. Mais dans ce cas, l'échantillonnage est effectué à un niveau beaucoup plus fin que pour la génération de nœuds et c'est une caractéristique très importante de PRM. En général, la valeur de la taille de pas doit être très petite pour garantir que toutes les collisions soient trouvées.

Dans le cas de la vérification de collision incrémentielle, le robot est positionné à q' et déplacé à chaque taille pas à pas le long de la ligne droite en Q entre q' et $q.''$
Un contrôle de collision est effectué à la fin de chaque étape. L'algorithme se termine dès qu'une collision est détectée ou lorsque q'' est atteint.

Dans le cas de la vérification de collision de subdivision, le point médian qm de la droite en Q entre q' et q'' est d'abord vérifié pour la collision. Ensuite, l'algorithme revient sur les droites entre (q', qm) et (qm, q''). La récursivité s'arrête lorsqu'une collision est détectée ou que la longueur du segment de ligne est inférieure à la taille du pas.
Dans les deux algorithmes, le chemin est considéré comme sans collision si aucune des configurations intermédiaires ne produit de collision. Aucun des deux algorithmes n'a un avantage théorique clair sur l'autre, mais en pratique, l'algorithme de vérification des collisions par subdivision a tendance à mieux fonctionner [13].

La raison en est que, en général, les trajets plus courts ont tendance à être sans collision. La vérification de collision de subdivision réduit la longueur du chemin local dès que possible. Il est également possible d'utiliser un algorithme de contrôle de collision par subdivision adaptative qui ajuste dynamiquement la taille des pas. Dans [15], la taille des pas est déterminée en reliant la distance entre le robot et les obstacles de l'espace de travail à la longueur maximale du chemin tracé par n'importe quel point du robot. En outre, la méthode de [15] est exacte, c'est-à-dire qu'elle trouve toujours une collision lorsqu'une collision existe, tandis que les techniques de discrétisation ci-dessus peuvent manquer une collision si la taille du pas est trop grande. La figure II-3 illustre comment les algorithmes de vérification de collision incrémentielle et de subdivision échantillonnent la ligne droite entre deux configurations q ′ et q ″ . Dans cet exemple, l'algorithme de subdivision effectue un plus petit nombre de vérifications de collision.
Si l'obstacle avait été proche de q ′, l'algorithme incrémental aurait effectué un plus petit nombre de vérifications de collision.

Requêtes de post-traitement :

Une étape de post-traitement peut être appliquée au chemin reliant qinit à qgoal pour améliorer sa qualité selon certains critères. Par exemple, la brièveté et la douceur peuvent être souhaitables.

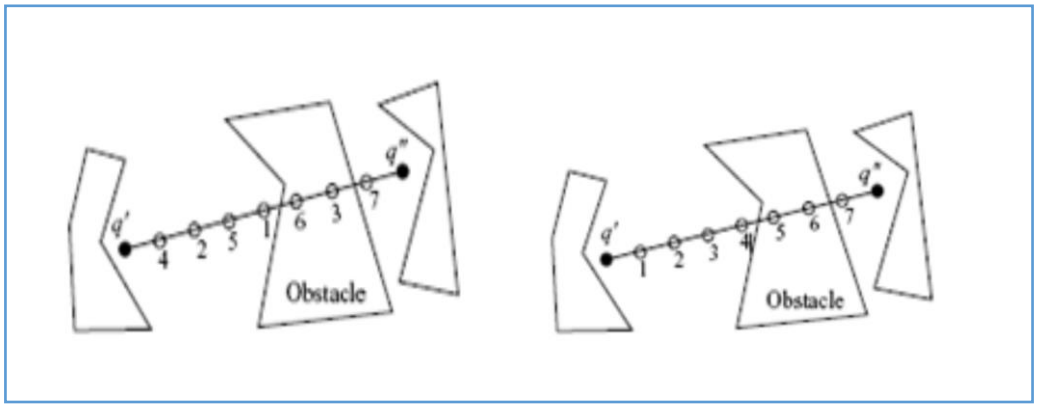


Figure II-3Echantillonnage le long de la ligne droite entre deux configurations q' et q ''.Les nombres correspondent à l'ordre dans lequel chaque stratégie vérifie la collision des échantillons. a) incrémentiel : l'algorithme renvoie l'échec après cinq vérifications de collusion .b) Subdivision : l'algorithme renvoie l'échec après trois vérifications de collision.

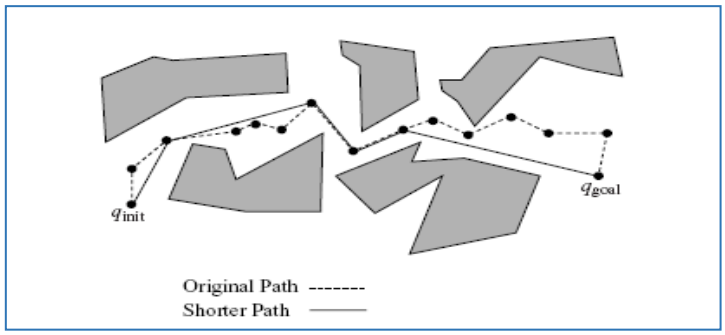


Figure II-4 Traitement du chemin renvoyé par PRM pour obtenir un chemin plus court avec l'approche gourmande.

Le post-traitement est applicable à tout algorithme de planification de chemin, mais est présenté ici pour l'exhaustivité des directives d'implémentation du PRM de base. A partir d'un chemin donné, un chemin plus court pourrait être obtenu en vérifiant si les configurations non adjacentes q1 et q2 le long du chemin peuvent être connectées avec le planificateur local.
Cette idée a été souvent décrite dans la littérature. Les points q1 et q2 pourraient être choisis au hasard. Une autre alternative serait une approche gourmande.
On Commence par qinit et on essaye de nous connecter directement au qgoal cible. Si cette étape échoue, on démarre à partir de la configuration après qinit et on réessaye. On Répète jusqu'à ce qu'une connexion puisse être établie avec qgoal, disons à partir du point q0. Maintenant, on définisse la cible sur q0 et on recommence, en essayant de nous connecter de qinit à q0, et on répète la procédure. Cette procédure peut également être appliquée dans la direction opposée.

Il existe diverses raisons pour lesquelles les configurations q1 et q2 le long d'un chemin peuvent ne pas avoir été connectées à un bord de l'étape de construction de la feuille de route de PRM. Ils peuvent ne pas être proches en fonction de la fonction de distance dist, et la requête k (k est nombre de voisins les plus proches à examiner pour chaque configuration) du voisin le plus proche peut ne pas les renvoyer en tant que voisins. Ils peuvent, cependant, être dans une partie relativement épurée de Qfree et un long bord les reliant peut toujours être possible. Ces cas se produiront plus fréquemment si la stratégie de connexion Création de feuilles de route clairsemées a été utilisée. Au lieu de raccourcir le chemin, un objectif différent peut être d'obtenir un chemin avec une courbure lisse. Une approche possible consiste à utiliser des courbes d'interpolation, telles que des splines, et à utiliser les configurations qui ont été calculées par PRM comme points d'interpolation pour les courbes. Dans ce cas, la vérification de collision est effectuée
Le long des courbes jusqu'à ce que des courbes qui satisfassent à la fois les propriétés de lissage et les critères d'évitement de collision soient trouvées.
Les étapes de post-traitement telles que le raccourcissement de chemin et le lissage de chemin peuvent améliorer la qualité du chemin, mais peuvent également imposer une surcharge importante sur le temps nécessaire pour signaler les résultats d'une requête. En général, si des chemins avec certains critères d'optimalité sont souhaités, il vaut la peine d'essayer de construire ces chemins pendant la phase de construction de la feuille de route de PRM.
Par exemple, une grande feuille de route dense donnera probablement des chemins plus courts qu'une feuille de route plus petite et plus clairsemée.

Un exemple : La figure II-5 (a) montre un problème de planification de mouvement pour un robot dans un espace de travail tridimensionnel. Le robot est un objet polyédrique rigide non convexe; il peut librement se déplacer et pivoter dans l'espace de travail tant qu'il n'entre pas en collision avec les obstacles.
L'espace de travail est composé d'une paroi mince rigide qui a un passage étroit. Un cadre de délimitation est défini qui contient le mur et est suffisamment petit pour ne pas permettre au robot de se déplacer d'un côté du mur à l'autre sans passer par le passage étroit.
L'objectif est de construire une feuille de route qu'un planificateur peut utiliser pour résoudre successivement des requêtes de planification de mouvement où qinit et qgoal apparaissent sur les deux côtés différents du mur. Le problème a six degrés de liberté, trois en translation et trois en rotation. La configuration q = (p, r) du robot peut être représentée par un point p exprimant la composante translationnelle et un quaternion r exprimant la composante rotationnelle.
Une configuration est générée en prélevant au hasard un échantillon à partir d'une distribution uniforme à partir d'un sous-ensemble de positions autorisées dans R3 et en choisissant un axe de rotation aléatoire et un angle aléatoire pour le quaternion (pour plus de détails, voir [14]).

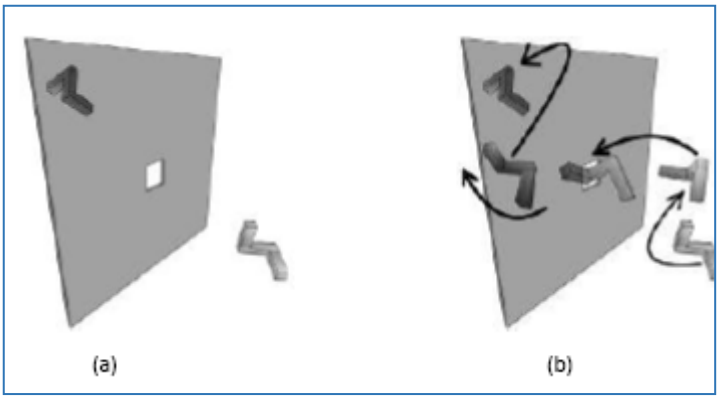


Figure II-5 Exemple d'un problème de planification de mouvement où le robot et les obstacles sont une collection d'objets polyédriques en trois dimensions. Des parties du robot de l'autre côté du mur sont indiquées par la couleur plus foncée. (a) La configuration initiale et finale de la requête. (b) Un chemin produit à partir d'un PRM avec n = 1000 et k = 10.

Afin de trouver les k voisins les plus proches d'une configuration, les configurations sont intégrées dans un espace où la distance euclidienne est définie. Une méthode qui fonctionne bien dans la pratique consiste à choisir une paire de points sur la surface du robot qui ont une distance maximale et à construire un vecteur à six dimensions emb (q) pour la configuration initiale du robot. Si q ′ est obtenu en appliquant une transformation de translation et de rotation à q, alors emb (q ′) est obtenu en appliquant les mêmes transformations à la paire de points dans emb (q). La distance métrique dist est alors définie comme la distance euclidienne des deux plongements. Pour chaque configuration et ses k homologues les plus proches, l'algorithme de vérification de collision de subdivision est utilisé pour vérifier si la ligne droite dans Q est sans collision. Les configurations intermédiaires entre q ′ = (p ′, r ′) et q ″ = (p ″ , r ″) sont obtenues en effectuant des interpolations linéaires sur p ′ et p ″ et des interpolations sphériques sur r ′ et r ″ . Le bord (q ′, q ″) est ajouté à la feuille de route lorsque toutes les configurations intermédiaires sont sans collision.
Une fois la feuille de route terminée, elle peut être utilisée pour résoudre les requêtes spécifiées par l'utilisateur. Les k voisins les plus proches des points de requête sont calculés et le planificateur local tente de leur connecter qinit et qgoal. Dès qu'ils sont connectés au même composant, un algorithme A * est exécuté sur le graphe pour trouver le chemin. La figure II-5 (b) montre les configurations intermédiaires d'un chemin renvoyé par la procédure ci-dessus.

II.4 Stratégies d'échantillonnage PRM

Plusieurs stratégies d'échantillonnage de nœuds ont été développées au fil des ans pour le PRM. Pour de nombreux problèmes de planification de trajectoire, un nombre étonnamment élevé de schémas d'échantillonnage généraux fournira des résultats raisonnables (voir par exemple la comparaison des schémas d'échantillonnage donnée dans [13]).

Cependant, sans aucun doute, le choix de la stratégie d'échantillonnage des nœuds peut jouer un rôle important dans les performances de la PRM. Cela a été observé dans les publications originales du PRM qui suggéraient des mécanismes pour générer des échantillons de manière non uniforme [12]. L'augmentation de la densité d'échantillonnage dans certaines zones de l'espace libre est appelée échantillonnage d'importance et il a été démontré à

plusieurs reprises qu'elle augmentait les performances observées des PRM. Dans cette section, nous décrivons plusieurs schémas d'échantillonnage de nœuds.

L'échantillonnage aléatoire uniforme utilisé dans les premiers travaux en PRM est le schéma d'échantillonnage le plus facile à mettre en œuvre. En tant que méthode d'échantillonnage aléatoire, elle a l'avantage que, en théorie, un adversaire malveillant ne peut pas vaincre le planificateur en construisant des entrées soigneusement conçues.

Il présente cependant l'inconvénient que, dans des exemples de planification difficiles, le temps d'exécution de PRM peut varier selon les différentes exécutions. Néanmoins, l'échantillonnage aléatoire fonctionne bien dans de nombreux cas pratiques impliquant des robots avec un grand nombre de degrés de liberté.

Il existe des cas où l'échantillonnage aléatoire uniforme a de mauvaises performances. Souvent, c'est le résultat du problème dit de passage étroit. Si un passage étroit existe dans Qfree et qu'il est absolument nécessaire de passer par ce passage pour résoudre une requête, un planificateur basé sur l'échantillonnage doit sélectionner un échantillon dans un ensemble potentiellement très petit afin de répondre à la requête de planification. Un certain nombre de méthodes d'échantillonnage différentes ont été conçues en tenant compte du problème du passage étroit et sont décrites ci-dessous.

Le problème des passages étroits reste un défi pour les planificateurs de PRM et constitue un domaine de recherche actif.

Le reste de cette section décrit les stratégies d'échantillonnage qui ont été développées en tenant compte du problème du passage étroit, puis d'autres stratégies d'échantillonnage générales.

II.4.1 Échantillonnage près des obstacles

Les méthodes d'échantillonnage basées sur les obstacles échantillonnent près de la frontière des obstacles de l'espace de configuration. La motivation derrière ce type d'échantillonnage est que les passages étroits peuvent être considérés comme des couloirs minces dans Qfree entourés d'obstacles. OBPRM [16] est l'un des premiers et très efficaces représentants des méthodes d'échantillonnage basées sur les obstacles. Initialement, OBPRM génère de nombreuses configurations au hasard à partir d'une distribution uniforme. Pour chaque configuration qin trouvée en collision, il génère une direction aléatoire v, et le planificateur trouve une configuration libre qout dans la direction v.

Enfin, il effectue une simple recherche binaire pour trouver la configuration libre la plus proche q de la surface de l'obstacle. La configuration q est ajoutée à la feuille de route, tandis que qin et qout sont rejetées.

L'échantillonneur gaussien [17] résout le problème du passage étroit en échantillonnant à partir d'une distribution gaussienne biaisée près des obstacles. La distribution gaussienne est obtenue en générant d'abord une configuration q1 de façon aléatoire à partir d'une distribution uniforme.

Ensuite, un pas de distance est choisi selon une distribution normale pour générer une configuration q2 au hasard au pas de distance de q1. Les deux configurations sont rejetées si les deux sont en collision ou si les deux sont sans collision. Un échantillon est ajouté à la feuille de route s'il est sans collision et que l'autre échantillon est en collision.

Dans [18], les échantillons sont générés dans un Qfree dilaté en permettant au robot de pénétrer par une petite distance constante dans les obstacles. La dilatation de Qfree élargit les passages étroits, ce qui permet au planificateur de capturer plus facilement la connectivité de l'espace. Au cours d'une deuxième étape, tous les échantillons qui ne se trouvent pas dans Qfree sont poussés dans Qfree en effectuant des opérations de rééchantillonnage local.

II.4.2 Échantillonnage à l'intérieur de passages étroits

Le planificateur de ponts utilise un test de pont pour échantillonner les configurations à l'intérieur de passages étroits. Dans un test de pont, deux configurations q′ et q″ sont échantillonnées de manière aléatoire à partir d'une distribution uniforme en Q. Ces configurations sont considérées pour être ajoutées à la feuille de route, mais si elles sont toutes deux en collision, alors le point qm à mi-chemin entre elle, est ajouté à la feuille de route si elle est sans collision. Ceci est appelé un test de pont car le segment de ligne entre q′ et q″ ressemble à un pont avec q′ et q″ à l'intérieur d'obstacles agissant comme des piliers et le point médian qm planant sur Qfree. Observez que la géométrie des passages étroits facilite la construction de ponts courts, tandis que dans les espaces ouverts, la construction de ponts courts est difficile. Cela permet au planificateur de ponts d'échantillonner des points à l'intérieur de passages étroits en favorisant la construction de ponts courts.
Une solution efficace au problème des passages étroits générerait des échantillons qui se trouvent à l'intérieur de passages étroits mais aussi loin que possible des obstacles. Les diagrammes de Voronoi généralisés (GVDs) ont exactement cette propriété. Bien que le calcul exact du GVD ne soit pas pratique pour les espaces de configuration de grande dimension, il est possible de trouver des échantillons sur le GVD sans le calculer explicitement.

Cela peut être fait par un schéma de rétraction. La rétraction est obtenue par une méthode de bissection qui déplace chaque configuration d'échantillon jusqu'à ce qu'elle soit équidistante de deux points sur la frontière de Qfree.
Une approche plus simple consiste à calculer le GVD de l'espace de travail et à générer des échantillons qui sont en quelque sorte conformes à ce GVD [27]. Par exemple, le robot peut avoir des points de poignée prédéfinis (par exemple, des points d'extrémité du plus long diamètre du robot) et l'échantillonnage peut placer ces points de poignée aussi près que possible du GVD dans l'espoir d'aligner l'ensemble du robot avec des passages étroits.
L'inconvénient de l'échantillonnage espace de travail-GVD est qu'il est en général difficile de générer des configurations du robot proches du GVD (les détails sont donnés dans [19].).
L'avantage de l'échantillonnage GVD de l'espace de travail est que le GVD capture des passages bien étroits dans l'espace de travail qui conduisent généralement à des passages étroits dans Qfree. De plus, une approximation du GVD de l'espace de travail peut être calculée efficacement en utilisant du matériel graphique [21], ce qui est l'une des raisons pour lesquelles cette méthode d'échantillonnage est populaire pour les visites virtuelles et les simulations associées.

II.4.3 Échantillonnage basé sur la visibilité

L'objectif du PRM basé sur la visibilité est de produire des feuilles de route de visibilité avec un petit nombre de nœuds en structurant l'espace de configuration en domaines de visibilité.
Le domaine de visibilité d'une configuration q comprend toutes les configurations qui peuvent être connectées à q par le planificateur local. Ce planificateur, contrairement à PRM qui accepte toutes les configurations libres générées au stade de la construction, ajoute à la feuille de route uniquement les configurations q qui satisfont à l'un des deux critères: (1) q ne peut être connecté à aucun nœud existant, c'est-à-dire que q est un nouveau ou (2) q relie au moins deux composants existants. De cette façon, le nombre de configurations dans la feuille de route est réduit.

II.4.4 Échantillonnage basé sur la manipulation

L'échantillonnage basé sur la manipulation [31] est une approche d'échantillonnage d'importance qui exploite la mesure de manipulation associée au manipulateur jacobien.

Intuitivement, la maniabilité caractérise la liberté de mouvement du bras pour une configuration donnée.
La motivation à utiliser la manipulation comme biais pour l'échantillonnage est la suivante. Dans les régions de l'espace de configuration où la manipulation est élevée, le robot a une grande dextérité, et donc relativement moins d'échantillons devraient être nécessaires dans ces zones.

Les régions de l'espace de configuration où la manipulation est faible ont tendance à être proches (ou à inclure) des configurations singulières du bras. Près des singularités, l'amplitude des mouvements possibles est réduite et, par conséquent, ces régions doivent être échantillonnées plus densément.
Soit J (q) la matrice jacobienne du manipulateur (c'est-à-dire la matrice qui relie les vitesses de l'effecteur terminal aux vitesses articulaires). Pour un bras redondant (par exemple, un bras avec plus de six articulations pour un espace de travail 3D), la maniabilité dans la configuration q est donnée par :

$$W(q)=\sqrt{det\, j(q)j^t(q)} \qquad \text{II-2}$$

Pour biaiser l'échantillonnage, une approximation de la fonction de densité cumulative (CDF) pour ω est créée. Les échantillons sont ensuite tirés d'une densité uniforme sur l'espace de configuration et rejetés avec une probabilité proportionnelle à la valeur CDF associée de leur valeur de manipulation.

II.4.5 Échantillonnage quasi aléatoire

Un certain nombre d'alternatives déterministes (parfois appelées quasi aléatoires) à l'échantillonnage aléatoire ont été utilisées. Ces alternatives ont d'abord été introduites dans le cadre de l'intégration Monte Carlo et visent à optimiser différentes propriétés de la distribution des échantillons.
L'utilisation de séquences quasi aléatoires présente l'avantage que le temps d'exécution est garanti identique pour toutes les exécutions en raison de la nature déterministe du processus de génération de points. Le planificateur résultant est une résolution terminée. L'analyse de la section 2.4 montre également pourquoi les séquences quasi aléatoires fonctionnent bien. Cependant, comme pour toute méthode d'échantillonnage déterministe, il est possible de construire des exemples où les performances du planificateur se détériorent. En guise de remède, il a été suggéré de perturber la séquence. La perturbation est obtenue en choisissant une configuration aléatoire à partir d'une distribution uniforme dans une petite zone autour du point d'échantillonnage ajouté à la séquence. La zone est progressivement réduite à mesure que de nouveaux points sont ajoutés à la séquence.
Certaines séquences quasi aléatoires peuvent également être vues comme générant des points dans une grille Multi-résolution dans Q.

II.4.6 Échantillonnage basé sur une grille

Les planificateurs basés sur des grilles sont apparus dans les premières publications sur la planification mais n'ont pas utilisé certaines abstractions clés de PRM telles que les primitives de vérification des collisions. Les nœuds d'une grille peuvent être une stratégie d'échantillonnage efficace dans le cadre PRM. Surtout lorsqu'ils sont combinés à des schémas de connexion de nœuds efficaces, ils peuvent entraîner des planificateurs puissants pour les problèmes survenant dans les milieux industriels [23]. Une manière naturelle d'utiliser la

recherche basée sur la grille dans un PRM consiste à utiliser une résolution plutôt grossière pour la grille et à tirer parti de l'abstraction de vérification des collisions; passer d'un nœud de grille q à un nœud de grille voisin q nécessiterait une vérification de collision, et donc un échantillonnage, à une résolution plus fine entre les nœuds. Pendant la phase de requête, des tentatives sont faites pour connecter qinit et qgoal aux points de grille voisins. La résolution de la grille utilisée pour construire la feuille de route peut être progressivement augmentée soit en ajoutant des points un par un, soit en ajoutant un hyperplan entier d'échantillons choisis pour combler le plus grand vide de la grille existante [23]. L'utilisation de séquences infinies basées sur des structures régulières, qui améliorent progressivement leur résolution, présente un intérêt particulier pour la planification de chemin.
Des travaux récents ont démontré l'utilisation de telles séquences pour construire des réseaux et d'autres structures régulières qui ont une structure de voisinage implicite, ce qui est très utile pour les PRM. Un algorithme de planification de chemin basé sur une grille est terminé.

II.4.7 Échantillonnage de connexion

L'échantillonnage de connexion [23] génère des échantillons qui facilitent la connexion de la feuille de route et peuvent être combinés avec toutes les méthodes d'échantillonnage décrites précédemment.

En règle générale, si un petit nombre de configurations est initialement généré, il peut exister quelques composants déconnectés à la fin de l'étape de construction. Si la feuille de route en construction est déconnectée dans un endroit où Qfree ne l'est pas, cet endroit peut correspondre à une zone difficile de Qfree, peut-être à un passage étroit de Qfree. L'idée sous-jacente à l'échantillonnage des connexions est de sélectionner dans la feuille de route un certain nombre de configurations susceptibles de se trouver dans ces régions et de les étendre. L'extension d'une configuration q implique de sélectionner une nouvelle configuration libre dans le voisinage de q comme décrit ci-dessous, en ajoutant cette configuration à la feuille de route et en essayant de la connecter à d'autres configurations de la feuille de route de la même manière que dans l'étape de construction. L'étape d'échantillonnage de connexion augmente la densité de la feuille de route dans les régions de Qfree qui sont considérées comme difficiles. Étant donné que les écarts entre les composants de la feuille de route sont généralement situés dans ces régions, la connectivité de la feuille de route est susceptible d'augmenter.

II.5 Choisir parmi différentes stratégies d'échantillonnage

Le choix entre différentes stratégies d'échantillonnage est une question ouverte. Ici, nous donnons quelques directives très approximatives sur la façon de choisir une stratégie d'échantillonnage.
Le succès du PRM doit être attribué en partie au fait que pour un large éventail de problèmes (problèmes difficiles mais pas «pathologiques», plusieurs stratégies d'échantillonnage simples fonctionnent bien. Par exemple, l'échantillonnage aléatoire uniforme fonctionne bien pour de nombreux problèmes rencontrés dans la pratique impliquant 3 à 7 degrés de liberté. Si l'uniformité de la durée de fonctionnement est un problème, l'échantillonnage quasi aléatoire et l'échantillonnage sur réseau offrent certains avantages. Lorsque la dimension augmente, et encore pour les problèmes qui ne présentent pas de comportement pathologique, l'échantillonnage aléatoire est la voie la plus simple à suivre. Lorsque des problèmes comportant des passages étroits sont pris en compte, des stratégies basées sur l'échantillonnage et conçues avec des passages étroits à l'esprit doivent être utilisées.
Des combinaisons de différentes méthodes d'échantillonnage sont possibles et, dans de

nombreux cas, essentielles à la réussite. Si πA et πB sont deux méthodes d'échantillonnage différentes, une méthode d'échantillonnage hybride pondérée π peut être produite en fixant π = (1 − w) πA + wπB. Par exemple, l'échantillonnage de connexion peut être utilisé en combinaison avec l'échantillonnage aléatoire ou l'échantillonnage OBPRM. Une stratégie d'échantillonnage peut également être considérée comme un filtre pour une autre.
Par exemple, l'échantillonneur gaussien peut être utilisé pour filtrer les nœuds créés selon le test de pont.
Aucune des méthodes d'échantillonnage décrites dans ce chapitre ne fournit clairement la meilleure stratégie pour tous les problèmes de planification. L'échantillonnage doit également être considéré en relation avec la stratégie de connexion utilisée et le planificateur local utilisé voir [12, 13]. Enfin, il faut souligner qu'il est possible de créer des instances de planification de parcours «pathologiques» qui seront arbitrairement difficiles pour tout planificateur basé sur l'échantillonnage.

II.6 Stratégies de connexion PRM

Un aspect important de PRM est la sélection de paires de configurations qui seront essayées pour les connexions par un planificateur local. L'objectif est de sélectionner les configurations pour lesquelles le planificateur local est susceptible de réussir. Comme cela a été discuté, un choix possible est d'utiliser le planificateur local pour connecter chaque configuration à tous ses k voisins les plus proches. La raison en est que les échantillons à proximité conduisent à de courtes connexions qui ont de bonnes chances d'être sans collision. Cette section présente quelques autres approches, leurs avantages et leurs inconvénients. De toute évidence, la fonction utilisée pour sélectionner les voisins et le planificateur local implémenté peut considérablement affecter les performances de toute stratégie de connexion décrite dans cette section.

II.6.1 Création de feuilles de route

Une méthode qui peut accélérer l'étape de construction de la feuille de route consiste à éviter le calcul des arêtes qui font partie du même composant connecté. Puisqu'il existe un chemin entre deux configurations dans un composant connecté, l'ajout du nouveau bord n'améliorera pas la connectivité de la feuille de route. Plusieurs implémentations de cette idée ont été proposées. Le plus simple est de connecter une configuration avec le nœud le plus proche dans chaque composant qui se trouve suffisamment près. Cette méthode évite de nombreux appels au planificateur local et accélère par conséquent l'étape de construction de la feuille de route.

Au fur et à mesure que le graphique est en cours de construction, les composants connectés peuvent être maintenus en utilisant une structure de données à ensemble disjoint rapide.
Avec la méthode ci-dessus, aucun cycle ne peut être créé et le graphique résultant est une forêt, c'est-à-dire une collection d'arbres. Puisqu'une requête ne réussirait jamais en raison d'un bord qui fait partie d'un cycle, il est en effet judicieux de ne pas consommer de temps et d'espace informatique et de stocker un tel bord. Dans certains cas, cependant, l'absence de cycles peut conduire la phase de requête à construire des chemins inutilement longs. Cet inconvénient peut être atténué en appliquant des techniques de post-traitement, telles que le lissage, sur le chemin résultant. Il a cependant été observé que permettre à certains bords redondants d'être calculés pendant la phase de construction de la feuille de route (par exemple, deux ou trois par nœud) peut améliorer considérablement la qualité du chemin d'origine sans surcharge significative [13]. Des travaux récents montrent comment ajouter des

cycles utiles dans les feuilles de route PRM qui donnent des chemins de meilleure qualité (plus courts) [25].

II.6.2 Connexion des composants connectés

La feuille de route construite par PRM vise à capturer la connectivité de Qfree. Dans certains cas, en raison de la difficulté du problème ou du nombre insuffisant d'échantillons générés, la feuille de route peut être constituée de plusieurs composants connectés.
La qualité de la feuille de route peut être améliorée en utilisant des stratégies visant à connecter différents composants de la feuille de route. L'échantillonnage des connexions tente de connecter différents composants de la feuille de route en plaçant plus de nœuds dans les régions difficiles de Qfree. La section II.2 décrit les planificateurs d'arbres basés sur l'échantillonnage qui peuvent être très efficaces pour connecter différents composants de la feuille de route. Ceci est exploité dans le planificateur .Des marches aléatoires et des planificateurs puissants tels que RPP [26] peuvent également être utilisés pour connecter des composants. D'autres stratégies sont décrites dans [27].

II.6.3 Évaluation paresseuse

L'idée derrière l'évaluation paresseuse est d'accélérer les performances en effectuant des vérifications de collision uniquement lorsque cela est absolument nécessaire. L'évaluation paresseuse peut être appliquée à presque tous les planificateurs basés sur l'échantillonnage présentés dans ce chapitre [23]. Dans cette section, l'évaluation paresseuse est décrite comme un schéma de connexion de nœud.

Lorsque l'évaluation paresseuse est utilisée, PRM fonctionne sur une feuille de route G, dont les nœuds et les chemins n'ont pas été entièrement évalués. On suppose que tous les nœuds et tous les bords d'un nœud à ses k voisins sont exempts de collisions. Une fois que PRM est présenté avec une requête, il connecte qinit et qgoal à deux nœuds proches de G. Le planificateur effectue ensuite une recherche graphique pour trouver le chemin le plus court entre qinit et qgoal, selon la fonction de distance utilisée. Ensuite, le chemin est vérifié comme suit. Tout d'abord, les nœuds de G sur le chemin sont vérifiés pour la collision. Si un nœud est trouvé en collision, il est retiré de G avec tous les bords qui en proviennent. Cette procédure est répétée jusqu'à ce qu'un chemin avec des nœuds libres soit découvert. Les bords de ce chemin sont ensuite vérifiés. Afin d'éviter des contrôles de collision inutiles, cependant, tous les bords le long du chemin sont d'abord vérifiés à une résolution grossière, puis à chaque itération, la résolution devient de plus en plus fine jusqu'à ce qu'elle atteigne la discrétisation souhaitée. Si un bord est trouvé en collision, il est supprimé de G. Le processus de recherche de chemins, de vérification de leurs nœuds puis de vérification de leurs bords est répété jusqu'à ce qu'un chemin libre soit trouvé ou que tous les nœuds de G aient été visités. Une fois qu'il est décidé qu'un nœud de G est en Qfree, cette information est enregistrée pour éviter de futures vérifications. Pour les bords, la résolution à laquelle ils ont été vérifiés pour la collision est également enregistrée de sorte que si un bord fait partie d'un chemin futur, les contrôles de collision ne sont pas répliqués. Si aucun chemin n'est trouvé et que les nœuds de G ont été épuisés, de nouveaux nœuds et arêtes peuvent être ajoutés à G. Les nouveaux nœuds peuvent être échantillonnés non seulement au hasard mais aussi dans les régions difficiles de Qfree [27].

Un schéma paresseux connexe attribue une probabilité à chaque bord d'être sans collision. Cette probabilité est calculée en tenant compte de la résolution à laquelle l'arête a été vérifiée. Les probabilités de bord peuvent être utilisées pour rechercher un chemin dans G qui a de bonnes chances d'être dans Qfree.

II.7 PARCOURS DES GRAPHES

Beaucoup de problèmes sur les graphes nécessitent que l'on parcoure l'ensemble des sommets et des arcs/arêtes du graphe.
Les deux principales stratégies d'exploration sont: le parcours en largeur consiste à explorer les sommets du graphe niveau par niveau, à partir d'un sommet donné ; le parcours en profondeur consiste, à partir d'un sommet donné, à suivre un chemin le plus loin possible, puis à faire des retours en arrière pour reprendre tous les chemins ignorés précédemment.
Ces algorithmes procèdent de même manière, par coloriage des sommets :
Initialement, tous les sommets sont coloriés en blanc (traduisant le fait qu'ils n'ont pas encore été "découverts").Lorsqu'un sommet est "découvert" (autrement dit, quand on arrive pour la première fois sur ce sommet), il est colorié en gris. Le sommet reste gris tant qu'il reste des successeurs de ce sommet qui sont blancs (autrement dit, qui n'ont pas encore été découverts). Un sommet est colorié en noir lorsque tous ses successeurs sont gris ou noirs (autrement dit, lorsqu'ils ont tous été découverts). De façon pratique, on va utiliser une liste "d'attente au coloriage en noir" dans laquelle on va stocker tous les sommets gris : un sommet est mis dans la liste d'attente dès qu'il est colorié en gris. Un sommet gris dans la liste d'attente peut faire rentrer dans la liste ses successeurs qui sont encore blancs (en les coloriant en gris). Quand tous les successeurs d'un sommet gris de la liste d'attente sont soit gris soit noirs, il est colorié en noir et il sort de la liste d'attente.

La différence fondamentale entre le parcours en largeur et le parcours en profondeur provient de la façon de gérer cette liste d'attente au coloriage en noir : le parcours en largeur utilise une file d'attente (FIFO), où le premier sommet arrivé dans la file est aussi le premier à en sortir, tandis que le parcours en profondeur utilise une pile (LIFO), où le dernier sommet arrivé dans la pile est le premier à en sortir.

II.7.1 Arborescence couvrante associée à un parcours

Au fur et à mesure du parcours, l'algorithme construit une arborescence de découverte des sommets accessibles depuis le sommet de départ V_0, appelée arborescence d'un parcours à partir de v_0. Cette arborescence contient un arc (v_i, v_j) si et seulement si le sommet v_j a été découvert à partir du sommet v_i (autrement dit, si c'est le sommet v_i qui a colorié v_j en gris). Ce graphe est effectivement une arborescence, dans la mesure où chaque sommet a au plus un prédécesseur, à partir duquel il a été découvert. La racine de cette arborescence est le sommet de départ du parcours, v_0. L'arborescence associée à un parcours de graphe est mémorisée dans un tableau $\prod$ tel que $\prod[v_j] = v_i$ si v_j a été découvert à partir de v_i, et $\prod[v_k] = null$ si v_k est la racine, ou s'il n'existe pas de chemin de la racine vers v_k.

II.7.2 Parcours en largeur (Breadth First Search = BFS)

Le parcours en largeur est effectué en gérant la liste d'attente au coloriage comme une file d'attente (FIFO = First In First Out). Autrement dit, l'algorithme enlève à chaque fois le plus vieux sommet gris dans la file d'attente (ce sommet sera nommé $S_{FirstOut}$), et il introduit tous les successeurs blancs de ce sommet dans la file d'attente, en les coloriant en gris. L'algorithme 1 décrit ce principe.

Algorithme 1 : parcours_en_largeur(BFS) (S, A, s0, Π, d)

```
   Fonction BFS (g, v0)
 Entrées : V ensemble des sommets, A ensemble des arc/arêtes, v0 sommet de départ
    Sorties :   Π arborescence couvrante, d tableau longueur des chemins depuis v0
 1  Declaration : couleur tableau des couleurs des sommets, File
 2   init_File(F)
 3  pour tout sommet vi ∈ V faire
 4     Π [vi 4] ← nil
 5     d [vi 5] ← ∞
 6     couleur [Vi 6] ← blanc
 7    d [v0] ← 0
 8    ajoute_fin_File (F, v0)
 9     couleur [v0] ← gris
 10    tant que est_vide(F) = faux faire
 11       enleve_debut_File (F, vi)
 12       pour tout vj ∈ succ (vi) faire
 13          si couleur [vj] = blanc alors
 14             ajoute_fin_File (F, vj)
 15              couleur [vj] ← gris
 16              Π [vj] ← vi
 17              d [vj] ← d [vi] + 1
 18       couleur [vi] ← noir
```

Complexité : Soient n et p le nombre de sommets et arcs de g, respectivement. Chaque sommet (accessible depuis v0) est mis, puis enlevé, une fois dans la file. À chaque fois qu'on enlève un sommet de la file, on parcourt tous ses successeurs, de telle sorte que chaque arc (ou arête) du graphe sera utilisé une fois dans l'algorithme. Par conséquent, si le graphe contient n sommets (accessibles à partir de v0) et p arcs/arêtes, alors BFS sera en :
- O(n 2) dans le cas d'une implémentation par matrice d'adjacence,
- O(n + p) dans le cas d'une implémentation par listes d'adjacence.

II.7.3 Parcours en profondeur (Depth First Search = DFS)

Le parcours en profondeur est obtenu en gérant la liste d'attente au coloriage en noir comme une pile (LIFO = Last In First Out). Autrement dit, l'algorithme considère à chaque fois le dernier sommet gris entré dans la pile, et introduit devant lui tous ses successeurs blancs. Ce principe est décrit dans l'algorithme 2

Algorithme 2 : Parcours en profondeur d'un graphe

```
1  Fonction DFS (g, v0)
     Entrée : Un graphe g et un sommet v0 de g
     Postcondition : Retourne une arborescence  d'un parcours en profondeur de g à partir de v0
      Déclaration : Une pile (LIFO) p initialisée à vide
2     pour tout sommet vi ∈ V faire
3        Π [vi] ← null
4        Colorier vi en blanc
5     Empiler v0 dans p et colorier v0 en gris
6     tant que la pile p n'est pas vide faire
7      Soit si le dernier sommet entré dans p (au sommet de p)
8        si ∃vj ∈ succ (vi) tel que vj soit blanc alors
9          Empiler vj dans p et colorier vj en gris
10         Π [vj] ← v
11       sinon
12       Dépiler vi de p et colorier vi en noir
```

Complexité : Chaque sommet (accessible depuis v0) est mis, puis enlevé, une fois dans la pile, comme dans BFS. Par conséquent, si le graphe contient n sommets (accessibles à partir de v0) et e arcs/arêtes, alors DFS sera en :
- O (n^2) dans le cas d'une implémentation par matrice d'adjacence,
- O (n + p) dans le cas d'une implémentation par listes d'adjacence.

II.8 Les problèmes de chemins optimaux

Les problèmes de chemins optimaux sont très fréquents dans les applications pratiques. On les rencontre dès qu'il s'agit d'acheminer un objet entre deux points d'un réseau, de façon à minimiser un coût, une distance ou une durée. Ils apparaissent aussi en sous-problèmes combinatoires, notamment les flots dans les graphes et les ordonnancements. Tout ceci a motivé très tôt la recherche d'algorithmes efficaces.
Définition : Soit G = (V, E) un graphe valué, on associe à chaque arc e = (vi, vj) une longueur l(e).
Le problème du plus court chemin entre vi et vj est de trouver un chemin µ (vi, vj) de vi à vj tel que :

$$l(\mu) = \sum_{u \epsilon \mu} l(u) \text{ soit Minimale} \qquad \text{II-3}$$

Plusieurs algorithmes sont proposées pour résolue ce problème, ces algorithmes sont différents suivant les propriétés des graphes (l(e) $\geq 0, \forall e \epsilon E$, G sans circuit,. . .) et suivant le problème considérée (recherche du plus court chemin d'un sommet vers tous les autres, ou bien entre deux sommets,. . .). Nous présentons deux algorithmes Dijkstra et A*.

II.8.1 Algorithme de Dijkstra

L'algorithme de Dijkstra est un algorithme créé en 1959 par un mathématicien hollandais, M. Dijkstra. Il est utilisé pour résoudre le problème de chemin le plus court. D'autres algorithmes tels que ceux de Floyd-Warshall ou Bellman-Ford ont été publiés durant la même décennie et constituent une variante de celui de Dijkstra, chacun ayant sas particularités
Son principe de fonctionnement est comme suit :
Numérotons les sommets du graphe G = (V, E) de 1 à n. Supposons que l'on s'intéresse aux chemins partant du sommet i. On construit un vecteur λ = (λ(1), λ(2), . . . , λ(n)) ayant n composantes tel que λ(j) soit égal à la longueur du plus court chemin allant de i au sommet j. On initialise ce vecteur à ci,j , c'est-à-dire à la première ligne de la matrice des coûts du graphe, définie comme indiqué ci-dessous :

$$ci,j = \begin{cases} 0 \; si \; i = j \\ \infty \neq j \; et \; (i,j) \notin E \\ \delta(i,j) \neq j \; et \; (i,j) \in E \end{cases} \qquad II\text{-}4$$

Où δ (i, j) est le poids (la longueur) de l'arc (i, j). Les ci,j doivent être strictement positifs. On construit un autre vecteur p pour mémoriser le chemin pour aller du sommet i au sommet voulu. La valeur p(i) donne le sommet qui précède i dans le chemin. On considère ensuite deux ensembles de sommets, S (visités) initialisé à {i} et T (non visités) initialisé à {1, 2, 3, . . . , n} − i. A chaque itération de l'algorithme, on ajoute à S un sommet jusqu'à ce que S = V de telle sorte que le vecteur λ donne à chaque étape le coût minimal des chemins de i aux sommets de S. [9]

On suppose ici que le sommet de départ (qui sera la racine de l'arborescence) est le sommet 1.

Algorithme 3 Algorithme de Dijikstra
Entrées: *G (V, E)*
1: **Pour** j ← 1 n **Faire**
2: λ(j) = ci,j , p(j) = NIL;
3: Fin **Pour**
4: S= { } ; **T**= {1,2,3,. . . ,n} ;
5: **Tant que T** n'est pas vide **Faire**
6: Choisir i dans **T** tel que λ(i) est minimum ;
7: Retirer i de **T** et l'ajouter à S ;
8: **Pour** chaque adjacent j de i, avec j dans **T Faire**
9: **Si** λ(j) > λ(i) + σ(i, j) **Alors**
10: λ(j) ← λ(i) + σ(i, j);
11: p(j) ← i;
12: Fin **Si**
13: Fin **Pour**
14: Fin **Tant que**

II.8.2 L'algorithme A*

L'algorithme A* a été créé pour que la première solution trouvée soit l'une des meilleures. Cet algorithme a été proposé pour la première fois par Peter E. Hart, Nils John Nilsson et Bertram Raphael en 1968. Il s'agit d'une extension de l'algorithme de Dijkstra de 1959. Il s'agit d'un algorithme [11]heuristique de programmation dynamique qui fournit généralement une solution approchée. L'algorithme A* est un algorithme de recherche de chemin dans un graphe entre un nœud initial et un nœud final. Il utilise une évaluation heuristique[1] sur chaque nœud pour estimer le meilleur chemin y passant, et visite ensuite les nœuds par ordre de cette évaluation heuristique.

→ Algorithme simple,

→ Ne nécessitant pas de prétraitement, et

→ Ne consommant que peu de mémoire

Algorithme très utilisé pour sa rapidité (jeux vidéo, ...). C'est un algorithme de recherche d'un plus

[1]Heuristique est une technique qui améliore l'efficacité d'un processus de recherche, en sacrifiant éventuellement l'exactitude ou l'optimalité de la solution.

court chemin dans un graphe. Il est basé sur une évaluation heuristique à chaque sommet pour estimer le meilleur chemin qui y passe. [11]

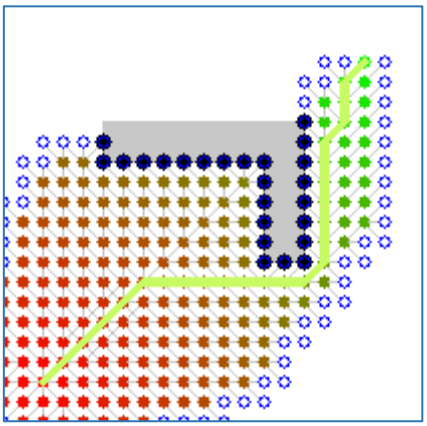


Figure II-6Illustration de l'algorithme A.*

Principe de fonctionnement est comme suit :

Au début on donne à chaque sommet dans le graphe G(V, E) une valeur heuristique h(n) : l'estimation du coût entre ce sommet et le sommet désirer (dans la planification du chemin on peut utiliser la distance euclidienne ou autres distances), ensuit à chaque itération la méthode choisie le chemin qui minimise le critère

$$f(u) = g(u) + h(u)$$

Où g(u) est le poids entre le sommet courant et le sommet adjacent, et h(u) est une heuristique admissible (i.e., une borne inférieure de la distance séparant les deux sommets (ces distances peuvent être calculé de différente manière; distance euclidienne, distance de Manhattan ou autre.

A* utilise deux listes, ces listes contiennent des nœuds d'un graphe. La première liste, appelée liste ouverte, va contenir tous les nœuds étudiés. Dès que l'algorithme va se pencher sur un nœud du graphe, il passera dans la liste ouverte (sauf s'il y est déjà).La seconde liste, appelée liste fermée, contiendra tous les nœuds qui, à un moment où à un autre, ont été considérés comme faisant partie du chemin solution. Avant de passer dans la liste fermée, un nœud doit d'abord passer dans la liste ouverte, en effet, il doit d'abord être étudié avant d'être jugé comme bon.

Pour déterminer si un nœud est susceptible de faire partie du chemin solution, il faut pouvoir quantifier sa qualité. Néanmoins, une méthode souvent utilisée et qui donne de bons résultats est de mesurer l'écartement entre ce nœud et le chemin à vol d'oiseau. On calcule donc la distance entre le point étudié et le dernier point qu'on a jugé comme bon. Et on calcule aussi la distance entre le point étudié et le point de destination. La somme de ces deux distances nous donne la qualité du nœud étudié. Plus un nœud à une qualité faible, meilleur il est. La recherche du chemin commence par le premier point, en étudiant tous ses voisins, en calculant leur qualité, et en choisissant le meilleur pour continuer. Chaque point étudié est mis dans la liste ouverte et le meilleur de cette liste passe dans la liste fermée, il va servir de base

pour la recherche suivante.

Ainsi, à chaque itération on va regarder parmi tous les nœuds qui ont été étudiés (et qui n'ont pas encore été choisis) celui qui a la meilleure qualité. Et il est tout à fait possible que le meilleur ne soit pas un voisin direct du point courant. Cela signifiera que le point courant nous éloigne de la solution et qu'il faut corriger le tir.

L'algorithme s'arrête quand la destination a été atteinte ou bien lorsque toutes les solutions mises de côté ont été étudiées et qu'aucune ne s'est révélée bonne, c'est le cas où il n'y a pas de solution. Dans l'algorithme suivant, on suppose que la fonction heuristique est monotone.

Algorithm 4 Pseudo-code de l'algorithme A*

```
   Entrées: G (V, E),
  S_init, S_désiré Sorties : Chemin
1:     H := Les valeurs heuristiques de tous les sommets ;
2:          Liste_ouverte := {S_init} ;
3:         Liste_fermée := {S_init} ;
4:     g:={0} ;
5:     f:={H(S_init)} ;
6:     Liste_provenance := {} ; Noeuds := {} ;
7: Tant que la Liste_ouverte n'est pas vide Faire
8:      Courant := Le sommet dans Liste_ouverte qui a le f minimum ;
9: Pour chaque adjacent de courant Faire
10:   Si l'adjacent est dans la Liste_fermée Alors
11:     Continue ;
12:          Fin Si
13:       Dist := E(Courant, adjacent) + g[Courant] ;
14: Si l'adjacent est dans Liste_ouverte Alors
15:    Si Dist < g[adjacent] Alors
16:       g[adjacent]:= Dist ;
17:       f[adjacent]:= g[adjacent] + H[adjacent] ;
18:       Liste_provenance [adjacent] := Courant ;
19:   Continue ;
20:        Si non
21:   Continue ;
22:       Fin Si
23:      Fin Si
24:   Ajouter l'adjacent à Liste_ouverte ;
25:   Ajouter E(Courant,adjacent) + g[Courant] à g ;
26:   Ajouter g[dernière] + H[adjacent] à f ;
27 :  Ajouter Courant à la Liste_provenance et ajouter l'adjacent à Noeuds ;
28:   Si Noeuds[dernière] est S_désiré Alors
29:  Returner Chemin := reconstruitChemin(Liste_provenance, Noeuds) ;
30:    Fin Si
31:  Fin Pour
32:  Enlever Liste_ouverte[Courant] ;
33:  Ajouter Courant à la Liste_fermée ;
 34:    Enlever g[Courant] ;
 35:    Enlever f[Courant] ;
36:  Fin Tant que
37:  Returner Pas de chemin ;
38: Fonction reconstruitChemin(Liste_provenance, Noeuds)
39: Chemin := {S_dsir} ;
 40: Tant que Chemin[dernière] ≠ S_init Faire
41    k := l'indice de Chemin[dernière] dans Noeuds ;
42:   ajouter Liste_provenance [k] à Chemin ;
43:  Fin Tant que
44:   Returner Chemin
45: Fin Fonction
```

Conclusion

Nous avons soulevé le problème de la planification de trajectoire et présenter quelque méthode pour résoudre ce problème. Ainsi Nous avons présenté dans ce chapitre deux méthodes pour parcourir un graphe: Parcours en largeur (Breadth First Search) et Parcours en profondeur (Depth First Search).En a vue aussi deux planificateurs de chemin optimaux ; Nous avons ainsi présenté dans ce chapitre des outils très important pour réaliser un planificateur de chemin par la carte de route probabiliste.

III. Chapitre3. Planification de mouvement d'un manipulateur RRR par PRM

III.1 Introduction :

Dans ce chapitre, nous allons donner l'étude de la carte de route probabiliste PRM et son implémentation pour le robot RRR que nous avons considéré. La construction de la PRM passe par différentes étapes. Elle commence par l'échantillonnage d'un ensemble de configurations hors collision. Le choix de ces configurations est fait aléatoirement en utilisant l'une des méthodes existantes (distribution uniforme ou en utilisant l'Importance Sampling). Ensuite, on passe à la connexion de la PRM qui consiste en le choix de paire de configurations voisines ensuite la recherche d'un chemin sans collision entre elles. Cette recherche est faite en utilisant un planificateur local. Si ce dernier arrive à trouver un chemin entre les deux configurations un arc connectant les deux configurations est rajouté à la PRM. A la fin de cette phase, nous obtenons une carte de route probabiliste prête pour l'utilisation pour des requêtes de planifications. Pour chaque requête, on donne la configuration initiale et finale et le planificateur procède à leur connexion à la PRM en utilisant le même planificateur local. Après une recherche est lancée dans le graphe que représente la PRM. Cette recherche est effectuée en utilisant l'un des algorithmes de recherche dans un graphe. Si un chemin existe il sera retourné par le planificateur. Pour améliorer la qualité du chemin planifié, un traitement postérieur est effectué sur ce chemin. Différentes techniques existent pour les étapes déjà citées et pour la maintenance de la carte de route.

Ce travail étend l'étude de [5] aux cas de scènes 3D pour robots manipulateurs. Le changement le plus important réside dans le planificateur local. C'est pour cela que nous allons lui dédié un traitement détaillé. Nous commençons par spécifier le robot RRR que nous avons considéré.

III.2 Robot à 3 ddl de type RRR:

Le robot que nous avons considéré est représenté par la figure (III-1). La position donnée détermine la pose initiale où toutes les coordonnées articulaires sont nulles. L'axe x du repère $\{w\}$ est orienté vers la droite, l'axe y est orienté vers l'intérieur et l'axe z vers le haut. Le repère $\{0\}$ a son axe x orienté vers l'intérieur de la page. Les transformations qui permettent de passer vers ce repère sont données par :

```
T_link1_0=eye(4,4);
T_link2_0=[0 1 0 0;0 0 -1 (0.5-0.025/2);-1 0 0 0.025;0 0 0 1];
T_link3_0=[0 1 0 -(0.5-0.025);0 0 -1 (0.5-0.025/2);-1 0 0 0;0 0 0 1];
```

Les axes de rotations sont les axes z des repères attachés aux segments et orienté vers le haut pour la première articulation et vers la gauche pour les deux autres. Les segments sont des polyèdres rectangulaires dont les dimensions sont spécifiées sur la figure. Le choix de la forme des segments est fait pour obéir à certaines suppositions sur lesquelles sont basées certaines fonctions que nous avons programmées.

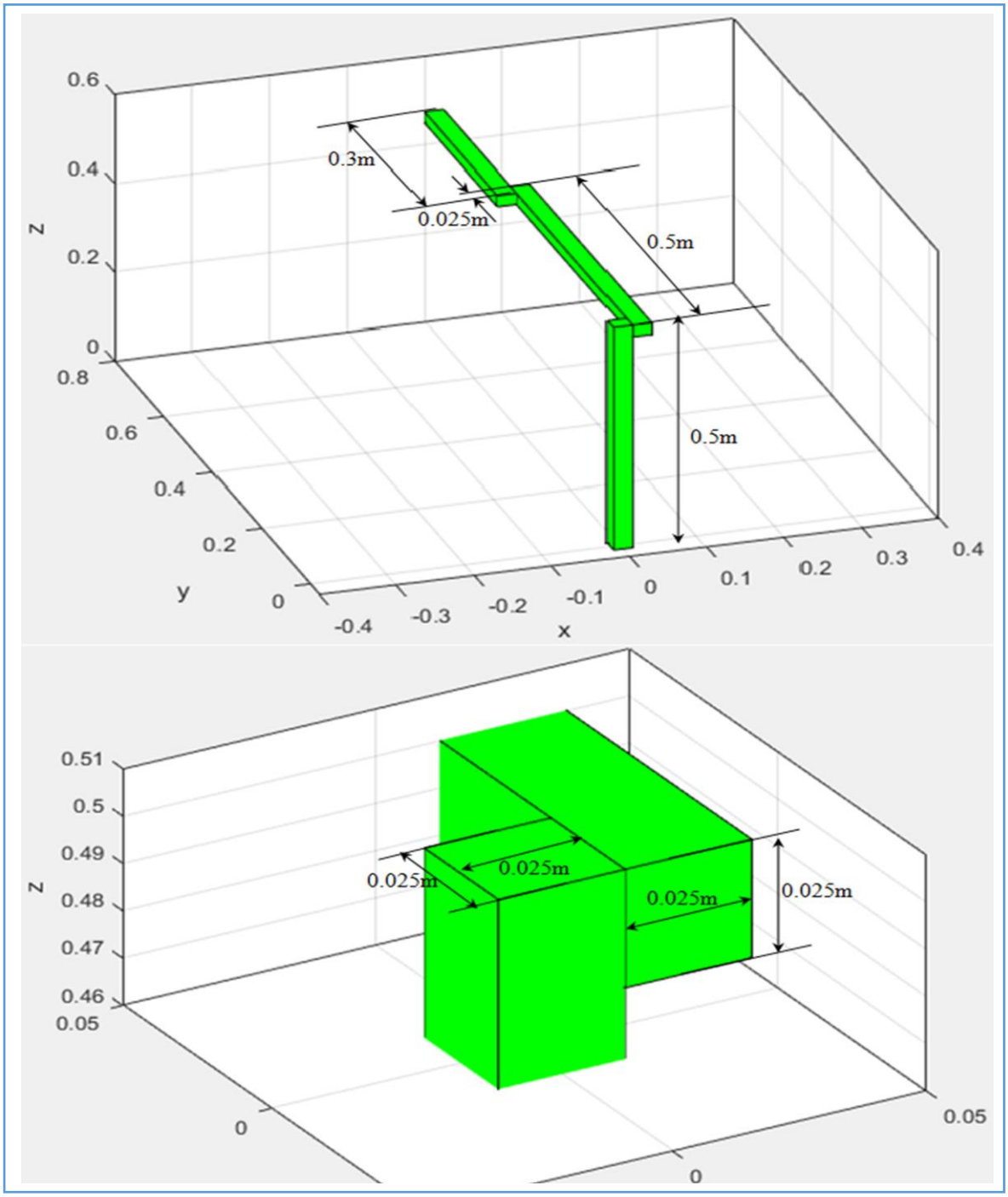


Figure III-1 Robot de 3 DLL (RRR) considéré.

III.3 Echantillonnage aléatoire de configurations :

Sans doute le choix de la stratégie d'échantillonnage peut jouer un rôle très important dans l'amélioration des performances de la PRM. L'échantillonnage uniforme (selon une distribution uniforme) est la technique la plus simple à implémenter. Il a l'avantage que les cas particulièrement difficiles peuvent être résolus par le planificateur si on lui donne assez de temps. Cependant, l'inconvénient est que le temps d'exécution peut changer beaucoup entre différentes requêtes. Néanmoins, l'échantillonnage uniforme a donné des résultats très satisfaisants pour beaucoup de cas impliquant des robots avec un nombre élevé de degrés de liberté.

L'augmentation de la densité d'échantillonnage dans une région de l'espace hors collision s'appelle 'Importance Sampling'. Souvent, ceci est utile contre certains problèmes comme la présence de passages étroits à travers desquels il est nécessaire de passer pour répondre à des requêtes de planification. Différentes techniques d'échantillonnage existent dans cette famille : échantillonnage près des obstacles, échantillonnage à l'intérieur des passages étroits, échantillonnage basé sur la visibilité …

III.3.1 Choix aléatoire d'une configuration selon une distribution uniforme:

Dans ce travail nous avons choisi l'échantillonnage uniforme. L'échantillonnage de la position selon une distribution uniforme est facile à accomplir, car il suffit de choisir les trois coordonnées x, y et z selon une distribution uniforme chacune. Dans MATLAB, ceci est réalisé en utilisant l'instruction rand().

Le cas à trois dimensions avec un corps solide implique en plus l'orientation. L'orientation peut être représentée de différentes façons (matrice de rotation, quaternion, Angles d'Euler, Axe-angle …). Dans ce cas, l'échantillonnage de la position se fait toujours de la même façon. Cependant, pour l'orientation, il a été montré que le choix selon une distribution uniforme de certains paramètres de l'orientation ensuite l'obtention des autres par le calcul n'obéit pas toujours à une distribution uniforme. Une façon de faire dont la validité est prouvée est d'utiliser la représentation par quaternion d'une rotation.

$$Q = \left[\sqrt{1-\xi_0}\,sin(2\pi\xi_1) \quad \sqrt{1-\xi_0}\,cos(2\pi\xi_1) \quad \sqrt{\xi_0}\,sin(2\pi\xi_2) \quad \sqrt{\xi_0}\,cos(2\pi\xi_2)\right]^T$$

$$= [w \quad x \quad y \quad z]^T \quad avec \quad w^2 + x^2 + y^2 + z^2 = 1 \qquad (III1)$$

Les paramètres ξ_0, ξ_1 et ξ_2 sont choisis selon une distribution uniforme pour chacun.

Pour un robot manipulateur RRR, les trois articulations sont purement rotoïdes avec 1 ddl et une façon valide de choisir aléatoirement les coordonnées articulaires θ_1, θ_2 et θ_3 est de choisir chacun par une distribution uniforme. C'est la méthode que nous avons adopté dans ce travail.

III.4 Connexion de la PRM :

Après qu'un ensemble de configuration ait été choisi aléatoirement, l'étape suivante consiste à connecter des paires parmi ces configurations pour lesquelles un chemin existe permettant au robot de passer d'une configuration à l'autre. L'existence de ce chemin est décidé en utilisant un planificateur local qui doit être simple et rapide. Le choix de ces paires de configurations doit augmenter les chances d'obtenir un chemin entre elles. L'idée est de choisir des configurations voisines pour que le chemin les reliant soit très court réduisant ainsi les chances de collision. La méthode utilisant ce concept est appelée méthode des k-voisins.

III.4.1 Choix des k voisins (k-nearest neighbors) :

Le problème que nous allons traiter ici est comment choisir les configurations voisines. Pour le cas d'un robot ponctuel où l'orientation n'est pas prise en considération, il suffit de prendre la distance euclidienne entre deux positions comme une fonction distance qui permettra de décider quelles sont les configurations qu'il faut considérer. Pour généraliser ceci au cas où le robot est un corps solide (représenté ici par un polyèdre), la fonction distance (qu'on appelle aussi métrique) doit être choisie de telle sorte qu'elle reflète la probabilité que le planificateur local échoue à trouver un chemin hors collision entre les deux configurations. Une solution alternative est de partager la fonction distance en deux parties. Une partie concerne la distance entre les deux configurations due à la translation et la deuxième concerne la différence dans l'orientation. Par exemple si le vecteur X représente la translation entre deux configurations et la matrice R représente la différence dans l'orientation $((X, R) \in SE(3)$, alors :

$$dist(q', q'') = w_t \|X' - X''\| + w_r f(R', R'') \qquad \text{III2}$$

Est une métrique pondérée avec comme composante de translation $\|X' - X''\|$ qui est la norme euclidienne standard et une fonction scalaire positive qui donne typiquement une mesure approximative de la distance entre les deux rotations dans $R', R'' \in SO(3)$. La rotation est mise à l'échelle par rapport à la translation en utilisant les poids w_t et w_r . Une bonne sélection de la fonction ($R^{'}$, $R^{''}$) est la longueur de la courbe géodésique (le plus grand arc sur le cercle unitaire dans l'espace 4D entre deux quaternion) entre$R^{'}$, $R^{''} \in SO(3)$.

Une difficulté avec cette approche réside dans la sélection des poids. De plus, son extension aux cas de systèmes articulés n'est pas facile. Quelques choix pour la fonction ($R^{'}$, $R^{''}$) sont :

Pour les angles d'Euler $(\theta_1, \phi_1, \eta_1)$, $(\theta_2, \phi_2, \eta_2)$:

$$f(R', R'') = w_r \sqrt{\Delta(\theta_1, \theta_2)^2 + \Delta(\phi_1, \phi_2)^2 + \Delta(\eta_1, \eta_2)^2} \qquad \textit{III-3}$$

Avec

$$\Delta(\theta_1, \theta_2) = \begin{cases} \theta_2 - \theta_1 + 2\pi & si\ (\theta_2 - \theta_1) < -\pi \\ \theta_2 - \theta_1 & si - \pi < (\theta_2 - \theta_1) < \pi \\ \theta_2 - \theta_1 - 2\pi & si\ (\theta_2 - \theta_1) > \pi \end{cases} \qquad \text{III-4}$$

La même expression est utilisée pour $\Delta(\phi_1, \phi_2)$, $\Delta(\eta_1, \eta_2)$. Dans ce travail, nous avons utilisé cette méthode pour calculer la distance entre deux angles ceci est justifié par le fait que chaque articulation a un seul ddl et par conséquent un axe de rotation fixe.

III.5 Planificateur local :

Le planificateur local doit être simple et rapide et il doit retourner un résultat vrai si une trajectoire hors collision existe entre les deux nœuds choisis et un résultat faut au cas contraire. Plusieurs planificateurs locaux existent parmi lesquels on peut citer les planificateurs à temps optimal, les planificateurs basés sur le champ de potentiel, segment linéaires $q = q_i + (q_j - q_i)t \; avec \; t \in [0,1]$, rotate-at-s, méthode incrémentale, méthode binaire... Dans ce travail nous allons utiliser un planificateur basé sur le calcul des intervalles interdits des coordonnées articulaires de façon incrémentale en partant de la première articulation ensuite la déduction de l'espace de configuration hors collision sous forme d'une collection de cellules. Un graphe de régions, représentant la connectivité de cette espace, est ensuite construit pour être utilisé pour des requêtes de planification de chemin. Nous donnons une étude détaillée de cet algorithme en plus d'un bref aperçu de deux autres à cause de leur simplicité.

III.5.1 Méthode incrémentale :

Dans cette méthode, on fait des petits pas le long de la trajectoire entre q_i, etq_j et on vérifie la configuration obtenue après chaque pas pour collision. Ainsi, à partir d'une configuration q_i on se déplace vers l'une des configurations directement voisines. Par exemple, pour un robot ponctuel dans un espace 3D, les configurations directement voisines de q_j sont $q_j + (\{-\varepsilon, 0, \varepsilon\}^3 - \{(0,0,0)\})$.Une définition analogue pour un robot manipulateur à n degrés de liberté est $q_j + (\{-\varepsilon, 0, \varepsilon\}^n - \{(0,0, \dots ,0)\})$. Une façon de choisir le pas ε est de le choisir en fonction de déplacement maximale parmi les déplacement de tous les points du robot pendant le mouvement élémentaire, ou bien choisir ε de façon indépendante ensuite amplifié le volume du robot par un rapport ε.

III.5.2 Méthode binaire :

La logique derrière cette méthode est que vu que les configurations q_i et q_j sont hors collision alors il en serait de même pour les configurations qui leurs sont proches et par conséquent la configuration située au milieu à plus de chance à être en collision. Ainsi, on commence par tester cette configuration contre la collision et si elle l'est il n'y pas de chemin entre la paire q_i, q_j et on aura rapidement trouvé la réponse. Si la configuration du milieu est hors collision, alors on test les deux configurations situées aux milieux des deux moitiés contre la collision selon la même logique. On continu de cette façon en testant à chaque fois 2^n configurations jusqu'à ce qu'une collision soit détectée où que l'on atteint une certaine tolérance concernant le pas.

III.5.3 Planificateur basé sur le calcul des intervalles interdits des coordonnées articulaires

Ce planificateur est basé sur le calcul d'une caractérisation approximative de l'espace de configurations hors collision en le déduisant à partir du calcul des intervalles interdits des coordonnées articulaires. Nous commençons par montrer comment cette approximation sera obtenue.

Au début, nous commençons par introduire la notion de projection de tranche de l'espace de configurations en collision (C-space obstacles) dénoté C_O. Pour un manipulateur de n ddl, l'espace des configurations en collision C_O est de dimension n. Nous représentons des approximations de C_O par l'union de tronches de dimension $(n-1)$ [Lozano_auto]. Chaque tronche représente un ensemble de configurations de dimension (n) qui diffèrent seulement par la valeur d'une seule coordonnée articulaire.

Soit $\theta = [\theta_1 \quad \cdots \quad \theta_n]^T$ le vecteur de coordonnées articulaires de dimension n dont les éléments représentent l'angle de rotation pour les articulations rotoïde ou le déplacement pour les articulations prismatiques. Soit S un ensemble de points θ, c'est à dire un sous ensemble de C-space. La projection en tranche de l'ensemble S pour des valeurs de $\theta_j \in [a_j, b_j]$ est définit par :

$$\Pi_{[a_j,b_j]}(S) = \{\theta = (\theta_1, \theta_2, \dots, \theta_{j-1}, \theta_{j+1}, \dots, \theta_n) | \theta \in S \; and \; \theta_j \in [a_j, b_j]\} \qquad \text{III5}$$

La projection en tranche est un volume de dimension (n-1). La figure suivante représente (dans l'espace 2D) la projection en tranche d'un disque de rayon 1 et centre (0,0) pour des valeurs de $\theta_2 \in [a_2, b_2]$= [1,2].

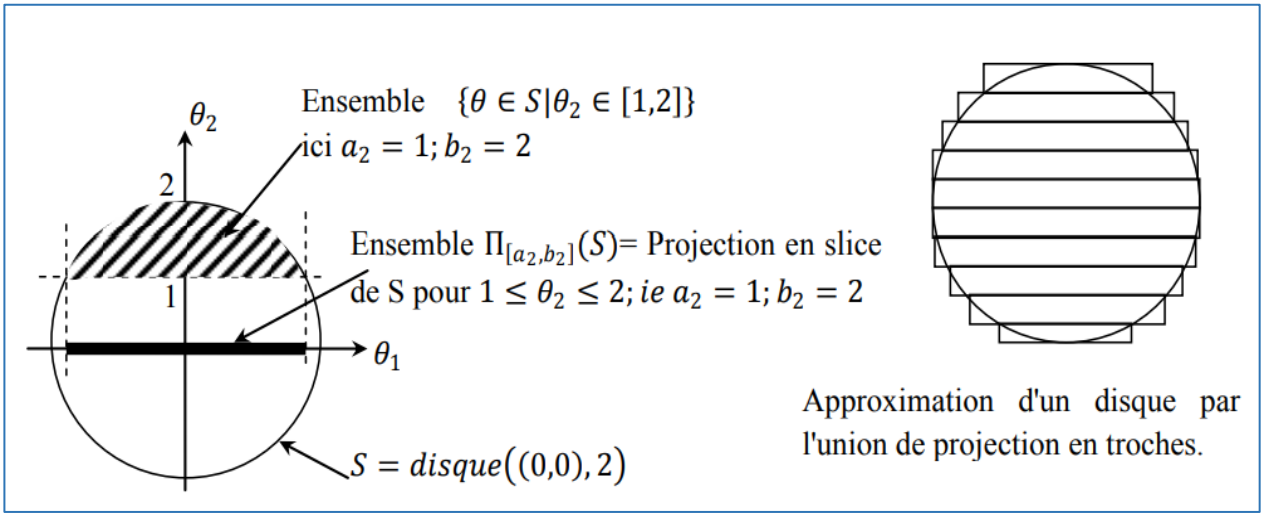


Figure III-2 Projection en tronche d'un disque de l'espace de dimension 2 (plan).

La projection en tronche d'un obstacle C pour$\theta_j \in [a_j, b_j]$ est $\Pi_{[a_j,b_j]}$.

Notons que la projection en tronche est une approximation conservative d'un segment d'un espace de dimension n qui est le C-space obstacles. Une approximation de l'obstacle entier est construite par l'union d'un ensemble de projections en tronches de dimension (n-1). Chacune d'elle correspond à un intervalle différent de valeurs de la même variable articulaire. La figure précédente illustre ceci. Chacune des projection en troche de dimension (n-1) peut aussi, a son tour, être approximée par l'union de projections en tronches de dimension (n-2) et ainsi de suite jusqu'à ce qu'on obtient l'union d'intervalles de dimension 1.

Si pour une application donnée on arrive à calculer le C-space obsatcles, on peut par complémentation obtenir l'espace de configurations libres (hors collision) qui est très précieux pour la planification. Une technique pour calculer l'espace de configurations hors collision est basée sur la caractérisation des surfaces des obstacles dans l'espace des configurations. L'inconvénient de cette technique est que les équations de ces surfaces sont difficiles à résoudre numériquement. Pour des manipulateurs simples (séquence de segment sans branches) contenant des articulations rotoïdes ou prismatiques, la procédure suivante peut être utilisée pour calculer l'espace de configuration sans collision de n segments :

1. $\theta_0$est une valeur constante sans intérêt. Pour $i = 1,2, \dots, n$ do :
2. Pour chaque valeur légale (hors collision) de θ_{i-1} :
3. Ignorer les segments après le segment i. Trouver les intervalles légaux de θ_i en faisant tourner (ou translater pour le cas prismatique) le segment i autour de la position actuelle de l'articulation i obtenue à partir des valeurs échantillonnées des coordonées $\theta_1, \dots, \theta_{i-1}$.
4. Si $i = n$ retourner les intervalles obtenues pour θ_n sinon échantillonner les intervalles de θ_i avec la résolution spécifiée, $i \leftarrow (i + 1)$, Aller à l'étape 2.

Noter que les calculs de base qui doivent être effectués sont ceux de la détermination des intervalles légaux pour un paramètre d'une articulation étant données les valeurs des paramètres des articulations précédentes. Une procédure pour faire ces calculs sera donnée après.

La nature récursive de la procédure de calcul de l'espace de configuration suggère l'utilisation d'une structure de données récursive. Dans notre cas, nous avons utilisé un arbre dont la profondeur est $(n - 1)$ pour un manipulateur à n ddls et dont le facteur de branchement est égale au nombre d'intervalles légaux de l'articulation. Les feuilles sont les intervalles légaux pour l'articulation n. Plusieurs intervalles dans les nœuds de l'arbre n'auront pas de descendants car elles produisent une collision pour un segment donné. La figure (III-3) représente un arbre d'intervalles légaux pour un robot à 3 ddls.

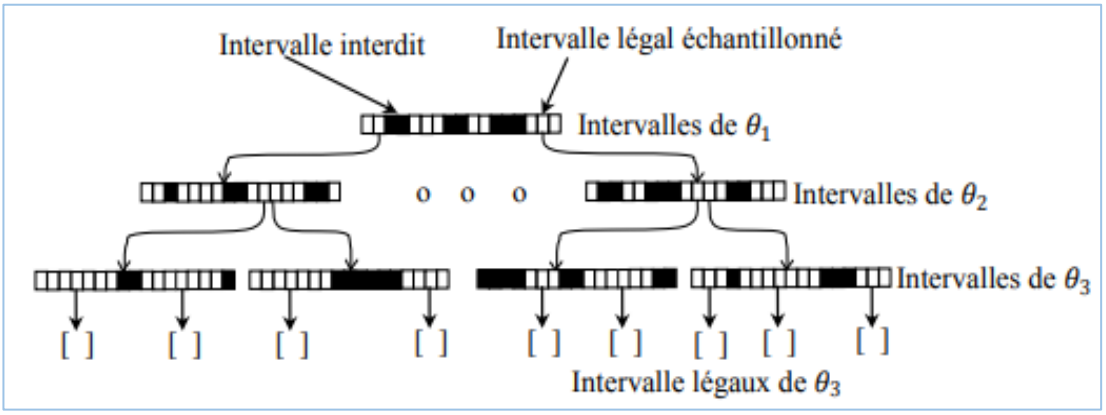


Figure III-3 Arbre des intervalles interdit et des intervalles légaux d'un robot à 3ddls

Dans notre programme nous avons représenté cet arbre de la façon suivante : Pour chaque degré de liberté θ_i on prend une combinaison des valeurs échantillonnées des angles précédentes et on associe une cell-array à cette combinaison. Ensuite, on calcul les intervalles permis de θ_i pour cette combinaison et on effectue leur échantillonnage. Après, pour chaque échantillon ainsi obtenue on associe une cell-array qu'on va utiliser de la même façon pour θ_{i+1}. Cette structure est représentée par la figure (III-4).

Ici il faut noter qu'en effectuant l'échantillonnage des angles articulaires, nous sommes entrain de considérer que ces derniers varient dans de petits intervalles de longueur $\Delta\theta$, mais ce que nous considérons dans le calcul de l'intervalle correspondant du segment suivant et seulement la valeur centrale. Pour garantir que des collisions ne soient manquées, il faut soit choisir un intervalle $\Delta\theta$ petit ou augmenter le volume du segment. Cette dernière solution est effectuée en calculant (de façon conservative) le plus grand déplacement δ_k parmi les déplacements des points du segment k quand les angles des articulations précédentes varient dans leurs intervalles d'échantillonnage. Ensuite, on choisi un polyèdre régulier qui borde la sphère de rayon δ_k et on fait sa somme de Minkovski avec le polyèdre représentant le segment k. Cette somme est le nouveau polyèdre représentant le segment k pour garantir l'évitement de collision. Nous n'avons pas implémenté cette procédure et nous comptons seulement sur l'augmentation de la résolution pour surmonter ce problème.

La structure récursive de l'espace C-free donne une idée sur la méthode de son calcul qui doit être récursive.

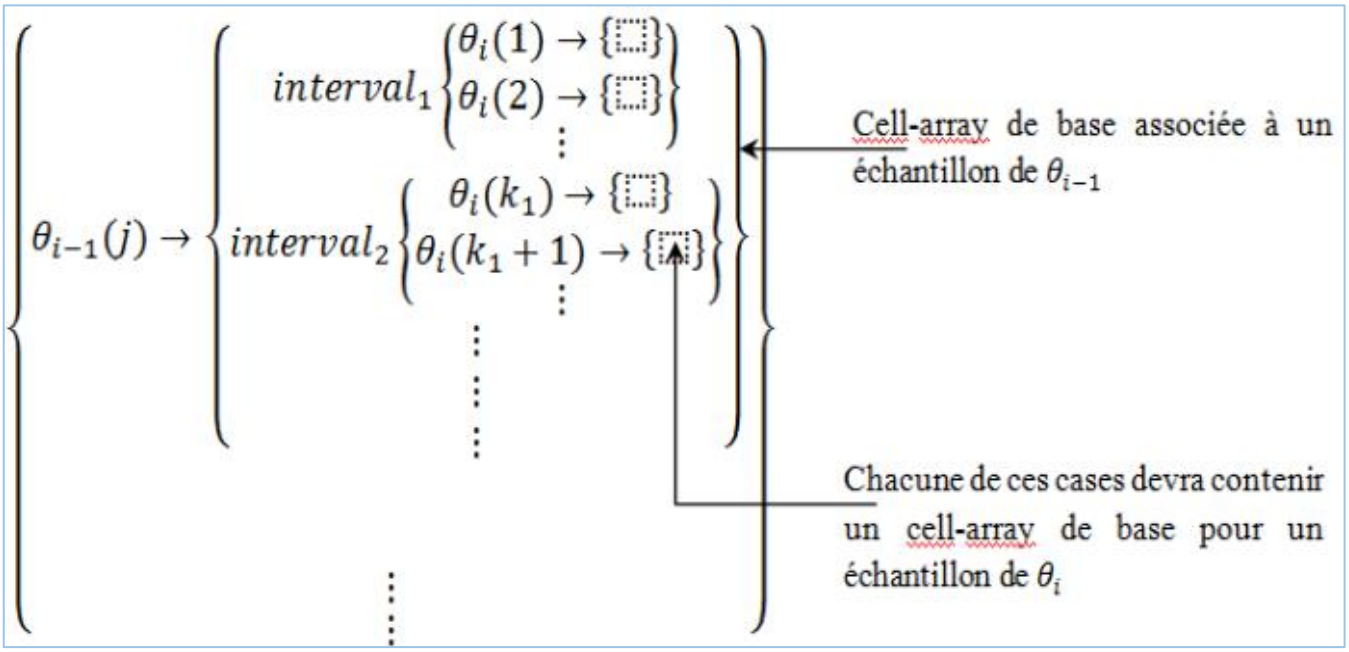


Figure III-4Cell-array de base associée à chaque échantillon de θ_{i-1}

Maintenant, on arrive au problème de base de comment calculer les intervalles légaux pour une articulation donnée, étant données les valeurs des paramètres des articulations précédentes. Pour le cas où les segments du robot et les obstacles sont des polyèdres convexes, ce problème peut être résolu analytiquement. Les intervalles interdits de θ_k seront bornés par les valeurs des angles (translations) pour lesquels le segment k du robot touche (entre en contact sans collision) l'un des obstacles, on appelle cette valeur de θ_k point de contacts. Pour les polyèdres, les points de contacts sont de trois types :

A. Un vertex d'un obstacle et une face d'un segment.

B. Un vertex du segment et une face d'un obstacle.

C. Une arrête d'un segment et une arrête d'un obstacle.

Chacun de ces types est appelé applicabilité et doit répondre à trois contraintes qui sont :

1. La contrainte d'appartenance (in-face constraint) : Une fois qu'on détermine le point de contact entre le plan contenant de la face de l'obstacle et le vertex du robot, il faut vérifier que ce point est à l'intérieur de la face et pas seulement sur son plan.
2. Contrainte d'orientation : Permet de connaître si le point est seulement un point de contact (le segment touche seulement l'obstacle) et pas un point d'intersection. Pour ceci, les vecteurs partant du vertex en question vers les vertex de l'autre côté des arrêtes émanant de ce vertex doivent pointer en dehors du volume de l'obstacle. Ceci est testé en vérifiant le signe des produits scalaires de ces vecteurs avec le normal à la face n voir figure (III-6b).
3. La troisième contrainte s'appelle contrainte d'accessibilité et concerne seulement le cas où les obstacles ne sont pas convexes (ayant des faces non accessibles) ce qui n'est pas permis sous nos suppositions.

Nous commençons par le calcul des points de contacts de type B. Nous considérons que le repère du segment à la position initiale est tel que l'axe de rotation de l'articulation et orienté dans la direction de l'axe z et que l'axe x est orienté le long du segment. L'axe y complète le repère. L'angle de rotation θ est celui entre l'axe x du repère attaché au segment avec l'axe x à la position initiale voir figure (III- 5). On doit aussi disposer de la transformation homogène T_{LW} qui permet de représenter le segment et les obstacles dans ce repère local connaissant leurs représentations dans le repère du monde.

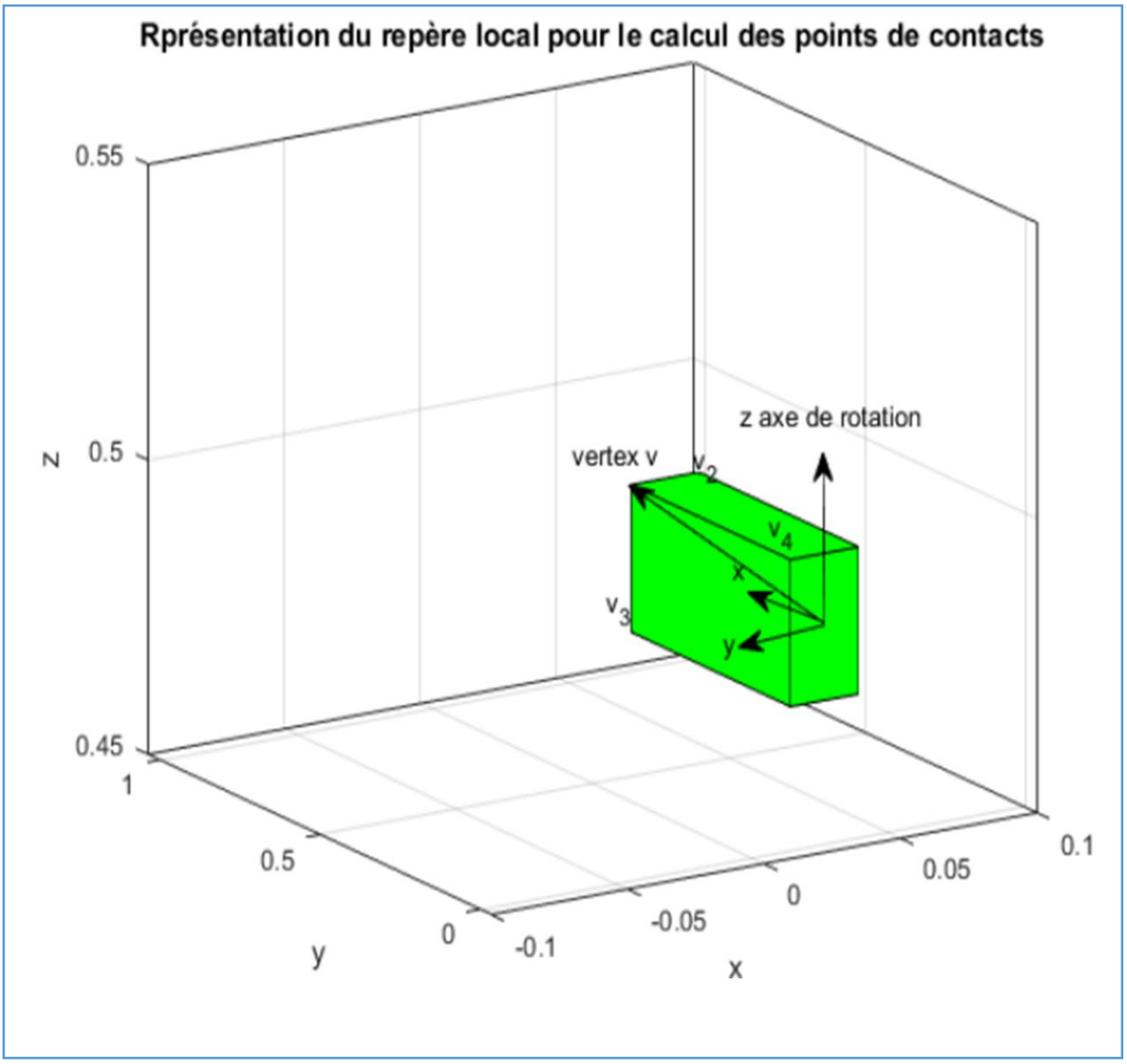
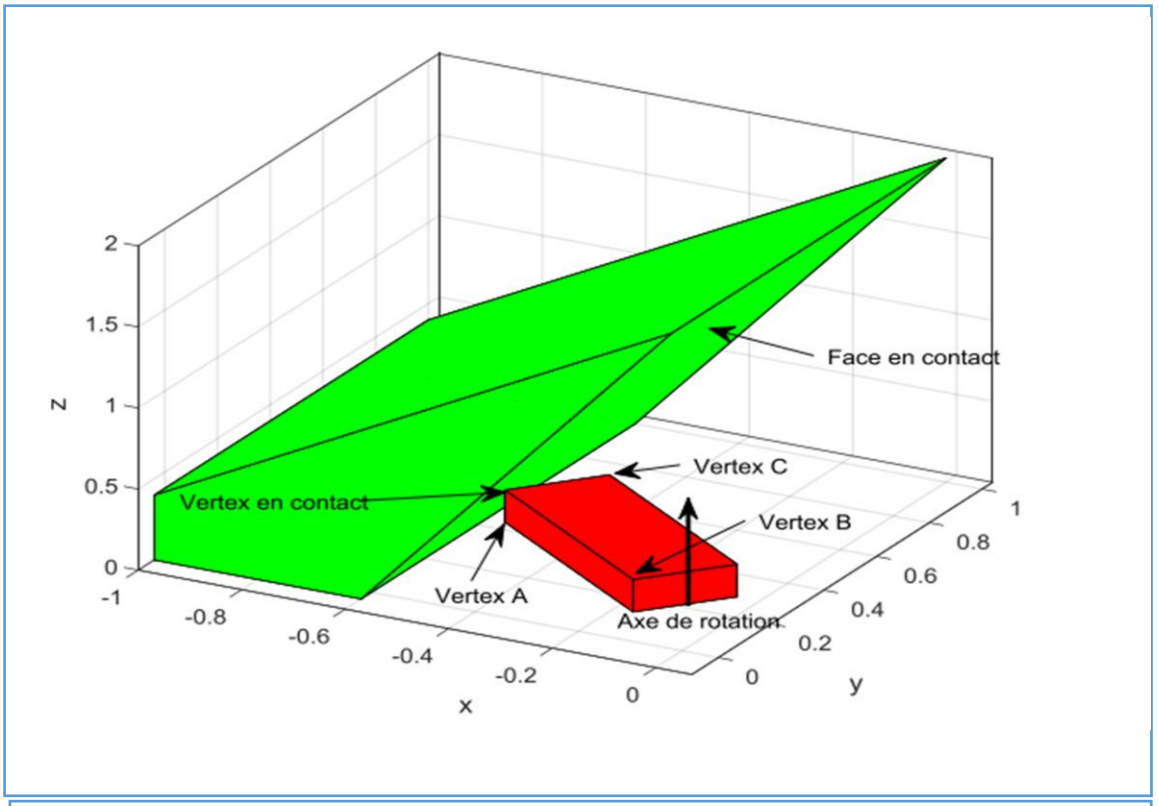


Figure III-6 Association du repère local pour le calcul des points de contacts



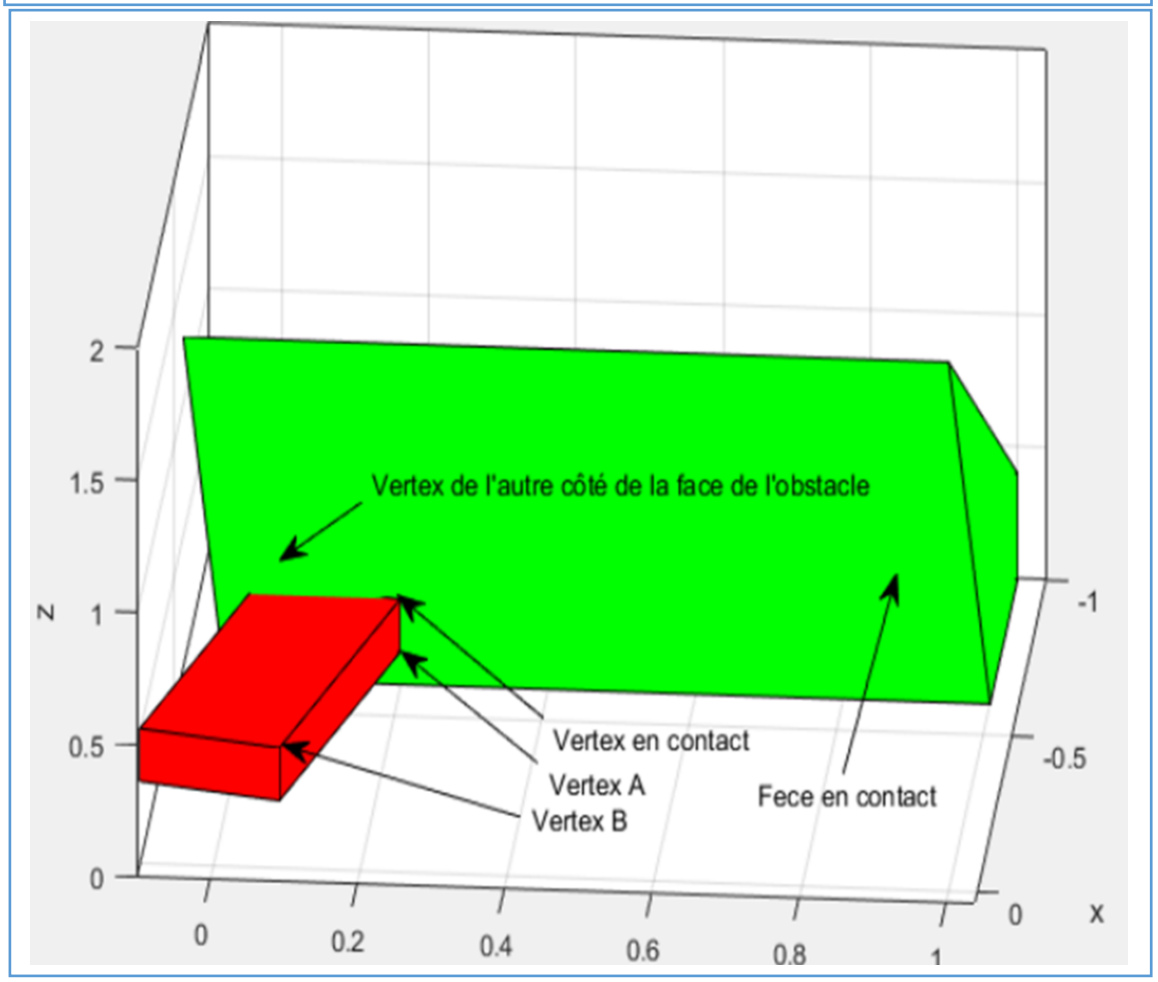


Figure III-7 contacte de type B : Un Vertex d'un segment avec une face d'un obstacle.

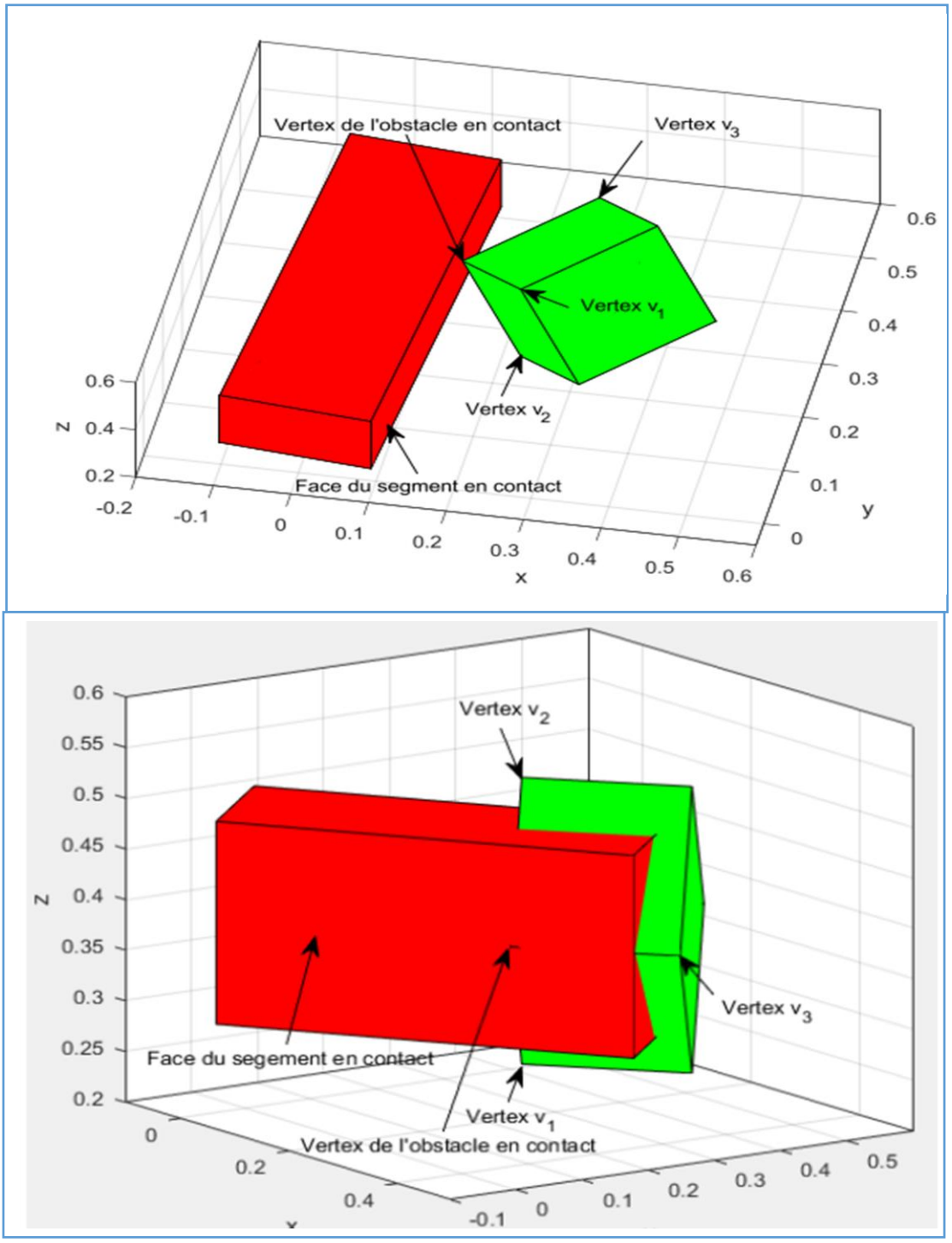


Figure III-8 Contact d'une face du segment avec un vertex de l'obstacle
a) (thet=-0.0876rad). Les trois autres vertex délimitant les arêtes qui joignent au vertex en contacts doivent être en dehors du segment. Ceci est une applicabilité valide.
*b) Les trois vertex de l'obstacle (**v1, v2 et v3**) sont situées du côté de l'intérieur du segment par rapport à la face de contact. Ceci n'est pas une applicabilité valide.*

Supposons que c'est le vertex v du segment qui entre en contact avec la face F de l'obstacle. Le vertex v est représenté par son vecteur de position v et la face F par son équation dans le repère du segment donnée par $n \cdot x + d = 0$ avec n le vecteur unitaire normal à la face. Les coordonnées du vecteur v après rotation remplacent x dans l'équation de la face pour donner une équation en θ (l'angle de contact) :

$$(n_x v_x + n_y v_y)\cos\theta + (n_y v_x - n_x v_y)\sin\theta = -d - n_z d_z \qquad \text{III6}$$

Le terme de droite de cette équation donne la distance du vecteur v après rotation de la face de l'obstacle. Le signe de la dérivée de cette quantité par rapport à θ ($sign(-(n_x v_x + n_y v_y)\sin\theta + (n_y v_x - n_x v_y)cos\theta)$permet de connaître si c'est l'incrémentation ou la décrémentation de θ qui cause la collision. La résolution de cette équation pour θ donne :

$$\theta = arccos\left(\frac{-n_z v_z - d}{\sqrt{(v_x^2 + v_y^2)(1 - n_z^2)}} + arctan(n_y v_x - n_x v_y, n_x v_x + n_y v_y)\right) \qquad \textit{III-7}$$

Pour les points de contacts de type A, les équations resteront valables il suffit seulement d'inverser les rôles entre l'obstacle et le segment voir figure (III-8).

Le calcul des points de contacts de type C est un peu plus difficile. Au début, l'arrête du segment et l'arrête de l'obstacle sont décrites par leurs équations paramétriques comme suit :

$$\begin{aligned} f_s &= m_s t_s + v_s \\ f_o &= m_o t_o + w_o \end{aligned} \qquad \text{III8}$$

avec v_s et w_o sont les vecteurs positions de l'une des vertex de l'arrête du segment et de l'arrête de l'obstacle respectivement. m_s est le vecteur représentant l'arrête en contact du segment et il est donné par $v_s - v_{s2}$ où v_{s2} est le vecteur position du deuxième vertex de la même arrête. m_o est défini de façon similaire.

Pendant la rotation, les points de l'arrête du segment décrivent un cercle parallèle au plan xy et dont l'équation est donnée par :

$$\begin{aligned} x^2 + y^2 &= (m_{sx} t_s + v_{sx})^2 + (m_{sy} t_s + v_{sy})^2 \\ z &= m_{sz} t_s + v_{sz} \end{aligned} \qquad \text{III9}$$

Ces deux équations peuvent être combinées pour donner une équation ne contenant pas t_s :

$$x^2 + y^2 = \left(m_{sx}\frac{(z - v_{sz})}{m_{sz}} + v_{sx}\right)^2 + \left(m_{sy}\frac{(z - v_{sz})}{m_{sz}} + v_{sy}\right)^2 \qquad \text{III10}$$

Le point de contact sur l'arrête de l'obstacle doit aussi appartenir au même cercle, ceci permet de substituer $x, y\ et\ z$ de l'équation paramétrique de l'obstacle dans l'équation précédente pour obtenir une équation quadratique en t_o :

$$pt_o^2 + qt_o + r = 0$$
$$p = (m_{ox}^2 + m_{oy}^2)m_{sz}^2 - (m_{sx}^2 + m_{sy}^2)m_{oz}^2$$
$$q = 2[(m_{ox}w_{ox} + m_{oy}w_{oy})m_{sz}^2 - (m_x^2 + m_y^2)(w_{oz} - v_{sz})m_{oz} - (m_{sx}v_{sx} + m_{sy}v_{sy})m_{sz}m_{oz}] \quad \text{III-11}$$
$$r = (w_{ox}^2 + w_{oy}^2)m_{sz}^2 - [(m_{sx}(w_{oz} - v_{sz}) + v_{sx}m_{sz})^2 + (m_{sy}(w_{oz} - v_{sz}) + v_{sy}m_{sz})^2]$$

Ayant t_o,, nous pouvons obtenir t_s du moment qu'on sait que les valeurs de z au point de contact doivent être les mêmes :

$$t_s = \frac{m_{oz}t_o + w_{oz} - v_{sz}}{m_{sz}} \quad \text{III12}$$

Après avoir obtenu les valeurs de t_o et t_s, nous devons tout d'abord vérifier qu'ils sont dans l'intervalle [0,1]ce qui constitue la contrainte d'appartenance (the in Edge constraint) Ensuite, on calcule le point d'intersection sur chacune des deux arrêtes (L pour le segment et O pour l'obstacle) :

$$\theta = arctan(l_x o_y - l_y o_x, l_x o_x + l_y o_y) \quad \text{III13}$$

Ces équations sont seulement valables pour le cas où $m_{sz} \neq 0$. Quand c'est le cas (le segment est complètement horizontal) alors dans ce cas le cercle décrit par le segment à $z = v_{sz}$ et le point de contact sur l'arrête de l'obstacle doit avoir $z = m_{oz}t_o + w_{oz} = v_{sz} \quad \therefore \quad t_o = (v_{sz} - w_{oz})/m_{oz}$. L'équation du cercle décrit par l'arrête du segment donne :

$$(m_{sx}t_s + v_{sx})^2 + (m_{sy}t_s + v_{sy})^2 = (m_{ox}t_o + w_{ox})^2 + (m_{oy}t_o + w_{oy})^2 \quad \text{III14}$$

Puisque on connait la valeur de t_o ceci est une équation quadratique de t_s. Les solutions doivent vérifier $t_o, t_s \in [0,1]$. La dérivée de la distance entre les deux arrêtes est plus difficile à obtenir, nous donnons ici seulement son expression :

$$\begin{aligned} dérivée = &\Big((w_{oy}m_{sx} - w_{ox}m_{sy})m_{oz} + ((v_{sz} - w_{oz})m_{sx} - v_{sx}m_{sz})m_{oy} \\ &- \Big((v_{sz} - w_{oz})m_{sy} - v_{yx}m_{sz}\Big)m_{ox}\Big)ksin\theta \\ &+ \Big((w_{oy}m_{sy} + w_{ox}m_{sx})m_{oz} + \Big((v_{sz} - w_{oz})m_{oy} - v_y m_{sz}\Big)m_{oy} \\ &+ ((v_{sz} - w_{oz})m_{sx} - v_{sx}m_{sz})m_{ox}\Big)kcos\theta \end{aligned} \quad \text{III15}$$

avec $k = sign((m_s \times m_o) \cdot (e_1 + e_2))$ .Les vecteurs e_1 et e_2 sont représentés sur la figure (III-8). Pour la contrainte d'orientation les équations suivantes doivent être vérifiées :

$$\begin{gathered} s = sgn(c \cdot e_1) = sgn(c \cdot e_2) \\ s' = sgn(c \cdot e_3) = sgn(c \cdot e_4) \quad \text{III16} \\ s \neq s' \end{gathered}$$

Des équations semblables peuvent être établies pour le cas d'une articulation prismatique. Dans ce travail nous avons considéré seulement des articulations rotoïdes.

Nous avons utilisé ces équations pour programmer une fonction qui calcul les points de contacts avec le signe de la dérivée de la distance de l'obstacle par rapport à θ pour un segment en rotation autour de l'axe z et en présence d'un ensemble d'obstacles. Cette fonction est la base des calculs nécessaires pour obtenir l'arbre des configurations permises pour un robot manipulateur. Nous avons appliqué cette fonction à un exemple pour obtenir les intervalles interdits de l'angle de rotation d'un polyèdre autour de l'axe z en présence d'un ensemble d'obstacles de forme polyédrique. Les détails de cette exemple ainsi que les résultats obtenus sont données dans par la figure (III-9). Le programme calculs les points de contacts pour chaque obstacle, ensuite il combine les points de contacts obtenus pour tous les obstacles avec leurs signes pour produire les intervalles interdits (qui conduisent à une collision) de l'angle de rotation. Ces intervalles interdits sont ensuite complémentés pour obtenir les intervalles permis (espace des configurations hors obstacles).

La fonction discutée précédemment est utilisée pour calculer de façon récursive les intervalles interdits d'un robot manipulateur de n articulations rotoides. Jusqu'à ce niveau le nombre d'articulations rotoides n'est pas limité

Après avoir calculé l'arbre représentant les configurations permises en utilisant la structure de données récursive. L'espace de configurations hors collision peut être représenté sous forme d'une matrice de 3D (pour le robot RRR) qui constitue un bitmap où chaque cellule qui est hors collision contiendra un 1. Cette matrice à elle-même est très précieuse pour la planification selon la méthode utilisée. Dans la figure (III-10), nous avons tracé les poses du robot pour un ensemble choisi aléatoirement de configurations parmi les configurations présentées comme configurations permises. Nous constatons que ces configurations sont toutes hors collision.

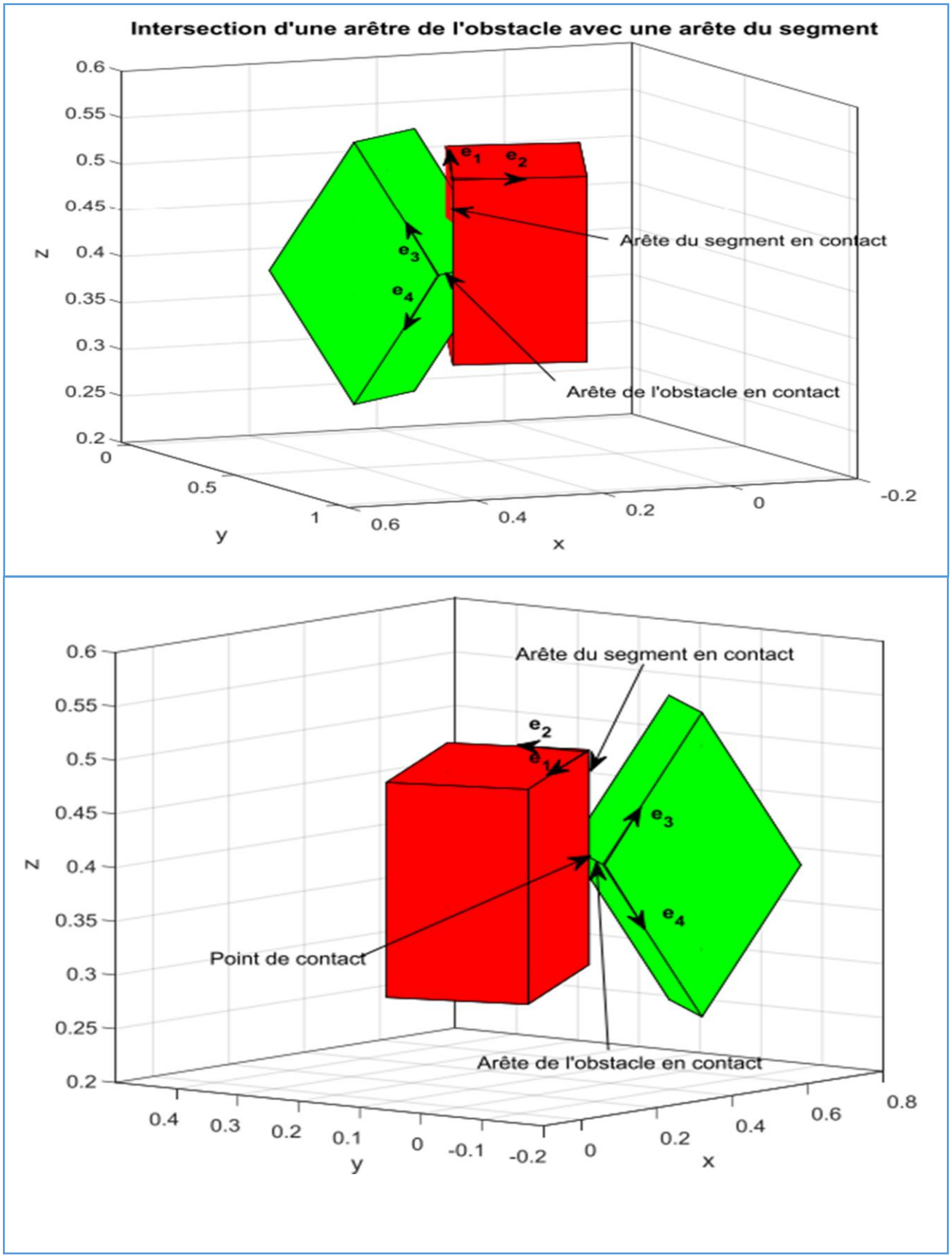
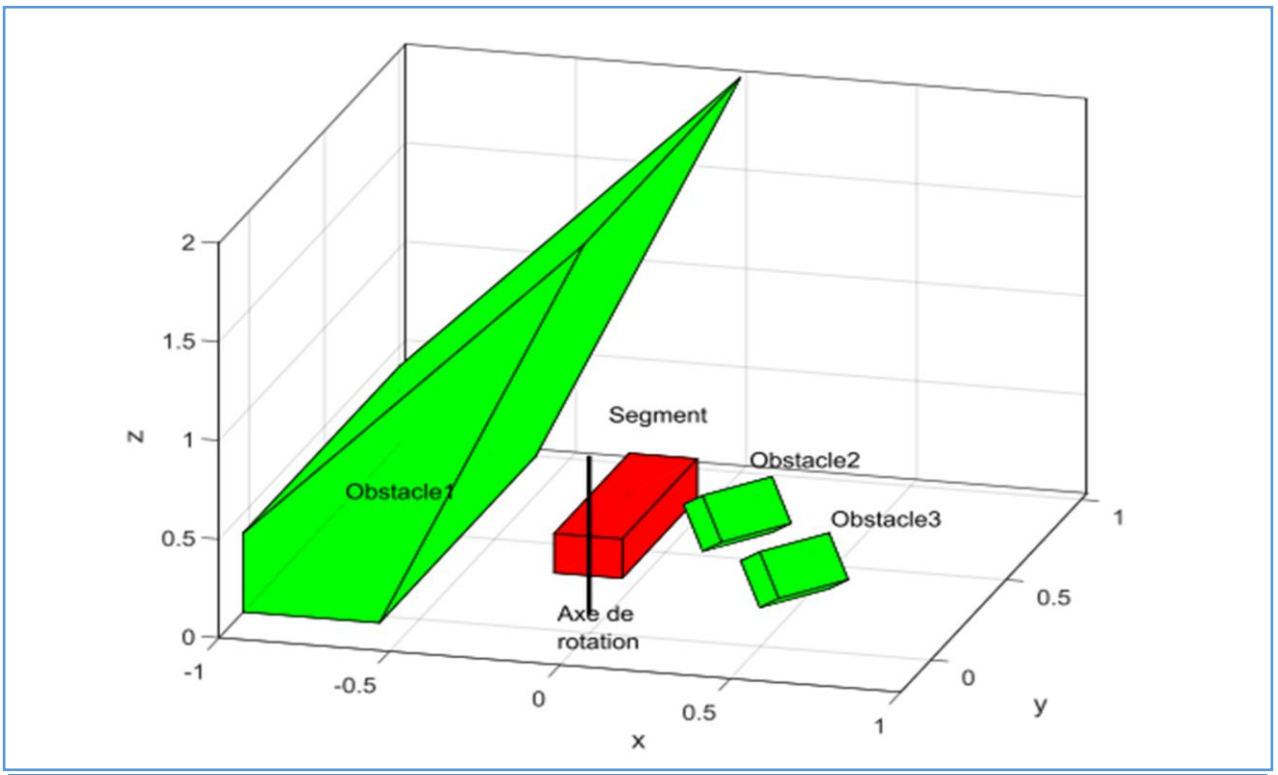
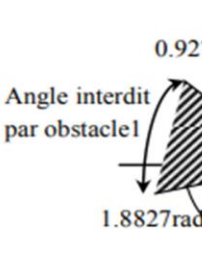


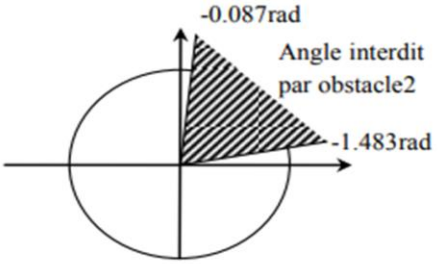
*Figure III-9: Contact arête-arête. Le premier n'est pas une applicabilité valide car l'un des vertex délimitant l'arête de l'obstacle est à l'intérieur du segment. Le deuxième cas représente une applicabilité valide (thet=-1.1876 rad). Les vecteurs (**e1** + **e2**) et (**e3** + **e4**) doivent pointer vers deux côtés différents du plan parallèle aux deux arêtes en contact.*





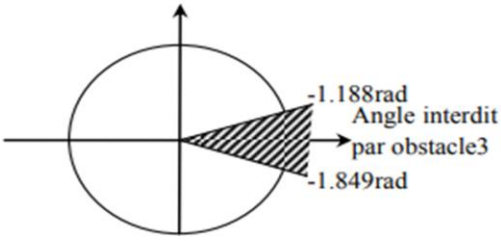
Les points de contacts calculés avec les signes de la dérivée de la distance de l'obstacle1 sont :

$$\begin{bmatrix} 0.9273 & 1.8827 \\ -1 & 1 \end{bmatrix}$$



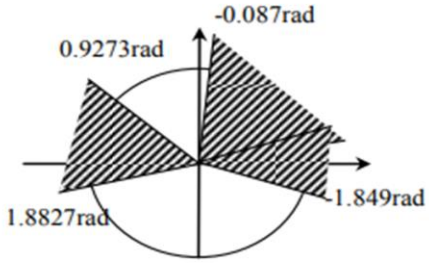
Les points de contacts calculés avec les signes de la dérivée de la distance de l'obstacle2 sont :

$$\begin{bmatrix} -1.483 & -0.087 \\ -1 & 1 \end{bmatrix}$$



Les points de contacts calculés avec les signes de la dérivée de la distance de l'obstacle2 sont :

$$\begin{bmatrix} -1.483 & -0.087 \\ -1 & 1 \end{bmatrix}$$



Combinaison de tous les points de contacts pour obtenir les gammes (intervalles, étendues) permises :

$$[-0.087, 0.9273], [1.8827, \pi], [-\pi, -1.849]$$

Figure III-10 Calcul des points de contacts et obtentions des gammes permises de l'angle articulaire.

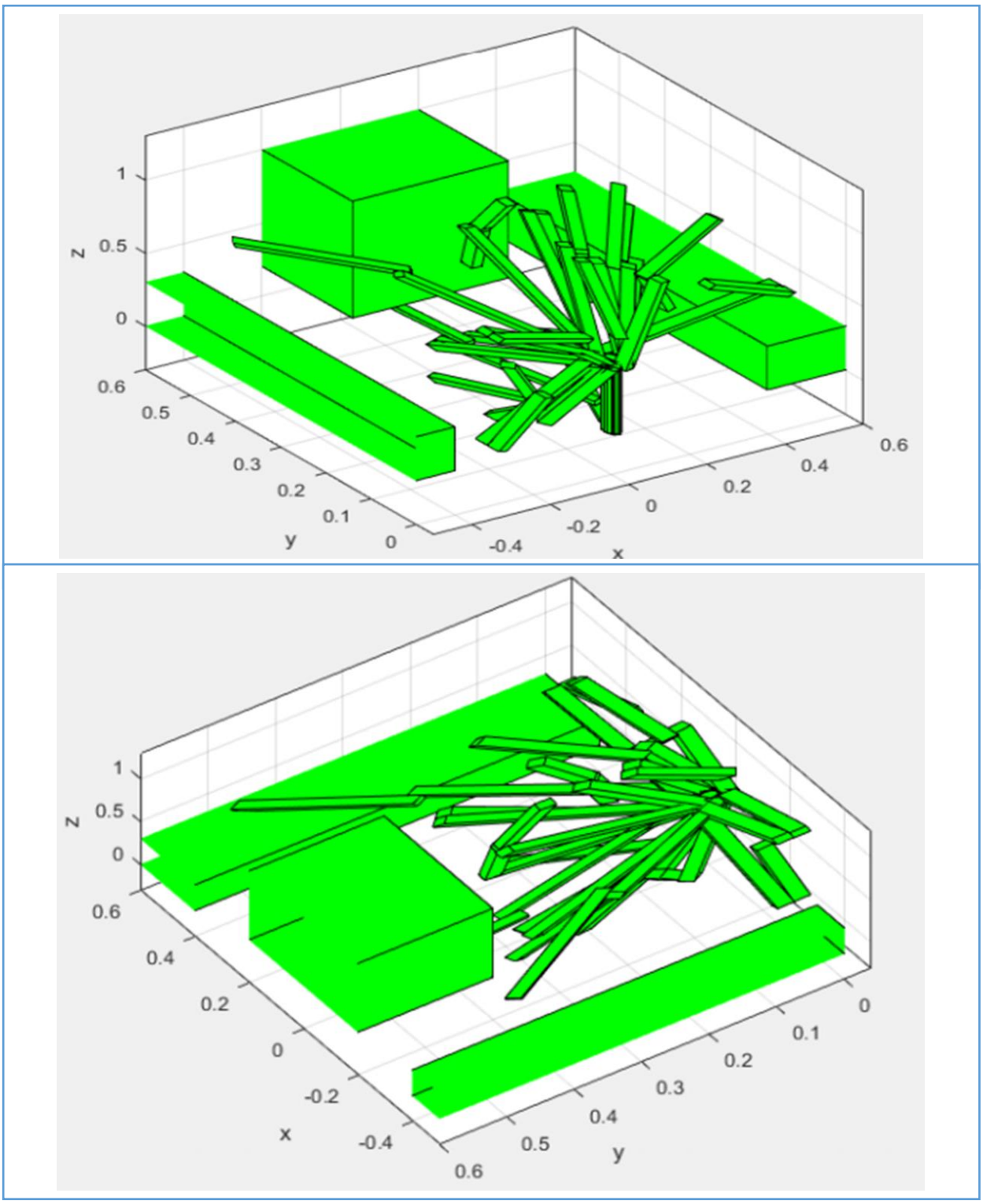


Figure III-11 Un ensemble de 20 configurations choisies aléatoirement parmi 692 configurations obtenues par la méthode de calcul des points de contacts et les intervalles interdits en utilisant un pas d'échantillonnage grossier de 0.5 rad

La figure suivante représente les cellules hors collision dans l'espace de configuration du robot à trois dlls. Le calcul a été fait en utilisant la fonction de détection des points de contacts entre les formes polyédriques (obstacles et segments du robot) pour générer un arbre représentant les configurations permises en utilisant un pas d'échantillonnage grossier $\Delta\theta = 0.5rad$. Les deux figures précédentes représentent les pauses du robot pour certaines de ces configurations. Ces deux figures confirment que le robot est effectivement hors collision pour toutes ces configurations.

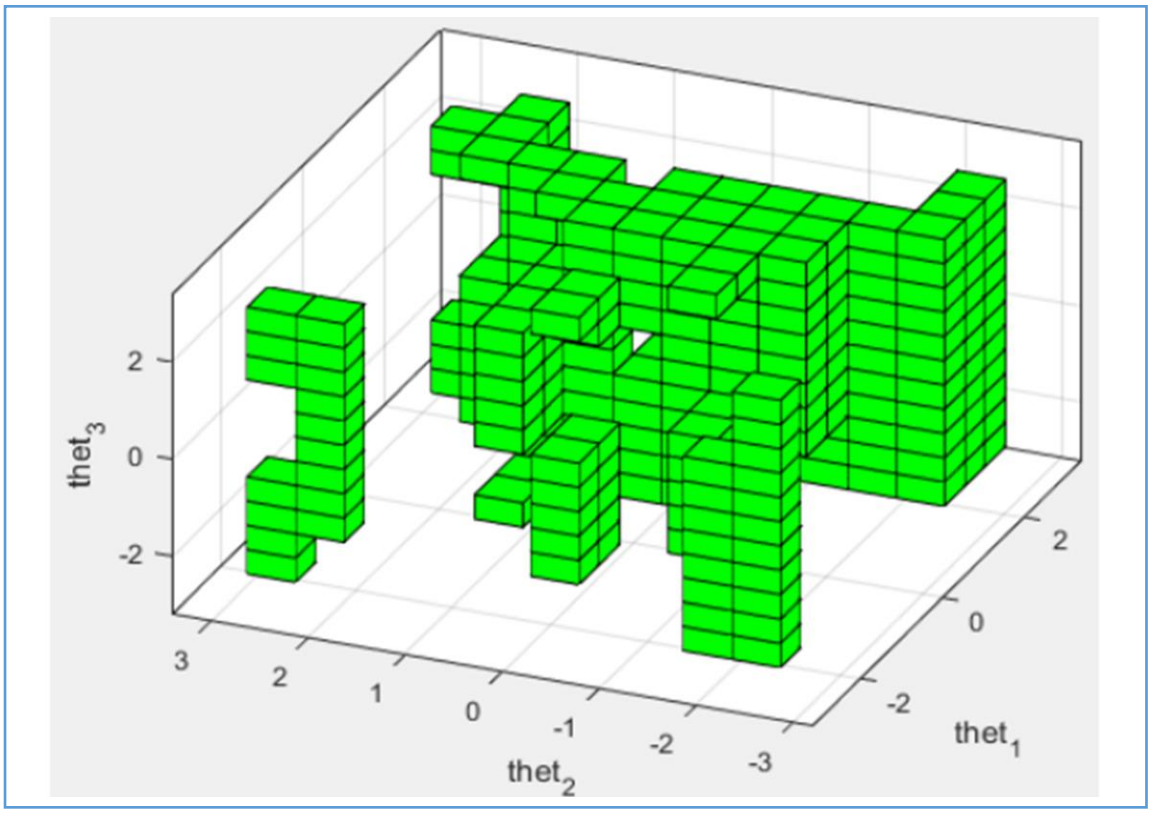


Figure III-12 Espace C-free de configuration hors collision (en vert) de la scène précédente et le robot 3ddls approximé selon $\Delta\theta = 0.5rad$.

Des techniques de planification de chemin existent utilisant la représentation dans l'espace des configurations des cellules hors collision [Faverjon]. Pour rechercher de tels chemins une représentation qui met en évidence la cohérence de l'espace hors collision doit être utilisée. Pour ceci nous avons choisi une méthode semblable à celle de [Lozano] dans laquelle l'espace hors collision est organisée en des régions construites autour de parties sous forme rectangulaire ou cubique appelées noyau de la région ainsi que toutes les cellules atteignables à partir du noyau en suivant une droite parallèle à l'un des trois axes de l'espace de configuration. Après avoir obtenu l'ensemble de ces régions, on cherche les connexions entre ces régions c'est à dire les chemins de passage d'une région vers les autres ceci est obtenu en cherchant les paires de cellules adjacentes qui appartiennent à deux régions différentes. A la fin de cette procédure nous obtenons un graphe de régions qui représente par ses nœuds les régions et par ses arcs les cellules de connexions entre ses régions. Dans MATLAB ce graphe est représenté par une cell-array de la forme représentée par la figure (III-12). Chaque ligne représente une région et elle contient trois matrices en plus du numéro de la région. La première matrice de dimension 3x2 représente les limites inférieure et supérieure du noyau de la région selon

les trois axes. La deuxième matrice donne une liste des cellules contenues dans région y compris celles du noyau. La troisième matrice donne la liste de connexion de la région avec les autres régions. q_{out} représente les trois coordonnées de la cellule de sortie de la région courante et q_{in} représente les trois coordonnées de la cellule d'entrée de la nouvelle région et enfin R_n donne le numéro de la nouvelle région.

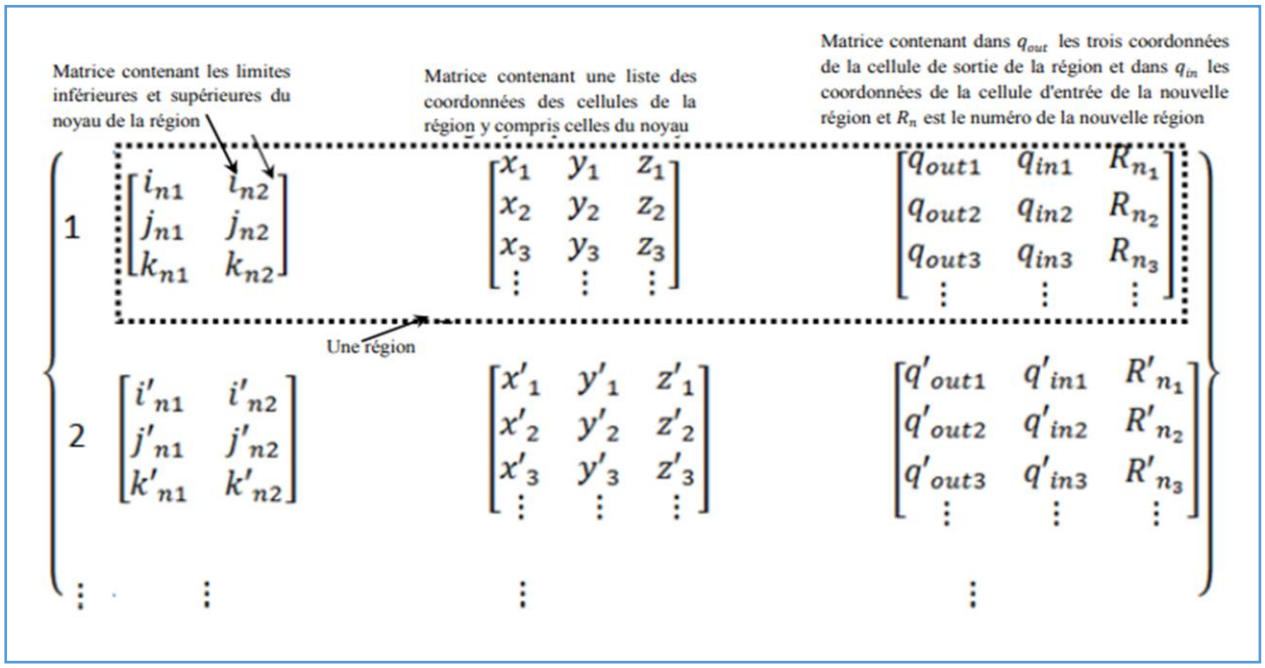


Figure III-13 Représentation du graphe de région

Pour obtenir cette représentation on procède de la façon de suivante :

Pour chaque cellule de l'espace des configurations :

1. si la cellule est hors collision et qu'elle n'est pas marquée il faut l'utiliser pour créer le noyau d'une nouvelle région et il faut la marquer comme cellule utilisée. Les limites inférieures et supérieures du noyau sont initialisées aux indices de la cellule.
2. Examiner sa voisine dans le sens de x positif si cette nouvelle cellule est hors collision et qu'elle n'est pas déjà marquée (n'appartient pas à une autre région) il faut la rajouter au noyau de la région courante et incrémenter la limite supérieure du noyau dans la direction de x et il faut marquer la cellule. Ensuite, examiner les deux cellules voisines dans le sens de y positif, si ces deux cellules sont hors collision et qu'elles ne sont pas marquées il faut les rajouter au noyau et il faut les marquer et incrémenter la limite supérieure du noyau dans le sens de y. Après, il faut examiner les quatre cellules voisines dans le sens de z positif et si elles ne sont pas marquées et qu'elles sont hors collision il faut les rajouter au noyau et les marquer et incrémenter la limite supérieure du noyau dans le sens de z.
   Il faut continuer cette extension dans le sens de x, y et z. Si à un moment donnée, on trouve qu'une cellule parmi les cellules d'extension dans une direction données est marquée

ou qu'elle est en collision il faut arrêter l'extension dans cette direction. L'extension s'arrête après avoir rencontrée.

3. Après avoir formé le noyau de la région, il faut rajouter les cellules accessibles à partir de ce noyau les marquer et les enregistrer dans la liste des cellules de la région.
4. Refaire les étapes 1, 2 et 3 pour une nouvelle cellule et continuer jusqu'à ce que l'ensemble de toutes les cellules soit marqué.

Après avoir formé les régions, il faut procéder à la recherche des connexions entre eux. Ceci est facilement accompli en examinant chaque paire de cellules voisines appartenant à deux régions différentes. A la fin de cette étape nous obtenons un graphe de régions qui sera à tout moment (tant que la scène reste inchangée) utilisé par le planificateur local pour trouver un chemin entre une configuration de départ et une configuration de destin

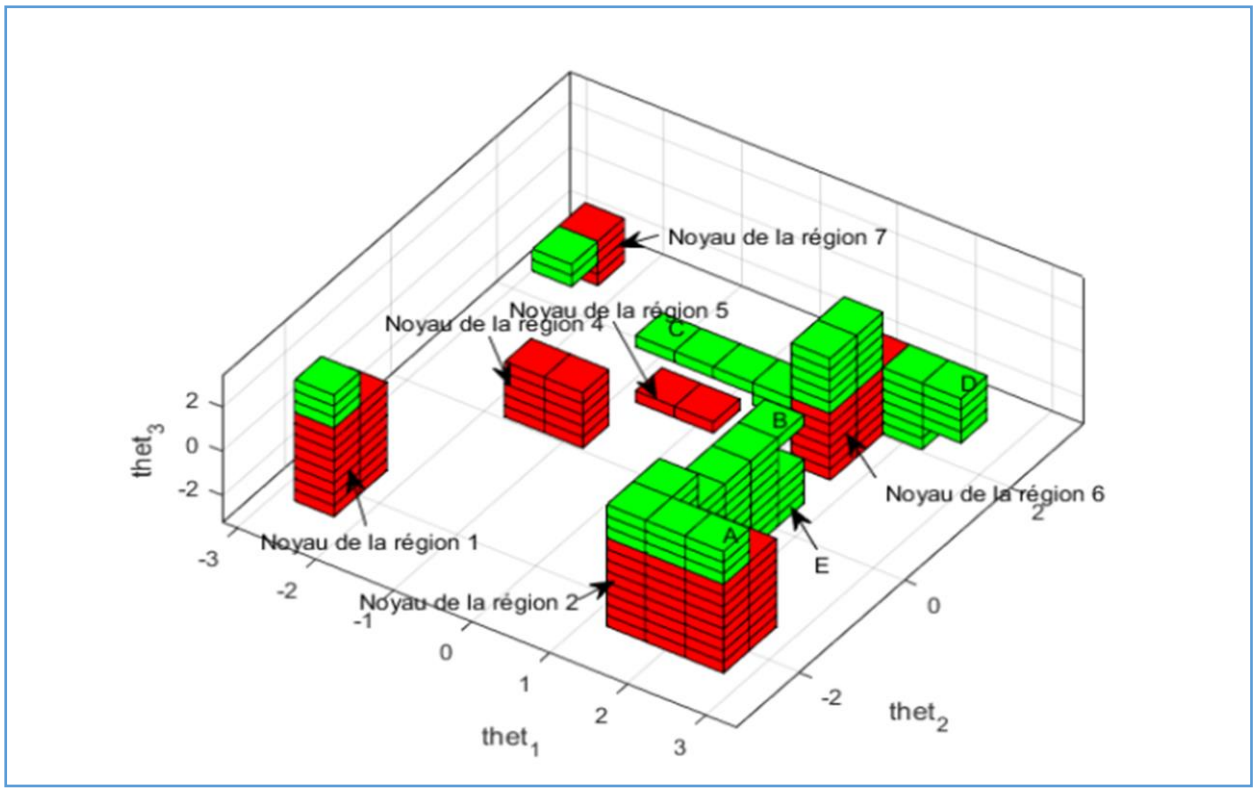


Figure III-14 Représentation des noyaux de certaines régions constituants le graphe de régions obtenue pour la scène précédente et le robot 3 ddls. Les cellules des noyaux en rouge et les autres cellules de régions en vert.

La figure précédente montre quelques régions parmi les 22 régions du graphe des régions obtenue pour la scène précédente et le robot 3dlls. Ici par exemple pour aller du point A au point B ou du point C au point D, il faut tout d'abord entrer dans le noyau de la région concernée partant du point A ou C en poursuivant une ligne droite. Ensuite, à l'intérieur du noyau, il faut poursuivre une autre ligne droite pour atteindre la cellule au frontière du noyau qui mène vers le point B (ou le point C) par une troisième ligne droite. Il est clair que cette façon de procéder quoique très simple n'est pas optimale, car pour aller par exemple du point B au point E on n'est pas obliger d'entrer dans le noyau ensuite aller vers E. Mais ce qui importe dans la formation d'une carte de route probabiliste est que le planificateur local doit être capable de décider sur l'existence ou pas d'un chemin entre deux configurations aussi rapidement que possible sans se préoccuper de la qualité de ce chemin. Des procédures de traitement postérieures doivent être prévues pour améliorer le chemin final.

La recherche par le planificateur local dans le graphe de régions pour un chemin entre une configuration initiale et une configuration de destination utilise l'algorithme A*. Ce dernier peut être résumé comme suit :

1. Créer un graphe G qui consiste seulement du nœud q_{start} appelé n_0. Mettre n_0 dans une liste appelée $OPEN$.
2. Créer une liste appelée $CLOSED$ qui est initialement vide.
3. Si $OPEN$ est vide, quitter avec échec.
4. Choisir le premier nœud de $OPEN$, l'enlever de $OPEN$ et le mettre dans $CLOSED$ est l'appeler n.
5. Si n est un nœud de destination, quitter avec succès avec la solution obtenue en traçant un chemin le long des nœuds de n à n_0 le long de G.
6. Étendre le nœud n, générant ainsi un ensemble M de ses descendants qui ne sont pas déjà des ancêtres de n dans G. Installer ces membres de M comme descendants de n dans G.
7. Etablir des pointeurs vers n de chacun des membres de M qui ne sont pas déjà dans G (c'est à dire qu'ils ne sont pas déjà dans $OPEN$ ou dans $CLOSED$). Rajouter ces membres of M à $OPEN$. Pour chaque membre m, de M qui est déjà dans $OPEN$ ou dans $CLOSED$ rediriger son pointeur vers n si le meilleur chemin à m trouvé jusqu'ici est à travers n. Pour chaque membre m de M qui est déjà dans $CLOSED$ rediriger les pointeurs de chacun de ses descendant dans G pour qu'ils pointent en arrière le long du meilleurs chemin trouver jusqu'à l'instant à ces descendants.
8. Réorganiser la liste $OPEN$ dans un ordre croissant de valeurs du coût $\hat{f}$. Pour les entrées de même valeur du coût, la priorité est donnée au nœud le plus profond dans l'arbre G.
9. Aller à l'étape 3.

La version que nous avons implémentée suit ces étapes sauf pour l'étape 7 pour la redirection des nœuds du graphe G car ce dernier ne nous intéresse pas.

Dans la figure qui suive, nous donnons une trajectoire obtenue en utilisant l'algorithme de recherche

A^*. Le chemin obtenu contient 26 cellules mais passe par 4 régions (noeuds) seulement {2,3,6,8}. Ce qui reflète la compacité de la représentation et comme conséquence l'augmentation de la vitesse de recherche. Dans le programme nous avons utilisé la notation décrite dans la figure. Pour chaque région r_i, le point q_b est le point qui détermine sa connexion avec la région précédente et pour la première région c'est tout simplement le point de départ q_s. Les points $\{point_1, point_2\}$ sont les points d'entrée et de sortie du noyau de la région. Les points $\{\ , q_{in}\}$ représentent les deux cellules adjacentes entre la région r_i et la région suivante et enfin le point $point_3$ est le point à l'entrée du noyau de la région suivante. La planification entre région se fait de la façon suivante : Les chemins entre le point q_b et $point_1$, ou entre $point_2$ et q_{out} ou bien entre q_{in} et $point_3$ sont toujours une droite parallèle à l'un des axes de coordonnées ce qui signifie qu'un seul angle doit varier. Le chemin entre $\{point_1, point_2\}$ (à l'intérieur du noyau) est un chemin de Manhatan

obtenue en variant seulement une coordonnée à la fois ($\theta_1\ ensuite\ \theta_2\ et\ enfin\ \theta_3$). Cette technique très simple de planification est aussi l'une des atouts qui améliore la vitesse de se planificateur. Pour les fonctions coûts nous les avons choisies comme suit : Le coût entre la région initiale r_s et la région courante r_i est une fonction $g(r_i)$ donnée par la somme des longueurs des portions droites du chemin plus la somme des normes euclidiennes des droites entre les points $\{point\_1, point_2\}$ à l'intérieure des noyaux. Le coût entre la région courante r_i et la région de destination r_g est donnée par la fonction f(r_i) qui est la distance entre les noyaux des deux régions qui est très facile à calculer car les noyaux sont des polyèdres rectangulaires dont les axes sont parallèles aux axes de coordonnées mais cette estimation est une borne inférieure du coût réel c'est à dire qu'elle ne pourra en aucun cas dépasser le coût réel. Ceci est une condition pour le bon fonctionnement de l'algorithme A^*

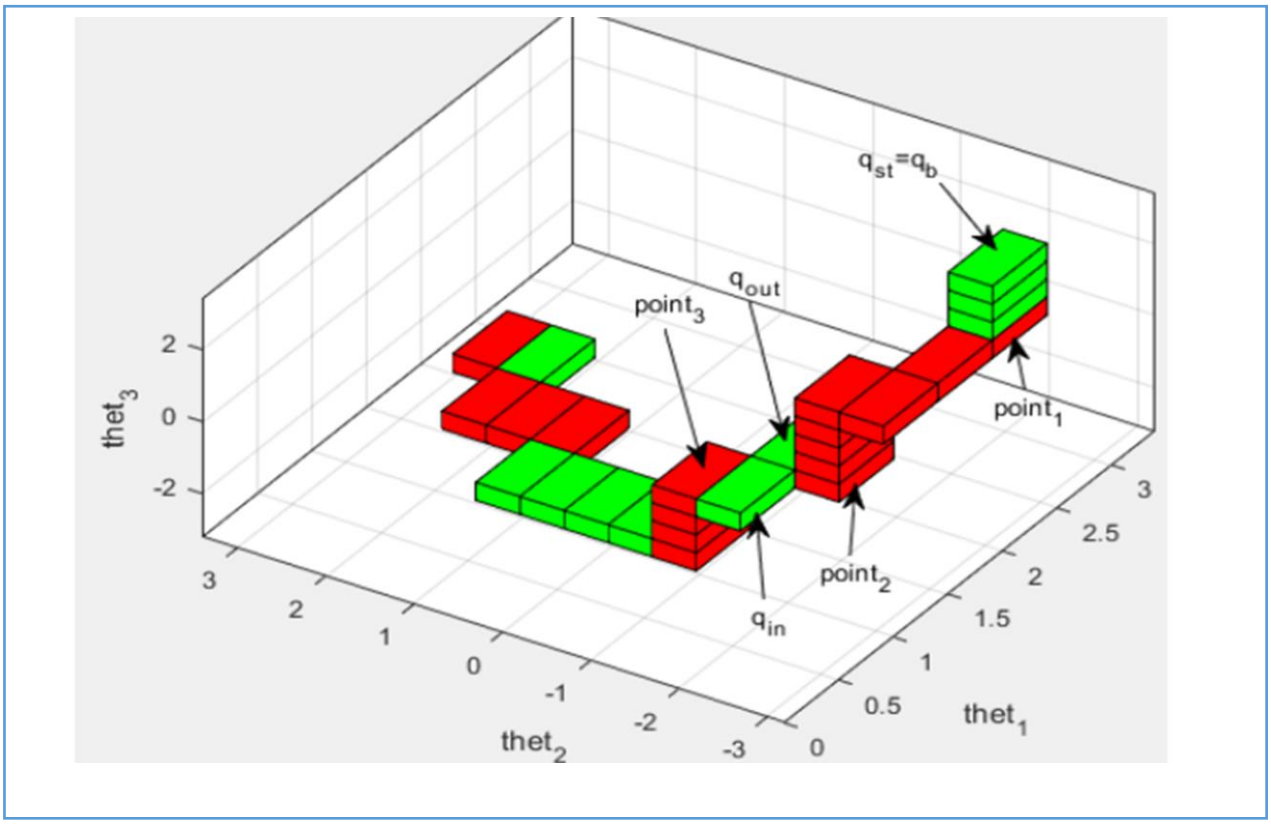
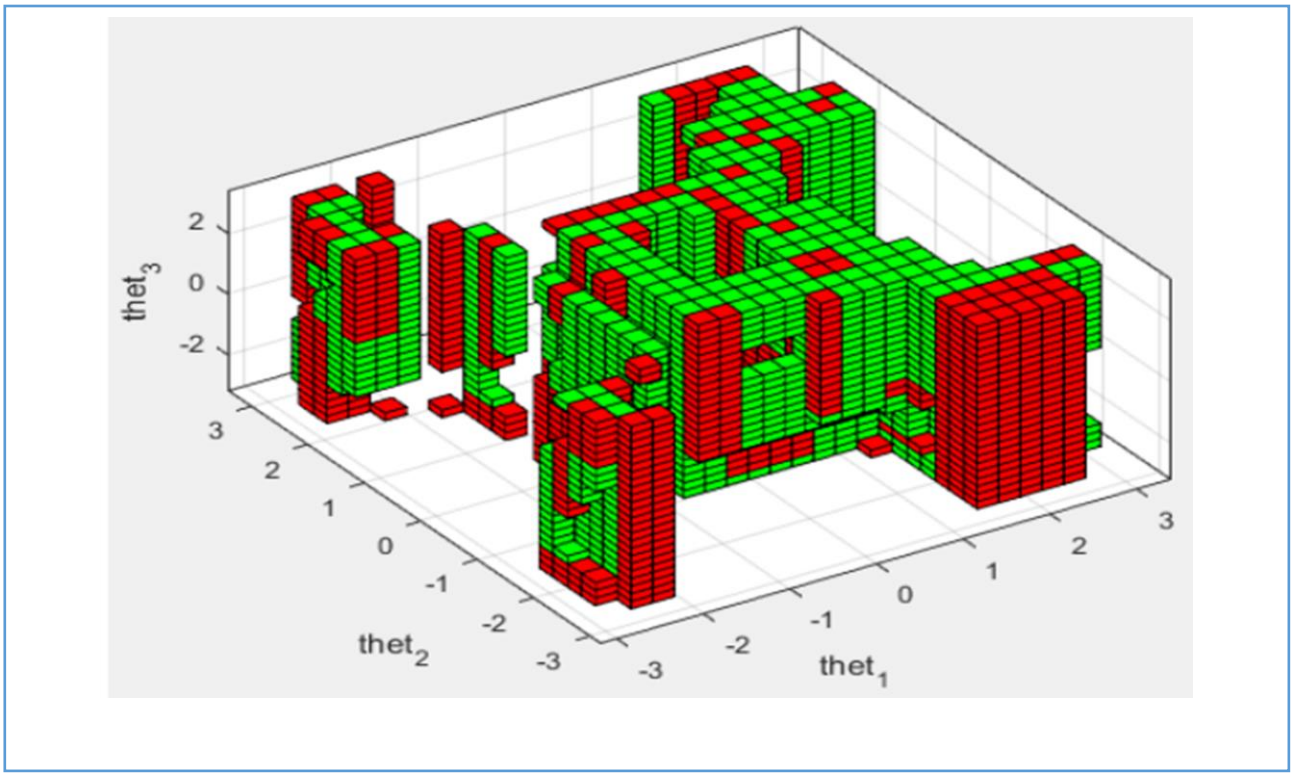


Figure III-15 Chemin dans l'espace des configurations pour le robot RRR et la scène précédente obtenue en utilisant le planificateur basé sur la représentation de l'espace de configuration par un graphe de région ensuite l'utilisation de l'algorithme A^* pour la recherche du chemin en terme de régions et les techniques simples pour la planification entre régions. La configuration initiale est $qs = 2.6084 -2.3916\ 2.3916$ et la configuration finale est $qg = 2.1084\ 2.6084 -2.3916$ en radian.

La figure suivante donne l'espace de configuration pour une résolution $\Delta\theta = 0.25rad$. Le graphe contient seulement 69 régions pour un espace de configurations de taille. La figure (III-15b) donne la trajectoire obtenue pour le même problème mais avec une résolution de $\Delta\theta = 0.05rad$. Pour ce cas l'espace de configurations contient 126x126x126 cellules mais le nombre de régions est seulement 637 ce qui montre la compacité de cette représentation. Le chemin représenté traverse 15 régions et contient 248 cellules.

.



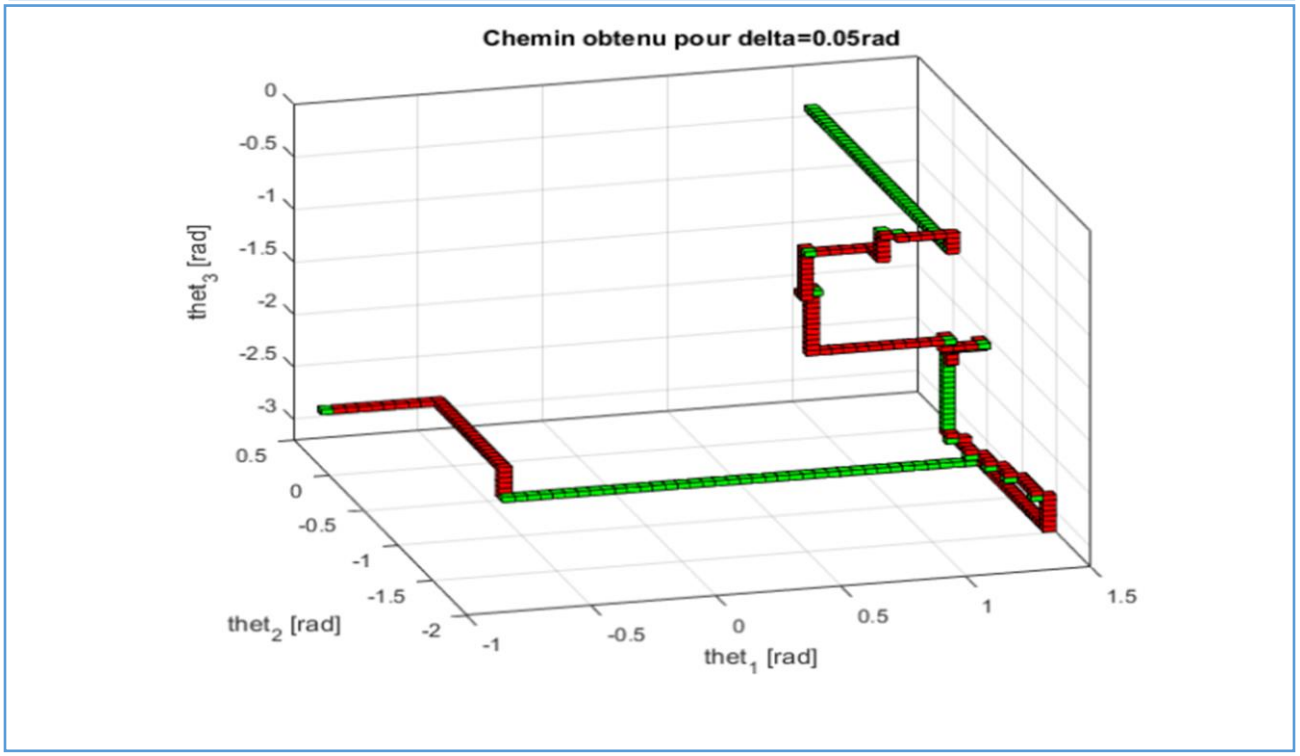


Figure III-16 a) Espace de configurations libres avec une résolution de $\Delta\theta = 0.25rad$. b) chemin obtenu pour $\Delta\theta$=0.05 rad.

La figure suivante montre une trajectoire dans l'espace de configurations accompagnée d'un ensemble de poses par lesquelles passe le robot le long de cette trajectoire. Il faut noter qu'en utilisant la résolution précédente, il n'était pas possible d'obtenir une solution et il a fallu passer à la résolution $\Delta\theta = 0.25rad$ pour avoir une bonne approximation de la connectivité de l'espace des configurations. C'est pour cette raison qu'on dit que la complétude de l'approche dépend de la résolution.

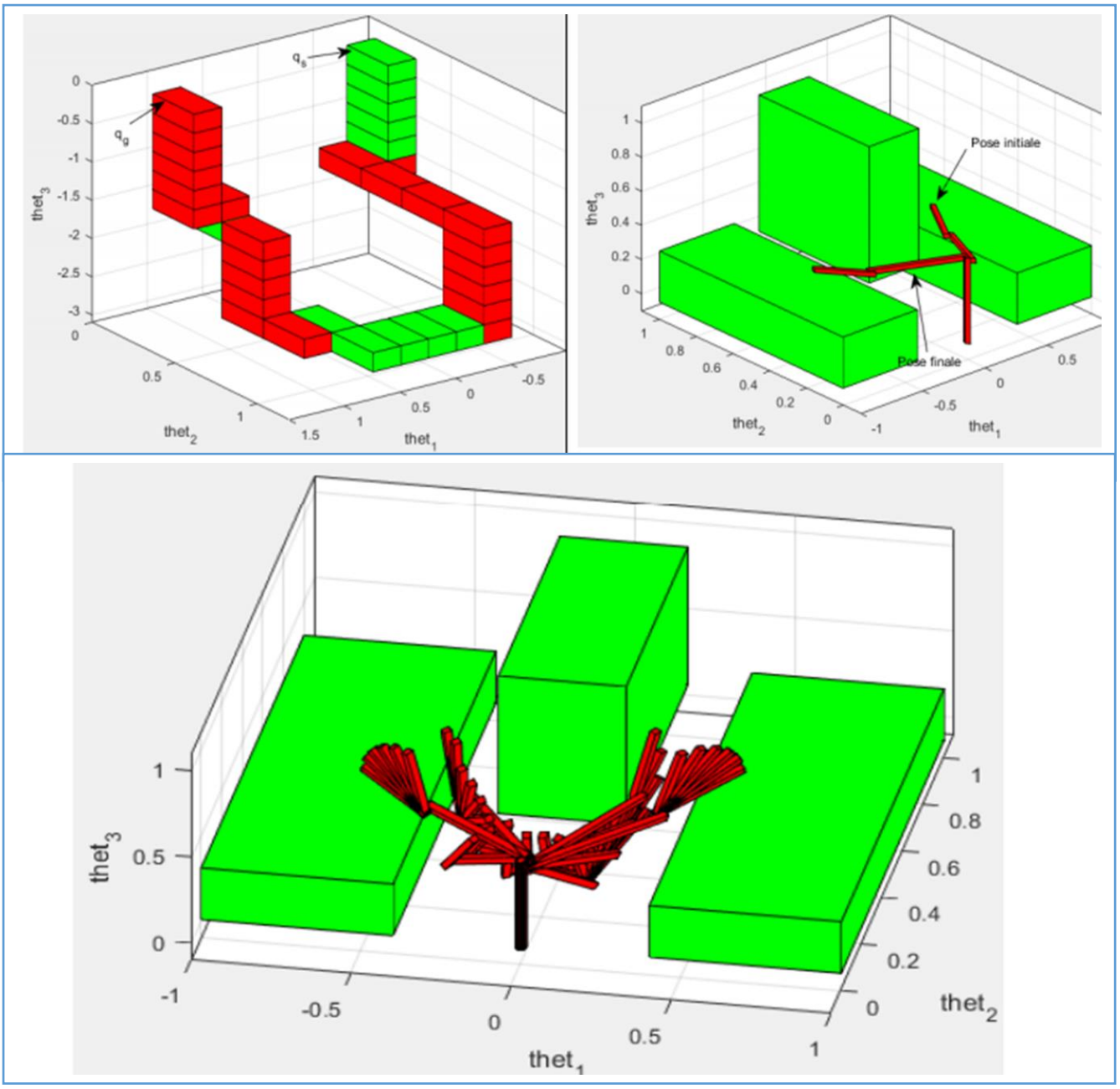


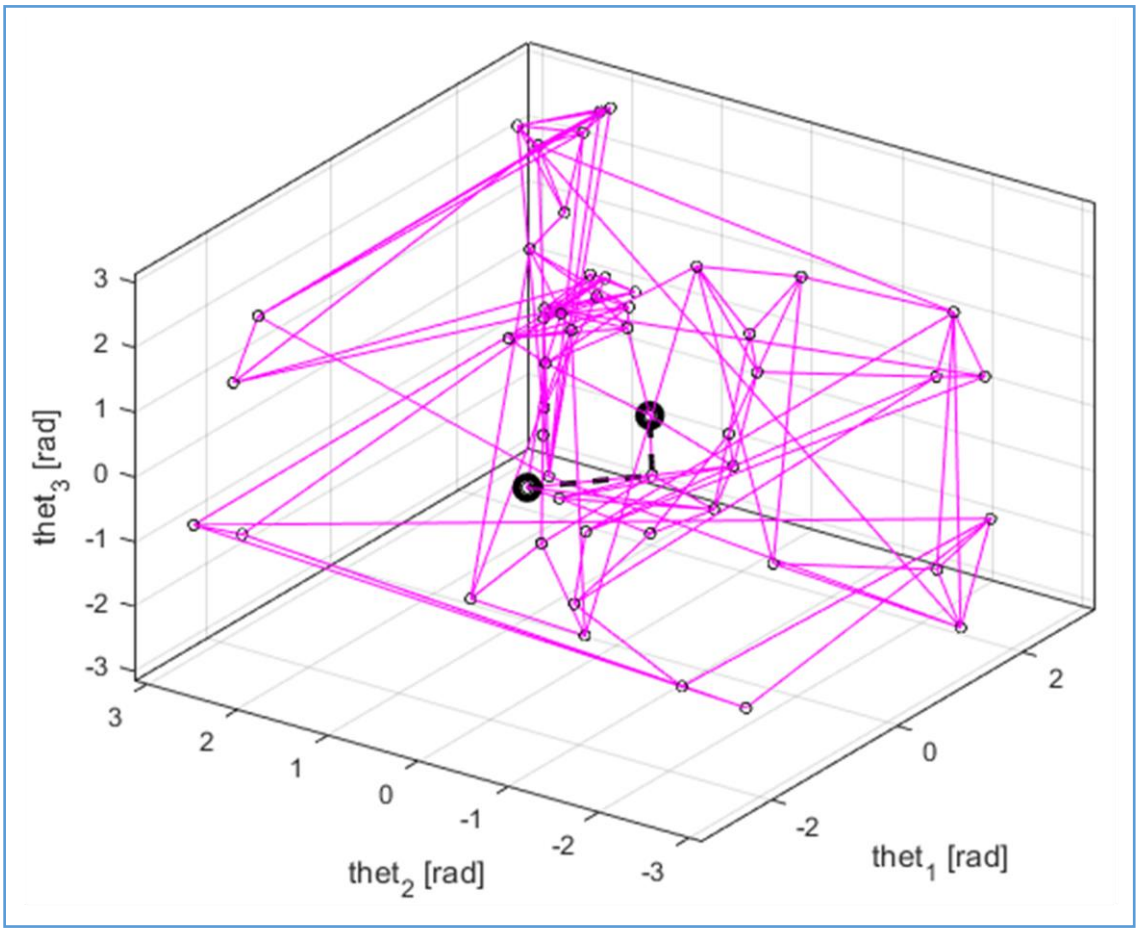
Figure III-17 Les poses du robot RRR durant la trajectoire obtenue par le planificateur précédent pour $\Delta\theta = 0.25rad$ *est les poses initiale et finale données* $\theta s = -0.7666\ 0.2334\ -0.2666$ *et* $\theta g = 0.9834\ 0.2334\ -0.2666$.

III.6 Phase de requêtes dans la PRM :

Après avoir formé la carte de route vient la phase de son utilisation pour résoudre des problèmes de planification dans la scène spécifiée. Etant données une configuration de départ θ_s et une configuration cible θ_g, on commence par les connecter à la PRM à travers deux de ces nœuds. Si cette connexion est réalisée avec succès, la méthode passe à la recherche d'une séquence de nœuds connectant les deux configurations. Si cette recherche réussie, la séquence trouvée est alors transformée à un chemin faisable pour le robot en recalculant les chemins locaux entre les différents nœuds de la séquence et en les compactant dans un seul chemin global.

Pour effectuer la recherche de chemin dans le graph qu'est la PRM, nous avons utilisé l'algorithme de profondeur d'abord (Depth-first) cette algorithme est simple à implémenter et permet d'obtenir toutes les solutions mais il peut devenir très long pour certains cas.

La figure suivante, donne une PRM obtenue pour le problème de planification pour le robot RRR ainsi qu'une solution parmi les 51 solutions retournées par la PRM. Ici nous avons fixé le nombre maximal de voisins à $k_{neighbors} = 4$, la distance maximale à $maxdist = 3 * \frac{\pi}{4}$ et le nombre maximal d'échantillons à 50. Le chemin contient 3 nœuds de la PRM mais 339 cellules de l'espace des configurations. La figure (III-17b) donne aussi le chemin réel après élagage dans l'espace de configurations. Vu que nous avons choisi le meilleur chemin retourné par la PRM, la version après élagage et celle avant élagage sont pratiquement les mêmes.



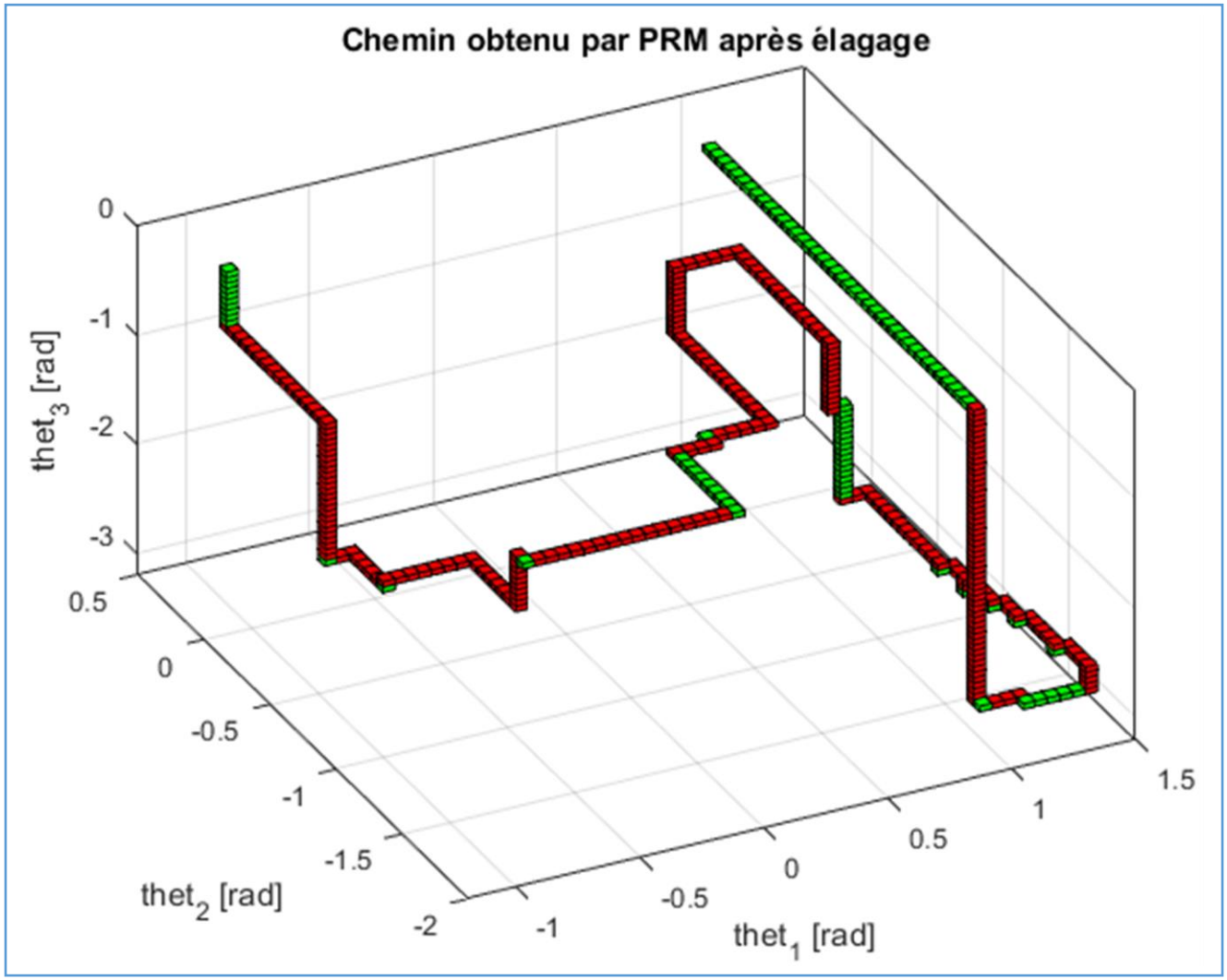


Figure III-18 Carte de route probabiliste dans l'espace de configurations, obtenue en utilisant la méthode de cycles pour l'échantillonnage ainsi qu'un chemin entre qs *et* qg *obtenues en utilisant la méthode de recherche en profondeur d'abord (Depth-first). Ici* $\Delta\theta = 0.05rad$

A titre d'illustration nous donnons dans la figure qui suit, la troisième solution retournée par la PRM. Ici, il faut noter que la version avant élagage contient 4005 cellules et la version après élagage contient 337 cellules qui montre la grande nécessité de l'élagage des chemins retournés par la PRM.

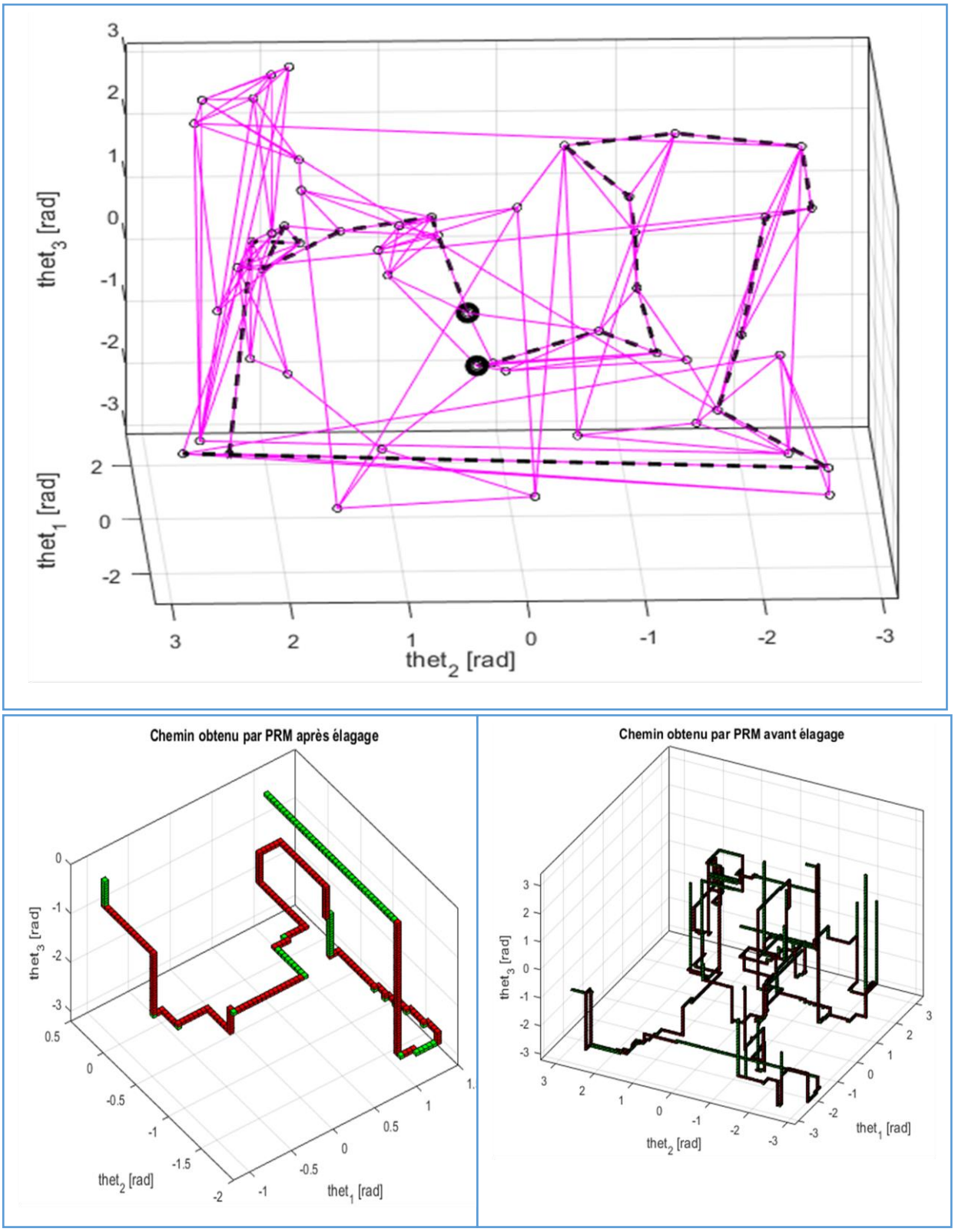


Figure III-19 a) PRM avec une chemin retourné entre $\theta set\ \theta g$. b et c) Chemins dans l'espace de configuration avant et après élagage pour $\Delta\theta = 0.05 rad$.

Nous donnons dans la figure qui suit un chemin avant et après élagage et nous donnons aussi les poses par lesquels passe le robot en parcourant le parcours. Nous constatons que d'un chemin retourné par la PRM de près de 340 cellules, nous obtenons un chemin de exactement 67 cellules seulement. On aurait pu faire mieux si on aurait considéré tous les 18 voisins d'une cellule.

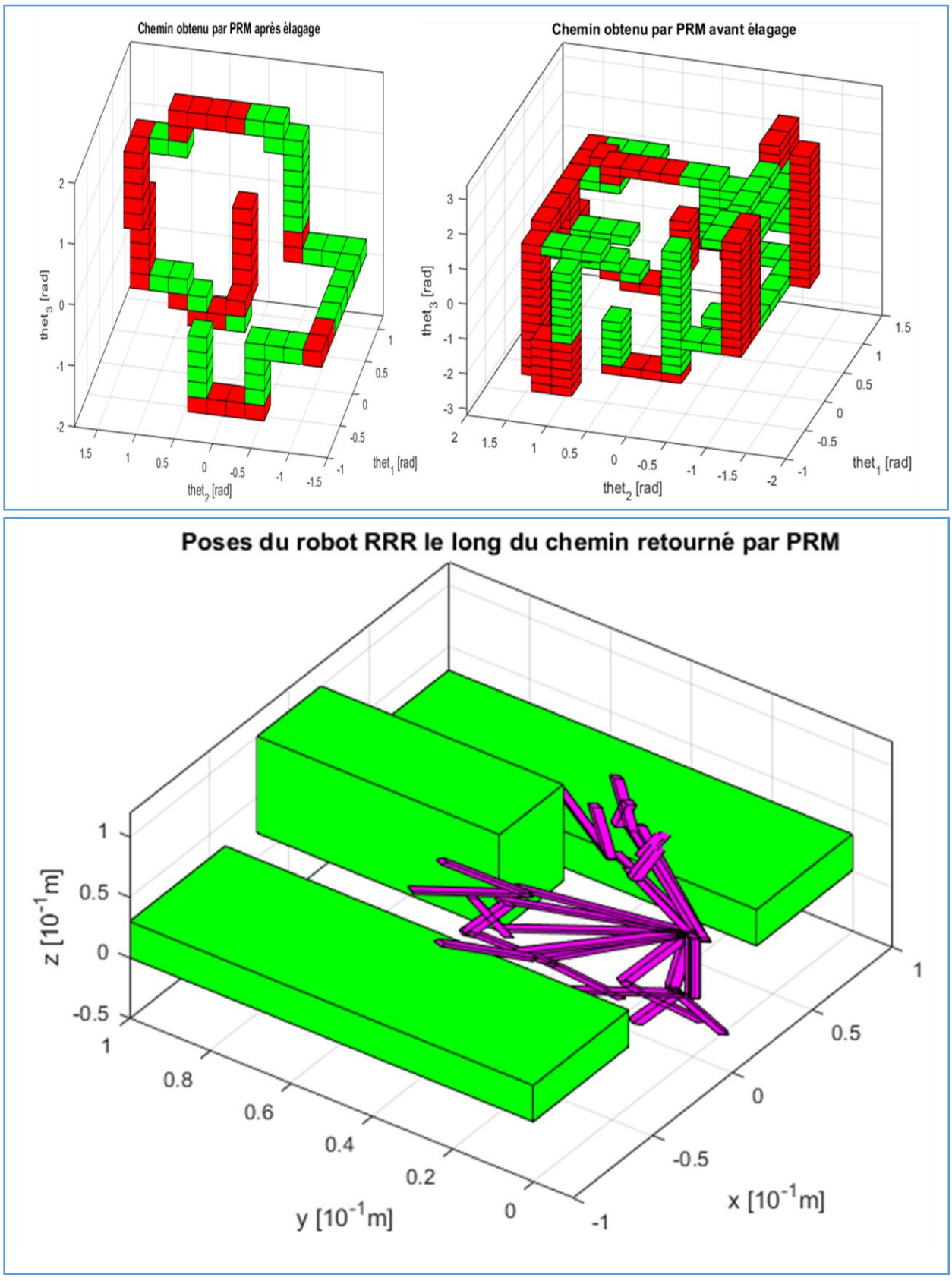


Figure III-20 Chemins dans l'espace de configuration obtenus par PRM en utilisant la méthode de cycle pour la connexion avant et après élagage et les poses traversées par le robot. $\Delta\theta = 0.25rad$

Conclusion :

Dans ce chapitre, nous avons présenté la méthode de planification de chemin utilisant une carte de route probabiliste pour un robot RRR dans une scène 3D. Nous avons expliqué les différentes étapes pour la construction de la carte et de son exploitation. Nous avons utilisé le l'échantillonnage uniforme pour le choix des configurations et la méthode des k-voisins pour la connexion combinée avec une métrique dédiée aux cas d'articulations rotoïdes.

La recherche des chemins locaux a été effectuée par un planificateur basé sur le calcul des intervalles légaux des coordonnées articulaires, leurs échantillonnages et la construction de l'espace de configurations libres dans l'espace articulaire. Ce dernier est représenté par un ensemble de régions connectées entre elles sous forme d'un graphe de régions pour faciliter la recherche.
Le planificateur local recherche la solution dans ce graphe de régions en utilisant l'algorithme A*. La complétude du planificateur local dépend de la résolution choisie.

La recherche dans le graphe de la PRM est effectuée en utilisant l'algorithme de recherche "profondeur d'abord" (Depth-first). Les solutions obtenues subissent un traitement postérieur appelé élagage pour améliorer leurs qualités.

CONCLUSION GENERAL

La ligne forte des travaux présentés dans ce manuscrit est de favoriser les méthodes probabilistes, en effet ces approches ont montré leur efficacité de traiter de manière générique des problèmes de planification de mouvement dans des espaces de dimension élevée et encombrés d'obstacles.

Nous avons présenté dans ce mémoire un planificateur de mouvement pour un manipulateur RRR à 3 dll basé sur la care de route probabiliste. Ce planificateur a permis de trouver un chemin pour notre robot de point initiale à un point final sans se heurter avec des obstacles, par ailleurs on a pu optimiser ce chemin en utilisant l'élagage.

Les résultats décrits dans ce mémoire ouvert la voie à plusieurs améliorations et extensions.

Les extensions futures de notre travail porteront, à notre sens, sur des planifications de tâche de manipulation pour des problèmes comportant plusieurs objets déplaçables. Une première étape dans ces investigations consistera à étudier le cas d'un seul objet déplaçable manipulé par un robot. Une autre extension consistera à développer un planificateur dans un espace changeant, où les obstacles sont dynamiques.

Bibliographie

[1] Laëtitia Matignon, Introduction à la robotique, GREYC-CNRS Université de Caen, 2011/2012.

[2] Jose Gutierrez Tapia, Modélisation et identification géométrique de robots utilisés pour des opérations d'usinage, Thèse de doctorat, Université Blaise Pascal - Clermont-Ferrand II, 2016.

[3] Wisama KHALIL, Etienne DOMBRE ;Bases de la modélisation et de la commande des robots-manipulateurs de type série, Université Numérique Ingénierie & Technologie, mai 2012.

[4] BENZATER Habiba, Contrôle de la trajectoire du bras manipulateur PUMA560 par les algorithmes évolutionnaires multi-objectifs NSGA-II et NSBBO-II, Université de la science et de technologie d'oran mohamed boudiaf, 2015.

[5] Zatout Arraoun, Planification et contrôle de trajectoires pour un bras de robot par une PRM, Mémoire de master, Universités A.Mira Bejaia,2018

[6] M.Rantanen, Improving Probabilistic Roadmap Methods for Fast Motion Planning, Copyright 2014 Tampere University Press and the author,2014.

[7] B.Mandil, Introduction Générale- Bras Manipulateurs, Universités A.Mira Bejaia, 2018

[8] Didier Müller, Introduction à la théorie des graphes, COMMISSION ROMANDE DE MATHÉMATIQUE, CRM, Décembre 2011

[9] Fabrice Colin, APPLICATIONS DE LA TOPOLOGIE ALGÉBRIQUE EN THÉORIE DES GRAPHES,mémoire vue de l'obtention du grade de maître ès sciences, Université de SHERBROOKE Canada, 1996

[10] http://www.apprendre-en-ligne.net/graphes/dijkstra/algorithme.html ,consulté le 10/07/2020

[11] https://khayyam.developpez.com/articles/algo/astar/ ,consulté le 18/07/2020

[12] M. Akinc, K. E. Bekris, B. Chen, A. Ladd, E. Plaku et L. E. Kavraki. Probabiliste feuilles de route d'arbres pour le calcul parallèle de plusieurs feuilles de route de requête. In International Symposium on Robotics Research, 2003. Livre à paraître.

[13] R. Geraerts et M. Overmars. Une étude comparative des planificateurs de feuilles de route probabilistes. Dans J.-D. Boissonnat, J. Burdick, K. Goldberg et S. Hutchinson, éditeurs, Algorithmic Foundations of Robotics V, pages 43–58. Springer-Verlag, 2003.

[14] J. J. Kuffner. Échantillonnage efficace et mesures de distance pour la planification de trajectoire de corps rigide 3D. Dans IEEE International Conference on Robotics and Automation, 2004.

[15] F. Schwarzer, M. Saha et J. C. Latombe. Vérification exacte des collisions des chemins du robot Dans J.-D. Boissonnat, J. Burdick, K. Goldberg et S. Hutchinson, éditeurs, Algorithmic Foundations of Robotics V, pages 25–42. Springer-Verlag, 2002.

[16] N. M. Amato, B. Bayazit, L. Dale, C. Jones et D. Vallejo. OBPRM: un PRM basé sur les obstacles pour les espaces de travail 3D. Dans P. Agarwal, L. E. Kavraki et M. Mason, éditeurs, Robotics: The Algorithmic Perspective, pages 156–168. AK Peters, 1998.

[17] V. Boor, N. H. Overmars et A. F. van der Stappen. La stratégie d'échantillonnage gaussien pour les planificateurs de feuilles de route probabilistes. Dans IEEE International

Conference on Robotics and Automation, pages 1018–1023, 1999.

[18] D. Hsu, L. E. Kavraki, J. C. Latombe, R. Motwani et S. Sorkin. Sur la recherche de passages étroits avec des planificateurs de feuille de route probabilistes. Dans e. une. P. Agarwal, éditeur, Robotics: The Algorithmic Perspective, pages 141 154. A.K. Peters, Wellesley, MA, 1998.

[19] M. Foskey, M. Garber, M. Lin et D. Manocha. Un planificateur de mouvement hybride à base de voronoï. Dans IEEE / RSJ International Conference on Intelligent Robots and Systems, 2001.

[20] C. Holleman et L. E. Kavraki. Un cadre pour l'utilisation de l'axe médian de l'espace de travail dans les planificateurs PRM. Dans IEEE International Conference on Robotics and Automation, pages 1408–1413, 2000.

[21] C. Pisula, K. Hoff, M. Lin et D. Manocha. Planification de chemin aléatoire pour un corps rigide basé sur un échantillonnage Voronoi accéléré par le matériel. Dans B. R. Donald, K. Lynch et D. Rus, éditeurs, New Directions in Algorithmic and Computational Robotics. AK Peters, 2001.

[22] P. Leven et S. Hutchinson. Utilisation de la manipulabilité pour biaiser l'échantillonnage lors de la construction de feuilles de route probabilistes. Transactions IEEE sur la robotique et l'automatisation, 19 (6): 1020–1026, décembre 2003.

[23] R. Bohlin. Planification de chemin en pratique: évaluation paresseuse sur une grille multi-résolution. Dans IEEE / RSJ International Conference on Intelligent Robots and Systems, 2001.

[24] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones et D. Vallejo. Choisir de bonnes métriques de distance et des planificateurs locaux pour des méthodes de feuille de route probabilistes. Dans IEEE International Conference on Robotics and Automation, pages 630–637, 1998.

[25] D. Nieuwenhuizen et M. H. Overmars. Cycles utiles dans les graphiques de feuille de route probabilistes. Dans IEEE International Conference on Robotics and Automation, pages 446–452, 2004.

[26] J. Barraquand et J. C. Latombe. Planification de mouvement de robot: une approche de représentation distribuée. International Journal of Robotics Research, 10 (6): 628–649, décembre 1991.

[27] R. Bohlin et L. E. Kavraki. Un algorithme aléatoire pour la planification du chemin du robot basé sur une évaluation paresseuse. Dans P. Pardalos, S. Rajasekaran et J. Rolim, éditeurs, Handbook on Randomized Computing, pages 221–249. Kluwer Academic Publishers, 2001.

Résumé

L'objectif de ce mémoire est de présenter la méthode de planification de chemin en utilisant une carte de route probabiliste pour un robot manipulateur RRR dans une scène 3D. La démarche adoptée pour l'atteinte de l'objectif est la suivante : nous avons commencé par la construction de la PRM qui passe par différentes étapes; au début on échantillonne un ensemble de configurations hors collision en utilisant une distribution uniforme. Ensuite, on utilise la méthode des k-voisins pour la sélection de paires de configurations de la PRM candidates pour la connexion. Le planificateur local que nous avons utilisé est basé sur la recherche de points de contacts entre les segments du robot et les obstacles qui sont supposés de formes polyédriques, nous avons considéré seulement le cas d'articulations rotoïdes mais l'approche peut être étendue aux cas prismatiques. Un arbre représentant les intervalles hors collision des coordonnées articulaires est ensuite établi et est utilisé pour obtenir un graphe de régions dans l'espace articulaire. La recherche de chemin dans ce graphe de régions est effectuée en utilisant l'algorithme Best-First. La carte de route probabiliste ainsi construite est utilisée pour répondre à des requêtes de planification. Ceci est effectué en utilisant l'algorithme de recherche "profondeur d'abord" (Depth-first). Le chemin ainsi obtenu subit un élagage (traitement postérieur) pour améliorer sa qualité. La simulation est réalisée sur MATLAB.

Mots-clés : carte de route probabiliste, méthode d'échantillonnage, planificateur locale, configuration voisine.

Abstract

The objective of this thesis is to present a path planning method using Probabilistic Road Map for an RRR robot in 3D scene. The approach adopted is as follows: The construction phase of the PRM is done in several steps. At the beginning, a set of configurations out of collision is sampled using a uniform distribution. Then, the selection of adjacent pairs of configurations that are candidates for connection is done using the k-neighbors method. The local planifier that we used in this work is based on the search of contact-points between robot links and obstacles which are both supposed polyhydric. We considered only the case of rotoïdes joints, but the approach could be extended to the case of prismatic joints as well. A tree representing the joint-coordinates intervals out of collision is then established and used to obtain a region graph in the configuration space. Path search in this region graph is done using the Best-First algorithm. The PRM so constructed is used to answer planification queries. This is done using the Depth-First algorithm. The final path is passed through a pruning function to improve its quality. Simulations are done using MATLAB.

Keywords: probability road map, sampling method, local planner, neighboring configuration.