République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et Recherche Scientifique
Université Saad Dahlab de Blida 1

C.D.T.A

Faculté des sciences
Département d'Informatique

En vue d'obtenir le diplôme de master
Domaine : Mathématique et informatique
Filière : Informatique
Spécialité : Informatique
Option : Ingénierie de logiciel

Planification de trajectoires sans collision pour un robot manipulateur basée sur des caméras Kinect

Présenté par :
- Allane Hamza
- Aid Houssam Eddine

Encadrant :
-M. HENTOUT Abdelfetah
Promoteur:
-M. CHIKHI Nacim Fateh

Soutenu le :
Devant le jury :

-M. FERFERA Sofiane	Président
-M. HAMOUDA Mohamed	Examinateur
-M. HENTOUT Abdelfetah	Encadrant
-M. CHIKHI Nacim Fateh	Promoteur

Année universitaire : 2017/2018

REMERCIEMENTS

JE REMERCIE DIEU DE NOUS AVOIR ACCORDÉ DES CONNAISSANCES DE LA SCIENCE ET DE NOUS AVOIR AIDÉS À RÉALISER CE MODESTE TRAVAIL. AU TERME DE CE MODESTE TRAVAIL JE TIENS À REMERCIER CHALEUREUSEMENT ET RESPECTIVEMENT TOUS CEUX QUI 'ONT CONTRIBUÉ DE PRÈS OU DE LOIN À LA RÉALISATION DE CE MODESTE PROJET DE FIN D'ÉTUDE, À SAVOIR NOS ENCADREURS. MES VIFS REMERCIEMENTS VONT À M. HENTOUT ABDELFETAH ET M. CHIKHI NACIM FATEH. MES VIFS REMERCIEMENTS VONT ÉGALEMENT AUX MEMBRES DU JURY POUR L'INTÉRÊT QU'ILS ONT PORTÉ À NOTRE RECHERCHE EN ACCEPTANT D'EXAMINER NOTRE TRAVAIL ET DE L'ENRICHIR PAR LEURS PROPOSITIONS. JE TIENS À REMERCIER TOUS LES ENSEIGNANTS QUI NOUS ONT SUIVIS DURANT NOTRE FORMATION POUR LE TRAVAIL ACCOMPLIE ET LEURS VALEUREUX CONSEILS. CE TRAVAIL FUT DIFFICILE MAIS TRÈS BÉNÉFIQUE À TOUT POINT DE VUE.

MERCI À TOUS.

Dédicace

A Nos pères, Aucune dédicace ne saurait être assez éloquente pour exprimer ce que tu mérites pour tous les sacrifices que tu n'as cessé de me donner depuis ma naissance, durant mon enfance et même à l'âge adulte.

A Nos mères, qui a toujours était à mes côtés , a toi la plus belle créature que Dieu a créée sur terre À cette source de tendresse, de patience et de générosité .

A Nos frères et nos sœurs, Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours eu pour vous.

A toute ma grande famille oncles et Tantes, Cousin et Cousine A tous mes ascendant Allah yarhamhom.

A tous mes amis.

A tous collègues de travail.

A tous les étudiants de la promotion.

A MiZoU.

A RaDoU.

Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fruit de votre soutien infaillible A tous ceux qui m'aiment.

Merci d'être toujours la pour moi.

ملخص

إن الروبوت هو آلية يتم التحكم فيها إلكترونيًا ومتعددة الأجزاء تؤدي المهام عن طريق التفاعل مع بيئتها. وتستخدم الروبوت على نطاق واسع في قطاع الصناعات التحويلية الصناعية والعديد من التطبيقات المتخصصة الأخرى. الهدف من عمل الماستير هذا هو العثور على تخطيط مسار للروبوت ، حيث يتم استخدام كاميرا Kinect للكشف عن مواقع الروبوت بالإضافة إلى العقبات. سوف يستغل هذا الروبوت المتحرك البيانات التي تسلمها كاميرا Kinect. أيضا، سيتم استغلال النهج القائم على منطق غامض، تم تطويره مسبقا، من أجل توليد المسارات وتجنب الاصطدام مع العقبات.

الكلمات المفتاحية : مناور الروبوتات. تخطيط مسار. تجنب الاصطدامات. المنطق الضبابي. الكاميرا Kinect

Résumé

Un robot manipulateur est un mécanisme à commande électronique, constitué de plusieurs segments, qui effectue des tâches en interagissant avec son environnement. Les manipulateurs sont largement utilisés dans le secteur de la fabrication industrielle et aussi beaucoup d'autres applications spécialisées. L'objectif de ce travail de Master est de générer des trajectoires sans collision pour un robot manipulateur en exploitant les données délivrées par une caméra Kinect afin de calculer les positions actuelles du robot et celles des obstacles. Aussi, une approche basée sur la logique floue, développée au préalable, sera exploitée pour la génération de trajectoires et l'évitement de collision avec les obstacles.

Mots-Clés : Robots manipulateurs, Planification de trajectoires, Évitement de collisions, Logique floue, Caméra Kinect.

Abstract

A manipulator robot is an electronically controlled, multi-segment mechanism that performs tasks by interacting with its environment. Manipulators are widely used in the industrial manufacturing sector and also many other specialized applications. The objective of this Master's work is to find a collision-free trajectory for a manipulator robot connecting an initial position to a target position. For this aim, the manipulator will exploit the data delivered by a Kinect camera to calculate the current robot positions as well as the obstacles. Also, an approach based on fuzzy-logic, previously developed, will be exploited for the generation of the trajectories and the avoidance of collision with obstacles.

Keywords: Manipulator robots, Trajectory planning, Collision avoidance, Fuzzy logic, Kinect camera.

Sommaire

Liste des figures

Liste des tableaux

Introduction générale

Un robot manipulateur est un système robotique articulé doté souvent d'effecteurs. Ce système est caractérisé par la longueur des segments le composant et la nature des articulations régissant son mouvement.

Dans de nombreux domaines, le robot manipulateur est devenu l'instrument de travail indispensable à l'exécution de tâches pénibles, répétitives ou qui exigent une grande précision. L'entreprise manufacturière est l'un des secteurs les plus fortement concernés. Les manipulateurs sont de plus en plus présents dans les ateliers. Ils sont capables de remplacer l'homme lors des interventions dangereuses ou inaccessibles ; par exemple en milieu hostile, sous-marin, nucléaire ou spatial.

Malgré les progrès technologiques, qui font de ce type de robots un outil de plus en plus sophistiqué, cette machine est rarement autonome. Son utilisation nécessite, pour réaliser une tâche donnée, de prescrire une trajectoire au robot manipulateur le menant de la configuration initiale ($Source$) à la configuration finale imposée ($Target$).

Cependant, une trajectoire donnée n'est pas toujours réalisable du fait d'éventuelles contraintes. Certaines de ces contraintes sont propres au robot manipulateur. Il peut s'agir par exemple des limites sur les débattements et sur les capacités des différents actionneurs engendrant le mouvement. D'autres contraintes sont imposées par la nature même de la tâche à réaliser (vitesses et accélérations restreintes par une charge sensible ou massive transportée, etc.). Une autre catégorie de contraintes est relative à l'environnement dans lequel évolue le robot. En effet, l'espace de travail du manipulateur est rarement dégage. L'encombrement de la scène peut alors interdire certains mouvements qui engendreraient éventuellement une collision du manipulateur avec l'un des objets qui l'entourent. Par conséquent, le calcul de la trajectoire doit alors inclure l'évitement de ces obstacles afin que le mouvement planifié ne soit pas collisionnel.

Le problème de planification de trajectoires et d'évitement d'obstacles peut être défini comme suit :« *Connaissons la configuration initiale du robot manipulateur dans un environnement connu ou inconnu qui peut contenir des obstacles statiques ou dynamiques, et connaissons l'ensemble des informations nécessaires sur les caractéristiques de cet environnement collectées par le système sensoriel du robot sous une forme spécifiée, alors*

comment peut-on procéder si nous voulons déplacer ce robot manipulateur d'une configuration initiale (Source) vers une configuration cible(Target) en évitant les obstacles existants ? ».

Le problème de planification de trajectoires pour les robots manipulateurs dans des environnements encombrés est un problème central en robotique. Il a fait l'objet de plusieurs recherches durant ces dernières années et a suscité de nombreux travaux durant ces dernières années, conduisant à un nombre très important d'approches. Elles sont généralement divisées en trois classes : les méthodes globales, les méthodes locales et les méthodes mixtes.

Aujourd'hui, il existe plusieurs méthodes d'intelligence artificielle. Ces méthodes sont développées pour atteindre un objectif commun : rendre le robot capable de résoudre des problèmes d'une manière intelligente. Ces méthodes d'intelligence artificielle sont considérées parmi les meilleures techniques qui permettent de résoudre différents types de problèmes.

La logique floue est largement appliquée dans différents domaines de la robotique, notamment pour la planification de trajectoires, l'évitement d'obstacles, la commande en vitesse, la commande en position, etc. Dans le domaine de planification, la logique floue est souvent employée pour les robots mobiles comme méthode alternative aux techniques conventionnelles de planification basées sur les modèles mathématiques. Cependant, la génération de mouvements pour les manipulateurs basée sur cette technique est plus difficile à cause de la structure mécanique complexe de ce type de robots (plusieurs articulations liées entre elles qui doivent se déplacer). Par conséquent, très peu de travaux ont été réalisés dans ce domaine par rapport à ceux effectués sur les robots mobiles.

L'objectif de ce travail consiste en la planification de trajectoires sans collisions pour un robot manipulateur dans le cas de tâches point à point. La trajectoire sans collisions générée doit permettre au robot de se rendre de sa position initiale $Source(x_E, y_E, z_E)_{Init}$ à sa position finale imposée $Target(x_E, y_E, z_E)_{Fin}$ en présence d'obstacles.

Dans le but de modéliser l'environnement d'évolution du robot, détecter les obstacles qui s'y trouvent et reconnaître leurs positions, ce robot manipulateur va exploiter les données délivrées par une caméra Kinect. Aussi, une approche basée sur la logique floue, développée au préalable, sera exploitée pour la génération de trajectoires et l'évitement de collision avec les obstacles présents dans l'espace de travail.

En outre de cette introduction, ce mémoire comporte quatre chapitres et une conclusion générale ; il est organisé comme suit :

- Le premier chapitre est consacré aux définitions de robots, leurs différents types (mobile et manipulateur) ainsi que les types de robots manipulateurs.
- Le deuxième chapitre consiste en une étude bibliographique non exhaustive des méthodes les plus importantes dans le domaine de planification de trajectoires sans collisions pour les robots, à savoir, les méthodes globales, les méthodes locales et les méthodes mixtes.
- Le troisième chapitre est consacré à la présentation de la méthode basée sur logique floue et de la définition de l'approche de contrôle utilisée.
- Le dernier chapitre présente l'implémentation et la validation de l'approche proposée.

Chapitre I
Introduction à la robotique

I.1. Introduction

Les robots sont partout, ils sont dans les usines, dans les champs, au fond des mers et dans l'espace, dans les jardins et les salons. Ils ont une importance économique grandissante. Ils n'ont pas seulement pénétré le monde industriel, ils sont aussi entrain de pénétrer notre vie quotidienne et notre culture.

Dans ce chapitre, nous allons donner quelques définitions sur les robots et leurs deux différentes catégories. Nous allons aussi donner des définitions selon le critère de mobilité ainsi que les types de robots. Enfin, nous allons présenter les robots manipulateurs et ses différents constituants.

I.2. Définition d'un robot

Le robot est un système automatique dont la partie opérative est une structure mécanique articulée (voir Figure1).

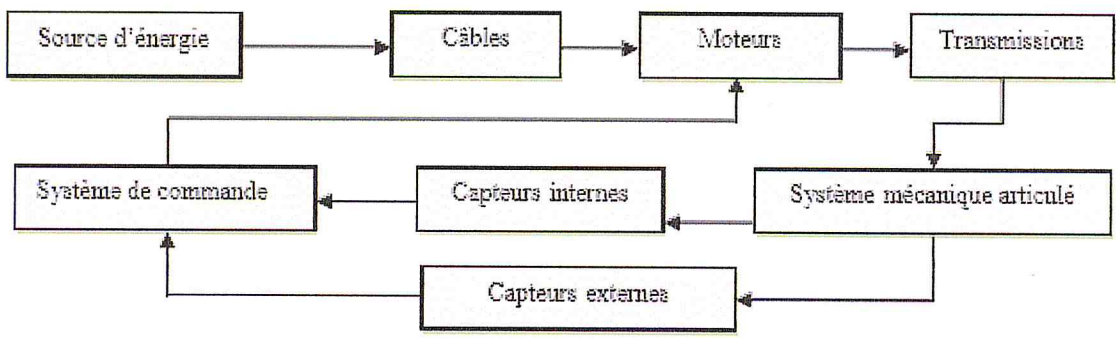


Figure 1 : Parties commande-contrôle et opérative du robot [3]

La définition que l'on donne actuellement au robot diffère quelque peu selon les pays. Selon le *Robot Institute of America* [1] « *un robot est un manipulateur reprogrammable à*

fonctions multiples. Il est conçu pour déplacer des matériaux, des pièces, des outils ou des instruments spécialisés suivant des trajectoires variables programmées, en vue d'accomplir des tâches très diverses ».Selon l'*Association Japonaise de Robotique Industrielle* [1], « *un robot est tout mécanisme permettant d'effectuer, en tout ou en partie, une tâche normalement réalisée par l'homme* ». L'*Association Française de Robotique Industrielle* (*AFRI*)[1]définit le robot comme étant « *une machine formée de divers mécanismes comportant divers degrés de liberté (ddl), ayant souvent l'apparence d'un ou de plusieurs bras se terminant par un poignet capable de maintenir un outil, une pièce ou un instrument de contrôle. En particulier, son unité de contrôle doit contenir un système de mémorisation, et il peut parfois utiliser des accessoires sensitifs et adaptables qui tiennent compte de l'environnement et des circonstances. Ces machines, ayant un rôle pluridisciplinaire, sont généralement conçues pour effectuer des fonctions répétitives, mais sont adaptables à d'autres fonctions* ». Enfin, *International Standard Organization* (*ISO*) [2] définit le robot comme « *une machine formée par un mécanisme incluant plusieurs ddl, ayant souvent l'apparence d'un ou plusieurs bras se terminant par un poignet capable de tenir des outils, des pièces ou un dispositif d'inspection* ».

I.3. Types de robots

On peut dire qu'il y a deux catégories de robots selon le critère de mobilité :*les robots manipulateurs* et *les robots mobiles* [4].

I.3.1. Robots manipulateurs

Ces robots sont conçus pour réaliser une tâche unique et indéfiniment répétée telle que l'assemblage des pièces dans une chaîne de production, le montage de voitures. Ce type n'est pas mobile ; aussi, le robot ne saura accomplir que la tâche pour laquelle il a été conçu.

La définition donnée aux robots manipulateurs industriels par la *Fédération International de la Robotique* sous le standard *ISO/TR 8373* est la suivante : « *Un robot industriel manipulateur est un contrôle automatique, reprogrammable ; c'est un manipulateur multifonction programmable en trois axes ou plus, qui peut être soit fixé dans un endroit, soit mobile et il est destiné à des applications d'automatisation industrielles* ». Ces robots étaient conçus essentiellement pour répondre aux besoins de l'industrie ; ils sont intelligents (parce

qu'ils sont programmables) et capables d'assurer un mouvement en trois dimensions. Les raisons principales d'utilisation de ces robots sont comme suit :

- Environnement dangereux qui présente beaucoup de risque pour la vie de l'homme.
- Réalisation d'un travail répétitif et cyclique.
- Réalisation d'un travail difficile à l'homme ou qui nécessite une haute précision (nanotechnologie, etc.).
- Ils peuvent fonctionner sans arrêt24h/24.
- Ils nécessitent un entretien minimal.
- Ils sont plus rapides que l'être humain et exécutent le travail en continu, d'une manière précise et constante.

Seulement, les robots manipulateurs ne sont pas encore prêts à remplacer l'humain parce que de nombreux obstacles techniques et sécuritaires se dressent encore. Par conséquent, la présence de l'être humain est indispensable pour assurer le contrôle et la maintenance [4].

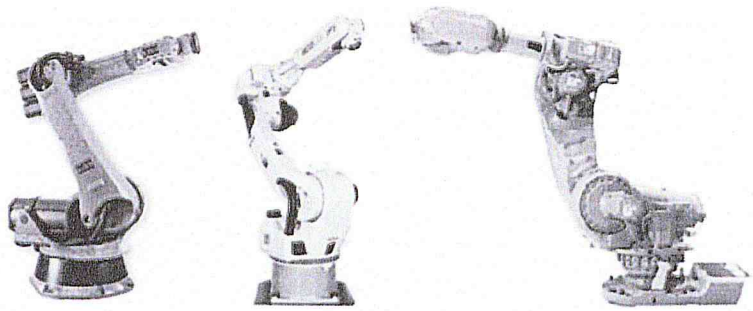


Figure 2 : Exemples de robots manipulateurs [4]

I.3.2. Robots mobiles

Ces robots, donnés par figure 3, exécutent des tâches qui nécessitent un déplacement dans un environnement. Ils sont en effet le plus souvent désignés par leur type de locomotion, qu'ils soient marcheurs, sous-marins ou aériens. Contrairement aux robots manipulateurs prévus pour travailler exclusivement dans des espaces connus et de manière répétitive, les robots mobiles sont destinés à évoluer de manière autonome dans des environnements peu ou non structurés. Néanmoins, le concepteur doit augmenter considérablement ses connaissances sur la localisation

et la navigation de systèmes autonomes et prendre en considération l'environnement d'évolution lors de la conception de ces robots. On peut citer les robots explorateurs comme exemple de ce type [4].

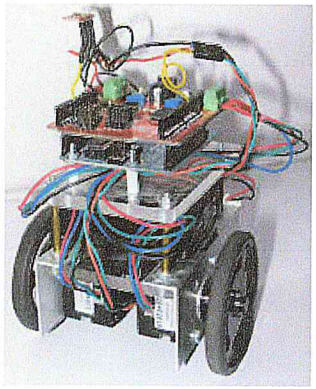


Figure 3 : Exemple d'un robot mobile autonome [4]

Dans la suite de ce travail, notre étude se limite aux robots manipulateurs ; les robots mobiles ne sont pas considérés.

I.4. Bras manipulateurs

Un bras manipulateur est généralement programmable avec des fonctions similaires à un bras humain. Les liens de ce manipulateur sont reliés par des axes permettant, soit du mouvement de rotation ou de translation (linéaire).Il peut être autonome ou contrôlé manuellement. Ce type de robots peut être aussi utilisé pour effectuer une variété de tâches avec une grande précision [5].Selon la figure 4, un robot manipulateur est composé de :

- (1) actionneur ou moteur,
- (2) axe ou articulation,
- (3) corps ou segment,
- (4) organe terminal,
- (5) effecteur ou outil, et enfin
- (6) la base.

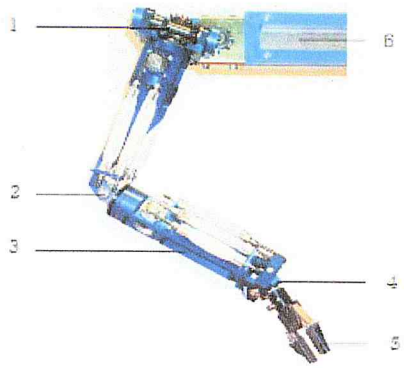


Figure 4 : Composants d'un bras manipulateurs [6]

I.5.Différent types de robots manipulateurs

Différents types de robots manipulateurs existent ; ils sont décrits brièvement dans ce qui suit [6] :

I.5.1. Robots SCARA (Selective Compliance Articulated)

C'est un robot à 3 axes série (RRT) ayant 3 ddl et un espace de travail cylindrique (Figure 5). C'est un robot à la fois précis et rapide ; il est utilisé dans les tâches d'assemblage.

Figure 5 : Robot SCARA

I.5.2. Robots cylindriques

Ce type de robots a 3 axes ayant 3 ddl (série RTT). Il est caractérisé par une rapidité et un espace de travail cylindrique (Figure 6).

Figure 6 : Robot cylindrique

I.5.3.Robots sphériques

Ce sont des robots à 3 axes (série RRT) ayant 3 ddl. Leurs espace de travail est sphérique (figure 7). Ils sont aussi caractérisés par leur grande charge utile.

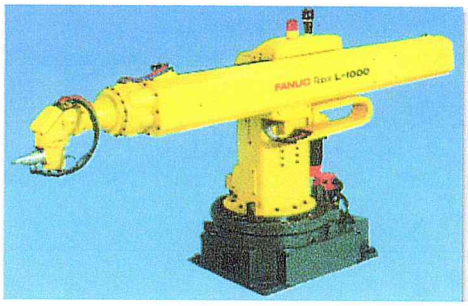


Figure 7 : Robot sphérique

I.5.4.Robots cartésiens

Ce sont des robots à 3 axes perpendiculaires 2 à 2, série, TTT, ayant 3 ddl (Figure 8). Ils sont caractérisés par leur bonne précision ; mais, ils sont lents.

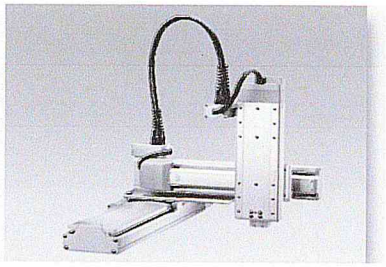


Figure 8 : Robot cartésien

I.5.5.Robots parallèles

Ce sont des robots à plusieurs chaînes cinématiques en parallèles (figure 9). Ces robots sont caractérisés par un espace de travail réduit, par leur rapidité et leur précision.

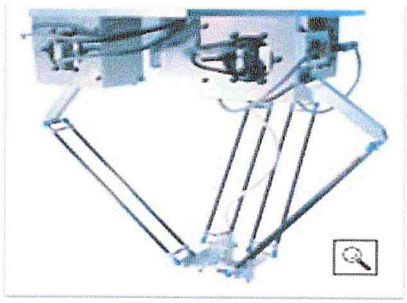


Figure 9 : Robots anthropomorphes

I.6. Conclusion

Ce chapitre a fourni une brève introduction aux mots-clés et aux différents paradigmes utilisés dans le cadre de ce travail.

Grâce à la robotique, l'être humain a réussi à créer des composants électroniques assez minimes, ce qui a contribué à la mise en œuvre des téléphones portables, et des ordinateurs portables. La rapidité et la précision des robots permettent de produire une grande quantité de divers produits avec une qualité meilleure, chose que l'être humain ne peut pas réaliser dans le même laps du temps.

D'après l'étude effectuée, nous pouvons dire qu'un robot est un système combiné de hardware et de software capable de collecter, traiter les données et effectuer des tâches intelligentes dans un environnement physique.

Le chapitre suivant est dédié à la description des principales classes des approches de planification de trajectoires sans collision, à savoir, les approches globales, les approches locales et les approches mixtes.

Chapitre II
Méthodes de planification de trajectoires sans collision

II.1. Introduction

Le problème de planification de trajectoires a suscité de nombreux travaux durant ces dernières années, conduisant à un nombre très important d'approches. Dans ce chapitre, nous résumons les approches de planification les plus significatives. Elles sont généralement divisées en trois classes : les méthodes globales, les méthodes locales et les méthodes mixtes [7].

Les approches globales ont besoin d'une connaissance complète de l'environnement. Ces méthodes cherchent toutes les trajectoires possibles. Ensuite, elles sélectionnent l'une d'entre elles selon un ensemble de critères. Ces approches assurent l'existence ou non d'une solution.

Les approches locales n'ont besoin que d'une connaissance partielle de l'espace de travail. Elles cherchent une trajectoire au fur et à mesure que le robot évolue dans l'espace de travail. Toutefois, même si une trajectoire sans collision existe, les méthodes locales peuvent aboutir à une situation de blocage.

Les approches mixtes sont une combinaison des deux approches précédentes. Elles opèrent en deux phases. La première phase est similaire aux approches globales. Elle permet de générer une trajectoire de référence réalisée hors ligne. La seconde phase est similaire aux approches locales. Elle adapte localement la trajectoire du robot à partir de la trajectoire de référence.

Dans ce qui suit, nous allons décrire chacune de ces trois classes en donnant leurs avantages et leurs inconvénients respectifs.

II.2. Planification de trajectoires

Le planificateur est le constructeur d'un plan ; il peut être un humain ou une machine. Si le planificateur est une machine, l'algorithme de planification est connu généralement comme le

planificateur. Une fois un plan est déterminé, on peut l'utiliser selon trois manières différentes [4] :

- **Exécution** : exécuter le plan soit en simulation soit sur un robot réel,
- **Raffinement** : construire un plan meilleur,
- **Inclusion hiérarchique** : intégrer le plan comme étant une action dans un algorithme d'un niveau plus haut.

II.2.1.Principaux ingrédients d'une planification de trajectoires

Les principaux ingrédients d'une planification de trajectoires sont donnés comme suit [4] :

- **État** : un problème de planification doit capturer toutes les situations possibles. En général, un état représente la position et l'orientation d'un robot à un moment précis,
- **Espace d'état** : l'ensemble de tous les états est appelé espace d'état,
- **Temps** : tous les problèmes de planification définissent une séquence de décisions qui doit être appliquée avec le temps. Le temps doit être modélisé explicitement,
- **Action** : un plan génère des actions qui manipulent les états,
- **État initial et état final** : un problème de planification implique un démarrage d'un état initial vers un état final spécifié, appelé objectif, cible, etc.,
- **Critères** : Généralement, deux types de critères peuvent être ajoutés pour la planification :
  - **Un critère de faisabilité** : trouver un plan qui cause une arrivée à un état objectif sans prendre en considération son efficacité,
  - **Critère d'optimalité** : trouver un plan faisable (menant à l'état objectif) qui optimise les performances d'une manière spécifique (temps d'exécution, longueur, etc.).
- **Plan** : un plan est une séquence d'actions qui s'exécute dans le temps et qui conduit d'un état initial à un état final en respectant un ou plusieurs critères. En général, un plan impose une stratégie, un comportement ou un décideur.

II.3. Méthodes locales

Les méthodes locales (ou réactives) sont utilisées lorsque nous ne disposons pas d'assez d'informations sur l'environnement d'évolution du robot. Il n'est alors pas possible de concevoir un trajet complet du robot vers sa cible avant son déplacement. Les méthodes de planification doivent donc prendre en compte de façon dynamique les nouvelles données sur l'environnement de travail et construire, au fur et à mesure, la trajectoire de déplacement du robot [8].

Le principe consiste à engendrer progressivement des incrémentations de déplacement pour permettre de se rapprocher du but en utilisant une information locale sur les contraintes imposées par l'environnement. Cette information peut être issue, par exemple, de capteurs de distances ou de capteurs tactiles ou encore, directement, par un algorithme de calcul de distances [9]. Ces méthodes ont les avantages suivants [9] :

- Rapidité pour l'obtention d'une solution,
- Possibilité de prendre en compte rapidement les obstacles imprévus,
- Obtention d'une trajectoire continue (non segmentée) qu'il n'est pas nécessaire de lisser.

Cependant, dans certains cas, les méthodes locales échouent et ne trouvent pas une trajectoire faisable alors qu'une solution existe.

II.3.1.Méthode des champs de potentiels

Cette méthode se base sur l'approche des champs de potentiels en physique. Ce sont ceux-ci qui guideront le robot vers sa destination finale dans son environnement. Cet algorithme, proposé par Khatib [10], fonctionne de la manière suivante : les obstacles sont représentés par des champs de potentiels répulsifs afin d'éviter tout contact avec ceux-ci. L'objectif est lui représenté par un champ de potentiel attractif. Le robot est vu comme une particule plongée dans un champ de potentiel équivalent à la somme du champ répulsif et du champ attractif. Le déplacement du robot est calculé de manière itérative. Celui-ci s'effectue suivant une descente du gradient (voir Figure 10).

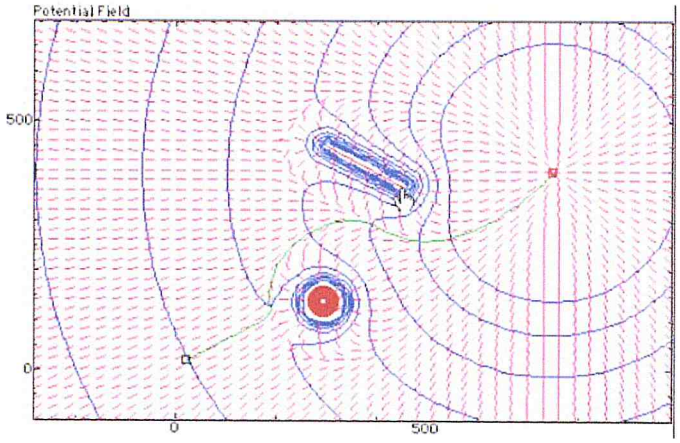


Figure 10 : Exemple d'application du champ de potentiel [11].

Cette méthode est très simple d'utilisation et son coût de calcul est faible. Cependant, elle possède plusieurs limitations. Le robot peut être pris dans des minima locaux qui le paralyseront. Des mouvements d'oscillations peuvent aussi apparaître et l'algorithme ne permet pas de passer entre deux obstacles assez proches. Afin de corriger ces défauts, plusieurs méthodes complémentaires ont été mises en place ; par exemple, le suivi des murs pour éviter le problème d'oscillations [8], etc.

II.3.2. Méthode de la fenêtre dynamique

Cette méthode, proposée par Fox [12], permet l'évitement d'obstacles en temps réel grâce à la configuration des commandes du robot. Elle travaille sur le domaine des vitesses possibles du robot qui permettent d'éviter les obstacles. Dès que l'environnement du robot est analysé, l'algorithme procède à une optimisation des commandes des moteurs en fonction de différents éléments comme le temps de parcours, l'optimisation de la vitesse, la distance aux obstacles, etc. Le résultat est un couple de valeur (v, ω) correspondant à la vitesse de translation et de rotation du robot. L'avantage de cette méthode est qu'elle prend en considération la cinématique du robot. Néanmoins, elle est difficilement intégrable dans un système multi-robots et ne fonctionne pas pour des obstacles en mouvement.

II.3.3.Méthode de diagramme de proximité

Cette méthode de planification de trajectoire, crée par Minguez et Montano[13], porte sur un diagramme de proximité des obstacles mis à jour continuellement lors du déplacement du

robot (voir Figure 11). La trajectoire choisie est celle où il n'y a pas d'obstacles détectés et également la distance la plus proche par rapport à l'objectif à atteindre. Plusieurs comportements du robot peuvent alors être implémentés. Il peut avancer en ligne droite vers l'objectif, passer entre deux obstacles ou les contourner. La vitesse donnée dépendra de la distance par rapport à l'obstacle. Cette méthode peut être adaptée à la morphologie du robot. Cependant, elle ne considère pas la cinématique du robot et la convergence vers la destination n'est pas assurée.

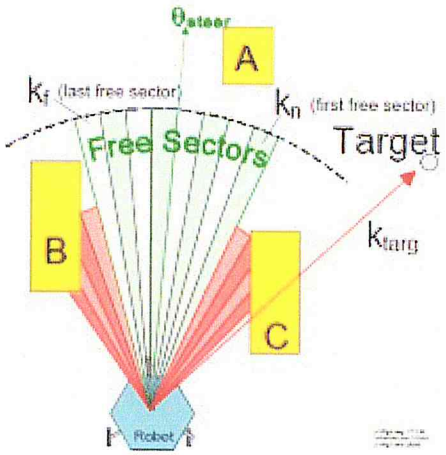


Figure 11 : Diagramme de proximité [14]

II.3.4. Champs de potentiel et recuit simulé

La combinaison des champs de potentiels et du recuit simulé est une approche créée par Zhang [19]. Elle propose une solution au problème de la non-convergence vers la destination finale lorsque cette dernière est située près des obstacles, ainsi que pour résoudre le problème des minima locaux.

II.3.5. Champs de potentiel et algorithme d'évolution bactérien

Les limitations dues à la méthode du champ de potentiel peuvent être réduites fortement par un algorithme couplant cette méthode et un algorithme génétique [20]. Ce nouvel algorithme peut trouver les valeurs optimales des différents champs de potentiels attractifs et répulsifs, permettant d'éviter notamment le problème des minima locaux. Néanmoins, les contraintes cinématiques et dynamiques du robot ne sont pas prises en compte.

II.3.6. Logique floue

La logique floue est largement appliquée dans différents domaines de la robotique, notamment pour la planification de trajectoires, l'évitement d'obstacles, la commande en vitesse, la commande en position, etc. Dans le domaine de planification, cette approche est souvent appliquée pour les robots mobiles comme méthode alternative aux techniques conventionnelles de planification basées sur les modèles mathématiques. Cependant, la génération de mouvements pour les robots manipulateurs basée sur la logique floue est plus difficile à cause de leur structure mécanique complexe (plusieurs articulations liées entre elles qui doivent se déplacer). Par conséquent, très peu de travaux ont été réalisés dans ce domaine par rapport à ceux effectués sur les robots mobiles [21].

Wu et ses collègues [22] ont proposé une approche basée sur la logique floue et les systèmes multi-agents pour la planification de mouvements des manipulateurs industriels. Dans cette approche, chaque articulation du robot est contrôlée par un agent. Aussi, la planification de mouvements s'effectue en deux niveaux : (*i*) un niveau haut dans lequel des comportements sont assignés à chaque agent et (*ii*) un niveau bas permettant de calculer la vitesse de l'articulation en fonction du comportement assigné.

Un autre contrôleur proposé par Zavlangas [23] utilise la logique floue pour la navigation et l'évitement d'obstacles pour un robot manipulateur industriel à 3 ddl. Ce contrôleur est divisé en plusieurs unités de contrôle floues séparées ; chaque unité est responsable d'une seule articulation.

Une approche génétique-floue pour le contrôle d'un système multi-robots manipulateurs partageant un espace de travail commun a été proposée dans [21]. Dans cette approche, chaque manipulateur considère les autres robots comme obstacles dynamiques. Le principe de l'approche consiste à produire un mouvement initial en utilisant les algorithmes génétiques ; après, un contrôleur flou est utilisé pour corriger le mouvement produit afin d'éviter les collisions avec les autres robots.

Fu et al. [24] ont utilisé la logique floue pour la planification de mouvements dans des environnements inconnus pour les manipulateurs à 3 ddl. Le contrôleur proposé est composé d'unités floues séparées, qui contrôlent individuellement chaque articulation.

Un autre travail est proposé par Salah [25] pour le contrôle d'un manipulateur à 6 ddl. Dans ce travail, le robot est supposé évoluer dans un champ de forces. Son orientation est

déterminée à partir de deux forces :(*i*) une force attractive dont la source est la cible et (*ii*) une force répulsive déterminée en utilisant le contrôleur flou.

II.4. Méthodes globales

Les méthodes globales sont basées sur une connaissance complète de l'environnement dans lequel le robot évolue. L'espace libre est généralement représenté dans l'espace des configurations du robot (ou $C - Space$) [26] par une série de paramètres donnant la position et l'orientation de l'effecteur. L'avantage de formuler le problème de planification dans le $C - Space$ est qu'il devient équivalent à la navigation d'un point objet.

Afin de pouvoir représenter l'espace des configurations libres, les méthodes globales ont besoin d'une transformation des obstacles à partir de leurs représentations cartésiennes en une représentation dans l'espace de configuration du robot [27]. Il est ainsi possible de déterminer les configurations qui emmènent le robot de la configuration initiale à la configuration finale souhaitée, en évitant celles qui provoquent des collisions.

II.4.1. Graphes

Ces méthodes de résolutions visent à analyser la topologie des espaces sans obstacles dans le but de simplifier le problème de recherche de plus court chemin dans un graphe entre le point de départ et le point de destination. Elles s'effectuent donc en deux étapes :

- la première étape consiste à créer le graphe de l'environnement à analyser,
- La seconde étape est de parcourir le graphe afin de trouver le plus court chemin. Celle-ci utilise des algorithmes assez connus comme ceux de Bellman [28] et Dijkstra [29].

II.4.2. Algorithme génétique

Cette méthode a été créée dans les années 60 par John Holland et est encore très utilisée de nos jours [30] dans le domaine de l'optimisation et de planification de mouvements. L'algorithme génétique fonctionne en simulant des solutions à chaque itération. Celles-ci sont représentées par une population d'individus possédant des gènes. À chaque itération, seul les gènes qui optimisent la fonction coût du trajet sont transmis à la nouvelle population et ainsi de suite jusqu'à ce qu'il ne reste que les gènes de la fonction coût minimale. Cette population sera alors le trajet recherché (Figure 12).

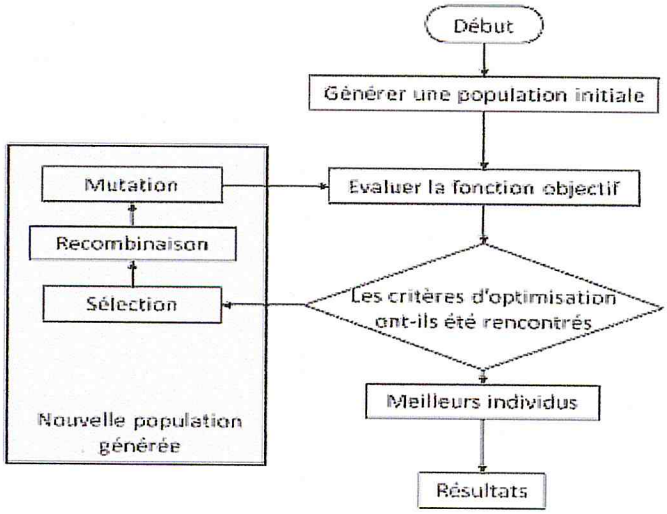


Figure 12: Organigramme d'un algorithme génétique [31]

II.5. Méthodes mixtes

La présence de minima locaux reste la cause la plus importante d'inefficacité des méthodes locales. D'autre part, les approches globales sont très coûteuses en temps de calcul et, en conséquence, elles ne sont pas adaptées pour des applications en temps réel. C'est pourquoi plusieurs chercheurs ont essayé de combiner les deux grandes techniques précédentes.

Les méthodes mixtes opèrent en deux phases. La première phase, réalisée hors ligne et appliquée généralement aux trois premiers ddl, est nettement inspirée des méthodes globales. La deuxième phase, inspirée des méthodes locales, est exécutée en ligne et traite l'intégralité des ddl [27]. Dans la première phase, une planification de type globale est réalisée ; il s'agit de construire un graphe représentant l'environnement et rechercher une trajectoire dans ce graphe. La principale différence repose sur le fait que le graphe est très simplifié et, par conséquent, le temps de calcul et l'espace mémoire utilisés sont réduits considérablement. Lors de la deuxième phase, une méthode locale est utilisée pour la navigation du robot et la trajectoire de référence est donc modifiée au fur et à mesure que le robot avance. La transition dans le graphe entre les différents nœuds ou cellules est calculée de différentes façons ; par exemple, à l'aide d'une fonction de potentiel. Ceci permet d'améliorer la trajectoire calculée initialement. Quelques travaux faisant coopérer les deux grandes classes sont cités ci-après.

II.5.1. DKP :DeterminiticKinodynamic Planning

Cette méthode de planification [32] est assez complexe ; mais, elle permet de respecter toutes les contraintes du robot et d'éviter les problèmes des minimas locaux.

DKP peut être décomposée en deux parties. La première planifie localement des bouts de trajectoires polynomiales (d'ordre deux) sur différents temps de parcours. Pour trouver un bout de trajectoire optimale, il suffit de chercher une solution optimale dans l'espace des paramètres des polynômes via une résolution géométrique de contraintes. Avec ce planificateur local, on construit un arbre de solutions locales. La seconde partie de l'algorithme est un planificateur global basé sur l'algorithme A* qui va choisir les bouts de trajectoires à adopter pour résoudre le problème.

L'avantage de cette méthode est qu'elle génère une grande diversité de bouts de polynômes, ce qui permet de trouver une solution même dans les environnements très complexes :

- Le robot Hilare au LAAS calcule la trajectoire avec une méthode globale pour les grands mouvements (espace non contraint) et déclenche la méthode des champs de potentiels si l'encombrement devient élevé (espace contraint) ou s'il détecte un obstacle imprévu.
- Les auteurs dans [33] et [34] utilisent une méthode mixte avec un planificateur global tels que les HEKM (*Hierarchical Extended KohonenMap*) et un planificateur local utilisant les champs de potentiel.

II.5.2. Méthode de Barraquand et Latombe

Barraquand [35] et Latombe [36] proposent une approche qui combine une méthode locale basée sur les champs de potentiel avec une représentation hiérarchique par « discrétisation » où l'environnement est modélisé par une carte 2D de potentiel avec un seul minimum (Cellule-cible) associée à une table de distance. Chaque cellule disponible à la navigation est modélisée par une structure de données comportant deux champs : le premier contient le niveau de potentiel par rapport à la cellule cible et le deuxième comporte la distance de cette cellule par rapport à l'obstacle le plus proche. Les deux tâches demandées à un générateur de trajectoires sont : faire évoluer le robot vers la cellule cible et éviter les obstacles. Ceci mène souvent à des situations de blocage.

Les solutions mentionnées auparavant évitent, dans la plupart des cas la convergence vers des minima locaux ou permettent de s'échapper de ces minima. Par contre, elles ne sont pas

adaptées à travailler en temps réel. En fait, le temps nécessaire pour la phase de reconnaissance hors ligne de l'environnement pénalise fortement leur application en ligne, notamment lorsque l'environnement est fortement encombré. Ces méthodes deviennent, elles aussi, très lentes quand l'espace de travail où le robot évolue change continuellement.

II.6. Conclusion

Dans ce chapitre, nous avons présenté les deux grandes classes de calcul de trajectoires sans collision. Les méthodes globales garantissent l'obtention d'une classe de solutions parmi lesquelles le choix d'une trajectoire particulière peut s'effectuer suivant des contraintes d'optimalité, mais elles ne sont pas adaptées pour travailler en ligne à cause de temps de calcul qui croit exponentiellement en fonction du nombre de ddl. Alors que les méthodes locales apportent rapidité de génération et gestion de l'environnement dynamique visible, mais elles n'ont toujours pas résolues les problèmes dus aux phénomènes de minima locaux (blocage). La troisième classe, méthodes mixtes, combine les deux premières et opère en deux phases. La première phase est inspirée des méthodes globales et la deuxième est inspirée des méthodes locales.

Le prochain chapitre décrit l'approche utilisée pour la planification de trajectoires sans collision pour les robots manipulateurs. C'est une des techniques de l'intelligence artificielle basée sur la logique floue. Ces dernières sont considérées comme étant une solution intéressante aux systèmes non linéaires où il est difficile d'établir des modèles mathématiques.

Chapitre III
Approche proposée pour la planification de trajectoires sans collision

III.1. Introduction

Ce chapitre est dédié à la description de l'approche proposée pour la planification de trajectoires sans collision pour les robots manipulateurs. Il est scindé en deux parties.

La première partie de ce chapitre décrit le contrôleur flou utilisé, développé dans [37]. Elle commence par présenter les notions principales constituant la base de conception des contrôleurs flous.

Quant à la deuxième partie de ce chapitre, elle est dédiée à la description de l'exploitation des images délivrées par la caméra Kinect pour le calcul des coordonnées cartésiennes des segments constituants le robot manipulateur.

III.2. Description de l'approche floue utilisée

L'utilisation des outils mathématiques de modélisation est appropriée et justifiée pour les systèmes bien définis. Mais, quand la complexité augmente, ces outils deviennent moins efficaces. Le traitement des systèmes complexes nécessite souvent la manipulation d'informations incertaines. De façon naturelle, l'être humain est capable de manipuler de tels systèmes. Il décrit son comportement par des méthodes approximatives au lieu de raisonner en termes mathématiques [38].

Un robot doit posséder des capacités de perception et de mouvement nécessaires pour qu'il puisse exécuter ses tâches dans son environnement [39]. La méthode générale pour contrôler un robot manipulateur consiste à calculer les modèles géométriques pour générer les mouvements du robot nécessaires pour l'accomplissement d'une tâche particulière [40]. Ces approches produisent de bons résultats pour des tâches répétitives dans des environnements connus [41].

Les techniques de l'intelligence artificielle basées sur la logique floue sont considérées comme une solution intéressante pour les systèmes non-linéaires où il est difficile d'établir des

modèles mathématiques [42]. L'évitement d'obstacle pour les robots manipulateurs appartient aux problèmes de cette catégorie [43].

La logique floue fournit un meilleur moyen d'automatiser l'expertise humaine. Elle permet aux experts humains d'exprimer leurs connaissances, généralement difficiles à mettre en application lors de la conception de systèmes de traitement classiques, de façon naturelle avec un minimum des règles pour exprimer les concepts ; ceci engendre donc un gain de temps et d'espace mémoire, ce qui donne une rapidité considérable aux moteurs d'inférence [42].

En tenant compte de ce constat, l'utilisation de la logique floue comme méthode de base du contrôleur utilisé est motivée aussi, d'un côté, par son efficacité dans la gestion de l'imprécision des modèles mathématiques du robot, et l'incertitude qui pourrait se présenter dans sa structure mécanique. D'un autre côté, aucun système d'équations cinématiques ou différentielles complexes n'est requis pour contrôler le robot. En effet, la prise de décision dans un système de contrôle flou est basée sur des termes linguistiques et des règles développées par des experts humains.

Les applications utilisant la logique floue sont plus faciles à réaliser et à utiliser. Des machines complexes peuvent devenir plus conviviales grâce à leur capacité de raisonnement avec des informations imprécises et d'explication de leurs décisions. Les applications de la logique floue qui ont été réalisées dans la littérature montrent les avantages de cette technique quand les modèles des systèmes sont difficiles à implémenter, ce qui en fait un outil robuste, simple et adéquat pour traiter ces problèmes [44].Ses avantages viennent notamment de ses capacités à [45] :

- formaliser et simuler l'expertise d'un opérateur ou d'un concepteur dans la conduite et le réglage d'un procédé,
- donner une réponse simple pour les procédés dont la modélisation est difficile,
- prendre en compte sans discontinuité des cas ou exceptions de natures différentes, et les intégrer au fur et à mesure dans l'expertise,
- prendre en compte plusieurs variables et effectuer de la « fusion pondérée » des grandeurs d'influence.

III.2.1.Logique floue : Définitions et concepts de base

Le terme d'ensemble flou apparaît pour la première fois en 1965 par Zadeh [46]. Il a réalisé depuis de nombreuses avancées théoriques majeures dans le domaine et a été rapidement accompagné par de nombreux chercheurs développant des travaux théoriques et appliqués plus tard pour la première fois par Mamdani [47]. Zadeh a introduit la notion de sous-ensembles flous permettant de graduer l'appartenance à un ensemble afin de traiter des données de nature imparfaite.

III.2.1.1.Sous-ensembles flous

Les ensembles flous sont définis par une fonction appelée fonction d'appartenance. Cette fonction associe à chaque élément de l'ensemble flou une valeur qui n'appartient pas forcement à un rang. La valeur associée indique le degré d'appartenance à un ensemble et la valeur la plus élevée indique un degré d'appartenance fort [4] .La définition donnée par Zadeh [46]est la suivante :*Soit X un espace de points (d'objets) avec x un élément générique* $X = \{x\}$*.Un ensemble flou (classe) A dans X est caractérisé parune fonction d'appartenance (caractéristique)*$Fa(x)$ *qui associe à chaque point de X une valeur réelle dans l'intervalle* [0,1], *la valeur* $Fa(x)$ *dans x représente le degré d'appartenance de x dans A* [4].

III.2.1.2.Fonctions d'appartenance

Chaque sous-ensemble flou peut être représenté par sa fonction d'appartenance qui correspond à la notion de « fonction caractéristique » en logique classique. Elle permet de calculer le degré d'appartenance d'un élément à ce sous-ensemble flou. Il existe plusieurs formes que la fonction d'appartenance peut prendre, à savoir triangulaire, trapézoïdale, singleton, etc.

Supposons que nous voulions définir l'ensemble des personnes de « taille moyenne ». En logique classique, nous conviendrons par exemple que les personnes de taille moyenne sont celles dont la taille est comprise entre 1.60m et 1.80m. La fonction caractéristique de l'ensemble donne « 0 » pour les tailles hors de l'intervalle [1.60m, 1.80m] et « 1 » dans cet intervalle (figure 13) [51].

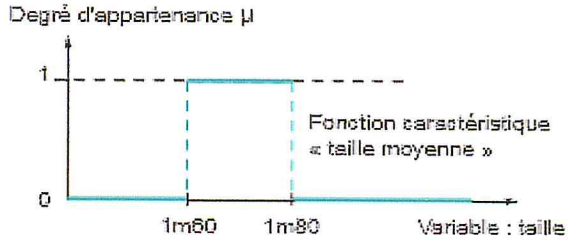


Figure 13 : Fonction caractéristique [45].

L'ensemble flou des personnes de « taille moyenne » sera défini par une « fonction d'appartenance » qui diffère d'une fonction caractéristique par le fait qu'elle peut prendre n'importe quelle valeur dans l'intervalle [0, 1]. À chaque taille possible correspondra un « degré d'appartenance » à l'ensemble flou des « tailles moyennes », compris entre 0 et 1 (figure 14) [51].

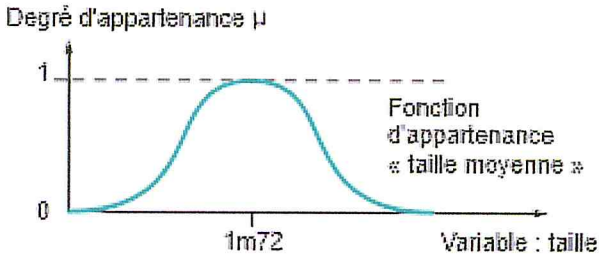


Figure 14 : Fonction d'appartenance [45].

III.2.1.3.Variables linguistiques

Le concept de variables linguistiques joue un rôle primordial dans la logique floue. Une variable linguistique est une variable dont les valeurs sont des mots au lieu de nombres. Figure 19 illustre un exemple de la variable linguistique « *vitesse* » avec trois termes linguistiques : *petite*, *moyenne* et *grande* [48].

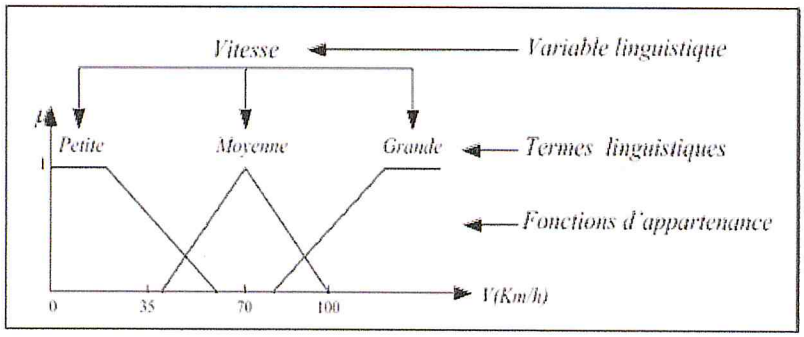


Figure 15 : Exemple d'une variable linguistique [48].

III.2.2.Composant d'un système de logique floue

Un système de logique floue se compose principalement de deux composants [4] : (*i*) une base de règles floue et (*ii*) un moteur d'inférence floue.

III.2.2.1. Base de règles

La base de règles d'un système de logique floue est constituée d'un ensemble de règles sous différentes formes [4] :

- *Sous forme d'affectation* : dans ce type de règles, une valeur est affectée à une variable (exemple, température=élevée).
- *Sous forme conditionnelle* : une condition spécifique est mentionnée ; si la condition est vérifiée, une action est exécutée (exemple, si la moyenne est supérieure ou égale à 10, alors l'étudiant passe à l'année suivante).
- *Sous forme non conditionnelle* : il n'a y pas de condition à vérifier avant d'exécuter une action, elle est sous forme d'une commande (exemple : aller à l'instruction c ; arrêter).

III.2.2.2. Moteur d'inférence

Le rôle du moteur d'inférence est la définition de l'ensemble de la sortie. Selon la base des règles, l'ensemble de sortie est une décision et le moteur d'inférence est le mécanisme de raisonnement approximatif ; généralement, il utilise des méthodes d'inférence spécifique [4].

III.2.3.Structure de base d'un système de contrôle flou

La structure générale d'un contrôleur flou est composée de trois parties essentielles comme montrées par (Figure 16).

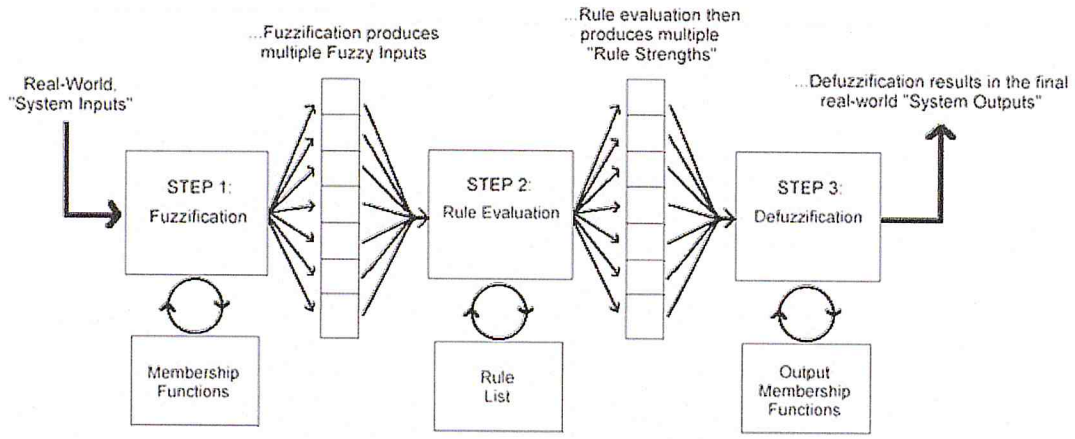


Figure 16 : Architecture interne d'un contrôle flou [37].

III.2.3.1.Fuzzification

L'étape de *fuzzification* a pour objectif de transformer les données numériques des variables d'entrées en variables linguistiques utilisables par le système d'inférence. Il s'agit alors de déterminer le degré d'appartenance de chaque entrée aux ensembles flous associés et prédéfinis dans la base de données du système flou.

III.2.3.2.Système d'inférence

Le système d'inférence est relié à une base de connaissances contenant les définitions des fonctions d'appartenance (paramètres et formes) associées aux variables d'entrée/sortie. Il contient aussi l'ensemble des règles floues sous forme « *Si condition Alors conclusion* » préalablement définies à partir de l'expertise d'un expert humain.

L'étape d'inférence consiste à appliquer chaque règle aux variables linguistiques calculées par l'étape de *fuzzification*. Il s'agit de calculer le degré de vérité des différentes règles floues et d'associer à chacune d'entre elles une valeur de sortie qui sera transférée à l'étape de *défuzzification*.

III.2.3.3.Défuzzification

L'étape de *défuzzification* consiste à transformer l'ensemble des valeurs de sortie des différentes règles résultant de l'inférence en une grandeur de commande précise (c.-à-d. déterminer la valeur quantitative en termes de fonctions d'appartenance des variables linguistiques). Il existe plusieurs méthodes de défuzzification telles que *la méthode de centre de gravité*, *la méthode de maximum* et *la méthode de la moyenne des maxima* [49].

III.3.Fonctionnement du système propose

Si le chemin du robot manipulateur est libre, c.-à-d. aucune présence d'obstacle ; donc dans cette situation, le robot se déplace vers la cible jusqu'à ce qu'un obstacle soit détecté. À ce point, la direction du bras est modifiée par le contrôleur flou d'évitement d'obstacle de telle sorte que le manipulateur se déplace aussi loin de l'obstacle. Le chemin de déplacement est représenté comme un ensemble de points de passage reliant le point de départ au point d'arrivée. Le principe de fonctionnement est présenté dans la figure suivante (Figure 17).

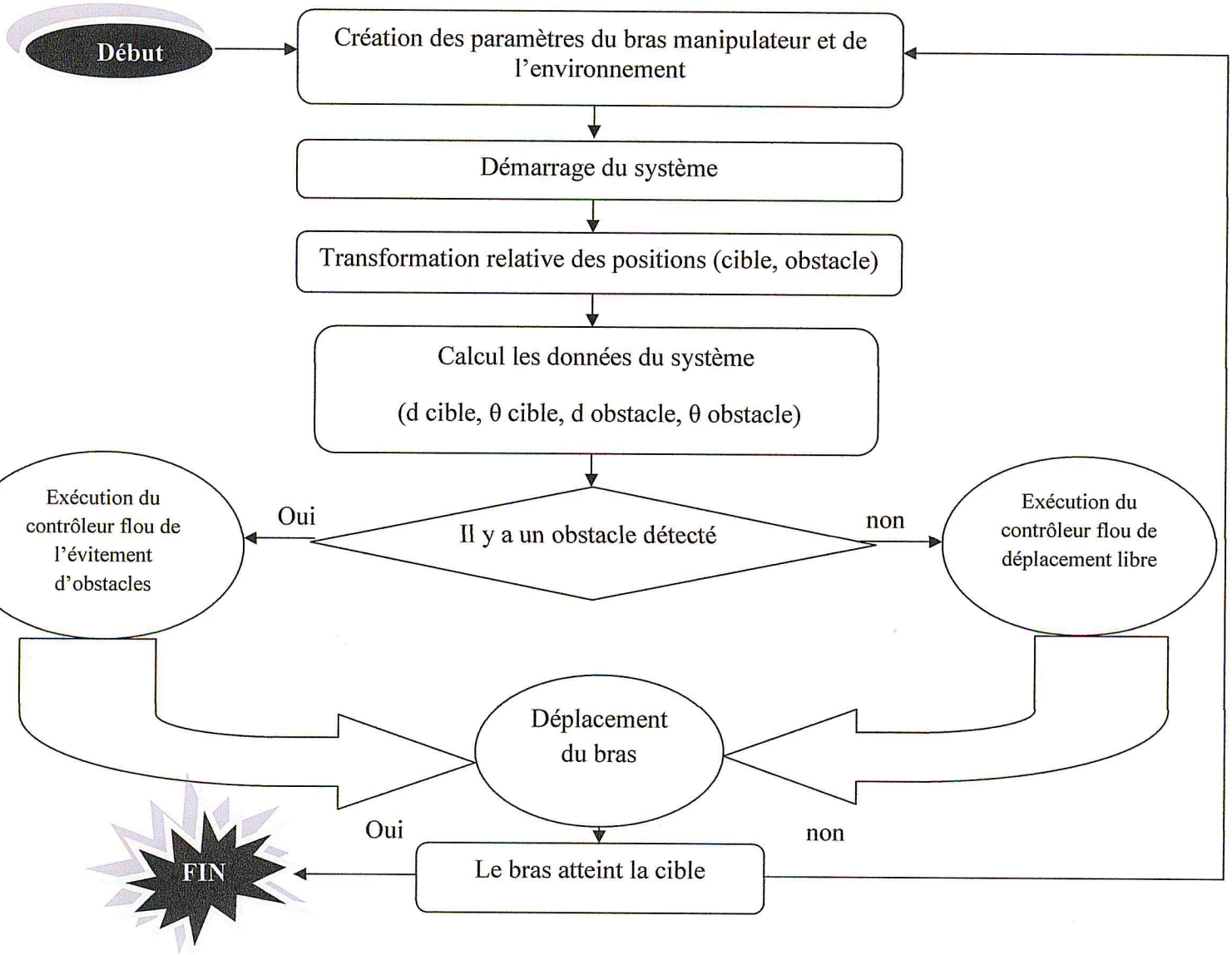


Figure 17 : Architecture fonctionnelle du contrôleur flou proposé.

III.3.1.Contrôleur flou de déplacement libre

Elle permet au manipulateur d'atteindre sa cible. Pour cela, chaque itération calcule les consignes de contrôle à envoyer aux articulations du robot permettant ainsi à l'effecteur de se rapprocher de la cible. Elle possède quatre entrées qui représentent les variables d'état du robot et n sorties (où n représente le nombre de ddl du robot) utilisées comme variables de contrôle. Les entrées sont données comme suit [50] :

- *Distance* (Δd) : c'est la distance euclidienne entre la position actuelle de l'effecteur et celle de la cible. Cette distance est calculée comme suit :

$$\Delta d = \sqrt{\left(x_{Effecteur} - x_{Cible}\right)^2 + \left(y_{Effecteur} - y_{Cible}\right)^2 + \left(z_{Effecteur} - z_{Cible}\right)^2} \quad (1)$$

- *Erreur d'altitude*$(\Delta Altitude)$: elle est calculée par l'équation (2) ; cette erreur correspond à la différence entre l'altitude de l'effecteur du robot et l'altitude de la cible (selon l'axe$\vec{z}$) :

$$\Delta Altitude = Z_{Effecteur} - Z_{Cible} \quad (2)$$

- *Erreur angulaire* $(\Delta\theta)$: cette erreur est définie comme la différence entre l'angle de l'effecteur et celle de la cible ; elle est donnée par l'équation (3).

$$\Delta\theta = \theta_{Effecteur} - \theta_{cible} \quad (3)$$

- *Direction de la cible* $(Target - dir)$: cette entrée permet d'indiquer si la cible est en face $(Front)$ ou derrière $(Behind)$ l'effecteur du robot sur le plan (XY) :

$$Target - dir = \begin{cases} Front\ si\left(x_{Effecteurr} - x_{Cible} < 0\right) \vee \left(y_{Effecteurr} - y_{Cible} < 0\right) \\ Behind\ si\left(x_{Effecteur} - x_{Cible} > 0\right) \vee \left(y_{Effecteurr} - y_{Cible} > 0\right) \end{cases} \quad (4)$$

Les sorties sont les consignes en position (angles) θ_i $(i = 1 \ldots ddl)$ des articulations du robot manipulateur qui doivent respecter les butées articulaires $\theta_i \in [\theta_i^{min}, \theta_i^{max}]$.

III.3.1.1.Fonctions d'appartenance des entrées/sorties du contrôleur

Il existe une variété de fonctions d'appartenance qui peuvent être utilisées pour représenter les ensembles flous. Le choix de ces fonctions dépend de la nature du problème et

des données. Comme le montre la figure 18a, l'entréeΔdest partitionnée en deux sous-ensembles flous de formes triangulaire et trapézoïdale. Quant à $\Delta\theta$, nous avons choisi sept sous-ensembles flous de formes singleton, triangulaire et trapézoïdale. Les termes linguistiques utilisés pour ces entrées sont donnés comme suit :

- *Z* : Zero, *S* : Small, *M* : Medium, *B* : Big.
- *B_UP* : Big UP, *M_UP* : Medium UP, *S_UP* : Small UP.
- *S_Down* : Small Down, *M_Down* : Medium Down, *B_Down* : Big Down.
- *B_Left* : BigLeft, *M_Left* : Medium Left, *S_Left* : Small Left.
- *S_Right* : Small Right, *M_Right* : Medium Right, *B_Right* : Big Right.

En général, les robots manipulateurs sont conçus avec des articulations rotatives où deux types de mouvements peuvent être distingués pour chaque articulation :

- *Mouvement de type* $Right/Left$: deux actions sont définies pour ce type :
  - *tourner à droite*,
  - *tourner à gauche*.
- *Mouvement de type*$Up/Down$: deux actions peuvent être effectuées :
  - *lever l'effecteur* du robot,
  - *baisser l'effecteur* du robot.

Par conséquent, les fonctions d'appartenance des variables de sortie $\theta_i$du contrôleur flou de déplacements libres qui appartiennent à l'intervalle $[-40, +40]$ des deux types de mouvements sont illustrées par la figure 18b.

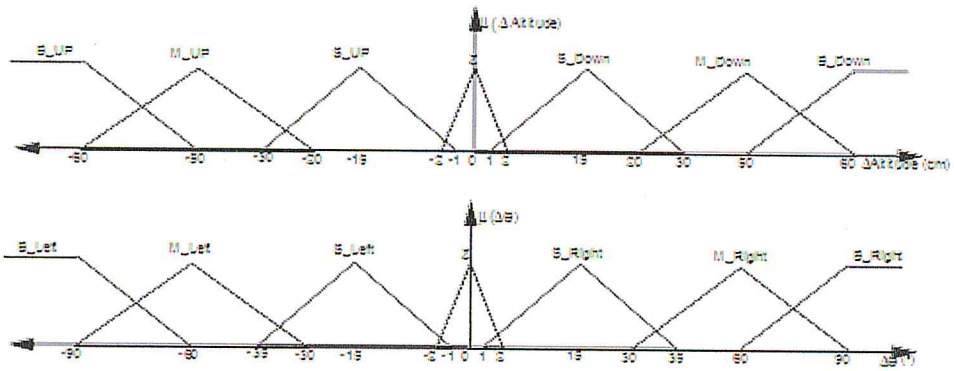
Les termes linguistiques utilisés pour les sorties de type de mouvements$Right/Left$ sont les suivants :

- *BMR* : Big Move Right, *BML* : Big Move Left.
- *MMR* : Medium Move Right, *MML* : Medium Move Left.
- *SMR* : Small Move Right, *SML* : Small Move Left.

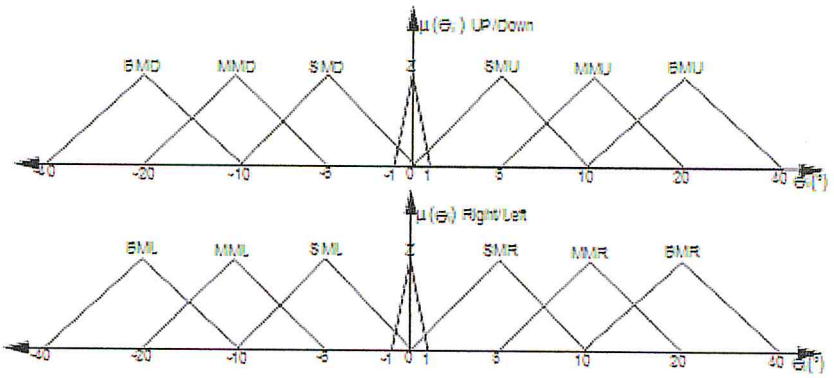
Les termes linguistiques utilisées pour les sorties de type de mouvements$Up/Down$ sont donnés comme suit :

- *BMU* : Big Move Up, *BMD* : Big Move Down.

- *MMU* : Medium Move Up, *MMD* : Medium Move Down.
- *SMU* : Small Move Up, *SMD* : Small Move Down.



(a) Fonctions d'appartenance floue des variables d'entrée



(b) Fonctions d'appartenance floue pour les variables de sortie.

Figure 18 : Fonctions d'appartenance des variables d'entrée et de sortie.

III.3.1.2.Base de règles floues utilisée

Le premier type de mouvement, *Mouvement de typeUp/Down*, permet au robot de minimiser l'erreur d'altitude. Quant au second type, *Mouvement de typeRight/Left* ,il minimise l'erreur angulaire. Le raisonnement suivi pour concevoir les règles du contrôleur flou de déplacements libres, récapitulées dans le tableau 1, est résumé comme suit :

- Si la position de la cible est à gauche ou à droite par rapport à la position de l'effecteur, le robot se déplace vers la cible en utilisant ses articulations ayant le type de mouvements$Right/Left$.
- Si la position de la cible est dessus ou en-dessous de la position de l'effecteur, le robot se déplace vers le haut ou vers le bas en utilisant ses articulations ayant le type$Up/Down$.
- Si$\Delta\theta \approx 0$, $\Delta Altitude \approx 0$ et $\Delta d \neq 0$ (c.-à-d. le manipulateur n'a pas encore atteint sa cible), le robot se déplace sur une ligne droite tout en utilisant les deux premières articulations de type $Up/Down$ selon les deux règles suivantes :
  - Si la cible est à l'avant de l'effecteur ($Target - dir = Front$), alors le robot effectue un mouvement $Down$(vers la bas) avec la première articulation et un mouvement Up (vers le haut) avec la deuxième articulation.
  - Si la cible est derrière l'effecteur ($Target - dir = Behind$), alors le robot effectue un mouvement Up (vers le haut) avec la première articulation et un mouvement $Down$ (ver le bas) avec la deuxième articulation.

<table>
<tr><th rowspan="2">$\Delta\theta$</th><th rowspan="2">$\Delta Altitude$</th><th colspan="3">$Distance$ (Δd)</th></tr>
<tr><th colspan="2">Near/Far</th><th>Zero</th></tr>
<tr><td rowspan="3">B_Left
M_Left
S_Left</td><td>B_Up
M_UP
S_UP</td><td colspan="2">-Faire un mouvement Up pour lesθ_i de type de mouvements $Up/Down$.
-Faire un mouvement $Left$ pourlesθ_i de type de mouvements $Right/Left$.</td><td rowspan="3">–</td></tr>
<tr><td>Z</td><td colspan="2">Faire un mouvement $Left$ pour lesθ_i de type de mouvement $Right/Left$.</td></tr>
<tr><td>S_Down
M_Down
B_Down</td><td colspan="2">-Faire un mouvement $Down$pour les θ_i de type de mouvements $Up/Down$.
-Faire un mouvement $Left$ pour lesθ_i de type de mouvements $Right/Left$.</td></tr>
<tr><td rowspan="2">Z</td><td>Big_Up
M_Up
S_Up</td><td colspan="2">-Faire un mouvement Up pour les $\theta_i$de type de mouvements $Up/Down$.</td><td>Faire un mouvement Up d'une articulation de type de mouvements $Up/Down$</td></tr>
<tr><td>Z</td><td>$Target_{dir} = Front$</td><td>$Target_{dir} = Behind$</td><td>–</td></tr>
</table>

		-Faire un mouvement $Down$ de la première articulation de type de mouvements $Up/Down$. -Faire un mouvement Up de la deuxième articulation de type de mouvements $Up/Down$.	-Faire un mouvement Up de la première articulation de type de mouvements $Up/Down$. -Faire un mouvement $Down$ de la deuxième articulation de type de mouvements $Up/Down$.	
	S_Down **M_Down** **B_Down**	Faire un mouvement $Down$pour les$\theta_i$de type de mouvement$Up/Down$.		Faire un mouvement $Down$ d'une articulation de type de mouvements$Up/Down$
B_Right **M_Right** **S_Right**	**B_Up** **M_Up** **S_Up**	-Faire un mouvement UP pour lesθ_i de type de mouvements $Up/Down$. -Faire un mouvement $Right$ pour lesθ_i de type de mouvements $Right/Left$.		–
	Zero	-Faire un mouvement $Right$pour lesθ_i de type de mouvements $Right/Left$.		
	S_Down **M_Down** **B_Down**	-Faire un mouvement $Down$ pour lesθ_i de type de mouvements $Up/Down$. -Faire un mouvement $Right$ pour lesθ_i de type de mouvements $Right/Left$.		

Tableau 1 : Règles floues du contrôleur flou de déplacement libre.

En tenant compte des règles du tableau 1, le tableau 2 donne plus de détails sur l'application de ces règles sur un manipulateur pris comme exemple illustratif. Il s'agit du robot ULM à 3 ddl ayant 3 articulations rotoïdes θ_1: $(Right/Left)$, θ_2: $(Up/Down)$, θ_3: $(Up/Down)$.

$\Delta\theta$	Δ***Altitude***	***Distance*** (Δd)	
		Near/Far	**Zero**
B_Left	**B_Up**	θ_1=BML, θ_2=BMU, θ_3BMU	–
	M_UP	θ_1=BML, θ_2=MMU, θ_3= SMU	–
	S_UP	θ_1=BML, θ_2=SMU, θ_3 =Z	–

	Z	θ_1=BML, θ_2=Z, θ_3=Z		–
	S_Down	θ_1=BML, θ_2=SMD, θ_3= Z		–
	M_Down	θ_1=BML, θ_2=MMD, θ_3= SMD		–
	B_Down	θ_1=BML, θ_2=BMD, θ_3= BMD		–
M_Left	**B_Up**	θ_1=MML, θ_2=BMU, θ_3=BMU		–
	M_UP	θ_1=MML, θ_2=MMU, θ_3=SMU		–
	S_UP	θ_1=MML, θ_2=SMU, θ_3= Z		θ_1=SML, θ_2=SMU, θ_3=Z
	Z	θ_1=MML, θ_2=Z, θ_3=Z		θ_1=SML, θ_2=Z, θ_3=Z
	S_Down	θ_1=MML, θ_2=SMD, θ_3= Z		θ_1=SML, θ_2=SMD, θ_3=Z
	M_Down	θ_1=MML, θ_2=MMD, θ_3= SMD		–
	B_Down	θ_1=MML, θ_2=BMD, θ_3= BMD		–
S_Left	**B_Up**	θ_1=SML, θ_2=BMU, θ_3=BMU		–
	M_UP	θ_1=SML, θ_2=MMU, θ_3= SMU		–
	S_UP	θ_1=SML, θ_2=SMU, θ_3= S		θ_1=SML, θ_2=SMU, θ_3=Z
	Z	θ_1=SML, θ_2=Z, θ_3=Z		θ_1=SML, θ_2=Z, θ_3=Z
	S_Down	θ_1=SML, θ_2=BMU, θ_3= Z		θ_1=SML, θ_2=SMD, θ_3=Z
	M_Down	θ_1=SML, θ2=MMD, θ3= SMD		–
	B_Down	θ_1=SML, θ_2=BMD, θ_3=BMD		–
		Near	**Far**	–
Z	**Big_Up**	θ_1=Z, θ_2=MMU, θ_3= Z	θ_1=Z, θ_2=BMU, θ_3= Z	
	M_Up	θ_1=Z, θ_2=SMU, θ_3= Z	θ_1=Z, θ_2=MMU, θ_3= Z	–
	S_Up	θ_1=Z, θ_2=SMU, θ_3= Z	θ_1=Z, θ_2=SMU, θ_3= Z	θ_1=Z, θ_2=SMU, θ_3= Z
	Z	***Target_dir*** =*Front*	*Target_dir = Behind*	θ_1=Z, θ_2=Z, θ_3=Z
		θ_1=Z, θ_2=SMD, θ_3= Z	θ_1=Z, θ_2=SMU, θ_3= Z	
	S_Down	θ_1=Z, θ_2=PMD, θ_3= Z	θ_1=Z, θ_2=PMD, θ_3= Z	θ_1=Z, θ_2=SMD, θ_3= Z
	M_Down	θ_1=Z, θ_2=MMD, θ_3= Z	θ_1=Z, θ_2=MMD, θ_3= Z	–
	B_Down	θ_1=Z, θ_2=MMD, θ_3= Z	θ_1=Z, θ_2=BMD, θ_3= Z	–
B_Right	**B_Up**	θ_1=BMR, θ_2=BMU, θ_3=BMU		–
	M_Up	θ_1=BMR, θ_2=MMU, θ_3= SMU		–
	S_Up	θ_1=BMR, θ_2=SMU, θ_3= Z		–
	Zero	θ_1=BMR, θ_2=Z, θ_3=Z		–
	S_Down	θ_1=BMR, θ_2=SMD, θ_3= Z		–
	M_Down	θ_1=BMR, θ_2=MMD, θ_3= SMD		–

	B_Down	θ_1=BMR, θ_2=BMD, θ_3=BMD	–
M_Right	**B_Up**	θ_1=MMR, θ_2=BMU, θ_3=BMU	–
	M_Up	θ_1=MMR, θ_2=MMU, θ_3= SMU	–
	S_Up	θ_1=MMR, θ_2=SMU, θ_3= Z	θ_1=SMR, θ_2=SMU, θ_3=Z
	Zero	θ_1=MMR, θ_2=Z, θ_3=Z	θ_1=SMR, θ_2=Z, θ_3=Z
	S_Down	θ_1=MMR, θ_2=SMD, θ_3= Z	θ_1=SMR, θ_2=SMD, θ_3=Z
	M_Down	θ_1=MMR, θ_2=MMD, θ_3SMD	–
	B_Down	θ_1=MMR, θ_2=BMD, θ_3=BMD	–
S_Right	**B_Up**	θ_1=SMR, θ_2=BMU, θ_3=BMU	–
	M_Up	θ_1=SMR, θ_2=MMU, θ_3= SMU	–
	S_Up	θ_1=SMR, θ_2=SMU, θ_3= S	θ_1=SMR, θ_2=SMU, θ_3=Z
	Zero	θ_1=SMR, θ_2=Z, θ_3=Z	θ_1=SMR, θ_2=Z, θ_3=Z
	S_Down	θ_1=SMR, θ_2=BMU, θ_3= Z	θ_1=SMR, θ_2=SMD, θ_3=Z
	M_Down	θ_1=SMR, θ2=MMD, θ3= SMD	–
	B_Down	θ_1=SMR, θ_2=BMD, θ_3=BMD	–

Tableau 2 : Règles floues pour le contrôleur flou de déplacements libres pour le manipulateur à 3ddl.

III.3.2.Contrôleur flou de l'évitement d'obstacles

Il est indispensable d'assurer que le robot ne rentre pas en collision avec les obstacles qui l'entourent. Pour cela, le contrôleur flou de l'évitement d'obstacles permet au robot de se déplacer librement sans collision dans son espace de travail. Pour des raisons de simplicité, les obstacles sont modélisés géométriquement par des cylindres caractérisés par les coordonnées cartésiennes (X_{ob}, Y_{ob}) de centres de leurs bases, leurs rayons R et leurs hauteurs Z_{Ob}(figure 19a).

Le contrôleur flou de l'évitement d'obstacles possède trois entrées (figure 19b) et n sorties. Les fonctions d'appartenance des variables d'entrées sont illustrées par la figure 19 [50].

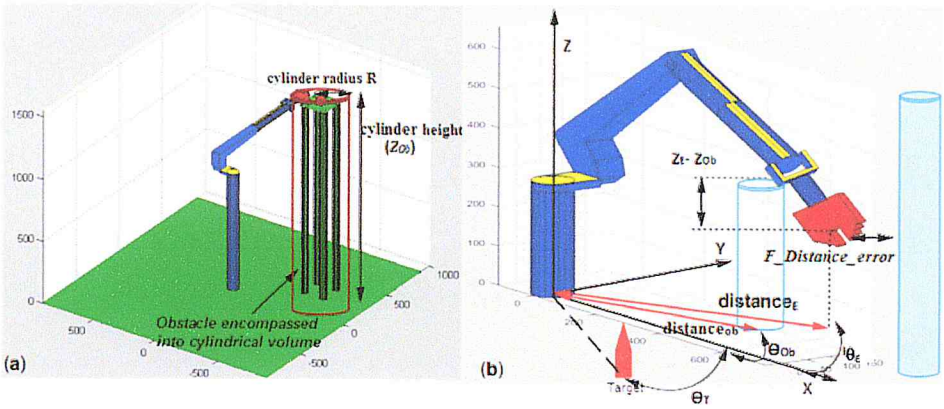


Figure 19 : (a) Obstacles englobant le volume cylindrique, (b) Entrées du contrôleur de l'évitement d'obstacles.

Les entrées sont définies comme suit (Figure 20) :

- *Distance de l'obstacle avant* ($F\_Distance\_error$) : c'est la distance euclidienne entre l'obstacle le plus proche devant le robot et son effecteur.

$$F\_Distance\_error = Min\left(F\_dist_{Ob\_i}\right), i = 1 \ldots m \quad (5)$$

où $F\_dist_{Ob\_i}$ correspond à la distance entre l'effecteur et l'obstacle $i$en face du robot ;m représente le nombre d'obstacles devant le robot (cf. équation 6).

$$F\_dist_{Ob\_i} = \sqrt{\left(x_{Effecteur} - x_{ob\_i}\right)^2 + \left(y_{Effecteur} - y_{ob\_i}\right)^2 + \left(z_{Effecteur} - z_{ob\_i}\right)^2} \quad (6)$$

- *Erreur d'altitude d'obstacles* ($Altitude\_error\_obs$) : elle représente la différence entre l'altitude de l'effecteur et l'altitude de l'obstacle (selon l'axe$\vec{z}$).

$$Altitude\_error\_obs = Z_{Effecteur} - Z_{Ob} \quad (7)$$

- *Erreur angulaire d'obstacle* ($Angle\_error\_obs$) : elle est définie comme étant la différence entre l'angle final calculé de l'effecteur et celle de l'obstacle. $Angle\_error\_obs$ est positive si l'obstacle est à droite ; elle est négative si l'obstacle se trouve à gauche.

$$Angle\_error\_obs = \theta_{Effecteur} - \theta_{Ob} \quad (8)$$

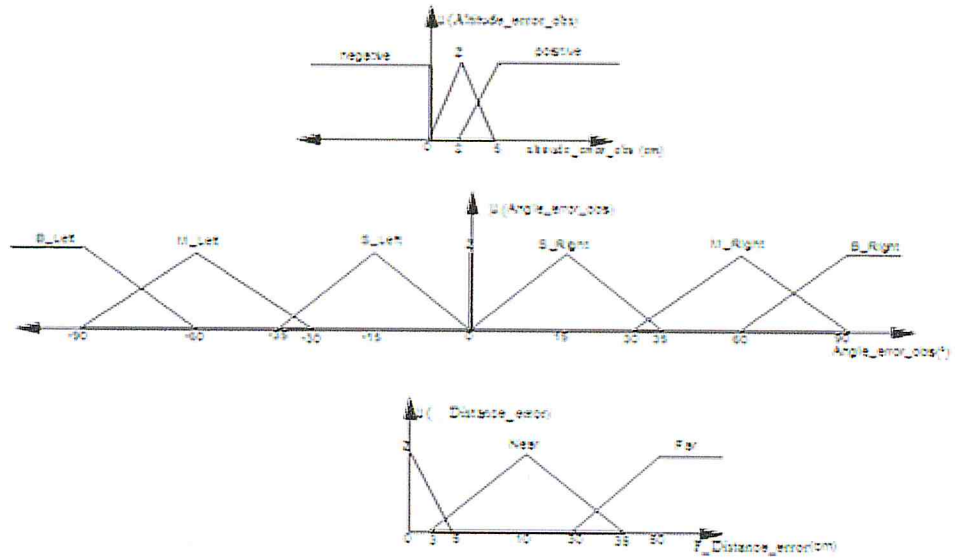


Figure 20 : Fonctions d'appartenance pour les variables d'entrée d'Exécution de l'évitement d'obstacles.

Durant le mouvement du robot, une éventuelle collision avec un obstacle existe dans le cas où $\theta_{obstacle} \epsilon [\theta_{Effecteur}, \theta_{cible}] \wedge (distance_{Effecteur} > distance_{Ob} - \frac{R}{2})$. Dans ce cas, le contrôleur proposé possède deux techniques pour éviter la collision : (*i*) évitement par le dessus et (*ii*) évitement par la droite/gauche. La figure 21 montre le mécanisme d'évitement permettant au robot d'éviter les obstacles par le dessus ; il se fait en deux étapes :

- **Étape 1** : le robot se déplace vers le haut en utilisant l'action Up jusqu'à ce que son effecteur soit au-dessus de l'obstacle ($Z_{Effecteur} > Z_{Ob}$ ou $Altitude\_error\_obs > 0$). Ce mouvement est réalisé en tenant compte de la présence éventuelle d'obstacles devant le robot. Si un obstacle est proche, le robot doit lever son effecteur tout en évitant la collision avec cet obstacle. Ce mouvement est effectué avec deux articulations de même type $Up/Down$ via l'action UP (mouvement vers le haut) pour la première articulation et via l'action $Down$ (mouvement vers le bas) pour la deuxième articulation.

- **Étape 2** : Une fois$Altitude_error_obs > 0$, le robot se déplacera sans collision vers sa cible soit par la droite soit par la gauche en minimisant$Angle_error_obs$. La direction du mouvement est déterminée comme suit :
    - Si $(\theta_{Effeteur} < \theta_{Ob} < \theta_{Cible})$, alors la cible est sur la gauche et l'évitement se fera par la gauche.
    - Si $(\theta_{Effecteur} > \theta_{Ob} > \theta_{Cible})$, alors la cible se trouve à droite et l'évitement se fera par la droite.

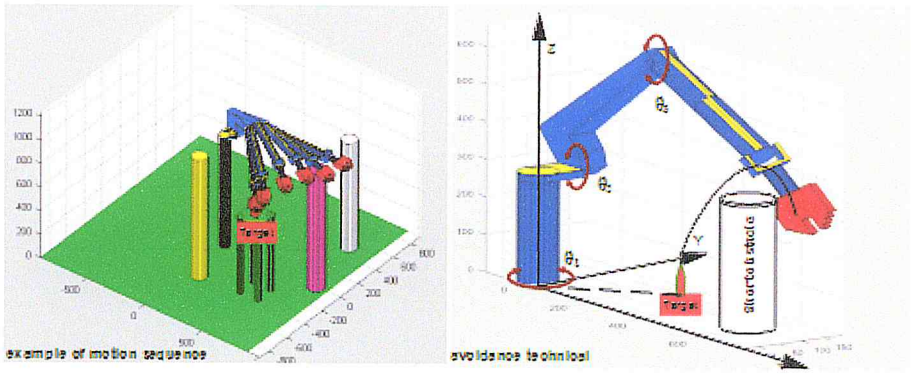


Figure 21: mécanisme d'évitement.

Ce mécanisme d'évitement doit respecter certaines contraintes. En effet, lorsque le manipulateur se déplace vers la cible en évitant un obstacle, il est important de faire un compromis entre l'évitement de cet obstacle et la convergence vers la position de la cible. En prenant en considération ces contraintes, le tableau 3 présente les règles de décision du contrôleur de l'évitement d'obstacles en utilisant la première et la deuxième technique, respectivement.

F_Distance_error	***Angle_error_obs***	***Altitude_error_obs***		
		Negative	**Zero**	**Positive**
Far	**B_Left**	θ_1=Z, θ_2=SMU, θ_3=SMU	θ_1=Z, θ_2=SMU, θ_3=Z	θ_1=BML, θ_2=Z, θ_3=Z
	M_Left	θ_1=Z, θ_2=SMU, θ_3=SMU	θ_1=Z, θ_2=SMU, θ_3=Z	θ_1=MML, θ_2=Z, θ_3=Z
	S_Left	θ_1=Z, θ_2=SMU, θ_3=SMU	θ_1=Z, θ_2=SMU, θ_3=Z	θ_1=SML, θ_2=Z, θ_3=Z
	Z (*Cible à gauche*)	θ_1=Z, θ_2=SMU, θ_3=SMU	θ_1=Z, θ_2=SMU, θ_3=Z	θ_1=SML, θ_2=Z, θ_3=Z
	Z (*Cible à droite*)	θ_1=Z, θ_2=SMU, θ_3=SMU	θ_1=Z, θ_2=SMU, θ_3=Z	θ_1=SMR, θ_2=Z, θ_3=Z

	S_Right	θ_1=Z, θ_2= SMU, θ_3=SMU	θ_1=Z, θ_2=SMU, θ_3=Z	θ_1=MMR, θ_2=Z, θ_3=Z
	M_Right	θ_1=Z, θ_2=SMU, θ_3=SMU	θ_1=Z, θ_2=SMU, θ_3=Z	θ_1=MMR, θ_2=Z, θ_3=Z
	B_Right	θ_1=Z, θ_2=SMU, θ_3=SMU	θ_1=Z, θ_2=SMU, θ_3=Z	θ_1=BMR, θ_2=Z, θ_3=Z
Near	**B_Left**	θ_1=Z, θ_2=SMU, θ_3= SMU	θ_1=Z, θ_2=SMU, θ_3=Z	θ_1=BML, θ_2=Z, θ_3=Z
	M_Left	θ_1=Z, θ_2=SMU, θ_3= SMU	θ_1=Z, θ_2=SMU, θ_3=Z	θ_1=MML, θ_2=Z, θ_3=Z
	S_Left	θ_1=Z, θ_2=SMU, θ_3=SMU	θ_1=Z, θ_2=SMU, θ_3=Z	θ_1=SML, θ_2=Z, θ_3=Z
	Z (*Cible à gauche*)	θ_1=Z, θ_2=SMU, θ_3= SMU	θ_1=Z, θ_2=SMU, θ_3=Z	θ_1=SML, θ_2=Z, θ_3=Z
	Z (*Cible à droite*)	θ_1=Z, θ_2=SMU, θ_3=SMU,	θ_1=Z, θ_2=SMU, θ_3=Z	θ_1=SMR, θ_2=Z, θ_3=Z
	S_Right	θ_1=Z, θ_2=V, θ_3= SMU	θ_1=Z, θ_2=SMU, θ_3=Z	θ_1=MMR, θ_2=Z, θ_3=Z
	M_Right	θ_1=Z, θ_2=SMU, θ_3= SMU	θ_1=Z, θ_2=SMU, θ_3=Z	θ_1=MMR, θ_2=Z, θ_3=Z
	B_Right	θ_1=Z, θ_2=SMU, θ_3= SMU	θ_1=Z, θ_2=SMU, θ_3=Z	θ_1=BMR, θ_2=Z, θ_3=Z
Z	**B_Left**	θ_1=Z, θ_2=SMU, θ_3=SMD	θ_1=Z, θ_2=SMU, θ_3= SMD	θ_1=BML, θ_2=Z, θ_3=Z
	M_Left	θ_1=Z, θ_2=SMU, θ_3=SMD	θ_1=Z, θ_2=SMU, θ_3= SMD	θ_1=MML, θ_2=Z, θ_3=Z
	S_Left	θ_1=Z, θ_2=SMU, θ_3= SMU	θ_1=Z, θ_2=SMU, θ_3= SMD	θ_1=SML, θ_2=Z, θ_3=Z
	Z (*Cible à gauche*)	θ_1=Z, θ_2=SMU, θ_3=SMD	θ_1=Z, θ_2=SMU, θ_3= SMD	θ_1=SML, θ_2=Z, θ_3=Z
	Z (*Cible à droite*)	θ_1=Z, θ_2=SMU, θ_3=SMD	θ_1=Z, θ_2=SMU, θ_3= SMD	θ_1=SMR, θ_2=Z, θ_3=Z
	S_Right	θ_1=Z, θ_2=SMU, θ_3=SMD	θ_1=Z, θ_2=SMU, θ_3= SMD	θ_1=MMR, θ_2=Z, θ_3=Z
	M_Right	θ_1=Z, θ_2=SMU, θ_3=SMD	θ_1=Z, θ_2=SMU, θ_3= SMD	θ_1=MMR, θ_2=Z, θ_3=Z
	B_Right	θ_1=Z, θ_2=SMU, θ_3=SMD	θ_1=Z, θ_2=SMU, θ_3= SMD	θ_1=BMR, θ_2=Z, θ_3=Z

Tableau 3 : Règles floues des techniques d'évitement pour un robot manipulateur à 3ddl.

III.4. Exploitation des images délivrées par la caméra Kinect

Cette deuxième partie décrit l'exploitation des images délivrées par la caméra Kinect pour le calcul des coordonnées cartésiennes des segments constituants le robot manipulateur :

- Au début, la caméra Kinect délivre une image du robot manipulateur à 3ddl (Figure 22).

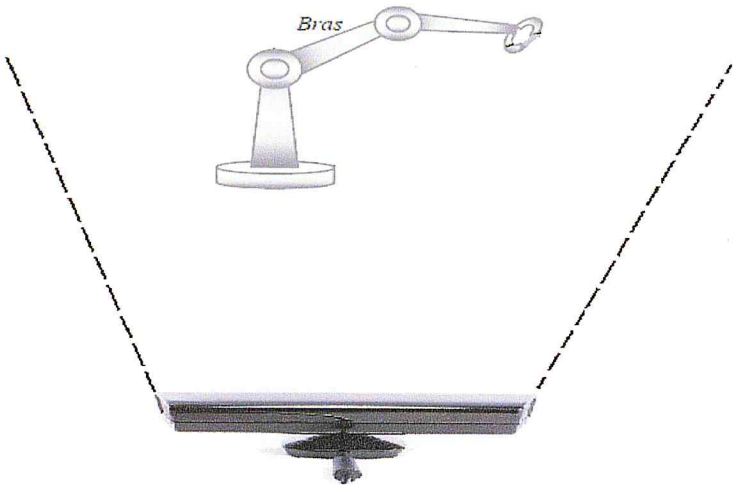


Figure 22 : Une image délivrée par la caméra Kinect pour un robot manipulateur à 3 ddl.

- À l'acquisition de cette image du bras manipulateur, une couleur différente est associée à chaque articulation. À cet effet et comme le robot a 3 ddl, on aura quatre couleurs différentes ; la quatrième couleur est associée à l'effecteur (figure 23).

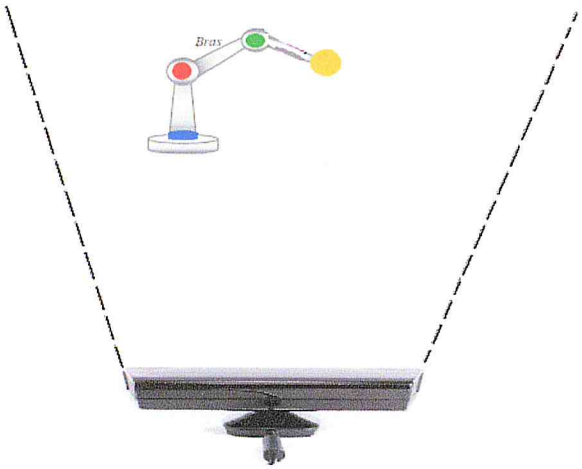


Figure 23 : Image du bras manipulateur avec les couleurs associées aux articulations.

- Par la suite, les centres de chacune des couleurs sont calculés et les coordonnées de chaque point (x, y, z)sont localisées en utilisant la caméra Kinect sur un repère à trois dimensions. Après, les distances entre tous les points couleur sont mesurées pour obtenir les configurations initiales du robot manipulateur (figure 24).

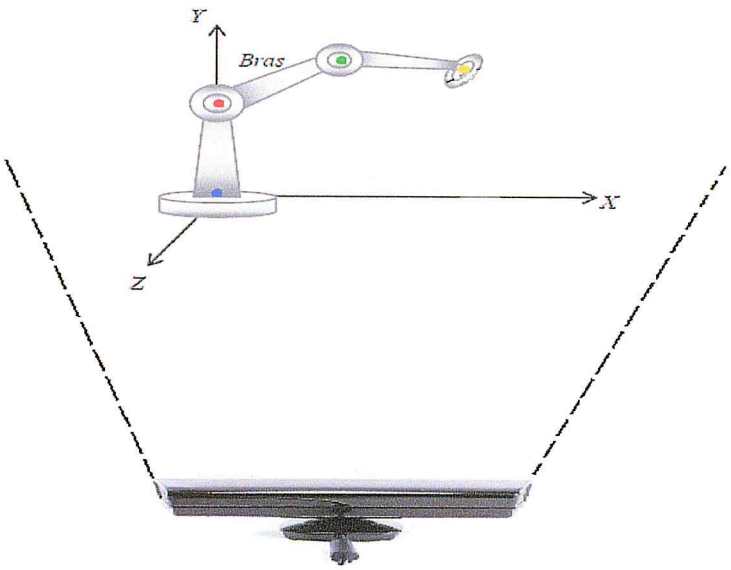


Figure 24 : Centre et coordonnées délivrées par la caméra Kinect.

- La caméra Kinect est utilisé aussi pour la localisation des obstacles (dans ce travail, bous avons considéré des obstacles de forme cylindrique). Un cylindre est caractérisé par un rayon R et une hauteur H(figure 25).

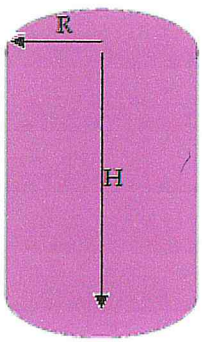


Figure 25 : Représentation d'un obstacle de forme cylindrique avec son rayon et sa hauteur.

- À la fin, plusieurs angles nécessaires au calcul de la trajectoire et l'évitement des obstacles peuvent être calculés (figure 26).

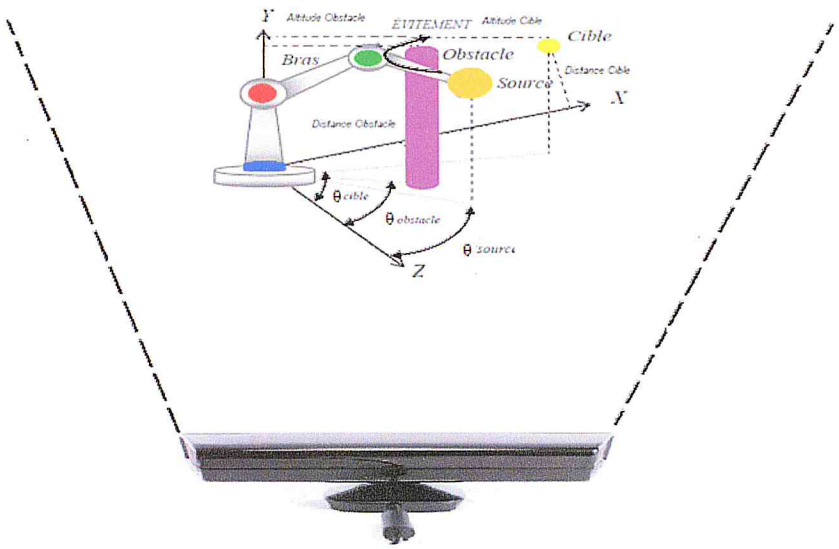


Figure 26 : Angles nécessaires pour l'évitement d'un obstacle cylindrique par le robot.

III.4.1. Kinect en robotique

La Kinect en robotique permet de doter les robots d'une capacité de reconnaissance de la présente d'humain dans son environnement, d'interpréter ses gestes et, d'interpréter son expression faciale et de faire de la reconnaissance vocale. Le capteur Microsoft Kinect (ou XBOX Kinect) est donc un capteur de choix pour les robots [51].Par exemple :

- Les chercheurs de Willow Garage ont utilisé la Kinect pour contrôler le robot PR2 et lui fournir de meilleures capacités de SLAM (Localisation et cartographie simultanées) [52].
- V-Sido est un logiciel de contrôle permettant de piloter un robot humanoïde en temps réel. En le couplant avec une caméra Kinect, il est possible de piloter les déplacements du robot directement avec son corps. Cela permettait une bonne immersion dans le personnage, que si on restait devant son clavier et son écran. Le système Kinect rajoute une nouvelle dimension, où le robot reproduit les moindres faits et gestes [53].

III.5.Conclusion

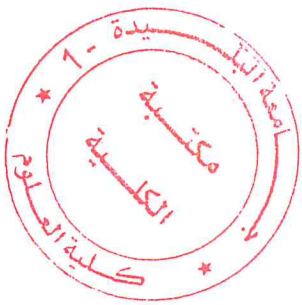
Dans ce chapitre, nous nous sommes intéressés au développement d'une approche basée sur la logique floue pour le contrôle des robots manipulateurs. Nous avons vu, au cours de ce

chapitre, l'intérêt d'une approche de contrôle tout en mettant l'accent sur les approches basées sur la logique floue.

La première partie, nous avons présenté la structure générale d'un système de contrôle flou et nous avons détaillé le contrôleur proposé en décrivant son architecture fonctionnelle ainsi que le fonctionnement de ses deux unités (Contrôleur flou de déplacements libres et Contrôleur flou de l'évitement d'obstacles).

Quant à la deuxième partie de ce chapitre, elle est dédiée à la description de l'exploitation des images délivrées par la caméra Kinect pour le calcul des coordonnées cartésiennes des segments constituants le robot manipulateur.

Notre contrôleur a été testé sur différents scénarios de simulation utilisant un robot manipulateur à 3 ddl.



Chapitre IV
Implémentation

IV.1. Introduction

Après avoir présenté l'approche proposée pour la planification de trajectoires sans collision et décrit l'exploitation des images délivrées par la caméra Kinect pour le calcul des coordonnées cartésiennes des segments constituants les robots manipulateurs dans le chapitre précèdent, nous allons présenter dans ce dernier chapitre l'implémentation et les résultats du travail réalisé, en commençant par définir les outils de développement utilisés dans cette application, suivis par les fenêtres de l'application.

Cette application a été conçue avec le logiciel Éclipse (Java) ainsi que la bibliothèque UFDW (Java Library) nécessaire à l'utilisation de la caméra Kinect.

IV.2.Outils matériels

IV.2.1.Caméra Kinect

La Kinect est une caméra 3D créée initialement par Microsoft pour l'industrie des jeux vidéo. Elle permet à l'utilisateur de contrôler le jeu sans utiliser de manette mais uniquement grâce aux mouvements de son corps [54].Devant l'étendue de possibilités offertes par la Kinect, des logiciels de développement « open source » ont été créées pour manipuler la Kinect par ordinateur et créer des applications utilisant cette caméra.

Figure 27 : CaméraKinect*XBOX360*

IV.2.1.1.Caractéristiques de la Kinect

Les principales caractéristiques de la caméra Kinect sont données dans ce qui suit [54] :

- **Capteurs** : la Kinect a une caméra couleur et capteur de profondeur. Elle dispose aussi d'un micro à reconnaissance vocale.
- **Motorisation** : il s'agit d'ensemble motorisé pour suivre les déplacements. L'inclinaison est ±27 degrés.
- **Champ de vision** : la caméra a un champ de vision horizontal de 57 degrés, un champ vertical de 43 degrés et une portée de 0.8m à 4.0m.
- **Flux de données** : le flux infrarouge est de 640x480 à 30 images par seconde. Aussi, le flux RVB (Rouge, Vert, Bleu) est 640x480 à 30 images par seconde. Enfin, le flux audio est de 16 bits à 16KHz.
- **Système de reconnaissance physique** : la caméra peut reconnaître jusqu'à 6 personnes et 2 joueurs actifs (4 joueurs actifs avec le SDK 1.0).La Kinect peut aussi reconnaître 20 articulations par squelette.

IV.2.1.2.Fonctionnement de la Kinect

Comme le montre la figure 28, la Kinect est composée de plusieurs capteurs audio, d'une caméra RVB, d'un projecteur infrarouge, d'une caméra infrarouge et d'un pied motorisé [54].

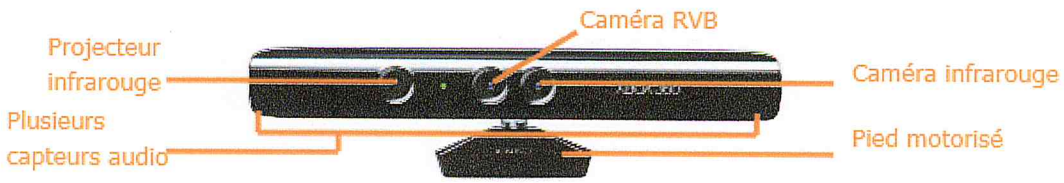


Figure 28 : Composants de la Kinect [54].

IV.2.1.2.1.Caméra couleur (RVB)

C'est la fonctionnalité la plus basique disponible sur la Kinect. Elle est utilisée pour récupérer une image et l'envoyer au format YUY2. Il est possible de convertir les données dans un autre format beaucoup plus léger et facile à traiter pour un processeur [55].

IV.2.1.2.2.Caméra de profondeur

Cette caméra est utilisée pour récupérer les mouvements d'un utilisateur dans l'espace. Elle est composée d'une caméra infrarouge photosensible. La caméra de profondeur fonctionne en analysant un modèle de tacheture de lumière laser infrarouge envoyée par un deuxième composant [56]. Elle est également capable de traquer les indexes du corps au pixel correspondant, déterminé par un algorithme propriétaire de détection du squelette.

IV.2.1.2.3.Caméra infrarouge

La caméra infrarouge utilisée par la caméra de profondeur peut également être utilisée indépendamment afin de produire des images en noir et blanc. La caméra infrarouge est un élément actif de la Kinect qui imite une source de lumière infrarouge afin d'illuminer l'environnement. La caméra reflète la lumière de la même façon qu'une caméra classique. Cette caméra permet une vue claire dans un environnement avec très peu de lumière [57].

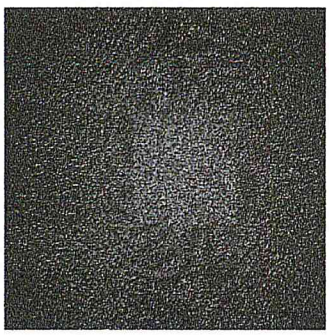


Figure 29 : Mire émise par la Kinect telle que perçue par la caméra infrarouge [57].

IV.2.1.2.4.Flux de la Kinect

Grâce aux composants précédents, la Kinect fournit trois flux différents [54] :

- **Flux audio** : il provient du capteur audio et permet la fonctionnalité de reconnaissance vocale.
- **Flux vidéo** : le flux vidéo provient de la caméra RVB qui fonctionne comme toutes les autres caméras 2D (Figure 30). Ce flux permet de fournir les images couleurs captées par la Kinect.

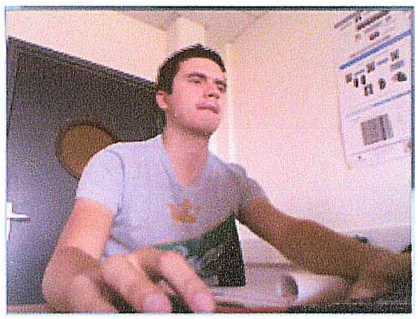


Figure 30 : Image couleur fournie par la Kinect [54].

- **Flux de profondeur :** le flux de profondeur est le plus important ; il fait l'originalité de la Kinect en la différenciant des caméras vidéo habituelles. Ce flux fournit une carte de profondeur donnant pour chaque pixel la distance dans l'espace 3D entre l'objet représenté par ce pixel et la caméra (Figure 31). On peut facilement transformer la carte de profondeur en images de profondeur en niveau de gris pour visualiser la profondeur.

Figure 31 : Image de profondeur fournie par la Kinect [54].

IV.2.1.2.5.Branchement et alimentation de la Kinect

La Kinect comporte un câble double USB/prise de courant. La prise USB permet de se connecter au PC qui doit traiter l'information fournie par la Kinect. La prise de courant permet d'alimenter le capteur Kinect [51].

IV.3.Outils de programmation

Notre travail devait être à la base sous Ubuntu ; nous avons essayé plusieurs versions de ce dernier mais nous avons rencontré des problèmes avec chacune : soit la Kinect fonctionne et la bibliothèque de logique floue (c.-à-d., fuzzylite) ne marche pas ; soit le contraire. À le fin, nous avons donc opté pour Windows et travaillé sur Éclipse (cependant, Java propose quelques fonctionnalités de la Kinect mais pas toutes).

IV.3.1.Logiciel Éclipse

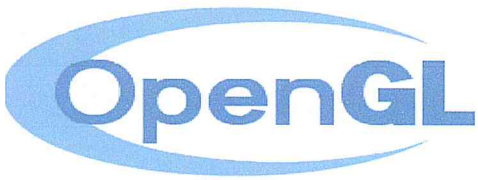
Éclipse est un IDE développé par IBM, *IntegratedDevelopmentEnvironment*, c'est-à-dire un logiciel qui simplifie la programmation en proposant un certain nombre de raccourcis et d'aide à la programmation. Il est gratuit et disponible pour la plupart des systèmes d'exploitation.

Au fur et à mesure, Éclipse compile automatiquement le code introduit, en soulignant en rouge ou jaune les problème qu'il décèle. Il souligne en rouge les parties du programme qui ne compilent pas, et en jaune les parties qui compilent mais peuvent éventuellement poser problème. Pendant l'écriture du code, cela peut sembler un peu déroutant au début, puisque tant que la ligne de code n'est pas terminée (en gros jusqu'au point-virgule), Éclipse indique une erreur dans le code. Il est déconseillé de continuer d'écrire le programme quand il contient des erreurs, car Éclipse est dans ce cas moins performant [58].

IV.3.2.Bibliothèque UFDW

La bibliothèque UFDW est une bibliothèque Java open source qui contient plusieurs classes Java pour créer des applications Java autonomes ou des applets Java avec une interface utilisateur graphique et un rendu 3D dans openGL à l'aide de la bibliothèque JOGL Java de JogAmp. De plus, la bibliothèque UFDW inclut la bibliothèque J4K, qui offre un moyen pratique d'utiliser le Kinect SDK de Microsoft en Java [59].JOGL héberge la version de développement de la liaison Java pour l'API OpenGL, et est conçu pour fournir des graphiques 3D supportés par le matériel aux applications écrites en Java [59].

La bibliothèque J4K est une bibliothèque Java open source populaire qui implémente une liaison Java pour le kit de développement logiciel Kinect de Microsoft. Il communique avec une bibliothèque Windows native, qui gère les flux de profondeur, de couleur, d'infrarouge et de squelette de la Kinect à l'aide de l'Interface Java Native (JNI). La bibliothèque J4K est compatible avec tous les périphériques Kinect (Kinect pour Windows, Kinect pour XBOX, ou Kinect 2). Elle permet aussi de contrôler plusieurs capteurs de n'importe quel type à partir d'une seule application, tant que les capacités du système le permettent. De plus, la bibliothèque J4K contient plusieurs classes Java pratiques qui convertissent les trames de profondeur compactées, les trames squelette et les trames de couleur reçues par un capteur Kinect en objets Java [60].



IV.4.Présentation de l'application

Dans cette partie, nous validons les différentes tâches implémentées dans le cadre de ce projet « application » comme suit :

IV.4.1.Fenêtre principale

La fenêtre principale de l'application développée est composée de (Figure 32) :

- six cases à coucher (Check Box),
- deux boutons : un pour éteindre la camera Kinect et l'autre pour faire les calculs,
- le flux vidéo diffusé par la caméra Kinect.

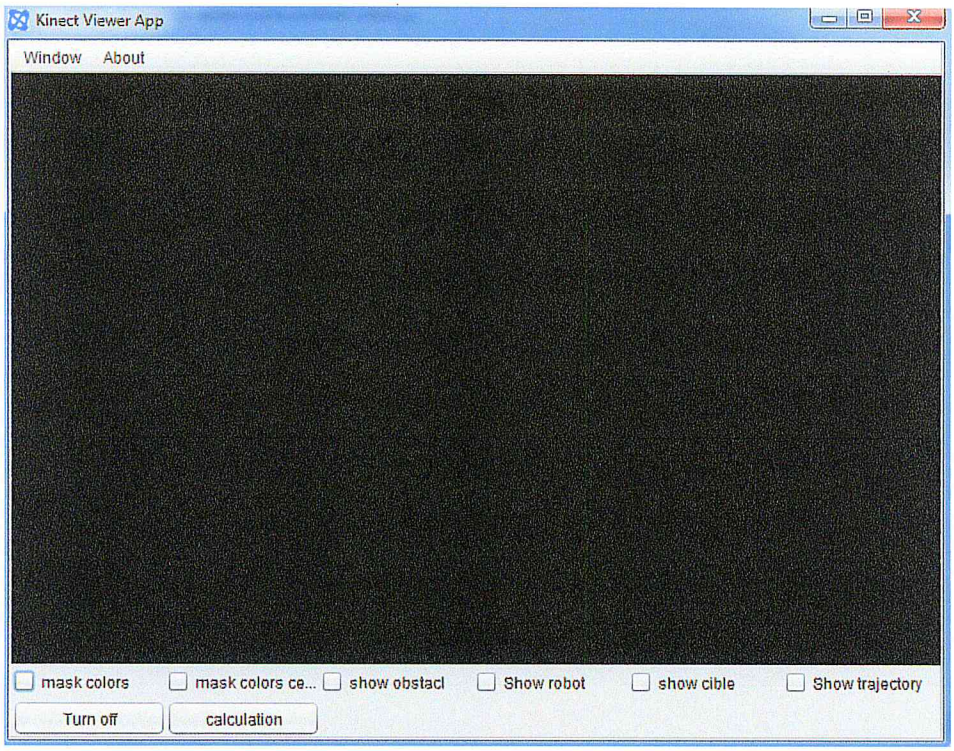


Figure 32 : Composantes de la fenêtre principale de l'application

IV.4.1.1. Cases à cocher

- **Maskcolors :** cette case permet de masquer toute l'image sauf les couleurs assignées aux articulations(qui sont nécessaire à notre application).
- **Maskcolors center :** cette case permet de masque toute l'image sauf les pixels des centres de chaque couleur.

- **Show obstacle :** cette case permet de masquer toute l'image sauf les obstacles (représentés par un rectangle au lieu d'un cylindre).
- **Show Robot :** cette case permet de tracer les liens entre les centres des couleurs pour construire des segments.
- **Show cible :** cette case permet de masquer toute l'image sauf la cible (qui est représentée par un rectangle).
- **Show trajectory :** cette case permet d'exécuter l'approche de la logique floue proposée pour planifier la trajectoire et éviter les obstacles.

IV.4.1.2. Bouton « Calculation »

Ce bouton permet de calculer les thêtas (Obstacle, Cible, Effecteur) et les angles de robots (Q_1, Q_2, Q_3) et les altitudes (Obstacle, Cible, Effecteur) ainsi que les paramètres d'entier (Distance_Target, Error_Altitude, Error_Angular, Treget_direction, Distance_Obstacle, Error_Altitude_ Obstacle, Error_Angular_ Obstacle) .Tous ces calculs sont affichés dans une nouvelle fenêtre (Figure 33).Cette fenêtre est composée aussi d'un bouton « Detect Obstacle » qui permet de détecter si un obstacle existe entre la cible et l'effecteur.

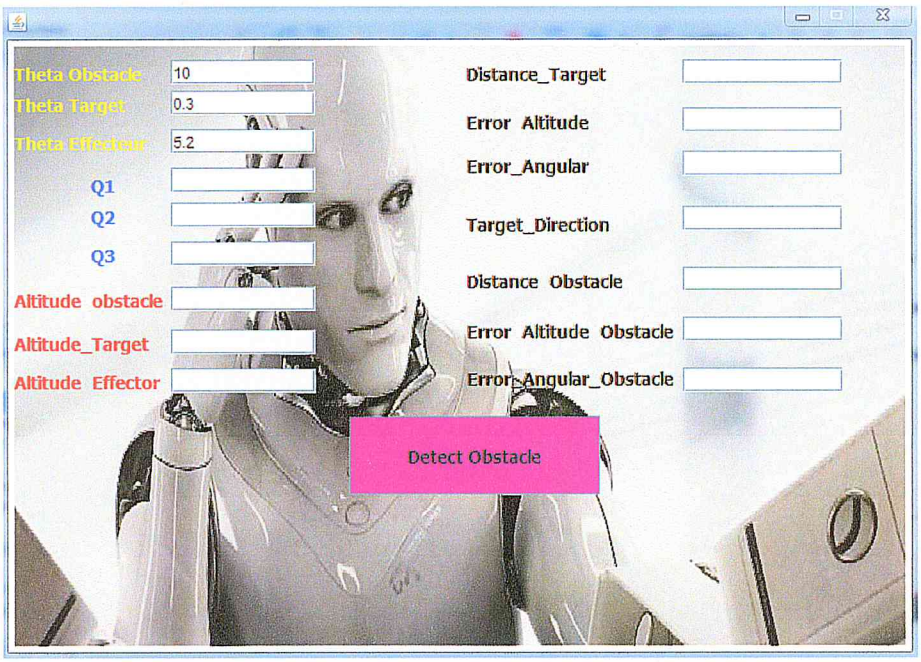


Figure 33: Fenêtre de calculs et de détection de présence d'obstacle.

IV.4.1.3. Bouton « Detect Obstacle »

1. Si $(\theta_{Effeteur} < \theta_{Ob} < \theta_{Cible})$ ou $(\theta_{Effecteur} > \theta_{Ob} > \theta_{Cible})$, alors il un obstacle est détecté entre l'effecteur et la cible ; dans ce cas, l'unité de « Obstacle Avoidance » va être déclenchée (Figure 34).

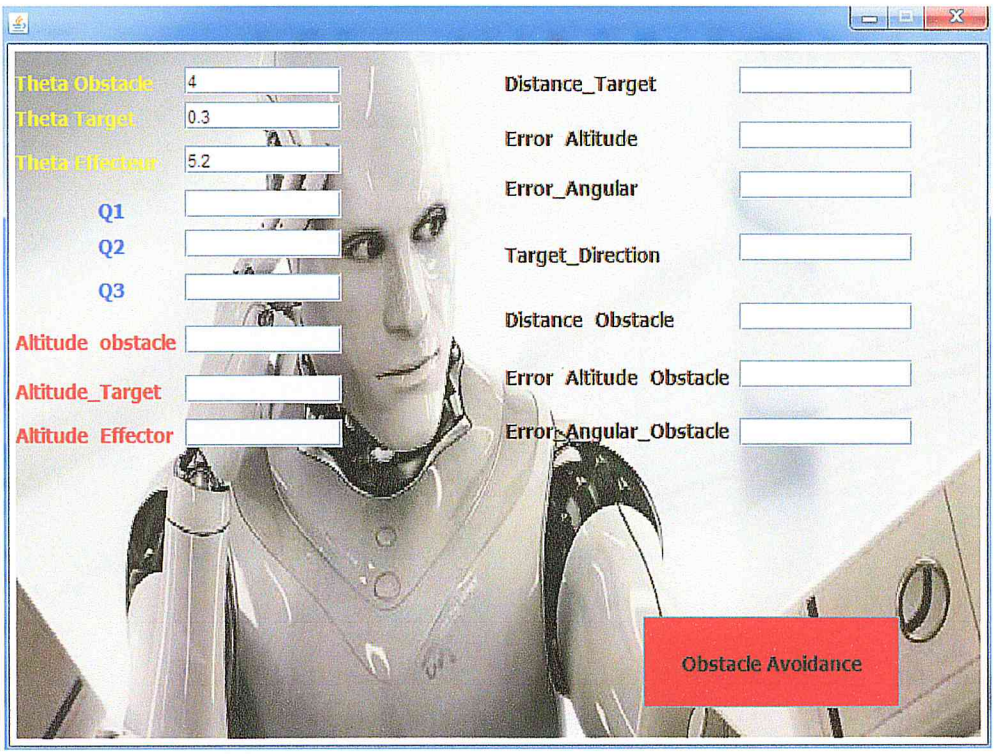


Figure 34: Fenêtre de calcul et la détection d'obstacle (« Obstacle Avoidance »).

2. Dans le cas contraire (aucun obstacle n'est détecté), l'unité « Free Movement » va être exécutée (Figure 35).

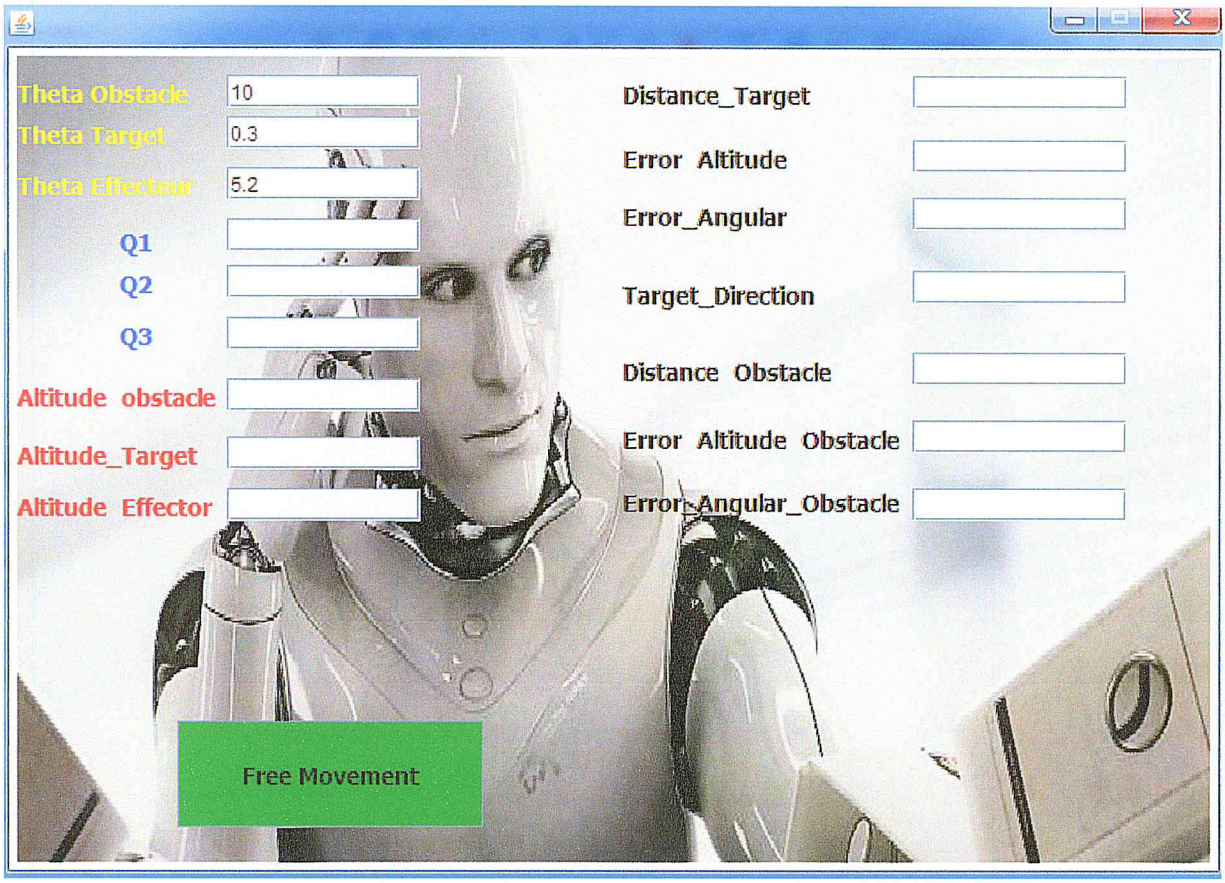


Figure 36: Fenêtre de calcul et la détection d'obstacle (« Free Movement »).

Conclusion :

Dans ce chapitre, nous avons présenté les outils matériel (la kinect) et programmations (logiciel éclipse et les bibliothèques) , ainsi que l'implémentation de notre application.

Conclusion générale

Les robots grâce à leurs domaines d'applications très variés, occupent un champ de recherche prédominant. Dans le cas des robots manipulateurs, le concept de planification de trajectoires et d'évitement d'obstacles est très important, parce qu'il contribue largement à réussir la mission pour laquelle ce type de robots a été conçu ; c'est la raison pour laquelle on trouve beaucoup de contributions et de recherches qui proposent des solutions différentes à cette problématique.

Les différentes propositions sont basées sur des méthodes issues de l'intelligence artificielle, et peuvent être classées en tant que méthodes globales, locales ou mixtes. Les approches globales ont besoin d'une connaissance complète de l'environnement. Ces méthodes cherchent toutes les trajectoires possibles. Ensuite, elles sélectionnent l'une d'entre elles selon un ensemble de critères. Ces approches assurent l'existence ou non d'une solution.

Les approches locales n'ont besoin que d'une connaissance partielle de l'espace de travail. Elles cherchent une trajectoire au fur et à mesure que le robot évolue dans l'environnement. Toutefois, même si une trajectoire sans collision existe, les méthodes locales peuvent aboutir à une situation de blocage.

Les approches mixtes sont une combinaison des deux approches précédentes. Elles opèrent en deux phases. La première phase est similaire aux approches globales. Elle permet de générer une trajectoire de référence réalisée hors ligne. La seconde phase est similaire aux approches locales. Elle adapte localement la trajectoire du robot à partir de la trajectoire de référence.

Pour aller au-delà et trouver une nouvelle façon de résoudre cette problématique, on a essayé l'approche de logique floue aussi une caméra Kinect pour détecter les obstacles et la position de robot initial et actuelle.

Perspectives :

- Appliquer la solution proposée sur un matériel réel.
- Utiliser une autre version de la Kinect pour des calculs plus précis.

Références bibliographiques

[1] P. Fisette, H. Buyse, J.C. Samin, MECA 2732 : "*Introduction à la Robotique*", (10 novembre 2004.

[2] A. Benali,"*Robotique et Automatisation Industriel*". 2004.

[3] PRIEL MARC "l*es robots industriels: caractéristiques, performances et choix* ": Edition AFNOR (1990) .

[4] Belkhouche Ismail . Mémoire de fin d'études pour l'obtention du diplôme de Master " *Planification de la trajectoire et évitement d'obstacle par les réseaux de neurones pour les robots mobile autonome* "(2010-2011).

[5] http://www.directindustry.fr/fabricant-industriel/bras-manipulateur-99148.html (30.05.2018).

[6] Amira Islam. « Développement d'un robot joueur d'échecs ». Académie militaire de Tunisie et école royale militaire de Belgique. (2011).

[7] Hentout.A "planifcation de trajectories sans collision pour robots manipilateurs " ecole militaire polytechnique (2004, 01,20).

[8] Maxim Vaidis " *conception réalisation et étude d'un essaim de robots au autonome protègent un groupe de personnes munies de semelle intelligent* " présenter àl'université du Québec à Chicoutimi comme exigence partielle de la maitrise en ingénierie en (décembre2017).

[9] Soyez frédéric, "*le planificateur évolué pour la robotique mobile intelligent* P .E.R.M.I''

[10] Khatib, O., "*Real-time obstacle avoidance for manipulators and mobile robots*", inRobotics and Automation, Proceedings, IEEE International Conference on, volume 2, pages500-505, 1985.

[11] https://www.calerga.com/products/Sysquake/robotnav.fr.html(06.06.2018).

[12] Fox, D., Burgard, W., and Thrun, S., "*The dynamicwindowapproach to collisionAvoidance*", IEEE Journal onRobotics and Automation, 4, 1997.

[13] Minguez, J. and Montano, L., "*Nearnessdiagram navigation (nd): a new real-timeCollision avoidanceapproach*", In Intelligent Robots and Systems (IROS

2000),Proceedings, 2000 IEEE/RSJ International Conference on, volume 3, pages 2094-2100vol.3, 2000.

[14] https://nl.wikipedia.org/wiki/Vector_Field_Histogram(07.06.2018).

[15] Yang, Y. and Brock, O., "*Elasticroadmaps:Globallytask-consistent motion forAutonomous mobile manipulation in dynamicenvironments*", In Robotics : Science andSystems II, August 16-19, 2006.

[16] Lamiraux, F., Bonnafous, D., and Lefebvre, O.,"*Reactivepathdeformation forNon-holonomic mobile robots*", Robotics, IEEE Transactions on, 20(6):967-977, 2004.

[17] Kurniawati, H. and Fraichard, T., "*Frompath to trajectorydeformation*", In IntelligentRobots and Systems, IROS 2007. IEEE/RSJ International Conference on, pages 159-164, 2007.

[18] Delsart, V. and Fraichard, T., "*NavigatingdynamicenvironmentsusingtrajectoryDeformation*", in Intelligent Robots and Systems, IROS 2008. IEEE/RSJ InternationalConference on, pages 226-233, 2008.

[19] Zhang, P.-Y., Lu, T.-S., and Song, L.-B., "*Soccer robot path planning based on the artificialpotentialfieldapproachwithsimulatedannealing*", Robotica, 22 :563-566, 2004.

[20] Montiel, O., Orozco-Rosas, U., and Sepúlveda, R., "*Path planning for mobile robots usingbacterialpotentialfield for avoidingstatic and dynamic obstacles*", Expert SystemswithApplications, 42(12):5177-5191, 2015.

[21] Merchán-Cruz.E. A, and Morris. A. S, Fuzzy-GA-basedtrajectoryplanner for robot manipulators sharing a commonworkspace. IEEE Transactions on Robotics, 22(4), 613–624(2006).

[22] Wu. X. J, Li. Q, and Heng. K. H, Development of a generalmanipulatorpathplannerusingfuzzyreasoning. IndustrialRobot: An International Journal, 32(3), 248–258(2005).

[23] Zavlangas. P. G, Tzafestas. S. G, Industrial robot navigation and obstacle avoidanceemployingfuzzylogic. Journal of Intelligent and RoboticSystems, 27(1), 85–97 (2000).

[24] Fu. Y. L, Jin. B, Li. H, and Wang. S. G, A robot fuzzy motion planning approach in unknownenvironments. Frontiers of Mechanical Engineering in China, 1(3), 336-340 (2006).

[25] Salah. K, Modélisation et commande d'un robot par méthodes intelligentes. Thèse de doctorat, Université Badji Mokhtar de Annaba, Algérie(2006).

[26] T .Lozano-Pérez, M.Wesley *"An Algorithm for planning collision-free Pathsamongpolyhedral Obstacle"*, communication of the ACM, Vol .22, No: 10,560-570,October 1979.

[27] HelguraArellanochiristin, *"Contribution àla résolution du problème de minima locaux dans la méthode de la planification de trajectoires sans collusion pour robots manipulateurs*'', Thèse de doctorat université de Poitiers, 2001.

[28] http://www.dil.univ-mrs.fr/~gcolas/algo-licence/slides/BF-slides.pdf. (2018, Avril 16).

[29] https://www.normalesup.org/~dconduche/informatique/PT/Cours/Dijkstra.pdf. (2018, Avril 16).

[30] Cakir, M. ,"*2d path planning of uavswithgeneticalgorithm in a constrainedenvironment*",In Modeling, Simulation, and AppliedOptimization (ICMSAO), 6th International.

[31] https://www.researchgate.net/figure/278645627_fig9_Figure-3-4-Organigramme-d%27un-algorithme-genetique-Vosniakov-et-al-2010Conference, pages 1-5, 2015

[32] Gaillard et al., *"DeterministicKinodynamic Planning"*, (Workshop of the UK Planning and SchedulingSpecialInterest Group, Italy)2010.

[33] Versino , C. and Gambardella , L ,M " *learning fine motion by using the hierarchicalextendedKohonenMap*" In von der Malsburg , C , Von seelen , W. Vorbruggen , J.C , and sendroft ,B (Eds) , proc . ICANN96, International conference on artificial neural network, Bochum, Germany, 17-19 July, Vol, 1112 of lectures votes in computer science Springer-Verlag, Berlin, pp 211-226-1996.

[34] Versino, C. and Gambardella,L, M, *"learning fine motion in robotics ": Design and Experiments*", IDSIA, CorsoElvesia 36. 6900 Lugano, Switzerland, by CRC press LLC, 2000.

[35] Jérôme Barraquand « *Robot Motion Planning: A Distributed Representation Approach* » (1991, December 1).

[36] D. Hsu, J.C. Latombe, and R. Motwani. « *Path Planning in Expansive Configuration Spaces* ». Proc. IEEE Int. Conf. on Robotics and Automation, 1997. A revised version of

this paper appeared in International Journal of Computational Geometry and Applications, 9(4-5):495-512, 1999.

[37] Maoudj Abderraouf, "Téléopération multi-robots hétérogènes :Application au cas des robots RobuTER/ULM et AGVM6", DOCTORAT 3emeCycle en" Électronique", option "Contrôle de Processus et renduinteractif", USTHB, Février 2018.

[38] X. Yang, M. Moallem and R-V. Patel, "*A layeredgoalorientedfuzzy motion planning strategy for mobile robot navigation*", IEEE transactions on systems, man, and cybernetics-part B : cybernetics, Vol. 35, No. 6, pp. 1214-1224, December 2005.

[39] P-G.Zavlangas, S-G.Tzafestas and K. Althoefer, "*Fuzzy obstacle avoidance and navigation for omnidirectional mobile robots*", ESIT 2000, pp. 375382, Aachen, Germany, 2000.

[40] E. Dombre et W. Khalil, "*Modélisation, Identification et Commande des Robots*", Hermès, deuxième édition, 1999.

[41] E. Colle, K. NaitChabane, S. Delarue, P. Hoppenot, « *ARPH : Comparaison d'une méthode classique et d'une méthode utilisant la coopération hommemachine pour exploiter la redondance de l'assistant robotisé* », LSC CNRS FRE 2494, Université d'Evry, 2006

[42] A. Fatmi, A. Al Yahmadi, L. Khriji and N. Masmoudi, "*A fuzzylogicbased navigation of a mobile robot*", Proceedings of World Academy of Science, Engineering and Technology, pp. 169-175,Vol. 15, ISSN 1307- 6884, 2006.

[43] K. Chafaa, "*Structure d'identification et de commande des systèmes non linéaires*", Thèse Doctorat, Université de Batna, Algérie, 2006.

[44] M. Mordjaoui, "*Modélisation des effets électromagnétiques - Apport de la logique floue et neuro-floue*", Thèse Doctorat, Université de Batna, Algérie, 2008.

[45] François CHEVRIE and François GUÉLY Cahier Technique n° 191« *La logique floue*», CT 191 édition mars 1998.

[46] Zadeh. L. A, Fuzzy sets. Information and Control, 8, pp. 338–353(1965).

[47] Mamdani. E, Application of fuzzyalgorithms for control of simple dynamics plant. proceedings of the institution of Electrical Engineers, Control and Science, vol. 121, pp. 1585-1588, 1974.

[48] Guenounou. O .Méthodologie de conception de contrôleurs intelligents par l'approche génétique: application à un bioprocédé. Thèse de Doctorat. Université de Toulouse, France., (2009).

[49] E.. COX, «*The fuzzy systems handbook*». Academic press, 1994

[50] Maoudj.A, Hentout.A, Bouzouia.B, and Toumi R., On-line fault-tolerantfuzzy-basedpath planning and obstacles avoidanceapproach for manipulator robots. International Journal of Uncertainty, Fuzziness and Knowledge-BasedSystems.(Acceptedpaper).(2017)

[51] https://www.generationrobots.com/fr/401430-capteur-microsoft-kinect.html (2018,Avril 29).

[52] https://www.robotshop.com/blog/fr/kinect-robots-redoutables-841 (2018, mai, 10).

[53] http://www.semageek.com/vsido-contrler-robot-humanode-laide-dun-kinect/ (16 janvier 2011).

[54] Maxime DEVANNE ''Modélisation 3D d'un corps humain à partir de caméras 3D Kinect ''. (Promotion FI 2012).

[55] https://connect.microsoft.com (2013,Novembre 18).

[56] https://www.google.com/patents/US20080106746 (2018, May 8) .

[57] http://www.irinfo.org/articles/03_01_2007_grossman.html (2018, March 1).

[58] http://www.enseignement.polytechnique.fr/informatique/profs/Julien.Cervelle/eclipse/ (2018, Juin 10).

[59] http://jogamp.org/jogl/www/ (2018, Juin 10).

[60] A. Barmpoutis. "TensorBody: Real-time Reconstruction of the Human Bodyand Avatar Synthesisfrom RGB-D", IEEE Transactions on Cybernetics, Vol. 43(5), Pages: 1347-1356,(2013 October) .