REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE SAAD DAHLEB DE BLIDA
FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE

MEMOIRE DE FIN D'ETUDE

Présenté par :

BOUMAHAMMED Mounira

FELFOUL Mustapha Younes

EN VUE D'OBTENIR LE DIPLOME DE MASTER

SPECIALITE : Systèmes Informatiques et Réseaux

Thème :

Contrôle d'une équipe de robots homogènes en formation Leader/Follower.

Soutenu le : 03-07-2018

Mme GHRIBI	Président
Mr DOUGA	Examinateur
Mr BENYAHIA.M	Promoteur
Mr HENTOUT.A	Encadreur

Année universitaire : 2017/2018

Nous remercions le bon Dieu de nous avoir accordé la force pour terminer ce modeste travail et de nous avoir donné la patience et le courage de concrétiser nos études.

Et nos très chers parents pour leur tendresse, leurs encouragements et leur soutien moral et matériel dans le but d'assurer notre réussite.

Remerciements

Que tous ceux, de près ou de loin, ont contribué, par leurs conseils, leurs encouragements ou leur amitié, à l'aboutissement de ce travail, trouvent ici l'expression de notre profonde reconnaissance.

Nos vifs remerciements, accompagnés de toute notre gratitude, vont à notre encadreur Monsieur HENTOUT, pour son aide, sa patience, et ses conseils utiles, durant la réalisation de notre travail.

Nos sincères remerciements s'adressent aussi à notre promoteur Monsieur BENYAHIA, pour son aide, ses avis éclairés et pour tout le savoir qu'il nous a transmis.

Notre reconnaissance va particulièrement à Monsieur KAZED, le chef du département d'électronique et Monsieur KAMECHE, pour leurs conseils utiles qu'ils n'ont cessé de nous prodiguer, les orientations et l'intérêt qu'ils nous ont montré durant la progression de ce travail.

Nous n'oublions pas d'adresser un grand merci à tous les enseignants, qui ont contribuées de près et de loin à l'enrichissement et à notre épanouissement intellectuel durant tout ce parcours universitaire.

Ce travail n'aurait sans doute pu voir le jour sans le soutien de nos proches et de nos amis que nous tenons affectueusement à remercier.

ملخص :

الهدف من هذا العمل هو السيطرة على فريق من الروبوتات المتنقلة المتجانسة في القائد / المتابع. تتكون المهمة من اقتراح حل من أجل التمكن من التلاعب عن بعد بنظام الروبوت المتعدد عن طريق الحفاظ على التكوين المرغوب سيتم تسمية الروبوتات في النظام على هذا النحو ؛ قائد وعدد من أتباع الروبوتات. الهدف النهائي هو تحريك النظام متعدد الروبوتات من موقع مبدئي إلى وضع نهاية محدد مسبقًا للتحقق من صحة الحل المقترح ، سيتم تنفيذ محاكاة للنظام متعدد الروبوتات والبيئة روبوت ، نظام روبوت متعدد

كلمات البحث زعيم / متابع ، روبوتات متجانسة ، محاكاة ، روس ، محاكاة جازيبو

Résumé

Le but de ce travail est de contrôler une équipe de robots mobiles homogènes en formation Leader/Follower. Le travail consiste à proposer une solution afin de pouvoir manipuler par télé-opération le système multi-robots en gardant une formation désirée.

Les robots du système seront désignés comme ceci ; un leader et un nombre de robots followers. Le but final consiste de déplacer le système multi-robots d'une position initiale à une position finale prédéfinie.

Pour valider la solution proposée, une simulation du système multi-robots ainsi que l'environnement sera effectuée.

Mots clés : Robot, système multi-robots, leader/follower, robots homogènes, simulation, ROS, simulateur Gazebo.

Abstract

The purpose of this work is to control a team of homogeneous mobile robots in Leader / Follower formation. The scheme consists of proposing a solution in order to be able to manipulate by remote operation the multi-robot system in order to keep the desired formation.

The robots in the system would be classified like this; a leader and a set of robots followers. The ultimate goal is to move the multi-robot system from an initial position to a predefined end position.

To validate the proposed solution, a simulation of the multi-robot system and the environment will be performed.

Keywords: Robot, multi-robot system, leader/follower, homogeneous robots, simulation, ROS, Gazebo simulator.

Table des matières

Chapitre 1 : Les systèmes multi-robots

1. Introduction 3

2. La Robotique : Un domaine pluridisciplinaire 3

2.1 Définition de la robotique 3

2.2 Évolution de la robotique 3

2.3 Définition d'un robot 3

2.4 Composants d'un robot 4

2.4.1 Actionneurs 4

2.4.2 Capteurs 5

2.4.3 Effecteurs 5

2.4.4 Contrôleurs 5

2.4.5 Processeurs 5

2.4.6 Logiciels 5

2.5 Les Robots mobiles 6

2.5.1 Définition 6

2.5.2 Modèle cinématique d'un robot mobile 7

2.5.3 Mode de fonctionnement 7

3. Apparition des Systèmes Multi-Robots 8

3.1. Des Systèmes mono-robot aux systèmes multi-robots 8

3.2. Taxonomie des systèmes multi-robots 9

3.3.1 Niveau de coopération 10

3.3.2 Niveau de reconnaissance 11

3.3.3 Niveau de coordination 11

3.3.4 Niveau de l'organisation 11

3.4 Caractéristiques des systèmes multi-robots 11

3.4.1 Communication 12

3.4.2 Composition de l'équipe 12

3.4.3 Taille de l'équipe 12

3.4.4 Architecture du système 12

3.5 Avantages des systèmes multi-robots 12

3.5.1 Comparaison entre un système multi-robots et un seul robot 12

3.6 Champs d'application des systèmes multi-robots 13

3.6.1 Tâches couvrant une région 13

3.6.2 Tâches trop dangereuses 13

3.6.3 Tâches nécessitant une mise à l'échelle de la population 13

3.6.4 Tâches nécessitant une redondance 14

4. Conclusion 14A

Chapitre 2 : Approche leader/follower

1. Introduction 15

2. Contrôle de formation : Influence Biologique 15

2.1 Catégorisations des approches de contrôle de formations 16

2.1.1 Approches basées sur le comportement 16

2.1.2 Approche de structure virtuelle 17

2.1.3 Approches Leader-Follower 17

3. L'Approche Leader/Follower 18

3.1 Etat de l'art 18

3.2 Modélisation de la formation Leader/Follower 19

3.3 Stratégie de contrôle 19

3.4 Les Différentes formations Leader/Follower 20

4. Conclusion 20

Chapitre 3 : Analyse et conception

1. Introduction 21

2. Description du Processus 21
2.1 Principes fondamentaux du Processus Unifiés (UP) 21
2.2 Les modèles 22
3. Étude des besoins 23
3.1 Description des acteurs 23
3.2 Diagramme de contexte 23
3.2.1 Description du diagramme de contexte 24
4. Conception détaillé 24
4.1 Diagramme des cas d'utilisation global 24
4.2 Diagramme de séquences 25
4.2.1 Diagramme de séquence : Cas d'intégration d'un nouveau robot 26
4.2.2 Diagramme de séquence : Cas de suppression d'un objet 27
4.2.3 Diagramme de séquence : Cas de lancement du simulateur 28
4.2.4 Diagramme de séquence : cas de télé opération d'un robot 29
4.2.5 Diagramme de séquence : Lancement de la tâche Leader/Follower 29
4.3 Diagramme de classes 30
6. Conclusion 32

Chapitre 4 : Implémentation et validation

1. Introduction 33
2. Outil de développement 33
2.1 Système d'exploitation Ubuntu 33
2.2 Middleware ROS 33
2.2.1 Introduction à ROS : 33
2.2.2 Pourquoi devrions-nous apprendre ROS ? 34
2.2.3 Architecture de communication : 34
2.3 Langage de programmation Python 36
2.4 Langage de programmation C++ 36

3. Le simulateur Gazebo 36
4. Le Robot Turtlebot 38
4. Implémentation 39
4.1 Présentation du Point Cloud 39
4.1.1 Appliquer la méthode du Point Cloud sur les capteurs du TurtleBot 39
4.2 Mise en œuvre de la solution proposée : 40
4.2.1 La classe PCL : 41
4.2.2 Rocon APPlication 42
5. Test et Validation de l'Application 42
5.1 Etape de lancement de l'application 42
5.1.1 Lancement d'un robot *Turtlebot* dans l'environnement *Gazebo* 43
5.1.2 Lancement de *Rviz* 45
5.1.3 Intégration d'autres robots *TurtleBot* 45
5.1.4 Désignation du robot Leader 49
5.1.5 Lancement de l'opération Leader/Follower 49
5.1.6 Création d'un environnement *Labyrinthe* 50
5.1.7 Consultation de l'état du système développé 51
6. Conclusion 52
Conclusion Générale 58

Liste des Figures

Figure 1.1 : Représentation schématique d'un robot 4

Figure 1.2 : Le robot MOTOMAN, Centre du Développement des Technologies Avancées CDTA 6

Figure 1.3 : Modélisation du robot dans le repère absolu 7

Figure 1.4 : Exemples de différents types de tâches coopératives multi-robots 9

Figure 1.5 : Schéma présentant les différentes dimensions utilisées pour classifier les systèmes multi-robots selon 10

Figure 2.1 : Exemples de systèmes biologiques présentant plusieurs formations 15

Figure 2.2 : Exemple de l'utilisation des formations dans la vie réelle Véhicules sous-marins et aériens autonomes (AUV) et (UAV) 16

Figure 2.3 : Exemple de l'objectif de contrôle de la formation 17

Figure 2.4 : Modélisation de l'approche Leader/Follower pour trois robots 19

Figure 2.5 : Différentes formes de formations Leader/Follower 20

Figure 3.1 : Processus du développement 21

Figure 3.2 : Diagrammes UML utilisés 22

Figure 3.3 : Diagramme de l'acteur généralisé 23

Figure 3.4 : Diagramme de contexte 23

Figure 3.5 : Diagramme de cas d'utilisation général 26

Figure 3.6 : Diagramme de séquence : intégrer robot 27

Figure 3.7 : Diagramme de séquence : supprimer objet 28

Figure 3.8 : Diagramme de séquence : lancement du simulateur 29

Figure 3.9 : Diagramme de séquence : cas de télé opération d'un robot 30

Figure 3.10 : Diagramme de séquence : Lancement de la tâche Leader/Follower 31

Figure 3.11 : Diagramme de classes de l'environnement de simulation 32

Figure 4.2 : Messages de communication entre les différents nœuds 34

Figure 4.3 : Robot dans un simulateur 2D, en vie réelle et dans un simulateur 3D 38

Figures 3.4 : les robots mobiles les plus utilisés dans Gazebo 38

Figure 4.1 : Le robot mobile Turtlebot 39

Figure 4.2 : Composants d'un robot Turtlebot2 39

Figure 4.3 : Illustration de nuages de points dans le champ de vision du TurtleBot 40

Figure 4.4 : Initialisation de la classe TurtlebotFollower 41

Figure 4.5 : Récupération de données du capteur 3D du robot 41

Figure 4.6 : Si le Leader est dans le champ de vision du follower 42

Figure 4.7 : Si le leader s'éloigne du champ de vision du follower 42

Figure 4.8 : Résultat de la commande roscore 43

Figure 4.9 : Construire un Package 44

Figure 4.10 : Le robot TurtleBot dans l'environnement Gazebo 44

Figure 4.11 : Réglage du capteur 3D du TurtleBot 45

Figure 4.12 : Acquisition d'images et de vidéos par une caméra Kinect 45

Figure 4.13 : Environnement visualisé dans Rviz 46

Figure 4.14 : La commande sudo 46

Figure 4.15 : Intégration de deux autres robots 47

Figure 4.16 : Lancement de la commande roslaunch 47

Figure 4.17 : Interface liste des ROS MASTER 48

Figure 4.18 : Gazebo concert avec l'adresse du localhost 48

Figure 4.19 : Liste des choix 49

Figure 4.20 : Trois robots TurtleBot dans l'environnement Gazebo 49

Figure 4.21 : Sélection et Manipulation du robot Leader 50

Figure 4.22 : Désignation des robots followers 51

Figure 4.23 : Les trois robots TurtleBot, un leader et deux followers, dans le Labyrinthe 52

Figure 4.24 : Consulter l'état du système 52

Figure 4.25 : les graphes du système 53

Liste des Tableaux

Tableau 3.1 : Description du diagramme de contexte 21

Liste des acronymes

SMR : Système multi-robots

ROS : Robotic Operating System

Introduction Générale

La robotique est un très bon exemple de domaine pluridisciplinaire impliquant de nombreuses thématiques telles que la mécanique, l'électronique, l'automatique, l'informatique ou l'intelligence artificielle [1]. Les résultats de recherche ont donné naissance à des systèmes multi-robots qui ont remplacé le système mono-robot, afin de profiter de leurs différents avantages, citons la fiabilité, la rapidité et la flexibilité [15].

Actuellement, les systèmes multi-robots sont utilisés pour aider et remplacer l'être humain dans la réalisation de certaines tâches et dans plusieurs domaines ; tel que : la médecine, le domaine militaire, l'industrie etc.. . La coopération de ces systèmes a connu un essor considérable ces dernières années en raison des vastes applications qui dépendent sur la collaboration entre ces robots. Une des fondations de ce domaine de recherche est le concept des formations Leader/Follower [13].

Dans l'approche Leader/Follower, l'un des robots qui se trouve dans l'environnement est arbitrairement choisi comme le Leader, ainsi qu'un certain nombre d'autres robots qui seront désignés comme des robots suiveur, la position souhaitée du Leader est définie par le manipulateur, ce robot sera manipulé par télé opération jusqu'à l'arrivée à la destination finale souhaitée, la formation doit être maintenue pendant que la trajectoire du leader change [56].

La thématique générale abordée dans notre travail est le contrôle ou la manipulation d'un système multi-robot avec un nombre limité de robots suivant une formation leader/follower.

Pour assurer une meilleure présentation du travail effectué et garantir la clarté de ce mémoire, outre cette introduction générale, notre travail se comporte de 5 chapitres, suivis d'une conclusion générale, chaque chapitre met en évidence une contribution particulière du travail :

- Le premier chapitre sera consacré aux systèmes multi-robots, en commençant par des définitions des notions utilisées en robotique et les systèmes multi-robots, ou nous allons donner un état de l'art, et une taxonomie de ces systèmes en essayant de catégoriser le nôtre, nous finirons par les champs d'application des SMR.

- Dans le deuxième chapitre, le problème des approches du Leader/Follower sera étudié, en illustrant l'influence biologique sur la robotique collective, ainsi qu'une vue générale sur les approches concernant le contrôle des formations des robots, suivie d'un état de l'art sur l'approche Leader/Follower en expliquant la stratégie de navigation dans notre cas.

- Dans le troisième chapitre, nous proposons notre solution et on décrit en détails la modélisation et la conception de notre système, cette étape comporte les diagrammes UML, ainsi qu'un ROS graphe, décrivant de manière détaillé le fonctionnement de ce système.

- Dans le quatrième chapitre, nous allons décrire l'implémentation et le déploiement de notre système, un test et une validation de l'application sera aussi effectuée.

Une conclusion générale sera donnée à la fin de ce mémoire, résumant la contribution de notre travail, et discutant les principaux résultats obtenus. Nous allons essayer aussi de fixer des perspectives envisageables d'amélioration.

Chapitre 1 :

Les Systèmes Multi-Robots

1. Introduction

Le domaine de la robotique fait face à une croissance rapide dans la complexité des besoins et des exigences pour des robots chargés de tâches multiples, capables de coordonner leurs actions.

Ce chapitre sera consacré à la présentation des notions fondamentales des robots, et des systèmes multi-robots, en particulier les robots mobiles, nous allons présenter un état de l'art sur ce domaine, suivi des motivations, taxonomie et caractéristiques des systèmes multi-robots, pour enfin conclure avec leurs champs d'application.

2. La Robotique : Un domaine pluridisciplinaire

2.1 Définition de la robotique

La robotique est un domaine pluridisciplinaire impliquant de nombreuses thématiques on y trouve des aspects concernant la mécanique, l'électronique, l'automatique, l'informatique ou l'intelligence artificielle [1].

2.2 Évolution de la robotique

La robotique est passée par plusieurs étapes pour arriver à l'état actuel. Les premières machines dans l'histoire étaient de très grande taille et n'avaient aucune intelligence [2]. Étant conçues pour réponde au besoin de l'industrie à cette époque, elles étaient basées sur les lois de la mécanique seulement, et ne peuvent être contrôlées que par un être humain.

L'apparition des premiers circuits électroniques a permis de programmer les composants des machines, ce qui a contribué à faire apparaître les premiers robots [2]. Depuis, la robotique avancée est apparue ; de là, on a introduit la notion de mobilité du robot.

2.3 Définition d'un robot

Il existe plusieurs définitions du terme robot , mais elles tournent toutes autour de celle-ci : Un robot est une machine équipée de capacités de perception, de décision et d'action qui lui permettent d'agir de manière autonome dans son environnement en fonction de la perception assurée par leurs capteurs [2]. On distingue deux principaux types de robot sur le critère de la mobilité : les robots manipulateurs les robots mobiles. Les robots manipulateurs ont une base fixe contrairement aux robots mobiles qui peuvent se déplacer.

2.4 Composants d'un robot

Un robot, en tant que système, se compose d'éléments, qui sont intégrés pour former un ensemble. Madame Akli, a considéré le robot comme un agent artificiel, ayant l'espace physique comme environnement.

Un agent est une entité équipée de la capacité de perception, saisissant son entourage grâce à des *capteurs*, prenant des décisions à l'aide du *contrôleur*, et enfin agissant en conséquence en usant des *effecteurs* ; il peut donc s'adapter seul aux variations de son environnement, de telle sorte que la tâche soit correctement exécutée en dépit de ces variations ; il doit comprendre un «corps» et un «cerveau». La figure suivante nous montre la représentation schématique du robot [3].

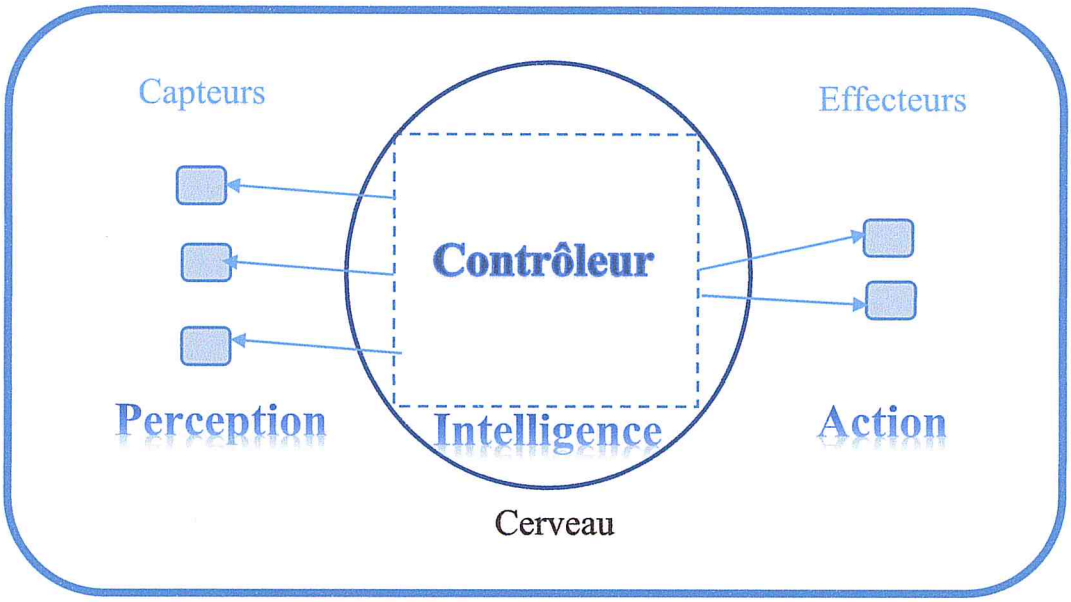


Figure 1.1 : Représentation schématique d'un robot [3].

Nous allons dans ce qui suit donner certaines définitions fondamentales.

2.4.1 Actionneurs

Ce sont des mécanismes qui permettent à/aux effecteur(s) d'exécuter une action, de convertir les commandes logicielles (Software) en mouvements physiques. Leur but primaire est de produire assez de force pour provoquer le mouvement du robot ; celle-ci représente la transformation d'une énergie source en énergie mécanique. La technologie des actionneurs est étroitement liée à l'énergie de base utilisée (pneumatique, hydraulique, électrique) [3].

2.4.2 Capteurs

Ce sont des outils de perception qui permettent de gérer les relations entre le robot et son environnement. Il existe deux types de capteurs :

- **Capteurs proprioceptifs** : ils mesurent l'état mécanique interne du robot (comme les capteurs de position, de vitesse ou d'accélération).
- **Capteurs extéroceptifs** : ils recueillent des informations sur l'environnement (comme la détection de présence, mesure de distance, etc.).

Les capteurs ont comme fonction de lire les variables relatives au mouvement du robot pour permettre un contrôle convenable [3].

2.4.3 Effecteurs

Ce sont tous les mécanismes à travers lesquels le robot peut effectuer des changements propres à lui, ou relatifs à l'environnement. Ces changements se font grâce aux actionneurs [3].

2.4.4 Contrôleurs

Le contrôleur reçoit les données de l'ordinateur (le cerveau du système), commande les mouvements des actionneurs, et coordonne les mouvements avec les informations reçues par les capteurs [3].

2.4.5 Processeurs

Le processeur est le cerveau du robot. Il calcule les mouvements des différentes parties du robot, la vitesse de chaque moteur et supervise les actions coordonnées du contrôleur et les capteurs. Dans certains systèmes, le contrôleur et le processeur sont intégrés ensemble en une seule unité, et dans d'autres cas, ce sont des unités séparées [3].

2.4.6 Logiciels

Trois groupes de logiciels sont utilisés dans un robot [3] :

- L'un est le système d'exploitation qui exploite le processeur.
- Le second est le logiciel robotique qui calcule le mouvement nécessaire de chaque moteur du robot.
- Le troisième groupe est la collection d'applications, les routines et les programmes développés pour utiliser le robot ou ses périphériques pour des tâches spécifiques telles que l'assemblage, le chargement de machines, la manutention et les routines de vision.

2.5.2 Modèle cinématique d'un robot mobile

Considérons la modélisation suivante sur un terrain plat. On note ***R (0, x, y, z),*** un repère fixe absolu, dont l'axe Z est vertical et ***R_0 (0_0, x_0, y_0, z_0)*** un repère mobile lié au robot. Généralement, sur un robot, le point de contrôle O_0 est fixé au centre de l'axe des roues motrices. L'état du robot, appelé *"situation"* selon [5] ou *"posture"* selon [6], il est défini par le vecteur :

$$\xi = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$$

- Dans lequel θ désigne l'orientation du robot.

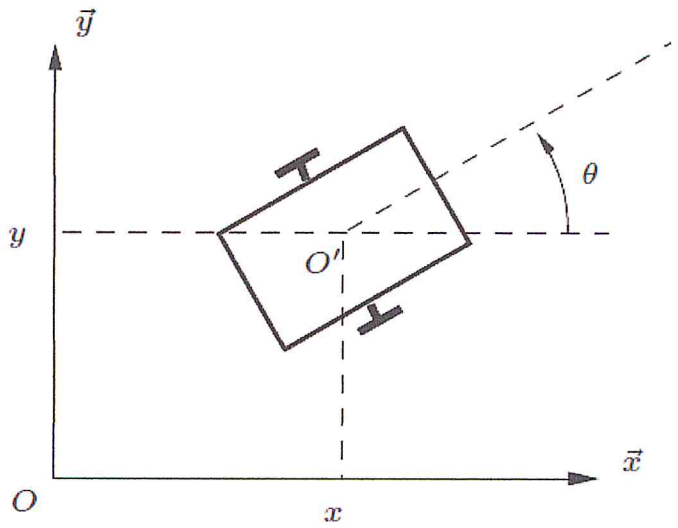


Figure 1.3 : Modélisation du robot dans le repère absolu [5].

2.5.3 Mode de fonctionnement

Il existe deux principaux modes de fonctionnement pour un robot mobile : télé opéré et autonome :

- ***Mode télé-opéré*** *:* dans ce mode, le robot est piloté à distance ; une personne donne des consignes au robot via une interface de commande. Ces consignes sont ensuite transmises au robot via un lien de communication.
- ***Mode autonome*** *:* un robot autonome est un robot qui doit prendre ces propres décisions, et qui est capable de percevoir correctement son environnement mais également de savoir comment réagir en conséquence, suivant son niveau d'autonomie.

3. Apparition des Systèmes Multi-Robots

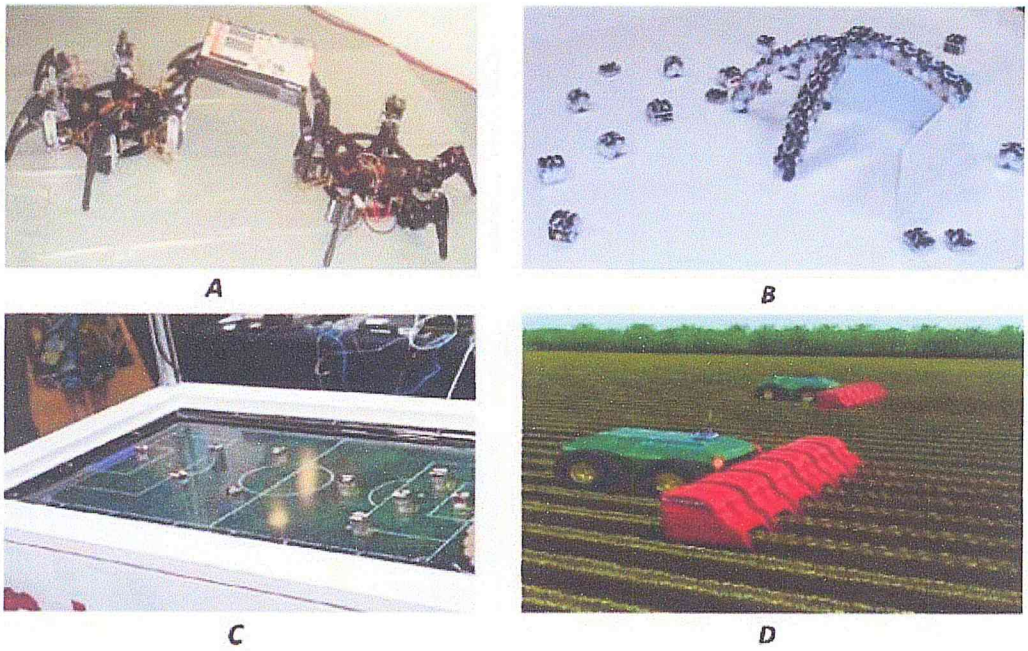
Au cours de ces dernières décennies, la robotique collective est rapidement devenue un domaine de recherche vaste et important, comprenant des idées et thèmes différents ; citons par exemple les systèmes multi-agents [7], la robotique distribuée [8], Swarm robotics [9, 10, 11], et finalement les systèmes multi-robots [12, 13, 14, 15].

En 1989, Brooks et Flynn du laboratoire *MIT Artificial Intelligence* ont proposé une idée radicale dans le domaine de l'exploration du système solaire : remplacer un grand robot explorateur par une collection de petits robots mobiles appelés *rovers* [16]. Ils ont estimé que le coût par kilogramme des *rovers* serait fortement réduit. Diminuer un peu la fiabilité pour chaque rover mobile serait acceptable, du moment que l'échec d'un rover unique ne mettrait pas en péril toute la mission. Cette idée radicale présente la principale motivation pour l'utilisation de systèmes multi-robots.

3.1. Des Systèmes mono-robot aux systèmes multi-robots

Un système multi-robots peut être défini comme étant un ensemble de robots opérant dans le même espace de travail [19]. Les motivations principales qui ont poussé les scientifiques de s'orienter vers les systèmes multi-robots ont données comme suit [20] :

- Amélioration du temps d'exécution d'une tâche : prenons l'exemple de pousser un objet lourd par un groupe de robots. Le travail ici consiste à diviser la force globale nécessaire à cette tâche sur les différentes entités robotiques qui peuvent détecter les points optimaux où appliquer leurs efforts et pousser l'objet ensemble dans la direction du mouvement souhaitée.
- Avec un groupe de robots mobiles, on bénéficie d'une distribution de capteurs et d'actionneurs sur les entités robotiques, ce qui rend le système plus tolérant à certains défauts de fonctionnement. Par exemple, la panne d'un robot n'entraîne pas l'arrêt systématique de la mission, du fait de la redondance des entités robotiques.
- L'utilisation d'un groupe de robots peut s'avérer une solution plus simple et moins chère que la conception d'un seul robot dont la structure et le contrôle peuvent être complexe et encombrants.



(A) SMR pour soulever un objet [21]. (B) SMR par assemblage [22]. (C) SMR de robots footballeurs [23]. (D) SMR agricoles [24].

Figure 1.4 : Exemples de différents types de tâches coopératives multi-robots.

3.2. Taxonomie des systèmes multi-robots

Les travaux précédents d'Iocchi et al. [13], Dudek et al. [7] et Cao et al. [17] ont tous tenté de catégoriser ou de classifier les systèmes multi-robots selon des groupes spécifiques. Cao et al. [17] ont commencé par fournir une organisation taxonomique en se basant sur des problèmes et des solutions déjà existants, contrairement à Dukek et al. [7] qui ont présenté une catégorisation selon la communication, et d'autres capacités. Finalement, Iocchi et al. [13] ont proposé une nouvelle classification en utilisant le terme *dimension* pour référencier certaines fonctionnalités spécifiques. On distingue un niveau de coopération, un niveau de reconnaissance, un niveau de coordination, et un dernier niveau d'organisation.

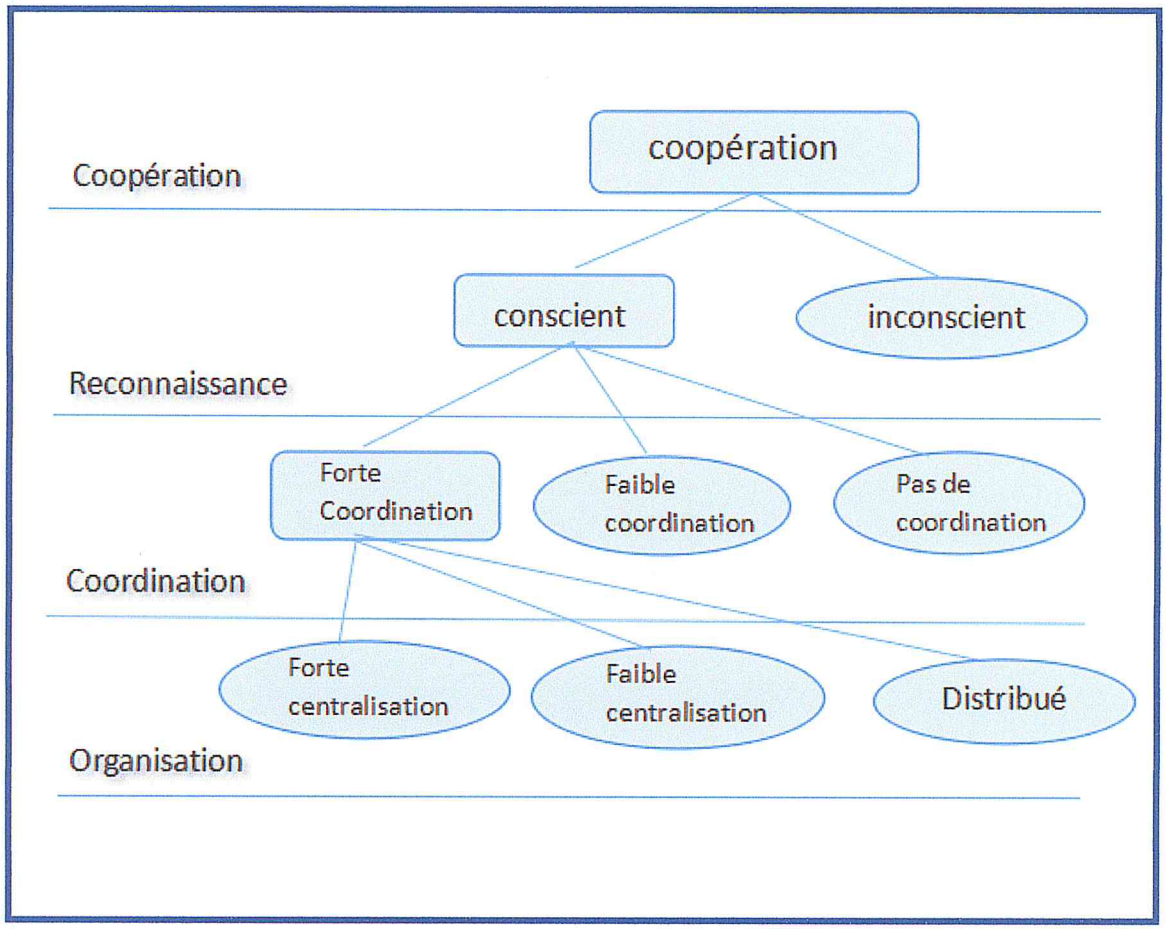


Figure 1.5 : Schéma présentant les différentes dimensions utilisées pour classifier les systèmes multi-robots selon [13].

Afin de mieux comprendre cette classification, il est important de définir clairement les niveaux utilisés dans cette taxonomie, en essayant de catégoriser notre système au fur et à mesure.

3.3.1 Niveau de coopération : le premier niveau concerne la capacité du système à coopérer pour accomplir une tâche spécifique. Au niveau de la coopération, nous distinguons des systèmes coopératifs de ceux qui ne sont pas coopératifs. Dans notre travail, nous nous intéressons seulement aux SMR coopératifs (un système coopératif est composé de *robots qui fonctionnent ensemble pour effectuer une tâche globale* [25]). Parmi les approches les plus répandues, nous citons l'approche Leader/Follower que nous allons détailler dans le chapitre suivant.

3.3.2 Niveau de reconnaissance : le deuxième niveau de cette structure hiérarchique concerne la connaissance que chaque robot de l'équipe a sur les autres robots du système. Les robots conscients reconnaissent l'existence d'autres individus dans leurs équipes, quant aux robots inconscients qui agissent sans aucune connaissance d'autres robots dans le système. Notons que la notion de reconnaissance n'est pas équivalente à la communication. En effet, utiliser un mécanisme de communication n'implique pas la conscience et vice-versa ; un SMR peut être conscient même s'il n'y a pas de communication directe parmi les robots.

3.3.3 Niveau de coordination : le troisième niveau concerne les mécanismes utilisés pour la coopération. Cette dernière est définie comme un état de coopération dans lequel les actions d'un groupe sont effectuées en réaction aux actions précédentes du groupe. La coordination peut être centralisée ou décentralisée. Il existe plusieurs manières afin qu'un robot puisse savoir les actions des autres membres du système, d'où vient le protocole de coordination, qui est défini comme un ensemble de règles que les robots doivent suivre afin d'interagir les uns avec les autres dans l'environnement.

3.3.4 Niveau de l'organisation : le quatrième niveau de la structure hiérarchique concerne la façon dont le système de décision est réalisé au sein du SMR. Un système centralisé a un agent qui est en charge d'organiser le travail des autres agents ; les autres membres ne peuvent agir que selon les instructions du chef. D'un autre côté, un système distribué est composé d'agents complétement autonomes dans la décision traitée l'un par rapport à l'autre. Plus précisément, une forte centralisation est utilisée pour caractériser un système dans lequel les décisions sont prises par le même agent prédéfini pendant toute la durée de la mission.

3.4 Caractéristiques des systèmes multi-robots

Le système multi-robots utilisé dans notre étude est un système distribué. La distribution est considérée comme étant l'autonomie de chaque composant du système et sa capacité de prendre des décisions sur les actions à effectuer [26]. Avec la classification introduite pour caractériser le système multi-robots, il existe un certain nombre de caractéristiques qui sont pertinentes pour le développement du système, Iocchi et al. [13] les ont regroupés dans quatre dimensions du système :

3.4.1 Communication : la communication dans les systèmes multi-robots a plusieurs significations. La question posée ici est : Comment les robots vont-ils communiquer entre eux ? Dans les SMR, les messages échangés entre les robots du système utilisent des mécanismes de communication. Cet échange peut se dérouler d'une manière directe où les robots peuvent explicitement communiquer entre eux, ou indirecte (stigmergique) ce qui signifie que les robots déduisent ce que l'autre fait en fonction d'autres actions de robots ou comment l'environnement a été changé [27].

3.4.2 Composition de l'équipe : Selon la composition de l'équipe, les SMR peuvent être divisés en deux classes principales, hétérogènes et homogène [28]. Les équipes homogènes sont composées de plusieurs robots ayant exactement le même niveau d'intelligence ainsi que le même mécanisme de contrôle ; tandis que dans les équipes hétérogènes, les robots sont différents soit dans les dispositifs matériels ou dans le mécanisme de contrôle.

3.4.3 Taille de l'équipe : Bien que certains travaux récents ont utilisé des SMR à grandes échelles [29] [30], le SMR considéré dans notre étude opère avec un nombre limité de robots.

3.4.4 Architecture du système : les auteurs dans [31] ont regroupé les architectures des SMR en deux classes :

- ***Architecture réactive*** *:* chaque robot de l'équipe poursuit une approche individuelle face aux changements environnementaux afin de réaliser une tâche prédéfinie.
- ***Architecture délibérative*** *:* les robots du système peuvent fournir de nouvelles stratégies suite aux changements environnementaux.

La différence entre les deux architectures s'appuie sur les différentes approches adoptées par le SMR pour se remettre d'une situation imprévue.

3.5 Avantages des systèmes multi-robots

Les avantages des systèmes multi-robots peuvent être présentés en comparant ces derniers avec un seul robot.

3.5.1 Comparaison entre un système multi-robots et un seul robot

Pour compléter une tache sophistiquée, un seul robot doit être conçu avec des structures complexes ce qui va entrainer que les coûts de conception, de construction et d'entretien seront beaucoup plus élevés. En plus, la défaillance d'une partie du robot peut affecter tout le système, ou ça sera difficile de prédire ce qui va se passer.

Quant à un système multi-robots, qui sera capable d'atteindre la même capacité d'un seul robot grâce à la coopération intergroupe entre les différents robots du système, de plus, le système peut bénéficier de la réutilisabilité des robots simples du qui ne demandent un faible coût de construction et de maintenance.

3.6 Champs d'application des systèmes multi-robots

Bien que les recherches sur les systèmes multi-robots soient toujours en pleine croissance, il existe de nombreuses applications qui commencent à résoudre certains problèmes avec les SMR. Un certain nombre d'applications sont présentées ci-dessous où les systèmes multi-robots seraient applicables, tout en mettant l'accent sur les propriétés des tâches en donnant des problèmes réels comme exemples :

3.6.1 Tâches couvrant une région

Les systèmes multi-robots seraient bien adaptés aux tâches nécessitant une grande surface, et cela grâce à leurs grandes capacités de détection, ce qui peut offrir une meilleure surveillance et une détection immédiate en cas d'événements dangereux.

Prenons l'exemple de la fuite accidentelle d'un produit chimique [32], dans ce cas, chaque robot du système peut patrouiller dans une zone au lieu de rester immobile, de plus, les robots peuvent également localiser la source, se déplacer vers la zone et prendre des mesures rapides pour régler ce problème.

3.6.2 Tâches trop dangereuses

L'évolutivité et la stabilité des systèmes multi-robots leurs permet de subir une perte de quelques robots avant que systèmes réalise la tache souhaitée, dans ce cas, les robots peuvent même remplacer les travailleurs.

Dans l'exemple de nettoyer un champ minier [33], le système multi-robots peut réaliser à moindre coût ce type de travail.

3.6.3 Tâches nécessitant une mise à l'échelle de la population

La charge du travail peut toujours changer à travers le temps, et grâce aux propriétés des systèmes multi-robots, la taille de ce dernier peut être mise à l'échelle en fonction de la charge de travail actuel.

Dans l'exemple des fuites d'huile après les accidents [34], au début, le système multi-robots doit maintenir une population élevée lorsque l'huile fuit rapidement, de même quand la source de fuite est bouchée, le système multi-robots peut réduire progressivement le nombre robots.

3.6.4 Tâches nécessitant une redondance

Grace à la redondance des systèmes multi-robots, ce dernier peut se dégrader paisiblement en évitant les pannes catastrophiques.

Dans le cas où le système multi-robots est utilisé dans les champs de bataille [35], le système a la capacité de créer des réseaux de communication dynamiques qui bénéficient de la robustesse et redondance du système lorsque certains de robots (ici nœuds de communication) sont en panne.

4. Conclusion

Dans ce chapitre, nous avons présenté d'une manière globale les définitions et les explications dans le concept de la robotique, plus précisément la robotique mobile, et les systèmes multi-robots, nous avons aussi présenté une vision d'ensemble des caractéristiques et avantages associées aux systèmes multi-robots. Le chapitre suivant sera consacré à la commande coopérative des systèmes multi-robots, nous allons nous concentrer sur l'approche Leader/Follower.

Chapitre 2 :

Approches Leader/Follower

1. Introduction

Le contrôle de la formation des systèmes multi-robot a reçu une attention considérable au cours des dernières années, dans la recherche en robotique. L'objectif du contrôle de la formation de plusieurs robots mobiles est de maintenir une orientation et une distance souhaitée entre deux ou plusieurs robots mobiles d'un point initial a un point final.

Ce chapitre sera consacré aux différentes approches de contrôle de formation, en particulier l'approche Leader/Follower que nous allons bien détailler ci-dessous.

2. Contrôle de formation : Influence Biologique

Au cours de ces dernières années, les scientifiques se sont inspirés du monde naturel ; plus précisément, la façon dont certaines espèces naturelles [36], telles que les insectes, les troupeaux d'oiseaux et les colonies d'abeilles se comportent dans un groupe.

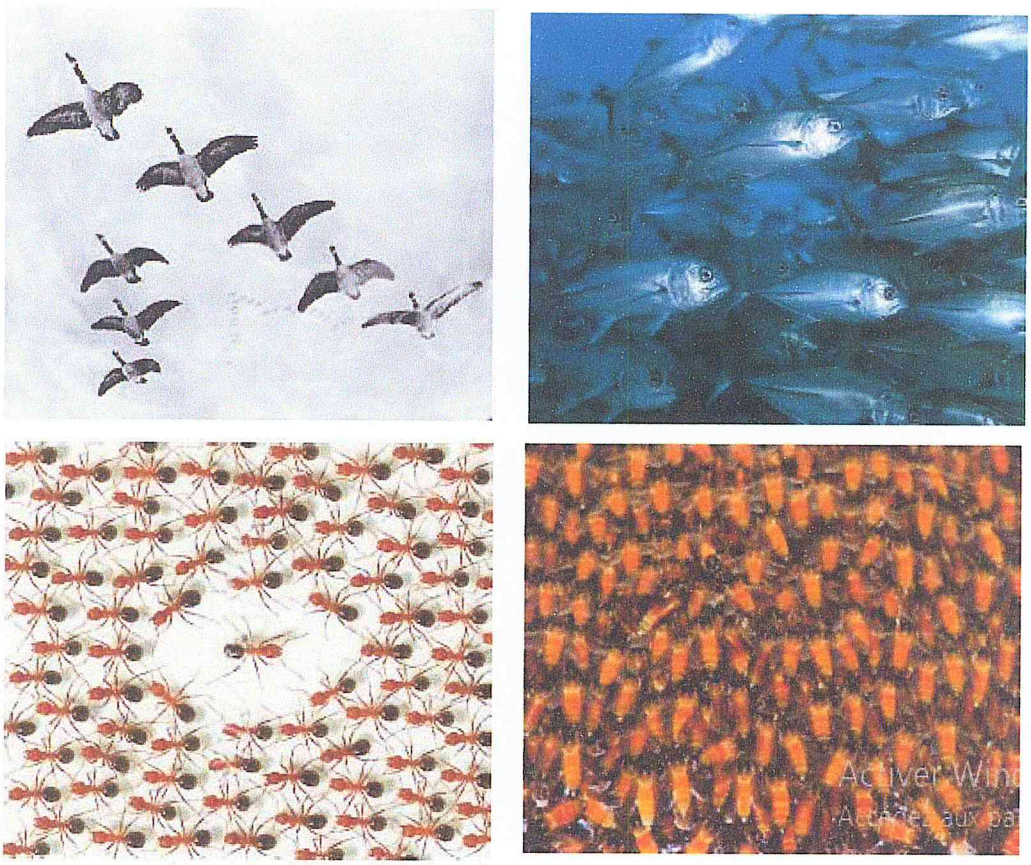


Figure 2.1 : Exemples de systèmes biologiques présentant plusieurs formations [36].

Das et al [37] et Sisto et Gu, [38] ont eu l'idée du contrôle de la formation. Cela signifie simplement contrôler la position relative et l'orientation des robots dans un groupe tout en permettant le groupe de se déplacer dans son ensemble.

L'objectif général est de concevoir de nouvelles lois de contrôle pour chaque robot selon différentes tâches de contrôle de sorte que le système multi-robots puisse former et maintenir un motif géométrique désiré ainsi que suivre une trajectoire désirée.

Figure 2.2 : Exemple de l'utilisation des formations dans la vie réelle
Véhicules sous-marins et aériens autonomes (AUV) et (UAV) [48].

2.1 Catégorisations des approches de contrôle de formations

Il existe de multiple stratégies et approches qui ont émergé pour le contrôle de formations qui peuvent être catégorisées comme suivant : approches basées sur le comportement, approches Leader-Follower et approches de structure virtuelle.

2.1.1 Approches basées sur le comportement

Dans cette approche, plusieurs comportements souhaités (par exemple, évitement d'obstacles, évitement de collision, attraction de cible) sont attribués à chaque robot ; et le contrôle final est dérivé d'une pondération de l'importance relative de chaque comportement [40] [41] [42].

Les avantages de cette approche sont donnés comme suit : son parallélisme, ses caractéristiques distribuées en temps réel, et moins d'informations sont communiquées entre les robots [43]. Par conséquent, il est très utile de guider un système de robots mobiles dans un environnement changeant. Cependant, il peut être difficile de décrire la dynamique du groupe et de garantir la stabilité du système. De plus, il est difficile d'analyser mathématiquement sa performance comportementale [44].

2.1.2 Approche de structure virtuelle

Dans cette approche, la formation entière est traitée comme une entité unique. [45] ont utilisé cette approche pour mettre en œuvre le contrôle des UAV. Aussi, [46] ont utilisé une combinaison de la structure virtuelle et des approches de suivi de trajectoire pour dériver une stratégie de contrôle pour coordination de plusieurs robots mobiles.

L'avantage de cette approche est qu'il est plus facile de décrire le comportement coordonné du groupe. Cependant, le contrôleur n'est pas en architecture distribuée et peut rencontrer des difficultés pour résoudre certaines applications de formation.

2.1.3 Approches Leader-Follower

Dans l'approche Leader-Follower [47] [48] [49] [50], un robot joue le rôle de leader qui génère la trajectoire de référence pour le groupe de robots ; le reste de robots du groupe agissent comme des suiveurs qui doivent garder la séparation et le relèvement souhaité par rapport au chef (c.-à-d., le leader). Il existe également différentes formes.

En effet, une fois le mouvement du leader donné, la séparation désirée et le relèvement souhaité du follower avec le leader peuvent être réalisés en choisissant une loi de commande locale sur chaque follower en fonction de sa dynamique de position relative. Ensuite, la stabilité de la formation est également garantie, c'est-à-dire que le groupe entier peut atteindre et maintenir la formation souhaitée.

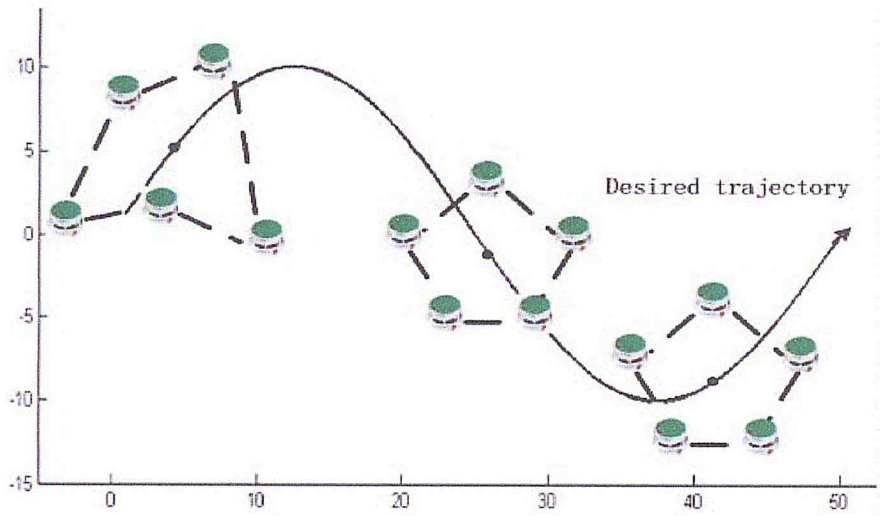


Figure 2.3 : Exemple de l'objectif de contrôle de la formation [47].

Nous allons opter pour l'approche Leader/Follower dans notre travail pour contrôler la formation du système multi-robots, en raison de sa simplicité, de sa flexibilité.

3. L'Approche Leader/Follower

3.1 Etat de l'art

Parmi les approches mentionnées ci-dessus, l'approche leader/follower a été la plus répandue, différentes méthodes de contrôle ont été développées pour les systèmes multi-robots [51]. Shen et ses collègues [52] ont considéré ce problème sans mesure directe de la vitesse linéaire du robot leader. Ils ont ainsi développé deux algorithmes non-linéaires: le premier est basé sur une méthodologie de contrôle par rétroaction dynamique adoptive (feedback); le deuxième algorithme est basé sur une méthodologie d'estimation de l'immersion et de l'invariance du second ordre, avec deux robots mobiles seulement, l'un jouant le rôle de leader et l'autre de follower, les deux robots se déplaçaient à l'intérieur d'un espace de travail libre suivant une formation en forme d'arcade (arch shape formation).

Dai et Lee [53] ont construit la formation de leader-waypoint-follower en fonction des états de mouvement relatifs du système multi-robots. L'approche proposée consiste à trouver un point de passage géométrique appelé waypoint afin de déplacer correctement le robot follower vers son waypoint, afin de maintenir la formation de ce système, les auteurs ont combiné deux méthodes : la méthode de contrôle de suivi stable et la méthode de recul de l'horizon. Pour valider leur approche, des simulations ont été réalisées avec trois robots qui se déplaçaient dans un environnement libre en ligne droite, en un cercle et une sinusoïde. Min et al, [54], ont présenté un algorithme de formation leader/follower basé sur la vision, où la trajectoire du leader est inconnue pour les robots followers. Ceci est accompli en appliquant un feedback linéaire d'entrée-sortie. Les auteurs ont utilisés des formations en ligne droite et en diagonale sans considérer les obstacles de l'environnement pur valider cette approche. Un autre travail similaire a été proposé par Vidal et al. [55] vise à permettre à un groupe de robots mobiles de maintenir visuellement des formations sans avoir une communication entre les robots. Pour ce but, des stratégies linéarisées de feedback des états complets ainsi que des feedbacks d'entrées-sorties ont été suggérées pour le leader et les followers, respectivement, dans le but d'obtenir une formation précise, chaque follower utilisait un flux optique pour estimer les positions relatives du leader, ce qui permettra au groupe de maintenir visuellement la formation désirée.

3.2 Modélisation de la formation Leader/Follower

En considérant la configuration de base, un Leader et un Follower sont notés R_l et R_f, respectivement. Selon la notation définie précédemment, les états et les entrées de R_l et R_f sont notés (x_l, y_l, θ_l), (x_f, y_f, θ_f), de même pour leurs vitesses angulaires et linéaires (V_l, ω_l) et (V_f, ω_f). Les équations du mouvement des deux robots sont données par :

$$d_{lf} = \sqrt{(x_l - x_f)^2 + (y_l - y_f)^2}$$
$$\psi_{lf} = \pi - arctan2(y_f - y_l, x_f - x_l) - \theta_l \quad (1)$$

La distance relative entre le leader et le follower est notée $d_{lf}=d_{fl}$ et Ψ_{lf} représente l'angle de palier de séparation entre les deux robots.

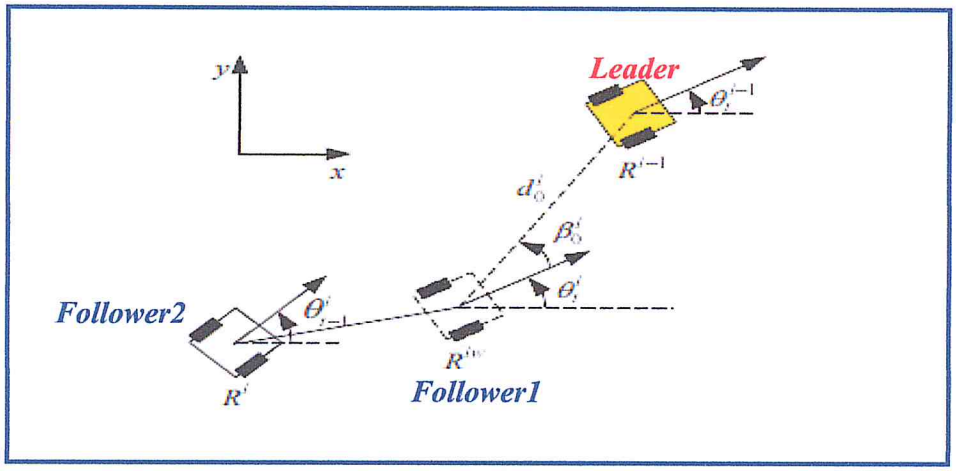


Figure 2.4 : Modélisation de l'approche Leader/Follower pour trois robots [5].

3.3 Stratégie de contrôle

Les auteurs dans [56] ont suivi la stratégie suivante : au début de l'expérimentation, l'un des robots qui se trouve dans la zone de communication de l'opérateur est arbitrairement choisi comme le leader, les autres joueront le rôle des followers, l'opérateur peut toujours choisir quelques robots comme followers et laisser les robots restants hors la formation. La position souhaitée du leader est définie par l'opérateur ; par contre, la position désirée du follower est définie par la position du follower d'une distance d_{lf}. Ainsi, si l'opérateur modifie la trajectoire du leader, la formation suivra automatiquement.

3.4 Les Différentes formations Leader/Follower

Différents types de formes ont été implémentés dans le contrôle de la formation [56]. Les formes de formation communes sont la colonne, la ligne, le coin, le triangle, le cercle, etc.,

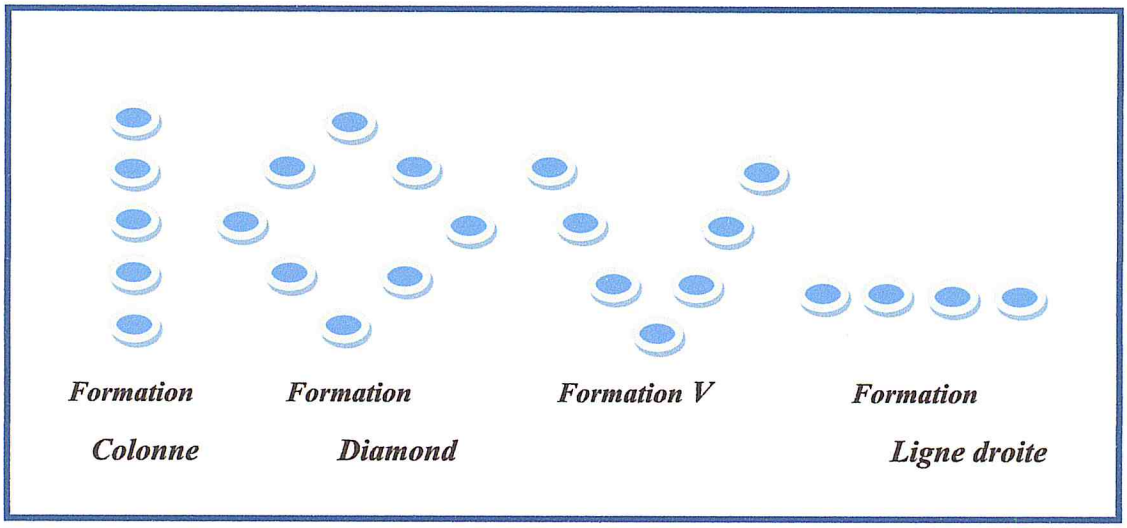


Figure 2.5 : Différentes formes de formations Leader/Follower.

4. Conclusion

Dans ce chapitre, nous avons présenté une vue générale sur les approches du contrôle de formation dans les systèmes multi-robots, après avoir détaillé l'influence biologique dans ce domaine de recherche. On s'est basé dans notre travail sur l'approche Leader/Follower qu'on a décrit et présenté un état de l'art, nous avons ensuite expliqué sa stratégie de navigation et illustré les différentes formes de formations que les robots peuvent prendre.

Le chapitre suivant sera consacré à l'analyse et la conception de notre systèm

Chapitre 3 :

Analyse

Et

Conception

1. Introduction

Dans ce chapitre, nous allons présenter la conception de notre travail. La démarche suivie s'appuie sur l'approche objet en utilisant UML (Unified Modeling Langage). La conception a pour objectif de formaliser les étapes du développement d'un système afin de le rendre plus fidèle aux besoins exigés.

UML n'est, toutefois, qu'un langage de modélisation et il doit être accompagné d'un processus qui devra guider la modélisation, étape par étape, jusqu'à la réalisation.

2. Description du Processus

Le processus suivi dans ce travail est un processus simplifié, inspiré de l'UP (Unified Process), qui est un cadre général d'une méthode générique de processus de développement fondé sur les travaux effectués dans [59]. Le processus adopté s'inspire également des pratiques de l'eXtreme Programming (XP) qui est une approche minimaliste à la mode centrée sur le code.

2.1 Principes fondamentaux du Processus Unifiés (UP)

Le UP est un processus de développement logiciel (Figure 1), qui est (*i*) itératif et incrémental, (*ii*) centré sur l'architecture, (*iii*) piloté par les risques et enfin (*iv*) conduit par les cas d'utilisation.

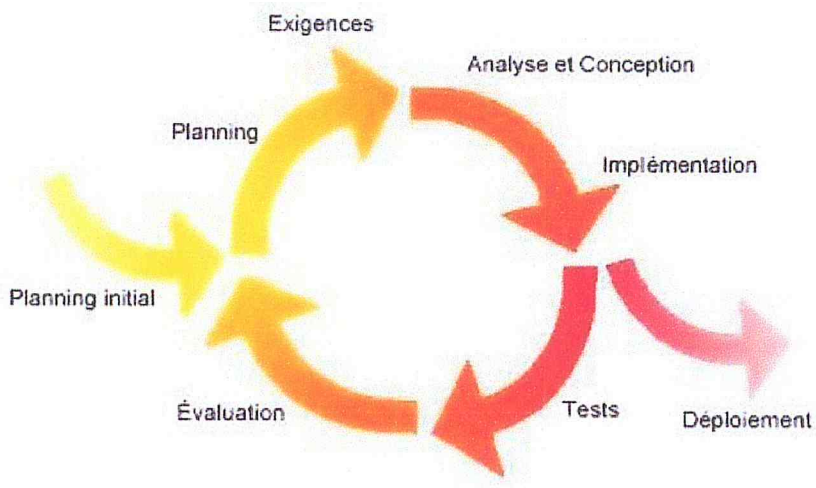


Figure 3.1 : Processus du développement [59].

2.2 Les modèles

Pour mener efficacement le processus, nous avons besoin de construire des modèles représentant notre application. Un modèle est une représentation abstraire, une abstraction du système à développer, du système en développement ou encore du système développé. Les modèles évoluent donc avec le système et jouent un rôle majeur dans chaque phase de développement. Pour matérialiser ces modèles (cas d'utilisation, analyse et conception en particulier), nous utilisons les diagrammes UML (Figure 2), adaptés à notre cas.

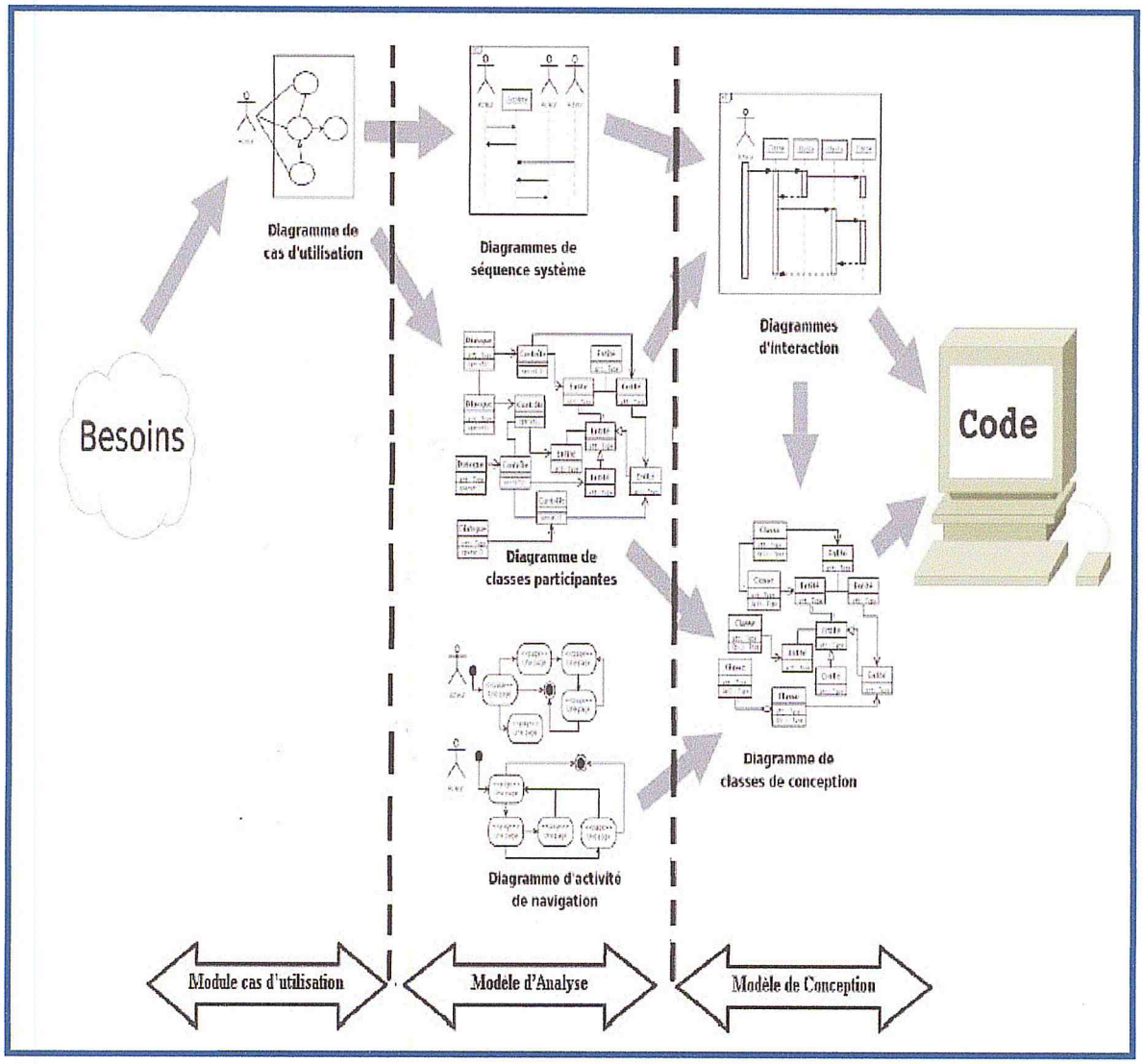


Figure 3.2 : Diagrammes UML utilisés.

3. Étude des besoins

3.1 Description des acteurs

Nous avons modélisé les acteurs de notre système, sur le schéma décrit par la figure suivante :

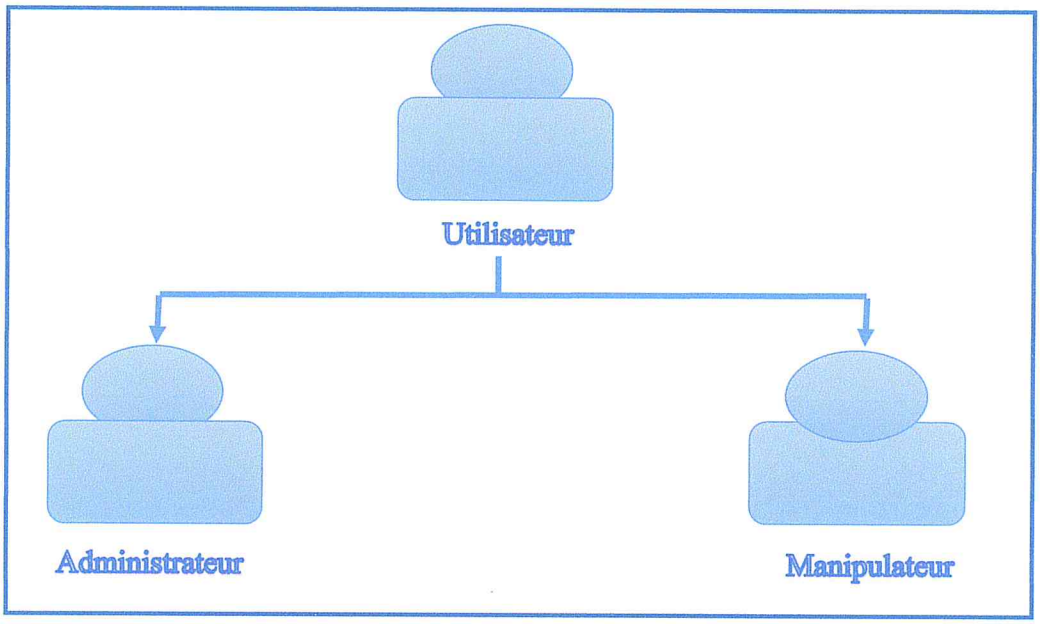


Figure 3.3 : Diagramme de l'acteur généralisé.

3.2 Diagramme de contexte

Dans cette phase, le système est considéré comme une boite noire qui reçoit et émet des messages en interaction avec le monde extérieur englobant les acteurs qui l'utilisent [59].

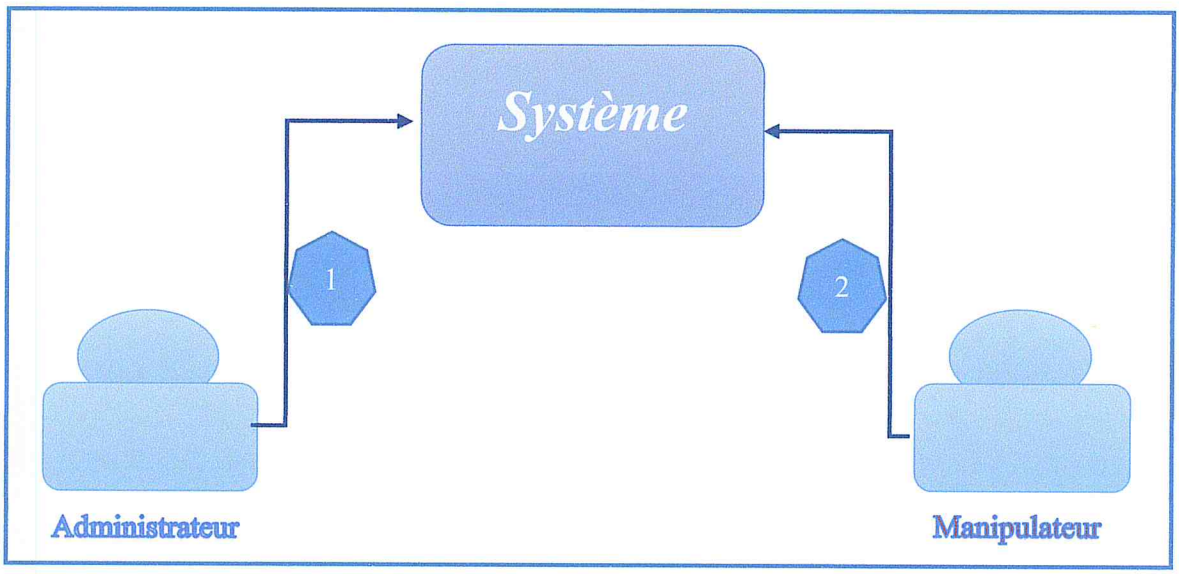


Figure 3.4 : Diagramme de contexte.

3.2.1 Description du diagramme de contexte

Tableau 1 suivant décrit les différents messages échangés entre les acteurs et le système.

Numéro	Émetteur	Message
1	Administrateur	▪ Intégrer des robots. ▪ Supprimer des robots. ▪ Initialiser les positions des robots. ▪ Intégrer des objets. ▪ Supprimer des objets. ▪ Définir les positions des objets. ▪ Consulter l'état du système.
2	Manipulateur	▪ Lancer la simulation. ▪ Sélectionner le robot Leader. ▪ Manipuler les robots par télé opération. ▪ Lancer la tâche Leader/Follower. ▪ Sélectionner les robots followers. ▪ Consulter l'état du système. ▪ Intégrer les objets. ▪ Supprimer les objets. ▪ Déplacer les objets.

Tableau 3.1 : Description du diagramme de contexte.

4. Conception détaillé

4.1 Diagramme des cas d'utilisation global

Un diagramme de cas d'utilisation capture le comportement d'un système. Il permet d'exprimer les besoins des utilisateurs afin d'entamer la partie du développement. Ce diagramme permet de donner une vision globale du comportement fonctionnel du système, ainsi que les acteurs (utilisateurs) et leurs interactions avec le système [59].

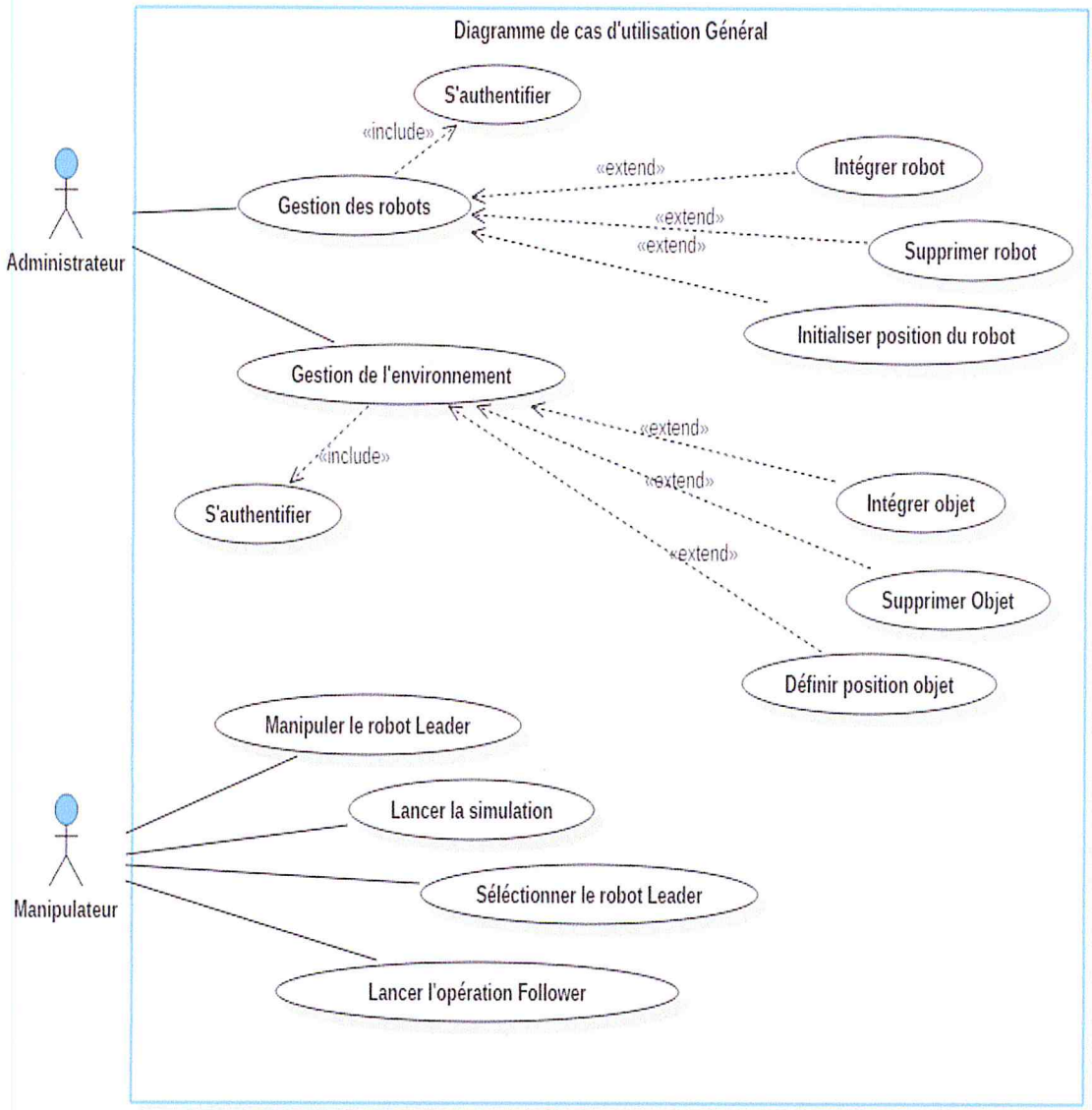


Figure 3.5 : Diagramme de cas d'utilisation général.

4.2 Diagramme de séquences

Le diagramme de séquence montre les interactions entre les objets, agencées en séquence dans le temps. Il montre en particulier les objets participants à l'interaction par leurs lignes de vie et les messages qu'ils échangent de façon ordonnée dans le temps [59]. Dans ce qui suit, nous présentons les principaux diagrammes de séquence de notre travail.

4.2.1 Diagramme de séquence : Cas d'intégration d'un nouveau robot

La Figure 6 suivante montre le diagramme de séquence d'intégration d'un nouveau robot mobile dans le système. Cette étape nécessite l'authentification de l'administrateur afin de pouvoir mettre à jour le fichier du nœud gazebo.parameters, qui contient les données des robots simulés dans l'environnement Gazebo.

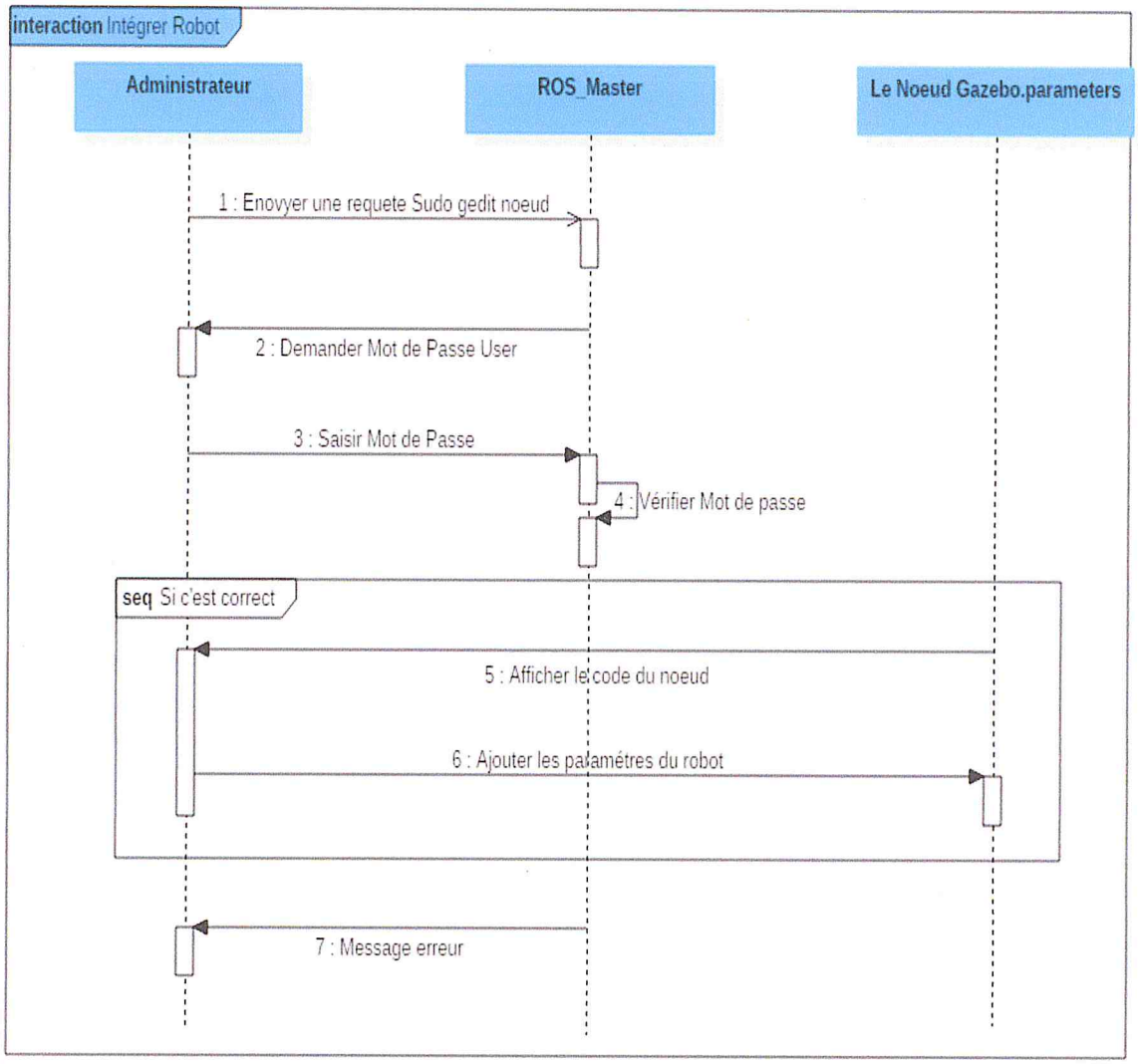


Figure 3.6 : Diagramme de séquence : intégrer robot.

4.2.2 Diagramme de séquence : Cas de suppression d'un objet

La Figure 7 suivante montre le diagramme de séquence de suppression d'un objet (obstacle ou autre) de l'environnement d'évolution des robots, cette étape aussi, nécessite l'authentification de l'administrateur afin de pouvoir mettre à jour le fichier du nœud gazebo.world, qui représente l'environnement de simulation.

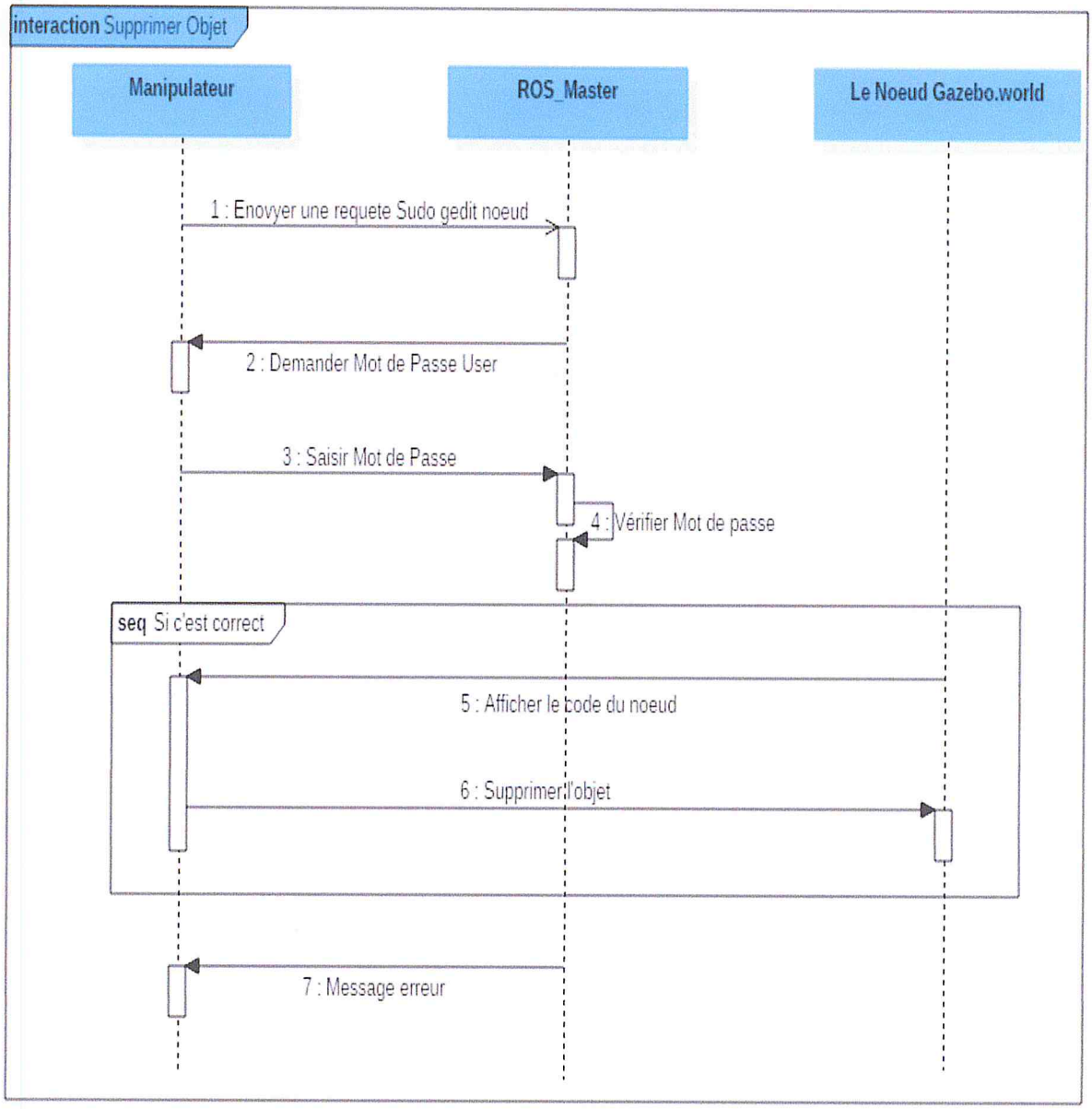


Figure 3.7 : Diagramme de séquence : supprimer objet.

4.2.3 Diagramme de séquence : Cas de lancement du simulateur

La Figure 8 suivante montre le diagramme de séquence de lancement du simulateur Gazebo d'un seul robot mobile ou d'un système multi-robots. Cette étape est faite par le manipulateur, les commandes seront données dans le chapitre 5.

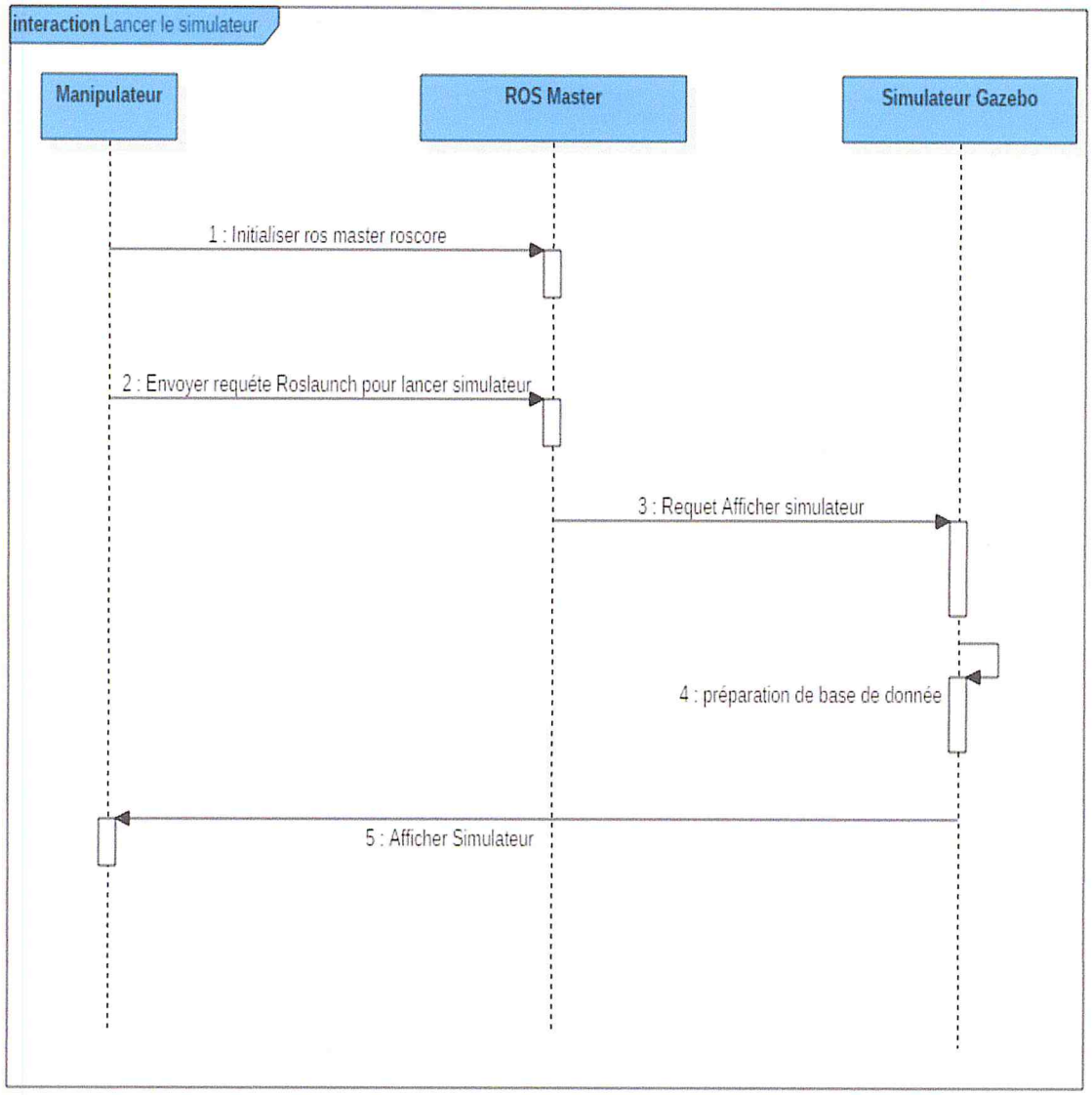


Figure 3.8 : Diagramme de séquence : lancement du simulateur

4.2.4 Diagramme de séquence : cas de télé opération d'un robot

La Figure 9 suivante montre le diagramme de séquence de manipulation d'un robot mobile par télé opération. Le robot sélectionné sera manipulé en utilisant une souris ou un Joystick, ce robot va jouer le rôle du Leader pour notre système multi-robots.

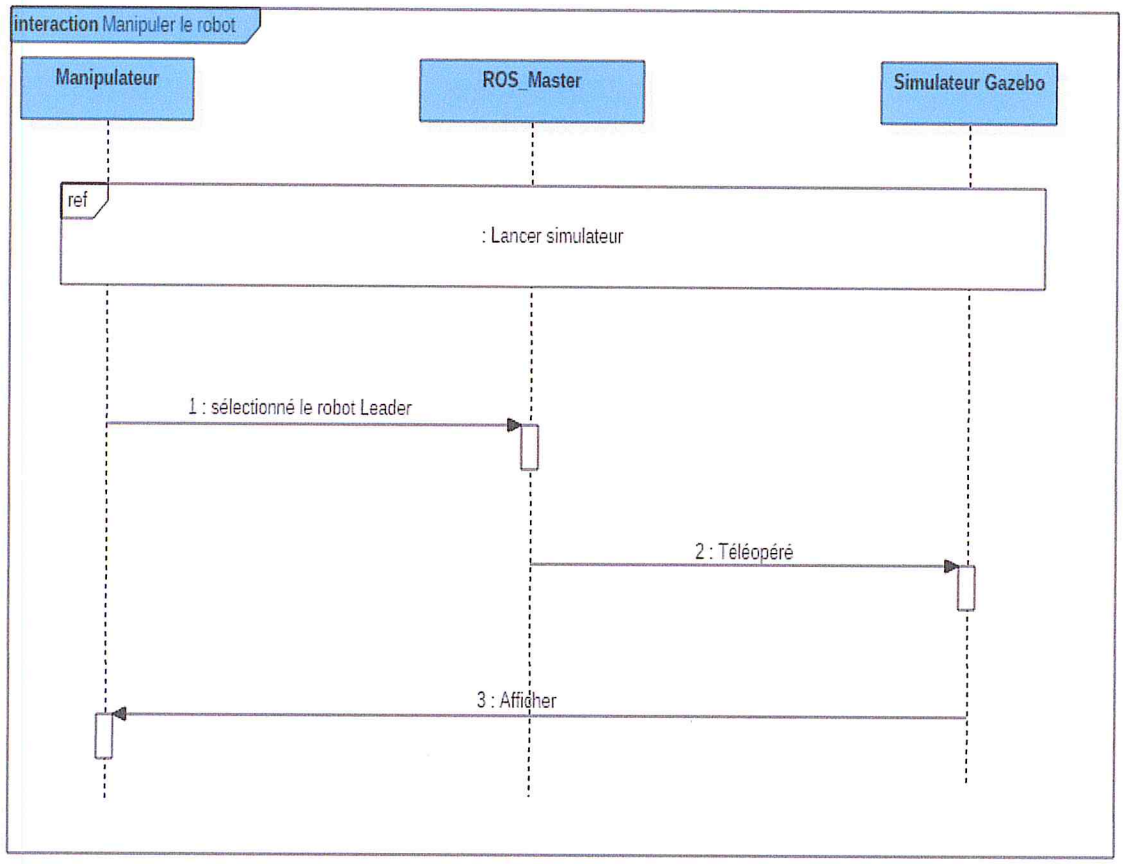


Figure 3.9 : Diagramme de séquence : cas de télé opération d'un robot.

4.2.5 Diagramme de séquence : Lancement de la tâche Leader/Follower

La Figure 10 suivante montre le diagramme de séquence de l'exécution de la tâche Leader/Follower. Après avoir sélectionné le robot Leader (étape précédente), est lorsque le simulateur Gazebo est lancé, le manipulateur va choisir les robots qui vont suivre le Leader.

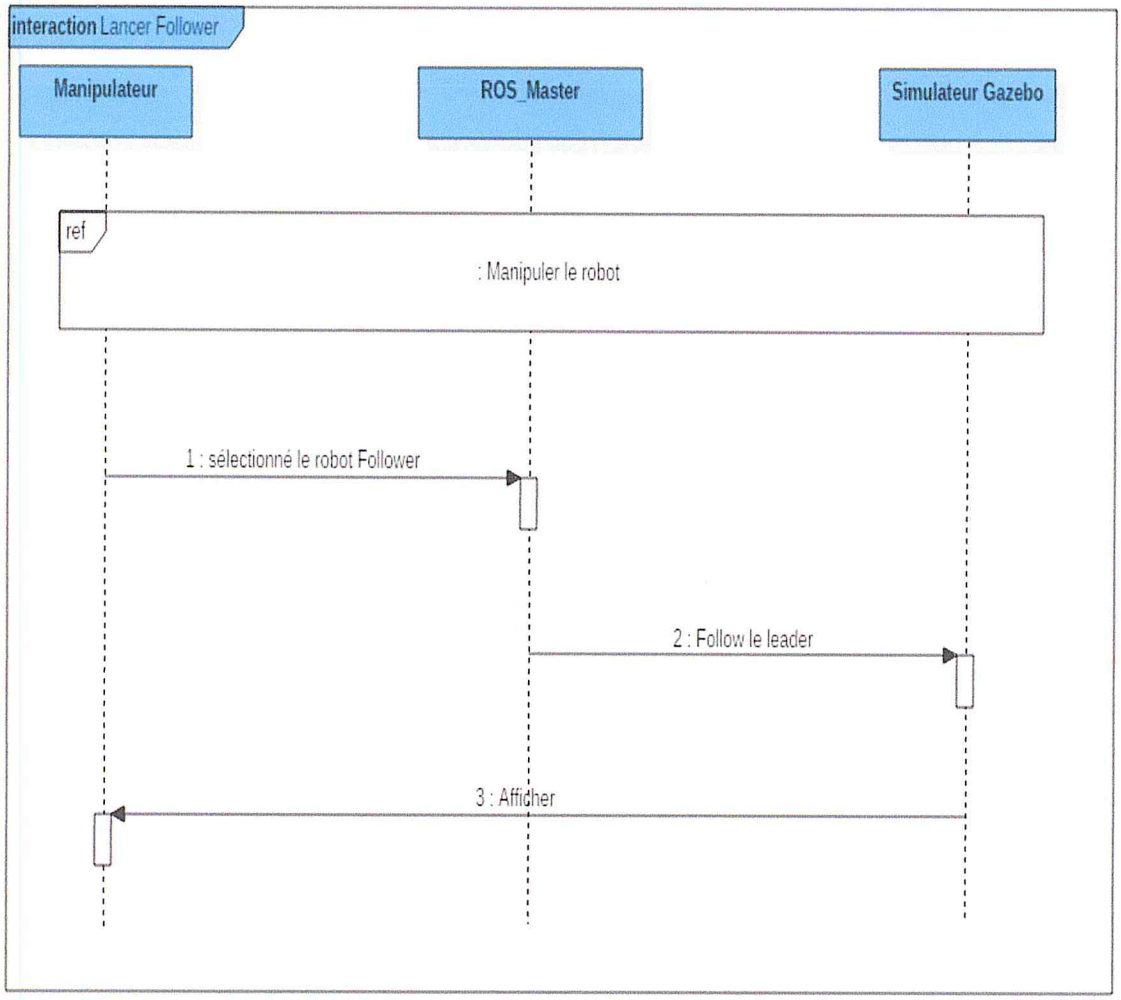


Figure 3.10 : Diagramme de séquence : Lancement de la tâche Leader/Follower.

4.3 Diagramme de classes

Le diagramme de classes constitue un élément très important de la modélisation. On constate souvent qu'un diagramme de classes proprement réalisé permet de structurer le travail de développement de manière efficace. Le diagramme de classes décrit les types des objets qui composent un système ainsi que les différents types de relations statiques existants entre eux [59].

La Figure 11 suivante montre le diagramme de classes de notre environnement de simulation.

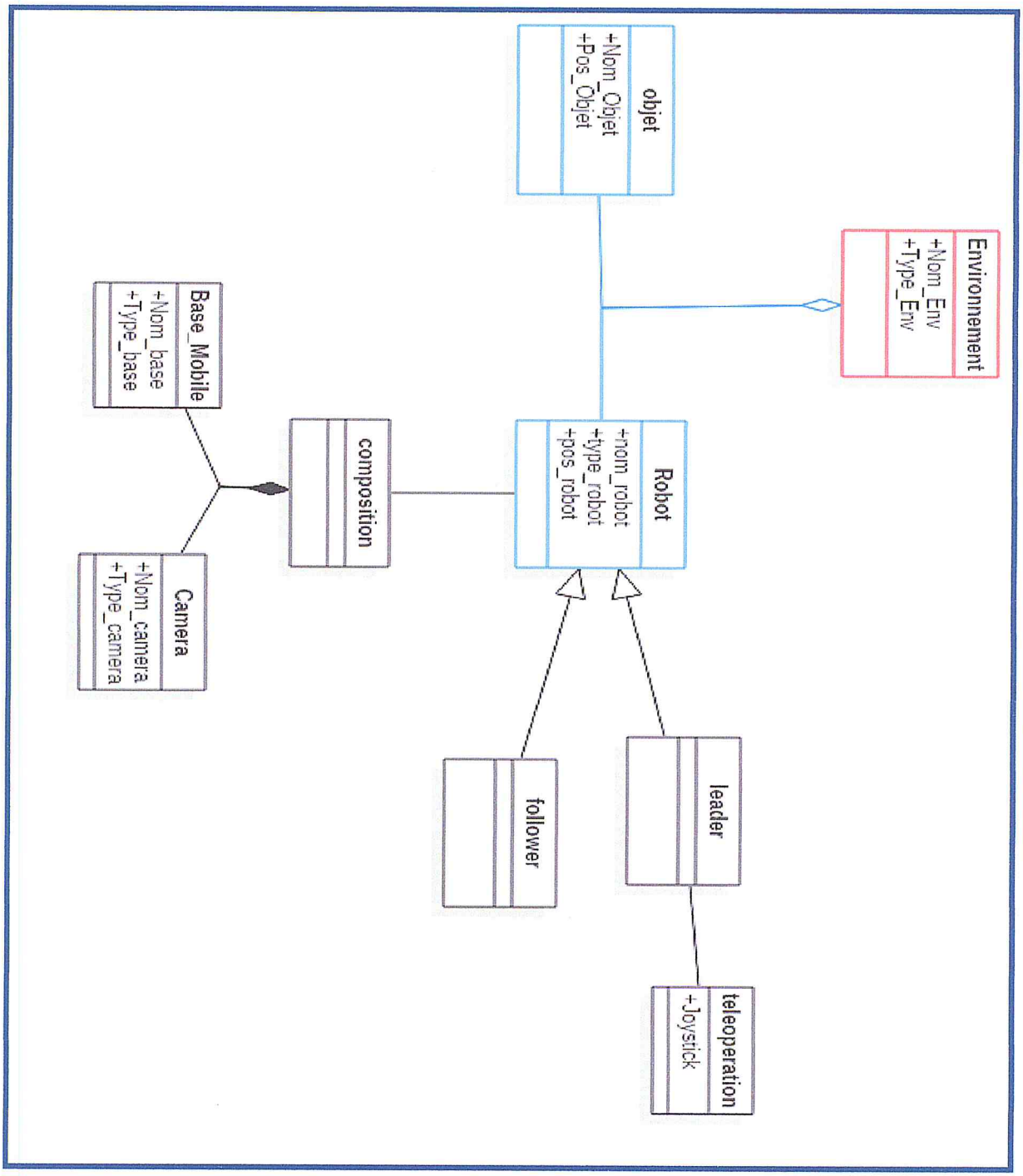


Figure 3.11 : Diagramme de classes de l'environnement de simulation.

6. Conclusion

Dans ce chapitre, nous avons adopté une démarche qui se base sur la méthode UP associée au formalisme UML afin de bien décrire l'aspect théorique du système à développer.

Cette démarche déploie trois de diagrammes UML ; chacun spécifie une partie du système à développer, à savoir : Diagramme des cas d'utilisation, digramme de séquence et enfin le diagramme de classes.

L'enchaînement et la communication entre ces diagrammes, nous a permis de bien comprendre comment modéliser, concevoir et désigner un passage souple et facile vers l'étape de réalisation de notre système qui fera l'objet du prochain chapitre.

Chapitre 4 :

Implémentation

Et

Validation

1. Introduction

Dans ce chapitre, nous allons présenter la mise en œuvre de notre application. Nous allons commencer par une présentation des outils de développement utilisés dans ce système. Par la suite, nous allons présenter le robot virtuel que nous avons utilisé ainsi que ses différentes fonctionnalités. Enfin, nous donnons quelques captures d'écran de l'exécution de notre application.

2. Outil de développement

L'environnement de travail est un choix décisif pour la réalisation d'une application. Dans ce qui suit, nous allons présenter le système d'exploitation ainsi que les différents produits Open Source choisis pour l'implémentation du système à développer.

2.1 Système d'exploitation Ubuntu

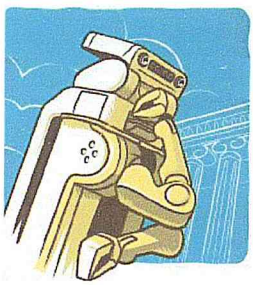
Nous avons utilisé *Ubuntu* pour la réalisation de notre application ; il s'agit d'un système d'exploitation *Linux* [60]. Dans ce travail, nous avons opté pour la version *Ubuntu 14.04* qui est une version *Long Term Support (LTS).*

Nous pouvons vérifier quelle version *Ubuntu* est utilisée en lançant la commande : *lsb_release–a* ; le résultat dans notre cas est *Description: Ubuntu 14.04.4 LTS.*

2.2 Middleware ROS

Nous avons utilisé la version *indigo* de ROS ; cette dernière est aussi *LTS.* Nous pouvons lancer la commande *roscore* pour voir il s'agit de quelle distribution ; ça nous affiche : * */rosdistro: indigo*

2.2.1 Introduction à ROS :



ROS (Robot Operating System) est un méta-système d'exploitation pour robots, peut fonctionner sur un ou plusieurs ordinateurs. Ros fournit plusieurs fonctionnalités : un simulateur 3D (Gazebo), une abstraction du matériel, contrôle de périphériques de bas niveau, SLAM (Simultaneous Localization And Mapping) qui veut dire une localisation et cartographie simultanée ainsi qu'une mise en œuvre de fonctionnalités couramment utilisées ainsi qu'une transmission de messages entre les processus et la gestion des packages installés [57].

2.2.2 Pourquoi devrions-nous apprendre ROS ?

L'utilisation de ROS peut réduire le temps de développement. ROS n'exige pas qu'on développe les systèmes et les programmes déjà existants, et comme ses codes peuvent être écrits en C++ et Python, ROS peut facilement transformer un système non-ROS en un système ROS simplement avec l'insertion de quelques codes standardisés [57]. En outre, ROS fournit de divers outils et logiciels qui sont communément utilisé, ce qui réduit finalement le temps de développement et de maintenance.

2.2.3 Architecture de communication :

Ros offre une architecture souple de communication entre ses différents modules [57].

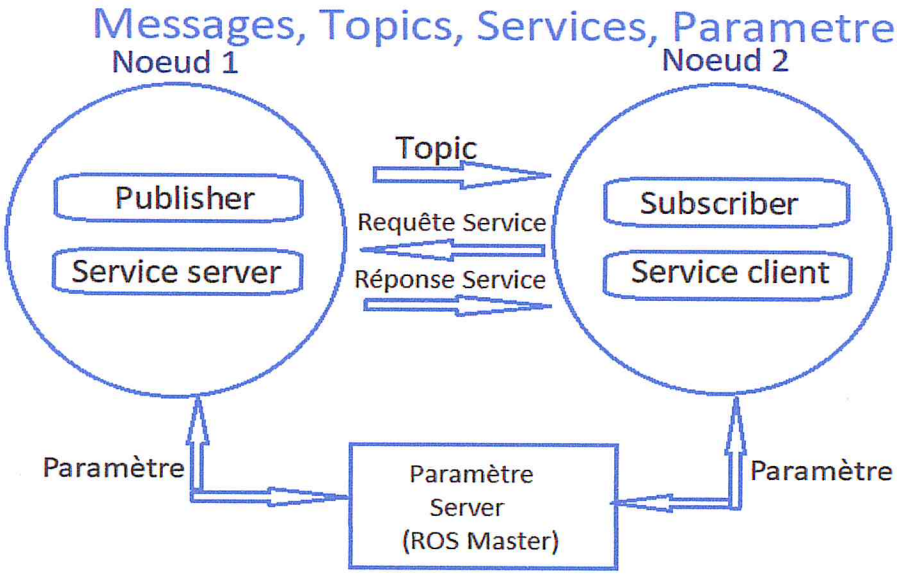


Figure 4.2 : Messages de communication entre les différents nœuds [57].

Nous allons expliquer dans cette section les termes ROS les plus fréquemment utilisés :

Paquet : Le développement sous ROS s'effectue sous forme de modules ou paquets (packages). Un développeur travaillant sous ROS peut développer un paquet ROS soumis à ses propres dépendances et contexte de compilation et d'exécution. Chaque paquet peut contenir les ressources suivantes : des fichiers sources, des librairies, des exécutables et des scripts. Un paquet ROS une fois compilé propose plusieurs programmes à exécuter. Ces programmes sont appelés des ***nœuds***.

Nœud : Un nœud ROS est un problème développé à l'intérieur d'un paquet connecté au réseau ROS. Les nœuds utilisent une librairie cliente de ROS afin de communiquer entre eux selon les spécifications et les protocoles de communication définis par le système ROS.

Par exemple, on peut développer un nœud qui a pour fonction de demander au robot de se déplacer d'un point A à un point B. Ce nœud peut alors faire appel à d'autres nœuds, chargés d'effectuer une planification de chemin ou encore d'activer les moteurs du robot. Pour chaque instance d'exécution d'un nœud ou d'un ensemble de nœuds, il est nécessaire d'exécuter préalablement le ROS Master (un paquet permettant aux différents nœuds de se trouver et de communiquer entre eux). Les nœuds peuvent publier ou s'abonner à des topics. Ils peuvent également déployer ou utiliser un service.

Topics : Un topic est un canal de communication par lequel les nœuds peuvent s'échanger des messages. Ils répondent à une architecture de type publication/inscription (publish/ subscribe) Cela permet aux nœuds de ne pas avoir à se préoccuper des autres nœuds qui communiquent avec eux : si un nœud doit partager une information, il lui suffit de déployer un topic et d'y publier des données. Le nœud intéressé par ces données n'a ensuite qu'à s'y abonner, sans avoir à connaître précisément le nœud à l'origine de l'information.

Services : Le modèle de communication adopté pour les topics est efficace et très flexible, mais il ne répond pas à tous les besoins en termes d'échange d'informations au sein d'un système distribué tel que ROS. Afin d'ouvrir une voie de communication « déconnectée », de type requête/réponse, ROS propose les services.

Un service est défini par une paire de messages : un pour la requête, l'autre pour la réponse. Un service est créé et déployé par un nœud. D'autres nœuds peuvent ensuite appeler le service via l'envoi d'une requête et ensuite attendre de recevoir la réponse.

Message : Comme nous avons déjà précisé, l'envoi et la réception de données dans le cas d'utilisation de topics et services se réalisent via l'envoi de messages.

ROS propose ainsi aux développeurs de définir, via une syntaxe particulière, leurs propres messages. Un message est une structure de données comprenant un certain nombre de champs typés. Les types classiques que l'on retrouve dans la majorité des langages de programmation : les entiers (integer), les réels (float), les booléens, ainsi que les tableaux...etc Une fois un message défini, il est également possible de l'utiliser à l'intérieur d'un autre message.

2.3 Langage de programmation Python

Il s'agit d'un langage de programmation puissant et facile à apprendre. Il dispose de structures de données de haut niveau et d'une approche de programmation orientée objet simple et efficace [61].

2.4 Langage de programmation C++

Le C++ est actuellement l'un des langages les plus utilisés dans le monde, aussi bien pour les applications scientifiques que pour le développement des logiciels [62].

3. Le simulateur Gazebo

3.1 Définition d'un simulateur : Un logiciel de simulation de robot est un environnement virtuel complet utilisé pour créer des applications des robots simples qui peuvent être construits, programmés et testés. Ces applications peuvent toujours être transférées sur un robot réel sans modification, ce qui va réduire les coûts et le temps [58]. Il existe plusieurs simulateurs dédiés aux systèmes multi-robots,

3.2 Simulateurs existants

Au cours des dernières années, les simulateurs virtuels ont joué un rôle important dans la recherche robotique tout en évitant les problèmes matériels courants, tels que la courte durée de vie de la batterie, les défaillances matérielles et les comportements inattendus ou dangereux.

Le nombre des simulateurs robot a augmenté durant les dernières années, on peut trouver des simulateurs commerciaux tels ques MRSim, RoboticsLab et V-Rep, et aussi des simulateurs Open source tels que Stage, Gazebo, Simbad, CARMEN et d'autres simulateurs gratuits dédiés aux systèmes multi-robots qui offre une simulation soit en 2D ou 3D.

Les simulateurs 2D ont été préférés aux simulateurs 3D pendant de nombreuses années, étant les derniers fortement restreints par les limitations matérielles en termes de puissance de programmation lors de la simulation physique. Mais avec les développements technologiques récents, les simulateurs 3D ont commencé à être plus utilisés. Dans la Figure suivante, nous voyons la même arène représentée dans Stage (simulateur 2D, vie réelle et Gazebo (simulateur 3D).

Figure 4.3 : Robot dans un simulateur 2D, en vie réelle et dans un simulateur 3D [58].

3.3 Le simulateur GAZEBO

Il s'agit d'un simulateur 3D utilisé uniquement sur Linux soutenu par OSRF (Open Source Robotics Foundation), Gazebo peut simuler des environnements réalistes tout en prenant en charge des simulations par odométrie et des capteurs robotiques standards comme Sonar, Laser, GPS, Caméra.., et toutes sortes de modèles de robots basiques. Les contrôleurs peuvent être écrits en C, C++, Python et Java en utilisant Player ou ROS, le code écrit est portable sur des véritables plates-formes robotiques [58].

3.4 Intégration avec La plateforme ROS

Etant Gazebo et ROS deux programmes utilisant des 'Topics' pour partager les données en cours, un pont (plugin bridge) est créé afin de permettre l'usage des différents outils et piles des deux programmes, ce plugin doit être créé et personnalisé selon les objectives de la simulation désirée tels que les choix et la création des topics nécessaires.

Figures 3.4 : les robots mobiles les plus utilisés dans Gazebo [58].

4. Le Robot Turtlebot

Dans notre simulation, nous avons utilisé un modèle de robot appelé *Turtlebot* donné par la figure 1. Il s'agit d'un robot destiné à la recherche et à l'éducation, développé par une société américaine spécialisée à partir d'un robot aspirateur modifié en guise de base mobile.

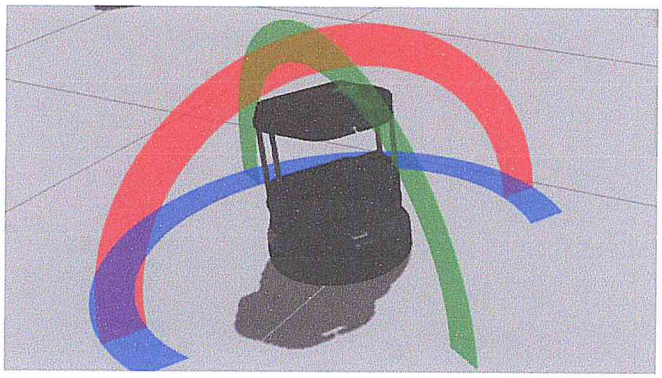


Figure 4.1 : Le robot mobile *Turtlebot*.

Ces composants les plus importants sont donnés comme suit :

- ***Une base mobile iClebo Kobuki*** : *iClebo Kobuki* est un robot low-cost possédant les caractéristiques suivantes (figure 2a) : (*i*) une vitesse de déplacement maximale de *70cm*/s, (*ii*) une vitesse de rotation maximale de *180°/s* et (*iii*) une charge utile de *10kg*.

- ***Une Kinect pour Xbox 360*** : la *Kinect Xbox 360* (figure 2b) embarquée sur le *Turtlebot* permet au robot d'avoir une vision de son environnement et remplace également efficacement un émetteur/récepteur laser. Elle possède notamment ces composants importantes : (*i*) Un capteur de perception 3D, (*ii*) Une Caméra RGB

(a) Base mobile *iClebo Kobuki* (b) Kinect *Xbox 360*

Figure 4.2 : Composants d'un robot *Turtlebot2*.

4. Implémentation

4.1 Présentation du Point Cloud

Un Point Cloud ou nuage de point est une méthode utilisée pour représenter une collection de points multi-dimensions dans le traitement de données 3D. En plus des coordonnées spatiales (x,y,z), chaque point peut également contenir des informations sur la couleur RGB. L'acquisition de ses données se fait à l'aide de capteurs tels qu'une caméra acoustique un scanner laser (Kinect dans le cas de notre robot TurtleBot) [63].

4.1.1 Appliquer la méthode du Point Cloud sur les capteurs du TurtleBot

Afin de suivre le robot Leader en toute sécurité, les robots follower ont besoin de plusieurs données sur l'espace qui les entoure et leurs propres coordonnées (position, vitesse, hauteur ...). Ces informations sont donc fournies par les capteurs installés sur le TurtleBot.

Grâce à ROS, il est facile et rapide d'obtenir un affichage en temps réel des nuages des points, permettant le robot follower de récupérer les coordonnées du robot Leader. Cette méthode met en disposition une multitude de fonction allant de la simple copie et concaténation de nuages de points jusqu'à la segmentation, comme le montre la figure suivante :

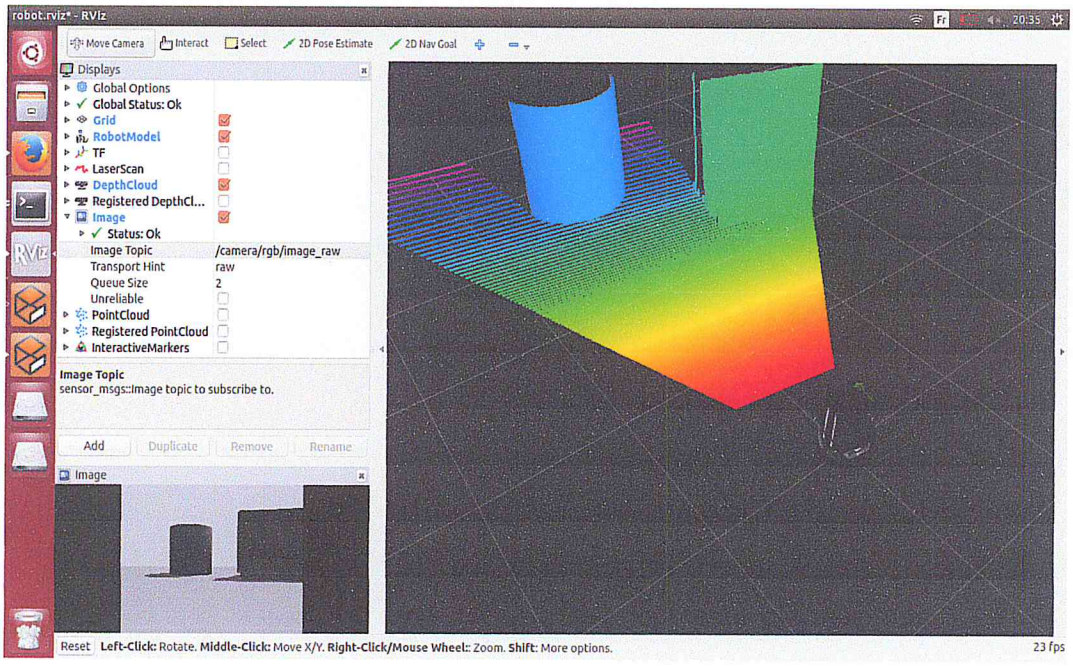


Figure 4.3 : Illustration de nuages de points dans le champ de vision du TurtleBot.

4.2 Mise en œuvre de la solution proposée :

Le nœud du robot Follower s'abonne aux données du capteur 3D.

```
namespace turtlebot_follower
{
typedef pcl::PointCloud<pcl::PointXYZ> PointCloud;

class TurtlebotFollower : public nodelet::Nodelet
{
public:

 TurtlebotFollower() : min_y_(0.1), max_y_(0.5),
                       min_x_(-0.2), max_x_(0.2),
                       max_z_(0.8), goal_z_(0.6),
                       z_scale_(1.0), x_scale_(5.0)
  {

  }

  ~TurtlebotFollower()
  {
    delete config_srv_;
  }
```

Figure 4.4 : Initialisation de la classe TurtlebotFollower

Le robot Follower ensuite, récupère les coordonnées venant du capteur 3D.

```
virtual void onInit()
{
  ros::NodeHandle& nh = getNodeHandle();
  ros::NodeHandle& private_nh = getPrivateNodeHandle();

  private_nh.getParam("min_y", min_y_);
  private_nh.getParam("max_y", max_y_);
  private_nh.getParam("min_x", min_x_);
  private_nh.getParam("max_x", max_x_);
  private_nh.getParam("max_z", max_z_);
  private_nh.getParam("goal_z", goal_z_);
  private_nh.getParam("z_scale", z_scale_);
  private_nh.getParam("x_scale", x_scale_);
  private_nh.getParam("enabled", enabled_);

  cmdpub_ = private_nh.advertise<geometry_msgs::Twist> ("cmd_vel", 1);
  markerpub_ = private_nh.advertise<visualization_msgs::Marker>("marker",1);
  bboxpub_ = private_nh.advertise<visualization_msgs::Marker>("bbox",1);
  sub_= nh.subscribe<PointCloud>("depth/points", 1, &TurtlebotFollower::cloudcb, this);

  switch_srv_ = private_nh.advertiseService("change_state", &TurtlebotFollower::changeModeSrvCb, this);

  config_srv_ = new dynamic_reconfigure::Server<turtlebot_follower::FollowerConfig>(private_nh);
  dynamic_reconfigure::Server<turtlebot_follower::FollowerConfig>::CallbackType f =
      boost::bind(&TurtlebotFollower::reconfigure, this, _1, _2);
  config_srv_->setCallback(f);
}
```

Figure 4.5 : Récupération de données du capteur 3D du robot.

4.2.1 La classe PCL : Cette classe permet également d'extraire les BoundingBox (les boîtes de délimitation) de l'axe du nuage de points [64].

Si le robot leader est dans le champ de vision du robot follower, alors il le suit jusqu'à ce que le leader ne soit plus dans le champ de vision du follower, dans ce cas, le robot follower s'arrête automatiquement.

Note : Rappelons que le champ de vision de la caméra Kinect est étendu entre 0.6 et 4 mètres (pas de vision de près).

```
if (n>4000)
{
  x /= n;
  y /= n;
  if(z > max_z_){
    ROS_DEBUG("No valid points detected, stopping the robot");
    if (enabled_)
    {
      cmdpub_.publish(geometry_msgs::TwistPtr(new geometry_msgs::Twist()));
    }
    return;
  }

  ROS_DEBUG("Centroid at %f %f %f with %d points", x, y, z, n);
  publishMarker(x, y, z);

  if (enabled_)
  {
    geometry_msgs::TwistPtr cmd(new geometry_msgs::Twist());
    cmd->linear.x = (z - goal_z_) * z_scale_;
    cmd->angular.z = -x * x_scale_;
    cmdpub_.publish(cmd);
  }
}
```

Figure 4.6 : Si le Leader est dans le champ de vision du follower

```
else
  {
    ROS_DEBUG("No points detected, stopping the robot");
    publishMarker(x, y, z);

    if (enabled_)
    {
      cmdpub_.publish(geometry_msgs::TwistPtr(new geometry_msgs::Twist()));
    }
  }

  publishBbox();
}

bool changeModeSrvCb(turtlebot_msgs::SetFollowState::Request& request,
                     turtlebot_msgs::SetFollowState::Response& response)
{
  if ((enabled_ == true) && (request.state == request.STOPPED))
  {
    ROS_INFO("Change mode service request: following stopped");
    cmdpub_.publish(geometry_msgs::TwistPtr(new geometry_msgs::Twist()));
    enabled_ = false;
  }
  else if ((enabled_ == false) && (request.state == request.FOLLOW))
  {
    ROS_INFO("Change mode service request: following (re)started");
    enabled_ = true;
  }

  response.result = response.OK;
  return true;
}
```

Figure 4.7 : Si le leader s'éloigne du champ de vision du follower

4.2.2 Rocon APPlication

Afin de faciliter la tâche à l'utilisateur de notre système, nous avons mis en place une interface qui fournit un cadre permettant d'établir des interactions entre les utilisateurs humains et le ROS_MASTER qui est en cours d'exécution. Pour cela on a utilisé le RAPP, il s'agit d'une application de robot ou (rocon). C'est une entité exécutée par le gestionnaire d'application du robot. Généralement, un rapp est simplement un lanceur ros avec une spécification de rapp (métadonnées, icône, interface publique). Pas de code, juste des dépendances sur les paquets de code. Le Rapp fournit une fenêtre GUI agréable pour visualiser les positions des articulations en tant que curseurs

Avec les interactions *rocon*, tout ce que l'utilisateur doit faire est de pointer un *remocon* sur un master ros en cours d'exécution, choisir un rôle et ils seront présentés avec un certain nombre de façons d'interagir avec le concert. Cela fonctionne pour de nombreux types d'interactions.

5. Test et Validation de l'Application

5.1 Etape de lancement de l'application

Pour exécuter ROS, il faut lancer la commande *roscore* (figure 8) ; le master sera configuré avec une adresse URI utilisant l'adresse IP du PC, ainsi qu'un numéro de port 11311 par défaut. Cette commande permet aux nœuds du master de pouvoir communiquer entre eux.

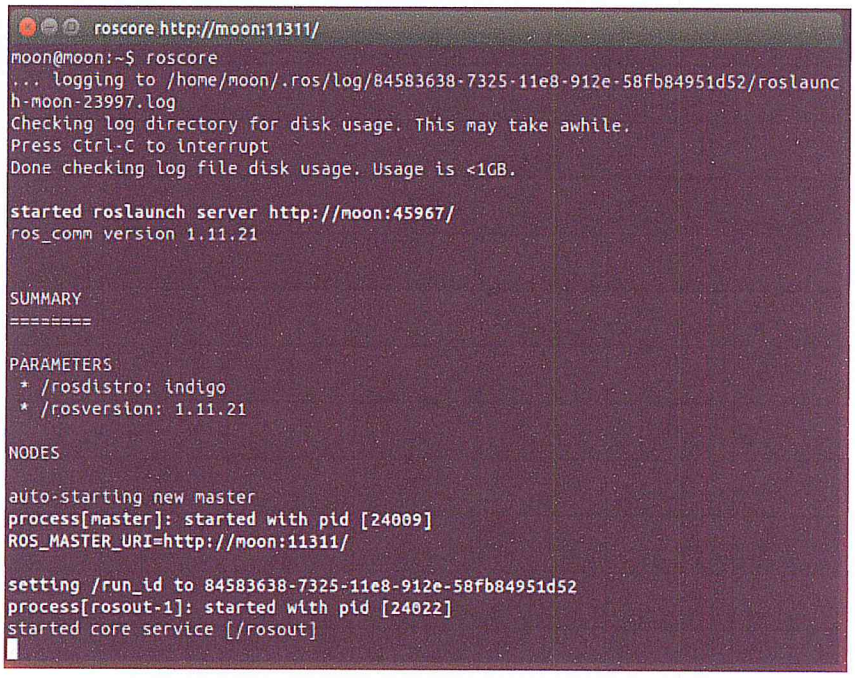


Figure 4.8 : Résultat de la commande *roscore*.

Lors de cette étape, il faut construire un nouveau package, dans le dossier *my_ws*, en utilisant les commandes suivantes (figure 9) :

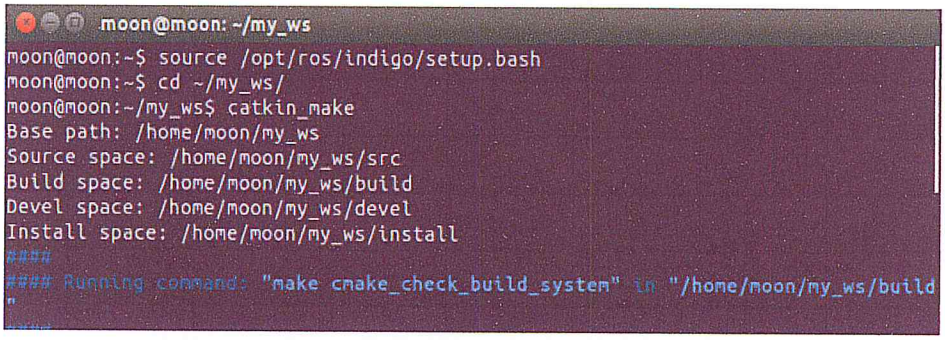


Figure 4.9 : Construire un Package.

5.1.1 Lancement d'un robot *Turtlebot* dans l'environnement *Gazebo*

Lors de cette phase, il y a lieu de lancer un *Turtlebot* dans l'environnement *Gazebo*. Pour cela, il faut ouvrir un nouveau terminal et exécuter la commande roslaunch turtlebot_gazebo turtlebot_world.launch. La figure 10 illustre l'environnement *Gazebo* avec le robot *TurtleBot* ainsi que d'autres objets.

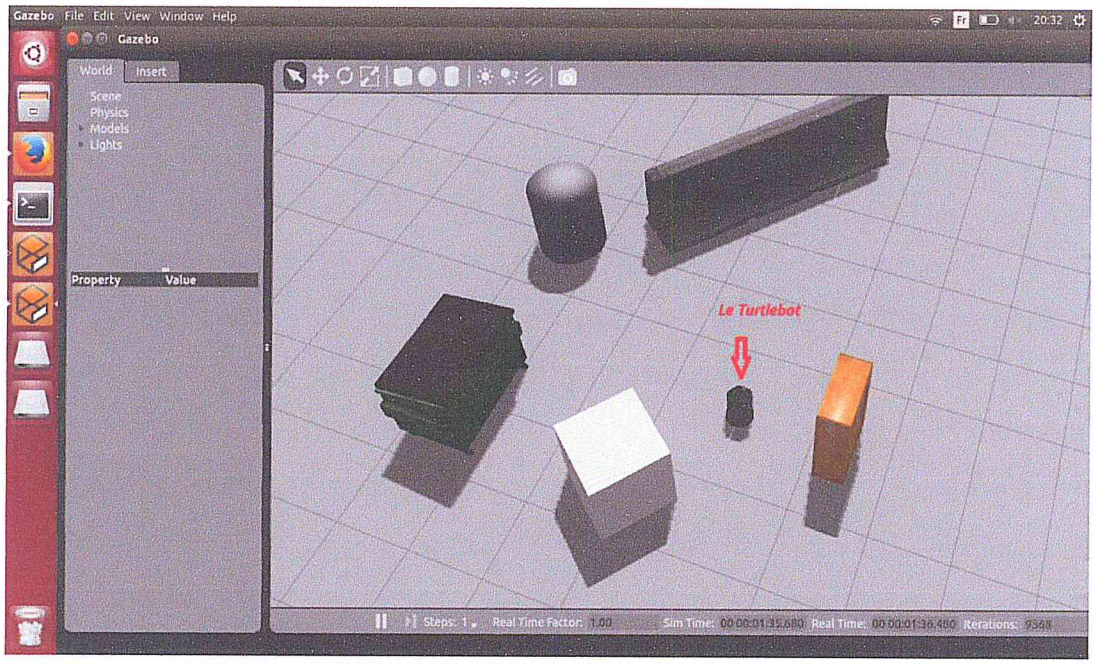


Figure 4.10 : Le robot *TurtleBot* dans l'environnement *Gazebo*.

TurtleBot utilise *Kinect* pour visualiser l'environnement en 3D, détecter et suivre les objets. Pour cela, il faut alors activer le capteur 3D du robot avec la commande *TURTLEBOT_3D_SENSOR=kinect* (figure 11).

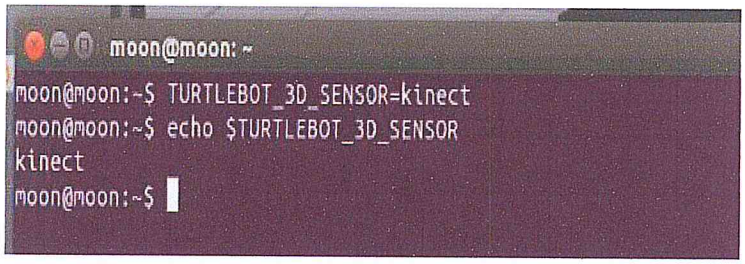
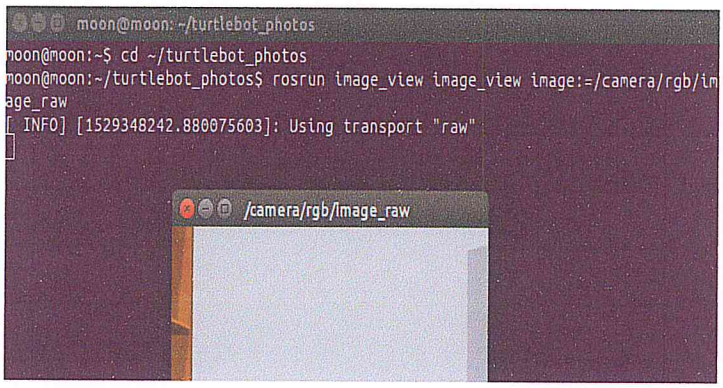
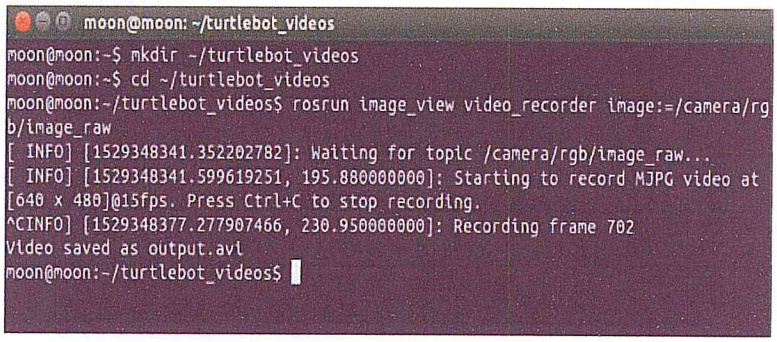


Figure 4.11 : Réglage du capteur 3D du TurtleBot.

Pour tester la Kinect, il faut prendre une image et enregistrer une vidéo. Ainsi, deux dossiers doivent être créés ; un premier *turtlebot_photos* pour stocker les images (figure 12a) et un deuxième *turtlebot_videos* pour sauvegarder les vidéos. De même pour les vidéos, on lance la commande *rosrun* et on clique sur Ctrl+C pour arrêter et enregistrer la vidéo (figure 12b).



(a) Acquisition d'images avec le capteur Kinect.



(b) Enregistrer une vidéo avec le capteur Kinect.

Figure 4.12 : Acquisition d'images et de vidéos par une caméra Kinect

5.1.2 Lancement de *Rviz*

Il s'agit d'un environnement de visualisation 2D pour ROS. La visualisation et la journalisation des informations capteurs constituent un élément important du développement et du débogage.

Le lancement de *Rviz* se fait pendant que *Gazebo* est aussi lancé, il suffit de taper la commande *roslaunch turtlebot_rviz_launchers view_robot.launch.* Le résultat est donné par la figure 13 qui illustre la perception de la profondeur donnée par Rviz.

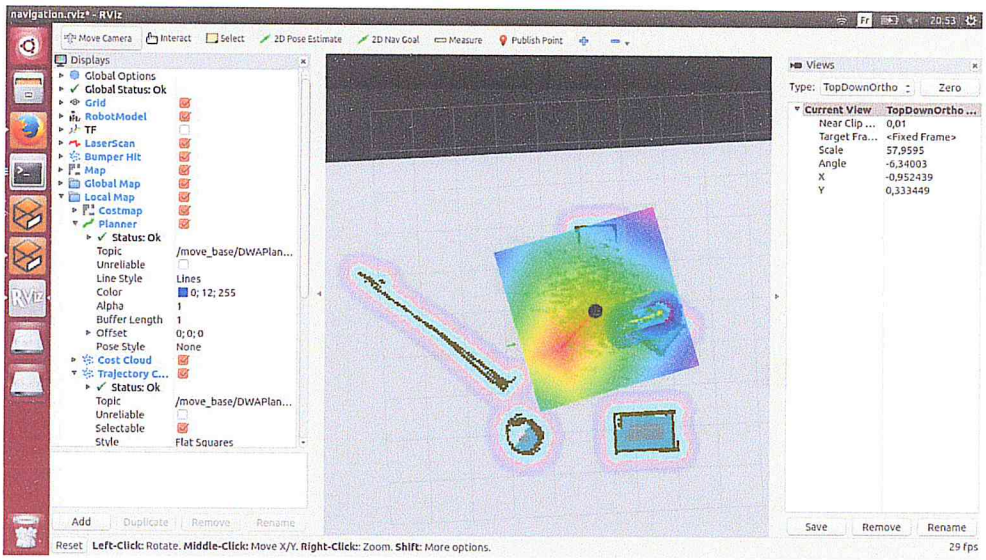


Figure 4.13 : Environnement visualisé dans *Rviz.*

5.1.3 Intégration d'autres robots *TurtleBot*

À cette phase, il y a lieu de dupliquer le robot *TurtleBot* et essayer de lancer trois robots mobiles simultanément. Pour cela, nous utilisons le package *Robotic In Concert*, et modifions le fichier *gazebo.parameters*. L'accès à ce fichier se fait avec la commande *sudo* qui permet d'exécuter d'autres commandes avec des privilèges élevés (figure 9). L'utilisateur sera ensuite invité à entrer son mot de passe ; s'il est correct, l'opération sera effectuée avec succès.

Figure 4.14 : La commande *sudo*.

Une fois le fichier *gazebo.parameters* est ouvert, deux autres robots sont intégrés en spécifiant le nom, le type et la position de chacun (voir figure 10 pour plus de détails).

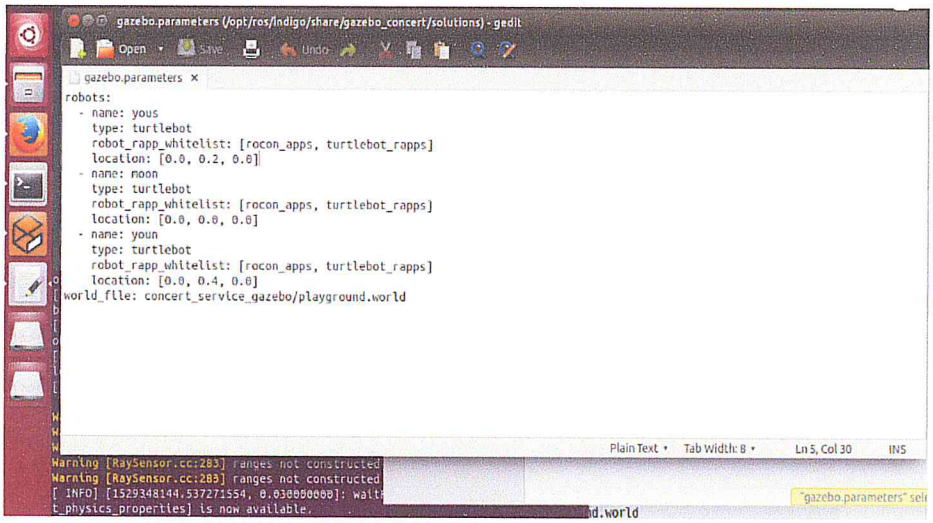


Figure 4.15 : Intégration de deux autres robots.

Par la suite, nous lançons la commande *roslaunch gazebo_concert concert.launch* ; le résultat est donné par la figure 16 :

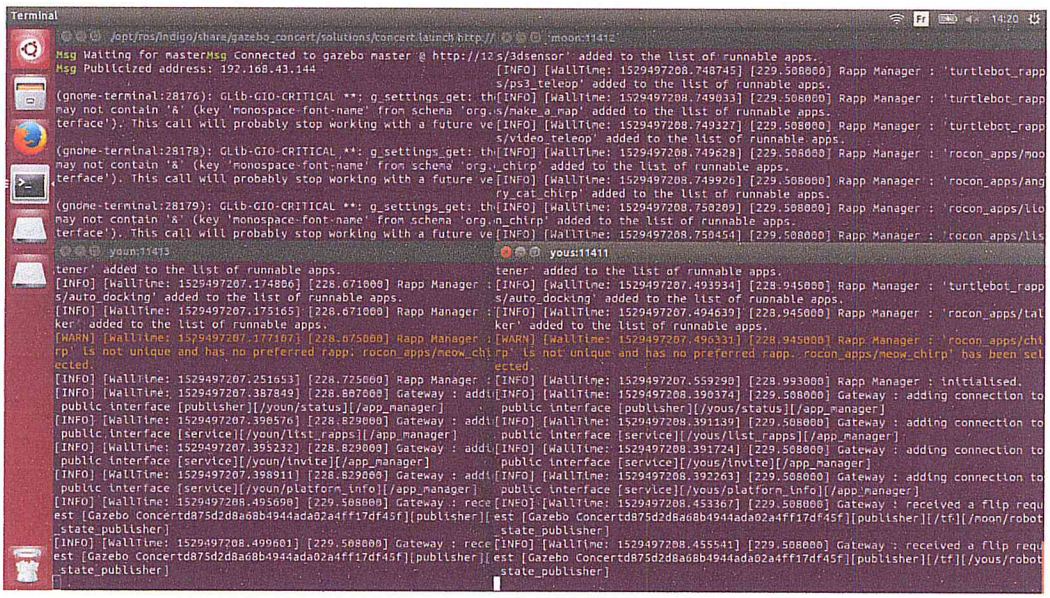


Figure 4.16 : Lancement de la commande *roslaunch*.

Puis, nous ajoutons *MASTER_URI* à la liste des masters, après avoir précisé l'adresse du *localhost* http://localhost:11311, pour que tous les nœuds puissent communiquer entre eux (figure 17).

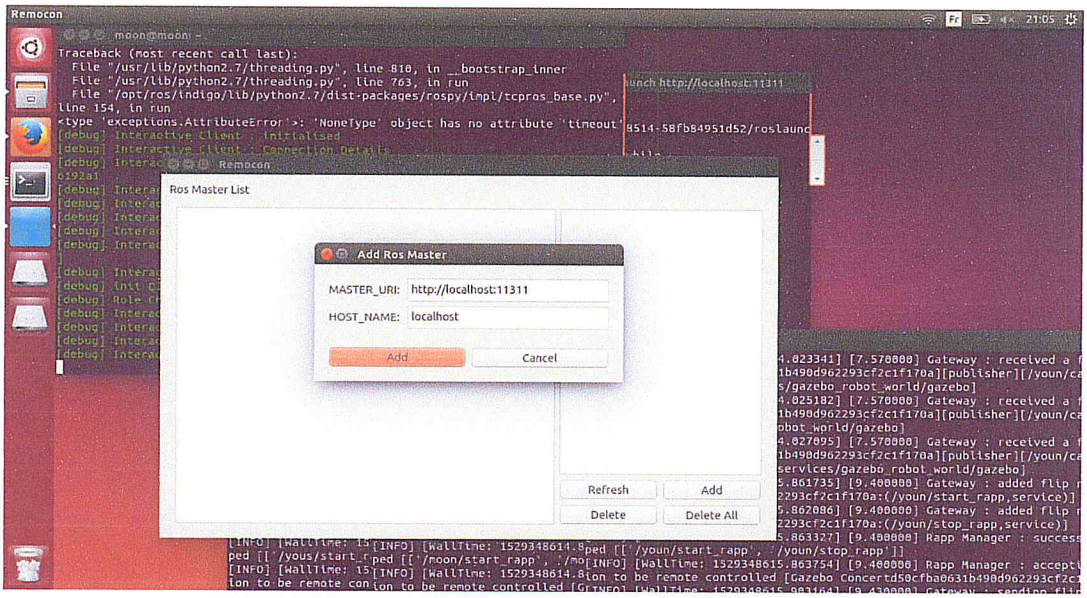


Figure 4.17 : Interface liste des *ROS MASTER*

Par la suite, il faut sélectionner *Gazebo Concert* pour visualiser la simulation des robots dans leur environnement (figure 18) :

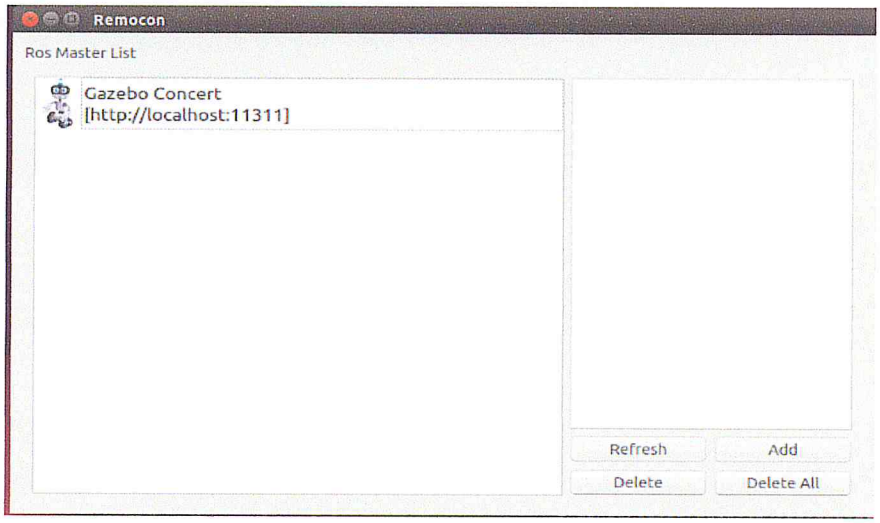


Figure 4.18 : Gazebo concert avec l'adresse du *localhost*.

La liste suivante illustrée par la figure 19 apparaît ; le manipulateur peut soit (*i*) consulter l'état du système, (*ii*) lancer la simulation dans l'environnement *Gazebo* soit (*iii*) manipuler les robots par télé opération.

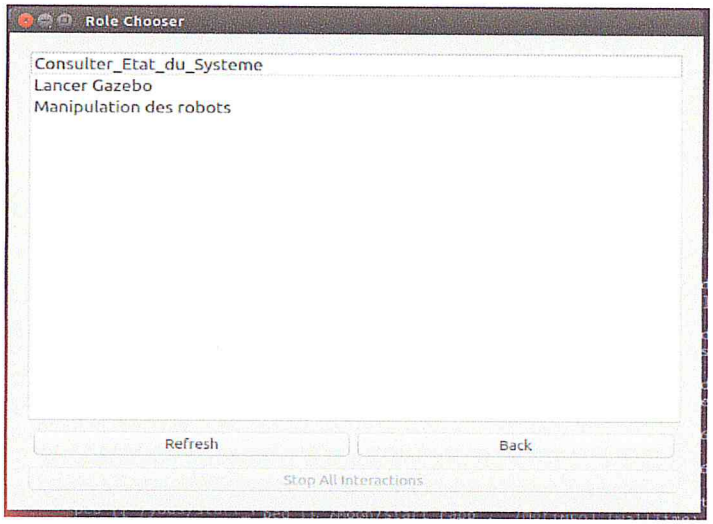


Figure 4.19 : Liste des choix.

Si nous choisissons de *Lancer Gazebo* afin de visualiser le résultat de la simulation, nous aurons les trois robots *TurtleBot* (figure 20).

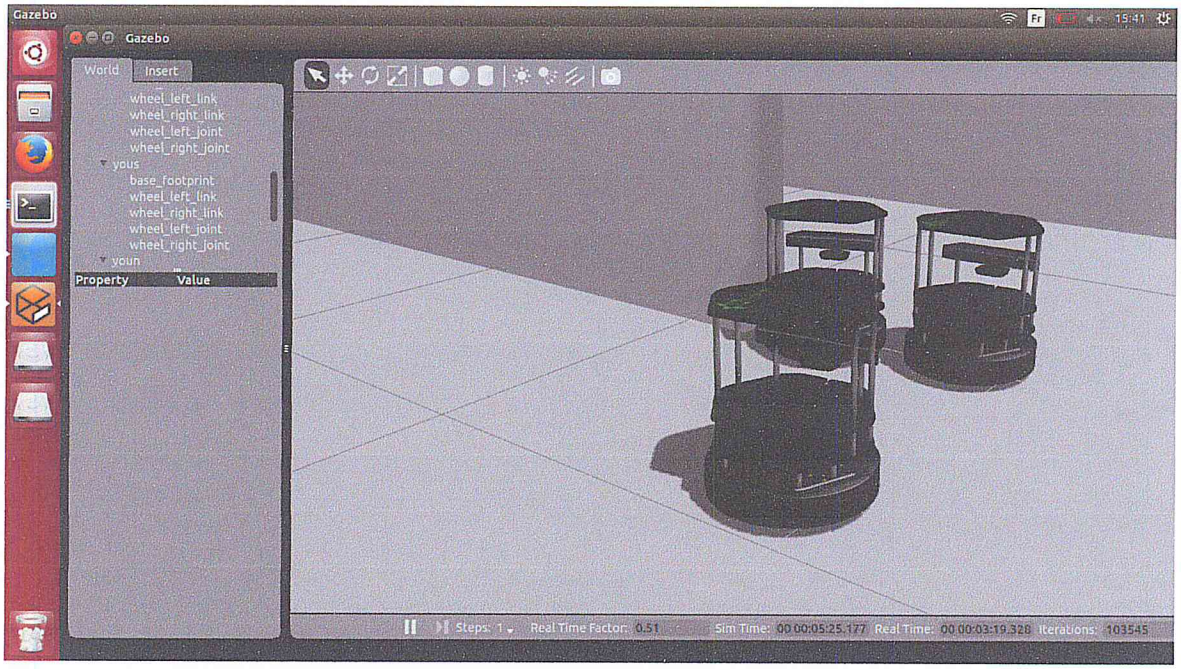


Figure 4.20 : Trois robots *TurtleBot* dans l'environnement *Gazebo*.

5.1.4 Désignation du robot Leader

Si par contre, nous avons sélectionné *Manipulation des robots*, la liste des robots existants apparaît (figure 21). À cette phase, il faut sélectionner le robot à manipuler en utilisant un Joystick ou une souris. Le robot sélectionné jouera le rôle du *Leader*. Aussi, nous pouvons visualiser le champ de vision de ce robot.

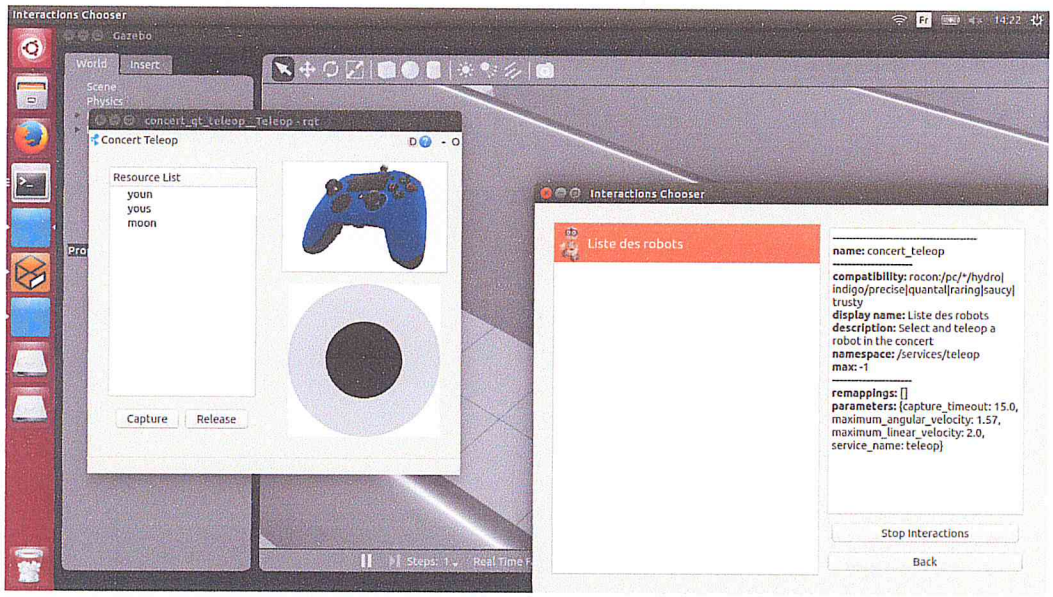


Figure 4.21 : Sélection et Manipulation du robot *Leader*.

5.1.5 Lancement de l'opération Leader/Follower

Une fois le robot *Leader* est choisi, nous pouvons alors sélectionner les autres robots followers et lancer la tâche *Leader/Follower*. Pour cela, il faut ouvrir un nouveau terminal et lancer la commande *roslaunch turtlebot_follower follower.launch simulation:=true __ns:=********, tout en spécifiant le nom du robot follower (figure 22).

Note : les étoiles **** représentent le nom (name) du robot follower.

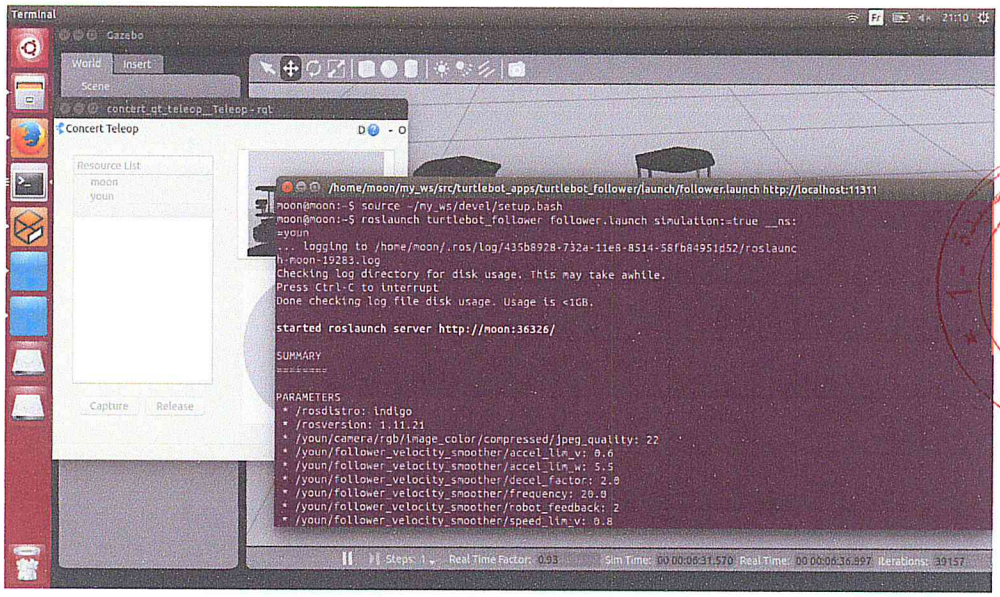


Figure 4.22 : Désignation des robots followers.

Les trois robots se trouvent à leurs positions initiales respectives et le robot *Leader* peut être manipulé en utilisant le joystick vers la destination finale souhaitée. Simultanément, nous constatons que, tant que le robot Leader se trouve dans le champ de vision des followers, ces derniers le suivent tout en gardant une certaine distance.

5.1.6 Création d'un environnement *Labyrinthe*

Dans ce scénario, nous avons créé notre propre environnement nommé *Labyrinthe*. La figure 23 montre les trois robots *TurtleBot*, un leader et deux followers, dans cet environnement.

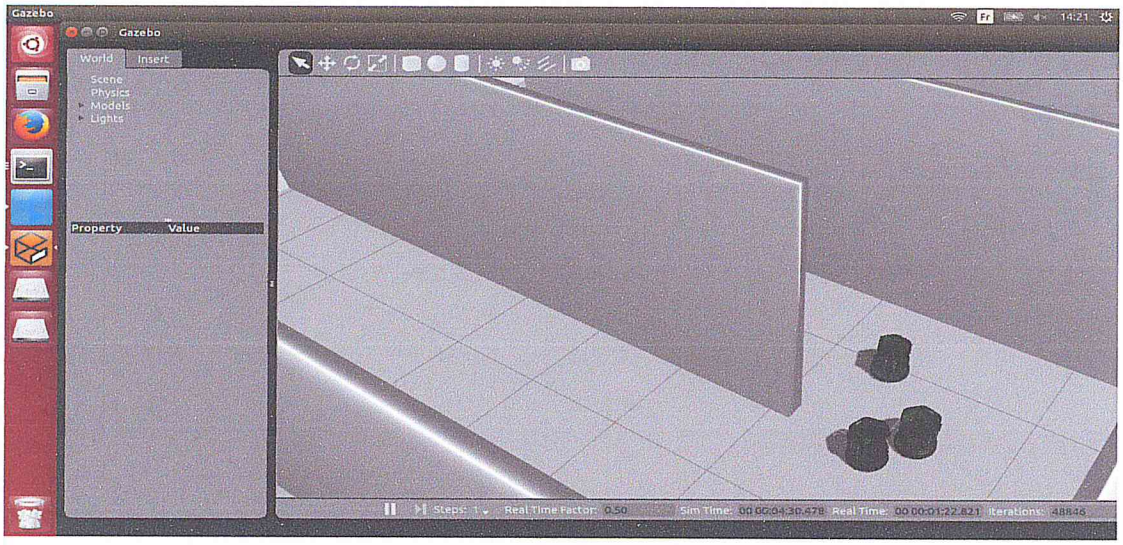


Figure 4.23 : Les trois robots *TurtleBot*, un leader et deux followers, dans le *Labyrinthe*.

5.1.7 Consultation de l'état du système développé

À ce stade, nous pouvons toujours consulter l'état du système (Figure 24). On peut choisir de voir un de ces deux graphes :

ROS graph (Figure 25 (a) et Gateway Graph (Figure 25 (b)).

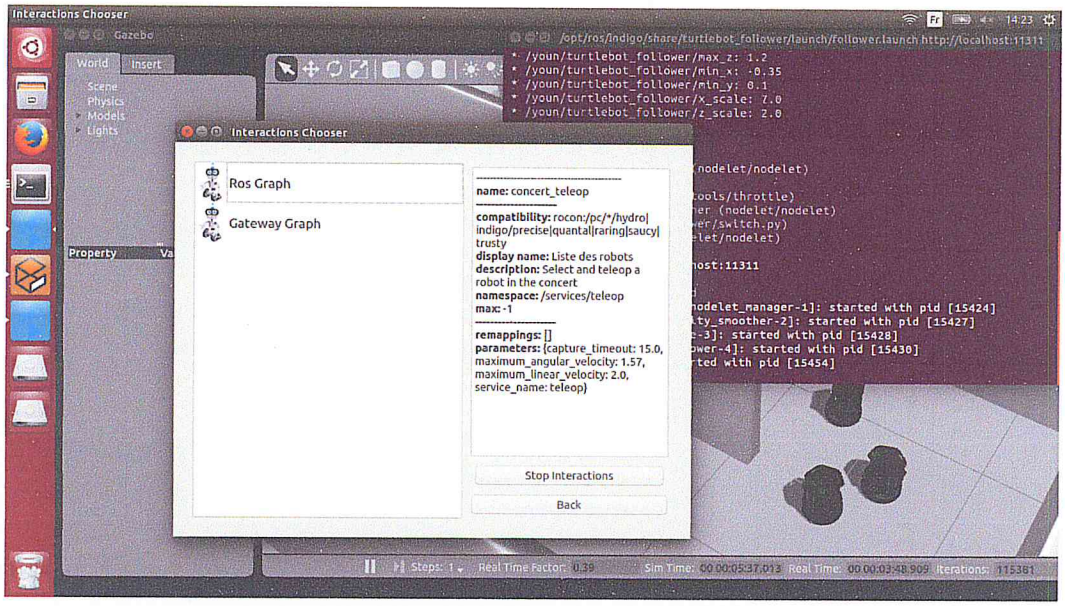
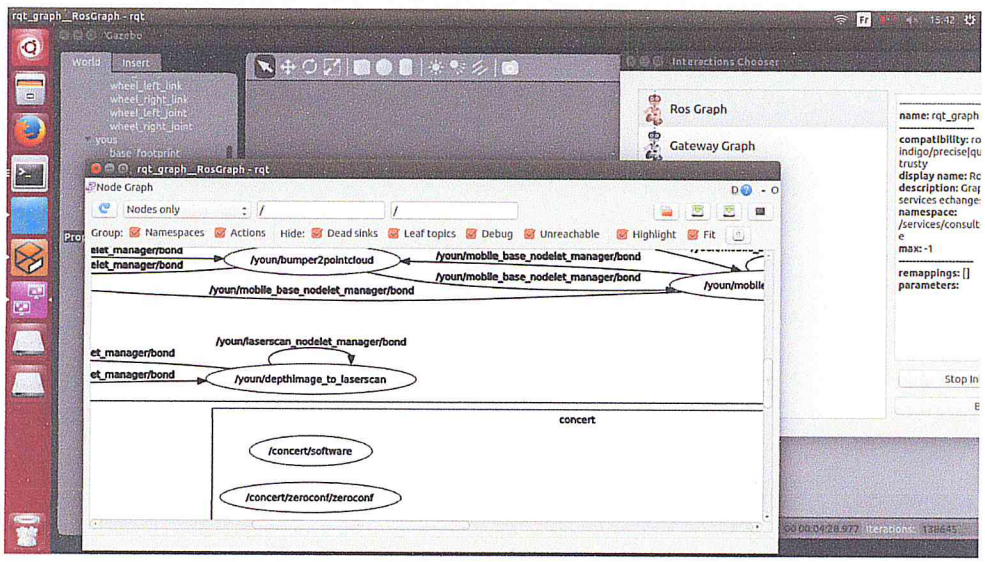
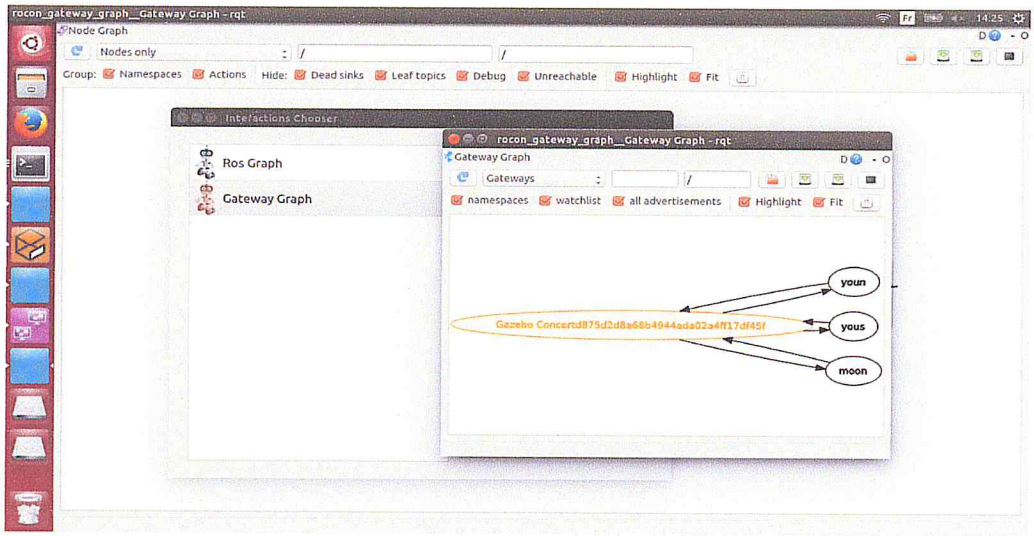


Figure 4.24 : Consulter l'état du système.



(a) ROS GRAPH

Gateway Graph : Si tout fonctionne correctement, le Gateway Graph nous affiche la liste de tous les robots du système.



(b) Gateway Graph

Figure 4.25 : les graphes du système.

6. Conclusion

Dans ce chapitre, nous avons montré comment il nous était possible de concrétiser la conception de la solution proposée. Nous avons commencé par définir les outils et les méthodes qu'on a utilisé pour développer notre système, ainsi en examinant les résultats obtenus de notre simulation, nous avons pu affirmer la validité de la solution que nous avons proposé et implémentée pour pouvoir réaliser ce travail.

Conclusion Générale

Conclusion Générale

Le domaine de la robotique fait face à une croissance rapide dans la complexité des besoins et des exigences pour les systèmes multi-robots chargés de tâches multiples, capables de coordonner leurs actions. En parallèle, une évolution similaire dans les systèmes d'exploitation des robots qui jouent le rôle d'un middleware entre les chercheurs et leurs robots, en offrant la possibilité d'effectuer des simulations des robots (manipulateurs ou mobiles), ainsi que des environnements (virtuels ou déjà existants).

Un des problèmes persistants dans le domaine de la robotique était la coopération des systèmes multi-robots, ce problème concerne la capacité du système à collaborer pour accomplir une tache spécifique. Une des approches les plus répandues est l'approche Leader/Follower, actuellement utilisée pour contrôler les formations des systèmes multi-robots. Dans cette approche, un des robots jouent le rôle du Leader qui génère la trajectoire pour les autres robots qui agissent comme des suiveurs (followers) qui doivent garder la séparation et le relèvement souhaité par rapport au Leader.

Le présent travail consistait à contrôler une équipe de robots mobiles homogènes en formation Leader/Follower. L'objectif global était d'avoir une solution permettant de simuler la manipulation d'un système multi-robots suivant cette formation.

Dans le premier chapitre, nous avons présenté d'une manière globale la robotique, plus précisément la robotique mobile et collective, et donné un état de l'art des systèmes multi-robots.

Dans le deuxième chapitre, nous avons étudié le problème de contrôle de formation des systèmes multi-robots et les différentes approches concernant ce domaine, en particulier, l'approche Leader/Follower.

Dans le troisième chapitre, nous avons proposé deux phases :

- Une phase d'analyse qui reprend plus en détails les fonctionnalités de chaque acteur de notre système.
- Une phase de conception vient consolider l'ensemble, en donnant une vue globale sur le système par les diagrammes dessinés.

Enfin, le quatrième chapitre a été consacré entièrement à l'implémentation et la validation de notre système développé.

Nous avons testé notre solution via un le simulateur Gazebo, ou nous avons créé notre propre environnement afin de pouvoir effectuer la manipulation du système multi-robots. Les résultats obtenus de la simulation ont montré l'efficacité de la solution proposée

Les perspectives pour d'éventuels travaux futurs concernent essentiellement l'implémentation et la validation expérimentale de cette simulation dans un environnement réel, il y a lieu de considérer un grand nombre de robots homogènes, et dans un environnement complexe (plusieurs objets/obstacles).

Enfin, la réalisation de ce projet, nous a permis, du coup, d'appliquer toutes les connaissances acquises durant nos études à l'université, et d'apprendre à utiliser de nouveaux outils et nouvelles technologies. Ainsi notre stage de fin d'étude au CDTA représente une expérience professionnelle et humaine des plus enrichissantes. L'immersion dans le milieu professionnel nous a permis en effet de découvrir un environnement social et culturel différent de celui auquel nous sommes habitués. L'expérience nous a comblés, et cela sur tous les plans.

Références bibliographiques

[1] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer. A fast and incremental method for loopclosure detection using bags of visual words. IEEE Transactions On Robotics, Special Issue on Visual SLAM, 2008.

[2] LAOUICI, Z. (2013). La Robotique & les différentes méthodes de navigation pour un robot mobile, mémoire de master, université d'Oran.

[3] Elaboration d'une stratégie de coordination de mouvements pour un manipulateur mobile redondant, *par* Isma Akli , USTHB - Magister 2007

[4] M.J. Aldon and L. Le Bris, 'Mobile robot localization using a light-stripe sensor'. Proceedings of the Intelligent Vehicles Symposium, pp. 255-259, Paris, France, October 24-26, 1994.

[5] J.-Y. Fourquet et M. Renaud. Coordinated Control of a Non-Holonomic Mobile Manipulator. In ISER'1999, pages 115–125, Sydney, Australie, mars 1999.

[6] G. Campion, G. Bastin et B. D'Andr´ea-Novel. Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots. IEEE Transactions on Robotics and Automation, vol. 12, no. 1, pages 47–62, 1996.

[7] Dudek, Gregory, Jenkin, Michael R. M., Milios, Evangelos, and Wilkes, David (1996), A taxonomy for multi-agent robotics, Autonomous Robots, 3, 375{397.

[8] Parker, Lynne E. (2000), Distributed Autonomous Robotic Systems 4, Ch. Current State of the Art in Distributed Robot Systems, pp. 3-12,Springer.

[9] G. Beni and J. Wang, Swarm intelligence in cellular robotic systems. In NATO Advanced Workshop on Robots and Biological Systems, Il Ciocco, Tuscany, Italy, 1989.

[10] B. K. Panigrahi, Y. Shi, and M.-H. Lim (eds.): Handbook of Swarm Intelligence. Series: Adaptation, Learning, and Optimization, Vol 7, Springer-Verlag Berlin Heidelberg, 2011. ISBN 978-3-642-17389-9.

[11] C. Blum and D. Merkle (eds.). Swarm Intelligence – Introduction and Applications. Natural Computing. Springer, Berlin, 2008.

[12] Arai, Tamio, Pagello, Enrico, and Parker, Lynne E. (2002), Editorial: Advances in Multi Robot Systems, IEEE Transactions on Robotics and Automation, 18(5), 655{661.

[13] Iocchi, Luca, Nardi, Daniele, and Salerno, Massimiliano (2001), Reactivity and Deliberation: A Survey on Multi-Robot Systems, Lecture Notes in Computer Science, 2103, 9{34.

[14] Gerkey, Brian P. and Matari¶c, Maja J (2003), Multi-Robot Task Allocation: Analyzing the Complexity and Optimality of Key Architectures, In Proceedings of the IEEE International Conference on Robotics and Automation, Taipei, Taiwan.

[15] Dias, M Bernardine, Zlot, Robert Michael, Kalra, Nidhi, and Stentz, Anthony (Tony) (2005), Market-Based Multirobot Coordination: A Survey and Analysis, (Technical Report CMU-RI-TR-05-13), Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

[16] R. A. Brooks & A. M. Flynn. Fast, cheap and out of control : A robot invasion of the solar system. pp. 478–485, 1989.

[17] Cao, Y. Uny, Fukunaga, Alex S., and Kahng, Andrew B. (1997), Cooperative Mobile Robotics: Antecedents and Directions, Autonomous Robots, 4(1), 7{23.

[18] Arkin, Ronald C. and Balch, Tucker (1998), Cooperative multiagent robotic systems, In Kortenkamp, David, Bonasso, R.P., and Murphy, R., (Eds.), Arti¯cial Intelligence and Mobile Robots, Cambridge, MA: MIT/AAAI Press.

[19] A. Farinelli, L. Iocchi & D. Nardi. Multirobot systems : a classification focused on coordination. pp. 2015–2028, 2004.

[20] L. Adouane. Architectures de Contrôle Comportementales et Réactives pour la Coopération d'un Groupe de Robots Mobiles Architecture de contrôle hybride pour systèmes multi-robots mobiles. Thèse de doctorat, Laboratoire d'Automatique de Besançon (UMR CNRS 5696), 2005.

[21] M. Agheli. Analytical Workspace, Kinematics, and Foot Force Based Stability of Hexapod Walking Robots. Thèse de doctorat, Worcester Polytechnic Institute, 2013.

[22] M. Saska, Vonásek V., Kulich V., Fišser D. & Krajník T. Bringing Reality to Evolution of Modular Robots : Bio-Inspired Techniques for Building a Simulation Environment in the SYMBRION Project. In "Reconfigurable Modular Robotics :

Challenges of Mechatronic and Bio-Chemo-Hybrid Systems". Piscataway : IEEE, pp. 1–6, 2011.

[23] D. Camacho, F. Fernandez & M. Rodelgo. Roboskeleton : An architecture for coordinating robot soccer agents. Engineering Applications of Artificial Intelligence, vol. 19, no. 2, pp. 179– 188, 2006.

[24] D. Camacho, F. Fernandez & M. Rodelgo. Roboskeleton : An architecture for coordinating robot soccer agents. Engineering Applications of Artificial Intelligence, vol. 19, no. 2, pp. 179– 188, 2006.

[25] F. R. Noreils, .Toward a robot architecture integrating cooperation between mobile robots: Application to indoor environment,. Int. Journal of Robotics Research, vol. 12, no. 1, pp. 79.98, 1993.

[26] E. Durfee, V. Lesser, and D. Corkill, .Trends in cooperative distributed problem solving,. IEEE Transactions on Knowledge and Data Engineering, vol. KDE-1, no. 1, pp. 63.83, March 1989.

[27] Broten, G., Monckton, S., Giesbrecht, J., Verret, S., Collier, J., and Digney, B. (2004), Towards Distributed Intelligence, (DRDC Su±eld TR 2004-287), Defence R&D Canada { Su±eld, Medicine Hat, Alberta.

[28] P. Stone, Layered Learning in Multiagent Systems. MIT Press, 2000.

[29] The effect of heterogeneity in teams of 100+ mobile robots,. in Proocedings of the 2003 NRL Workshop on Multi-Robot Systems, A. C. Shultz and L. E. Parker, Eds., vol. II. Washington, DC: Kluwer Academic Publishers, 2003.

[30] K. Konolige et al., .Centibots: Large scale robot teams,. in Proocedings of the 2003 NRL Workshop on Multi-Robot Systems, A. C. Shultz and L. E. Parker, Eds., vol. II. Washington, DC: Kluwer Academic Publishers, 2003.

[31] L. Iocchi, D. Nardi, and M. Salerno, .Reactivity and deliberation: a survey on multi-robot systems, in Balancing Reactivity and Deliberation in Multi-Agent Systems (LNAI 2103), E. P. M. Hannebauer, J. Wendler, Ed. Springer, 2001, pp. 9.32.

[32] A. Arsenio and M. I. Ribeiro. Absolute localization of mobile robots using natural landmarks. In Proceedings of the International Conference on Electronics, Circuits and Systems, 1998.

[33] N. Ayache and O. Faugeras. Maintaning representations of the environment of a mobile robot. IEEE Transactions on Robotics and Automation, 5(6) :804 – 819, 1989.

[34] I. A. Bachelder and A. M. Waxman. Mobile robot visual mapping and localization: A viewbased neurocomputational architecture that emulates hippocampal place learning. Neural Networks, 7(6/7) :1083–1099, 1994.

[35] I. A. Bachelder and A. M. Waxman. A view-based neurocomputational system for relational map-making and navigation in visual environments. Robotics and Autonomous Systems, 16 :267–298, 1995.

[36] Anderson, B., Yu, C., Fidan, B. & Hendrickx, J. (2008). Rigid grap control architectures for autonomous formations. Control Systems Magazine, IEEE.

[37] Das, A.K., Fierro, R., Kumar, V., Ostrowski, J.P., Spletzer, J. & Taylor, C.J. (2002). A vision-based formation control framework. IEEE Transactions on Robotics and Automation, 18, 813–825.

[38] Gu, D. & Hu, H. (2006). Receding horizon tracking control of wheeled mobile robots. IEEE Transactions on Control Systems Technology, 14, 743–749.

[40] Balch, T. & Arkin, R. (1998). Behavior-based formation control for multirobot teams. Robotics and Automation, IEEE Transactions on, 14, 926 –939.

[41] Brooks, R.A. (1985). A robust layered control system for a mobile robot. Tech. rep., Cambridge, MA, USA.

[42] Brunete, A., Hernando, M., Gambao, E. & Torres, J.E. (2012).
A behaviour-based control architecture for heterogeneous modular, multiconfigurable, chained micro-robots. Robotics and Autonomous Systems, 60,
1607–1624.

43] Dougherty, R., Ochoa, V., Randles, Z. & Kitts, C. (2004). A behavioral control approach to formation-keeping through an obstacle field. In Aerospace Conference, 2004. Proceedings. 2004 IEEE, vol. 1, –175 Vol.1.

[44] Lawton, J., Beard, R. & Young, B. (2003). A decentralized approach to

formation maneuvers. Robotics and Automation, IEEE Transactions on, 19, 933 – 941.

[45] Ren, W. & Beard, R.W. (2005). Consensus seeking in multiagent systems under dynamically changing interaction topologies. Automatic Control, IEEE Transactions on, 50, 655–661.

[46] Ghommam, J., Mehrjerdi, H., Saad, M. & Mnif, F. (2010). Formation path following control of unicycle-type mobile robots. Robotics and Autonomous Systems, 58, 727 – 736.

[47] Chen, H. (2009). Towards Multi-robots Formation: Study on Vision-based Localization

System. Thesis, City University of Hong Kong.

[48] Das, A.K., Fierro, R., Kumar, V., Ostrowski, J.P., Spletzer, J. & Taylor, C.J. (2002). A vision-based formation control framework. IEEE Transactions on Robotics and Automation, 18, 813–825.

[49] Desai, J., Ostrowski, J. & Kumar, V. (1998). Controlling formations of multiple mobile robots. In Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on, vol. 4, 2864 –2869 vol.4.

[50] Desai, J.P., Ostrowski, J.P. & Kumar, V. (2001). Modeling and control of formations of nonholonomic mobile robots. IEEE Transactions on Robotics and Automation, 17, 905.

[51] Qian, D., Tong, S., & Li, C. (2016) leader-following formation control of multiple robots with uncertainties through sliding mode and nonlinear distrbance observer. ETRI Journal, 38(5), 1008-1018.

[52] Shen, D., Sun, Z., sun, W, (2014). Leader-follower formation control without leader's velocity information. Science china information Sciences,57(9), 1-12

[53] Dai, Y., & Lee, S. G. (2012). The leader-follower formation control nonholonomie mobile robots. International journal of control, Automation and systemes, 10(2), 350-361.

[54] Min, H, J, Drenner, A., Papaikolopoulos, N.(2009, may). vision-based leader-follower formation with limitedinformation. In robotics and Automation, 2009. ICRA4'09 IEEE international inference on (app. 351-356). IEEE.

[55] Vidal, R, Shakernia, O., & sastry, S. (2004). Following the flock [formation control]. IEEE robotics & Automation Magazin, 11(4), 14-20.

[56] International Journal on Artificial Intelligence Tools Vol. 16, No. 4 (2007) 565-582 APPLYING A TAXONOMY OF FORMATION CONTROL IN DEVELOPING A ROBOTIC SYSTEM————————————,HARRY CHIA-HUNG HSU,ALAN LIU

[57] Programming Robots with ROS A PRACTICAL INTRODUCTION TO THE ROBOT OPERATING SYSTEM, Morgan Quigley, Brian Gerkey & William D. Smart

[58] : Erard, P. J. (1999). Simulation par évènement discrets. Presses Polytechniques et Universitaires Romandes (PPUR).

[59] The Unified Modeling Language Reference Manual Seconde Edition, James Rumbaugh Ivar Jacobson, Grady booch.

[60] Mourani, G. (2000). Securing and optimizing Linux: RedHat edition. Open Network Architecture and OpenDocs Publishing.

[61] Python. Consulté le 4 septembre 2017 de https://i.ytimg.com/vi/Z7soD7Yj9uw/hqdefault.jpg

[62] Marguin, O. (2004). Cours d'informatique, C++ : LES BASES.

[63] Lien vers la page d'aide de PCL : http ://pointclouds.org/documentation/tutorials

[64]Mémoire de Master II en Informatique Appliquée aux Systèmes d'Information Géographique (IASIG)

ANNEXE

1. Types de Simulation

En robotique, on peut distinguer deux types de simulateurs, ceux restreints à la validation d'un composant particulier, et les autres qui comprennent la simulation de tout l'environnement pour tester les performances des robots. Il n'en n'existe pas beaucoup, nous allons nous intéresser à l'un d'entre cette deuxième catégorie, qui est eux le simulateur Gazebo.

1.1. Simulation Haptique Interactive :

Le Dispositif haptique est un robot qui, au lieu de manipuler le monde physique conformément à une commande utilisateur, applique des forces à la main ou au doigt de l'utilisateur en fonction des forces résultant du mouvement de l'utilisateur dans un environnement virtuel. Comme si le virtuel est soumis à des forces.

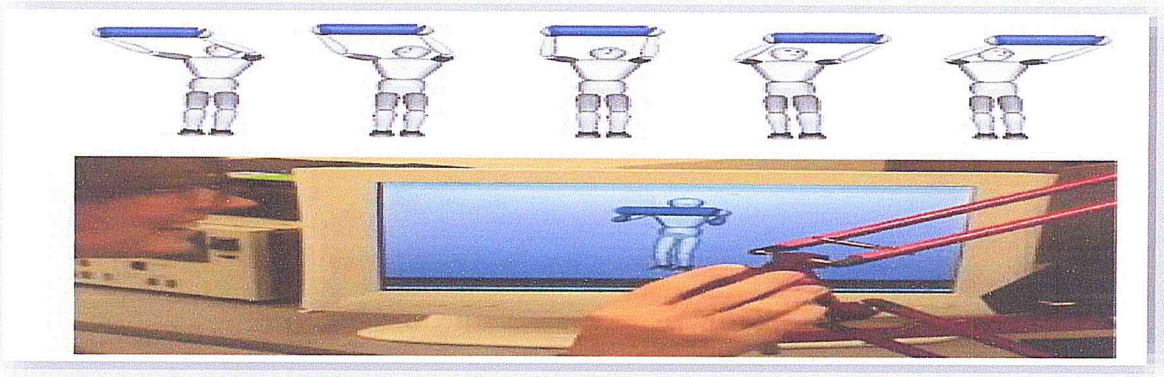


Figure [1] : Un utilisateur commande un robot virtuel pour déplacer l'objet manipulé en utilisant un dispositif haptique. Le dispositif permet à l'utilisateur d'interagir et de contrôler la tâche de manipulation. Alors que le robot humanoïde ajuste de façon autonome ses énergies de posture.

Un dispositif haptique peut également servir de dispositif d'entrée, comme représenté sur la figure [1], l'utilisateur spécifie un mouvement pour l'objet détenu par les robots humanoïdes. La séquence d'images de la figure [1] montre le résultat sur tout le mouvement de la figure si l'objet est commandé de se déplacer d'un côté à l'autre.

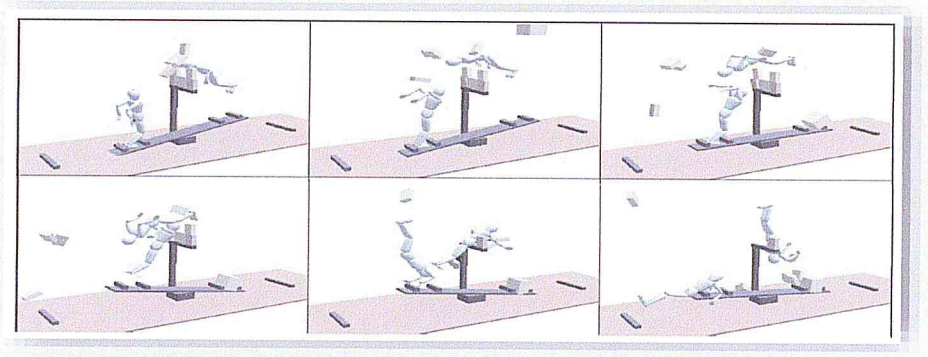
Alors que les objets sont manipulés de manière interactive, la figure ajuste de façon autonome sa posture en fonction d'énergies de posture simples.

Le dispositif haptique est utilisé pour générer les forces du ressort virtuel qui apparaissent à l'utilisateur comme les forces de contraintes provoquées par le contact avec un environnement réel. Ce cadre permet également à l'ombrage haptique de lisser les bords dans des données polygonales et d'afficher des propriétés de surface, telles que le frottement et la texture.

1.2. Simulation Dynamique :

La relation dynamique entre tous les points de contact existants peut être décrite. Cette relation est caractérisée par les masses perçues aux points de contact. Une force exercée au niveau d'un point de contact, qu'il s'agisse d'une collision avec un autre objet ou de l'interaction avec un utilisateur, peut être traduite en forces à tous les points de contact apparentés. Les calculs nécessaires peuvent être effectués avec un algorithme récursif efficace.

La représentation de l'espace de contact permet de décrire facilement l'interaction entre les groupes de systèmes dynamiques sans avoir à examiner les équations complexes du mouvement du système individuel de la Terre. En tant que tel, le modèle de collision peut être développé avec la même facilité que si on envisageait l'interaction seulement entre les corps simples. L'impact et les forces de contact entre les corps en interaction peuvent alors être efficacement résolus. Pour la simulation de la dynamique du corps articulé se déplaçant dans l'espace libre. La figure [2] illustre un environnement virtuel. Deux personnages humanoïdes sont en collision et interagissent avec de nombreux objets dans l'environnement.



2. Comparaison entre les simulateurs Open source

Le tableau suivant représente une comparaison entre les simulateurs Open source les plus utilisés

	Stage	**Gazebo**	**Simbad**
SE	Mac, Linux, Win	Linux	Win, Mac, Linux
Type de Simulateur	2D	3D	3D
Langage de Programmation	C, C++, Python, Java,	C, C++, Python, Java,	Java,
Portabilité	Oui	Oui	Limité
Capteurs	gamme, odomètre	Caméra, gamme, odomètre	Caméra, gamme, contact
Exigences	Faible	Norma	Faible
Environnement réaliste	Non	Oui	Non
Dynamique des objets	Non	Oui	Non

Comparaison entre les Simulateurs Open-Source.

3. L'interface grafique de Gazebo

L'éditeur est composé des 2 parties :

- **La palette :** à gauche, on trouve deux onglets. L'onglet Insertion nous permet d'insérer des pièces (liens et autres modèles) dans la scène pour créer un modèle. L'onglet Modèle affiche une liste de toutes les pièces qui composent le modèle que vous construisez.
- **La vue 3D** : sur la droite, elle nous permet d'avoir un aperçu de votre modèle et interagir avec lui pour modifier ses propriétés.

Les outils GUI de la barre d'outils supérieure peuvent être utilisés pour manipuler les articulations et les liens dans la vue 3D.

4. La Teléopération :

4.1. Présentation de la Teléopération :

Le terme Teléopération est composé de deux mots, « télé » de racine grecque signifie à distance, et le mot latin « opération » qui désigne l'exécution d'une certaine tâche. La Teléopération désigne donc un mode d'exécution d'une tâche, d'une action, d'un mouvement ou d'un travail à distance.

En robotique, la Teléopération, appelée aussi télérobotique, désigne les principes et les techniques qui permettent à un opérateur humain d'accomplir une tâche à distance, à l'aide d'un système robotique d'intervention, commandé à partir d'une station de contrôle, par l'intermédiaire d'un canal de transmission.

La Teléopération est un axe de recherche actuel, elle suscite beaucoup d'intérêts et trouve son application dans divers disciplines et activités de la vie moderne. Elle concerne la réalisation de travaux dépassant les capacités physiques de l'homme tels que la manipulation d'objets lourds, ou présentant un danger (cas de robots industriels) ,ou bien des travaux qui sont réalisés sur des sites inconnues ou hostiles à l'homme (tels que: les robots d'exploration martiens , robots industriels, les robots sous-marins, les robots permettant la détection et la réparation de fissures dans des conduits radioactifs, les robots chirurgiens, les robots destinés à l'exploration de l'univers, etc...). On peut

citer de grands domaines d'applications de la Teléopération, ex le domaine nucléaire, militaire, domaine spatial, le domaine médical.

4.2. Les fonctions de la Teléopération

On distingue trois fonctions principales en Teléopération, la perception la décision et l'action :

- **La perception** : assure le contrôle de la situation du site distant à partir d'informations issues de capteurs.
- **La décision** : permet de choisir les actions à réaliser en se basant sur l'information issue du module de perception.
- **L'action** : réalisée à l'aide d'un système de commande, elle permet d'agir sur l'environnement.

Ces atouts ont permis de téléopérer des robots esclaves qui se situent à distances considérables par rapport au site maitre.