République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche scientifique

ECOLE NATIONALE POLYTECHNIQUE

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département d'Electronique

Laboratoire des Dispositifs de

Communication et de Conversion Photovoltaïque

Thèse de Doctorat En Electronique

Option : **Electricité Solaire**

Présentée par :

KAMEL BOUDJIT

Magister en Electronique de l'USTHB

Pour l'obtention du titre de

Docteur en Sciences

Thème

Contribution à l'Etude des Commandes Multi-Moteurs en Temps Réel

Composition du Jury :

HADDADI MOURAD	**Professeur**	**ENP**	**Président**
LARBES CHERIF	**Professeur**	**ENP**	**Promoteur**
AIT- CHEIKH Mohamed Salah	**Professeur**	**ENP**	**Examinateur**
BARAZANE LYNDA	**Professeure**	**USTHB**	**Examinatrice**
HASNI MOURAD	**Professeur**	**USTHB**	**Examinateur**
NEMRA ABDELKRIM	**MCA**	**EMP**	**Examinateur**

ENP 2019

Laboratoire de Dispositifs de communication et de Conversion Photovoltaïque (LDCCP),
Ecole Nationale Polytechnique 10 Rue des Frères OUDEK, El-Harrach, BP.182, 16200 El Harrach, Alger,
Algérie www.enp.edu.dz

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche scientifique

ECOLE NATIONALE POLYTECHNIQUE

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département d'Electronique

Laboratoire des Dispositifs de

Communication et de Conversion Photovoltaïque

Thèse de Doctorat En Electronique

Option : **Electricité Solaire**

Présentée par :

KAMEL BOUDJIT

Magister en Electronique de l'USTHB

Pour l'obtention du titre de

Docteur en Sciences

Thème

Contribution à l'Etude des Commandes Multi-Moteurs en Temps Réel

Composition du Jury :

HADDADI MOURAD	**Professeur**	**ENP**	**Président**
LARBES CHERIF	**Professeur**	**ENP**	**Promoteur**
AIT- CHEIKH Mohamed Salah	**Professeur**	**ENP**	**Examinateur**
BARAZANE LYNDA	**Professeure**	**USTHB**	**Examinatrice**
HASNI MOURAD	**Professeur**	**USTHB**	**Examinateur**
NEMRA ABDELKRIM	**MCA**	**EMP**	**Examinateur**

Année Universitaire ENP 2019

Laboratoire de Dispositifs de communication et de Conversion Photovoltaïque (LDCCP),
Ecole Nationale Polytechnique 10 Rue des Frères OUDEK, El-Harrach, BP.182, 16200 El Harrach, Alger,
Algérie www.enp.edu.dz

ملخص

في مجال الروبوتات والعديد من التطبيقات الأخرى ، تعرضت رؤية الكمبيوتر والتعرف على الأشياء وتتبع الأهداف للكثير من الأبحاث النشطة مع العديد من الأساليب المختلفة والنتائج المثيرة للاهتمام. في مشروعنا ، نعتبر مشكلة التتبع الهدف التلقائي من أجل العثور على طريقة فعالة ومفيدة AR.Drone كمنصة جوية لتتبع الأهداف. من أجل ضمان التعرف الصحيح على الهدف بواسطة AR.Drone 2.0 للطائرات بدون طيار. تم استخدام على التحكم في الطائرة بدون طيار (ROS) كمؤشر هدف. يعمل نظام تشغيل روبوت ar_track_alvar ، تم استخدام رمز الاستجابة السريعة و.

تهدف هذه الرسالة إلى استخدام آلية التحكم المرئية للتحكم في الكوادبوتور ، الذي يسعى لتحقيق الهدف. إن الطبيعة غير الخطية لرباعي الدوران ، من ناحية ، وصعوبة الحصول على نموذج دقيق لها ، من ناحية أخرى ، تشكل تحديين جديين في تصميم وحدة تحكم لهذه الطائرة بدون طيار. الحل المحتمل لهذه المشاكل هو استخدام أساليب التحكم الذكي مثل تلك التي تعتمد على الشبكات العصبية الاصطناعية والضوابط الضبابية.

والتحكم الضبابي في هذه الأطروحة لتحديد الأهداف وتتبعها. تظهر النتائج (ANNs) يقترح أسلوب جديد يعتمد على الشبكات العصبية الاصطناعية التجريبية والمحاكاة لإثبات جدوى الطريقة المقترحة لتتبع الأهداف.

كلمات البحث: نظم تحديد الهوية والتحكم ، الذكاء الاصطناعي ، التعرف على الصور ، التعرف على وجوه ، الكوادبوت

Abstract

In robotics and many other applications, computer vision, object recognition and object tracking have been subjected to a lot of active research with many different approaches and interesting results. In our project, we consider the automatic target-tracking problem in order to find an efficient and useful method for UAV (Unmanned Aerial Vehicle). The AR.Drone 2.0 has been used as the air platform for target tracking. In order to guarantee a correct recognition of the target by the AR.Drone, a QR code and ar_track_alvar are have been used as target marker. A Robot Operating System (ROS) achieves the drone control.

This thesis aims to use a visual- based control mechanism to control a quadrotor, which is in pursuit of a target. The nonlinear nature of a quadrotor, on one hand, and the difficulty of obtaining an exact model for it, on the other hand, constitute two serious challenges in designing a controller for this UAV. A potential solution for such problems is the use of intelligent control methods such as those that rely on artificial neural networks and fuzzy logic controller.

A novel technique based on Artificial Neural Networks (ANNs) and fuzzy logic controller is proposed in this thesis for the identification and tracking of targets. Experimental results and simulations are shown to demonstrate the feasibility of the proposed method for target tracking

Keywords: Identification and control systems, Artificial intelligence, Image recognition, object recognition, Quadrotor.

Résumé

Dans le domaine de la robotique et de nombreuses autres applications, la vision par ordinateur, la reconnaissance d'objets et le suivi d'objets ont fait l'objet de nombreuses recherches actives avec de nombreuses approches différentes et des résultats intéressants. Dans notre projet, nous examinons le problème de suivi automatique des cibles afin de trouver une méthode efficace et utile pour les UAV (Unmanned Aerial Vehicle). L'AR.Drone 2.0 a été utilisé comme plate-forme aérienne pour le suivi des cibles. Afin de garantir une reconnaissance correcte de la cible par l'AR.Drone, un QR code et ar_track_alvar ont été utilisés comme marqueur de cible. Le contrôle du drone est réalisé par un système d'exploitation robot (ROS : Robot Operating System).

Cette thèse vise à utiliser un mécanisme de contrôle visuel pour contrôler un quadrotor à la recherche d'une cible. La nature non linéaire d'un quadrotor, d'une part, et la difficulté d'obtenir un modèle exact de celui-ci, d'autre part, constituent deux défis de taille dans la conception d'un contrôleur pour ce drone. Une solution potentielle à de tels problèmes consiste à utiliser des méthodes de contrôle intelligentes telles que celles reposant sur des réseaux de neurones artificiels et des contrôleurs flous.

Une nouvelle technique basée sur les réseaux de neurones artificiels (RNA) et le contrôleur flou est proposée dans cette thèse pour l'identification et le suivi de cible. Les résultats expérimentaux et les simulations démontrent la faisabilité de la méthode proposée pour le suivi de cible.

Mots-clés : Systèmes d'identification et de contrôle, Intelligence artificielle, Reconnaissance d'images, Reconnaissance d'objets, Quadrotor.

Dédicaces

A MA TRÈS CHÈRE MÈRE

Source inépuisable de tendresse, de patience et de sacrifice. Ta prière et ta bénédiction m'ont été d'un grand secours tout au long de ma vie.

Quoique je puisse dire et écrire, je ne pourrai exprimer ma grande affection et ma profonde reconnaissance.

J'espère ne jamais te décevoir, ni trahir ta confiance et tes sacrifices.

Puisse dieu tout puissant, te préserver et t'accorder santé, longue vie et bonheur.

A MON DÉFUNT PÈRE

De tous les pères, tu es le meilleur.

Tu as été et tu seras toujours un exemple pour moi par tes qualités humaines, ta persévérance.

Trouve dans ce travail le fruit de toutes tes peines et de tous tes efforts.

En ce jour, j'espère réaliser l'un de tes rêves.

Aucune dédicace ne saurait exprimer mes respects, ma reconnaissance et mon profond amour.

Que dieu t'accorde la paix éternelle et t'accueille dans son paradis.

A LA MÉMOIRE DE MON PETIT FRÈRE NABIL

Tu as été toujours dans mon esprit et dans mon cœur, je te dédie aujourd'hui ma réussite.

Que dieu, le miséricordieux, t'accorde son éternel paradis.

A MES CHERS FRÈRES ET SŒURS

Je ne peux exprimer à travers ces lignes tous mes sentiments d'amour et de tendresse envers vous mes chers frères et sœurs.

Puisse l'amour et la fraternité nous unissent à jamais. Je vous souhaite la réussite dans votre vie, avec tout le bonheur qu'il faut pour vous combler.

Merci pour votre précieuse aide à la réalisation de ce travail.

A vous mes chers frères et sœur ; je dédie cette thèse.

A MA FEMME ET MON FILS IMRAN

Je dédie ce travail à ma femme et à mon fils Imran qui, grâce à eux, j'ai eu la volonté et le courage de poursuivre mes études.

Remerciements

Il est impossible de synthétiser dans quelques lignes ma gratitude pour toutes les personnes qui, chacune de sa façon, m'ont aidé, soutenu ou simplement accompagné pendant ces longues années. Le seul fait de faire une liste implique d'indiquer une priorité et une préférence, quand en réalité on voudrait pouvoir donner à tout le monde la première place.

Je remercie spécialement mon directeur de thèse Pr. LARBES CHERIF qui a eu confiance en moi en me proposant ce doctorat et en me donnant pleine liberté et indépendance dans l'organisation de mon travail. Indépendance que ne lui a jamais empêché de me suivre de près, en me donnant toujours conseil, guidance et soutien quand j'en ai eu besoin.

Je ne manquerais pas non plus de dire un grand merci aux membres du jury qui ont accepté, sans réserve aucune, d'évaluer cette thèse à sa juste valeur, et de me faire part de leur rem arques sûrement pertinentes qui, avec un peu de recul, contribueront, sans nul doute, au perfectionnement du présent travail.

Merci à Mr. HADDADI MOURAD, Professeur à l'ENP Ecole Nationale Polytechnique pour m'avoir fait l'honneur de présider mon Jury de thèse.

Je tiens aussi à exprimer mes sincères remerciements à Mr. AIT- CHEIKH Mohamed Salah, Professeur à l'ENP qui a aimablement accepté de participer à cette commission d'examen.

Mes remerciements s'adressent également à Madame BARAZANE LYNDA Professeure à l'USTHB pour l'honneur qu'elle me fait en acceptant d'examiner mon travail et de faire partie du jury de soutenance.

Mes remerciements s'adressent également à Mr. HASNI MOURAD Professeur à l'USTHB pour avoir accepté d'être mon examinateur malgré les lourdes tâches attachées à sa fonction.

Mes remerciements vont aussi à Mr. NEMRA ABDELKRIM, MCA de l'EMP d'avoir accepté d'examiner et d'évaluer cette thèse.

Une reconnaissance et des remerciements particuliers à toute ma famille qui m'ont toujours soutenu et aidé au cours de ces années.

Sur un plan plus personnel, je voudrais témoigner toute ma reconnaissance à ma mère, pour son soutien, ses encouragements durant ces dernières années.

Je tiens plus largement à exprimer ma reconnaissance à toutes celles et à tous ceux qui ont contribué directement ou indirectement au bon déroulement de mes travaux.

Table des Matières

Table des matières

Chapitre II : Identification et Suivi de cible à Base d'un Contrôleur Flou

Chapitre IV : Identification et Suivi de Cible à Base d'un Contrôleur ANN

Liste des Tableaux

Table des Figures

Liste des Abréviations

ADC: Analog Digital Converter
ANN: Area Neural Network
FLC ; Fuzzy Logic Controller
GPS: Global Positioning System
IDL: Interface Définition Langage
IOS: International Organization for Standarization
IP: Internet Protocol
Lipo: Lithium Plymer
MCMM: Multi-charges Multi-missions
MLP: Multilayer perceptron
MVA: Mini Air Vehicle
NED: North-East-Down
OS: Operating System
PWM: Pulse Width Modulation
QVGA: Quarter Video Graphics Array
ROS: Robot Operating System
TCP: Transmission Control Protocol
UAV: Unmanned Aerial Véhicul
UDP : User Datagram Protocol
VGA : Video Graphics Ar

Introduction Générale

INTRODUCTION GENERALE

Le domaine des drones (Unmanned Aerial Vehicule - UAV) est en constante évolution depuis les débuts de l'aviation. À l'origine, les recherches dans ce domaine étaient principalement motivées par des applications militaires. En effet, les drones étaient, et demeurent aujourd'hui, la meilleure solution pour éviter la perte de pilotes lors de missions dangereuses. Cependant, compte tenu des complexités additionnelles inhérentes aux drones, le développement de ce domaine s'est effectué plus lentement que pour les systèmes avec pilote [1]. L'apparition de capteurs de plus en plus précis, l'augmentation constante de la puissance de calcul des processeurs ainsi que l'avancement des connaissances dans le domaine de l'aéronautique ont permis à cette tendance de s'inverser. En effet, le domaine des drones croît de manière exponentielle depuis le début des années 80. L'utilisation de drones est désormais monnaie courante dans plusieurs domaines d'applications telles que l'arpentage, la surveillance de pipeline et la photographie aérienne [2].

Ces dernières années, l'utilisation de véhicules aériens sans pilote (UAV) multi-rotors a gagné en popularité, en particulier le drone à quatre rotors communément appelé le quadrotor. En effet, le drone multirotor est capable d'effectuer un décollage et un atterrissage vertical, un vol stationnaire avec une agilité exceptionnelle [3]. Ces capacités permettent à l'UAV multi-rotor d'exécuter diverses tâches telles que le sauvetage, la surveillance, l'inspection et les opérations militaires.

De nos jours, l'AR.Drone est l'un des quadrotors les plus utilisés en tant que plate-forme de recherche dans la plupart des grandes universités du monde. L'AR.Drone a été choisi comme plate-forme dans cette thèse en raison de son prix relativement bon marché et de son électronique embarquée. L'électronique embarquée est déjà équipée d'une carte mère, d'un processeur ARM Cortex A9, d'une puce Wi-Fi, d'un capteur accéléromètre, d'un capteur gyroscope, d'un capteur à ultrasons, et de deux caméras. Cette plate-forme est également équipée d'un système d'exploitation en temps réel qui lui permet d'effectuer différentes tâches simultanément, telles que la communication via un réseau Wi-Fi, l'échantillonnage de données vidéo, le traitement d'images, l'estimation de l'état et le contrôle en boucle fermée.

Au cours des dernières décennies, de nombreux contrôleurs classiques ont été développés et testés pour les UAVs. Cependant, des incertitudes et des problèmes d'inexactitude sont souvent rencontrés dans les techniques de contrôle [4]. Ces problèmes peuvent être résolus en utilisant l'intelligence artificielle. Les contrôleurs à base de logique floue (FLC) et les réseaux neuronaux est l'une des méthodes d'intelligence artificielle les plus actives.

Pour contrôler le quadrotor, les chercheurs ont utilisé des contrôleurs classiques pour contrôler l'altitude, le tangage, le roulis et le lacet. Les chercheurs ont utilisé aussi les informations de vision pour la navigation et le contrôle. De nos jours, de nombreuses applications sur les quadrotors sont développées pour effectuer des tâches de manière autonome, sans aucune interaction ou commande de la part d'un humain. Par conséquent, le développement d'un système permettant à un quadrotor de faire de la surveillance, telle que la détection et le suivi d'un objet en mouvement, nous mènera à des tâches plus avancées exécutées par des drones à l'avenir [5].

Actuellement, les universités et les instituts de recherche utilisent des UAV comme objets de recherche dans les domaines de l'informatique et de la robotique. Ces recherches sont principalement axées sur la surveillance, l'inspection et la recherche et sauvetage. Les UAV équipés de systèmes de surveillance, tels que caméra et GPS, se sont bien comportés en matière de sécurité. La caméra est l'un des composants les plus importants de la surveillance à l'aide d'un drone [6]. Par conséquent, le traitement des images est nécessaire pour obtenir des informations de la caméra. Il existe déjà de nombreux algorithmes développés à cet effet.

Le suivi d'objets, tout comme la détection d'objets, est l'un des domaines importants de la vision par ordinateur. Le suivi d'objet peut être défini comme un processus permettant de suivre un objet dans une séquence d'images. Le niveau de difficulté du suivi d'objet dépend du mouvement de l'objet, du changement de motif de l'objet et de l'arrière-plan, de la modification de la structure de l'objet, de l'occlusion d'objet par objet ou d'objet par arrière-plan et du mouvement de la caméra. Le suivi d'objet est généralement utilisé dans un contexte d'application de haut niveau qui nécessite l'emplacement et la forme d'un objet à partir de chaque image.

Dans cette thèse, des contrôleurs intelligents ont été conçus pour le suivi de cible et évitement d'obstacles ainsi le suivi de trajectoire basé sur des contrôleurs intelligents à base de logique floue et réseau neuronal artificiel. Les informations acquises par les caméras forment les entrées pour les contrôleurs intelligents. Les informations obtenues du système de vision peuvent être facilement adaptées aux règles des contrôleurs. Les algorithmes proposés sont basés sur un petit nombre de paramètres d'entrée, ce qui entraîne de faibles exigences en puissance de calcul. Les études présentées avaient pour objectif principal des tests de simulation et des tests réels sur plateforme AR.Drone.

Afin d'atteindre nos objectifs, nous avons structuré notre thèse en cinq chapitres.

Dans le premier chapitre, nous aborderons un historique des Drones et particulièrement les quadrotors, ainsi que leurs avantages et inconvénients et quelques domaines d'application. Ensuite le principe du vol du quadrotor sera traité, où le lacet, le tangage et le roulis seront définit. Dans la dernière section de ce chapitre, la modélisation du quadrotor est fournie. Le modèle mathématique du quadrotor est très important car il décrit la manière dont ce dernier se déplace en fonction de ses entrées. Grâce à ces équations, il est possible de définir et de prédire les positions atteintes par le drone en recherchant uniquement les quatre vitesses du moteur.

Le chapitre deux traite la partie matérielle et programmation (Hardware et software), les différentes parties du Quadrotor AR. Drone seront détaillées, ainsi que l'organisation de ROS (Robot Operating System) et son principe de fonctionnement, par la suite nous aborderons le logiciel Fuzylite utilisé pour la conception des contrôleurs flous, en dernier nous nous étalerons sur la partie Matlab / AR.Drone et les différents blocs utilisés pour le suivi de trajectoire.

Dans le chapitre trois de cette thèse, une application utilisant un contrôleur flou sera présentée. Cette application utilise les informations visuelles de la caméra embarquée située à l'intérieur de l'AR.Drone. Une démonstration des contrôleurs basés sur la simulation a été fournie à l'aide de la combinaison des outils open source FuzzyLite, 'ar_track_alvar' et du ROS Gazebo avec Tum_ARDrone Simulator. Le contrôleur flou agit sur le lacet de l'UAV afin de maintenir l'appareil à une distance de sécurité en face de l'objet. Par la suite ce même contrôleur sera

élargi pour une application d'évitement d'obstacles. Des tests de simulation et des tests réels ont été réalisés pour valider le contrôleur flou proposé.

Dans le chapitre quatre, nous présentons une méthode originale pour un suivi visuel robuste, elle est basée sur un contrôleur à base de réseau neuronal. Cette application utilise les informations visuelles de la caméra de face du quadrotor AR.Drone 2.0. L'algorithme proposé est basé sur la détection de la cible (étiquette) et pourrait également être utilisé pour détecter des objets de formes différentes en temps réel. L'architecture proposée dans ce chapitre a également été testée avec les réseaux neuronaux Multilayer Perceptron (MLP). Les simulations et les résultats expérimentaux seront présentés et interprétés.

Dans le dernier chapitre, un contrôleur à base de logique floue (FLC) est implémenté dans le quadrotor AR.Drone afin de le faire suivre une référence de trajectoire donnée. La distance entre la position et l'angle de l'AR.Drone par rapport au point de référence est utilisée comme entrée du FLC. En ce qui concerne la sortie, la valeur de tangage et le taux de lacet constitueront le signal de contrôle de l'AR.Drone. Les données de navigation de l'AR.Drone sont la vitesse avant (V_x), la vitesse latérale (V_y) et le lacet. L'algorithme FLC sera implémenté dans AR.Drone à l'aide du logiciel Matlab / Simulink. En outre, l'algorithme a été testé dans différentes trajectoires, telles une forme carrée, circulaire et spirale. Les résultats de simulation et tests pratiques ont montré que l'AR.Drone peut très bien suivre une trajectoire donnée avec diverses positions et orientations initiales.

Enfin, on termine par une conclusion générale regroupant l'essentiel des résultats de nos travaux et les éventuelles perspectives.

Chapitre I :

Généralités Sur les Quadrotors et Modélisation

I.1. INTRODUCTION

Le quadrotor est classé dans la catégorie des systèmes volants les plus complexes vu le nombre d'effets physiques qui affectent sa dynamique à savoir les effets aérodynamiques, la gravité, les effets gyroscopiques, les frottements et le moment d'inertie [7]. Cette complexité résulte essentiellement du fait que l'expression de ces effets différent pour chaque mode de vol. En effet les modèles dynamiques du quadrotor proposés dans la littérature changent en fonction des taches planifiées et en fonction des milieux de navigation définis a priori par l'opérateur.
Dans ce chapitre nous commencerons par un historique des Drones et particulièrement les quadrotors, ainsi que leurs avantages et inconvénients et quelques domaines d'application. Ensuite le principe du vol du quadrotor est abordé, où le lacet, le tangage et le roulis sont définit. Dans la dernière section de ce chapitre, la modélisation du quadrotor est fournie. Ce résultat est très important car il décrit la manière dont l'hélicoptère se déplace en fonction de ses entrées. Grâce à ces équations, il est possible de définir et de prédire les positions atteintes par l'hélicoptère en recherchant uniquement les quatre vitesses du moteur.

I.2. LES DRONES

I.2.1. Définition

Il est intéressant de lire ce que retiennent les dictionnaires pour définir un drone. Le Petit Robert nous apprend que le mot drone est apparu dans la langue française en 1954, il vient de l'anglais signifiant « Faux bourdon ». L'autre définition du Petit Robert indique que c'est un petit avion de reconnaissance, sans pilote, télécommandé ou programmé. L'encyclopédie Universalis propose une définition un peu plus générale. Elle indique qu'un drone (en anglais U.A.V. pour Unmanned Aerial Vehicle) est un véhicule aérien sans pilote. Il peut donc faire appel au concept de l'avion, de l'hélicoptère ou de la fusée par exemple. Il sert, d'une façon générale, pour des missions de surveillance du champ de bataille, d'acquisition du renseignement ou de combat. Si le drone est d'abord apparu pour répondre à des besoins militaires, il est aussi désormais envisagé pour des applications civiles comme dans le domaine de la surveillance de zones et d'installations, dans l'agriculture, etc...

Toujours selon les dictionnaires, un drone est un aéronef disposant d'une ou plusieurs charges utiles nécessaires à l'observation ou destinées au combat (missiles, bombes). Généralement, il est contrôlé et piloté à partir de stations au sol, avec ou sans le relais des satellites, mais il peut aussi effectuer des missions de manière autonome. Dans l'avenir, il est prévu d'assister les drones par des systèmes de contrôle disposés sur des aéronefs (poste de commandement volant ou avion de combat). Chargés dans leur majorité de l'acquisition de renseignements, les drones disposent de différents capteurs opérant dans différentes longueurs d'ondes (domaines du visible, de l'infrarouge ou des ondes radars).

Ils peuvent aussi être équipés de moyens d'écoute électronique et de brouillage.

Aujourd'hui, lorsque l'on parle de drone et de leur mise en œuvre il est plus juste de parler de système de drone. En effet, le drone fait partie d'un système qui est composé d'un ou plusieurs vecteurs aériens, d'une ou plusieurs stations sol de commande ainsi que de liaisons de données entre le vecteur et la partie sol. Il peut y avoir des drones terrestres, marins, sous-marins et aériens. On voit donc que la définition s'étend progressivement et englobe les nombreux systèmes autonomes. Si on se restreint aux drones aériens, on peut les classer en différentes catégories en fonction de leurs tailles aujourd'hui très variées : de quelques centimètres à plusieurs mètres. Leurs formes également, tout comme leurs types de propulsion, certains sont équipés de réacteurs, d'autres d'hélices, d'autres encore utilisent des rotors, à l'instar des drones hélicoptères par exemple.

I.2.2. Classification

La classification des drones est un exercice très difficile, dans la mesure où elle est différente selon les pays. Cependant les drones aériens peuvent être classés selon trois critères que sont l'altitude de croisière, l'endurance en termes de temps de vol et leur dimension principale.

Dans ce cadre, le domaine opérationnel des drones peut se décomposer en trois segments :

- Les drones tactiques ;
- Les drones de moyenne altitude et longue endurance (MALE) permettant d'utiliser une charge utile de l'ordre de 100 kg ;
- Les drones de haute altitude et longue endurance (HALE).

Le segment tactique se décompose lui-même en six segments :

- Les micro-drones (Micro Air Vehicule ou MAV), pouvant être contenu dans une sphère de 30 cm ;
- Les mini-drones (Mini Air Vehicule ou MAV également), pouvant être contenu dans une sphère de 70 cm ;
- Les drones de très courte portée (TCP) ;
- Les drones moyenne portée lents (multi-charges multi-missions ou MCMM lents) ;
- Les drones rapides basse altitude (MCMM rapides) ;
- Les drones maritimes tactiques (DMT).

Cela peut surprendre de distinguer en deux segments les micro-drones et les mini-drones, mais la différence d'échelle entre les deux impose aujourd'hui encore des contraintes fortes pour le choix des matériaux, des capteurs et des systèmes embarqués. Par conséquent ces deux familles sont fortement différenciées par l'autonomie en vol et la qualité des contrôles, cependant la miniaturisation des cartes électroniques jointe à l'augmentation des capacités de calculs des mini-systèmes embarqués tend à réduire ces écarts.

I.2.3 Les rôles des drones

I.2.3.1. Les drones militaires

De nos jours les drones jouent un rôle important dans les applications militaires. Ils ne sont plus utilisés aujourd'hui comme seule plate-forme de reconnaissance, mais de plus en plus aussi

comme systèmes d'armes autonomes. Les avantages militaires des drones de combat se manifestent principalement dans la lutte contre les insurrections et le terrorisme. Les restrictions centrales dans l'utilisation des drones se situent toutefois dans les domaines technologiques et doctrinaux. Une supplantation des avions avec pilote dans la conduite de la guerre ne s'envisage pas.

Les drones militaires peuvent se subdiviser en trois catégories :

I.2.3.1.1. Les drones stratégiques

Sont utilisés pour la reconnaissance à large échelle sur les régions ennemies. En font partie des systèmes comme le Global Hawk, qui peut opérer à une altitude maximale de 20000 mètres jusqu'à 40 heures et qui a une portée de 3000 miles nautiques. Font partie des drones opérationnels les systèmes Predator et Reaper, qui volent à une altitude maximale respectivement de 7500 mètres et 15000 mètres. Ils sont utilisés dans les zones d'intervention militaires et peuvent servir tant à la reconnaissance que pour des attaques.

I.2.3.1.2. Les drones tactiques

Ces drones se déplacent à des altitudes de vol plus basses et sur des trajets plus courts. Leur principale fonction consiste à permettre aux commandants sur place la surveillance des activités ennemies, sans mettre en danger leurs propres soldats. Un exemple de ce cas est le système Dragon Eye. Contrairement aux drones stratégiques et opérationnels, qui peuvent être soit pilotés à distance, soit programmés au préalable pour le vol autonome, les drones tactiques sont toujours pilotés par des opérateurs. Ils sont aussi utilisés fréquemment par les forces de police pour le contrôle de foules humaines et la surveillance des frontières.

I.2.3.1.3. Les drones opérationnels

Tous les types de drones sont utilisés exclusivement pour l'acquisition d'informations. La plupart (95%) des engagements ont servi à des buts de reconnaissance et de surveillance. Ce chiffre atteste d'un fait essentiel : les drones ont été jusqu'à présents plutôt une plate-forme de reconnaissance que de combat. Considéré au plan historique, leur rôle dans la conduite du combat pour les forces combattantes est celui d'un multiplicateur, qui augmente l'efficacité des unités de combat par des reconnaissances dans la zone d'engagement. Dans le développement de la technologie des drones, qui a été forcé surtout depuis la guerre du Vietnam par les USA et par Israël, l'utilisation d'avions sans pilote en tant que systèmes d'armes autonomes n'a joué jusqu'il y a peu qu'un rôle secondaire comme :

- Observation et surveillance ;
- Ecoute des signaux électromagnétiques ;
- Détection de missiles balistiques grâce à une alerte avancée ;
- Relais de communication ;
- Illumination de cibles ;
- Brouillage.

I.2.3.2. Les drones civils

Après les drones militaires, une autre révolution est en marche : celle des drones civils. Les utilisations de ces avions sans pilotes commandés à distance sont nombreuses, jusqu'au projet de livraison à domicile de colis achetés sur internet.
Nous présentons ici les applications par catégories. Certaines d'entre elles ne sont pas compatibles avec la régulation actuelle de la circulation aérienne.

Pour notre classification nous nous sommes basés sur la masse et l'altitude

Tous les avantages reconnus des drones pour les applications militaires sont transposables aux applications civiles, les environnements « Terne, Sale et Dangereux » peuvent se rencontrer dans le domaine civil. Des missions civiles très similaires aux missions de défense comme la surveillance des frontières et la surveillance des personnes (lors de manifestations publiques par exemple ou lors de simples missions de police) sont facilement envisageables. D'autres applications, plus originales, sont étudiées à plus ou moins long terme : observation de la terre, télécommunications, transport, ...

Nous identifions trois grandes catégories de drones candidats à des applications civiles :

Mini et micro-drones à basse altitude

- Photos aériennes, inspection.
- Publicité.
- Epandage agricole, surveillance des cultures, garde de troupeaux.
- Expériences scientifiques, mesures atmosphériques.
- Déminage.

L'évolution de cette branche sera fortement liée à l'évolution de la technologie correspondante. Envisagée uniquement dans le cadre des règles actuelles de l'aéromodélisme, l'utilisation des mini-drones est cependant assez limitée, notamment à cause de l'obligation de pilotage 'à vue '.

MALE

- Activités gouvernementales (police, douanes, environnement, ...).
- Missions scientifiques.
- Surveillance d'infrastructures (réseau routier, lignes électriques, pipe-lines, ...).
- Géodésie.
- Transport de fret.
- Transport de passagers.

Les drones de cette catégorie sont sur le marché (militaire) mais ne peuvent pas être intégrés au trafic civil.

HALE géostationnaire

- Radiodiffusion (télévision, ...)
- Télécommunications mobiles
- Environnement (incendies, pollution maritime, ...)

I.3. HISTORIQUE

En 1907 un hélicoptère à quatre rotors a été conçu par Louis Breguet. Ce fut le premier aéronef à voilure tournante capable d'un vol stationnaire d'une altitude de quelques pieds [8].

Tôt dans l'histoire du vol, les configurations de quadri-rotor ont été considérées comme la solution pour résoudre les problèmes persistants de vol vertical. En 1920, Etienne Oehmichen, dessine six modèles expérimentaux de giravions, parmi eux il essaya le deuxième modèle (Figure I.1), celui avec un châssis en tube d'acier et des rotors bipales aux extrémités des quatre bras, la machine a pu se stabiliser latéralement. Ce giravion présentait un degré considérable de stabilité et de contrôlabilité dans son temps, et fait plus d'un millier de vols d'essai pendant le milieu des années 1920 [9].

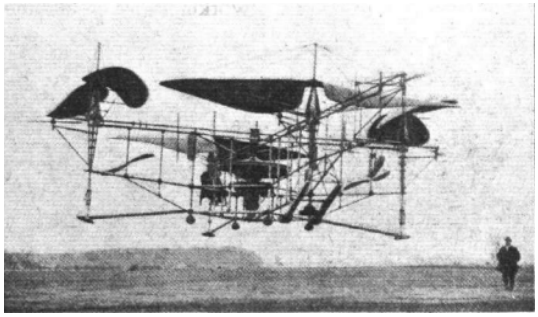


Figure I.1. Le modèle N°2

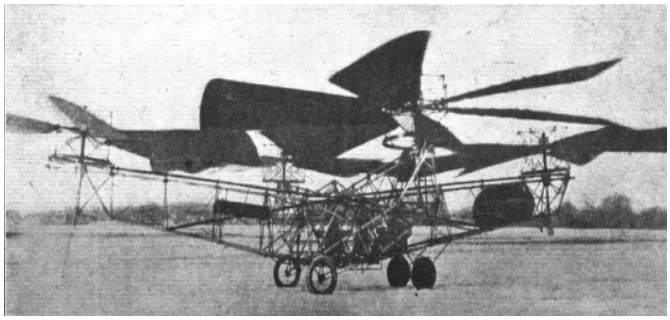


Figure I.2. Le quadri-rotor de **de Bothezat,** 1922.

En Octobre 1922 le quadri-rotor expérimental construit au début des années 1920 pour l'Armée de l'air des États-Unis (Figure I.2), par le Dr George de Bothezat et Ivan Jérôme, a effectué son premier vol. Environ 100 vols ont été effectués à la fin de 1923. Le plus haut jamais atteint était d'environ de 5 m. L'armée a annulé le programme en 1924, et l'avion a été abandonné, car il était de faible puissance, mécaniquement complexe, difficile à contrôler, et capable seulement de vol d'avancement dans un vent favorable [10].

En 1955, Comme résultat de poursuite des travaux d'Œhmichen en 1920 et de Bothezat en 1922, Convertawings.Inc a réalisé un prototype d'un quadri-rotor (Figure I.3) qui a depuis volé avec succès [11].

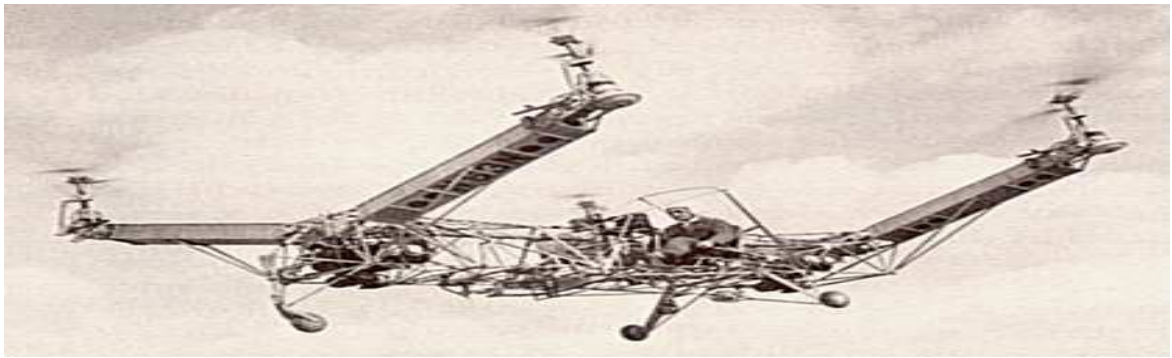


Figure I.3. Le prototype de Convertawings.Inc

Le drone choisi pour cette thèse est de type quadrotor.

Un quadrotor est un hélicoptère, qui est soulevé et manœuvré par quatre rotors. Il peut être manœuvré dans l'espace en trois dimensions uniquement en ajustant les vitesses des moteurs individuellement [12].

I.3.1. Avantages et inconvénients des quadrotors

Les quadrotors ont la capacité de faire un vol stationnaire qui permet à l'utilisateur une grande souplesse de manœuvre, cette capacité permet également au véhicule de décoller et d'atterrir verticalement, libérant l'utilisateur de contraintes opérationnelles généralement rencontrés avec les aéronefs à voilure fixe en raison de la dépendance sur les aérodromes. Bien que les quadrotors aient une vitesse limitée par rapport aux aéronefs, ils ont la capacité de prendre une route directe vers une zone déterminée, un avantage sur les véhicules terrestres qui peuvent être contraints d'éviter les obstacles tels que la croissance des plantes.

Bien que les quadrotors présentent de nombreux avantages, ils ont aussi quelques inconvénients. L'utilisation de la puissance de la batterie de manière continue limite la durée de vol des missions, ils sont également sensibles aux perturbations. La taille du quad-rotors est limitée par rapport au poids des moteurs et au nombre de capteurs transportés.

I.3.2. Utilisation des quadrotors

Les quadrirotors sont fréquemment utilisés pour réaliser des prises de vues aériennes, que ce soit dans le domaine civil (tournage d'un film, cartographie d'un terrain, etc.) ou militaire (envoie dans une zone dangereuse, etc.). Voici quelques applications déjà réalisées :

- Photos aériennes, inspection.
- Publicité.
- Epandage agricole, surveillance des cultures, garde de troupeaux.
- Expériences scientifiques, mesures atmosphériques.
- Déminage.
- Activités gouvernementales (police, douanes, environnement, ...).
- Missions scientifiques.
- Surveillance d'infrastructures (réseau routier, lignes électriques, pipe-lines, ...).
- Géodésie.
- Radiodiffusion (télévision, ...)
- Télécommunications mobiles
- Environnement (incendies, pollution maritime, ...)

I.3.3. Description générale du quadrotor

Un quadrotor est un robot mobile aérien à quatre rotors définit dans l'espace par 6 DDL [13]. Ces 4 rotors sont généralement placés aux extrémités d'une croix, et l'électronique de contrôle est habituellement placée au centre de la croix. Afin d'éviter à l'appareil de tourner sur lui-même sur son axe de lacet, il est nécessaire que deux hélices tournent dans un sens, et les deux autres dans l'autre sens. Pour pouvoir diriger l'appareil, il est nécessaire que chaque couple d'hélice tournant dans le même sens soit placé aux extrémités opposées d'une branche de la croix. Le fonctionnement d'un quadrotor est assez particulier. En faisant varier astucieusement la puissance des moteurs, il est possible de le faire monter/descendre, de l'incliner à gauche/droite (roulis) ou en avant/arrière (tangage) ou encore de le faire pivoter sur lui-même (lacet) [14]. Le quadrotor a six degrés de libertés, trois mouvements de rotation et trois mouvements de translation, ces six degrés doivent être commandés à l'aide de quatre déclencheurs seulement ; donc c'est un système sous actionné (le nombre des entrées inférieure au nombre des sorties).

I.3.4. Les mouvements du quadrotor

Dans les hélicoptères classiques, quand le rotor principal tourne, il produit un couple réactif qui inciterait le corps de l'hélicoptère à tourner dans la direction opposée si ce couple n'est pas contrarié. Ceci est habituellement fait en ajoutant un rotor de queue qui produit une poussée dans une direction latérale. Cependant, ce rotor avec son alimentation électrique associée ne fait aucune contribution à la poussée. Par contre, en cas de quadrotor, le rotor droit et le rotor gauche tournent dans le sens des aiguilles d'une montre et dans la direction opposée les rotors avant et arrière, ceci neutralise effectivement le couple réactif non désiré et permet au véhicule de planer sans tourner hors de la commande. D'ailleurs, différemment aux hélicoptères classiques, toute l'énergie dépensée pour contrecarrer le mouvement de rotation contribue à la force de poussée [15].
Les mouvements de base de quadrotor sont réalisés en variant la vitesse de chaque rotor changeant de ce fait la poussée produite. Le quadrotor incline vers la direction du rotor plus lent, qui tient compte alors de la translation le long de cet axe. Par conséquent, comme à un hélicoptère classique, les mouvements sont couplés, signifiant que le quadrotor ne peut pas réaliser la translation sans roulement ou tangage, ce qui signifie qu'un changement de la vitesse d'un rotor se traduit dans un mouvement en au moins trois degrés de liberté. Par exemple, augmentant la vitesse de propulseur gauche aura comme conséquence un mouvement de roulis (le quadrotor incline vers le rotor plus lent, vers la droite), un mouvement de lacet (l'équilibre entres les rotors qui tourne dans le sens des aiguilles d'une montre et les rotors qui tourne dans le sens inverse est perturbé ayant pour résultat un mouvement de rotation horizontal), et une translation (le mouvement de roulis incline l'armature et avec lui, l'orientation de la force de poussée). Cet accouplement est la raison pour laquelle nous pouvons commander les six degrés de liberté de quadrotor avec seulement quatre commandes (le couple appliqué par les moteurs sur chaque propulseur).
Le quadrotor a cinq mouvements principaux :

- Mouvement vertical

- Mouvement de roulis
- Mouvement de tangage
- Mouvement de lacet
- Translations horizontales

I.4. MODELISATION

La modélisation d'un système dynamique commandable est l'étape initiale avant de passer à la conception d'un contrôleur. Dans le cas des drones quadrotors, la modélisation a déjà été étudiée par les pionniers dans ce domaine d'application [16]. Les auteurs abordent initialement la modélisation dynamique du système en utilisant les équations de Newton-Euler. D'autres auteurs présentent l'analyse de la modélisation en utilisant les équations d'Euler-Lagrange [17]. Ce chapitre décrit de façon détaillée l'analyse dynamique d'un drone quadrotor pour l'obtention d'un modèle non linéaire. Dans une première partie, les repères de référence utilisés sont décrits. Ensuite, l'analyse cinématique est abordée. Puis, l'analyse dynamique du quadrotor à partir des équations de Newton-Euler est présentée et la représentation d'état du modèle dynamique est obtenue en appliquant une linéarisation jacobéenne. Le formalisme d'Euler- Lagrange sera utilisé pour obtenir la modélisation du système complet, en incluant la dynamique de la charge pendulaire.

Le quadrotor, est un avion composé de quatre moteurs, équipé d'une carte électronique en milieu et des moteurs aux quatre extrémités. Avant de décrire le modèle mathématique d'un quadrotor, il est nécessaire d'introduire les coordonnées de référence dans lequel nous décrivons la structure et la position. Pour le quadrotor, c'est possible d'utiliser deux systèmes de référence. Le premier est fixe et le second est mobile. Le système de coordonnées fixe, appelé aussi inertiel, est un système dans lequel la première loi de Newton est considérée comme valide. En tant que système de coordonnées fixe, nous utilisons le Systèmes O_{NED}, où NED est pour North-East-Down. Comme on peut le constater depuis la figure I.4, ses vecteurs sont dirigés vers le nord, l'est et le centre de la terre.

Figure I.4. Système de référence fixe O_{NED}

Le système de référence mobile que nous avons mentionné précédemment est associé au barycentre du quadrotor. Dans la littérature scientifique, il s'appelle système O_{ABC}, où ABC correspond à Aircraft Body Center. La figure I.5 illustre la coordination entre les deux systèmes de coordonnées.

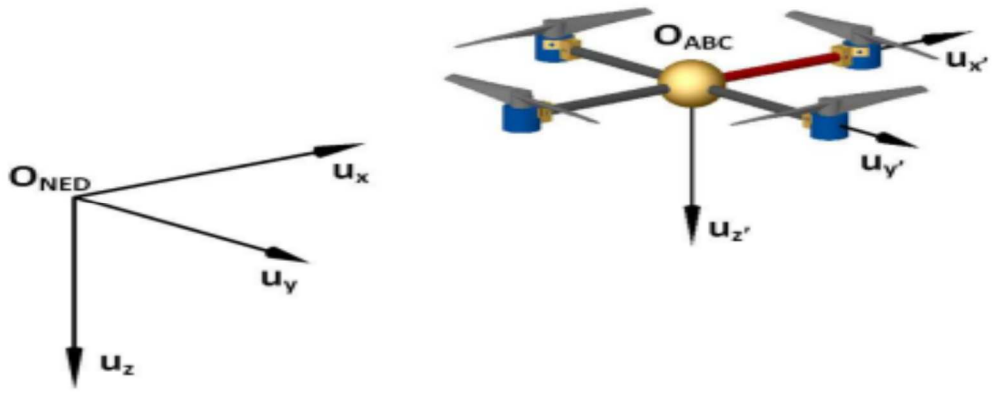


Figure I.5. Système de référence mobile et système de référence fixe.

L'attitude et la position du quadrotor peuvent être contrôlées en changer les vitesses des quatre moteurs. Les forces et moments suivants peuvent être effectués sur le quadrotor : la poussée provoquée par la rotation des rotors, le moment de Tangage et le moment de Roulis causé par la différence de poussée de quatre rotors, de la gravité, de l'effet gyroscopique et de moment de lacet. L'effet gyroscopique apparaît seulement dans le quadrotor dans la construction légère. Le moment de Lacet est causé par le déséquilibre des vitesses de rotation des quatre rotors. Le moment de Lacet peut être annulé lorsque deux rotors tournent dans le sens opposé. Alors que les hélices sont divisées en deux groupes. Dans chaque groupe il y a deux moteurs diamétralement opposés que nous pouvons facilement observer grâce à leur sens de rotation. À savoir, on distingue :

Hélices avant et arrière (numéros 2 et 4 dans la figure I.6), tournant dans le sens antihoraire ;

Hélices droite et gauche (numéros 1 et 3 dans la figure I.6), tournant dans le sens des aiguilles d'une montre.

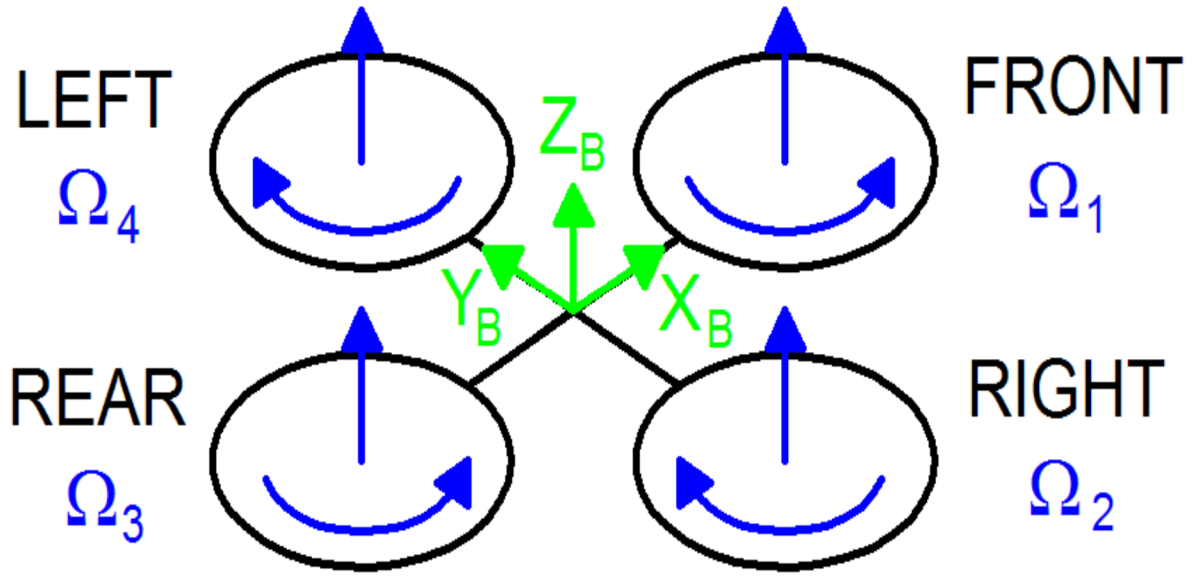


Figure I.6. Sens de rotation des hélices.

Le mouvement spatial de l'avion à cadre rigide peut être divisé en deux parties : le mouvement barycentre et mouvement autour du barycentre. Six degrés de liberté sont nécessaires pour décrire tout mouvement d'espace-temps. Ils sont trois mouvements barycentre et trois mouvements angulaires, à savoir trois translation et trois mouvements de rotations le long de trois axes.

Le contrôle des mouvements à six degrés de liberté peut être mis en œuvre en ajustant les vitesses de rotation de différents moteurs. Les mouvements comprennent les mouvements avant et arrière, les mouvements latéraux, les mouvements verticaux, mouvement de roulis et mouvement de tangage et de lacet. Le mouvement de lacet du quadrotor peut être réalisé par un couple réactif produit par le rotor. La taille du couple réactif est relative à la vitesse du rotor. Lorsque les quatre vitesses du rotor sont les même, les couples réactifs s'équilibrent et le quadrotor ne tourne pas, alors que si les quatre vitesses du rotor ne sont pas identiques, les couples réactifs ne seront pas équilibré, et le quadrotor va commencer à tourner.

Lorsque les quatre vitesses du rotor augmentent et diminuent de manière synchrone, le mouvement vertical est également requis. En raison de quatre entrées et de six sorties dans un quadrotor, ce dernier est considéré comme un système complexe non linéaire sous-actionné. Afin de le contrôler, certaines hypothèses sont formulées dans le processus de modélisation à savoir : le quadrotor est un corps rigide ; la structure est symétrique ; et l'effet de sol est ignoré. En fonction de la vitesse de rotation de chaque hélice, il est possible d'identifier les quatre mouvements de base du quadrotor, illustrés dans les figures I.7 à I.10.

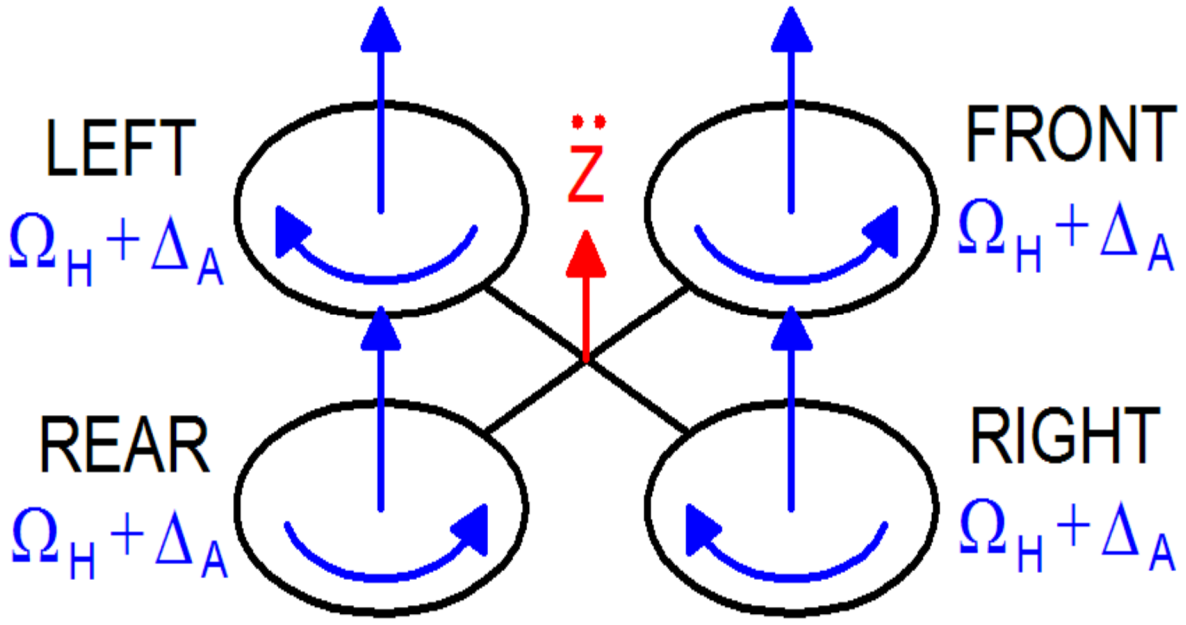


Figure I.7. Poussée ou mouvement du Gaz (Throttle).

Cette commande est obtenue en augmentant (ou en diminuant) toutes les vitesses de l'hélice du même montant. Il en résulte un cadre WRT à corps vertical, à force verticale, qui soulève ou abaisse le quadrotor. Si l'hélicoptère est en position horizontale, la direction verticale du cadre inertiel et celle du cadre fixé au corps coïncident. Sinon, la poussée fournie génère à la fois accélérations verticales et horizontales dans le cadre inertiel.

La figure I.8 illustre la commande de tangage pour le quadrotor.

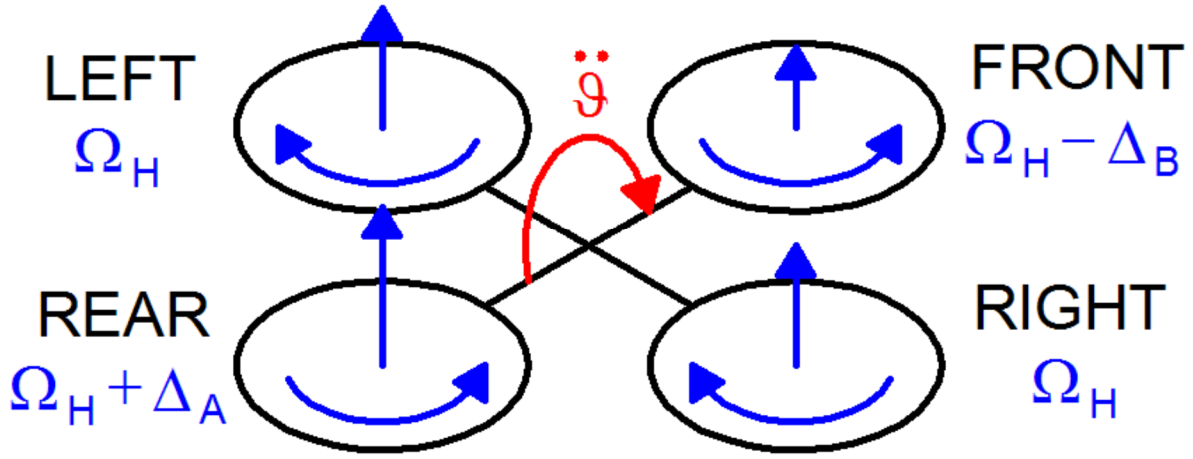


Figure I.8. Pitch (Tangage).

Cette commande est très similaire au Roulis et est obtenue en augmentant (ou en diminuant) la vitesse de l'hélice arrière et en diminuant (ou augmentant) la vitesse de l'hélice avant. Il en résulte un couple par rapport à l'axe y_B qui fait tourner le quadrotor. La poussée verticale globale est la même qu'en vol stationnaire, d'où cette commande ne conduit qu'à une accélération de l'angle de tangage (en première approximation).

La figure I.9 illustre la commande de Roulis pour le quadrotor.

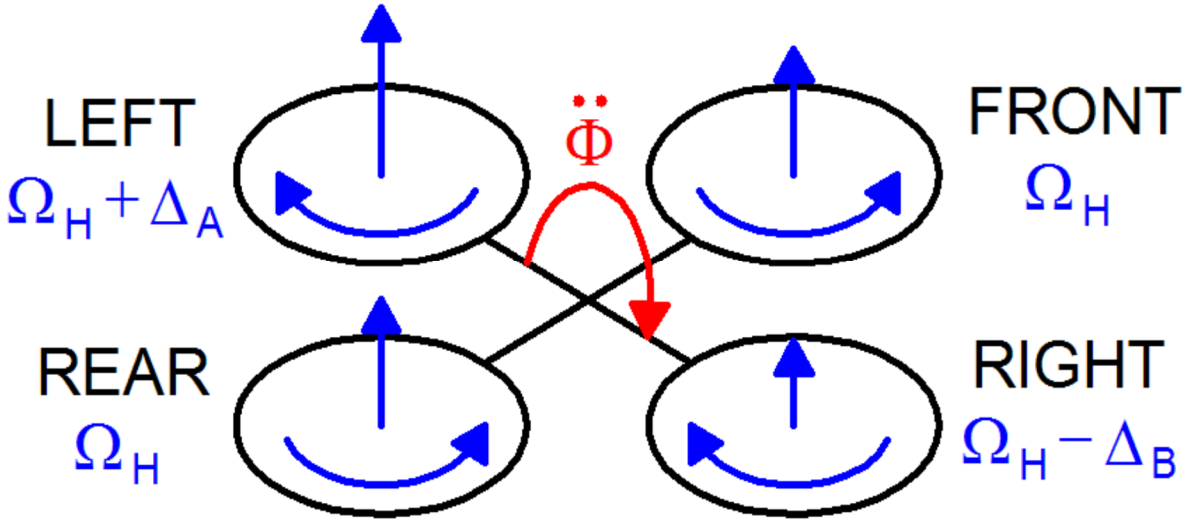


Figure I.9. Roll (Roulis).

Cette commande est fournie en augmentant (ou en diminuant) la vitesse de l'hélice gauche et en diminuant (ou augmentant) la bonne. Il en résulte un couple par rapport à l'axe x_B qui fait tourner le quadrotor. La poussée verticale globale est la même qu'en vol stationnaire, par conséquent, cette commande ne conduit qu'à une accélération de l'angle de roulis (en première approximation).

La figure I.10 illustre la commande de lacet pour le quadrotor..

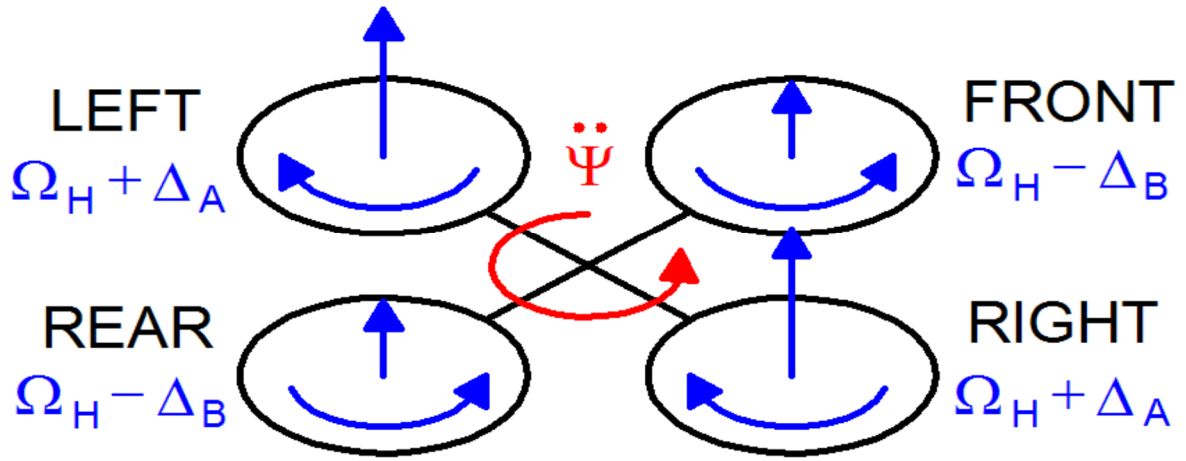


Figure I.10. Yaw (Lacet).

Cette commande est obtenue en augmentant (ou en diminuant) la vitesse des hélices avant-arrière et en diminuant (ou augmentant) celle du couple gauche-droit. Il en résulte un couple par rapport à l'axe z_B qui fait tourner le quadriloteur. Le mouvement de lacet est généré grâce au fait que la gauche-droite les hélices tourne dans le sens des aiguilles d'une montre, tandis que les hélices avant-arrière tournent dans le sens inverse des aiguilles d'une montre. Par conséquent, lorsque le couple global est déséquilibré, l'hélicoptère tourne autour de lui-même autour de z_B . La poussée verticale totale est la même qu'en vol stationnaire, par conséquent, cette commande ne conduit qu'à une accélération de l'angle de lacet (en première approximation).

I.4.1. Angles d'Euler

Les angles d'Euler sont trois angles introduits par Leonhard Euler pour décrire l'orientation d'un corps rigide. Pour décrire une telle orientation dans l'Espace Euclidien Tridimensionnel, trois paramètres sont requis. Ils peuvent être donnés en plusieurs façons : nous allons utiliser les angles de ZYX Euler [18]. Ils sont également utilisés pour décrire l'orientation d'un cadre de référence par rapport à un autre et ils transforment les coordonnées d'un point dans un repère dans les coordonnées du même point dans un autre cadre de référence. Les angles d'Euler sont généralement notés $\emptyset \in]-\pi,\pi], \theta \in]\frac{\pi}{2},\frac{\pi}{2}[, \omega \in]-\pi,\pi]$. Les angles d'Euler représentent une séquence de trois rotations élémentaires, à savoir : rotations autour des axes d'un système de coordonnées, car toute orientation peut être obtenu en composant trois rotations élémentaires. Ces rotations commencent à partir d'une orientation standard connue. Cette combinaison utilisée est décrite par les matrices de rotation suivantes :

$$R_x(\emptyset) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\emptyset) & -s(\emptyset) \\ 0 & s(\emptyset) & c(\emptyset) \end{bmatrix} \quad \text{(I.1)}$$

$$R_y(\theta) = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ 0 & 1 & 0 \\ -s(\theta) & 0 & c(\theta) \end{bmatrix} \quad \text{(I.2)}$$

$$R_z(\psi) = \begin{bmatrix} c(\psi) & -s(\psi) & 0 \\ s(\psi) & c(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{(I.3)}$$

Où $c(\emptyset) = \cos(\emptyset), s(\emptyset) = \sin(\emptyset), c(\theta) = \cos(\theta), s(\theta) = \sin(\theta), c(\psi) = \cos(\psi), s(\psi) = \sin(\psi)$
Donc, les coordonnées de la position inertielle et la référence du corps les coordonnées sont liées par la matrice de rotation $R_{zyx}(\emptyset, \theta, \psi) \in SO(3)$:

$$R_{zyx}(\emptyset, \theta, \psi) = R_z(\psi).R_y(\theta).R_x(\emptyset)$$

$$= \begin{bmatrix} c(\theta)c(\psi) & s(\emptyset)s(\theta)c(\psi) - c(\emptyset)s(\psi) & c(\emptyset)s(\theta)c(\psi) + s(\emptyset)s(\psi) \\ c(\theta)s(\psi) & s(\emptyset)s(\theta)s(\psi) + c(\emptyset)c(\psi) & c(\emptyset)s(\theta)s(\psi) - s(\emptyset)c(\psi) \\ -s(\theta) & s(\emptyset)c(\theta) & c(\emptyset)c(\theta) \end{bmatrix} \quad \text{(I.4)}$$

Cette matrice décrit la rotation du système de référence du corps à la référence inertielle.

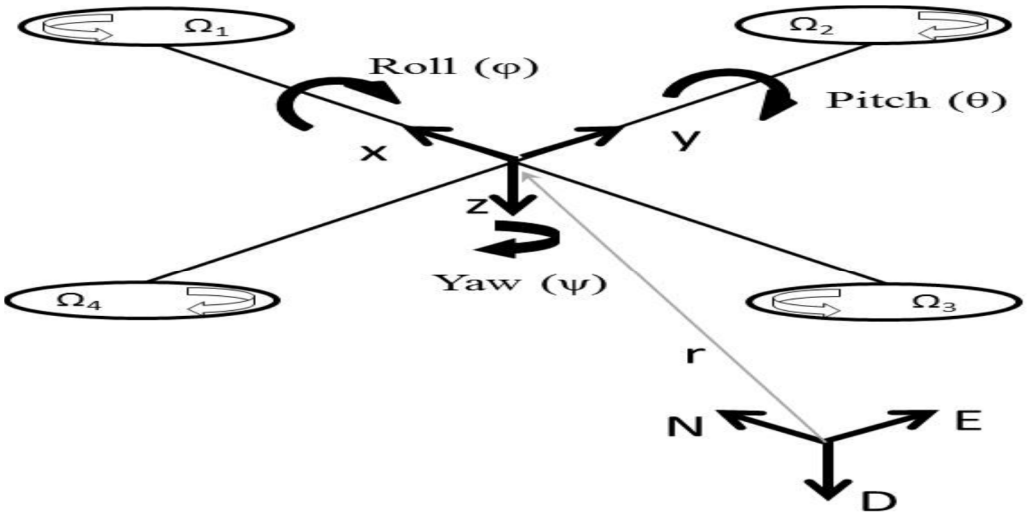


Figure I.11. Angles d'Euler pour un drone Quadrotor.

I.4.2. Modèle mathématique du quadrotor

Nous fournissons ici un modèle mathématique du quadrotor, exploitant les équations de Newton et d'Euler pour le mouvement 3D d'un corps rigide. Le but de cette section est d'approfondir la compréhension de la dynamique du quadrotor et de fournir un modèle suffisamment fiable pour simuler et contrôler son comportement. Appelons $[x\ y\ z\ \emptyset\ \theta\ \psi]^T$ le vecteur contenant la position linéaire et angulaire du quadrotor dans le cadre terrestre et $[u\ v\ w\ p\ q\ r]^T$ le vecteur contenant les vitesses linéaires et angulaires dans le cadre du corps. De la dynamique du corps 3D, il s'ensuit que les deux cadres de référence sont liés par les relations suivantes :

$$v = R \,.\, v_B \tag{I.5}$$

$$w = T \,.\, w_B \tag{I.6}$$

Où $v = [\dot{x}\ \dot{y}\ \dot{z}]^T \in \mathbb{R}^3, w = \left[\dot{\emptyset}\ \dot{\theta}\ \dot{\psi}\right]^T \in \mathbb{R}^3, v_B = [u\ v\ w]^T \in \mathbb{R}^3, w_B = [p\ q\ r]^T \in \mathbb{R}^3$

et T est une matrice pour les transformations angulaires [19]

$$T = \begin{bmatrix} 1 & s(\emptyset)t(\theta) & c(\emptyset)t(\theta) \\ 0 & c(\emptyset) & -s(\emptyset) \\ 0 & \frac{s(\emptyset)}{c(\theta)} & \frac{c(\emptyset)}{c(\theta)} \end{bmatrix} \tag{I.7}$$

Où $t(\theta) = \tan(\theta)$. Le modèle cinématique du quadrotor est donc :

$$\begin{cases} \dot{x} = w[s(\emptyset)s(\psi) + c(\emptyset)c(\psi)s(\theta)] - v[c(\emptyset)s(\psi) - c(\psi)s(\emptyset)s(\theta)] + u[c(\psi)c(\theta)] \\ \dot{y} = v[c(\emptyset)c(\psi) + s(\emptyset)s(\psi)s(\theta)] - w[c(\psi)s(\emptyset) - c(\emptyset)s(\psi)s(\theta)] + u[c(\theta)s(\psi)] \\ \dot{z} = w[c(\emptyset)c(\theta)] - u[s(\theta)] + v[c(\theta)s(\emptyset)] \\ \dot{\emptyset} = p + r[c(\emptyset)t(\theta)] + q[s(\emptyset)t(\theta)] \\ \dot{\theta} = q[c(\emptyset)] - r[s(\emptyset)] \\ \dot{\psi} = r\frac{c(\emptyset)}{c(\theta)} + q\frac{s(\emptyset)}{c(\theta)} \end{cases} \tag{I.8}$$

La loi de Newton énonce la relation matricielle suivante qui est la force totale agissant sur le quadrotor :

$$m(w_B \wedge v_B + \dot{v}_B) = f_B \tag{I.9}$$

Où m la masse du quadrotor $\wedge$ est le produit croisé et $f_B = [f_x\ f_y\ f_z]^T \in \mathbb{R}^3$ est la force totale.

L'équation d'Euler donne le couple total appliqué au quadrotor :

$$I.\dot{w}_B + w_B \wedge (I.w_B) = m_B \tag{I.10}$$

Où $m_B = [m_x\ m_y\ m_z]^T \in \mathbb{R}^3$est le couple total et I est la matrice diagonale d'inertie :

$$I = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \in \mathbb{R}^{3x3}$$

Ainsi, le modèle dynamique du quadrotor dans le cadre du corps est :

$$\begin{cases} f_x = m(\dot{u} + qw - rv) \\ f_y = m(\dot{v} - pw + ru) \\ f_z = m(\dot{w} + pr - qu) \\ m_x = \dot{p}I_x - qrI_y + qrI_z \\ m_y = \dot{q}I_y + prI_x - prI_z \\ m_z = \dot{r}I_z - pqI_x + pqI_y \end{cases} \tag{I.11}$$

Les équations sont valables tant que l'on suppose que l'origine et les axes du cadre coïncident avec le barycentre du quadrotor et les axes principaux.

I.4.2.1. Forces et moments

Les forces externes dans le cadre de la carrosserie, f_B sont données par :

$$f_B = mgR^T.\hat{e}_z - f_t\hat{e}_3 + f_w \quad \text{(I.12)}$$

Où $\hat{e}_z$est le vecteur unitaire dans l'axe inertiel z, $\hat{e}_3$ est le vecteur unitaire dans l'axe z du corps, g est l'accélération gravitationnelle, f_t est la poussée totale générée par les rotors et $f_w = \left[f_{wx}\, f_{wy}\, f_{wz}\right]^T \in \mathbb{R}^3$ sont les forces produites par le vent sur les quadrotors. Les moments externes dans le corps, m_B sont donnés par :

$$m_B = \tau_B - g_a + \tau_w \quad \text{(I.13)}$$

Où g_a représente les moments gyroscopiques causés par la rotation combinée des quatre rotors et de la carrosserie du quadrotor, $\tau_B = \left[\tau_x\, \tau_y\, \tau_z\right]^T \in \mathbb{R}^3$ sont les couples de commande générés par les différences de vitesses du rotor et $\tau_w = \left[\tau_{wx}\, \tau_{wy}\, \tau_{wz}\right]^T \in \mathbb{R}^3$ sont les couples produits par le vent sur les quadrotors. g_a est donné par :

$$g_a = \sum_{i=1}^{4} J_P(w_B \wedge \hat{e}_3)(-1)^{i+1}\Omega_i \quad \text{(I.14)}$$

où J_P est l'inertie de chaque rotor et Ω_i est la vitesse angulaire du ième rotor. D'après [14], le terme J_P s'avère être petit et, pour cette raison, les moments gyroscopiques sont supprimés dans la formulation du contrôleur. La figure I.12 montre les forces et les moments agissant sur le quadrotor.

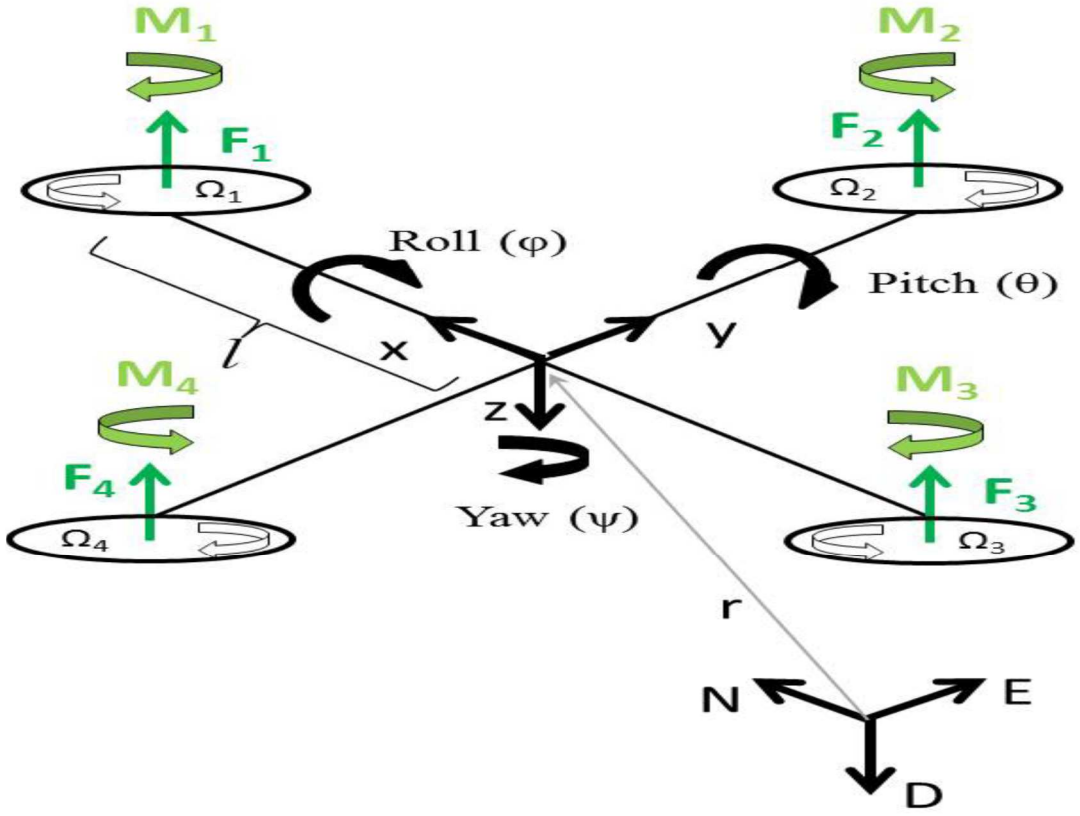


Figure I.12. Forces et moments agissant sur Quadrotor.

En outre, de nombreux phénomènes aérodynamiques et affectent le vol du quadrotor, tels que les effets de sol : lorsqu'il vole près du sol (ou lors de l'atterrissage), le flux d'air généré par les

hélices perturbe la dynamique du quadrotor. Ainsi, le modèle dynamique complet du quadrotor dans le cadre du corps est remplacé par l'expression de la force dans (I.11) :

$$\begin{cases} -m_g[s(\theta)] + f_{wx} = m(\dot{u} + qw - rv) \\ mg[c(\theta)s(\emptyset)] + f_{wy} = m(\dot{v} - pw + ru) \\ mg[c(\theta)c(\emptyset)] + f_{wz} - f_t = m(\dot{w} + pv - qu) \\ \tau_x + \tau_{wx} = \dot{p}\, I_x - qrI_y + qrI_z \\ \tau_y + \tau_{wy} = \dot{q}I_y + prI_x - prI_z \\ \tau_z + \tau_{wz} = \dot{r}I_z - pqI_x + pqI_y \end{cases} \quad \text{(I.15)}$$

I.4.2.2. Dynamique d'actionneur

Nous considérons ici les entrées pouvant être appliquées au système afin de contrôler le comportement du quadrotor. Les rotors sont au nombre de quatre et les degrés de liberté que nous contrôlons sont nombreux : généralement, les entrées de contrôle considérées sont une pour la poussée verticale et une pour chacun des mouvements angulaires. Considérons les valeurs des forces et couples d'entrée proportionnelles aux vitesses au carré des rotors ; leurs valeurs sont les suivantes :

$$\begin{cases} f_t = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ \tau_x = bl(\Omega_3^2 - \Omega_1^2) \\ \tau_y = bl(\Omega_4^2 - \Omega_2^2) \\ \tau_z = d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2) \end{cases} \quad \text{(I.16)}$$

Où l est la distance entre chaque rotor et le centre du drone, b le facteur de poussée et d le facteur de traînée. En substituant (I.16) en (I.15), nous avons : Ainsi, le modèle dynamique du quadrotor dans le cadre du corps est :

$$\begin{aligned} -m_g[s(\theta)] + f_{wx} &= m(\dot{u} + qw - rv) \\ mg[c(\theta)s(\emptyset)] + f_{wy} &= m(\dot{v} - pw + ru) \\ mg[c(\theta)c(\emptyset)] + f_{wz} - f_t = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) &= m(\dot{w} + pv - qu) \\ bl(\Omega_3^2 - \Omega_1^2) + \tau_{wx} &= \dot{p}\, I_x - qrI_y + qrI_z \\ bl(\Omega_4^2 - \Omega_2^2) + \tau_{wy} &= \dot{q}I_y + prI_x - prI_z \\ d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2) + \tau_{wz} &= \dot{r}I_z - pqI_x + pqI_y \end{aligned} \quad \text{(I.17)}$$

I.4.2.3. Modèle d'espace d'état

Organiser le vecteur de l'État de la manière suivante :

$$x = [\emptyset\ \theta\ \psi\ r\ u\ v\ w\ x\ y\ z]^T \in \mathbb{R}^{12} \quad \text{(I.18)}$$

$$\begin{cases} \dot{\emptyset} = p + r[c(\emptyset)t(\theta)] + q[s(\emptyset)t(\theta)] \\ \dot{\theta} = q[c(\emptyset)] - r[s(\emptyset)] \\ \dot{\psi} = r\frac{c(\emptyset)}{c(\theta)} + q\frac{s(\emptyset)}{c(\theta)} \\ \dot{p} = \frac{I_y - I_z}{I_x}rq + \frac{\tau_x + \tau_{wx}}{I_x} \\ \dot{q} = \frac{I_z - I_x}{I_y}pr + \frac{\tau_y + \tau_{wy}}{I_y} \\ \dot{r} = \frac{I_x - I_y}{I_z}pq + \frac{\tau_z + \tau_{wz}}{I_z} \\ \dot{u} = rv - qw - g[s(\theta)] + \frac{f_{wx}}{m} \\ \dot{v} = pw - ru + g[s(\emptyset)c(\theta)] + \frac{f_{wy}}{m} \\ \dot{w} = qu - pr + g[c(\theta)c(\emptyset)] + \frac{f_{wz} - f_t}{m} \\ \dot{x} = w[s(\emptyset)s(\psi) + c(\emptyset)c(\psi)s(\theta)] - v[c(\emptyset)s(\psi) - c(\psi)s(\emptyset)s(\theta)] + u[c(\psi)c(\theta)] \\ \dot{y} = v[c(\emptyset)c(\psi) + s(\emptyset)s(\psi)s(\theta)] - w[c(\psi)s(\emptyset) - c(\emptyset)s(\psi)s(\theta)] + u[c(\theta)s(\psi)] \\ \dot{z} = w[c(\emptyset)c(\theta)] - u[s(\theta)] + v[c(\theta)s(\emptyset)] \end{cases} \quad \text{(I.19)}$$

Ci-dessous, nous obtenons deux formes alternatives du modèle dynamique utiles pour étudier le contrôle. À partir de la loi de Newton, nous pouvons écrire :

$$m\dot{v} = R.f_B = mg\,\hat{e}_z - f_t R.\hat{e}_3 \quad \text{(I.20)}$$

Donc :

$$\begin{cases} \ddot{x} = -\frac{f_t}{m}[s(\emptyset)s(\psi) + c(\emptyset)c(\psi)s(\theta)] \\ \ddot{y} = -\frac{f_t}{m}[c(\emptyset)s(\psi)s(\theta) - c(\psi)s(\emptyset)] \\ \ddot{z} = g - \frac{f_t}{m}[c(\emptyset)c(\theta)] \end{cases} \quad \text{(I.21)}$$

Maintenant, une simplification est faite en plaçant $\left[\dot{\emptyset}\ \dot{\theta}\ \dot{\psi}\right]^T = [p\ q\ r]^T$. Cette hypothèse est vraie pour les petits angles de mouvement. Ainsi, le modèle dynamique du quadrotor dans le cadre inertiel est :

$$\begin{cases} \ddot{x} = -\frac{f_t}{m}[s(\emptyset)s(\psi) + c(\emptyset)c(\psi)s(\theta)] \\ \ddot{y} = -\frac{f_t}{m}[c(\emptyset)s(\psi)s(\theta) - c(\psi)s(\emptyset)] \\ \ddot{z} = g - \frac{f_t}{m}[c(\emptyset)c(\theta)] \\ \ddot{\emptyset} = \frac{I_y - I_z}{I_x}\dot{\theta}\dot{\psi} + \frac{\tau_x}{I_x} \\ \ddot{\theta} = \frac{I_z - I_x}{I_y}\dot{\emptyset}\dot{\psi} + \frac{\tau_y}{I_y} \\ \ddot{\psi} = \frac{I_x - I_y}{I_z}\dot{\emptyset}\dot{\theta} + \frac{\tau_z}{I_z} \end{cases} \quad \text{(I.22)}$$

Redéfinir le vecteur de l'État comme suit :

$$x = [x\ y\ z\ \psi\ \theta\ \emptyset\ \ \dot{x}\ \dot{y}\ \dot{z}\ p\ q\ r\ ]^T \in \mathbb{R}^{12} \quad \text{(I.23)}$$

Il est possible de réécrire les équations du quadrotor dans l'espace d'états :

$$\dot{X} = f(X) + \sum_{i=1}^{4} g_i(X)\, u_i \tag{I.24}$$

Où

$$f(X) = \begin{cases} \dot{x} \\ \dot{y} \\ \dot{z} \\ q\,\frac{s(\emptyset)}{c(\theta)} + r\,\frac{c(\emptyset)}{c(\theta)} \\ q[c(\emptyset)] - r[s(\emptyset)] \\ q[s(\emptyset)\,t(\theta)] + r\,[c(\emptyset)t(\theta)] \\ 0 \\ O \\ g \\ \frac{I_y - I_z}{I_x}\,qr \\ \frac{I_z - I_x}{I_y}\,pr \\ \frac{I_x - I_y}{I_z}\,pq \end{cases} \tag{I.25}$$

Et

$$g_1(X) = [0\ 0\ 0\ 0\ 0\ 0\ g_1^7\ g_1^8\ g_1^9\ 0\ 0\ 0]^T \in \mathbb{R}^{12}$$

$$g_2(X) = \left[0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \frac{1}{I_x}\ 0\ 0\right]^T \in \mathbb{R}^{12}$$

$$g_3(X) = \left[0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \frac{1}{I_y}\ 0\right]^T \in \mathbb{R}^{12}$$

$$g_4(X) = \left[0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \frac{1}{I_z}\right]^T \in \mathbb{R}^{12}$$

Avec :

$$g_1^7 = -\frac{1}{m}\,[s(\emptyset)s(\psi) + c(\emptyset)c(\psi)s(\theta)]$$

$$g_1^8 = -\frac{1}{m}\,[c(\psi)s(\emptyset) - c(\emptyset)s(\psi)s(\theta)]$$

$$g_1^9 = -\frac{1}{m}\,[c(\emptyset)\,c(\theta)]$$

I.4.3. Modèle linéaire

Définissez u le vecteur de contrôle : $u = \left[f_t\ \tau_x\ \tau_y\ \tau_z\right]^T \in \mathbb{R}^4$ La procédure de linéarisation est développée autour d'un point d'équilibre $\bar{x}$ qui, pour une entrée fixe $\bar{u}$, est la solution du système algébrique : ou plutôt cette valeur du vecteur d'état, qui sur une entrée constante est la solution du système algébrique :

$$\hat{f} = (\bar{x}\,, \bar{u}\,) = 0 \tag{I.26}$$

Puisque la fonction $\hat{f}$ est non linéaire, des problèmes liés à l'existence d'une unicité de la solution du système (I.26) se posent. En particulier, pour le système en main, la solution est difficile à trouver sous forme fermée à cause des fonctions trigonométriques liés les uns aux autres de manière non élémentaire [20]. Pour cette raison, la linéarisation est effectuée sur un modèle simplifié appelé à petites oscillations. Cette simplification est faite en approximant la fonction sinus avec son argument et la fonction cosinus avec unité. L'approximation est valide si l'argument est petit. Le système résultant est décrit par les équations suivantes :

$$\begin{cases} \dot{\emptyset} \approx p + r\,\theta + q\,\emptyset\,\theta \\ \dot{\theta} \approx q - r\emptyset \\ \dot{\psi} \approx r + q\emptyset \\ \dot{p} \approx \frac{I_y - I_z}{I_x} rq + \frac{\tau_x + \tau_{wx}}{I_x} \\ \dot{q} \approx \frac{I_z - I_x}{I_y} pr + \frac{\tau_y + \tau_{wy}}{I_y} \\ \dot{r} \approx \frac{I_x - I_y}{I_z} pq + \frac{\tau_z + \tau_{wz}}{I_z} \\ \dot{u} \approx rv - qw - g\,\theta + \frac{f_{wx}}{m} \\ \dot{v} \approx pw - ru + g\,\emptyset + \frac{f_{wy}}{m} \\ \dot{w} \approx q\,u - pr + g + \frac{f_{wz} - f_t}{m} \\ \dot{x} \approx w\,(\emptyset\,\psi + \theta) - v(\psi - \emptyset\,\theta) + u \\ \dot{y} \approx v\,(1 + \emptyset\,\psi\,\theta) - w\,(\emptyset - \psi\,\theta) + u\,\psi \\ \dot{z} \approx w - u\,\theta + v\,\emptyset \end{cases} \quad \text{(I.27)}$$

I.5. CONCLUSION

Ce chapitre nous a permis d'avoir des concepts préliminaires sur les robots volants et leur principe de fonctionnement. Le quadrotor est l'un des robots volants qui sont en investigation ces dernières années. Ce système est constitué de quatre rotors, deux de ces rotors tournent dans un sens et les deux autres dans le sens inverse. En variant les vitesses de rotation de ces rotors, le quadrotor peut faire des mouvements différents aussi bien en translation qu'en rotation.

L'utilisation du formalisme de Newton-Euler nous a permet d'établir le modèle dynamique du quadrotor. A partir du modèle obtenu, nous concluons que le quadrotor est un système sous actionné. De plus, la complexité du modèle, le non linéarité, et l'interaction entre les états du système, peuvent se voir clairement.

Dans le chapitre suivant nous aborderons une description matérielle est software utilisé dans notre travail de thèse.

Chapitre II :

Description de la Plateforme

Hardware et software

II.1. INTRODUCTION

À mesure que les technologies évoluent, elles se miniaturisent et gagnent en mobilité. Afin d'exploiter cette mobilité les systèmes embarqués se doivent d'être toujours plus performants et autonomes. Notre choix s'est porté sur l'AR.Drone PARROT pour son accessibilité du point de vue financier et technologique, sans oublier son appartenance à la famille quadrotor *open source* donnant la possibilité d'accès à des programmes sources et donnant ainsi l'opportunité aux développeurs de créer des applications à partir d'une plate-forme ouverte nommée ROS (Robot Operating System) qui nous avons choisie en raison de ces principes mentionnés dans ce chapitre.
Ce chapitre focalise sur la partie matérielle et programmation (Hardware et software), les différentes parties du Quadrotor de AR. Drone seront détaillées, ainsi que l'organisation de ROS et leurs principes de fonctionnement, par la suite nous aborderons le logiciel fuzylite utilisé pour la conception des contrôleurs flous,en dernier, nous nous étalerons sur la partie Matlab / AR.Drone et les différents blocs utilisés.

II.2. PARTIE MATERIELLE

II.2.1. Le Quadrotor Parrot AR.Drone

L'entreprise française Parrot commercialise l'A.RDrone, premier drone destiné au grand public, sorti en août 2010 en France. Il s'agit d'un petit multi-rotor quadrotor au design futuriste. Il peut se piloter avec un appareil sous iOS (iPhone, iPod touch, iPad), Android ou Symbian (Téléphones Nokia) via une liaison Wifi. La figure II.1, illustre une photo du Quadrotor AR.Drone.

.

Figure 0II.1. AR.Drone Parrot.

L'AR.Drone PARROT est une plateforme quadrotor équipée de moteurs brushless, lui assurant sa sustentation. Chaque moteur se situe à l'extrémité d'un des quatre bras.

Afin de piloter les moteurs, une carte spécialement dédiée à leur manipulation se situe près d'eux. Elle permet de retransmettre les ordres envoyés par le noyau et la carte mère du drone. Un microprocesseur ARM 9 est ainsi embarqué et gère le drone [21]. Par exemple, lorsque le drone doit se déplacer longitudinalement, l'ARM va communiquer avec chaque carte de gestion des moteurs afin de pouvoir leur transmettre les ordres. C'est par les différences de vitesses de rotation de chaque moteur que le mouvement est insufflé au drone. Cela est également possible grâce au fait que chaque paire d'hélices est contrarotative par rapport à l'autre.

Même si le drone est soumis à la dynamique des vols pour assurer sa stabilité, différents gyroscopes, une caméra orientée vers le sol et des senseurs à ondes ultrasonores sont présents pour que l'ARM puisse stabiliser le drone et acquérir des informations quant à la vitesse et la distance au sol de la plateforme. Toute cette batterie de senseurs est présente pour rendre automatique le décollage, l'atterrissage ainsi que certains types de hover de la part du drone. Les trims sont également gérées automatiquement, grâce à la présence d'un fichier dans le noyau du drone, définissant les valeurs par défaut, les paramètres physiques de l'AR.Drone sont dans l'annexe 1.

Le Parrot possède également une caméra embarquée permettant de filmer l'avant du véhicule. Celle-ci peut être utilisé pour retransmettre en temps réel les images. En effet, le drone est une plateforme wifi et peut donc communiquer avec diverses plateformes interactives (smartphone, tablette, ordinateur, …).

II.2.2. Présentation des différentes parties de l'AR Drone

La carte mère embarque un processeur Parrot P6 spécialement conçu pour l'AR.Drone. Elle est composée de :

❖ ***Système électronique embarqué***

- Processeur ARM Cortex A8 Cadencé @1 GHz
- DSP Vidéo Cadencé @800 MHz
- 128 Mo de RAM DDR2 à 200 MHz
- 128 Mo de Flash
- Wi-Fi b/g/n
- USB high speed (2.0)
- Linux OS 2.6.32

❖ ***Système de guidage inertiel***

- Accéléromètre MEMS 3-axes
- Gyroscope à 3-axes
- Magnétomètre à 3-axes
- Capteur de Pression

❖ ***Spécifications***

- Vitesse en vol : 5 m/s ; 18 km/h (16,4 fps ; 11,2 mph)
- Poids :
  - 380 g avec la coque d'extérieur (0,8 livres)
  - 420 g avec la coque d'intérieur (0,9 livres)
- Autonomie : environ 12 minutes

❖ ***Sécurité***

- Coque en EPP pour les vols en intérieur.
- Arrêt automatique des hélices en cas de contact
- Batterie UL2054
- Interface de contrôle avec un bouton d'arrêt d'urgence (coupe l'alimentation des moteurs)

❖ ***Structure aéronautique***

- Hélices à haute efficacité spécifique.
- Structure en fibre de carbone.

❖ ***Moteurs et énergie***

- 4 moteurs brushless (28 500 tr/min, puissance : 14,5 W)
- Batterie Lithium-polymère (trois cellules ; 11,1 V ; 1 000 mAh)
- Capacité de décharge : 10 C
- Temps de rechargement de la batterie : 90 minutes

❖ ***Caméra frontale***

- Caméra grand angle 92°, capteur CMOS
- Enregistrement et diffusion directe des images sur iPhone ou sur clef USB
- Résolution caméra de 1280x720 pixels @30 fps (HD)

❖ ***Caméra Verticale***

- Caméra à haute vitesse. 64° diagonale de la lentille, Capteur CMOS
- Enregistrement et diffusion directe des images sur iPhone ou sur clef USB
- Résolution de 320x240 (QVGA) @60 fps
- Utilisée pour mesurer la vitesse au sol
- Permet la stabilisation, même avec un vent léger

❖ ***Autres détecteurs de l'AR.Drone***

- Validation des tirs des drones ennemis

- Estimation de la distance
- Positionnement d'objets virtuels
- Calcul des marqueurs d'objets virtuels
- Distance de détection : de 30 centimètres à 5 mètres (1 à 16,4 pieds)

❖ ***Altimètre à ultrasons***

- Fréquence d'émission : 40 kHz
- Portée de 6 mètres (19,7 pieds)
- Sensibilité verticale

Les figures II.2 et II.3, illustrent les différentes faces de la carte mère du Quadrotor AR.Drone.

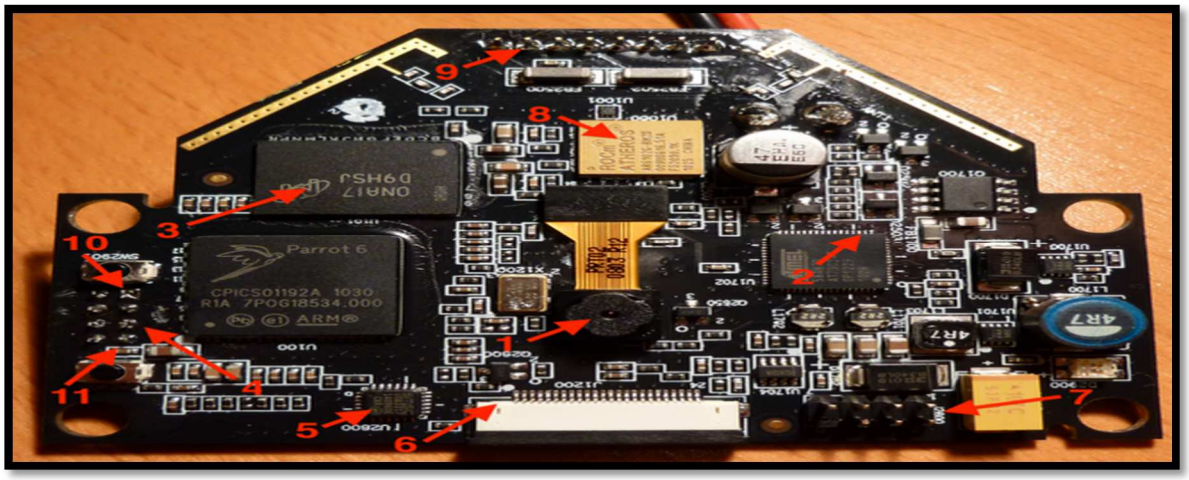


Figure II.2. Vue de dessus de la carte mère.

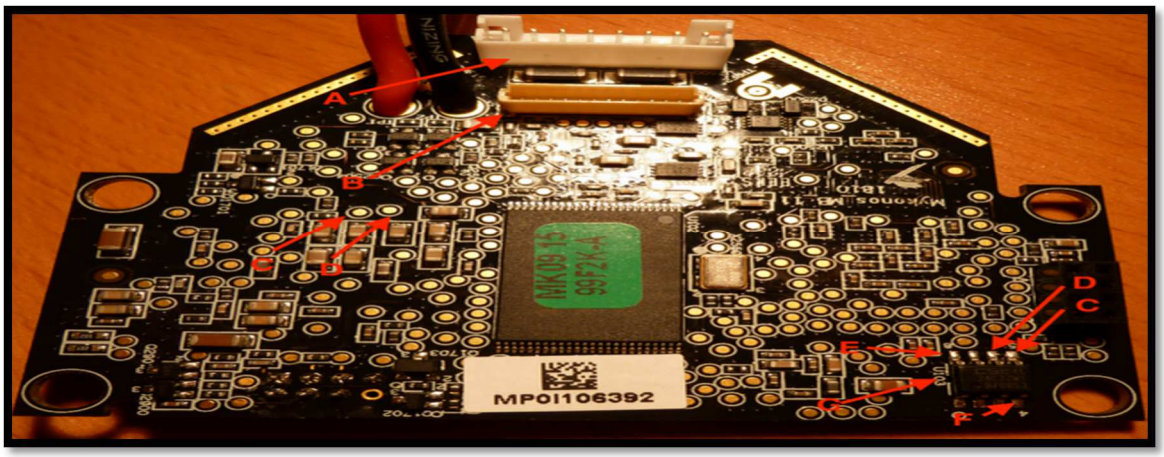


Figure II.3. Vue du dessous de la carte mère.

II.2.3. La vision par caméra

Le drone dispose de deux caméras CMOS :

✓ Une verticale située sous le drone qui permet de visualiser le terrain avant l'atterrissage : d'angle 64° en résolution QCIF à une cadence de 60 fps.

- ✓ Une horizontale située à l'avant du drone qui permet de voir où se déplace le drone et de détecter les mires pour la réalité augmentée. Son angle est de 93° en résolution VGA à une cadence de 15 fps.

La carte mère intègre également la caméra verticale de l'AR. Drone. Sa fréquence est de 60 images par seconde et sa résolution de 176*144 pixels (standard QCIF). La figure II.4, montre une photo de la caméra verticale.

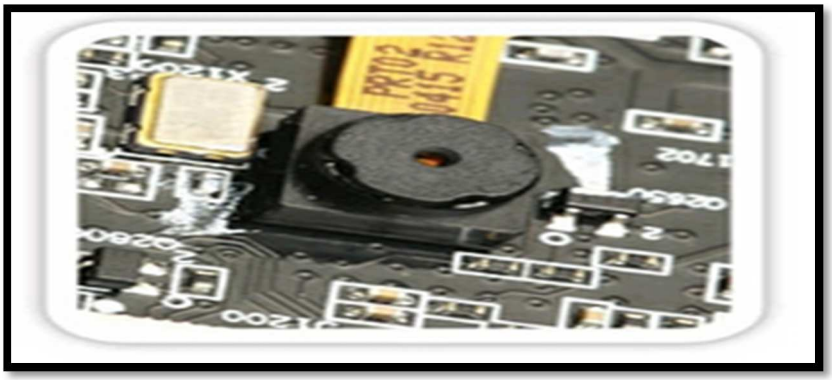


Figure II.4. Caméra verticale.

Les données de cette caméra couplées à celles des capteurs de la carte de navigation permettent de stabiliser l'AR Drone. La vitesse de déplacement de l'AR Drone est reconstruite en corrélant les points significatifs des différentes images successives. La figure II.5 illustre une description du champ de vision de la caméra verticale de l'AR.Drone.

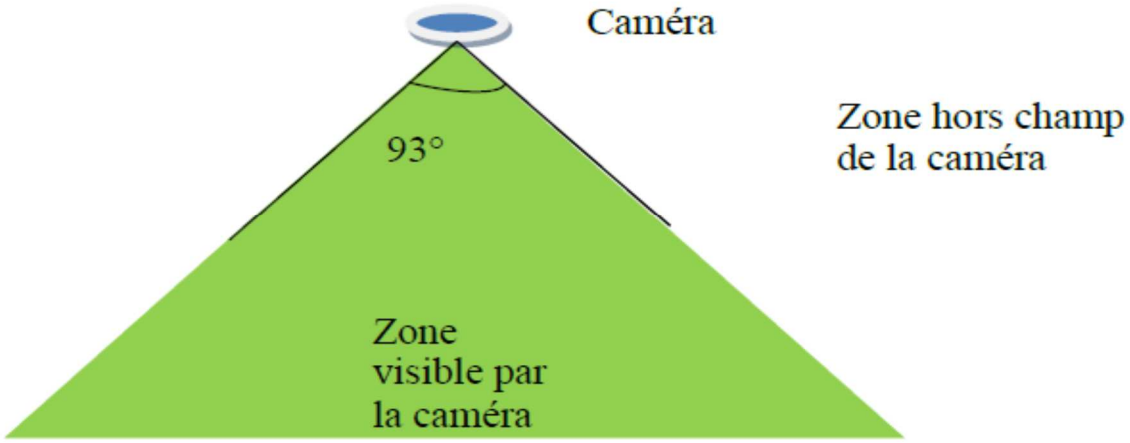


Figure II.5. Vue d'ensemble du champ de vision de la caméra verticale du Drone.

Les deux caméras du drone utilisent des objectifs différents. Un objectif est une lentille (ou un jeu de lentilles) qui permet de diriger (de faire converger) les rayons lumineux vers le capteur.

La structure de l'AR Drone est fabriquée en polypropylène expansé, un matériau qui combine à la fois légèreté et solidité. Elle comporte l'emplacement destiné à recevoir la batterie. Le support de batterie est monté sur la mousse pour isoler les cartes électroniques des vibrations

engendrées par les moteurs. La figure II.6, montre la photo de la structure du drone ainsi que le placement de la caméra frontale.

Figure II.6. Structure et caméra frontale.

La batterie est positionnée légèrement à l'arrière de l'appareil afin de faire coïncider le centre de gravité et le centre géométrique de l'appareil. La caméra frontale HD 720p de grand angle offre une vue panoramique à 93°. Elle est connectée à la carte mère par un flux qui passe sous le blister de protection. La vidéo de la caméra frontale est acquise en VGA puis compressée en QVGA à l'aide d'un codec MJPEG. Le flux vidéo est diffusé par WIFI vers l'iOS. Le logiciel embarqué crée son propre réseau WIFI-Ad hoc.

II.2.4. La croix centrale

La croix centrale compose l'ossature de l'AR Drone. Les tubes sont fabriqués en fibre de carbone, les supports moteurs en plastique PA 6,6. Ces matériaux ont été choisis pour procurer à la fois légèreté et résistance à la croix. La figure (II.7), montre la photo de la croix centrale.

Figure II.7. Photo de la croix centrale.

Les supports moteurs comportent chacun deux paliers lisses autolubrifiants pour une meilleure rotation de l'axe. Le démontage et le remontage des moteurs est aisé.

Une flèche est inscrite sous la croix pour indiquer comment elle doit être insérée dans la structure de l'AR. Drone.

Des numéros permettent aussi d'identifier les types d'hélices à installer sur les moteurs de chaque tube : moteur 1 : « C », moteur 2 : « A », moteur 3 « C » et moteur 4 : « A ».
La croix comporte deux faisceaux de câbles, un pour alimenter les moteurs en courant et un autre pour contrôler la vitesse de rotation. Les deux faisceaux sont divisés en quatre et sont terminés par des connecteurs pour chaque moteur.

II.2.5. Moteur Brushless

Les moteurs brushless ont été spécialement conçus pour l'AR Drone en vue de lui garantir de bonnes performances ainsi qu'une longue durée de vie. La figure suivante, montre la photo du moteur Brushless.

Figure II.8. Moteur brushless.

La puissance de chaque moteur est de 15 Watts, les moteurs effectuent 28000 tr/min en vol stabilisé, ce qui correspond à 3300 tr/min pour les hélices. La vitesse de rotation des moteurs peut varier entre 10350 et 41400 tr/min. Ce moteur est associé avec sa carte électronique, qui comprend un microcontrôleur basse consommation 8bits ainsi qu'un ADC 10 bits pour gérer la vitesse de rotation.

On a aussi 3 vis métalliques longues et 3 vis métalliques courtes. Les vis métalliques longues maintiennent la carte électronique au moteur et les trois vis courtes maintiennent le moteur à la croix centrale.

II.2.6. Les hélices

Ces hélices ont été conçues spécialement pour l'AR Drone par l'équipe qui a gagné le concours de micro drone organisé par la DGA. Elles ont été étudiées pour minimiser la consommation électrique des moteurs tout en procurant un maximum de poussée. La figure (II.9), montre une photo des hélices utilisées par Parrot.

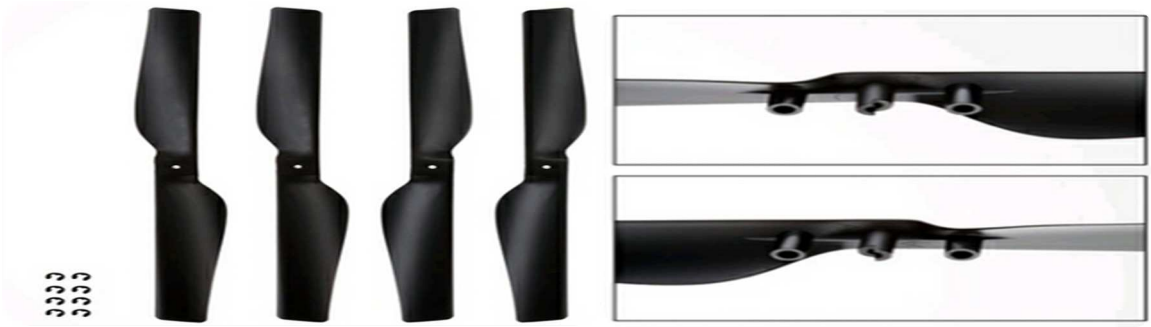


Figure II.9. Vue sur les Hélices.

Les hélices comportent l'inscription « C » ou « A ». Les hélices « C » tournent dans le sens horaire tandis que les hélices « A » tournent dans le sens antihoraire. Elles doivent être montées comme suit : moteur 1 : « C », moteur 2 : « A », moteur 3 « C » et moteur 4 : « A ». Les numéros des moteurs correspondent aux numéros des tubes sur lesquels ils sont montés. Ces numéros sont indiqués sous la croix centrale.

II.2.7. Batterie

La batterie est une batterie Lithium-Polymer trois cellules, d'une capacité de 1000 jusqu'à 1500 mAh à 11,1 volts. Elle inclut un circuit de protection (PCM) qui protège la batterie contre les charges et décharges excessives ainsi que des courts-circuits.

La figure II.10, montre le modèle de la batterie de Parrot.

Figure II.10. Lithium Polymère Batterie.

Le niveau de batterie est calculé en fonction de la tension délivrée effectivement par la batterie LiPo (caractéristiques nominales : 11,1volts et 1A.h). Si la batterie a une tension de 12,5 Volts alors son niveau affiché est de 100% si elle délivre 9,5 volts alors son niveau est de 0%. Pour éviter qu'une batterie trop faible cause une casse matérielle (en déconnectant le Wifi ou n'entrainant plus suffisamment les moteurs) le drone se met en sécurité en atterrissant quelles que soient les commandes de l'utilisateur.

La batterie possède deux connecteurs : un connecteur de décharge pour alimenter l'AR.Drone et un autre connecteur de charge.

II.2.8. La carte de navigation

La carte de navigation comprend un microcontrôleur couplé à un convertisseur ADC ainsi que de nombreux capteurs [22]. La figure II.11, illustre une vue générale de la carte de navigation.

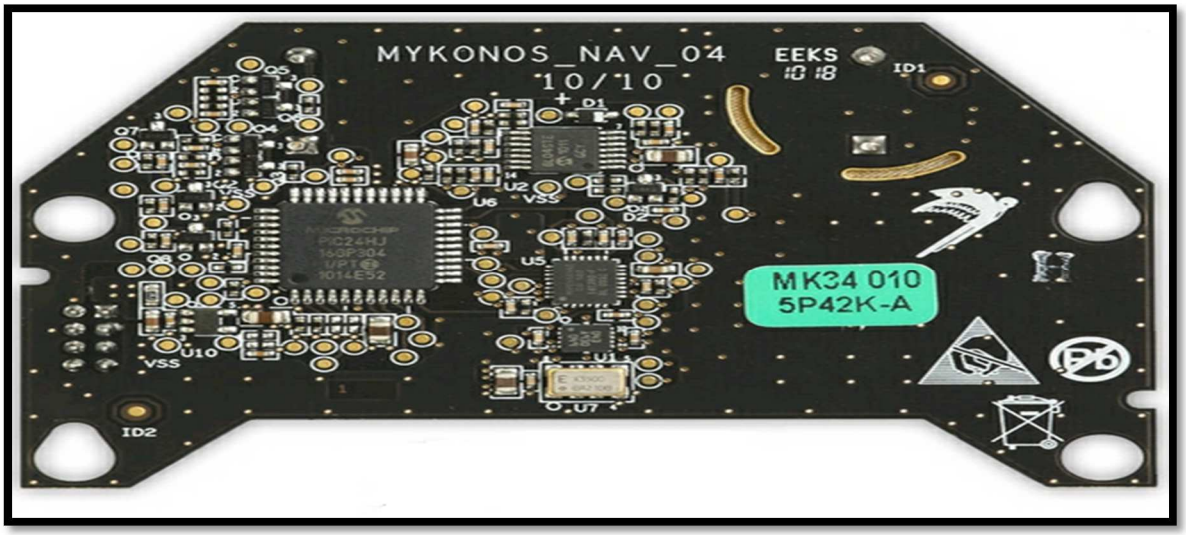


Figure II.11. Carte de navigation.

La carte de navigation comprend :

- Un capteur à ultrasons, composé d'un transmetteur et d'un récepteur. Il permet de mesurer l'altitude de l'AR Drone, et ce jusqu'à six mètres d'altitude. La mesure s'effectue à une fréquence de 25 Hz.
- Un accéléromètre numérique trois axes, positionné au centre de gravité de l'appareil. L'accéléromètre est utilisé dans la gamme ± 2 g. Un ADC 10bits intégré permet l'acquisition des données à 200 Hz. Les données sont envoyées au microcontrôleur par une liaison I2C.
- Un gyromètre MEMS sur deux axes et un gyromètre piézoélectrique de précision pour la mesure de cap, permettent de mesurer et de contrôler les vitesses de rotation du drone. La plage de mesures de ces capteurs est de ±500 °/s. Ces capteurs analogiques sont numérisés par l'ADC 12 bits du microcontrôleur.

Ces capteurs composent l'unité de mesure inertielle. La mise en commun des informations en provenance de ces capteurs permet de calculer les angles d'Euler de l'AR Drone, utiles pour stabiliser l'appareil.

II.2.9. La hauteur

Le télémètre à ultrasons permet de mesurer la distance entre la partie inférieure du drone et ce qui se trouve en dessous. Ce télémètre émet des ultrasons à 40 kHz (vibration de l'air à une fréquence supérieure à celle audible par l'homme, soit, supérieure à 20 kHz) avec une partie émettrice (notée Tx) et reçoit l'onde une fois qu'elle a rebondi sur l'obstacle (phénomène d'écho) avec une partie réceptrice (notée Rx). En fonction du temps de rebond de l'onde, le capteur en déduit la distance de l'obstacle. La figure II.12, montre une image du capteur à ultrasons.

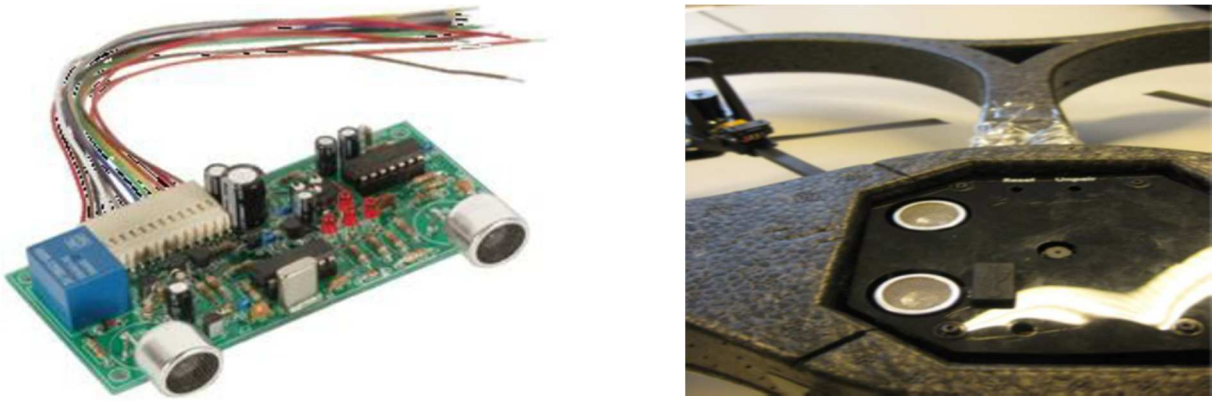


Figure II.12. Sonar équipant l'AR.Drone.

Pour comprendre un peu mieux les différents protocoles suivi par l'AR drone dans la gestion interne d'informations, voici un schéma synoptique le résumant, ainsi on peut suivre les étapes depuis l'acquisition des données par les différents capteurs jusqu'à la stabilisation totale assurée par les quatre moteurs ou le mouvement désiré grâce à des signaux PWM générés par l'ARM9.

La figure suivante, montre le schéma global du fonctionnement de l'AR.Drone [23].

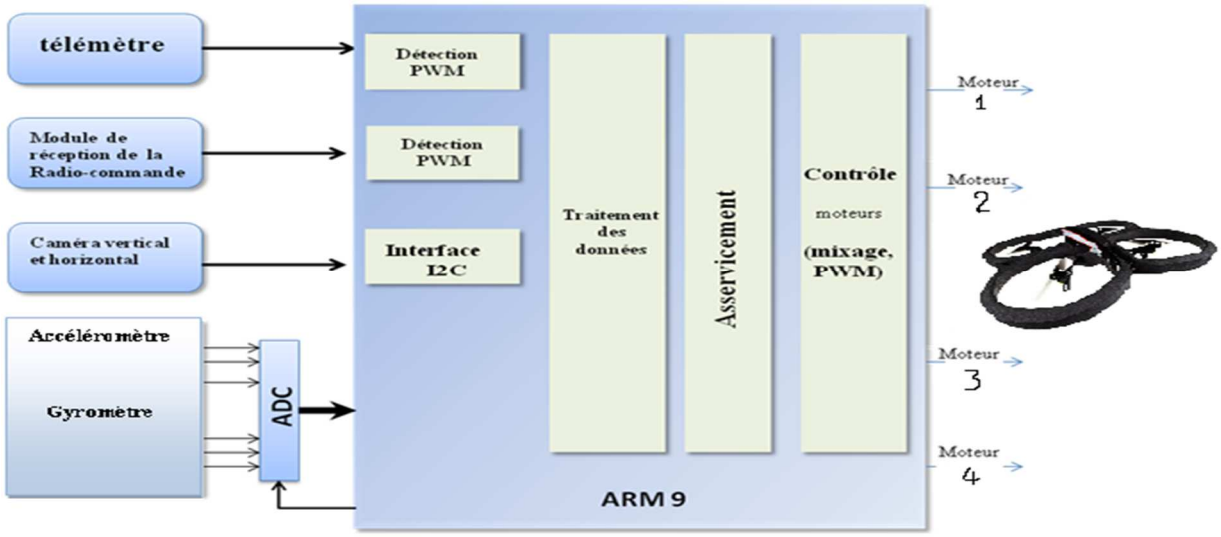


Figure II.13. Fonctionnement interne de l'AR drone.

L'ARM926EJ sera chargé de :

- Du mixage des ordres (lacet, tangage, roulis, et puissance moteurs) wifi et génération de la PWM pour piloter chaque moteur individuellement.
- De la récupération des données des deux caméras verticale et horizontale via une interface I2C.
- De la récupération des données du télémètre ou de wifi via une détection PWM.
- De l'acquisition (commandes de l'ADC) et du traitement des données numérisées issues des gyromètres et accéléromètres.
- Du démarrage automatique et de la sustentation à une certaine hauteur grâce au télémètre.
- Des asservissements du lacet, du tangage et du roulis.

II.3. La communication avec le Drone

Le fabriquant a réalisé un drone facilement pilotable par Wifi afin que ceux qui le veulent puissent développer leurs propres applications sur la plateforme de leur choix.

1- Le drone réagit par rapport à des commandes qui lui indiquent :

- De décoller, d'atterrir
- De monter, de descendre
- De tourner à gauche, de tourner à droite
- D'avancer, de reculer
- De se déplacer sur la gauche ou sur la droite

2- Il est aussi possible de mettre à jour le logiciel embarqué dans le drone (le firmware). C'est lui qui permet de comprendre les ordres reçu et fait fonctionner les moteurs en fonction de ceux-ci.

3- Le drone envoie en permanence des flux vidéo de ses 2 caméras

4- Le drone envoie en permanence des informations sur son état (niveau de charge de la batterie, orientation spatiale , position des cibles de réalité augmentée ...)

Pour initialiser l'envoi de données du drone au PC il faut que le logiciel de pilotage envoie d'abord une commande vers le drone. La figure II.14, illustre un schéma synoptique pour la communication du Drone.

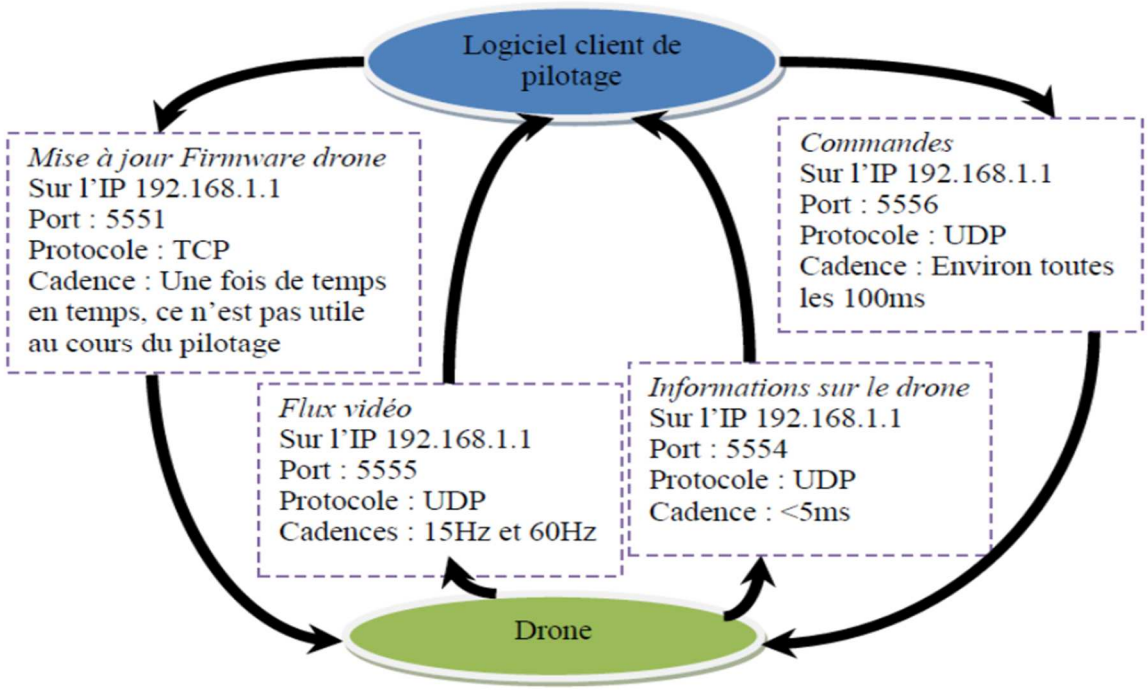


Figure II.14.Schéma synoptique de communication.

Nous allons dans un premier temps décrire la relation client-serveur entre le PC et le drone. Le serveur est dans notre cas le drone, en effet c'est bien notre PC (le client) qui va se connecter au multi-rotor pour lui donner des ordres. En retour le drone donnera des réponses sous forme de mouvements ou de données. Les deux acteurs de la communication sont chacun identifiés clairement par une adresse IP (Internet Protocol). Le protocole de communication est soit le TCP (Transmission Control Protocol) ou UDP (User Data-gram Protocol). On peut comprendre la différence entre ces deux protocoles à travers les deux analogies suivantes :

- **TCP :** on peut le comparer à une conversation téléphonique. La communication s'effectuera uniquement à condition que les deux contacts aient décrochés le téléphone et dits « Allo ?!?». Une fois la communication établie, il n'y a quasiment aucun risque de perte ou de modification de données. Ce protocole garantie la transmission des données.
- **UDP :** peut être comparé au service postal. Lorsqu'une lettre est postée il se peut qu'elle se perde ou bien qu'elle arrive à destination après une lettre qui avait pourtant été postée après. De la même manière, deux processus peuvent aussi utiliser le protocole UDP pour s'envoyer des données. Avec UDP, aucune connexion n'est nécessaire mais contrairement au TCP, l'UDP ne garantit pas la qualité du service proposé. Certaines données peuvent être perdues, arrivées dans le désordre ou bien modifiées voire en double. Le contrôle, si nécessaire, devra être alors effectué/implémenté par l'utilisateur.

Dans notre cas, avec une connexion point à point, les données ne peuvent pas arriver dans le désordre, elles peuvent cependant être erronées ou modifiées.

On peut remarquer que l'UDP, ne nécessitant pas de vérification et utilisera moins de ressources. C'est pour cela que ce protocole a été choisi pour communiquer avec le quadrotor. Nous pouvons résumer le fonctionnement des deux protocoles par la figure II.15.

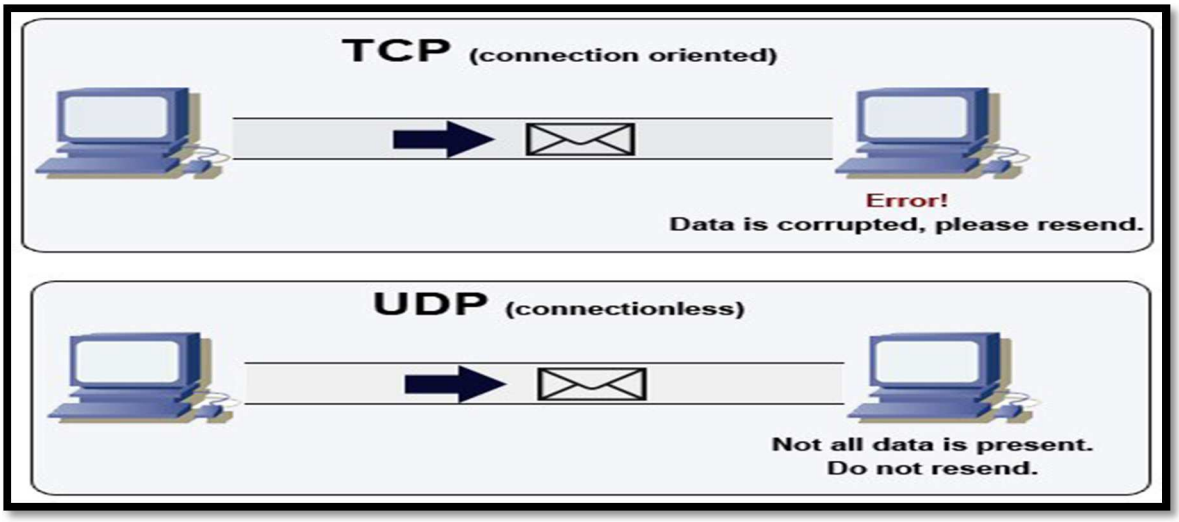


Figure II.15. Différence entre TCP et UDP

L'AR. Drone dispose de plusieurs ports sur lesquels transitent différents types d'informations via UDP, voici les principaux :

Tableau II.1. Différents Ports de communication.

Numéro de Port	Type de données transportées	Réception/Émission (client)
5556	Commandes bas-niveau dites AT	Emission
5554	Navigation (navdata)	Réception
5555	Flux vidéo	Réception

II.3. PARTIE LOGICIELLE

II.3.1. ROS (Robot Operating System)

ROS est un système d'exploitation complet pour la robotique de service. De même que les systèmes d'exploitation pour PC, serveurs ou appareils autonomes, ce méta système d'exploitation est entre le système d'exploitation et le middleware [24].

C'est un logiciel qui permet le développement des applications robotiques où il fournit des services proches d'un système d'exploitation (l'abstraction du matériel, la gestion multiple des

processus…) mais aussi des fonctionnalités de haut niveau (appels asynchrones, appels synchrones, base de données centralisée de données, système de paramétrage du robot…).

II.3.1.2. De l'intérêt d'un OS pour les robots

Avant les OS robotiques, chaque concepteur de robot, chaque chercheur en robotique passait un temps non négligeable à concevoir matériellement son robot ainsi que le logiciel embarqué associé. Cela demandait des compétences en mécanique, électronique et programmation embarquée. Généralement, les programmes ainsi conçus correspondaient plus à de la programmation embarquée, proche de l'électronique, qu'à de la robotique proprement dite, telle que nous pouvons la rencontrer aujourd'hui dans la robotique de service. La réutilisation des programmes était non triviale car fortement liée au matériel sous-jacent.

L'idée principale d'un OS robotique est d'éviter de réinventer la roue à chaque fois et de proposer des fonctionnalités standardisées faisant abstraction du matériel, tout comme un OS classique pour PC, d'où l'analogie de nom.

Un autre avantage de l'OS robotique comme ROS est de rassembler des savoir-faire de différentes disciplines. En effet, concevoir et programmer un robot, c'est :

- Gérer le matériel en écrivant les pilotes
- Gérer la mémoire et les processus
- Gérer la concurrence et la fusion de données
- Proposer des algorithmes de raisonnement abstrait faisant largement appel à l'intelligence artificielle

La robotique nécessite donc des compétences très différentes, compétences généralement hors de portée d'une seule personne.

II.3.1.3. L'histoire de ROS

Il existe de nombreux framework robotiques réalisés dans une optique précise, à des fins de prototypage. ROS a été voulu plus généraliste même si ses concepteurs pensent qu'il n'est pas l'OS ultime capable de tout faire.

ROS est développé et maintenu par une société californienne, Willow Garage, fondée en 2006 par Scott Hassan, un des premiers employés de Google qui a participé au développement de la technologie du moteur de recherche et qui est aussi à l'origine des Yahoo Groups (en fait de eGroups qui est devenu Yahoo Groups). Le PDG de Willow Garage est Steeve Cousin, un ancien d'IBM.

Willow Garage est une société privée qui entretient des liens étroits avec l'université de Stanford qui est proche géographiquement de Willow Garage (à Palo Alto en Californie).

Willow Garage se présente comme un laboratoire de recherche et un incubateur technologique pour la robotique personnelle, centré sur la recherche plus que sur les profits (au moins au début).

Willow Garage développe aussi bien du logiciel avec ROS que du matériel avec leurs robots PR2et TurtleBot. Dans tous les cas, ce qui est produit est open source (licences BSD). Leur idée

est que si l'on souhaite voir arriver les robots dans nos foyers, il faut pour cela accélérer les recherches en fournissant des bases hardware et software solides et qui soient open source.

Il semblerait que Willow Garage veuille bâtir la communauté de la robotique plutôt que la robotique en elle-même. Interrogé par le magazine l'Expansion, Scott Hassan indique que ses objectifs sont identiques à ceux d'Irobot mais que la stratégie pour y parvenir est différente.

4 raisons de croire au succès de Willow Garage d'après le singularityhub :

- Ils souhaitent proposer les moyens de ne plus réinventer la roue afin d'accélérer les recherches en robotique.
- Ils ont les fonds nécessaires.
- Ils ont l'oreille du monde de la recherche.
- Ils veulent favoriser le déploiement de leur technologie gratuitement avant de penser à gagner de l'argent.

II.3.1.4. L'organisation générale de ROS

La philosophie de ROS se résume dans les 5 grands principes suivants :

- Peer to Peer.
- Basé sur des outils.
- Multi langages.
- Léger.
- Gratuit et open source.

Reprenons chacun de ces principes :

Peer to Peer : Un robot suffisamment complexe est composé de plusieurs ordinateurs ou cartes embarquées reliées par Ethernet, parfois même des ordinateurs externes au robot pour des tâches de calcul intensif. Une architecture peer to peer couplée à un système de buffering et à un service spécial nommé master dans ROS, permettant à chacun des acteurs de dialoguer en direct avec un autre acteur, de manière synchrone ou asynchrone en fonction des besoins.

Basé sur des outils : ROS a adopté un design qui utilise un grand nombre de petits outils (nommés exécutables) pour communiquer ses différents composants. L'avantage de cette solution est que si un problème survient sur un exécutable cela n'affecte le reste du système, rendant ainsi le système plus robuste et plus évolutif qu'un système basé sur une architecture centralisé.

Multi langages : ROS est neutre d'un point de vue langage et peut être programmé en différents langages. La spécification de ROS intervient au niveau message. Les connexions peer to peer sont négociées en XML-RPC qui existe dans un grand nombre de langages. Pour supporter un nouveau langage, soit on reconditionne (rewrappe) les classes C++, soit on écrit les classes permettant de générer les messages qui sont décrits en IDL (Interface Definition Language).

Léger : Afin de lutter contre le développement d'algorithmes plus ou moins liés avec l'OS robotique et donc difficilement réutilisables, les pilotes et autres algorithmes sont contenus dans des exécutables indépendants. Cela assure la réutilisabilité maximale et surtout le maintien d'une taille réduite. Ce mécanisme rend ROS facile d'usage, la complexité se trouvant dans les librairies. Cette organisation facilite en plus le test unitaire. Enfin, ROS peut utiliser du code (pilotes et algorithmes) issus d'autres projets open source, tel que :

- Librairie de traitement d'image et de vision artificielle : Open CV.
- Algorithme de planification : Open Rave.
- Fuzzylite.

Gratuit et open source : Nous avons déjà expliqué les raisons de ce choix. Notons toutefois que l'architecture choisie est cohérente avec ce choix : ROS passe des données grâce à de la communication interprocess. De ce fait, les modules n'ont pas besoin d'être liés dans un unique process, facilitant ainsi l'usage des différentes licences.

II.3.1.5. Les grands principes de ROS

ROS n'est pas dépendant d'un langage. A ce jour, trois librairies principales ont été définies pour ROS qui permettent de programmer respectivement ROS en Python, en Lisp ou en C++. En plus de ces trois librairies, deux librairies expérimentales sont proposées, qui permettent de programmer ROS en Java ou en Lua [25].

Le principe de base d'un OS robotique est de faire fonctionner en parallèle un grand nombre d'exécutables qui doivent pouvoir échanger de l'information de manière synchrone ou asynchrone. Par exemple, un OS robotique doit interroger à une fréquence définie les capteurs du robot (capteur de distance à ultrasons ou infrarouge, capteur de pression, capteur de température, gyroscope, accéléromètre, caméras, microphones…), récupérer ces informations, les traiter (faire ce que l'on appelle la fusion de données), les passer à des algorithmes de traitement (traitement de la parole, vision artificielle, localisation et cartographie simultanée,…) et enfin contrôler les moteurs en retour. Tout ce processus s'effectue en continue et en parallèle. D'autre part, l'OS robotique doit assurer la gestion de la concurrence afin d'assurer l'accès efficace aux ressources du robot.

Nous décrivons ci-dessous les concepts regroupés dans ROS sous le nom de « ROS Computation Graph » et qui permettent d'atteindre ces objectifs. Il s'agit des concepts utilisés par le système en cours de fonctionnement.

Les nœuds : un nœud est une instance d'un exécutable. Il peut correspondre à un capteur, un moteur, un algorithme de traitement, de surveillance… Chaque nœud qui se lance se déclare au Master. La figure suivante, montre un schéma du fonctionnement interne de ROS.

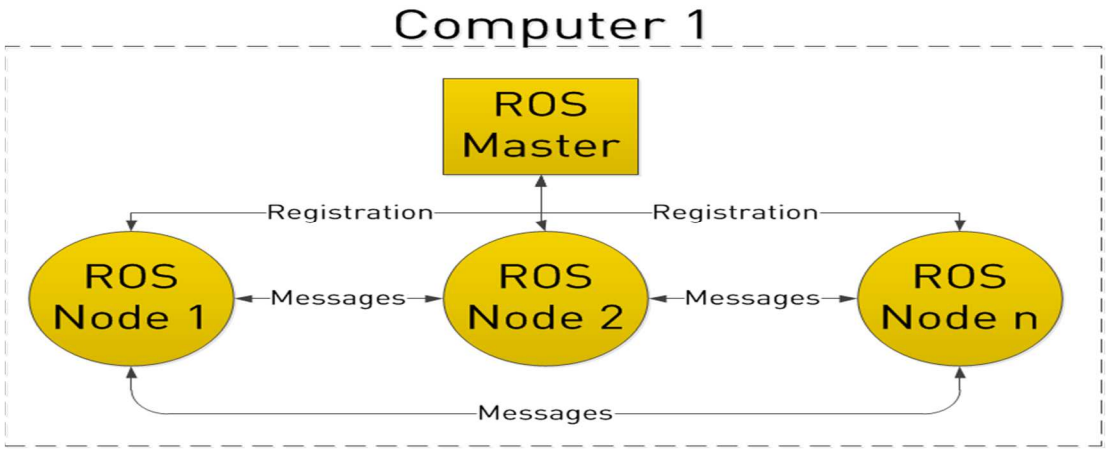


Figure II.16. Organisation de ROS.

Le master : Le Master est un service de déclaration et d'enregistrement des nœuds qui permet ainsi à des nœuds de se connaitre et d'échanger de l'information. Le Master est implémenté via XMLRPC. Le Master comprend une sous-partie très utilisée qui est le Paramètre Server. Celui-ci, également implémenté sous forme de XMLRPC, comme son nom l'indique est une sorte de base de données centralisée dans laquelle les nœuds peuvent stocker de l'information et ainsi partager des paramètres globaux.

Les topics : L'échange de l'information s'effectue soit de manière asynchrone via un topic ou de manière synchrone via un service.

Un topic est un système de transport de l'information basé sur le système de l'abonnement / publication (subscribe / publish). Un ou plusieurs nœuds pourront publier de l'information sur un topic et un ou plusieurs nœuds pourront lire l'information sur ce topic. Le topic est en quelque sorte un bus d'information asynchrone un peu comme un flux RSS. Cette notion de bus many-to-many asynchrone est essentielle dans le cas d'un système distribué.

Le topic est typé, c'est-à-dire que le type d'information qui est publiée (le message) est toujours structuré de la même manière. Les nœuds envoient ou reçoivent des messages sur des topics.

Les messages : Un message est une structure de donnée composite. Un message est composé d'une combinaison de types primitifs (chaines de caractères, booléens, entiers, flottants…) et de message (le message est une structure récursive). Par exemple un nœud représentant un servomoteur du robot, publiera certainement son état sur un topic (selon ce que vous aurez programmé) avec un message contenant par exemple un entier représentant la position du moteur, un flottant représentant sa température, un autre flottant représentant sa vitesse…

Les services : Le service permet une communication synchrone entre deux nœuds. La description des services est stockée dans nom_package/srv/monServiceType.srv. Ce fichier décrit les structures de données des requêtes et des réponses. La figure suivante, illustre le principe de fonctionnement des services et des topics.

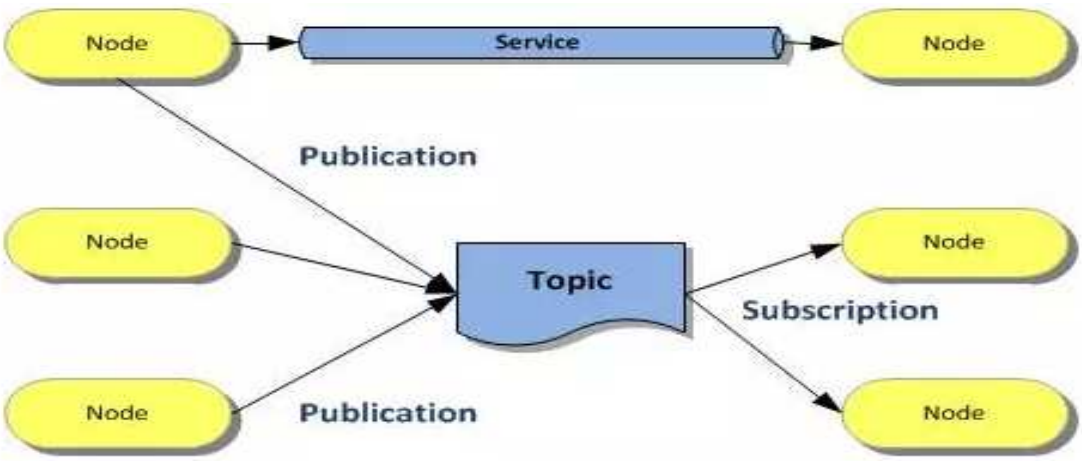


Figure 0.17. Principe de fonctionnement des topics et des services dans ROS.

Les bags : Les « bags » sont des formats pour stocker et rejouer les messages échangés. Ce système permet de collecter par exemple les données mesurées par les capteurs et les rejouer ensuite autant de fois qu'on le souhaite afin de faire de la simulation sur des données réelles. Ce système est également très utile pour déboguer un système a posteriori.

II.3.1.6. L'organisation des ressources dans ROS

Les ressources de ROS sont organisées dans une structure hiérarchique sur disque. Quatre concepts importants se détachent :

- Le package.
- Le stack.
- Le repository.
- L'univers.

Reprenons chacun de ces concepts :

Le package : Un package est un répertoire qui contient les nœuds, les librairies externes, des données, des fichiers de configuration et un fichier de configuration xml nommé manifest.xml.

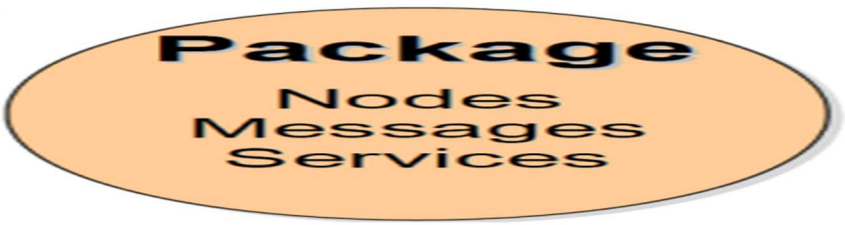


Figure 0II.18. Le package.

Le stack : Le stack est une collection de packages. Elle propose une agrégation de fonctionnalités telles que la navigation, la localisation… Une stack est un répertoire qui contient les répertoires des packages ainsi qu'un fichier de configuration nommé stack.xml.

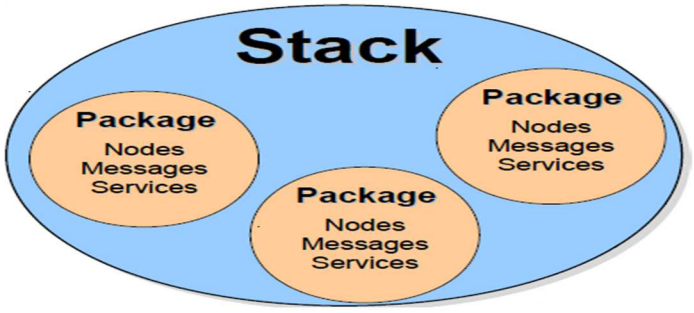


Figure 0.19. Le stack.

Le repository : En plus de ces deux notions très importantes, on relève également la notion de distribution. Une Distribution, comme dans Linux, est un ensemble de stacks versionnées.

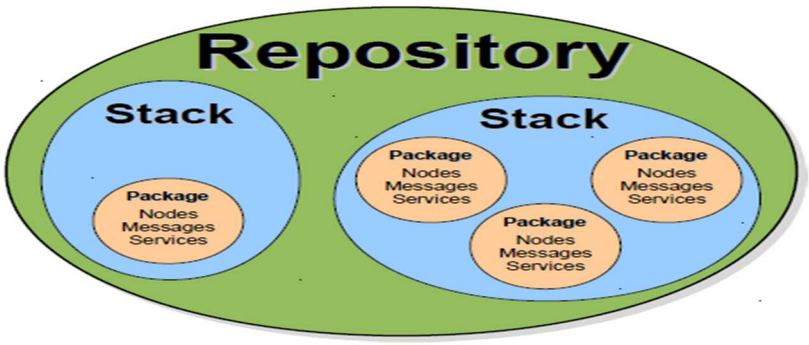


Figure 0.20. Le repository.

L'univers : C'est trois structures sont les structures fondamentale pour créer un univers ROS

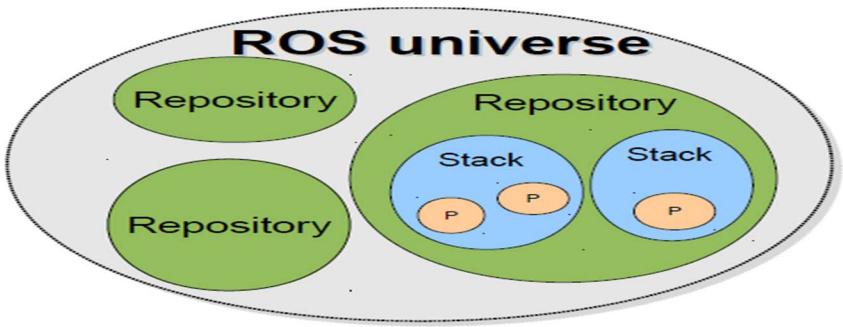


Figure 0.21. L'univers de ROS.

II.3.1.6. L'URDF

L'URDF (Unified Robot Description Format) est une contribution intéressante de ROS dans la robotique. Il s'agit un format XML permettant de décrire sous la forme d'un fichier standardisé un robot complet. Le robot ainsi décrit peut être statique ou dynamique et des propriétés physiques et de collision peuvent y être ajoutées.

L'URDF est utilisé par exemple par le simulateur Gazebo pour représenter le robot.

II.3.1.7. Les outils utilisés

Comme nous l'avons dit plus haut, ROS est une collection d'outils et d'algorithmes. Certains sont très utilisés lors de la programmation, de la simulation ou de l'exécution des comportements des robots. Citons quelques outils ou algorithmes que le programmeur de ROS retrouvera souvent :

- Gazebo : un simulateur 3D.
- Rviz : un système de visualisation 3D.
- Le package « tf » qui permet de manipuler des coordonnées et des transformations.
- Opencv : qui permet le traitement d'images.
- PointCloudLibrary : reconstruction d'environnement 3D à partir de mesures d'un laser.

II.3.2. Le simulateur Gazebo

La simulation des robots est essentielle dans chaque boîte à outils robotique. Un simulateur bien conçu permet de tester rapidement des algorithmes pour des robots de conception et d'effectuer les tests de régression en utilisant des scénarios réalistes [26].

C'est la raison pour la quel nous avons choisi Gazebo, car il offre la possibilité de simuler avec précision et efficacité les différents robots existent dans des environnements internes et externes complexes. C'est un outil de base pour la création des modèles physiques et des environnements de simulation avec des graphismes de haute qualité.

Gazebo est libre et OpenSource avec une communauté dynamique. Il est utilisé comme un nœud intégré au ROS.

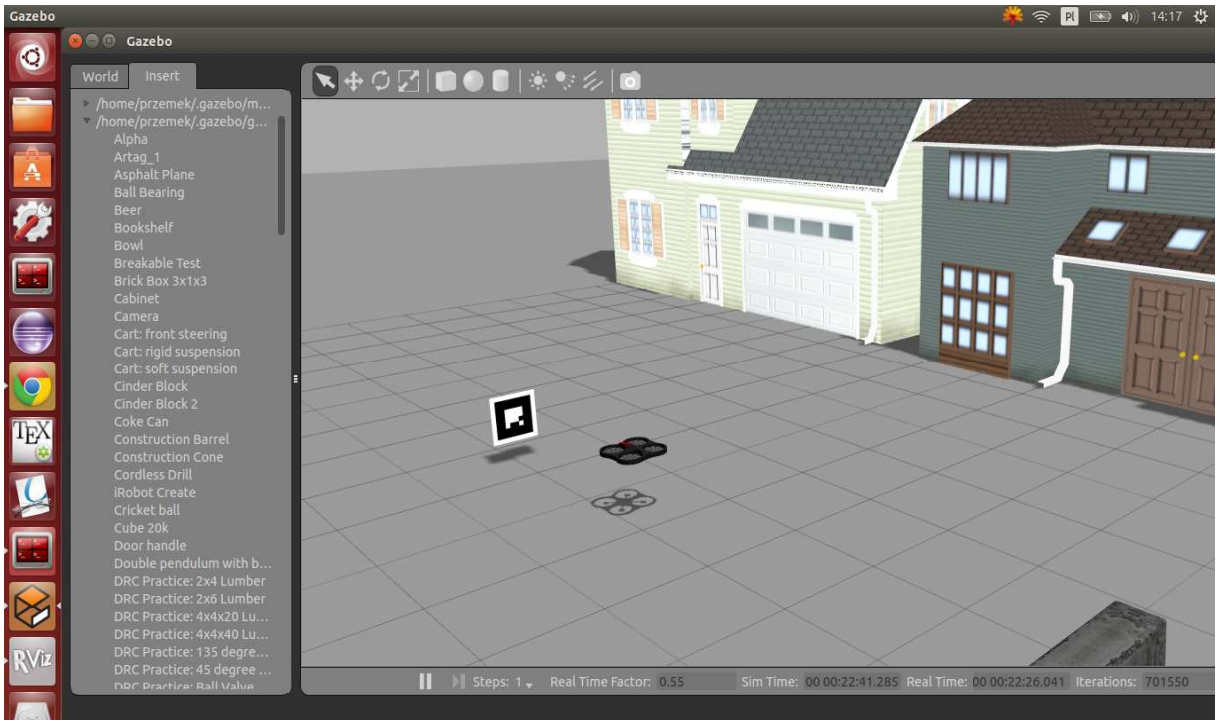


Figure II.22. L'environnement de simulation Gazebo.

II.3.2.1. Définition

Gazebo est un simulateur utilisé pour créer des applications embarquées pour un robot sans dépendre physiquement de la machine réelle. Il permet la modélisation 3D d'un robot ainsi que son environnement. Il comporte un simulateur capable d'émuler le mouvement d'un robot réel dans un espace de travail réel.

II.3.2.2. Objectif

Gazebo est capable de créer un scénario 3D sur un ordinateur avec des robots, des obstacles et de nombreux autres objets. Gazebo utilise également un moteur physique pour l'éclairage, la gravité, inertie, etc. des tests peuvent être évalué pour des robots dans des scénarios difficiles ou dangereux sans aucun dommage pour les robots réels. La plupart du temps il est plus rapide pour exécuter un simulateur au lieu de commencer le scénario entier sur un robot réel [27].

Initialement Gazebo a été conçu pour évaluer les algorithmes pour les robots.

II.3.2.3. Fonctionnement

La simulation par gazebo se compose de plusieurs éléments qui travaillent ensemble pour simuler un morceau de matériel. En général, il y a trois éléments. Le format de description de robot (dans ce cas son URDF) L'environnement de simulation (Gazebo) Le contrôleur (s) (Les nœuds ROS)

L'image suivante décrit comment ces éléments coopèrent pour fournir des services ROS avec laquelle le robot simulé peut être contrôlée. Le SDF Décrit les robots des éléments visuels et physiques, belvédère, qui crée alors des services pour le robot décrit. Ces services sont accessibles par des nœuds ROS utilisant des appels de service. Le nœud de contrôleur utilise ces services pour créer de nouveaux appels de services avec lesquels le robot peut être contrôlé.

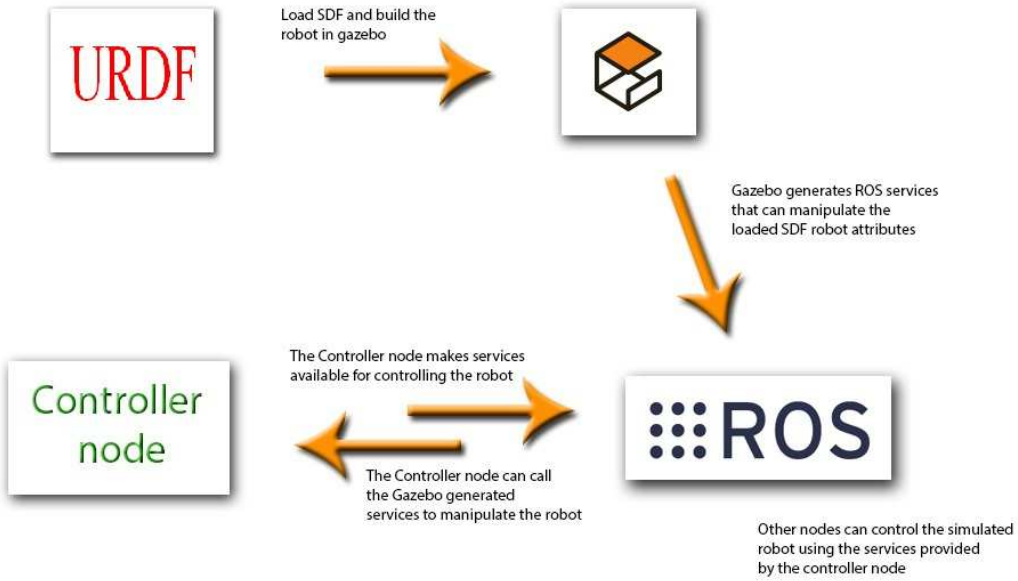


Figure II .23. Principe de fonctionnement de Gazebo.

II.3.3. Fuzzylite

Fuzzylite est une bibliothèque libre et open-source de contrôle par logique floue, programmé en C ++ pour de multiples plates-formes (Windows, Linux, Mac, iOS, Android). Son objectif est de faciliter la création des contrôleurs floues en quelques étapes en utilisant la programmation orientée objet sans nécessiter de bibliothèques tierces [28].

Pour ce projet QtFuzzyLite (figure II.24) est utilisé. C'est l'interface graphique de Fuzzylite, basée sur Qt. Son objectif est de permettre de concevoir visuellement des contrôleurs flous et d'interagir avec eux en temps réel.

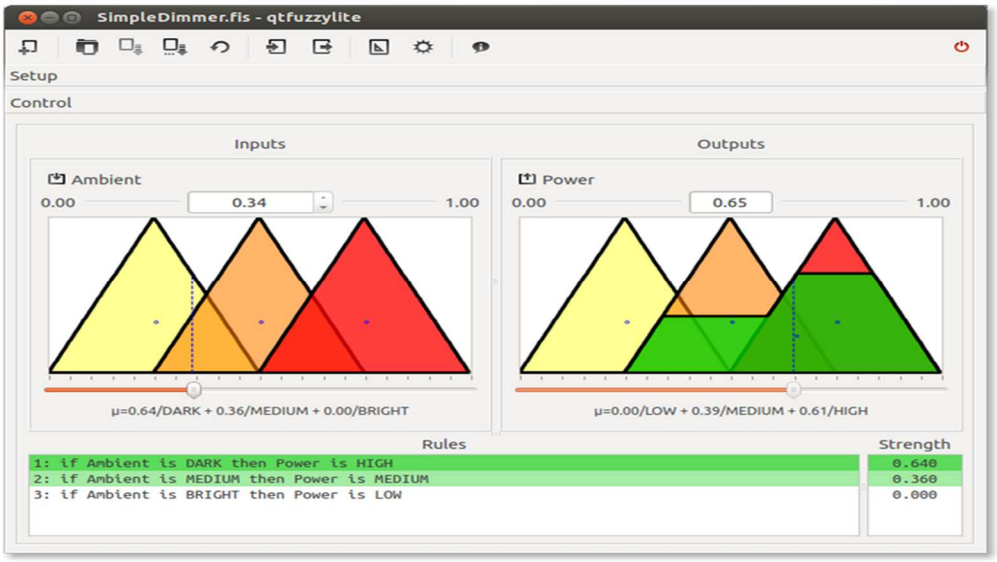


Figure II.24. L'interface graphique Qtfuzzylite.

II.3.3.1. Caractéristiques

Les bibliothèques FuzzyLite présentent les fonctionnalités suivantes :

- ✓ (6) contrôleurs : Mamdani, Takagi-Sugeno, Larsen, Tsukamoto, Tsukamoto inversé, hybrides
- ✓ (21) Termes linguistiques : (4) Base : triangle, trapézoïde, rectangle, discret.
- ✓ (9) Étendues : cloche, cosinus, gaussien, produit gaussien, forme de pi, différence sigmoïde, produit sigmoïde, pointe.
- ✓ (5) Bords : binaire, concave, rampe, sigmoïde, en forme de s, en forme de z.
- ✓ (3) Fonctions : constante, linéaire, fonction.
- ✓ (7) Méthodes d'activation : général, proportionnel, seuil, premier, dernier, plus bas, plus haut.
- ✓ (8) Conjonction et implication (normes T) : minimum, produit algébrique, différence bornée, produit drastique, produit einstein, produit Hamacher, minimum nilpotent, fonction.
- ✓ (10) Disjonction et agrégation (Normes S) : maximum, somme algébrique, somme bornée, somme drastique, somme einstein, somme hamacher, maximum nilpotent, somme normalisée, somme non bornée, fonction.

- ✓ (7) Défuzzificateurs: (5) Intégral: centroïde, bissectrice, maximum du maximum, maximum du maximum, moyenne du maximum.
- ✓ (2) Pondéré : moyenne pondérée, somme pondérée.
- ✓ (7) Couvertures : aucune, extrêmement, rarement, quelque peu, très, fonctionne.
- ✓ (3) Importateurs: FuzzyLite Language fll, Fuzzy Inference System fis, Fuzzy Control Language fcl.
- ✓ (7) Exportateurs : C ++, Java, langage FuzzyLite fll, ensemble de données FuzzyLite, script R, logiciel de système d'inférence flou fis, langage de contrôle fuzzy fcl.
- ✓ (30+) Exemples de contrôleurs Mamdani, Takagi-Sugeno, Tsukamoto et hybrides de fuzzylite, Octave et Matlab, chacun inclus dans les formats suivants : C ++, Java, fll, fld, R, fis et fcl.

II.3.3.2. L'interface graphique QtFuzzylite

QtFuzzyLite est (très probablement) la meilleure application disponible pour concevoir et exploiter directement des contrôleurs de logique floue en temps réel. fuzzylite est une bibliothèque de contrôle de logique floue gratuite et à code source ouvert programmée en C ++ pour plusieurs plates-formes (Windows, Linux, Mac, iOS, par exemple). jfuzzylite est la bibliothèque fuzzylite équivalente pour les plateformes Java et Android. La figure II illustre un exemple de l'interface graphique de QtFuzzylite.

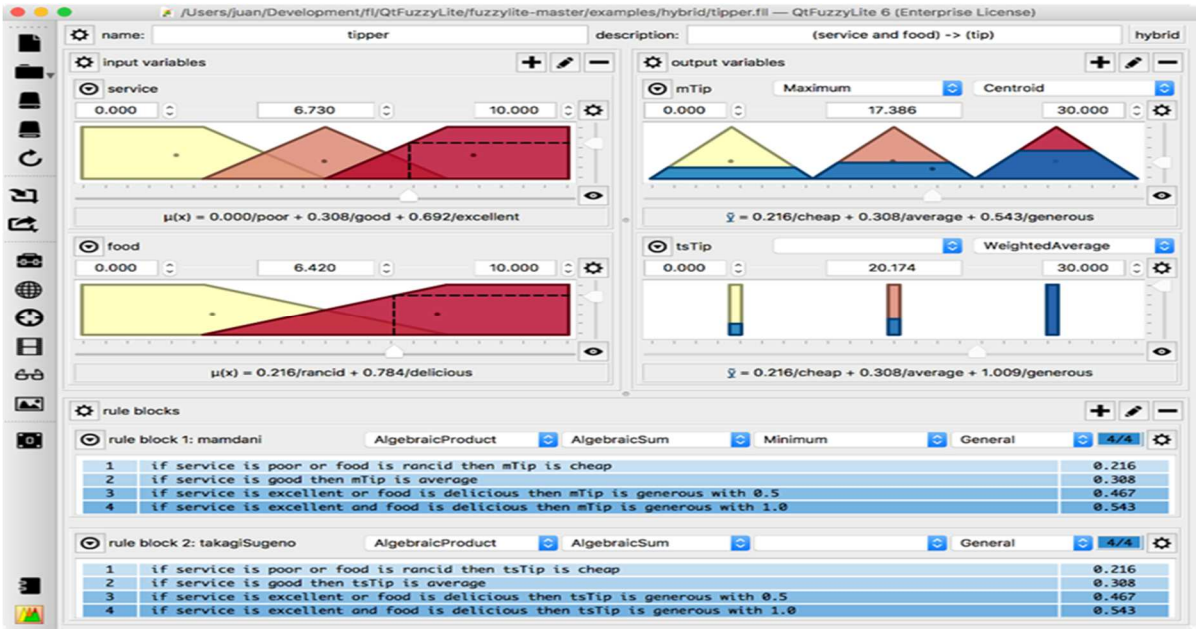


Figure II.25. L'interface graphique QtFuzzylite.

De plus, vous pouvez facilement :

- Créez vos propres classes héritant de fuzzylite, enregistrez-les dans les usines et intégrez-les pour fonctionner dans fuzzylite.
- Utilisez plusieurs blocs de règles dans un même moteur, chacun contenant un nombre quelconque de règles (éventuellement pondérées) et différents opérateurs de conjonction, de disjonction et d'activation.

- Écrivez les règles d'inférence simplement naturellement, par exemple, "si un obstacle est laissé, alors l'orientation est correcte".
- Renvoie une valeur de sortie par défaut, verrouille les valeurs de sortie dans des plages spécifiques, verrouille la valeur de sortie valide précédente lorsqu'aucune règle n'est activée.
- Explorez l'espace de fonctions de votre contrôleur.
- Utilisez la bibliothèque entière sur plusieurs threads car elle est sécurisée pour les threads.

II.3.4. MATLAB et AR Drone Simulink Development Kit (ARDSDK)

MATLAB est une plate-forme de langage de programmation de haut niveau pouvant effectuer des tâches de calcul. C'est une interface pratique pour la mise en œuvre d'opérations de contrôle en temps réel. Matlab sera utilisé dans le chapitre 5 pour développer des algorithmes permettant de naviguer de manière autonome. MATLAB est une application largement utilisée par les ingénieurs et les scientifiques pour l'informatique et le domaine technique.

Le kit de développement AR Drone Simulink (ARDSDK) a été développé en tant que projet de stage de recherche d'été par Pieter J. Mosterman et David Escobar Sanabria, au cours de l'année 2013. Ce travail a été soumis sur le site officiel de Mathworks et peut être téléchargé ici [29]. Le kit comprend des blocs pour la simulation ainsi que le contrôle Wi-Fi en temps réel des plates-formes AR.Drone. Le kit facilite la navigation dans AR.Drone en envoyant des commandes de contrôle et en récupérant les états du drone en temps réel.

Cette boîte à outils contient une bibliothèque Simulink et un ensemble de 5 blocs (Figure II.26), qui pour son bon fonctionnement, ils nécessitent une version supérieure à la 2013a de Real Time Windows Target et évidemment un drone AR 2.0.

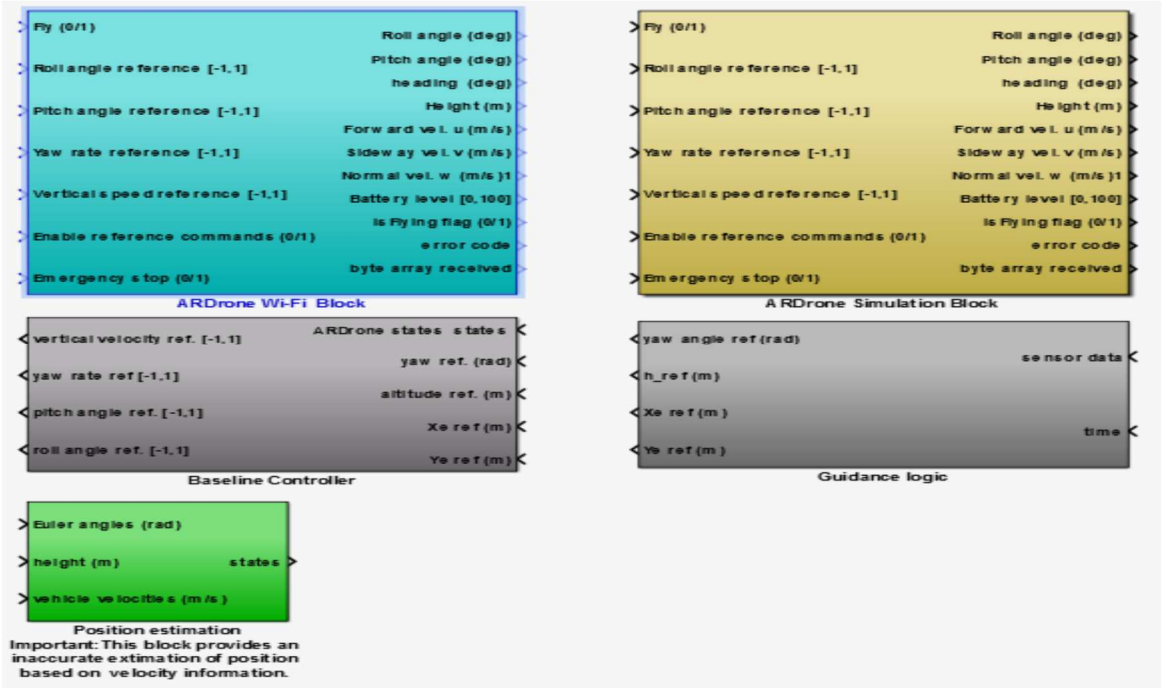


Figure II.26. Bloc de simulation.

Au sein de ces blocs, le bloc Wi-Fi AR Drone est le plus important en ce qui concerne cette thèse Ces blocs ont les mêmes variables pour leur entrée et leur sortie, ce qui aide au moment de sa connexion et permet la transition correcte des données entre les blocs [30].

II.3.4.1. Position estimation

Ce bloc consiste à estimer la position dans laquelle se trouve l'AR. Drone, c.-à-d. la hauteur, les angles d'orientation et les vitesses.

II.3.4.2. Guidance logic

Au niveau de ce bloc les références sont ajoutées et les décisions de validation de ces points sont prises afin que l'AR. Drone suit la trajectoire imposée.

II.3.4.3. Baseline controller

Ce bloc contient un dispositif de la commande qu'on désire appliquer sur notre système afin de contrôler l'AR. Drone, cette commande classique est basée sur un contrôleur proportionnel et la dynamique de l'erreur de sortie, elle n'a aucune exigence sur les paramètres de système à commander.

II.4. CONCLUSION

Dans ce chapitre nous avons, abordés la présentation des différentes parties du quadrotor Parrot AR. Drone. Une présentation succincte de tous les éléments regroupant les différentes parties sont détaillées ainsi que leur domaine d'utilisation. Par la suite nous avons donné une description du système d'exploitation ROS qui est chargé de la partie développement. Comme nous avons donné un aperçu sur le simulateur Gazebo. Par la suite nous avons abordés le logiciel Fuzzylite spécialisé dans la conception des contrôleurs flous, et en derniers nous avons fait une brève présentation du lien entre le Drone de Parrot AR.Drone et Matlab/Simulink.

Dans le chapitre trois, nous détaillerons les étapes suivi pour la conception des contrôleurs flous qui seront utilisés pour le suivi de cible.

Chapitre III :

Identification et suivi de Cible à Base d'un Contrôleur Flou

III.1. INTRODUCTION

Dans ce chapitre, nous introduisons les bases de la logique floue ainsi que son principe de fonctionnement, par la suite nous décrierons le processus de conception du contrôleur flou utilisé par le quadrotor pour le suivi de cible et l'évitement de l'obstacle. En dernier nous présenterons les résultats de simulation obtenus pour différents positions du quadrotor par rapport à la cible pour l'identification et le suivi, aussi les résultats de simulation pour l'évitement d'obstacle en utilisant un contrôleur flou seront présentés. Par la suite le contrôleur développé sera implanté sur une plateforme réelle qui est le quadrotor AR.Drone 2.0 pour suivi de cible. Les résultats obtenus seront discutés et interprétés.

III.2. LA LOGIQUE FLOUE

La logique floue fait partie de l'intelligence artificielle et aussi est une extension de la logique booléenne créée par Lotfi Zadeh en 1965 en se basant sur sa théorie mathématique des ensembles flous, qui est une généralisation de la théorie des ensembles classiques. En introduisant la notion de degré dans la vérification d'une condition, permettant ainsi à une condition d'être dans un autre état que vrai ou faux. La logique floue confère une flexibilité très appréciable aux raisonnements qui l'utilisent, ce qui rend possible la prise en compte des imprécisions et des incertitudes [31-34].

Un des intérêts de la logique floue pour formaliser le raisonnement humain est que les règles sont énoncées en langage naturel.

III.2.1 Historique

En 1978 ce fut la première application industrielle de la logique floue par la société danoise F.L.Smidth qui réalisa le contrôle d'un four à ciment. En 1965, bien avant cette application, la logique floue a été proposée pour la première fois par le professeur Lotfi Zadeh à l'université de Berkeley aux USA, en se basant sur sa théorie mathématique des ensembles flous, qui est une généralisation de la théorie des ensembles classiques. En 1975 le professeur Mamdani de l'université de Londres développe une stratégie pour le contrôle des procédés, ses travaux ont été repris et améliorés.

Les premiers systèmes flous ont été initialement implémentés au Japon, au début des années 80, où ils ont connu une expansion rapide en milieu industriel. Depuis, la logique floue est largement utilisée par les ingénieurs, ses applications ne cessent d'augmenter [35 - 37].

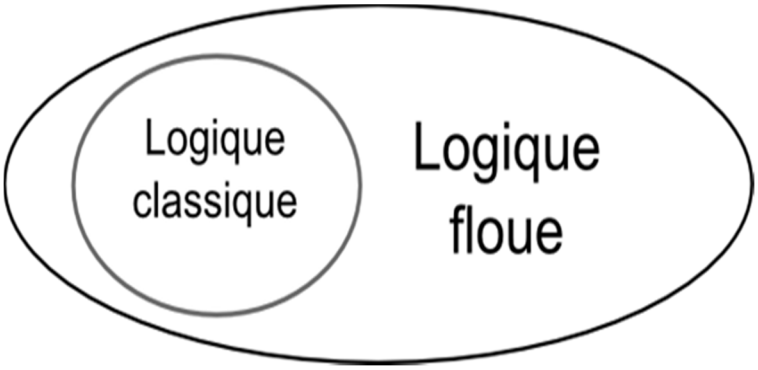


Figure III.1.Ensembles flous.

III.2.2. Avantages et inconvénients de la logique floue

Ses avantages viennent notamment de ses capacités à :

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

Et ses inconvénients :

- Le manque de directives précises pour la conception d'un réglage : choix des grandeurs à mesurer, détermination de la fuzzification, des inférences et de la défuzzification.
- L'approche artisanale et non systématique : l'implémentation des connaissances de l'opérateur est souvent difficile.
- L'absence d'un modèle valable, on est confronté à l'impossibilité de la démonstration de la stabilité du circuit de réglage en toute généralité.
- La précision de réglage souvent peu élevée.
- La cohérence des inférences non garantie a priori : apparition de règles d'inférence contradictoires possible.
- Il n'existe pas de théorie générale qui caractérise rigoureusement la stabilité et la robustesse.

III.2.3 Principes de la logique floue

Logique floue : « logique qui substitue à la logique binaire une logique fondée sur des variables pouvant prendre, outre les valeurs « vrai » ou « faux », les valeurs intermédiaires « vrai » ou « faux » avec une certaine « probabilité ».

III.2.3.1. Les règles floues

L'outil le plus utilisé dans les applications de la logique floue est « les règles floues ». Ces règles floues peuvent être composées en parallèle, mais peuvent également être enchainées dans certaines applications.

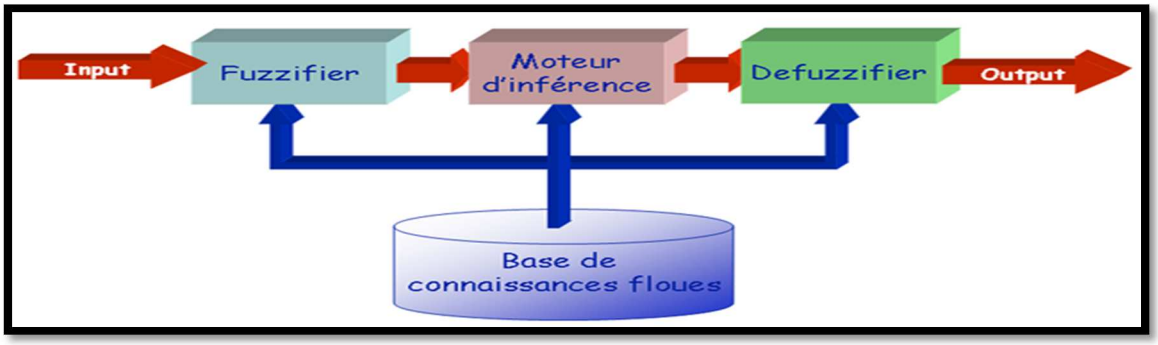


Figure III.2. La structure de base d'une commande floue.

III.2.3.2.La fuzzification

La fuzzification est l'étape qui consiste en la quantification floue des valeurs réelles d'une variable. La fuzzification des variables est une phase délicate du processus mis en œuvre par la logique floue. Elle est souvent réalisée de manière itérative et requiert de l'expérience.

III.2.3.3. L'inférence

Opération logique par laquelle on admet une proposition en vertu de sa liaison avec d'autres propositions tenues pour vraies.

III.2.3.4. La défuzzification

En commande floue, la défuzzification par la méthode du centre de gravité est presque toujours utilisée. Elle prend en compte l'influence de l'ensemble des valeurs proposées par la solution floue.

III.2.3.5. Le moteur d'inférence

L'inférence floue est le processus de formulation de la relation entre les entrées et les sorties par logique floue. Cette relation offre une base avec laquelle la décision est prise par le système flou. L'inférence floue fait appel alors aux concepts expliqués dans les sections précédents, à savoir : les fonctions d'appartenance, les opérateurs flous et les règles floues. Nous distinguons une variété importante d'inférences floues, mais nous nous contenterons d'en présenter quatre types : L'inférence de Max-min (Mamdani), Max-prod, Som-prod et de Sugeno [38 - 40].

III.3. CONCEPTION DU CONTROLEUR FLOU POUR SUIVI

La figure (III.3) présente un exemple qui illustre une classe générale des problèmes qu'un drone est censé résoudre pour suivre une cible. Dans cette figure, le drone doit détecter toutes les cibles afin d'identifier la cible à suivre, une fois l'objet identifié le drone va se déplacer dans sa direction jusqu'à ce qu'il arrive en face où il doit maintenir une distance de sécurité imposée par le développeur.

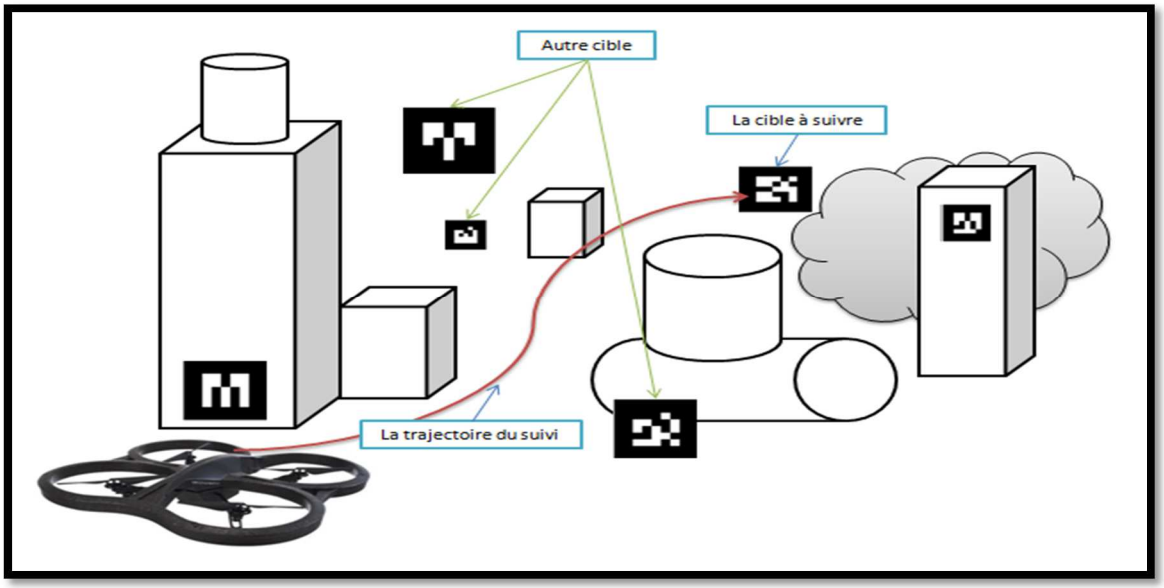


Figure III.3 : Exemple de suivi de cible par le quadrotor.

III.3.1. Les taches du contrôleur flou pour le suivi

Le contrôleur de suivi a pour but de commander le quadrotor afin de maintenir un angle d'orientation minimal lors de la translation sur son axe X vers un marqueur visuel. Pour ce faire, le quadrotor doit se translater à travers les axes X et Y pour réduire l'erreur de position, tourner le long de l'axe Z pour réduire l'erreur d'orientation et en translatons le long de l'axe X pour suivre la cible. La figure III.4 représente la tâche du contrôleur de suivi.

D'après la figure, nous constatons que la variable h est la hauteur de cible par rapport au sol, hs est l'altitude détectée par le quadrotor.

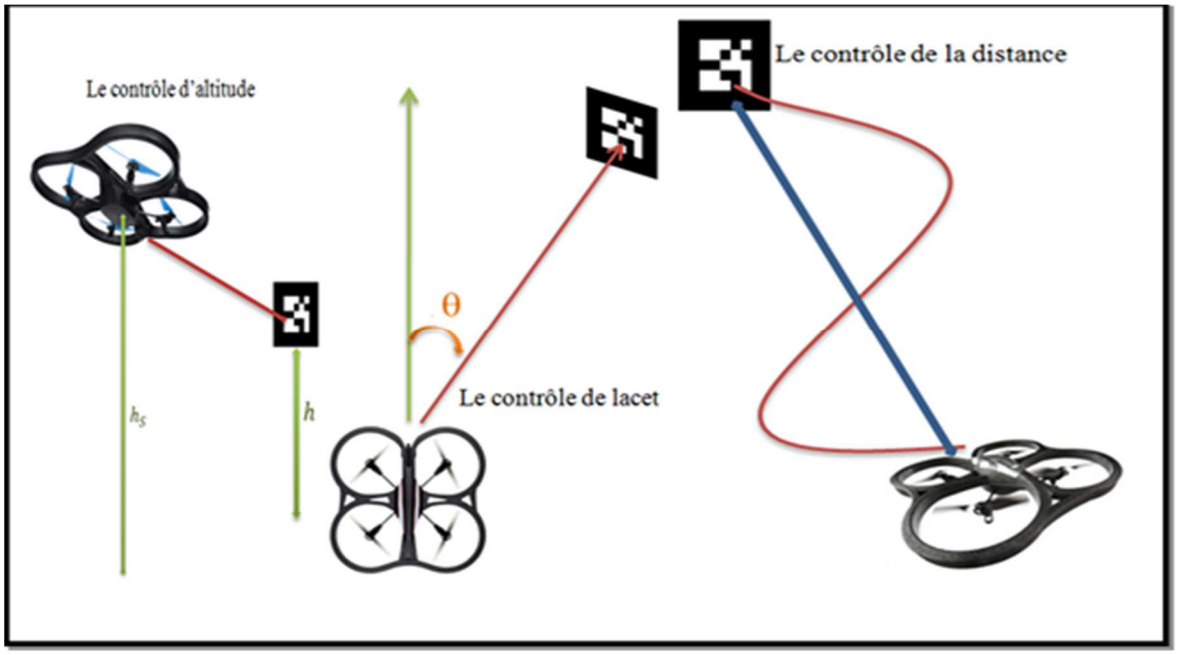


Figure III.4. Principe du contrôleur de suivi.

III.3.2. Conception du Système

L'objectif principal dans cette partie est de contrôler un véhicule aérien sans pilote (UAV) afin de suivre certains panneaux. Le panneau cible contient des marqueurs spéciaux, tels que le code QR. Dans ce travail, nous voulons examiner le problème d'identification et de suivi automatique. Nous utilisons AR.Drone comme plate-forme aérienne pour le suivi d'une cible fixe. Premièrement, nous devons garantir que le quadrotor doit reconnaître la cible correctement. Deuxièmement, nous devons laisser le quadrotor essayer d'atteindre la position voulue et passer doucement près de la cible. Ensuite, en fonction des informations obtenues par l'image de la caméra frontale de l'AR.Drone, nous pouvons calculer les coordonnées relatives par rapport à la cible et déplacer le quadrotor près de la cible.

Les contrôleurs flous développés sont implémentés sur le drone AR.Drone en utilisant une architecture de servo-dynamique telle que présentée dans la figure III.5. Dans ce schéma, les références de vitesse générées par le contrôleur sont utilisées comme références d'entrée pour le quadrotor.

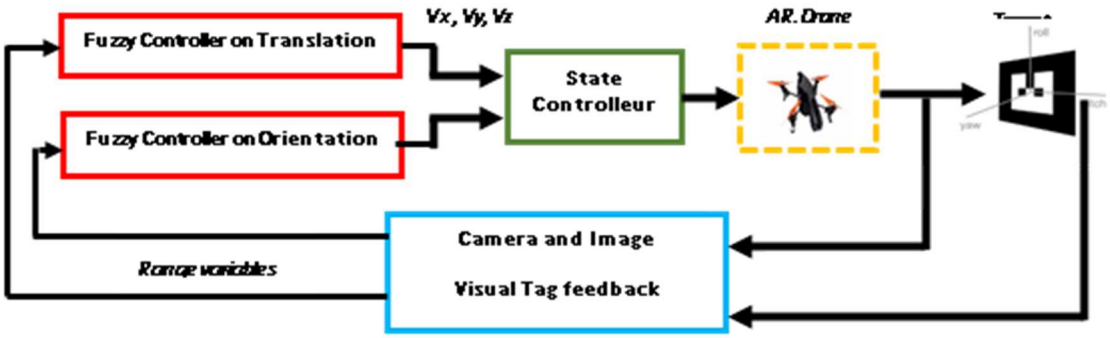


Figure III.5. Schéma fonctionnel du système de contrôle flou pour suivi de cible.

Pour développer les contrôleurs flous, nous avons utilisé la combinaison suivante : ROS (Robot Operating System), Gazebo avec le simulateur Tum_ardrone [41], le package ROS ar_track_alvar et les outils de FuzzyLite. ROS est un logiciel qui permet le transport de données de capteurs et de contrôles via des "Topics" à l'aide des modèles de transmission de messages éditeur / abonné. Le simulateur Tum_ardrone est un package qui permet une simulation de l'AR.Drone dans des environnements 3D. FuzzyLite est une bibliothèque Fuzzy Logic Controller open source, écrite en C ++ et dotée d'une interface graphique appelée QtFuzzyLite pour la conception des règles floues. Le package ROS ar_track_alvar est utilisé pour l'identification des marqueurs.

Le flux de travail du contrôleur flou est structuré de la façon suivante :

Les premières fonctions d'appartenance et / ou règles sont créées / modifiées dans l'interface graphique QtFuzzyLite. Ensuite, le code C ++ du contrôleur flou est exporté à l'aide de l'interface graphique. Le code généré est inséré dans le nœud du contrôleur dans ROS et compilé. Ensuite, le nœud du contrôleur est connecté au simulateur Tum_ardrone. Par la suite il est évalué en l'implémentant sur l'AR.Drone. Si le test montre des résultats inacceptables, le processus est redémarré avec les modifications apportées aux fonctions d'appartenance. Si les tests sont satisfaisants, l'AR.Drone est testé avec le contrôleur.

Le matériel utilisé dans cette partie de la thèse est :
- AR.Drone 2.0
- Ordinateur portable avec Ubuntu 12.04 et ROS Fuerte

Le logiciel utilisé dans cette partie de la thèse est :
- ROS Fuerte
- Ubuntu12.04
- Paquet ROS de balise AR ar_track_alvar
- Package ROS Ardrone_autonomy Lab, wrapper ROS du pilote AR.Drone 2.0
- Package ROS Tum_ardrone.

III.3.3. Système de contrôle

La figure III.6 décrit le schéma synoptique pour suivi de cible. Si le quadrotor est correctement orienté vers la cible, il doit maintenir une distance de sécurité par rapport à la cible, c'est la principale hypothèse retenue lors de la mise en œuvre de notre contrôleur. La figure ci-dessous montre que, pour une cible fixe, l'angle par rapport à l'axe longitudinal du quadrotor diminue à mesure que la distance augmente. Lorsque la distance D tend vers l'infini, la correction de l'angle d'orientation tend vers zéro.

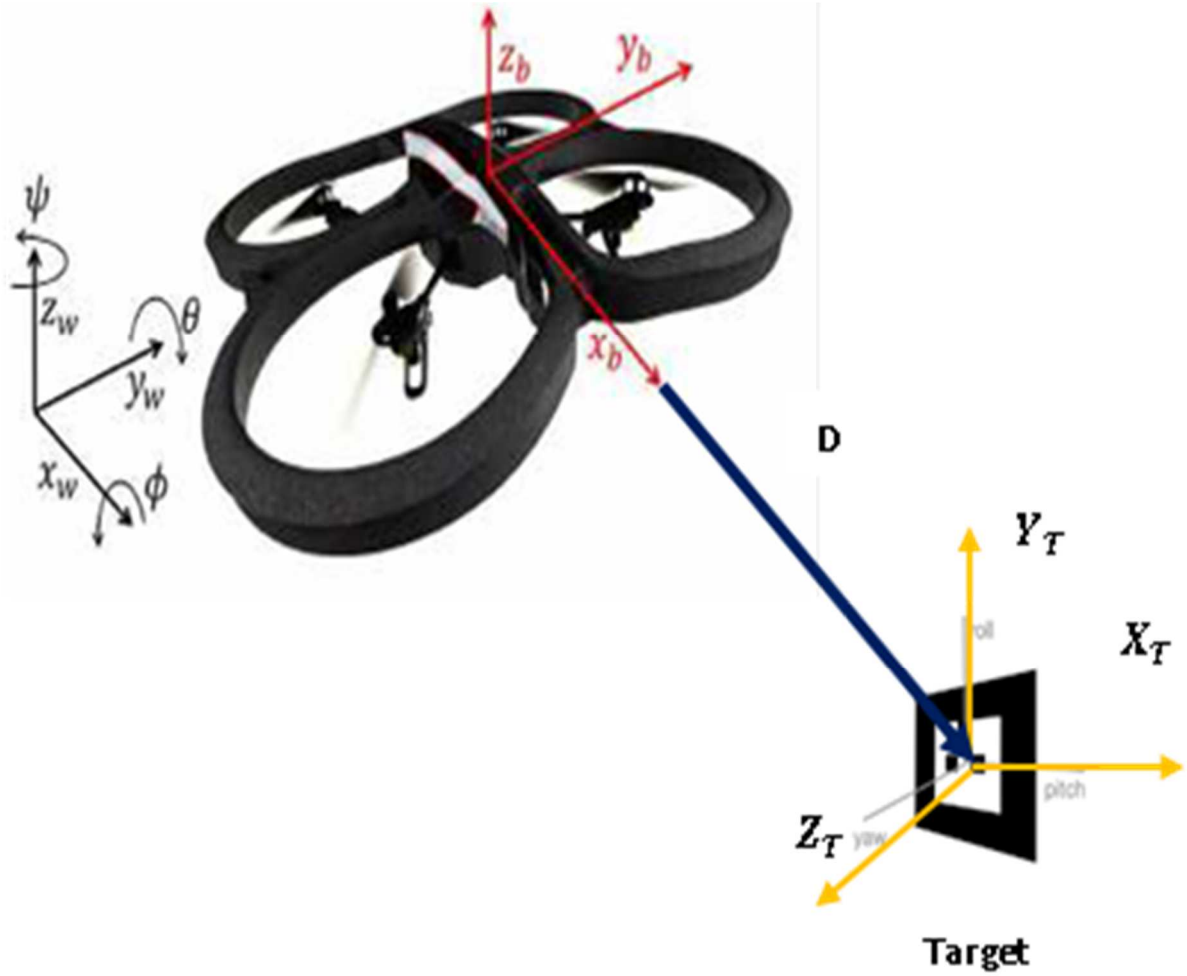


Figure III.6. Représentation des tâches de contrôle.

III.4. ETUDE DE SIMULATION

Dans cette section, nous décrivons la conception des contrôleurs flous développés et leurs simulations pour le suivi de cible. Les règles du contrôleur flou sont présentées dans cette section. La conception est testée en utilisant Gazebo comme environnement de simulation et de visualisation. Les données de vol et les informations visuelles de balises ont été utilisées comme entrées pour les contrôleurs de mouvement de rotation et de translation.

Pour le contrôle, les informations visuelles d'étiquette (balise) sont utilisées pour contrôler le mouvement de translation et de rotation du quadrotor. La figure III.7 montre la configuration des nœuds dans ROS pour effectuer le contrôle de suivi, le nœud de couleur verte est le nœud du contrôleur de suivi nommé 'fuzzy_tracker ' en tant qu'entrées, le nœud prend une capture d'image relative au centre de la cible et son descripteur d'identité (ID). Le nœud de couleur bleue Gazebo ROS a été utilisé pour simuler les flux d'images provenant de l'AR.Drone. L'ar_track_alvar (noeud de couleur rouge) est capable de traiter les images et de fournir des informations au nœud «fuzzy_tracker».

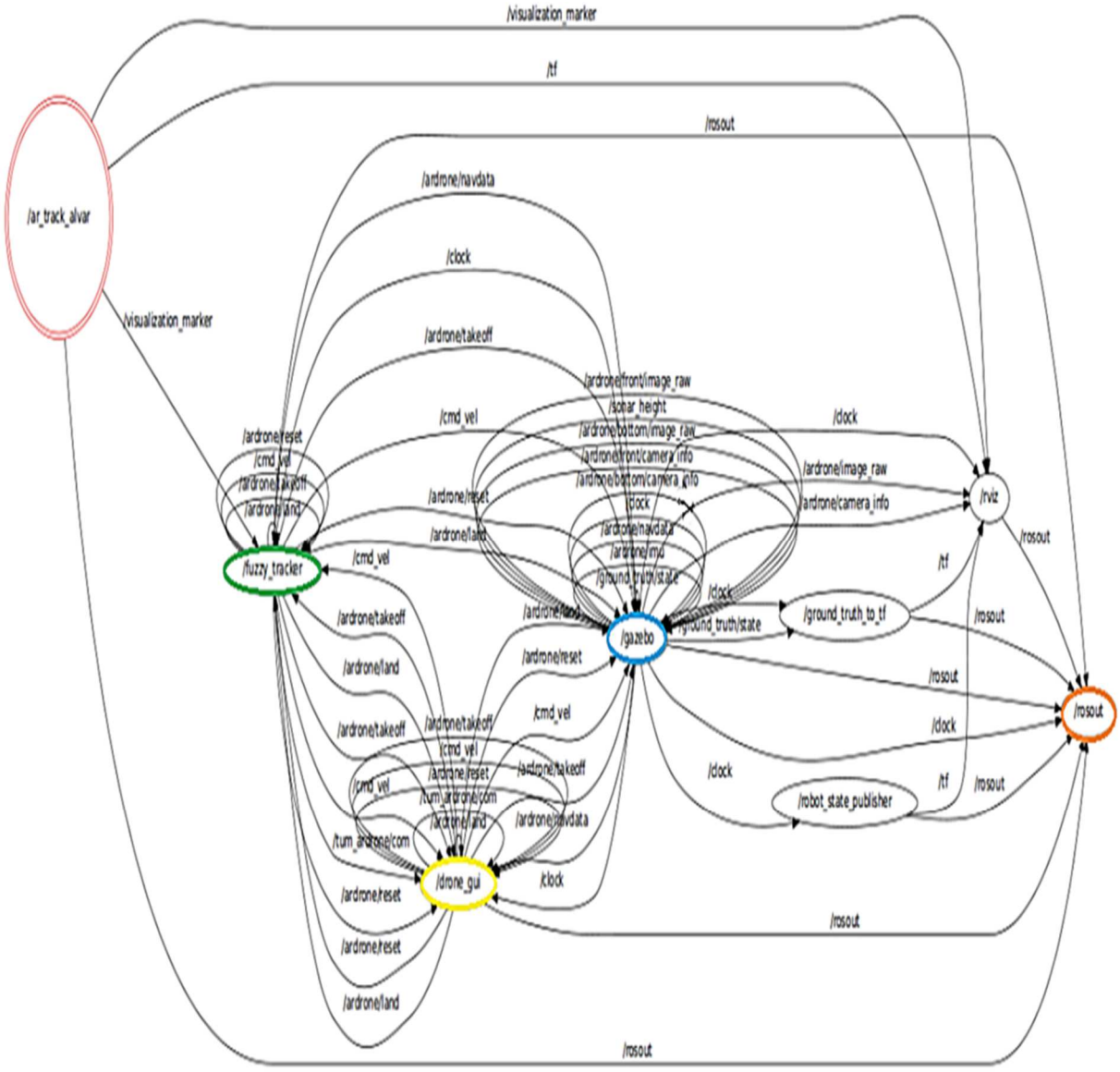


Figure III.7.La configuration des nœuds dans ROS pour la simulation.

Le flux de travail de la conception du contrôleur floue utilisant à la fois le logiciel et le matériel est présenté ci-dessous :

1. Modifier les fonctions d'appartenance et / ou les règles dans l'interface graphique FuzzyLite

2. Exportez le code C ++ du contrôleur floue à l'aide de l'interface graphique FuzzyLite

3. Insérer le code généré dans le contrôleur simulé dans ROS et le compiler

4. Exécutez le nœud ROS du contrôleur avec Tum_simulator (Gazebo).

5. Passez à l'étape 6 si le contrôleur est prêt, sinon répétez l'étape 4.

6. Exécutez le nœud ROS du contrôleur avec les pilotes Ardrone_autonomy et le matériel AR.Drone 2.0.

7. Répétez les étapes 1 à 6 si les tests matériel du contrôleur présentent un comportement indésirable.

Le contrôleur de suivi de cible est mis en œuvre avec un total de 20 règles pour contrôler le mouvement linéaire dans les trois axes et le mouvement de rotation le long de l'axe z. Chacune des règles était basée sur un raisonnement logique et sur le modèle du quadrotor.

L'algorithme de défuzzification doit être clair et le processus de détermination du signal de sortie doit être clairement identifié. De plus, la défuzzification devrait être logique, avoir un degré d'adhésion élevé. Pour la raison évoquée ci-dessus, nous avons sélectionné la méthode du centre de gravité pour les contrôleurs. Ceci peut être calculé en utilisant l'équation 2.

$$C.G = \frac{\sum_{i=1}^{n} O_i \cdot \mu_i}{\sum_{i=1}^{n} \mu_i} \quad \text{(III.1)}$$

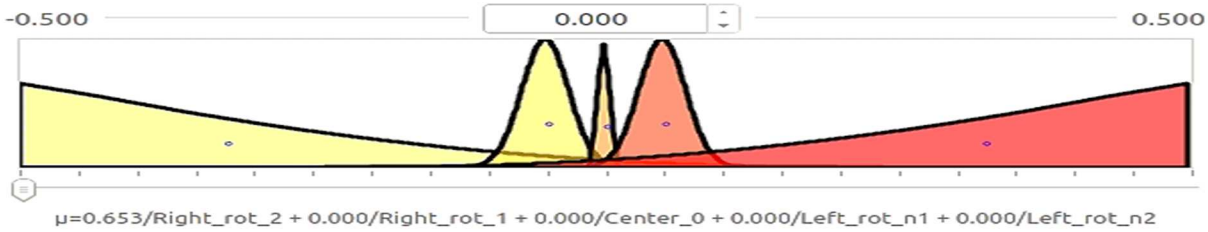
Où n représente le nombre d'éléments de la fonction d'appartenance. La valeur maximale de n est égale au nombre de règles floues. μ_i est la pondération de la règle n, O_i est la variable de sortie de l'ensemble flou. Le tableau 3 répertorie les noms des fonctions d'appartenance entrées / sorties pour la rotation autour de l'axe des z.

III.4.1. Le contrôleur de suivi de cible

Le contrôle du suivi de cible a pour but de contrôler la position du quadri-rotor pour maintenir un angle d'orientation minimum en erreur lors de la translation de quadri-rotor selon son axe X vers un marquer visuel. Pour cela, le quadrotor doit se translater selon l'axes X et l'axe Y pour, tourner le long de son axe Z et se translater le long de son axe X pour suivre la cible.

III.4.1.1. Le mouvement de rotation suivant l'axe Z

La Figure III.8 montre les cinq fonctions d'appartenances établis pour contrôler la rotation du quadrotor suivant l'axe z.



ar_rot_z (entrée)

cmd_z_rot (sortie)

Figure III.8 Le mouvement de rotation suivant l'axe Z.

Le tableau ci-dessus présente les entrées et sorties des règles floues :

Tableau III.1 Les entrées et sorties des règles floues (rotation en Z).

Entrée	Entrée MF	Sortie	Sortie MF
ar_rot_z	Right_rot_2	cmd_z_rot	rot_2
ar_rot_z	Right_rot_1	cmd_z_rot	rot_1
ar_rot_z	Center_0	cmd_z_rot	stay
ar_rot_z	Left_rot_n1	cmd_z_rot	rot_n1
ar_rot_z	Left_rot_n2	cmd_z_rot	rot_n2

- **if** ar_rot_z **is** Right_rot_2 **then** cmd_z_rot **is** rot_2
- **if** ar_rot_z **is** Right_rot_1 **then** cmd_z_rot **is** rot_1
- **if** ar_rot_z **is** Center_0 **then** cmd_z_rot **is** stay
- **if** ar_rot_z **is** Left_rot_n1 **then** cmd_z_rot **is** rot_n1
- **if** ar_rot_z **is** Left_rot_n2 **then** cmd_z_rot **is** rot_n2

Les cinq règles dans le tableau (III.1) se sont les règles qui contrôlent le mouvement de rotation suivant l'axe Z. la figure III.9 illustre le signal de commande pour la rotation suivant l'axe Z.

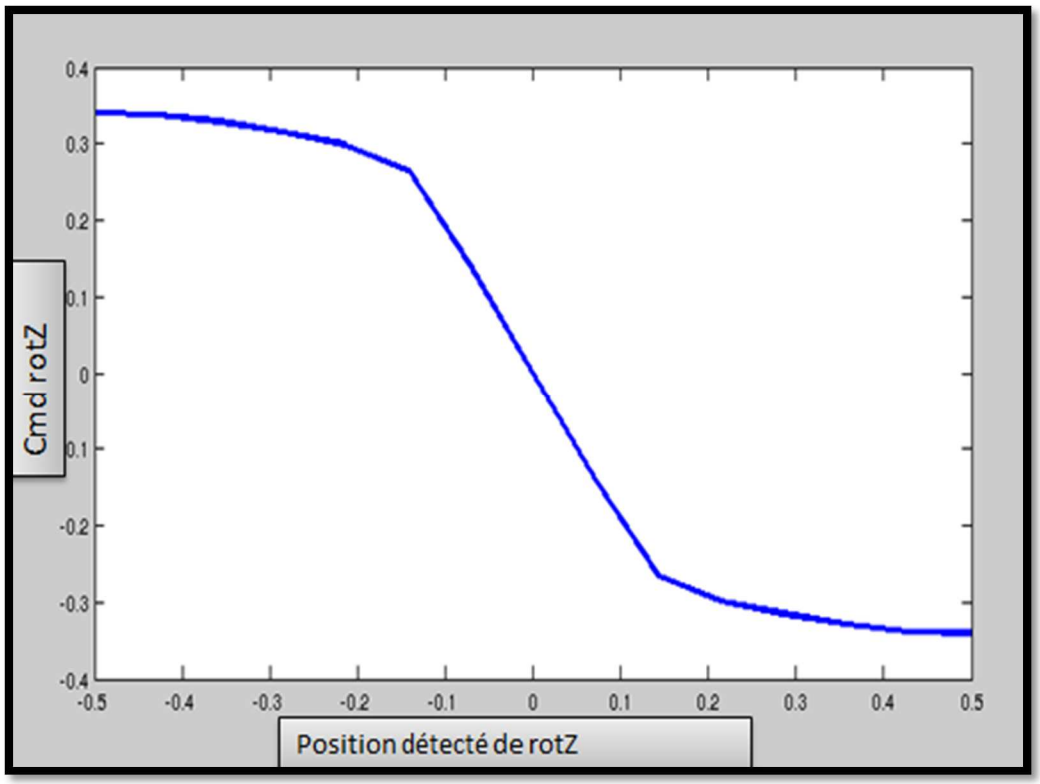
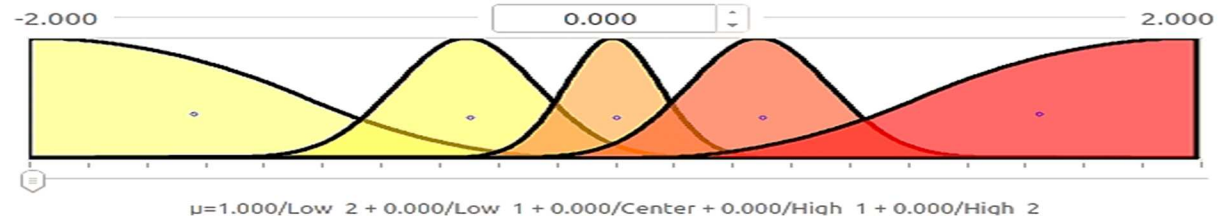


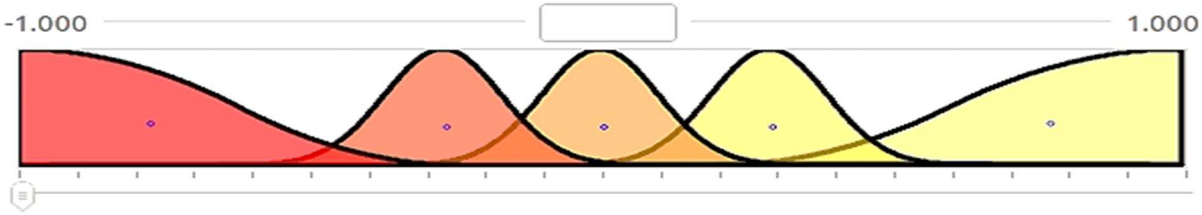
Figure III.9. Signal de commande rotZ de la régulation par logique floue de l'entrée par rapport à la sortie.

III.4.1.2. Le mouvement de translation suivant l'axe Z

La Figure III.10 montre les cinq fonctions d'appartenances établis pour contrôler la translation du quadrotor suivant l'axe z.



ar_pose_z (entrées)



cmd_z_vel (sorties)

Figure III.10. Le mouvement de translation suivant l'axe Z.

Le tableau I.2 présente les entrées et sorties des règles floues suivantes :

Tableau III.2 Les entrées et sorties des règles floues (translation en Z).

Entrée	Entrée MF	Sortie	Sortie MF
ar_pose_z	too_high	cmd_z_vel	go_lower
ar_pose_z	high	cmd_z_vel	go_down
ar_pose_z	Center	cmd_z_vel	stay
ar_pose_z	low	cmd_z_vel	go_up
ar_pose_z	too_low	cmd_z_vel	go_higher

- if ar_pose_z is High_2 then cmd_z_vel is low_2
- if ar_pose_z is High_1 then cmd_z_vel is low_1
- if ar_pose_z is Center then cmd_z_vel is stay
- if ar_pose_z is Low_1 then cmd_z_vel is high_1
- if ar_pose_z is Low_2 then cmd_z_vel is high_2

Le graphe dans la figure III.11 présente les résultats de la commande pour la translation en du quadrotor.

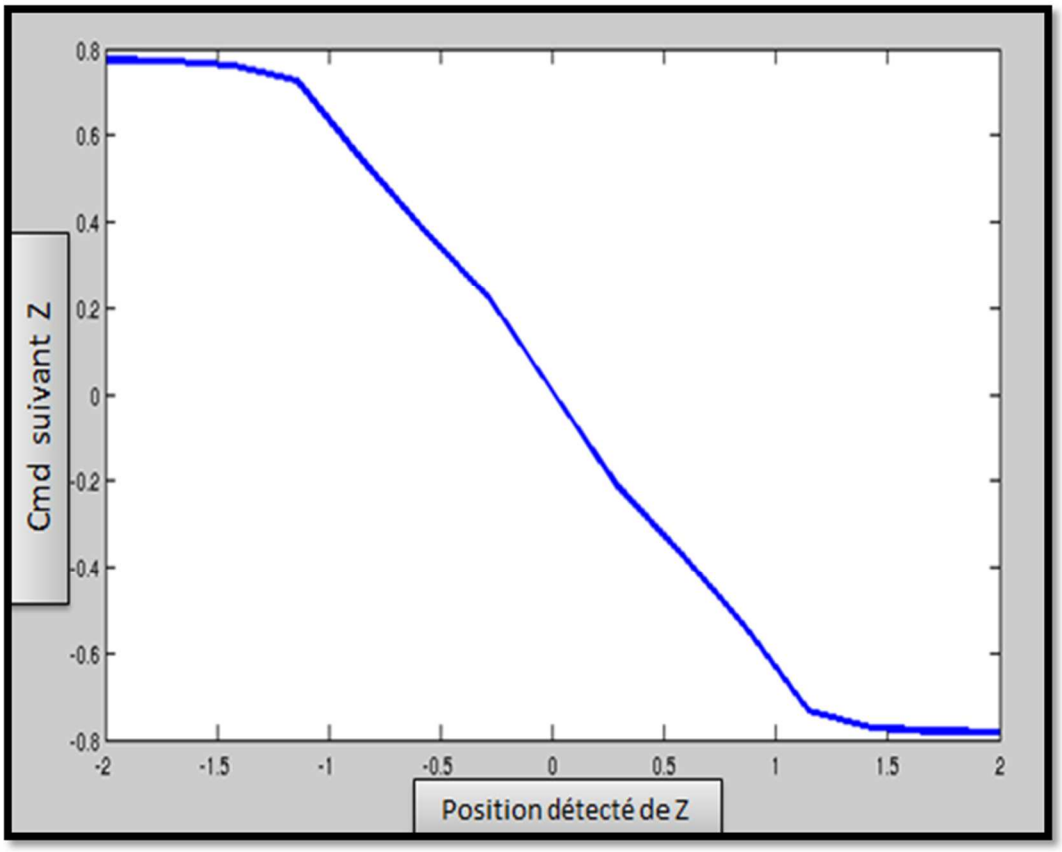


Figure III.11. Signal de commande Z de la régulation par logique floue de l'entrée par rapport à la sortie.

III.4.1.3. Le mouvement de translation suivant l'axe X

La Figure III.12 montre les quatre fonctions d'appartenances établis pour contrôler la translation du quadrrotor suivant l'axe z.

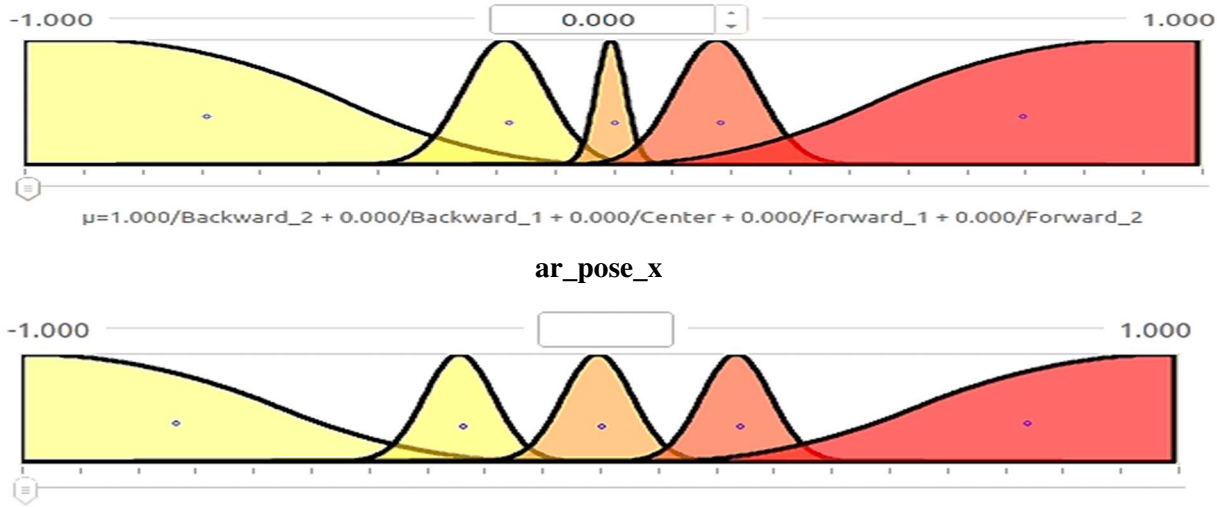


Figure III.12 Le mouvement de translation suivant l'axe X.

Le tableau ci-dessus présente les entrées et sorties des règles floues :

Tableau III.3 Les entrées et sorties des règles floues (translation en X).

Entrée	Entrée MF	Sortie	Sortie MF
ar_pose_x	Far_backward	cmd_x_vel	fast_backward
ar_pose_x	backward	cmd_x_vel	backward
ar_pose_x	Center	cmd_x_vel	stay
ar_pose_x	forward	cmd_x_vel	forward
ar_pose_x	Far_forward	cmd_x_vel	fast_forward

- **if** ar_pose_x **is** Backward_2 **then** cmd_x_vel **is** backward_2
- **if** ar_pose_x **is** Backward_1 **then** cmd_x_vel **is** backward_1
- **if** ar_pose_x **is** Center **then** cmd_x_vel **is** stay
- **if** ar_pose_x **is** Forward_1 **then** cmd_x_vel **is** forward_1
- **if** ar_pose_x **is** Forward_2 **then** cmd_x_vel **is** forward_2

La courbe dans la figure suivante illustre les résultats de la commande pour la translation selon l'axe X.

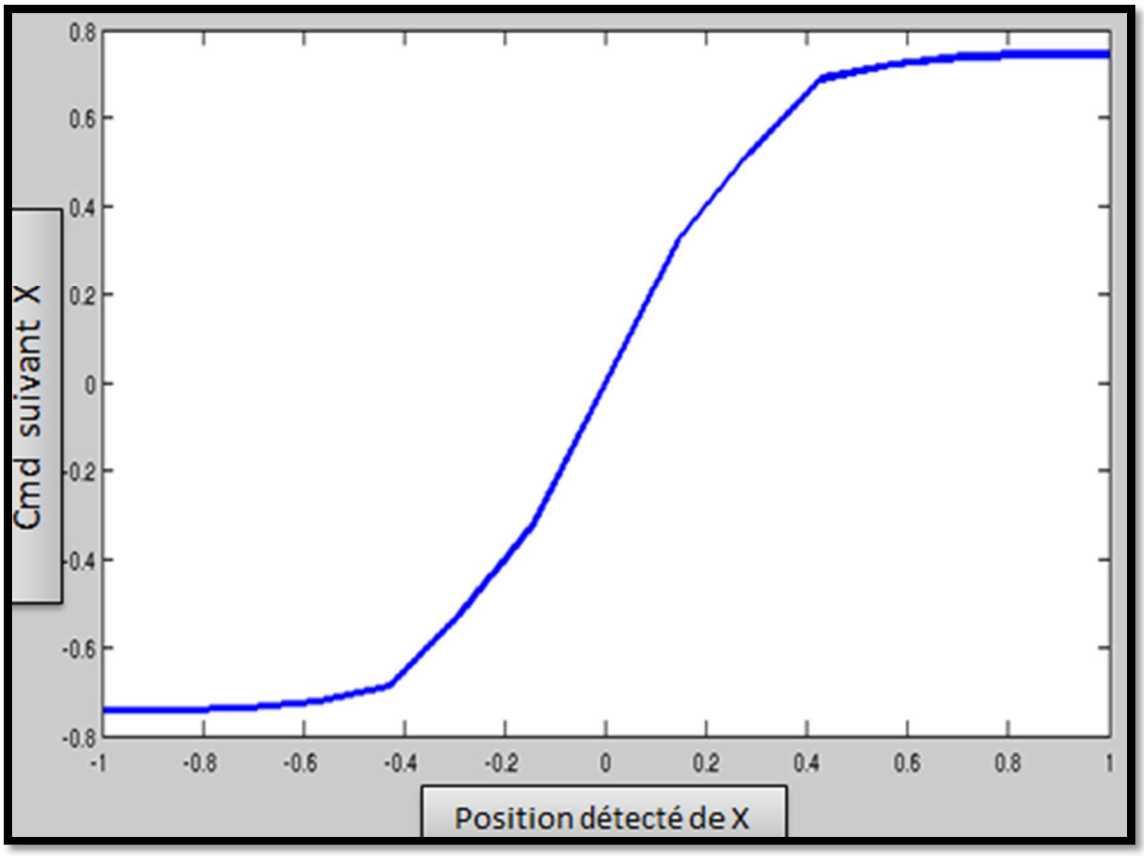


Figure III.13. Signal de commande X de la régulation par logique floue de l'entrée par rapport à la sortie.

III.4.1.4. Le mouvement de translation suivant l'axe Y

La Figure III.14 montre les quatre fonctions d'appartenances établis pour contrôler la translation du quadrotor suivant l'axe y.

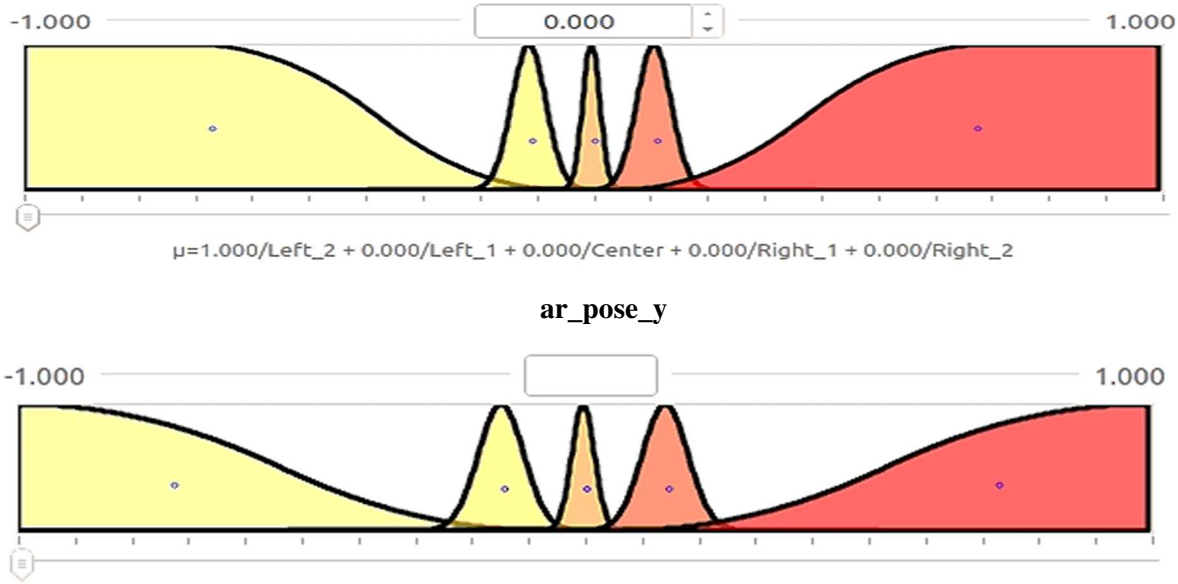


Figure III.14. Le mouvement de translation suivant l'axe Y.

Le tableau ci-dessus présente les entrées et sorties des règles floues :

Tableau III.4 Les entrées et sorties des règles floues (translation en Y).

Entrée	Entrée MF	Sortie	Sortie MF
ar_pose_y	Far_left	cmd_y_vel	go_far_left
ar_pose_y	Left	cmd_y_vel	go_left
ar_pose_y	Center	cmd_y_vel	stay
ar_pose_y	Right	cmd_y_vel	go_right
ar_pose_y	Far_right	cmd_y_vel	go_far_right

- **if** ar_pose_y **is** Left_2 **then** cmd_y_vel **is** left_2
- **if** ar_pose_y **is** Left_1 **then** cmd_y_vel **is** left_1
- **if** ar_pose_y **is** Center **then** cmd_y_vel **is** stay
- **if** ar_pose_y **is** Right_1 **then** cmd_y_vel **is** right_1
- **if** ar_pose_y **is** Right_2 **then** cmd_y_vel **is** right_2

Les résultats Y de la régulation par logique floue de l'entrée par rapport à la sortie sont illustré par la figure III.15.

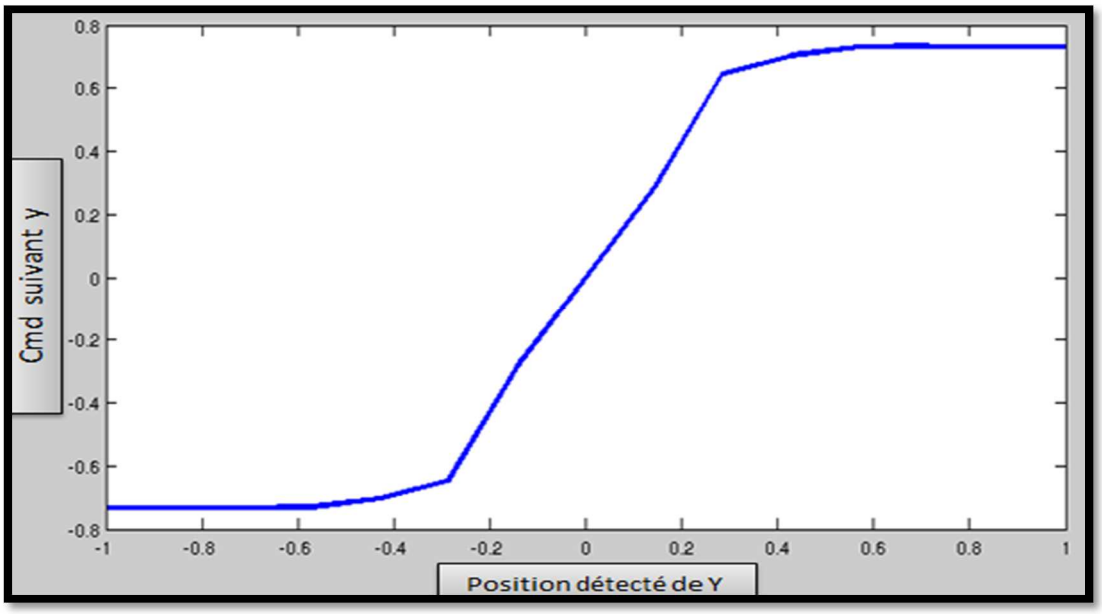


Figure III.15. Signal de commande Y de la régulation par logique floue de l'entrée par rapport à la sortie.

III.4.2. Premier test : la cible en face le quadrotor

Dans ce test, la cible se trouve devant le quadrotor à une distance de 10 m, comme illustré dans la figure III.16.

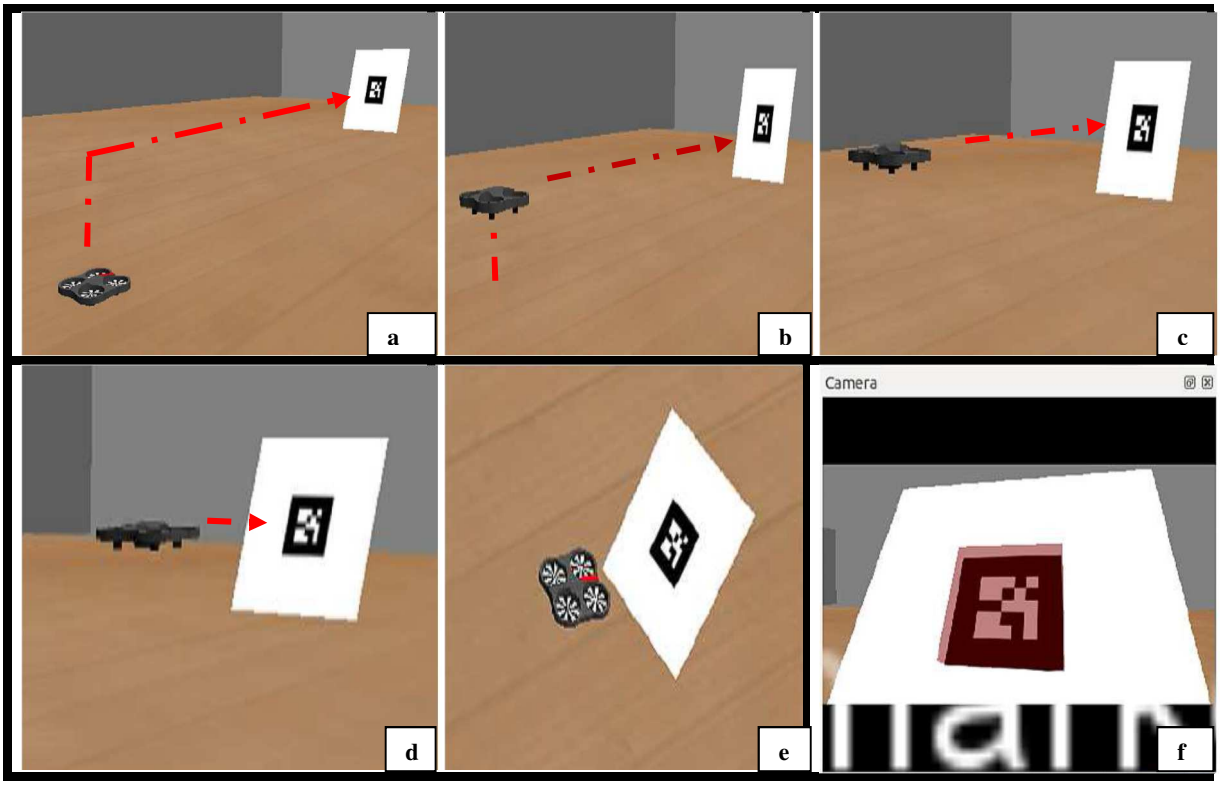


Figure III.16. Résultat de simulation du premier test.

Gazebo est un simulateur pour les systèmes multi-robot, qui génère à la fois un retour de capteur réaliste et des interactions physiquement possibles entre les objets. Tandis que Rviz est un environnement de visualisation 3D faisant partie des piles de visualisation ROS. Dans cette partie, nous avons créé un environnement dans Gazebo pour tester les algorithmes de suivi de cible. La figure III.15 montre que notre quadrotor, après le décollage, se dirige tout droit vers la cible, la ligne rouge correspondant au chemin suivi. Si la cible est visible par le quadrotor, l'AR.Drone démarre le suivi. Une fois que le drone détecte la cible, il ajuste son lacet plusieurs fois pour que la cible apparaisse toujours au centre de l'image (figures III.16 (b) (c) et (d). De cette façon, le drone se dirigera vers la cible. La figure III.17 montre le chemin parcouru par le quadrotor.

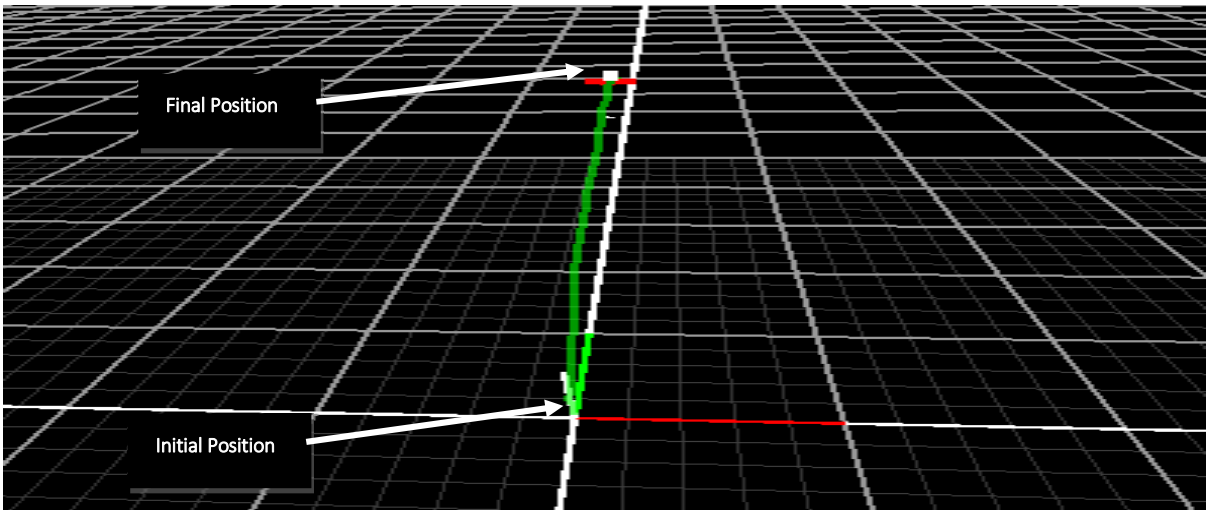


Figure III.17. Illustration de la trajectoire parcourue par le quadrotor dans le premier test.

La figure III.18 montre l'évolution de l'erreur pour les contrôleurs de vitesses latérale et longitudinaleV_x, V_y et V_z. Cette figure montre comment l'erreur sur l'axe X a été réduite à zéro en moins de 41 secondes. La figure III.19 illustre l'évolution des vitesses du quadrotor selon les trois axes x, y, z.

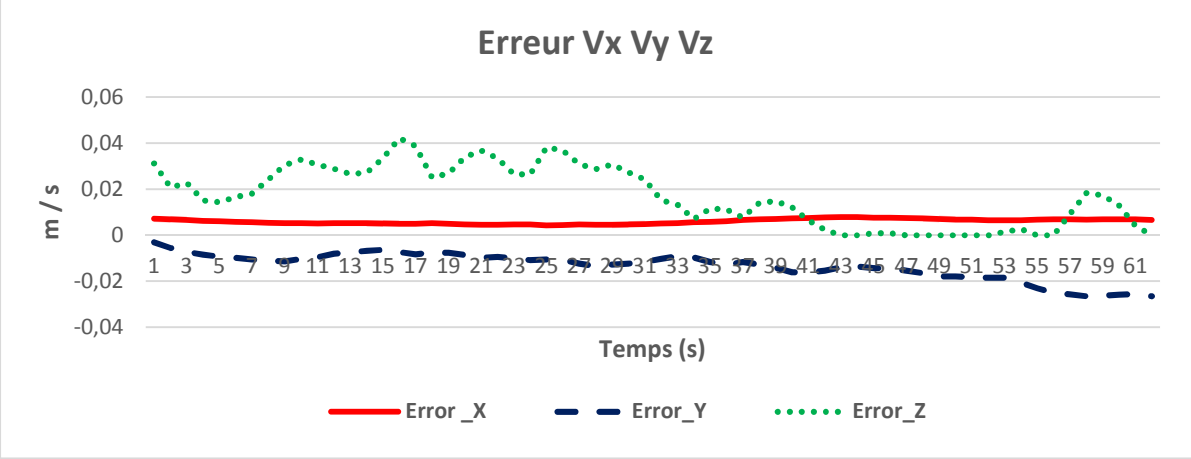


Figure III.18. Evolution des erreurs pour les contrôleurs de vitesses longitudinales et latérales dans le premier test de simulation.

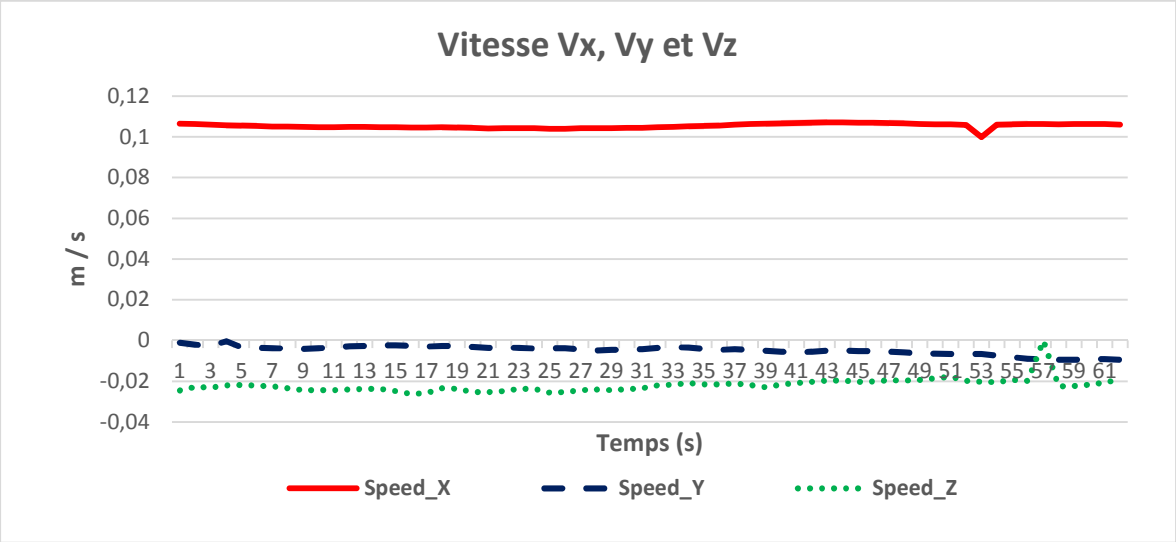


Figure III.19. Evolution des vitesses du quadrotor selon les axes x, y et z dans le premier test de simulation.

La figure III.20 montre l'évolution de la rotation selon l'axe z de l'erreur absolue du quadrotor.

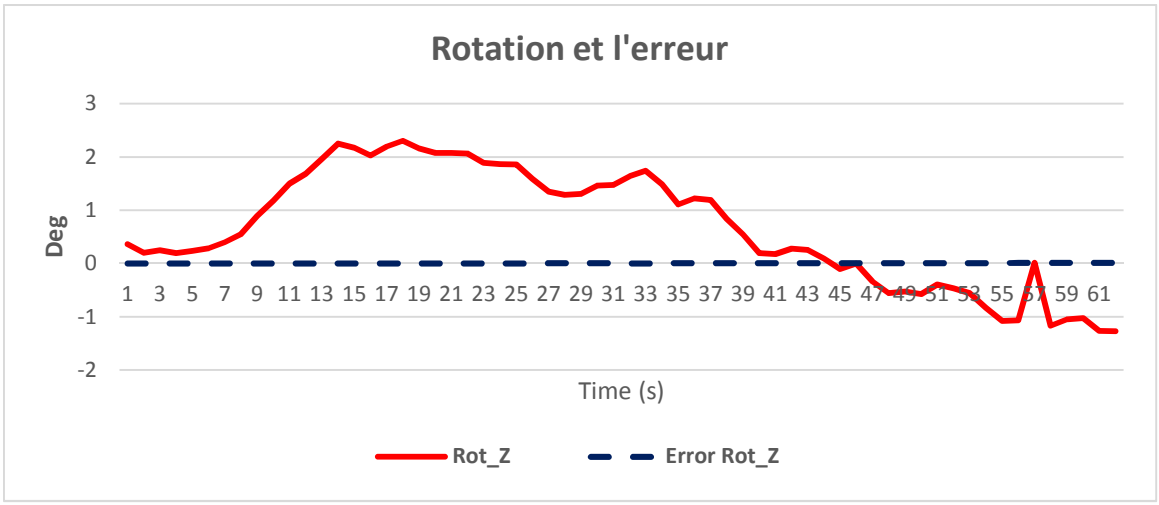


Figure III.20. Evolution de la rotation et de l'erreur du quadrotor le long de l'axe z dans le premier test de simulation.

D'après la figure III.20, nous constatons que l'erreur de rotation est presque nulle tout au long du parcours du quadrotor, nous remarquons que le drone change sa rotation suivant la position de la cible.

III.4.3. Deuxième test : la cible est à une hauteur par rapport au quadrotor

Ce test a pour but de montrer l'évolution de la commande de la vitesse suivant l'axe Z.

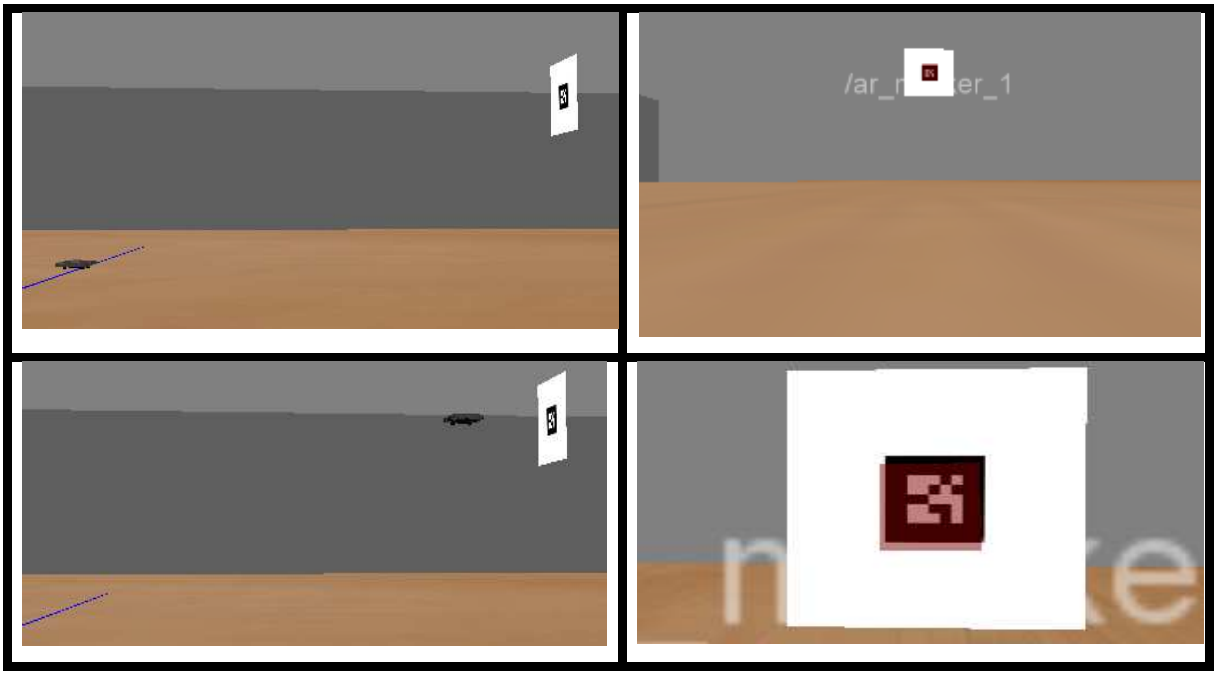


Figure III.21. Résultat de simulation du second test de simulation.

La trajectoire suivie par le quadrotor lors du deuxième test de simulation est représentée dans la figure III.22.

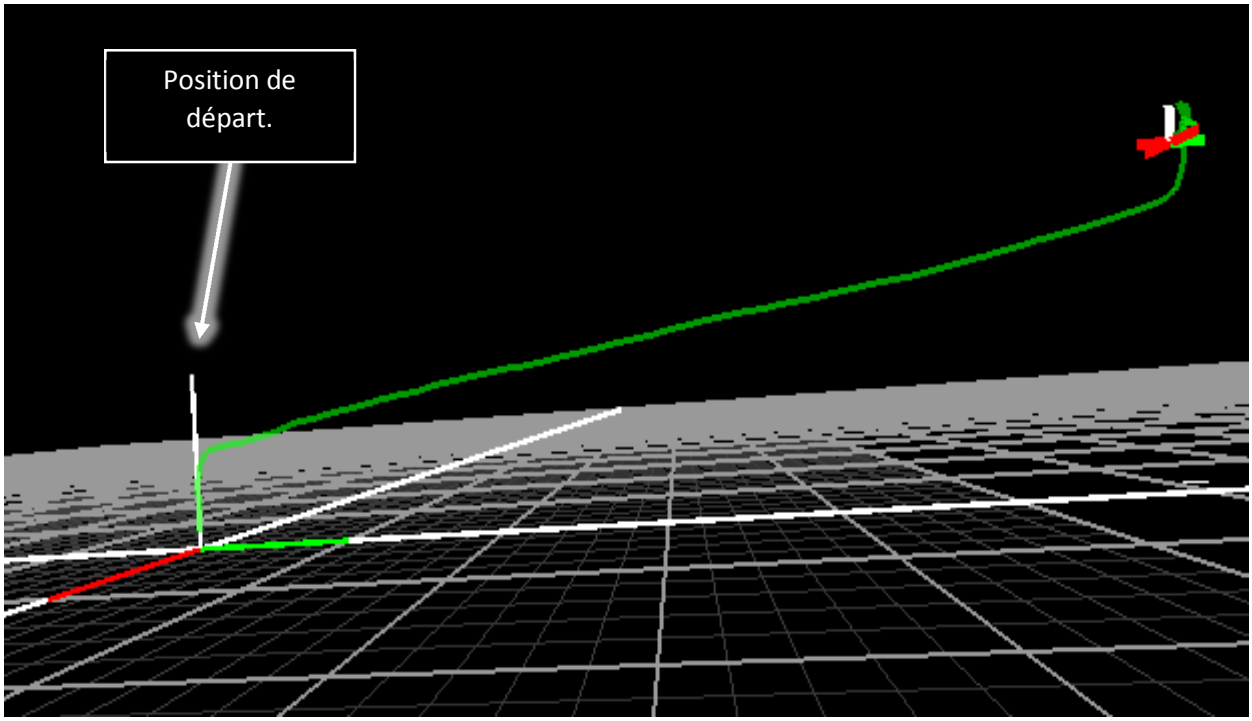


Figure III.22. Trajectoire effectuée par le quadrotor lors du deuxième test de simulation.

La figure III.23, illustre la vitesse du quadrotor le long des axes x, y et Z.

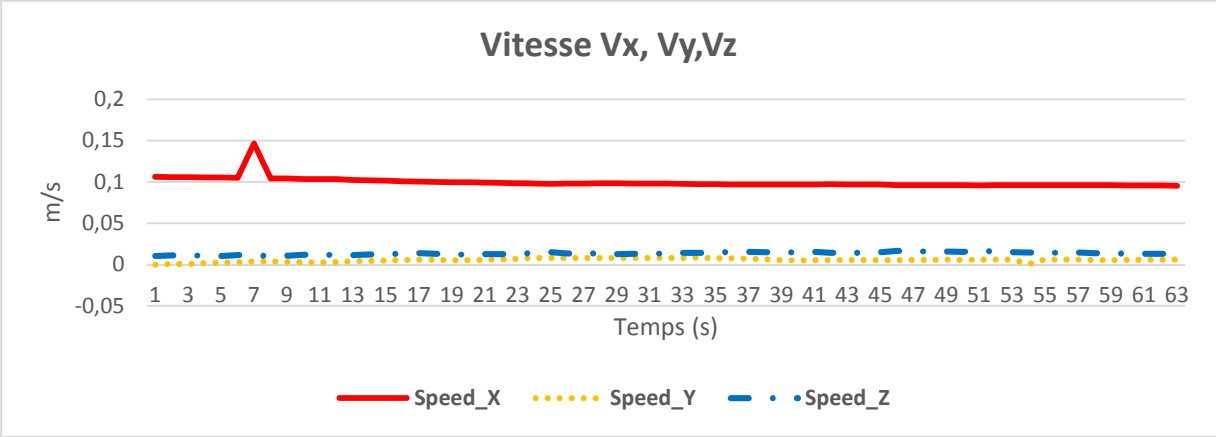


Figure III.23. Evolution des vitesses du quadrotor selon les axes x, y et z dans le deuxième test de simulation.

D'après la figure ci-dessus, nous remarquons que la vitesse du quadrotor suivant l'axe x est presque constante, et elle est autour de 0,1m/s par contre la vitesse suivant les axes y et z elle est autour de zéro.

La figure III.24, montre l'évolution de la rotation du quadrotor suivant l'axe z.

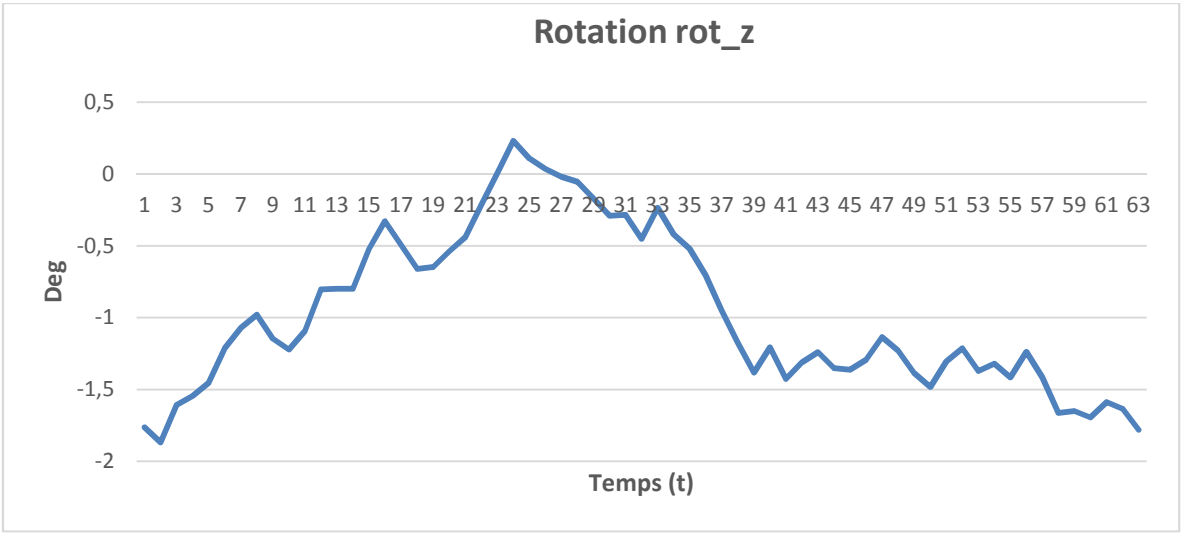


Figure III.24. Evolution de la rotation du quadrotor le long de l'axe z dans le deuxième test de simulation.

D'après la figure ci-dessus nous remarquons que le quadrotor réalise plusieurs rotations dans le but de se positionner en face de la cible.

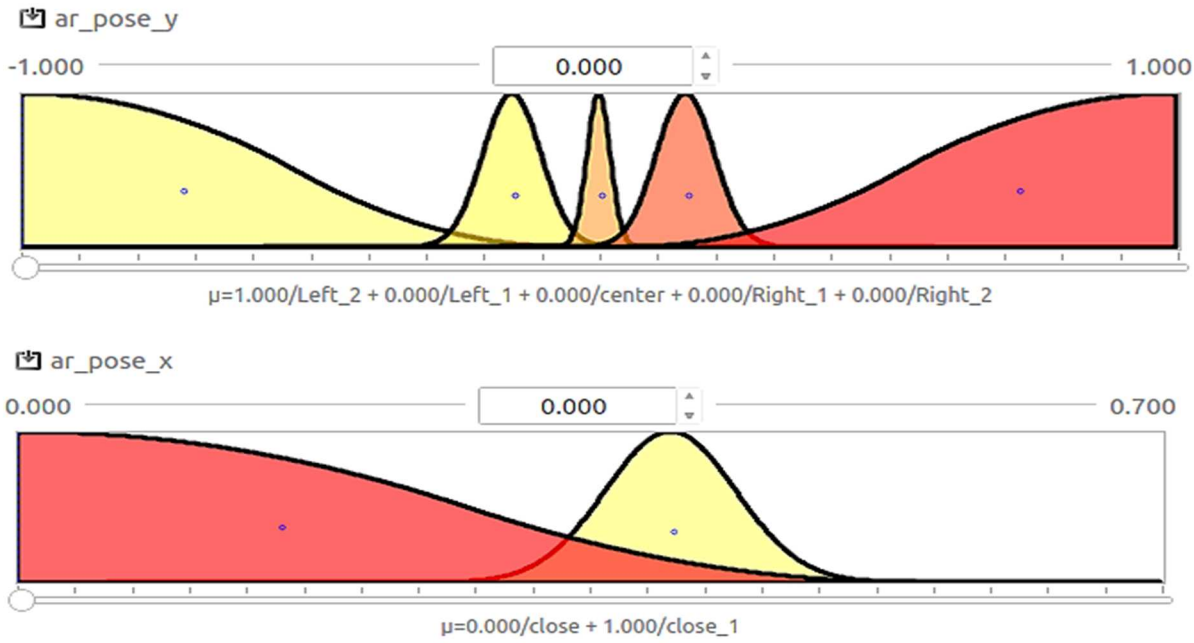
III.4.4. Conception du contrôleur flou pour l'évitement d'obstacle

Le contrôle de l'évitement a pour but de contrôler la direction du quadrotor afin d'éviter un obstacle représenté par un marqueur visuel. Pour cela, le quadrotor doit se translater selon les axes X et Y en même temps pour la détection du marqueur.

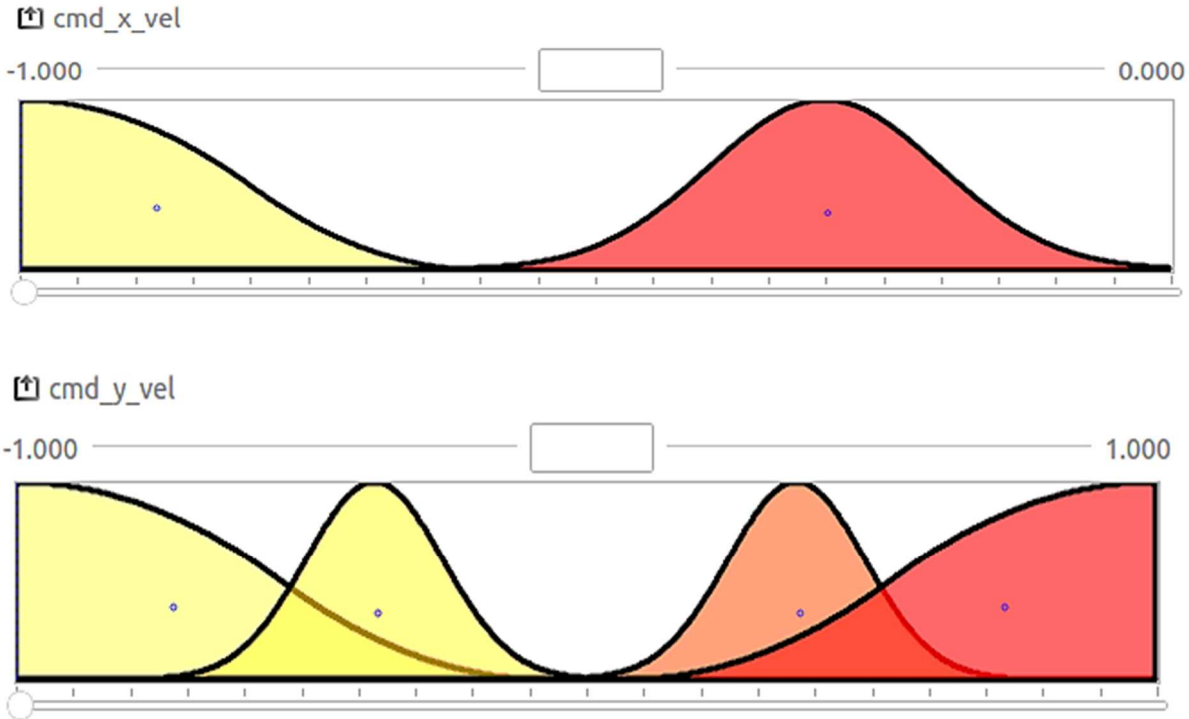
Le contrôleur flou de l'évitement est mis en œuvre dans un total de 12 règles pour contrôler le mouvement linéaire dans les deux axes (X et Y) et le mouvement de rotation le long de l'axe Z.

III.4.4.1. Le mouvement de translation suivant les axes X et Y

La Figure III.25 montre les quatre fonctions d'appartenances établies pour contrôler la translation du quadrotor suivant les axes X et Y.



ar_pose_y et ar_pose_x (entrées)



cmd_x_vel et cmd_y_vel (sorties)

Figure III.25. Le mouvement de translation suivant l'axe x (en haut) et l'axe y (en bas).

Le tableau ci-dessous présente les entrées et sorties des règles floues :

Tableau III.5 Les entrées et sorties des règles floues (translation en X et Y).

Entrée	Entrée MF	Sortie	Sortie MF
ar_pose_y ar_pose_x	Left_2 close	cmd_x_vel cmd_y_vel	forward left_1
ar_pose_y ar_pose_x	Left_1 close_1	cmd_x_vel cmd_y_vel	forward_1 left_2
ar_pose_y ar_pose_x	center close	cmd_x_vel cmd_y_vel	forward_1 right_1
ar_pose_y ar_pose_x	center close_1	cmd_x_vel cmd_y_vel	forward right_1
ar_pose_y ar_pose_x	Right_1 close_1	cmd_x_vel cmd_y_vel	forward_1 right_2
ar_pose_y ar_pose_x	Right_2 close	cmd_x_vel cmd_y_vel	forward right_2

- **if** ar_pose_y **is** Left_2 **and** ar_pose_x **is** close **then** cmd_x_vel **is** forward **and** cmd_y_vel **is** left_1
- **if** ar_pose_y **is** Left_1 **and** ar_pose_x **is** close_1 **then** cmd_x_vel **is** forward_1 **and** cmd_y_vel **is** left_2
- **if** ar_pose_y **is** center **and** ar_pose_x **is** close **then** cmd_x_vel **is** forward_1 **and** cmd_y_vel **is** right_1
- **if** ar_pose_y **is** center **and** ar_pose_x **is** close_1 **then** cmd_x_vel **is** forward **and** cmd_y_vel **is** right_2
- **if** ar_pose_y **is** Right_1 **and** ar_pose_x **is** close_1 **then** cmd_x_vel **is** forward_1 **and** cmd_y_vel **is** right_2
- **if** ar_pose_y **is** Right_2 **and** ar_pose_x **is** close **then** cmd_x_vel **is** forward **and** cmd_y_vel **is** right_1

Les graphes suivants montrent les surfaces floues

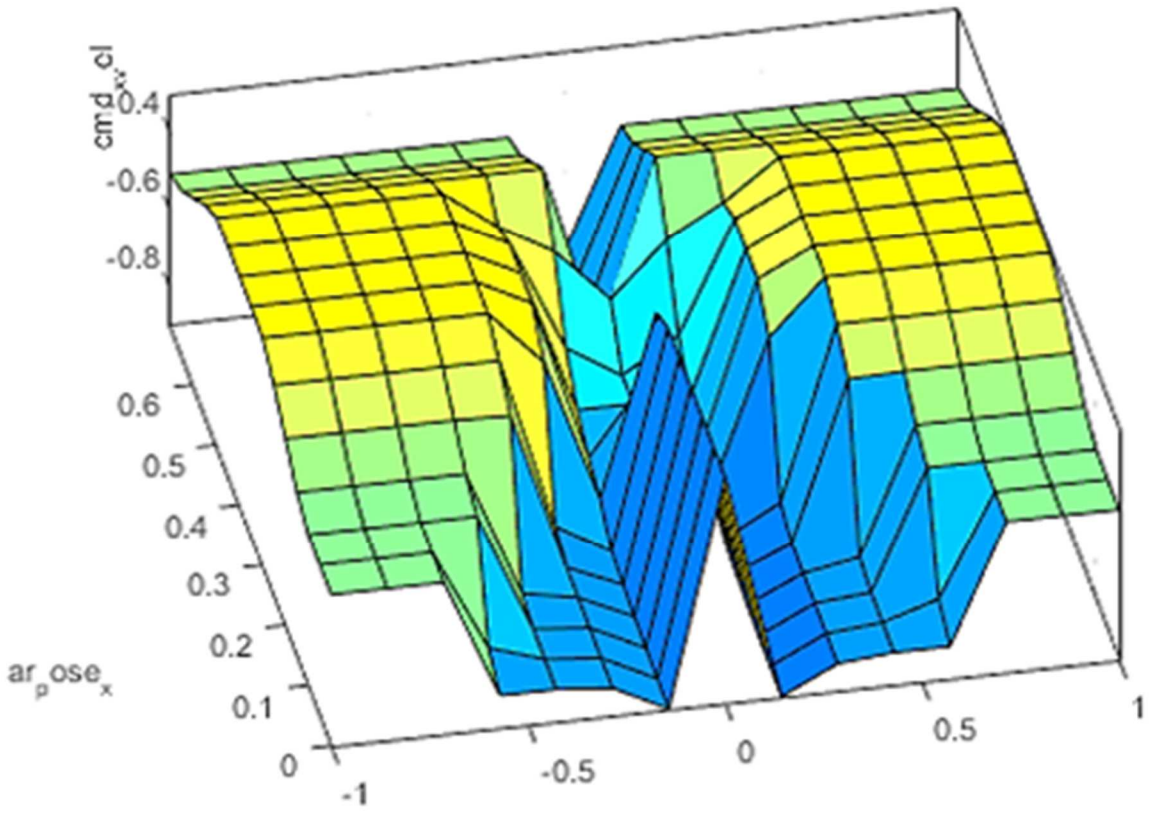


Figure III.26. Surface Floue selon X.

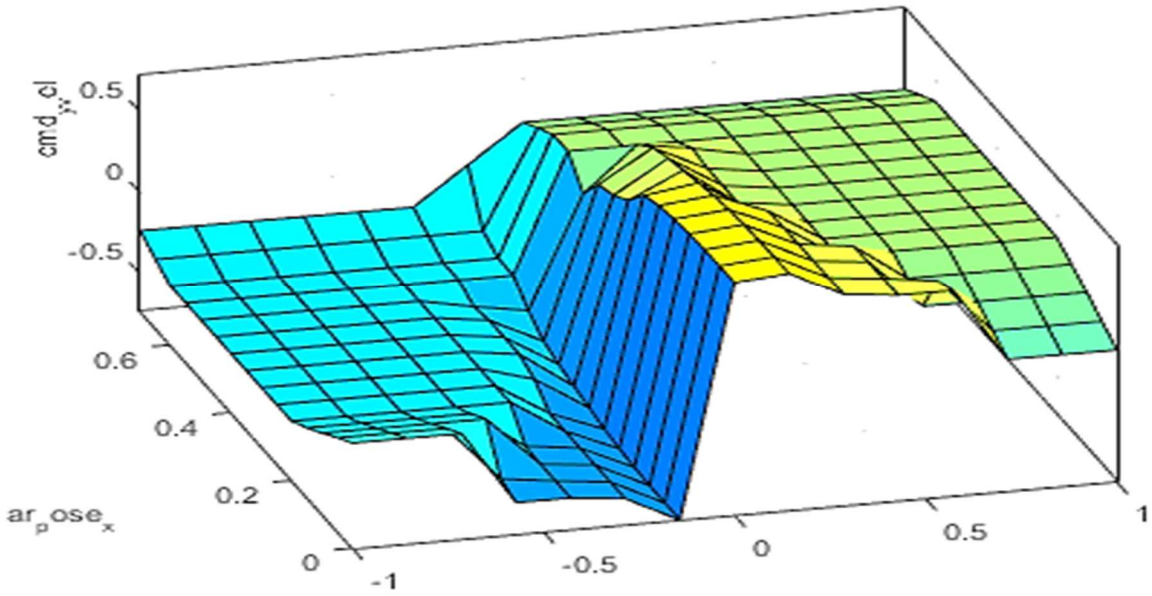


Figure III.27. Surface Floue selon Y.

III.4.4.2. Troisième test : évitement d'obstacle

Comme une évolution complémentaire pour notre projet nous avons réalisé un contrôleur flou qui a pour but de commander un AR.Drone pour maintenir une trajectoire donnée en évitant des obstacles. Ce contrôleur a été testé avec le simulateur Gazebo et la figure (III.28) montre l'environnement réalisé.

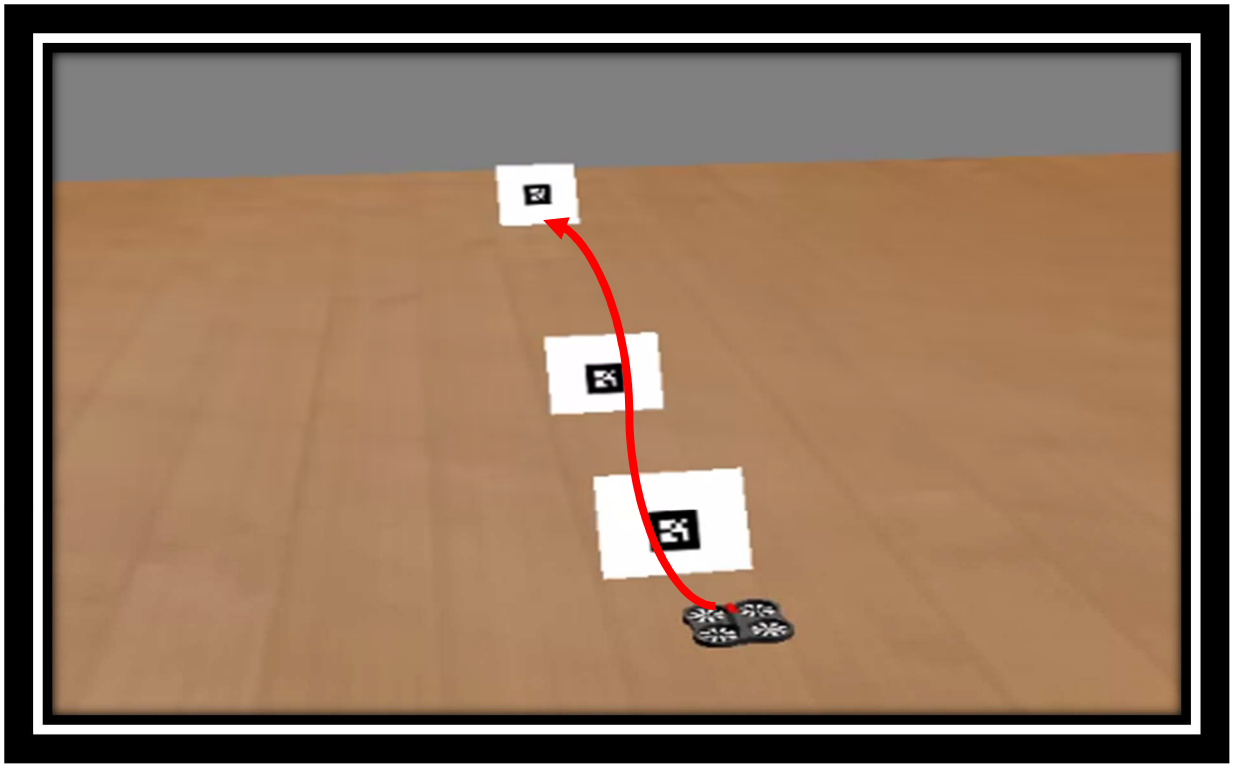
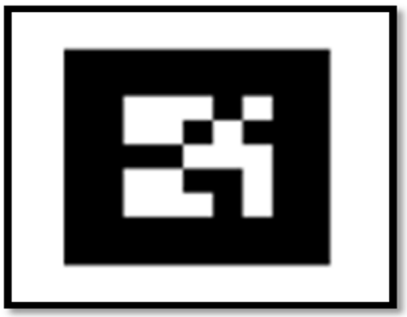
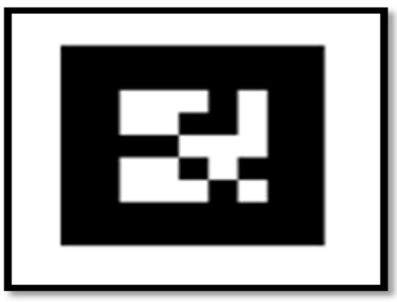


Figure III.28. Environnement de simulation sous Gazebo du troisième test.

Dans la simulation montrée par la figure (III.28) nous avons considéré la balise (A) comme obstacle et la balise (B) comme point d'arrivée.



(A)



(B)

La figure III.29, présente les résultats de simulation pour l'évitement d'obstacle et suivi de cible lors du troisième test.

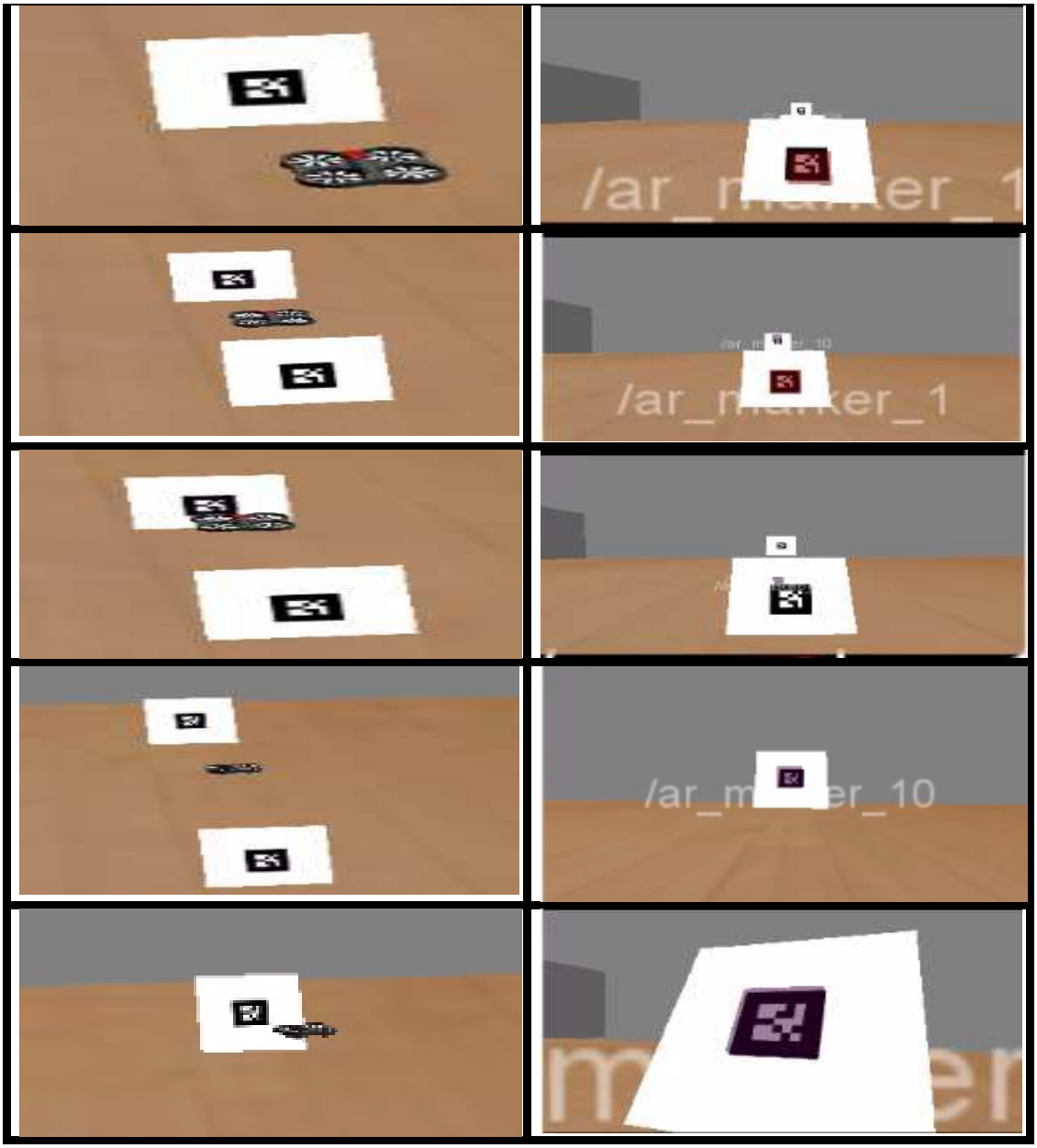


Figure III.29. Capture de scène de l'évitement d'obstacle lors du troisième test de simulation.

Dans cette figure nous constatons que l'AR.Drone s'est déplacé suivant son axe X jusqu'à atteindre la première balise (A) où il l'a évité en se plaçant au-dessus.

Quand il arrive à la deuxième balise (A) nous voyons qu'il l'a évité en se plaçant au-dessus.

Après les deux premières balise l'AR.Drone a continué sa trajectoire jusqu'à son arrivée à la balise (B) considérée comme un point d'arrivée.

Nous remarquons aussi qu'à chaque fois que le quadrotor identifie une cible, il dessine un carré avec une couleur identique à chaque balise.

La figure III.30, montre l'évolution de la vitesse linéaire pour les trois axes.

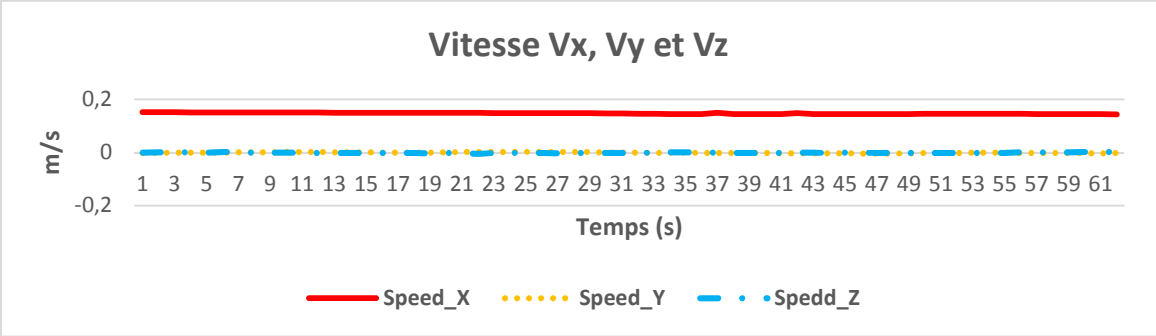


Figure III.30. Vitesse linéaire pour les trois axes lors du troisième test de simulation.

D'après la figure ci-dessus, nous remarquons que le quadrotor se déplace tout au long de l'axe X avec une vitesse linéaire constante, par contre pour les deux autres axes à savoir Y et Z le quadrotor maintien une vitesse linéaire proche du zéro.

La figure III.31, illustre l'évolution de la rotation du quadrotor lors du troisième test de simulation.

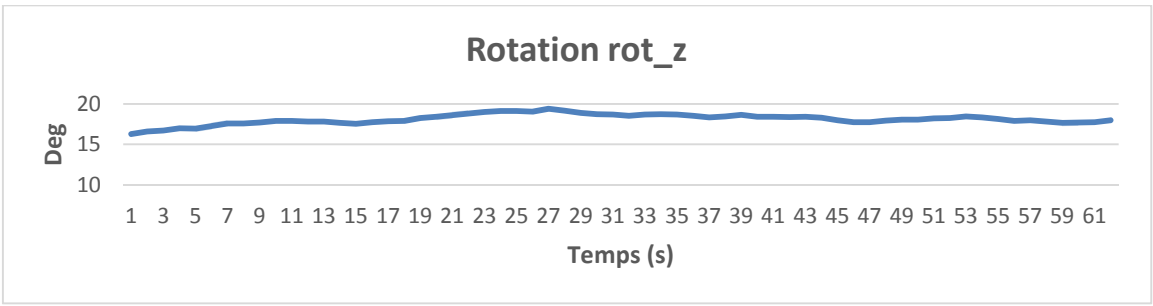


Figure III.31. Rotation suivant l'axe z lors du troisième test de simulation.

D'après la figure ci-dessus, nous remarquons que le quadrotor réalise des orientations selon l'axe Z pour pouvoir éviter les cibles 1 et 2.

III.4.4.3. Quatrième test : Evitement et suivi de cible

Comme exemple d'application des deux contrôleurs précédemment présentés, à savoir le contrôleur du suivi et le contrôleur de l'évitement d'obstacle, un environnement a été réalisé sous Gazebo comme le montre la figure III.32, qui a pour but de commander le quadrotor pour maintenir une trajectoire prédéfinie par l'ordre ascendant des balises visuelles (1,2…5), une fois la dernière balise dans le champ de vision du quadrotor, celui-ci s'approche d'elle et maintien une distance de 1 m.

Les tâches de commande que doit suivre le quadrotor, en utilisant le contrôleur du suivi de cible et le contrôleur de l'évitement d'obstacle, sont les suivantes :

- Evitement de la cible 1 et recherche de la cible 2.
- Rotation devant la cible 2 et recherche de la cible 3.
- Rotation devant la cible 3 et recherche de la cible 4.
- Evitement de la cible 4 et recherche de la cible 5.
- Maintien d'une position fixe devant la cible.

La figure ci-dessous présente la trajectoire que doit suivre l'AR Drone pour atteindre sa cible.

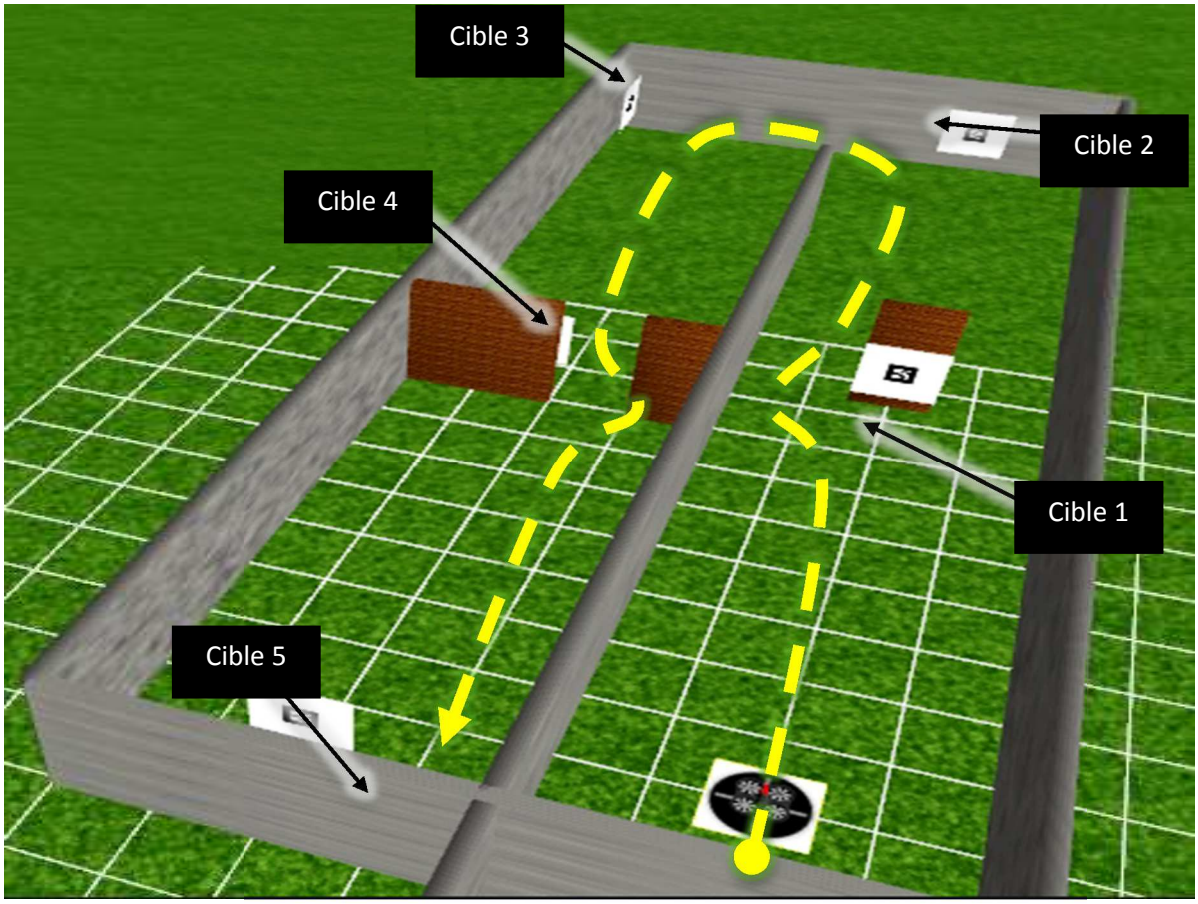


Figure III.32. Le trajet que le quadrotor doit réaliser.

La figure III.33, illustre les résultats obtenus pour le suivi et l'évitement.

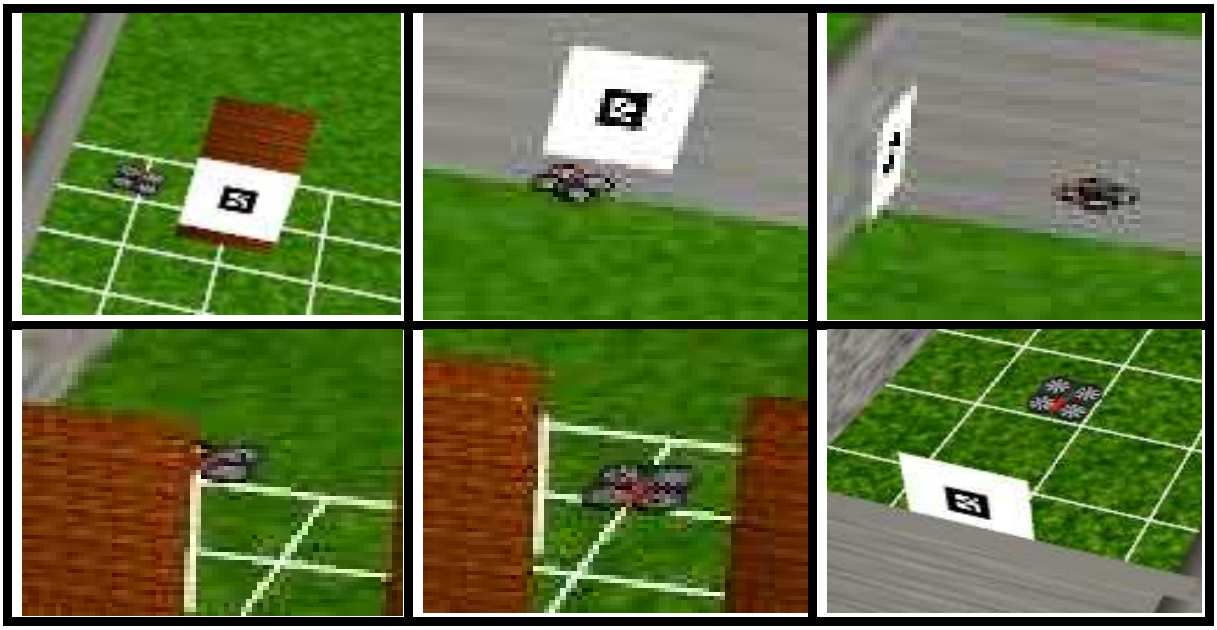


Figure III.33. Résultat de simulation pour le quatrième test de simulation.

La figure III.34, montre le chemin parcouru par le quadrotor lors de ce dernier test de simulation.

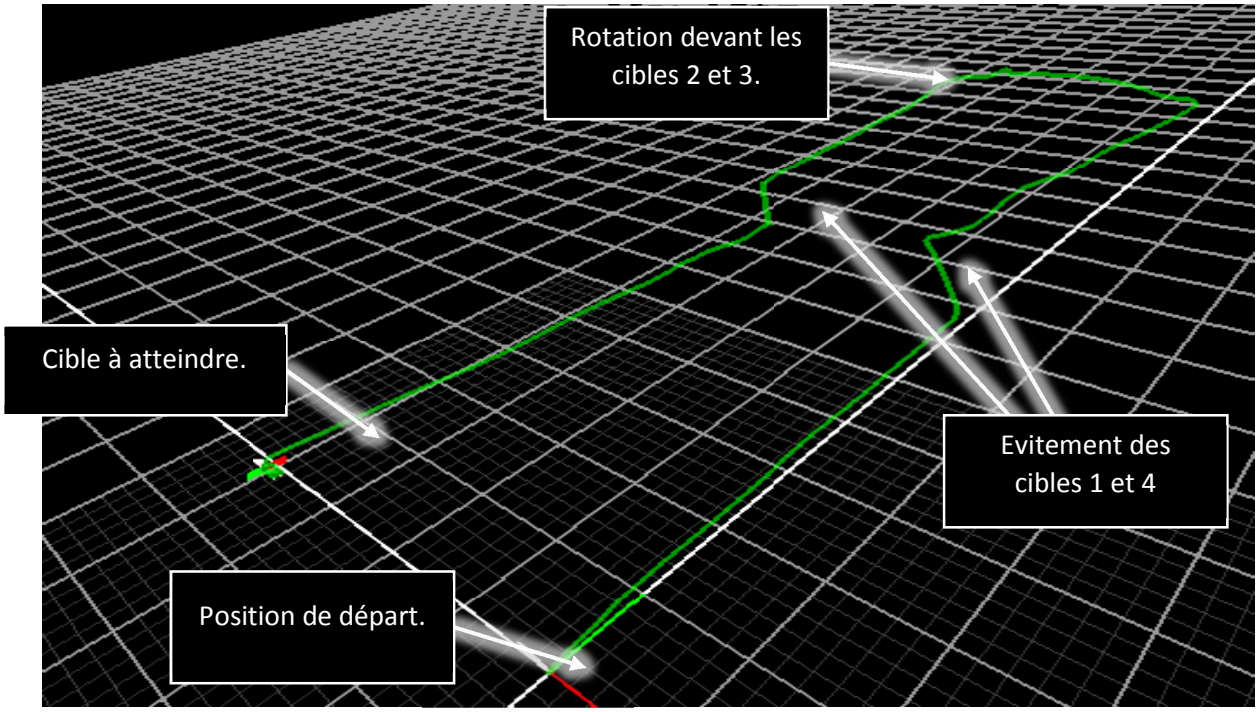


Figure III.34. Chemin parcouru par le quadrotor lors du dernier test.

D'après la figure nous remarquons que l'AR.Dronea bien effectué un évitement de la première cible jusqu'à la quatrième tout en suivant une trajectoire prédéfinie d'avance pour atteindre la dernière cible.

La figure III.35, illustre les vitesses linéaires du quadrotor selon les trois axes.

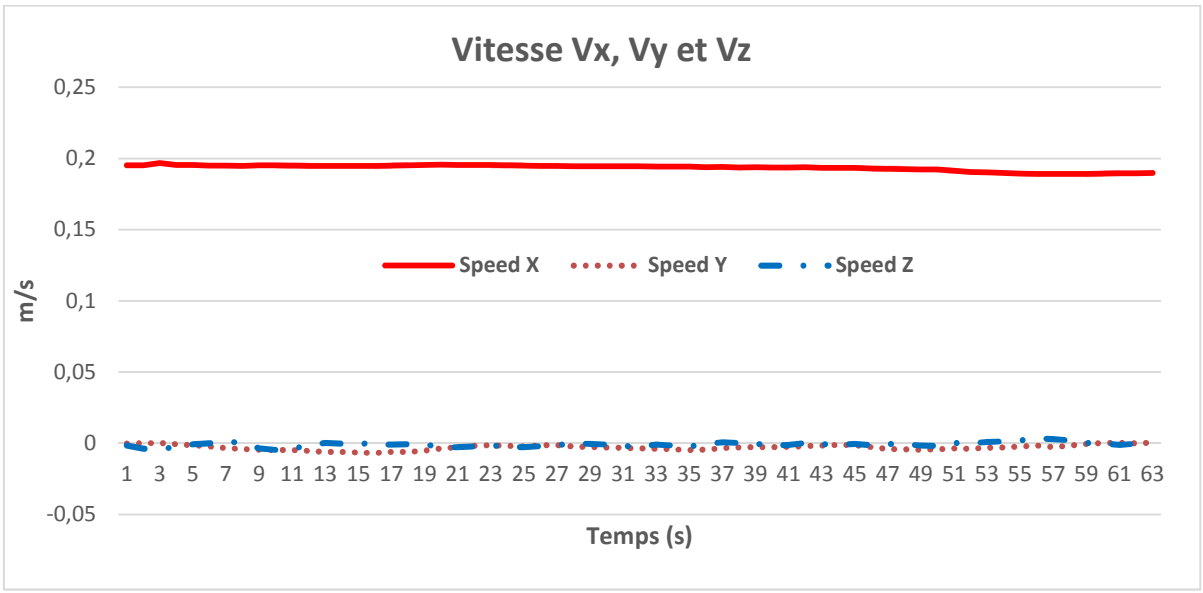


Figure III.35. Vitesse linéaire selon les trois axes pour le dernier test de simulation.

D'après la figure ci-dessus, nous constatons que le quadrotor maintien une vitesse linéaire constante suivant l'axe des X, même chose pour les deux autres axes sauf que leurs vitesses sont presque nulles.

La figure III.36, montre l'évolution de la rotation du quadrotor suivant l'axe Z.

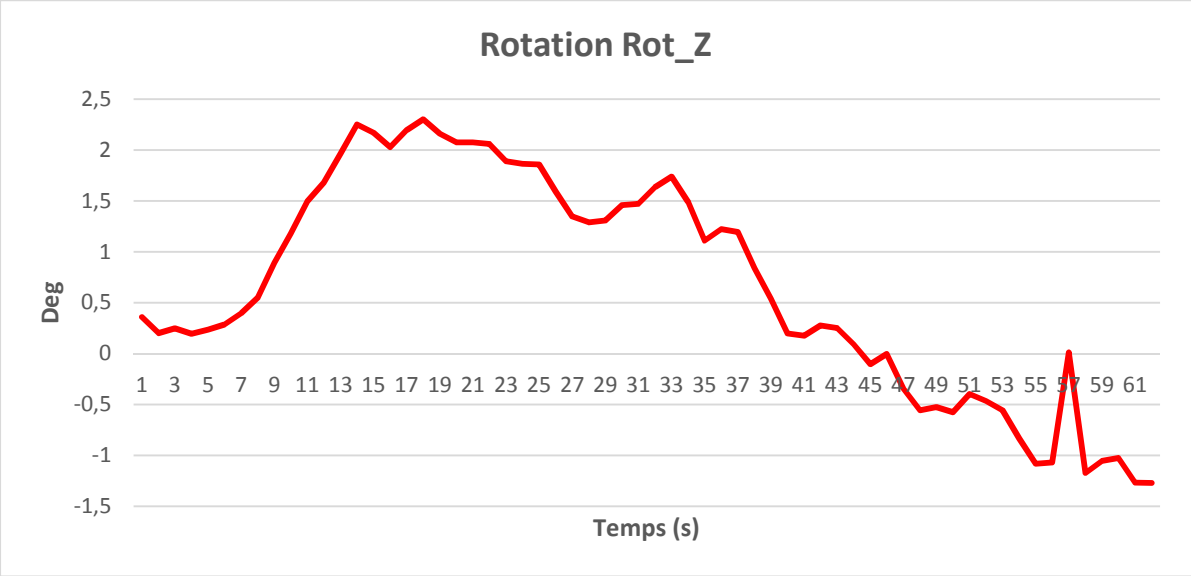


Figure III.36. Rotation du quadrotor suivant l'axe Z pour le dernier test de simulation.

D'après la figure ci-dessus, nous remarquons que le quadrotor effectue plusieurs rotations pour éviter les cibles.

III.5. TEST EXPERIMENTAL

Dans la partie matérielle, un ordinateur portable a été utilisé pour les communications et le contrôle du quadrotor l'AR Drone 2.0. ROS (Robot Operating System) a été utilisé comme système d'exploitation afin de contrôler le quadrotor de la même manière que les tests de simulation.

Dans les tests des contrôleurs flous développés sur le matériel, la configuration de la figure III.36 a été utilisée. Dans cette figure, le nœud en couleur rouge 'ardrone_driver' est le responsable du contrôle de l'AR Drone. Ce nœud utilise l'ar_track_alvar le nœud en vert pour la détection et l'identification de la cible. Les informations de la balise visuelle sont envoyées vers le nœud de contrôleur flou qui est en bleu 'fuzzy_tracker' pour les traiter avec les règles floues déjà développées en simulation.

Les résultats du traitement sont les commandes qui vont être envoyées au 'drone_gui', le nœud en jaune est responsable de la génération des commandes pour l'AR Drone à partir d'une interface graphique.

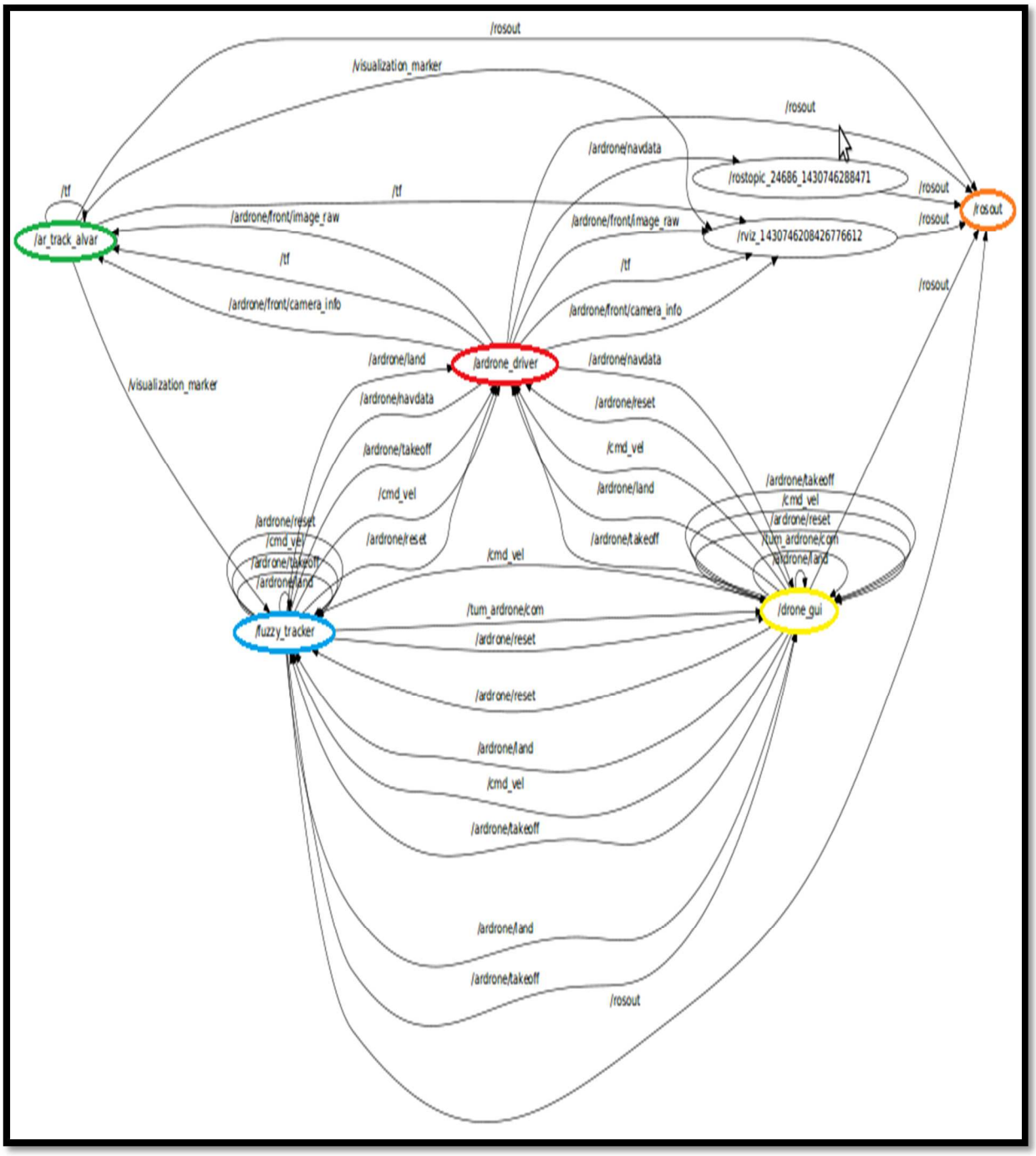


Figure III.37. Configuration ROS pour l'expérimentation réelle.

Après avoir appuyé sur le bouton Take-Off dans l'interface graphique, il est recommandé de déplacer le quadrotor légèrement pendant environ dix ou quinze secondes, de manière à ce que l'algorithme lance le processus de suivi de cible. Si l'algorithme de suivi est capable d'identifier l'objet, il affichera un carré rouge autour de la cible. Ensuite, après avoir appuyé sur le bouton de décollage, le quadrotor décolle et commence à suivre l'objet de manière autonome. Par la suite, le processus du contrôleur flou est activé.

La figure III.38 montre certaines images capturées par la caméra lors de l'exécution de ce test. Les figures III.38 (a) et (b) montrent le début de l'essai. Les figures III.38 (c) et (d) montrent l'image capturée au milieu du test et la figure III.38 (e) montre le quadrotor en face la cible.

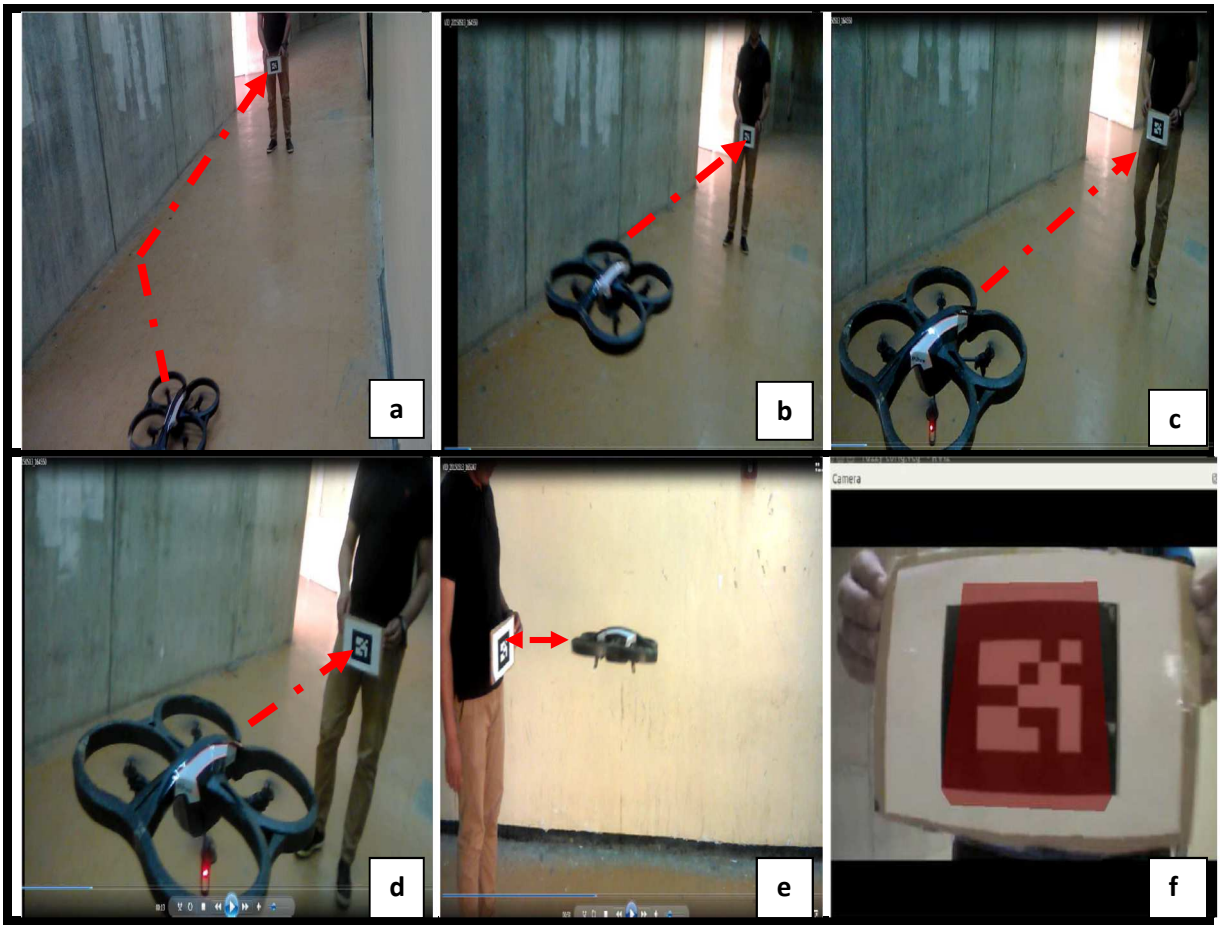


Figure III.38. Expériences en temps réel montrent les performances du système de suivi de cible proposé.

La figure III.38 montre que le quadrotor suit parfaitement la cible. Nous remarquons à la figure III.38 (e) que l'AR.Drone suit l'objet cible avec une distance de sécurité fixe et conserve l'image de la cible centrée sur sa caméra frontale. D'après les résultats de test réel nous constatons que le quadrotor suit parfaitement la cible.

La figure III.39 illustre l'évolution de la vitesse selon les trois axes x, y et z. Nous notons que le quadrotor se déplace rapidement vers la cible avec une vitesse de départ de 1 m / s, mais une fois la cible se rapproche du quadrotor, la vitesse diminue et atteint 0,1 m / s. Cela est justifié par le fait que le drone doit respecter une distance de 0,5 m une fois proche de la cible.

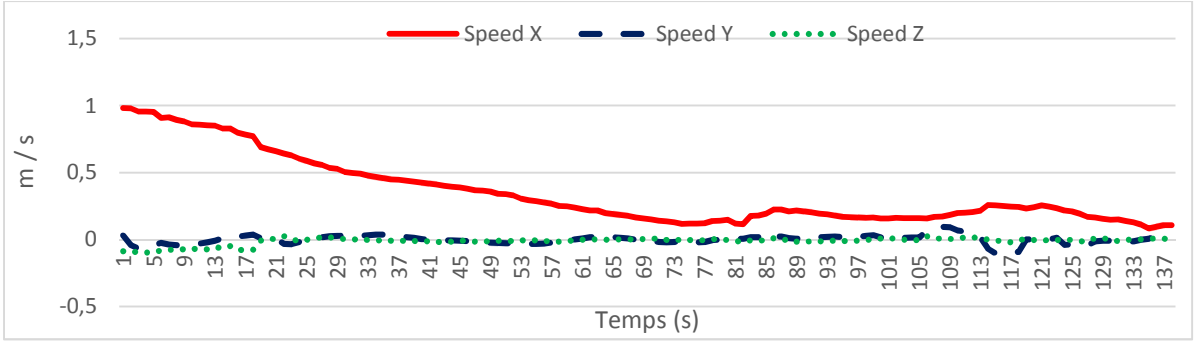


Figure III.39. Vitesse linéaire du quadrotor selon les axes x, y et z pour le test expérimental.

La figure III.40 montre l'évolution de l'erreur de vitesse selon les trois axes x, y et z.

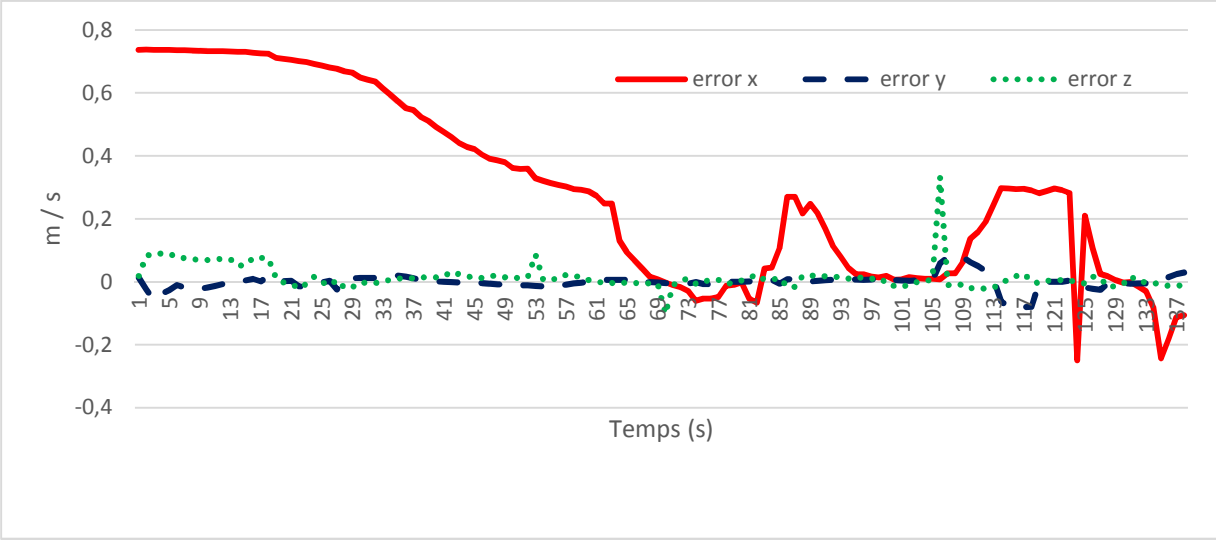


Figure III.40. Evolution de l'erreur de vitesse selon les axes x, y et z pour le test expérimental.

La figure III.40, montre que l'erreur de vitesse sur l'axe des x est très grande au décollage elle atteint 0,7 m / s, par la suite, elle diminue progressivement pour atteindre presque zéro. Pour les erreurs sur les axes y et z, notons que leurs valeurs oscillent autour de zéro.

III.6. CONCLUSION

Dans ce chapitre, des contrôleurs flous ont été conçus et mis en œuvre pour l'identification et le suivi de cible à l'aide d'un quadrotor AR.Drone. Une démonstration des contrôleurs basée sur la simulation a été fournie à l'aide de la combinaison des outils open source FuzzyLite, ROS Gazebo et 'ar_track_alvar' avec Tum_ardrone Simulator. Les résultats de la simulation ont montré des performances acceptables pour le suivi de cible. Les résultats de simulation obtenus pour différents tests sont prometteurs. Les variables de contrôle ont été testées, conformément au compromis inévitable entre stabilité, précision et rapidité de la réponse du système.

Une fois que le contrôleur flou optimal a été obtenu, nous avons procédé à des tests réels en utilisant un AR.Drone. La réponse rapide du contrôleur et la petite erreur commise lors du test démontrent l'excellent comportement du contrôleur.

En général, le contrôleur flou peut être utilisé pour résoudre le problème de suivi de cible à l'aide de l'AR.Drone. D'après les expériences effectuées, il apparaît que le quadrotor est capable de suivre une cible donnée avec différentes positions initiales.

Dans le chapitre suivant, nous allons traiter le suivi de cible en utilisant un contrôleur à base d'un modèle neuronal.

Chapitre IV :

Identification et Suivi de Cible A Base d'un Contrôleur ANN

Identification et Suivi de Cible A Base d'un Contrôleur ANN

IV.1. INTRODUCTION

Le suivi visuel est l'un des composants les plus cruciaux du système de vision robotique. C'est encore plus difficile lorsque l'objet change d'échelle, de pose ou d'éclairage. De nombreuses approches en vision par ordinateur ont été proposées pour effectuer un suivi visuel à long terme. Le suivi visuel reste une tâche ardue en vision par ordinateur.

Dans ce chapitre, nous commencerons par une introduction sur les réseaux de neurones artificiels et leurs principes de fonctionnement ainsi que leurs domaines d'utilisation, par la suite nous présenterons les résultats et tests d'apprentissage du réseau de neurone que nous avons conçu, aussi les différentes architectures testées et les paramètres choisis, le tout dans un but d'optimisation. Nous introduirons aussi la détection de balise pour le suivi de la cible.

Dans ce chapitre, nous décrivons aussi les résultats obtenus en simulation sur Gazebo et les tests en environnement réel en utilisant le quadrotor AR.Drone 2.0.

IV.2. RESEAUX DE NEURONES

IV.2.1. Définition des réseaux de neurones artificiels

Les réseaux de neurones désignent à la fois l'étude des systèmes biologiques et leur modélisation informatique, plus ou moins simplifiée, à différentes applications, comme la reconnaissance de caractères, son premier terrain d'application historique. Dans ce cadre a été définie la notion de réseaux de neurones artificiels. Les RNs sont des réseaux fortement connectes de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire (neurone artificiel) calcule une sortie unique sur la base des informations qu'il reçoit. En effet, dans un réseau de neurones artificiel (Figure IV.1), chaque neurone possède plusieurs entrées et une sortie. Chacune de ses entrées se voit affecter un poids (dit poids synaptique) diffèrent, et si la somme ainsi pondérée des signaux des différentes entrées dépasse un seuil, la sortie prend une valeur positive (le neurone déclenche) [42]. Les RNs forment une solution idéale pour certains problèmes qui nécessitent un raisonnement, ou qui sont de complexité élevée.

IV.2.2. Le neurone artificiel

La figure IV.1 montre la structure d'un neurone artificiel. Chaque neurone artificiel est un processeur élémentaire. Il reçoit un nombre variable d'entrées en provenance de neurones amont. A chacune de ces entrées est associé un poids w abréviation de weight (poids en anglais) représentatif de la force de la connexion. Chaque processeur élémentaire est doté d'une sortie unique, qui se ramifie ensuite pour alimenter un nombre variable de neurones avals. A chaque connexion est associé un poids.

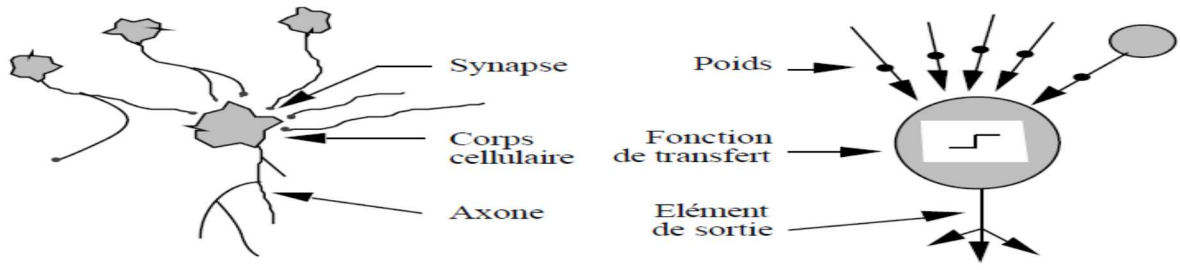


Figure IV.1. Mise en correspondance neurone biologique / neurone artificiel.

La figure IV.2 donne les notations que nous utilisons dans cette thèse.

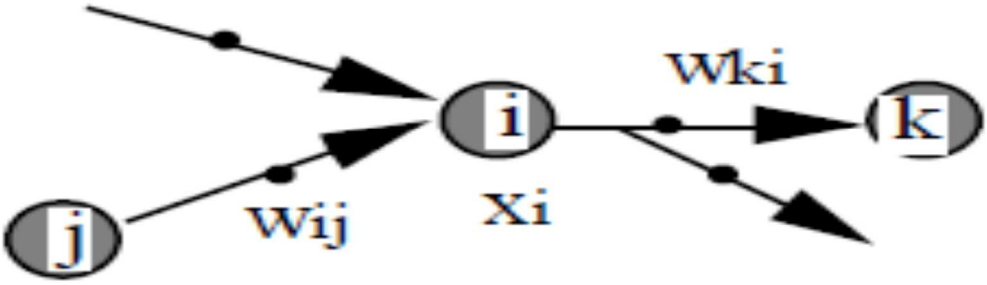


Figure IV.2. Structure d'un neurone artificiel. Pour le neurone d'indice i, les entrées sur celui-ci sont de poids w_{ij} alors que les connexions avals sont de poids w_{ki}.

IV.2.3. La fonction d'activation des neurones

Chaque neurone collecte les informations fournies par les neurones de la couche précédente avec lesquels il se trouve en relation et calcule alors son potentiel d'activation. Celui-ci est ensuite transformé par une fonction pour déterminer l'impulsion envoyée aux neurones de la couche suivante (potentiel de sortie), comme schématisé dans la figure IV.3.

L'activation d'un neurone est donnée par la somme des potentiels de sortie de ses prédécesseurs, pondérée par les poids synaptiques. Ce potentiel d'activation est ensuite transformé par une fonction afin de déterminer le potentiel de sortie [43].

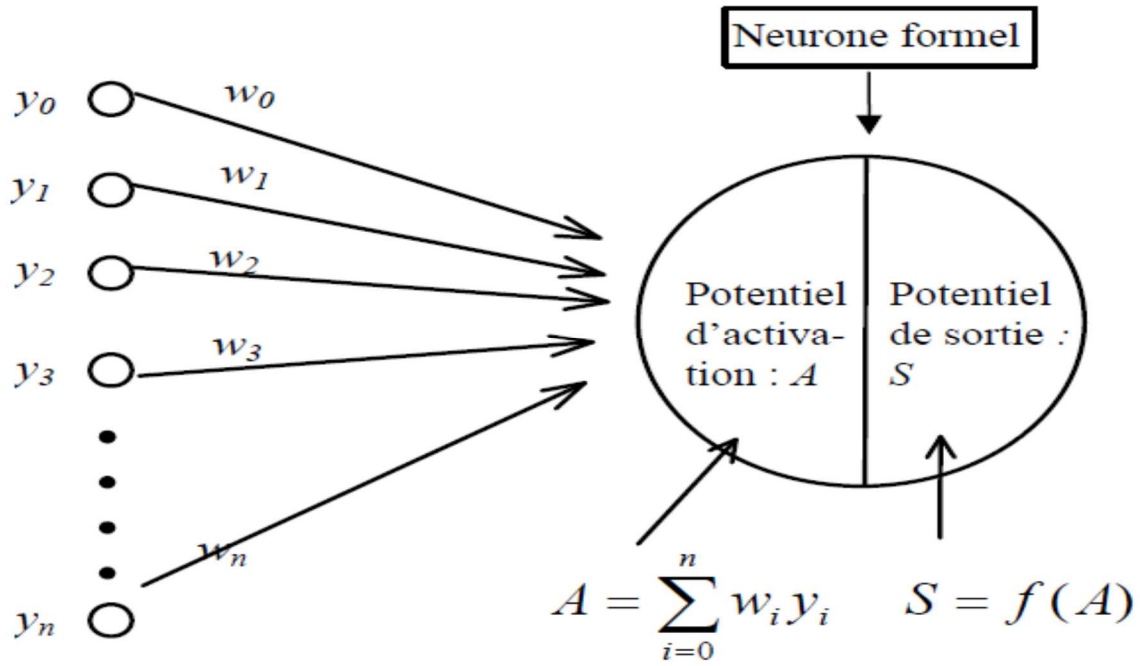


Figure IV.3. Schéma d'un neurone artificiel.

yi, désigne les sorties des prédécesseurs du neurone,
wi, désigne les poids synaptiques.

En principe, toute fonction choisie croissante et impaire peut être utilisée, mais le plus souvent on fait appel à des fonctions ramenant le résultat à l'intérieur de bornes prédéfinies. Plusieurs fonctions répondant à cet impératif peuvent être envisagées, mais la plus communément utilisée est la fonction sigmoïde représentée sur la figure IV.4, et dont l'expression analytique est la suivante :

$$S = f(A) = \frac{1}{1+e^{-A}} \qquad \text{(IV.1.)}$$

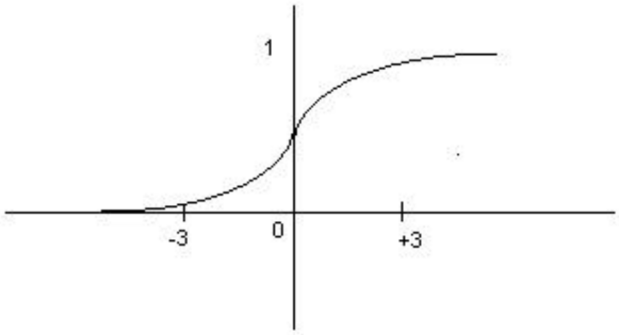


Figure IV.4. Fonction sigmoïde.

D'autres formes de fonctions sont parfois utilisées, telles que des fonctions binaires à seuil (figure IV.5 a et b) ou, linéaires à seuil (figure IV.5, c) ou encore des fonctions en escalier (figure IV.5, d).

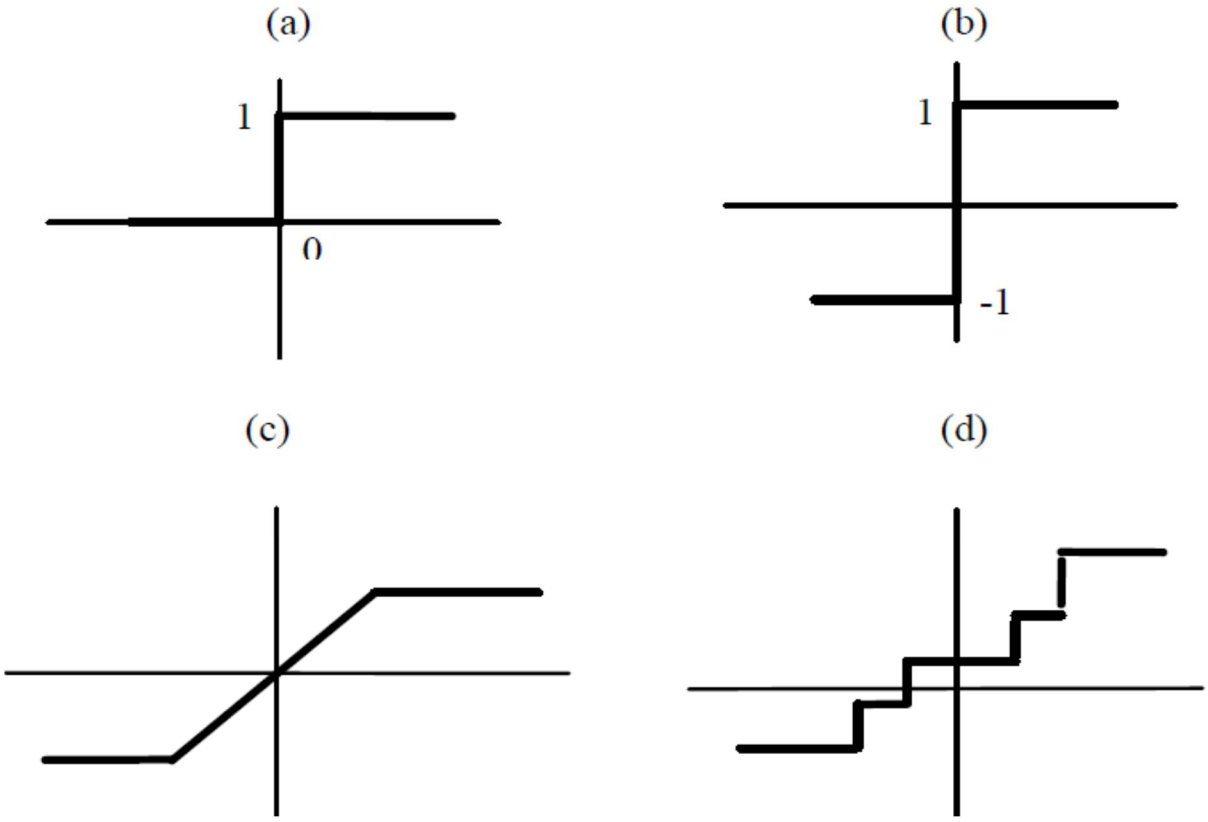


Figure IV.5. Exemples de fonctions d'activation.

On distingue deux phases. La première est habituellement le calcul de la somme pondérée des entrées (a) selon l'expression suivante :

$$a = \sum (w_i . e_i) \qquad \text{(IV.2)}$$

A partir de cette valeur, une fonction de transfert calcule la valeur de l'état du neurone. C'est cette valeur qui sera transmise aux neurones avals. Il existe de nombreuses formes possibles pour la fonction de transfert. Les plus courantes sont présentées sur la figure IV.6. On remarquera qu'à la différence des neurones biologiques dont l'état est binaire, la plupart des fonctions de transfert sont des fonctions continues, offrant une infinité de valeurs possibles comprises dans l'intervalle [0,+1] (ou [-1, +1]).

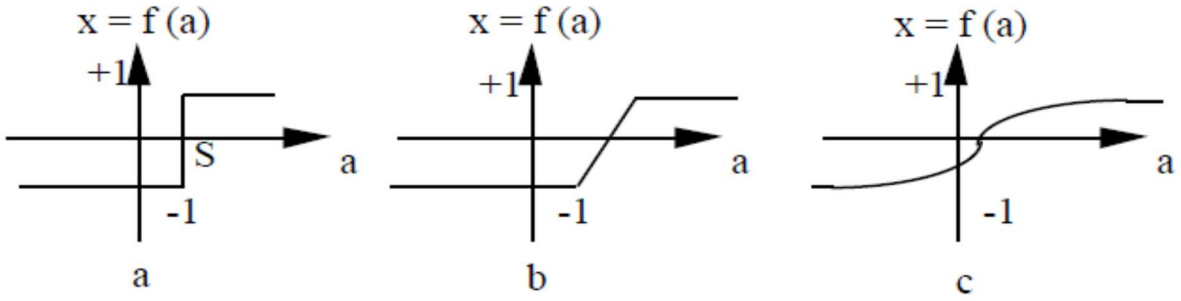


Figure IV.6. Différents types de fonctions de transfert pour le neurone artificiel, a : fonction à seuil (S, la valeur du seuil), b : linéaire par morceaux, c : sigmoïde.

Nous constatons que les équations décrivant le comportement des neurones artificiels n'introduisent pas la notion de temps. En effet, et c'est le cas pour la plupart des modèles actuels de réseaux de neurones, nous avons affaire à des modèles à temps discret, synchrone, dont le comportement des composants ne varie pas dans le temps [44].

IV.2.4. Variables descriptives

Ces variables décrivent l'état du système. Dans le cas des réseaux de neurones qui sont des systèmes non autonomes, un sous-ensemble des variables descriptives est constitué par les variables d'entrée, variables dont la valeur est déterminée extérieurement au modèle.

IV.2.5. Structure d'interconnexion

Les connexions entre les neurones qui composent le réseau décrivent la topologie du modèle. Cette topologie peut être quelconque, mais le plus souvent il est possible de distinguer une certaine régularité.

Réseau multicouche (au singulier) : les neurones sont arrangés par couche. Il n'y a pas de connexion entre neurones d'une même couche et les connexions ne se font qu'avec les neurones des couches avales (figure IV.7). Habituellement, chaque neurone d'une couche est connecté à tous les neurones de la couche suivante et celle-ci seulement. Ceci nous permet d'introduire la notion de sens de parcours de l'information (de l'activation) au sein d'un réseau et donc définir les concepts de neurone d'entrée, neurone de sortie. Par extension, on appelle couche d'entrée l'ensemble des neurones d'entrée, couche de sortie l'ensemble des neurones de sortie. Les couches intermédiaires n'ayant aucun contact avec l'extérieur sont appelés couches cachées [45].

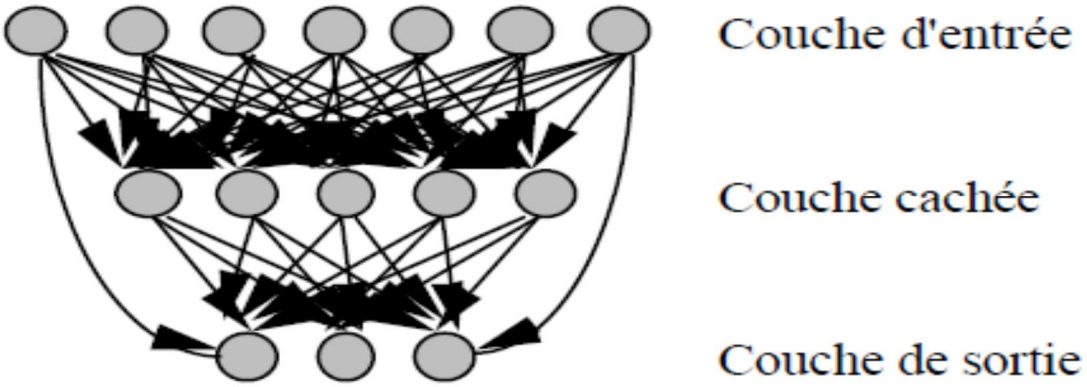


Figure IV.7. Définition des couches d'un réseau multicouche.

Réseau à connexions locales : Il s'agit d'une structure multicouche, mais qui, à l'image de la rétine, conserve une certaine topologie. Chaque neurone entretien des relations avec un nombre réduit et localisé de neurones de la couche avale (figure IV.8). Les connexions sont donc moins nombreuses que dans le cas d'un réseau multicouche classique.

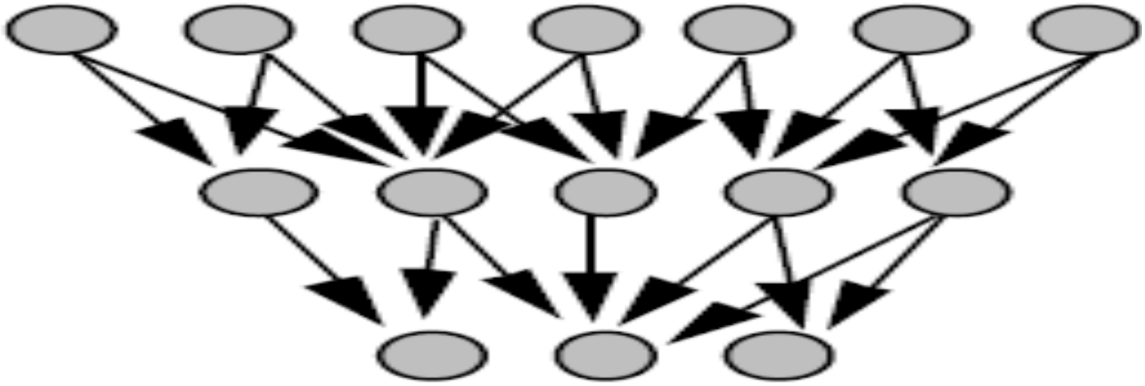


Figure IV.8. Réseau à connexions locales.

Réseau à connexions récurrentes : les connexions récurrentes ramènent l'information en arrière par rapport au sens de propagation défini dans un réseau multicouche. Ces connexions sont le plus souvent locales (figure IV.9).

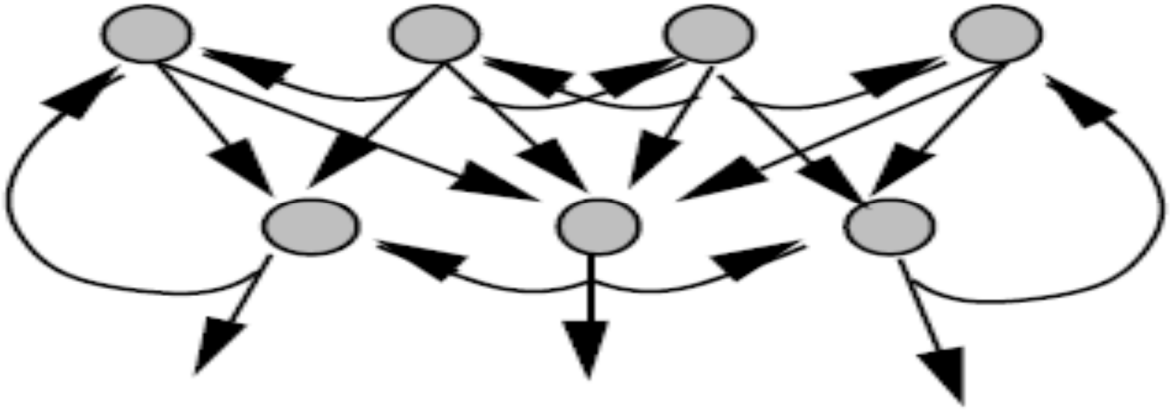


Figure IV.9. Réseau à connexions récurrentes.

Réseau à connexion complète : c'est la structure d'interconnexion la plus générale (figure IV.10). Chaque neurone est connecté à tous les neurones du réseau (et à lui-même).

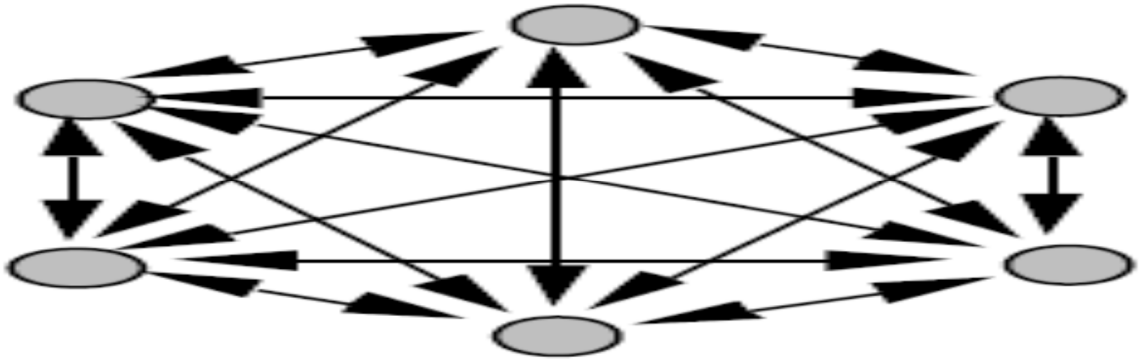


Figure IV.10. Réseau à connexion complète.

Il existe de nombreuse autres topologies possibles, mais elles n'ont pas eu à ce jour la notoriété des quelques-unes que nous avons décrites ici.

IV.2.6. Perceptron

Avant d'aborder le comportement collectif d'un ensemble de neurones, nous allons présenter le Perceptron (un seul neurone) en phase d'utilisation. L'apprentissage ayant été réalisé, les poids sont fixes. Le neurone de la figure IV.11 réalise une simple somme pondérée de ses entrées, compare une valeur de seuil, et fourni une réponse binaire en sortie. Par exemple, on peut interpréter sa décision comme classe 1 si la valeur de x est +1 et classe 2 si la valeur de x est -1.

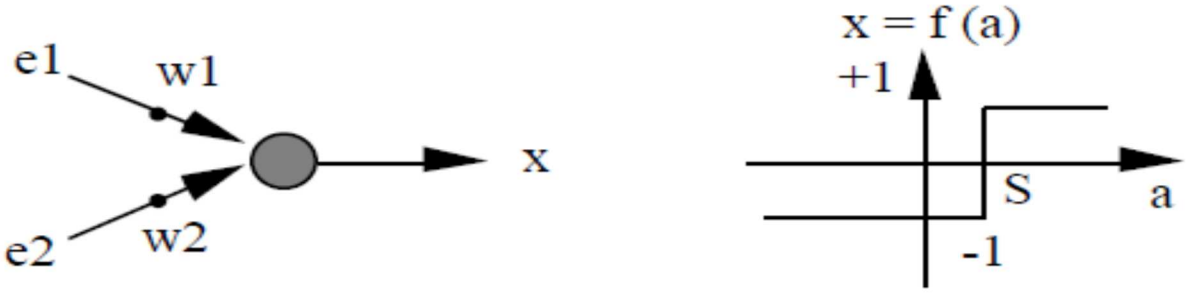


Figure IV.11. Le Perceptron : structure et comportement. Les connexions des deux entrées e_1 et e_2 au neurone sont pondérées par les poids w_1 et w_2. La valeur de sortie du neurone est notée x. Elle est obtenue après somme pondérée des entrées (a) et comparaison à une valeur de seuil S.

IV.2.7. Apprentissage

L'apprentissage est vraisemblablement la propriété la plus intéressante des réseaux neuronaux. Elle ne concerne cependant pas tous les modèles, mais les plus utilisés.

Définition :

L'apprentissage est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré. L'apprentissage neuronal fait appel à des exemples de comportement.

Dans le cas des réseaux de neurones artificiels, on ajoute souvent à la description du modèle l'algorithme d'apprentissage. Le modèle sans apprentissage présente en effet peu d'intérêt. Dans la majorité des algorithmes actuels, les variables modifiées pendant l'apprentissage sont les poids des connexions. L'apprentissage est la modification des poids du réseau dans l'optique d'accorder la réponse du réseau aux exemples et à l'expérience. Il est souvent impossible de décider a priori des valeurs des poids des connexions d'un réseau pour une application donnée. A l'issu de l'apprentissage, les poids sont fixés : c'est alors la phase d'utilisation. Certains modèles de réseaux sont improprement dénommés à apprentissage permanent. Dans ce cas il est vrai que l'apprentissage ne s'arrête jamais, cependant on peut toujours distinguer une phase d'apprentissage (en fait de remise à jour du comportement) et une phase d'utilisation. Cette technique permet de conserver au réseau un comportement adapté malgré les fluctuations dans les données d'entrées.

Au niveau des algorithmes d'apprentissage, il a été défini deux grandes classes selon que l'apprentissage est dit supervisé ou non supervisé. Cette distinction repose sur la forme des exemples d'apprentissage. Dans le cas de l'apprentissage supervisé, les exemples sont des couples (Entrée, Sortie associée) alors que l'on ne dispose que des valeurs (Entrée) pour l'apprentissage non supervisé. Remarquons cependant que les modèles à apprentissage non supervisé nécessitent avant la phase d'utilisation une étape de labélisation effectuée par l'opérateur, qui n'est pas autre chose qu'une part de supervision.

IV.2.7.1. L'algorithme d'apprentissage

Le réseau se paramètre à l'aide d'un échantillon d'apprentissage, qui associe des formes présentées à un résultat désiré. C'est l'algorithme d'apprentissage qui ajuste les poids synaptiques en cherchant à minimiser une fonction de coût.

C'est sans aucun doute la découverte de l'algorithme de rétro-propagation du gradient de l'erreur par Rumelhart, Hinton et Williams en 1986, qui a marqué le pas décisif dans l'application des réseaux de neurones artificiels. C'est encore aujourd'hui le plus utilisé dans la plupart des applications.

La fonction de coût, E minimisée par cet algorithme n'est autre que la somme des carrés des erreurs produites par le réseau eu égard au résultat désiré.

Ainsi, si l'échantillon d'apprentissage comporte s exemples décrits chacun par un vecteur d'entrée de dimension m, $X_i = (x_{i,1}, x_{i,2}, \dots\dots, x_{i,m})$ et un vecteur de sortie de dimension n, $D_i = (d_{i,1}, d_{i,2}, \dots\dots, d_{i,n})$ la fonction à minimiser s'écrit :

$$E = \sum_{i=1}^{s} \sum_{j=1}^{n} \frac{(y_{i,j} - d_{i,j})^2}{2} \qquad \text{(IV.3)}$$

$y_{i,j}$ représente les sorties produites par le réseau alors que $d_{i,j}$ représente les sorties désirées.

Le calcul de la quantité E, permet de déterminer dans un deuxième temps, la variation des poids synaptiques, $\Delta w_{i,j}$ de la façon suivante :

$$\Delta w_{i,j} = -\frac{\partial E}{\partial w_{i,j}} \varepsilon \qquad \text{(IV.4)}$$

$0 < \varepsilon < 1$, est un paramètre permettant de contrôler la rapidité de convergence de l'algorithme.

Après avoir initialisé les poids synaptiques de façon aléatoire, un premier calcul est effectué dans l'ordre topologique du réseau. La sortie obtenue est alors comparée à la sortie désirée et la fonction de coût, E est évalué.

L'erreur totale commise par le système est ensuite rétro propagée de la couche de sortie vers la couche d'entrée et les poids synaptiques sont modifiés selon la formule ci-dessus, ce qui permet d'initier un nouveau calcul. Le processus se poursuit ainsi jusqu'à ce que l'utilisateur intervienne pour y mettre fin ou jusqu'à ce qu'une valeur de la fonction de coût, fixée a priori soit atteinte.

IV.2.7.2. La loi de Hebb, un exemple d'apprentissage non supervisé

La loi de Hebb (1949) s'applique aux connexions entre neurones, comme le représente la figure IV.12.

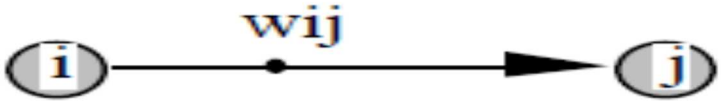


Figure IV.12. i le neurone amont, j le neurone aval et w_{ij}le poids de la connexion.

Elle s'exprime de la façon suivante

"Si 2 cellules sont activées en même temps alors la force de la connexion augmente".

La modification de poids dépend de la coactivation des neurones présynaptiques et post synaptique, ainsi que le montre la table 1. x_i et x_j sont respectivement les valeurs d'activation des neurones i et j, ∂w_{ij}(dérivée partielle du poids) correspond à la modification de poids réalisée.

Table IV.1. La loi de Hebb.

x_i	x_j	∂w_{ij}
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**+**

La loi de Hebb peut être modélisée par les équations suivantes (w(t+1) est le nouveau poids, $w_{ij(t)}$ l'ancien) :

$$w_{ij}(t+1) = w_{ij}(t) + \partial w_{ij}(t) \quad \text{(IV.5)}$$

$\partial w_{ij}(t) = x_i . x_j$ (la coactivité est modélisée comme le produit des deux valeurs d'activation)

L'algorithme d'apprentissage modifie de façon itérative (petit à petit) les poids pour adapter la réponse obtenue à la réponse désirée. Il s'agit en fait de modifier les poids lorsqu'il y a erreur seulement.

1/ Initialisation des poids et du seuil S à des valeurs (petites) choisies au hasard.

2/ Présentation d'une entrée E_1= (e_1, ... e_n) de la base d'apprentissage.

3/ Calcul de la sortie obtenue x pour cette entrée :

$a = \sum (w_i . e_i)$-S (la valeur de seuil est introduite ici dans le calcul de la somme pondérée)

x = signe (a) (si a > 0 alors x = +1 sinon a ≤ 0 alors x = -1)

4/ Si la sortie x est différente de la sortie désirée d_1 pour cet exemple d'entrée E_1 alors modification des poids (μ est une constante positive, qui spécifie le pas de modification des poids) :

$$w_{ij}(t+1) = w_{ij}(t) + \mu(x_i . x_j) \quad \text{(IV.6)}$$

5/ Tant que tous les exemples de la base d'apprentissage ne sont pas traités correctement (i.e. modification des poids), retour à l'étape 2.

IV.2.8 Présentation des réseaux de neurones artificiels habituellement utilisés.

Les réseaux de neurones artificiels s'inspirent du fonctionnement du cerveau et du système nerveux ou, plus exactement, de la représentation que l'on s'en fait.

Il existe deux grandes catégories de réseaux :

- Les réseaux à apprentissage supervisé dans lesquels le système apprend à reconnaître des formes à partir d'un échantillon d'apprentissage qui associe les modalités portées par des variables censées caractériser une forme, et la forme elle-même. Sur cet échantillon les résultats correspondant aux divers ensembles d'informations données au système sont connus. C'est à partir de là que le système se paramètre ;
- Les réseaux à apprentissage non supervisé, qui sont utilisés lorsque l'utilisateur du réseau n'est pas en mesure de présenter au système un échantillon mettant en regard une somme d'informations et la forme qu'elle est censée représenter. Le réseau s'auto-organise de façon à découvrir des formes récurrentes dans les informations qu'il reçoit, mais il le fait sans aide extérieure, contrairement aux réseaux à apprentissage supervisé. Le plus connu de cette catégorie de réseaux est celui de Kohonen (1984) Dans les différents domaines, c'est le premier type de réseau que l'on utilise, le plus généralement, appelé réseau à couches.

IV.2.9. La mise en œuvre d'un réseau de neurones artificiels

Après avoir déterminé l'architecture du réseau, sa mise en œuvre nécessite trois échantillons de données. Le premier, destiné à l'apprentissage servira au paramétrage, le second servira à la validation et le troisième sera un échantillon test destiné à évaluer les capacités de généralisation du réseau.

Durant la phase d'apprentissage, l'erreur commise diminue, jusqu'à tendre asymptotiquement vers zéro si l'architecture du réseau a été correctement choisie. Mais il convient de remarquer que plus l'erreur est faible, c'est-à-dire plus le réseau apprend à reconnaître les formes qui lui sont présentées, moins il risque d'être capable de généralisation. En effet, au-delà d'un certain point, si le réseau reconnaît de mieux en mieux les formes appartenant à l'échantillon d'apprentissage, il donne de piètres résultats sur un échantillon inconnu. On dit alors qu'il se spécialise. Or le but d'un tel système est bien de reconnaître des formes inconnues jusqu'alors. Il convient donc d'arrêter la phase d'apprentissage lorsque le réseau semble donner les

meilleurs résultats en généralisation. C'est l'échantillon de validation qui permet de suivre la capacité du système à généraliser.

Typiquement, les courbes décrivant l'erreur commise sur l'échantillon d'apprentissage et sur l'échantillon de validation durant la phase d'apprentissage se présentent comme l'indique la figure IV.13.

Enfin, le réseau étant supposé correctement paramétré, il est appliqué à l'échantillon test de façon à mesurer sa capacité à reconnaître des formes qui n'ont jamais été vues auparavant.

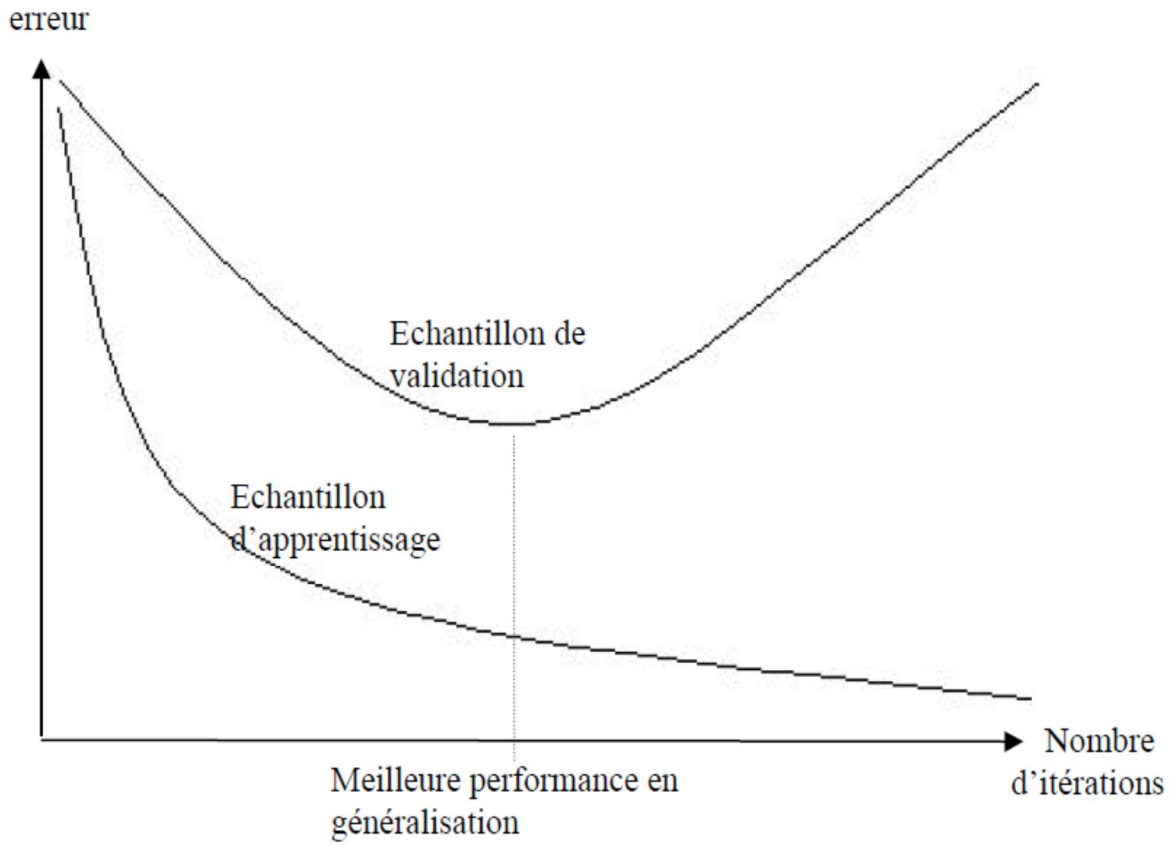


Figure IV.13. Evolution des courbes d'erreur durant la phase d'apprentissage.

IV.2.10. Application des réseaux de neurones

Les réseaux de neurones servent aujourd'hui à toutes sortes d'applications dans divers domaines. Par exemple, on a développé un autopilote pour avion, ou encore un système de guidage pour automobile, on a conçu des systèmes de lecture automatique de chèques bancaires et d'adresses postales, on produit des systèmes de traitement du signal pour différentes applications militaires, un système pour la synthèse de la parole, des réseaux sont aussi utilisés pour bâtir des systèmes de vision par ordinateur. Pour faire des prévisions sur les marchés monétaires, pour évaluer le risque financier ou en assurance, pour différents processus manufacturiers, pour le diagnostic médical, pour l'exploration pétrolière ou gazière, en robotique, en télécommunication, et j'en passe ! Bref, les réseaux de neurones ont aujourd'hui un impact considérable et, il y a fort à parier, que leur importance ira grandissant dans le futur.

IV.3. CHOIX DE l'ARCHITECTURE

La fonction d'activation choisie est la fonction sigmoïde. La détermination de la taille du réseau et le nombre de neurones de chaque couche fait partie du choix judicieux pour optimiser le comportement du réseau. Il a et établi qu'un réseau a une seule couche cachée, peut faire l'approximation de n'importe quelle fonction [46].

Concernant le nombre de neurones pour la couche cachée, un nombre trop grand permet d'apprendre les exemples mais diminuant ses performances de généralisation (on parle alors de sur-apprentissage), et une couche cachée ayant peu de neurones ne permet pas au réseau de faire correctement la tache demandée (on parle alors de sous-apprentissages).

Pour l'apprentissage, on utilisera l'algorithme de retro-propagation du gradient, qui va agir sur les poids et les modifier de manière à se rapprocher de plus en plus du comportement désiré pour accomplir le suivi de cible.

IV.2.1. Paramètre et architecture du réseau

Afin que notre réseau converge et apprenne de ses exemples, on procède à une série de tests, pour fixer les paramètres du MLP à savoir le pas d'apprentissage, le nombre d'itération maximal et le nombre de neurones dans la couche cachée, le but étant de réduire l'erreur quadratique (nommée MSE) en sortie. Le corpus d'apprentissage doit être aussi pris en considération.

IV.2.2. Corpus d'apprentissage

Pour notre apprentissage, nous avons eu recours à un corpus d'apprentissage contenant 644 exemples. Ces derniers ont été capturés par un superviseur humain sous Gazebo, la manière de procéder était de placer le quadri-rotor dans une position donnée par rapport à la cible et de le guider manuellement vers la cible de façon qu'il y soit en face. 9 positions ont été choisies à savoir la combinaison des repères suivant haut, bas, droite, gauche donnant comme position de départ proportionnel à la cible (haut, en haut à gauche, à droite, au milieux, en bas à droite etc..).

Le partitionnement des données récoltées a été fait selon un ratio de 0.8, c'est à dire 80% pour l'apprentissage et 20% pour le test de l'apprentissage.

IV.2.3 Paramétrage de notre réseau

Comme indiqué précédemment, pour avoir un meilleur apprentissage (un taux d'erreur MSE plus petit) nous jouerons sur le pas d'apprentissage, le nombre d'itérations maximal et le nombre de neurones dans la couche cachée. Nous avons alors testé 2952 combinaisons possibles de la façon suivante :

- Pour un nombre d'itération maximale de 50 à 600 à intervalles de 50 faire :
- Pour un nombre de neurones dans la couche cachée de 5 à 120 faire :
- Pour une valeur de pas d'apprentissage appartenant à 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 faire :
- Faire l'apprentissage du PMC avec le paramétrage courant et retourner l'erreur quadratique MSE
- Sauvegarder les informations pour plus de comparaison (graphe d'apprentissage, exemples, median..)

Le choix de la meilleure architecture/paramétrage sera déterminé en grande partie par l'erreur quadratique sur les données de tests, mais on jugera aussi de la convergence de la courbe

d'apprentissage (courbe d'apprentissage et de teste) pour vérifier les cas de minimaux locaux et de fluctuations sur la solution finale, on utilisera l'erreur moyenne pour valider notre choix.

IV.3. CONCEPTION DU CONTROLEUR A BASE DE RESEUX NEURONAL

Le système de suivi des cibles présenté dans cette section a été développé en utilisant ROS (Robot Operating System). L'algorithme ANN a été implanté en tant que nœud ROS, qui reçoit des informations d'état des capteurs de la caméra, forme le réseau neuronal en utilisant l'état d'entrée et envoie des commandes de vitesse et de translation à un Quadrotor AR.Drone.

L'environnement de simulation Gazebo a été utilisé pour évaluer le comportement de l'algorithme ANN dans une simulation physique. Le package RVIZ a été utilisé pour visualiser l'état actuel de l'environnement pendant la simulation, y compris la détection des marqueurs.

L'objectif de cette partie étant la reconnaissance d'objet, le problème de vision a été simplifié en utilisant des balises AR au lieu des fonctions visuelles. Les balises AR pourraient être placées dans le monde de la simulation et suivies de manière fiable à l'aide du package ROS ar_track_alvar. Ce paquet peut fournir une estimation de pose fiable pour plusieurs étiquettes détectées ainsi que leurs identifiants. Les états de détection des étiquettes ont été utilisés comme états d'entrée du réseau neuronal. Pour reconnaître la bordure à chaque marqueur, un algorithme de détection de marqueur est requis. ar_track_alvar a été développé et est largement utilisé comme méthode représentative pour le suivi basé sur les marqueurs. La méthode proposée est simple et nécessite une faible puissance de calcul. Après avoir obtenu les données, nous avons développé un système assurant la détection et le suivi des objets. La figure IV.14 ci-dessous illustre le processus de développement du système réalisé. Les coordonnées de la caméra par rapport à la cible sont détaillées en annexe 2.

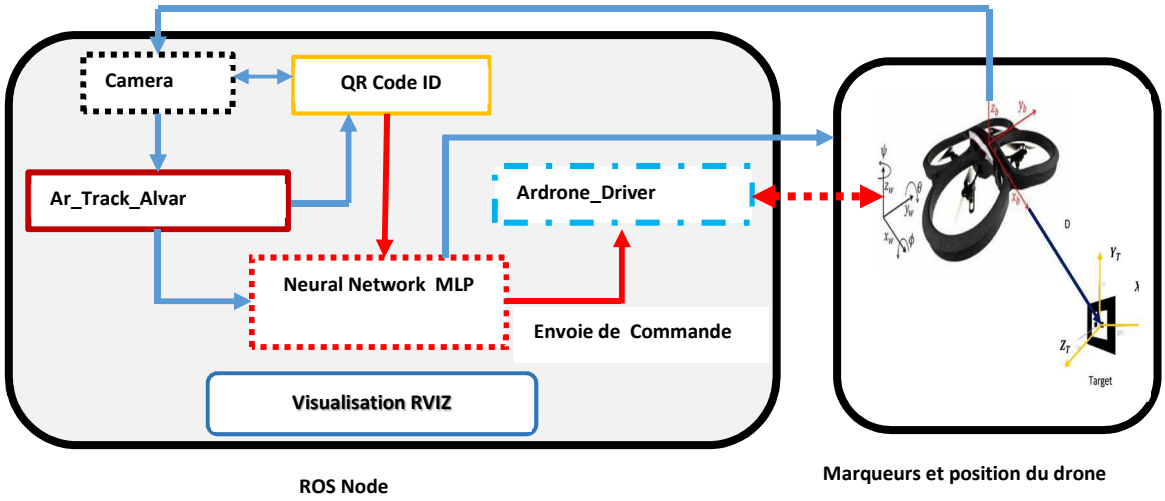


Figure IV.14. Structure de l'implémentation logicielle du système de contrôle distribué basée sur ROS.

La plate-forme fonctionne sous le système d'exploitation ROS. Ce dernier est un framework open source pour le développement de systèmes de contrôle pour la robotique. La structure facilite les communications entre les différents modules en utilisant une interface centralisée de

transmission de messages. Plusieurs modules peuvent être utilisés comme composants pour un système plus vaste.

IV.3.1. Conception du réseau neuronal

La conception d'un réseau de neurones implique la sélection de son modèle, de son architecture, de son algorithme d'apprentissage et de ses fonctions d'activation pour ses neurones en fonction des besoins de l'application. L'objectif de notre application est de localiser et de suivre une cible spécifique (tag).

Pour notre travail, nous utilisons un type de réseau neuronal MLP (Multilayer Perceptron). MLP et les nombreux autres réseaux neuronaux apprennent à utiliser un algorithme appelé back propagation [47 - 51]. Avec la propagation arrière, les données d'entrée sont présentées de manière répétée au réseau neuronal. A chaque présentation, la sortie du réseau neuronal est comparée à la sortie souhaitée et une erreur est calculée. Cette erreur est ensuite renvoyée au réseau neuronal et utilisée pour ajuster les poids de telle sorte que l'erreur diminue à chaque itération de l'opération et que le modèle neuronal se rapproche de plus en plus de la sortie souhaitée. Ainsi, pour notre travail, un BPNN est sélectionné pour l'application en question, et il se compose de trois couches : une couche d'entrée (nœuds de sources), une couche cachée (avec une fonction d'activation sigmoïde) et une couche de sortie.

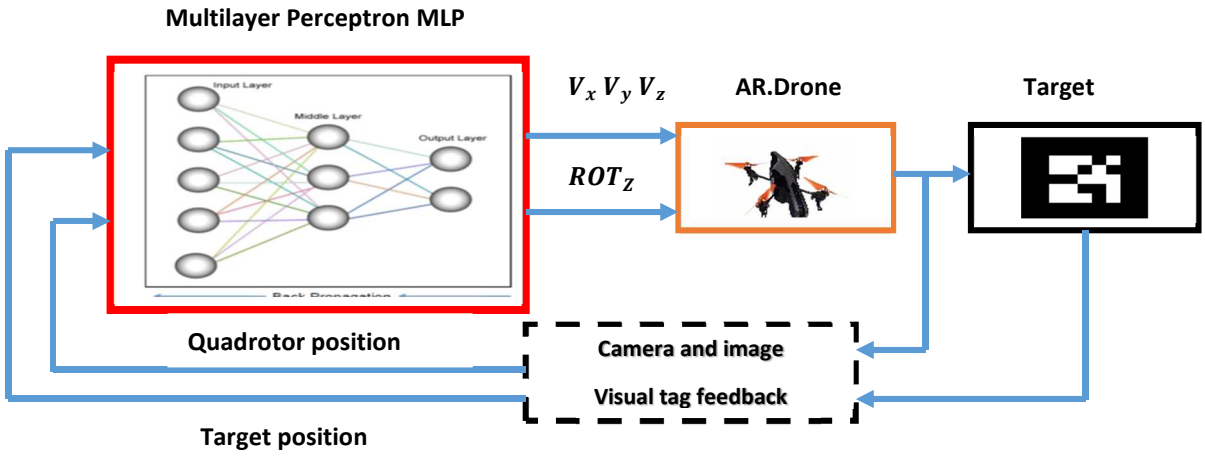


Figure IV.15. Architecture du réseau neuronal proposé.

Pour notre application, l'algorithme d'apprentissage comporte deux phases.

- Un modèle d'entrée d'apprentissage est présenté à la couche d'entrée du réseau. Le réseau propage le motif d'entrée de couche en couche jusqu'à ce que la couche de sortie génère le motif de sortie.

- Si ce modèle est différent de la sortie souhaitée, une erreur est calculée puis propagée à travers le réseau depuis la couche de sortie vers la couche d'entrée. Les poids sont modifiés à mesure que l'erreur se propage.

IV.3.2. Couche d'entrée

Les couches d'entrée d'un réseau neuronal sont déterminées par les caractéristiques des entrées de l'application [52 – 55]. Dans notre travail, la couche d'entrée est l'information reçue par la caméra frontale du drone et représente la position euclidienne de la cible par rapport au drone. Ces données sont décrites par les valeurs suivantes :

- Pose.Position X : Représente l'angle d'orientation de la cible par rapport au drone. Cette valeur est zéro si la cible est sur la ligne x du drone. Elle est négative si la cible se trouve à gauche de la ligne x du drone et elle est positive si la cible est à droite de la ligne x du drone.

- Pose.Position Y : Représente la hauteur de la cible par rapport au drone. Cette valeur est zéro si la cible est à la même hauteur que le drone. Elle est négative si la cible est au-dessus du drone et elle est positive si la cible est en dessous du drone.

- Pose.Position Z : Représente la distance qui sépare la cible du drone. Cette valeur est toujours positive. Sa variation dépend des mouvements du drone.

La figure IV.16 représente les points de référence euclidiens du drone par rapport à la cible.

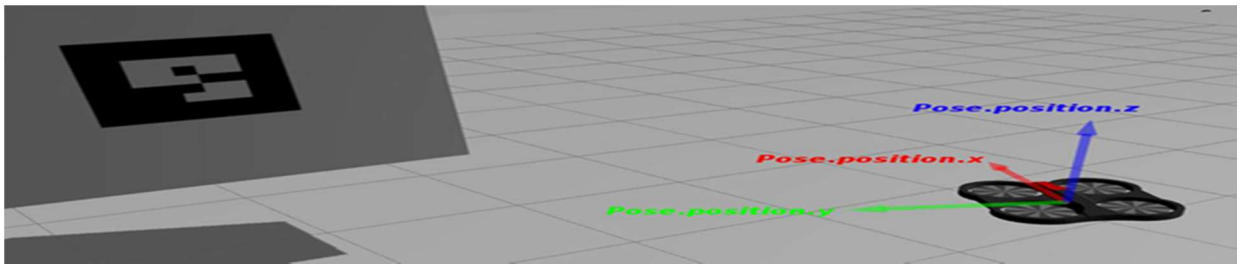


Figure IV.16. Les repères euclidiens du drone.

- Pose.Orientation X : Représente l'orientation de la cible par rapport au drone, le long de l'axe des x en quaternion.

- Pose.Orientation Y : Représente l'orientation de la cible par rapport au drone, le long de l'axe y d'un quaternion.

La figure IV.17 montre les points de référence de rotation du drone.

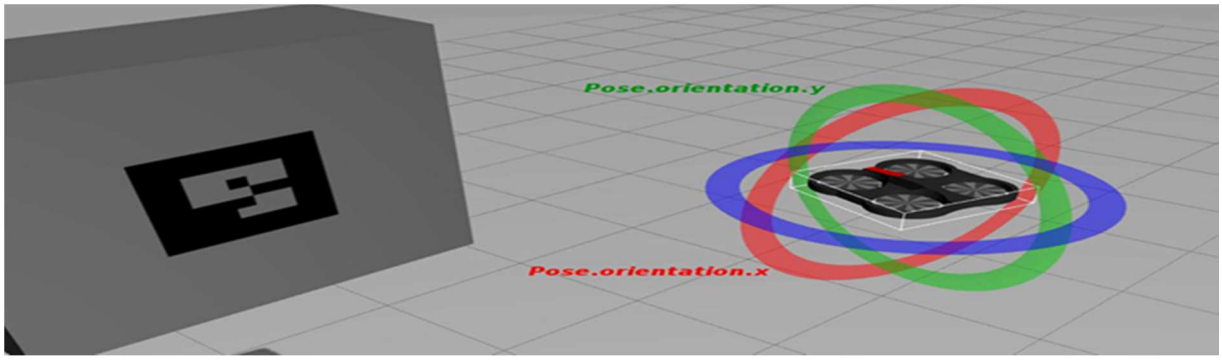


Figure IV.17. Points de référence en rotation du drone.

IV.3.3. Couche de sortie

La couche de sortie du réseau est conçue en fonction des besoins de l'application. La couche de sortie est l'information relative au mouvement du drone. Ces données sont représentées par les variables suivantes :

- ***Vitesse linéaire X*** (V_X)***:*** Représente la vitesse linéaire du drone le long de l'axe des x. Cela permet au drone d'avancer ou de reculer.
- ***Vitesse linéaire Y*** (V_Y) ***:*** Représente la vitesse linéaire du drone le long de l'axe des y. Il permet au drone d'effectuer un mouvement latéral vers la droite ou vers la gauche.

- ***Vitesse linéaire Z*** $(\boldsymbol{V_Z})$ ***:*** Représente la vitesse linéaire du drone le long de l'axe des z. Cela permet au drone de monter ou de descendre.
- ***Vitesse angulaire Z*** $(\boldsymbol{Rot_Z})$***:*** Représente la rotation du drone le long de l'axe z dans un quaternion. Cela permet au drone de tourner à droite ou à gauche.

IV.3.4. La fonction d'activation

En général, une fonction non linéaire est utilisée pour la couche cachée. Dans notre travail, nous utilisons la fonction sigmoïde donnée par l'équation (IV.1), car elle est continue, non linéaire et différentiable.

$$\emptyset_{(Z)} = \frac{1}{1+e^{-Z}} \qquad \text{(IV.7)}$$

IV.3.5. Formation du réseau neuronal

Dans notre travail, l'algorithme de formation Back Propagation est utilisé pour l'apprentissage en réseau neuronal supervisé. Pendant l'entraînement, le réseau est présenté avec une grande quantité de modèles d'entrée (jeu d'apprentissage). Les sorties correspondantes souhaitées sont ensuite comparées aux sorties du réseau réel. L'erreur entre la réponse souhaitée et la réponse réelle est utilisée pour mettre à jour les poids des interconnexions du réseau. Cette mise à jour est effectuée après chaque présentation du modèle. Le processus d'entraînement se poursuit à plusieurs époques, jusqu'à ce qu'une petite erreur satisfaisante soit générée. Cette erreur est utilisée pour évaluer la capacité du réseau à généraliser. Habituellement, l'ensemble de formation et / ou l'architecture doivent être évalués à ce stade. La règle de modification des poids ou de l'apprentissage utilisé est celle de Widrow et Ho également appelée règle du delta.

Pour résoudre le problème du minimum local, nous avons utilisé deux approches. La première consiste à refaire plusieurs fois l'apprentissage de notre réseau neuronal en modifiant les poids initiaux ; la deuxième approche consiste à varier l'étape d'apprentissage.

Pour notre apprentissage, nous avons utilisé une méthode d'apprentissage contenant 644 exemples. Celles-ci ont été obtenues par simulation sous ROS Gazebo, la manière de procéder était de placer le quadrotor par rapport à la cible et de le guider manuellement vers la cible pour qu'il soit devant elle.

Le partitionnement des données collectées a été effectué selon un ratio de 0,8, soit 80 % pour l'apprentissage et 20 % pour le test d'apprentissage.

Dans notre application, nous avons commencé avec une étape d'apprentissage égale à 0,1 et nous avons fait varier cette valeur jusqu'à trouver un bon compromis entre vitesse et convergence. Le tableau IV.2 montre les configurations possibles pour le processus d'apprentissage.

Notons que l'apprentissage est bon si le taux d'erreur est inférieur à 0,020. Dans les résultats de l'entraînement sur les réseaux neuronaux, le MSE donnerait des valeurs différentes, nous avons donc effectué neuf exercices expérimentaux pour trouver le résultat optimal.

Tableau IV.2. Les différentes configurations pour le processus d'apprentissage.

Expérience	Nombre maximum d'itération	L'étape d'apprentissage	Nombre de couche cachés
Configuration 1	100	0.1	120
Configuration 2	600	0.01	80
Configuration 3	100	0.01	105
Configuration 4	400	0.01	120
Configuration 5	500	0.7	50
Configuration 6	400	0.8	50
Configuration 7	250	0.2	80
Configuration 8	350	0.3	25
Configuration 9	100	0.6	85

La figure IV.18 montre les résultats choisis pour le processus d'apprentissage. Ils sont présentés par une courbe d'apprentissage (en bleu) et une courbe de test d'apprentissage (en rouge).

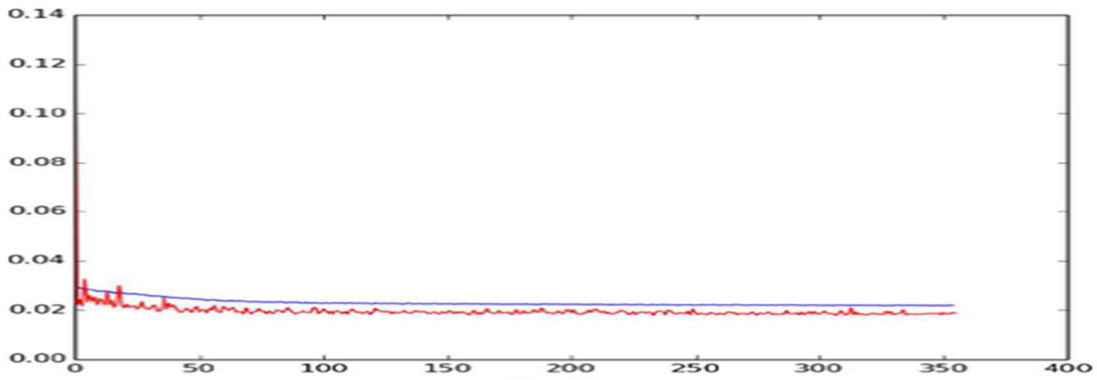


Figure IV.18. Courbe d'erreur quadratique moyenne de l'entraînement de la configuration 8.

Pour la configuration (8), on constate que la courbe d'apprentissage du (MLP) converge avec succès vers une solution finale. La courbe de test converge également sans fluctuations. Pour notre application, nous avons sélectionné cette configuration.

Pour effectuer le processus de formation, le réseau neuronal nécessite certains paramètres, énumérés ci-dessous :

1- L'architecture du réseau neuronal est composée de 3 couches (entrée, couche cachée et couche de sortie).

2- La couche d'entrée a 5 neurones, celle cachée a 25 neurones et la sortie 4 neurones.

3- La fonction d'activation sigmoïde est utilisée.

4- L'initialisation du poids est réalisée avec Widrow et Ho.

5- 350 Epoque maximum.

6- Le taux d'apprentissage est égal à 0,3.

7- La cible d'erreur est égale à 0.02.

IV.4. RESULTATS DE SIMULATION

Pour le suivi de la cible, nous avons utilisé les informations de tag pour établir les manœuvres du quadrotor; l'algorithme de réseau neuronal envoie les commandes nécessaires au drone pour suivre la cible. La figure IV.19 montre la configuration des nœuds dans ROS pour effectuer le contrôle et le suivi. Dans cette figure, le nœud de couleur rouge représente le nœud du contrôleur de suivi de cible qui est le package 'NNcontroller'. En tant qu'entrées, le nœud prend une capture d'image calculée par rapport au centre de la cible détectée et au descripteur d'identité cible (ID) de la cible. Avec ces entrées, le nœud crée un vecteur de contrôle basé sur l'ID cible. ROS Gazebo (nœud de couleur bleue) a été utilisé pour simuler les flux d'images provenant de l'AR.Drone. Les scènes du simulateur ont été extraites à l'aide d'une caméra intégrée dans le "Tum_simulator". Le 'ar_track_alvar' (nœud de couleur verte) est capable de traiter les images et de fournir des informations pour le 'NNcontroller'.

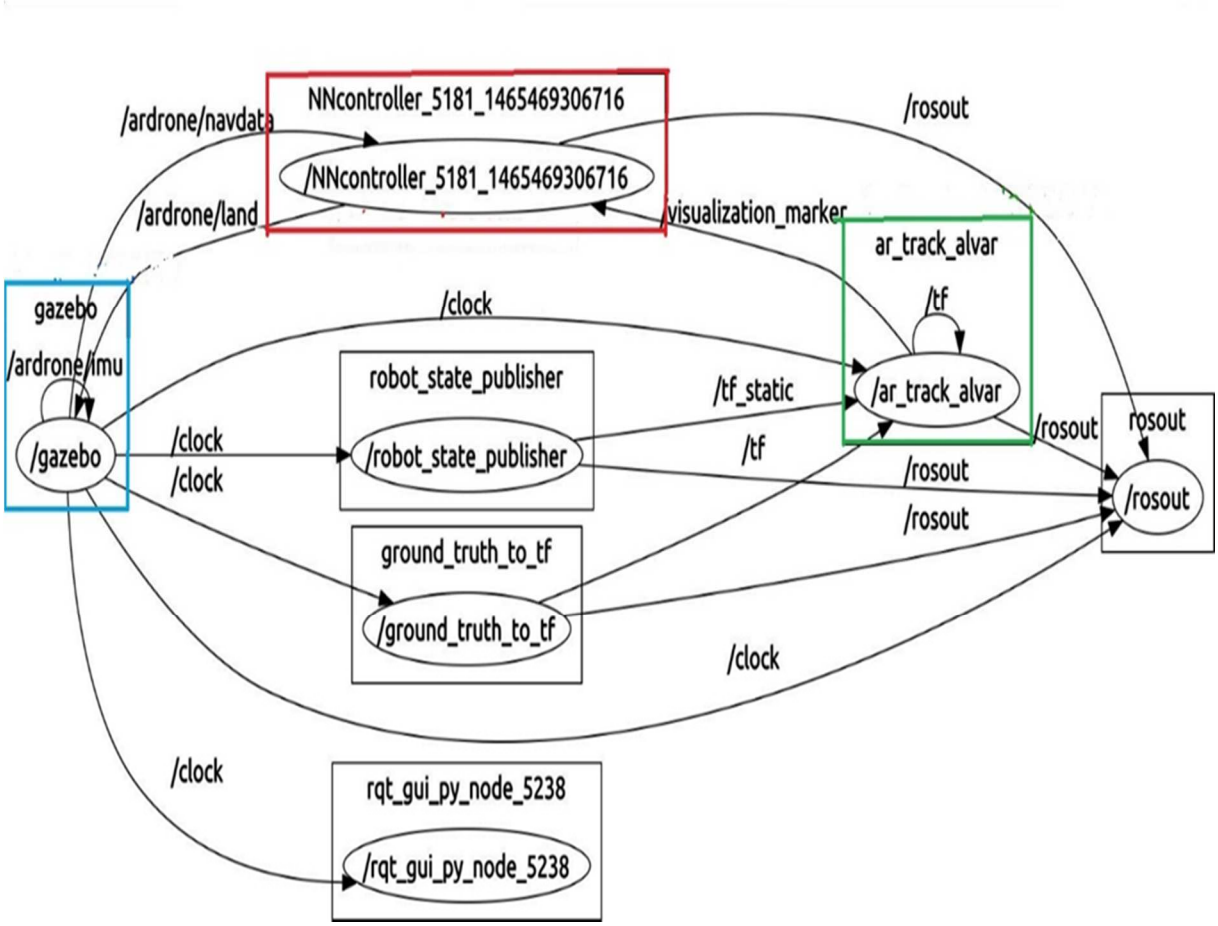


Figure IV.19. Interaction entre tous les processus de Gazebo et du réseau neuronal dans ROS lors des tests de simulation.

Le graphique précédent a été généré automatiquement par l'outil rxgraph, qui peut inspecter et surveiller tout graphique ROS au moment de l'exécution. Sa sortie restitue les nœuds sous forme d'ovales, les sujets sous forme de carrés et la connectivité sous forme d'arcs.

De nombreux tests de simulation ont été effectués pour évaluer l'efficacité du réseau de neurones dans différentes positions de la cible par rapport au quadrotor. Le but de ces tests de simulation est de prouver que le réseau neuronal développé répond favorablement à différentes situations.

IV.4.1. Premier test : le quadrotor est à droite de la cible

Dans ce test, le quadrotor est positionné à droite au même niveau que la cible comme indiqué dans la figure IV.20.

Figure IV.20. Résultat de simulation du premier test.

À partir des résultats du premier test, on note qu'après le décollage du drone il détecte la cible, et il effectue une rotation vers la gauche, puis se positionne au même niveau que la balise. Le quadrotor se dirige vers la cible en tournant à droite et à gauche pour ajuster sa trajectoire. Une fois le drone proche de la cible, il maintient une distance de 1 m de l'étiquette. Pendant toute la durée du vol, le drone utilise sa caméra frontale et lorsque la cible est visible il dessine un carré coloré violet. La figure IV.21 montre le chemin parcouru par le drone pour suivre la cible.

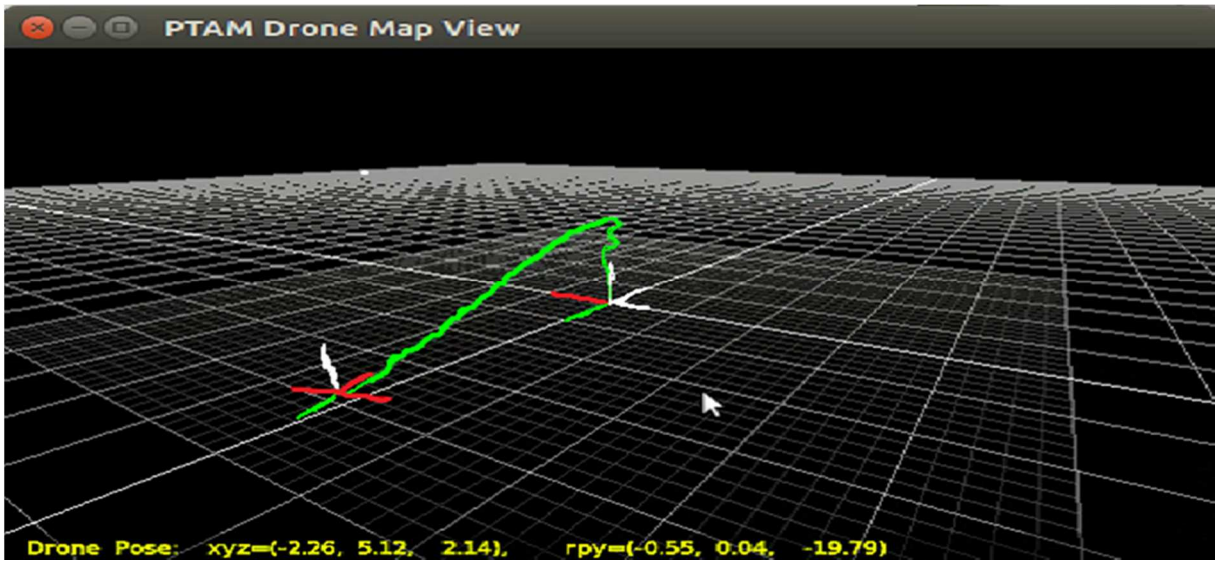
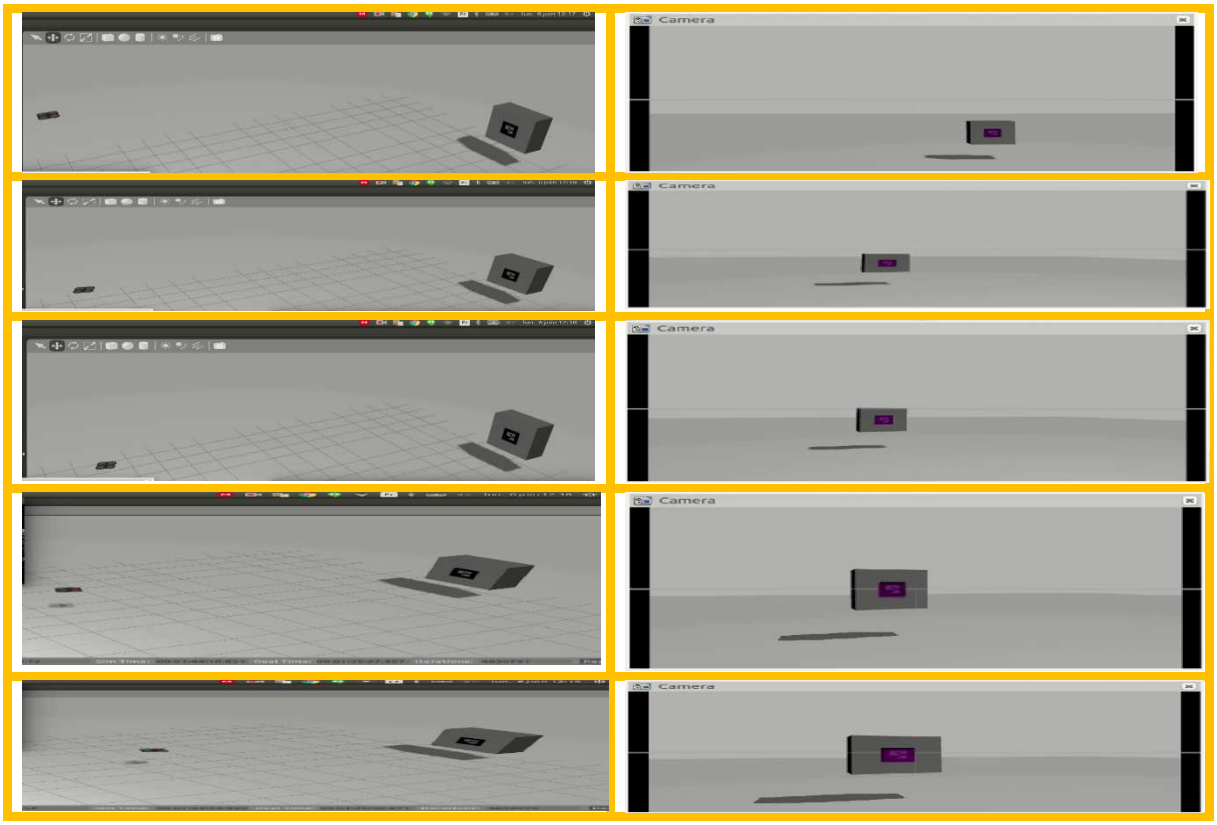


Figure IV.21. Illustration d'une trajectoire du quadrotor enregistrée par Gazebo.

Dans ce test, le quadrotor a terminé avec succès sa tâche de suivi de cible en exécutant l'ordre suivant : un lacet, un tangage et un autre lacet pour corriger sa trajectoire. Le quadrotor est placé devant la cible, mais reste légèrement incliné vers la droite tout en gardant une distance de sécurité. Le programme du réseau neuronal est conçu pour ordonner au quadrotor de faire plus de roulis et de lacets pour corriger l'inclinaison. Les résultats du premier test sont jugés acceptables.

IV.4.2. Deuxième test : Quadrotor à gauche et au bas de la cible

Dans le deuxième test, le quadrotor est positionné à gauche au bas de la cible, comme le montre la figure IV.22.



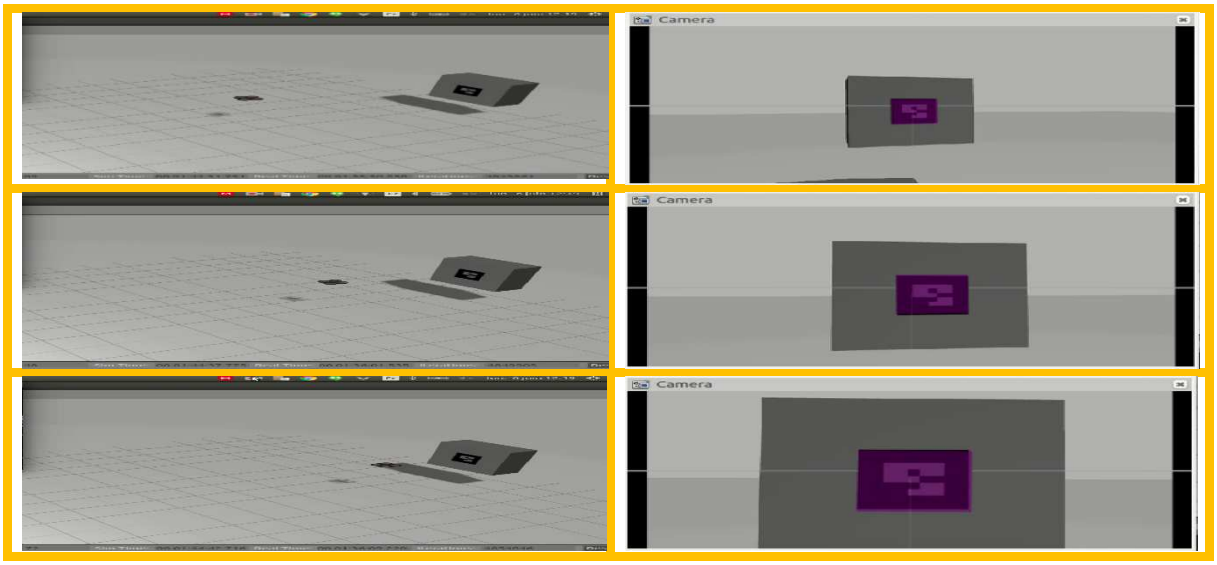


Figure IV.22. Résultats de simulation du deuxième test.

À partir de la figure IV.22, le drone détecte la cible, tourne vers la droite et se positionne devant la cible. Ensuite, le quadrotor se déplace vers la cible et effectue des rotations à droite et à gauche et une translation pour corriger sa trajectoire.

Dans ce deuxième test, le quadrotor a réussi à accomplir sa mission de manière autonome en exécutant dans l'ordre un lacet, et un roulis. Le quadrotor se dirige directement vers la cible et maintient une distance de sécurité une fois à proximité de la cible. L'algorithme de réseau neuronal a fourni au quadrotor les commandes nécessaires pour suivre la cible. La figure suivante illustre le trajet parcouru par le drone lors du deuxième test sous Gazebo.

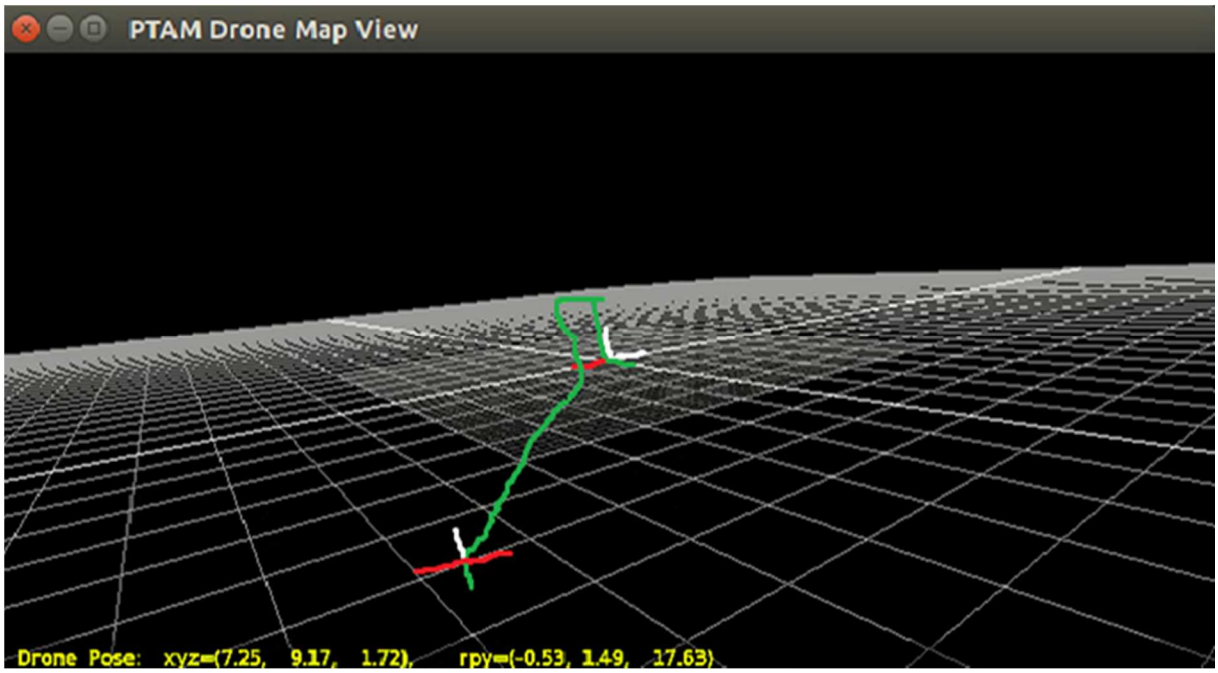


Figure IV.23. Illustration de la trajectoire du quadrotor enregistrée par Gazebo pour le deuxième test.

La figure IV.24 montres les orientations du quadrotor par rapport à la cible le long des deux axes x et y pour le premier et deuxième test de simulation.

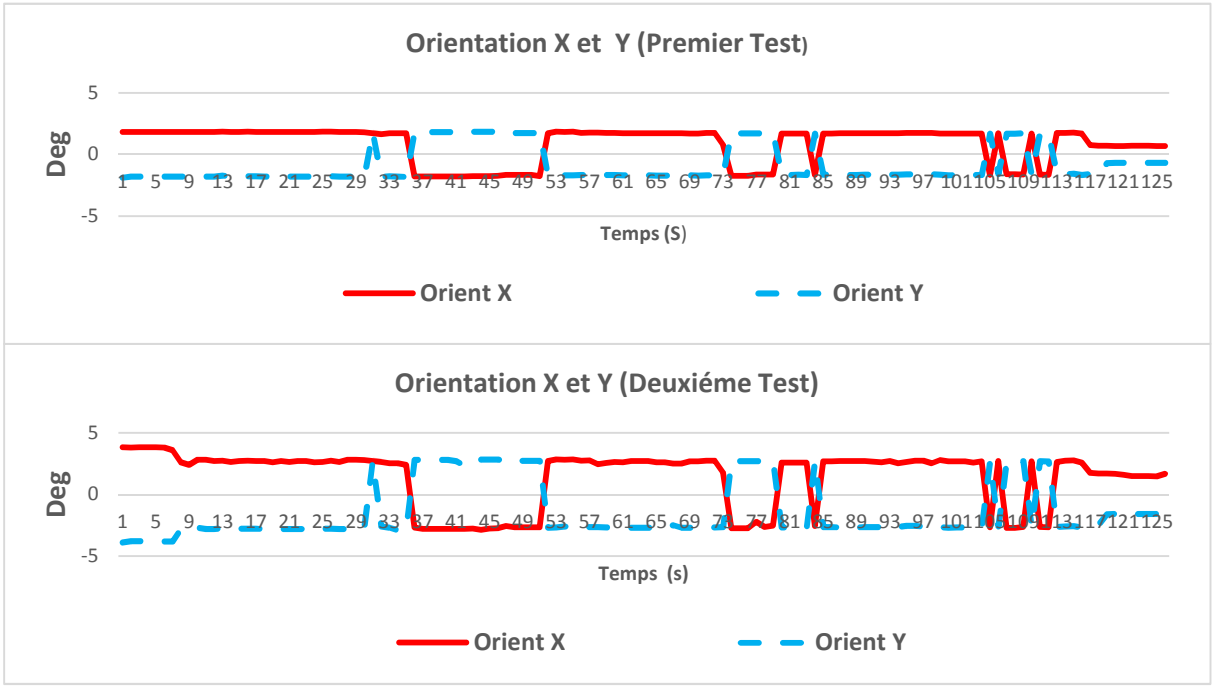


Figure IV.24. L'évolution des orientations selon les deux axes X et Y pour les essais 1 et 2.

La figure IV.25 montre les positions du quadrotor par rapport à la cible le long des trois axes x, y et z pour le premier et deuxième test de simulation.

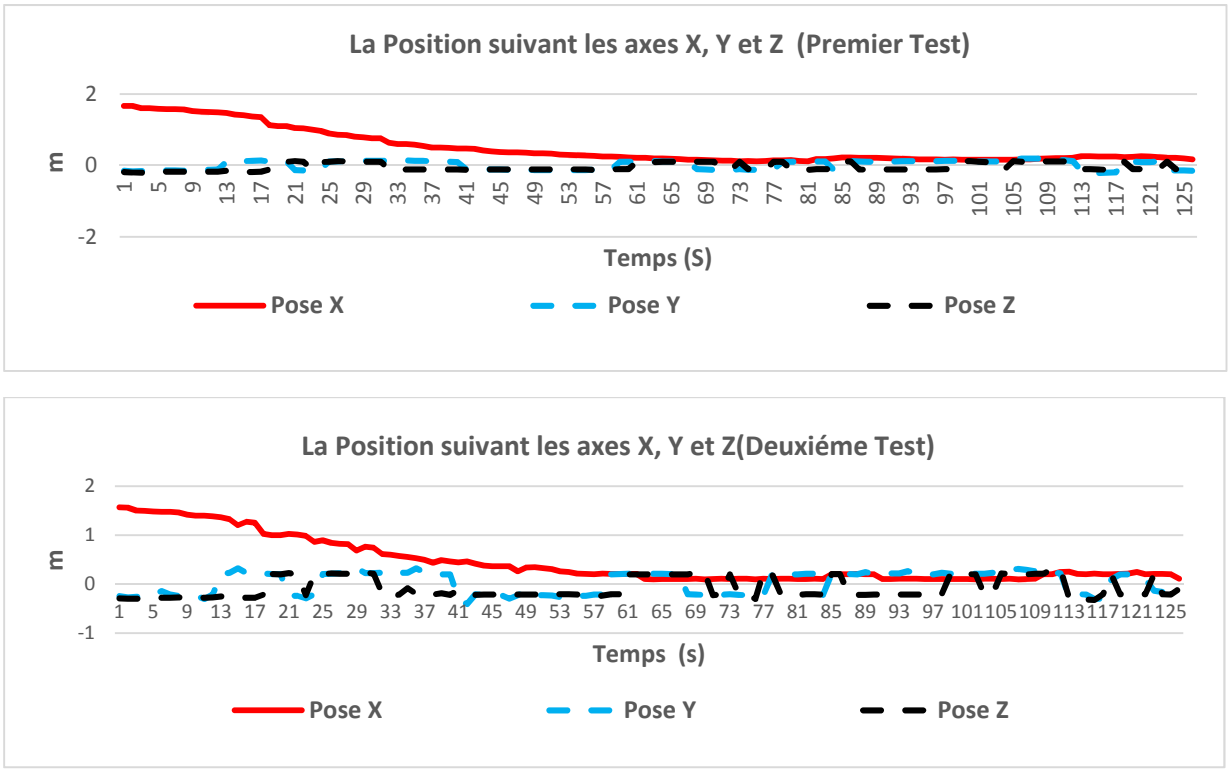


Figure IV.25. Evolution de la position du drone le long des trois axes pour les essais 1 et 2.

La figure IV.26 illustre l'évolution de la vitesse linéaire du drone selon les trois axes x et y pour le premier et deuxième test de simulation.

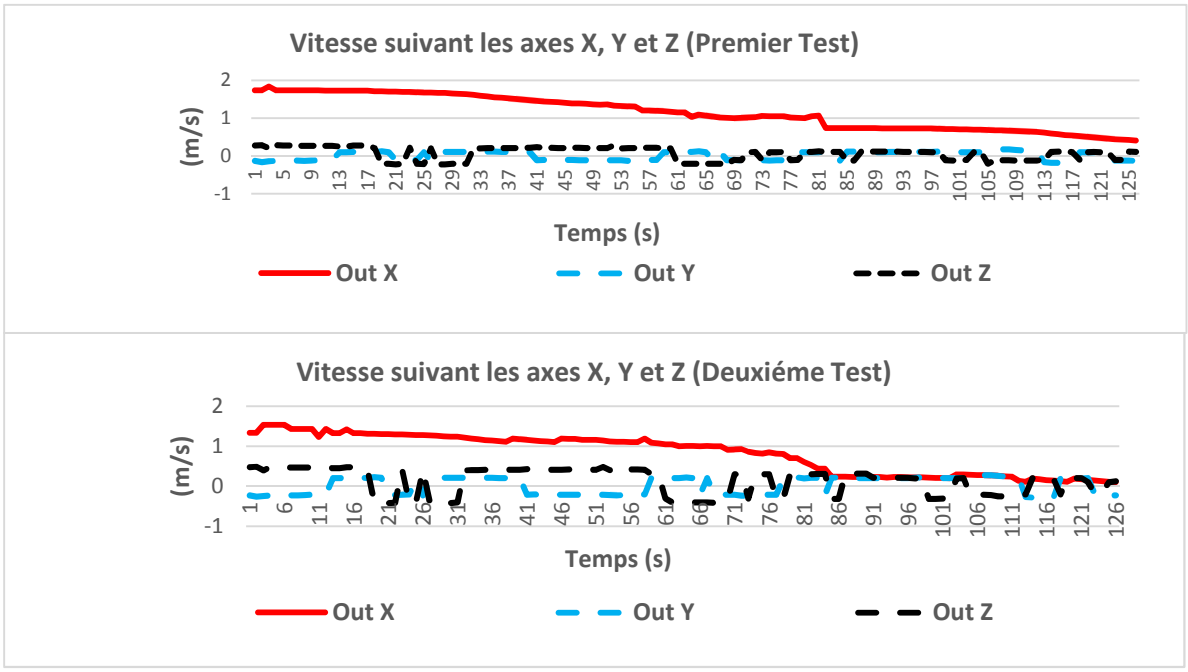


Figure IV.26. Evolution de la vitesse linéaire selon les trois axes pour les essais 1 et 2.

La figure IV.27 montre l'évolution de la vitesse angulaire du drone le long de l'axe z pour le premier et deuxième test de simulation.

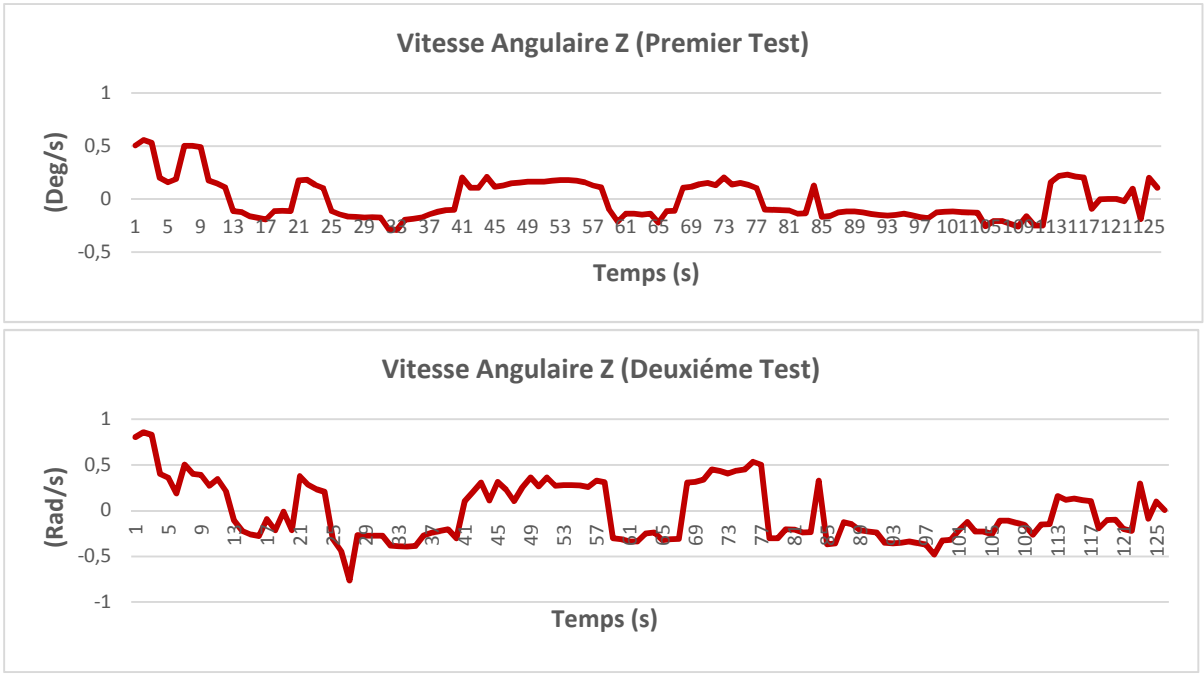


Figure IV.27. Evolution de la vitesse angulaire le long des axes z pour les tests 1 et 2.

Un environnement dans Gazebo a été créé pour tester les algorithmes de suivi de cible à l'aide d'un réseau neuronal. Tous les tests montrent qu'après le décollage, le quadrotor se dirige

directement vers la cible. Si la cible est visible par le quadrotor, l'AR.Drone lance le suivi de la cible. Une fois que le drone est à une distance acceptable pour détecter l'étiquette, un carré violet apparaît autour de l'étiquette en utilisant le logiciel ROS «ar_track_alvar» comme indiqué sur toutes les figures. À ce stade, les coordonnées de l'étiquette par rapport à la caméra du quadrotor sont disponibles pour le contrôleur de réseau neuronal.

IV.5. RESULTATS EXPERIMENTAUX

Afin de réaliser notre expérience, une application ROS a été développée. L'application commence par établir une connexion à l'AR.Drone en utilisant un réseau Wi-Fi. Une fois la connexion établie, différents nœuds sont lancés. Les nœuds sont utilisés aux fins suivantes :

- ***Package Ardrone_driver:*** envoie des commandes au quadrocopter. Les commandes sont envoyées dans un paquet UDP toutes les 5 ms et publient les images vidéo reçues du quadrotor instantanément à la rubrique ardrone / front / image_raw.

- ***Package NNController:*** le nœud du contrôleur contient les informations du contrôleur neuronal et du jeu de données ANN.

- *Package ar_track_alvar:* responsable de l'identification et de la détection de la cible, prend les trames vidéo et envoie un signal au package NNController.

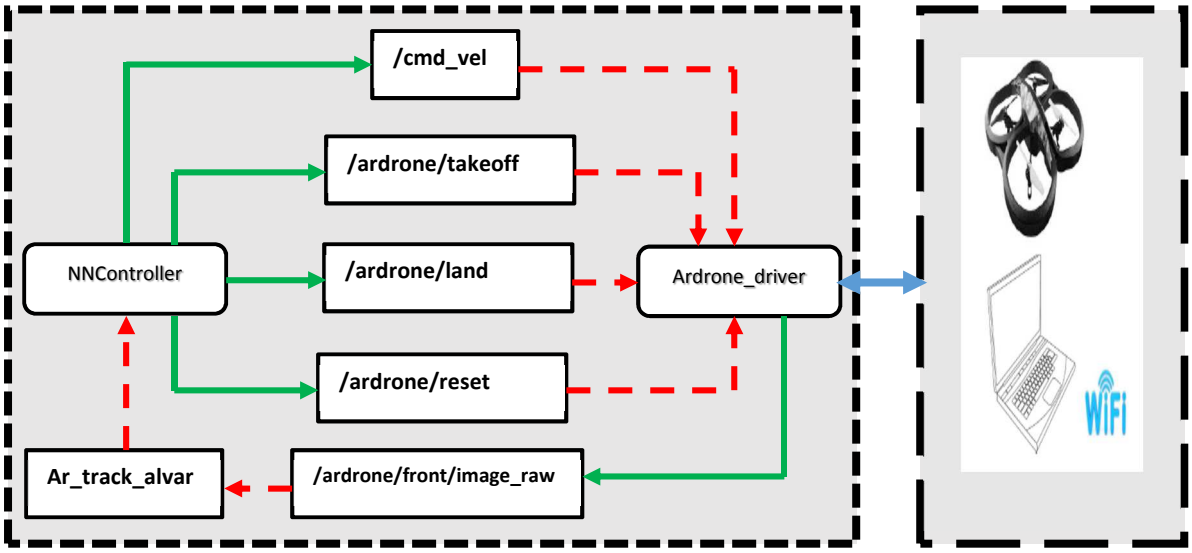


Figure IV.28. Hiérarchie de communication entre ardrone_driver et les packages implémentés pour la détection de cibles.

Après le décollage, nous contrôlons le quadrotor à l'aide du clavier de l'ordinateur. En appuyant sur une touche, ANN est activée et l'AR.Drone passe en mode autonome. Les contrôleurs basés sur ANN sont développés dans des simulations sous Gazebo ROS puis transférés vers un AR.Drone réel.

Une fois la connexion établie, l'application commence immédiatement à diffuser la vidéo obtenue à partir de la caméra frontale du quadrotor. Lorsque la cible (étiquette) est visible par le quadrotor, un cadre rouge apparaît autour de la cible. La figure IV.29 montre les séquences vidéo des résultats pratiques pour le suivi de cible.

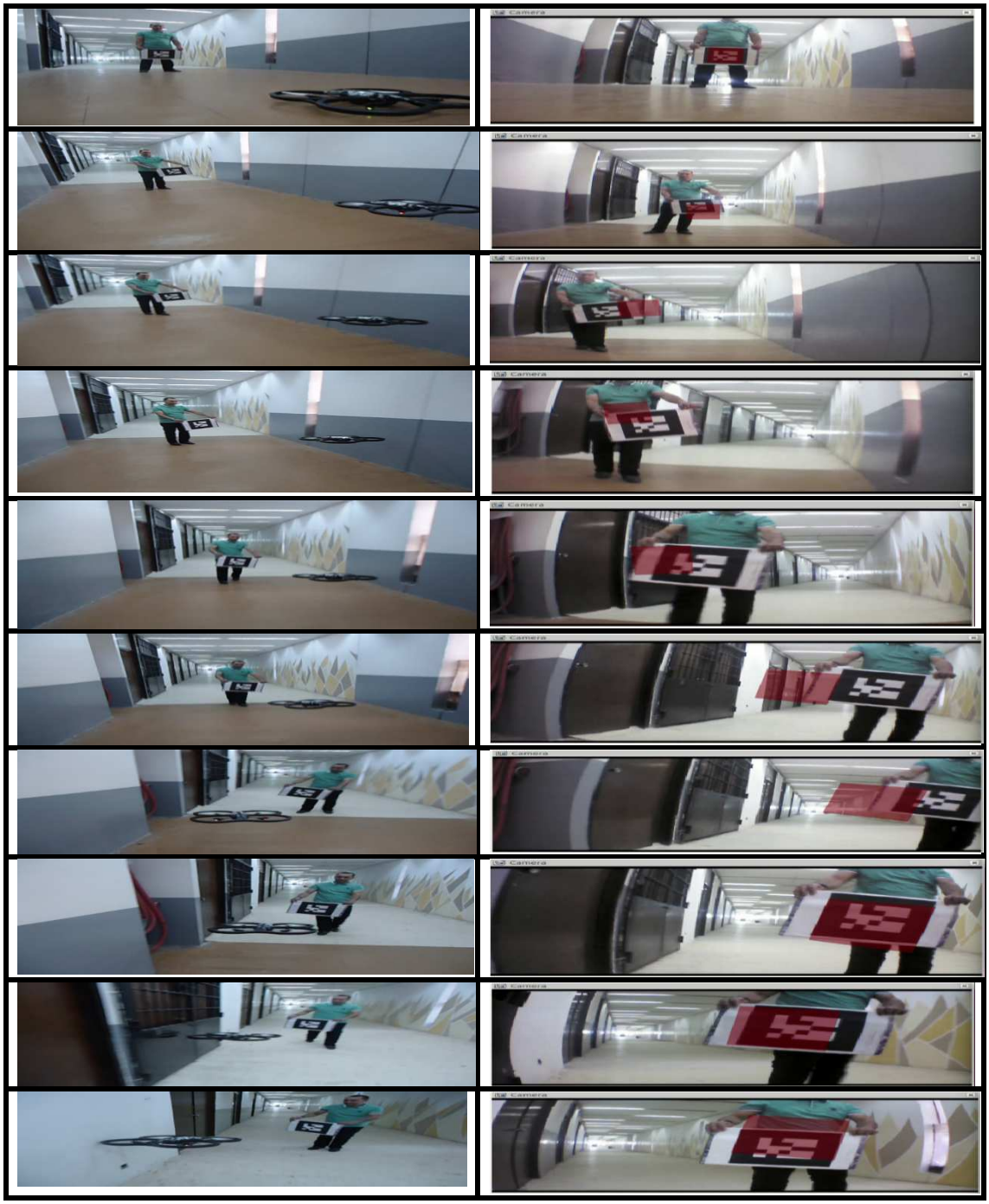


Figure V.29. Expériences en temps réel pour démontrer les performances du système de suivi de cible proposé.

A partir des résultats obtenus, on note que le quadrotor suit parfaitement la cible (tag). Nous avons observé qu'après décollage, le quadrotor se déplace directement vers la cible. Une fois la cible identifiée, un carré rouge autour de la cible est tracé. L'algorithme de réseau neuronal agit alors sur le quadrotor pour corriger sa trajectoire. Pour le résultat de la détection en temps réel avec ROS, on peut voir l'expérience et la capture d'écran sur la caméra du drone. La figure IV.30, montre l'évolution de l'orientation de la cible par rapport au drone le long des axes x et y, appelés respectivement Pose.orientation X et Pose.Orientation Y. Ces deux paramètres ont été utilisés comme entrées pour le contrôleur de réseau neuronal.

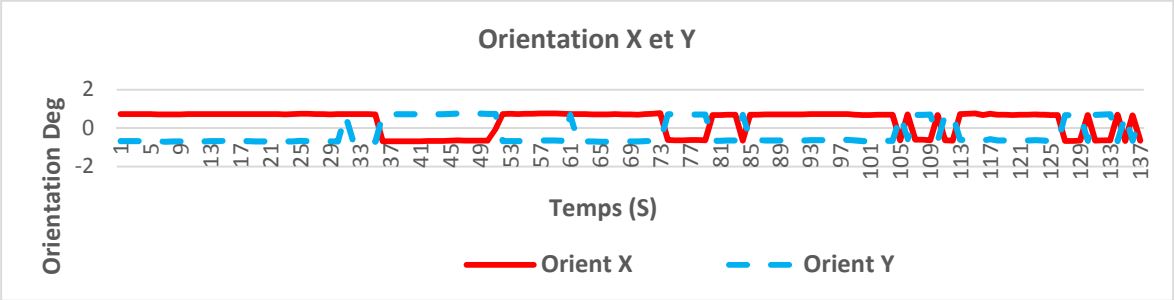


Figure IV.30. L'évolution des orientations selon les deux axes X et Y.

A partir de la figure d'orientation, on peut voir que le drone est orienté selon l'axe des abscisses, il réalisera une autre orientation inverse sur l'axe des ordonnées.

La figure IV.31 illustre les trois positions du drone par rapport à la cible. Trois positions ont été utilisées comme entrées pour le contrôleur de réseau neuronal réalisé.

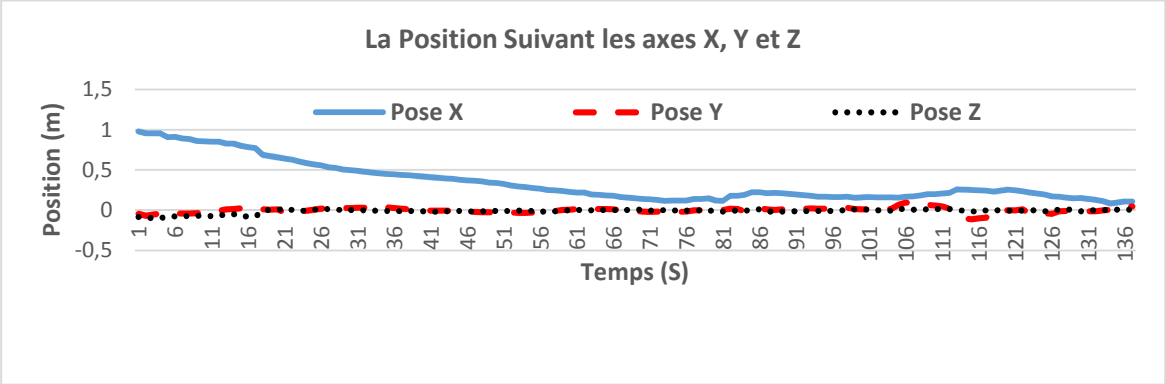


Figure IV.31. Evolution de la position du drone le long des trois axes.

A partir de cette figure, on peut voir que le drone avance rapidement le long de l'axe des x. L'évolution de la position du drone le long des deux autres axes y et z varie lentement vers l'avant et reste inchangée par rapport à l'axe des abscisses.

La figure IV.32, illustre les variations de la vitesse linéaire le long des trois axes x, y et z. Nous remarquons que le quadrotor se déplace rapidement vers la cible avec une vitesse de départ de 1 m / s, mais une fois la cible proche, la vitesse diminue. Cela est justifié parce que le quadrotor doit maintenir une distance de 1 m de la cible.

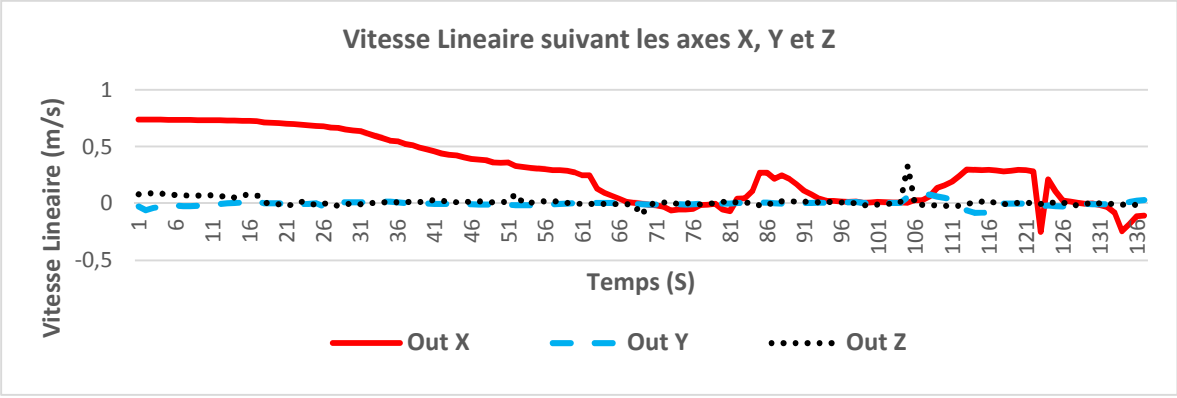


Figure IV.32. Variation de la vitesse linéaire selon l'axe des x, l'axe des y et l'axe des z.

Les données de la vitesse linéaire le long des trois axes sont utilisées comme données de sortie par le contrôleur de réseau neuronal conçu. La figure IV.33 illustre les variations de la vitesse angulaire le long de l'axe z.

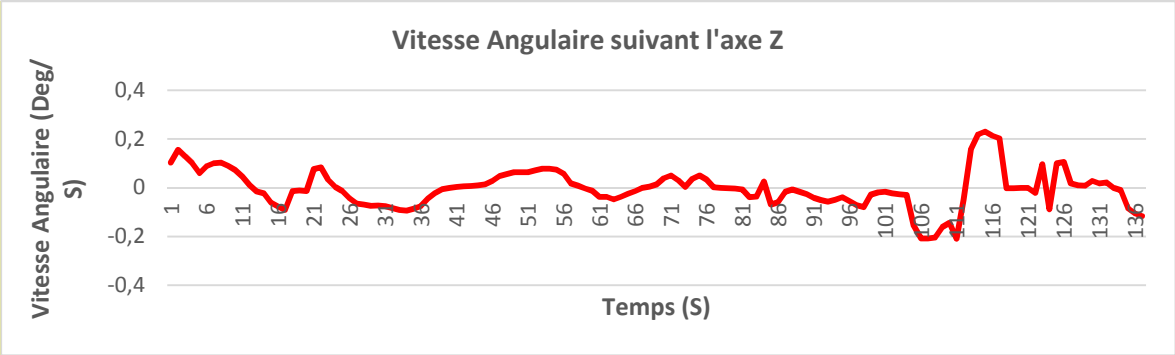


Figure IV.33. Variations de la vitesse angulaire le long de l'axe des z.

La vitesse angulaire est utilisée en tant que sortie par le contrôleur de réseau neuronal. La figure IV.32 montre que le quadrotor change d'orientation pour corriger sa trajectoire.

IV.6. CONCLUSION

Dans ce chapitre, nous avons présenté un système de vision pour l'identification et le suivi de cible à l'aide d'un drone Parrot AR.Drone 2.0. La technique proposée est basée sur la détection d'un marqueur par la caméra et en utilisant un contrôleur de réseau neuronal. Ensuite, la distance entre le quadrotor et la cible est estimée et l'algorithme MLP du réseau neuronal est utilisé pour voler vers le marqueur. Une fois la position cible atteinte, la position de l'étiquette est à nouveau détectée. Pour corriger la dérive possible à l'approche de la cible, la distance entre le quadrotor et la cible est estimée et utilisée pour corriger la position du drone.

Sur la base des résultats obtenus, on peut conclure que l'algorithme utilisant les réseaux neuronaux artificiels reconnaît la cible tout comme le module de reconnaissance utilisé par ar_track_alvar dans ROS. Ainsi, l'algorithme proposé fournit de meilleures performances dans le processus d'identification, ce qui est très important pour les applications temps réel. Les résultats des tests expérimentaux et les simulations montrent que la mise en œuvre du contrôleur neuronal pour la détection et le suivi des objets a été un succès.

Dans le dernier chapitre nous aborderons le suivi de trajectoire en utilisant un contrôleur flou.

Chapitre V :

Suivi de Trajectoire A Base d'un Contrôleur Flou

Suivi de Trajectoire A Base d'un Contrôleur Flou

V.1. INTRODUCTION

Dans ce chapitre, nous allons élaborer en premier une simulation sur le quadrotor par la suite nous utiliserons la plateforme réel AR.Drone sous Matlab / Simulink, pour cela nous introduisons des trajectoires prédéfinies d'avance en utilisant la technique du Waypoints dans le but d'imposer au quadrotor des points désirés selon les trois axes X, Y et Z avec une durée en seconde pour chaque point. Une description de chaque bloc de simulation sera détaillée. Par la suite nous passerons aux tests pratiques en introduisons les mêmes trajectoires utilisées lors de la simulation.

V.2. DESCRIPTION DU SYSTEME DE CONTROLE

Le système conçus doit pouvoir suivre des trajectoires simples telles que : élévation verticale, lignes droites, courbes lisses, cercles, polygones réguliers, entre autres. Ces trajectoires sont dans une base de données et si nécessaire, nous pouvons en ajouter de nouvelles. A cet effet un contrôleur flou est proposé pour stabiliser le fonctionnement du quadrotor pour suivre des trajectoires dans la simulation et pour les tests réels. Le contrôleur proposé est réalisé avec un système flou, ce système est basé sur le concept de la logique floue afin de traiter les signaux du quadrotor et générer une réponse. Pour cela, 4 pilotes ont été développés, dont les entrées sont : la hauteur et les trois angles d'orientation pour chacun. La figure V.1, illustre le schéma synoptique de la conception.

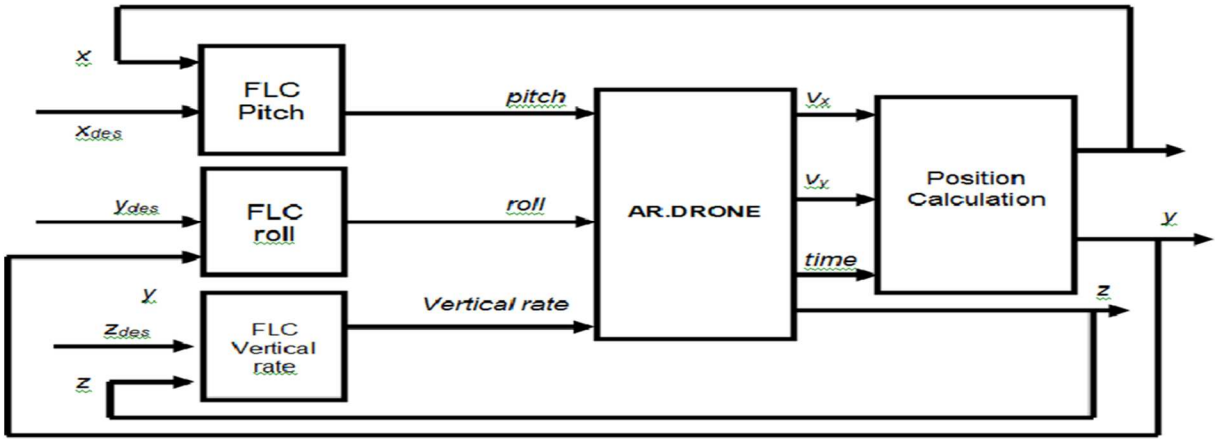


Figure V.1. Schéma synoptique de la commande.

La trajectoire désirée est ajoutée au système via une série de points préalablement établis, chacun de ces points étant validé dans une étape de traitement de trajectoire. Par la suite nous déterminons si le point a déjà été atteint et qu'un temps d'attente spécifique est validé pour chaque point. Le contrôleur reçoit les informations de chacun point souhaité et envoie les signaux nécessaires de sorte que le quadrotor effectue le déplacement. La position est estimée grâce à l'utilisation des capteurs intégrés dans l'AR.Drone pour déterminer l'emplacement et le retour d'information au système. Pour visualiser les performances du système, il est possible d'utiliser des outils dans le programme Simulink à l'aide des étendues graphiques de cet outil.

Pour la conception des contrôleurs flous nous avons utilisés l'outil sous Matlab / Simulink qui est FuzzyToolBox , avec cette boîte à outils il est possible d'utiliser et d'ajouter un modèle flou pour contrôler le quadrotor. Le contrôle de ce dernier a été réalisé à l'aide de 4 contrôleurs flous différents (figure V.2), qui seront exécutés simultanément pour le suivi de trajectoire. Ces contrôleurs sont celui de l'élévation et ceux des 3 angles d'orientation du système (roulis, tangage et lacet).

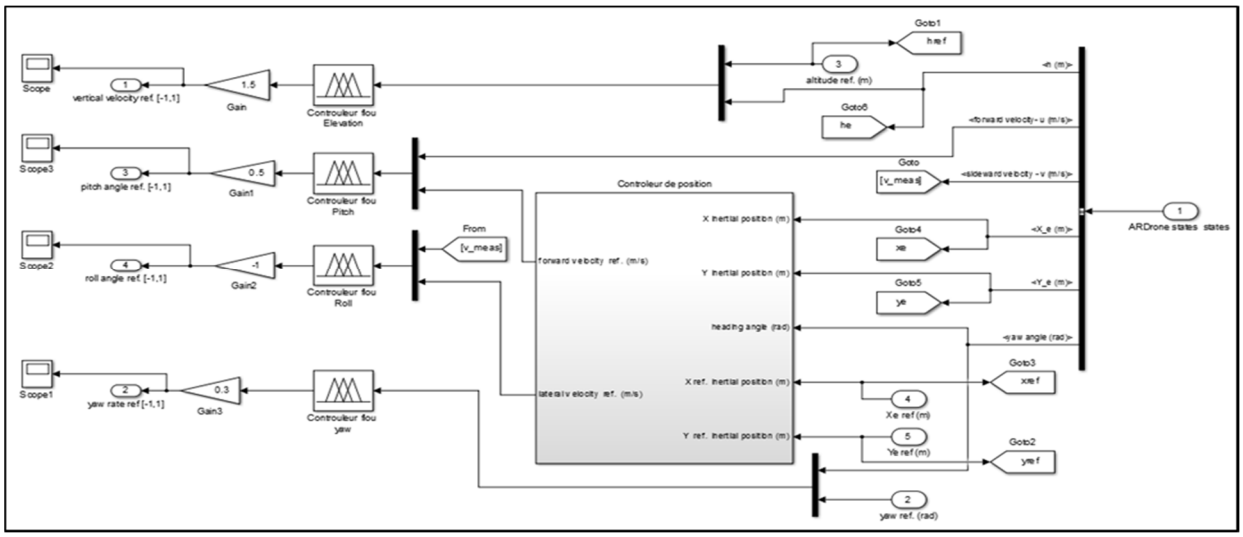


Figure V.2. Les contrôleurs flous intégrés dans le bloc de simulation.

À l'entrée des contrôleurs, les valeurs mesurées et les valeurs de référence désirées de chaque variable sont obtenues, respectivement. La méthode d'inférence traite les valeurs d'entrée après leur fusion. Ces valeurs entrent dans le système d'inférence floue, par la suite il exécute le processus de contrôle nécessaire pour chaque contrôleur.

En tant que méthode de déffuzification, nous avons la méthode du Centroïde, qui calcule la valeur du Centroïde Moyenne de toutes les règles existantes. A la sortie de chacun de ces contrôleurs, une valeur est ajoutée à chacune des 4 variables, pour atteindre les points établis dans la trajectoire de référence.

Chacun des contrôleurs flous mis en œuvre présente les caractéristiques suivantes :

Elévateur FLC : Dispose de cinq fonctions d'appartenances pour chaque entrée et cinq autres pour la sortie, les entrées ont une plage de [0 18] et la sortie avec une plage de [-10 10]. Toutes les fonctions d'appartenance sont de type triangulaire.

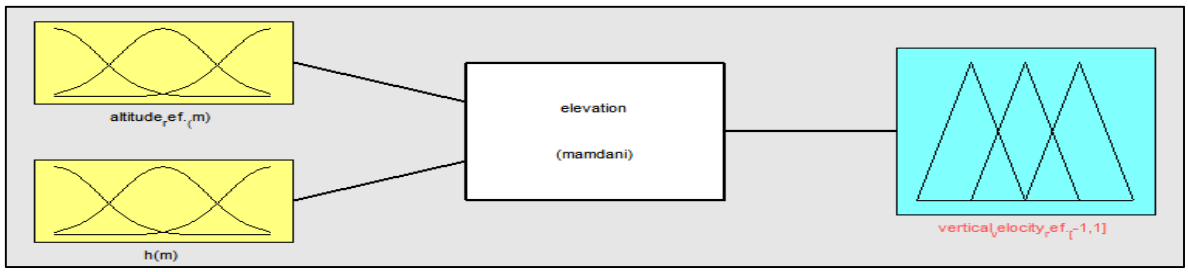


Figure V.3. Fonctions d'appartenance pour l'Elévation.

La figure suivante présente la sortie de la vélocité verticale.

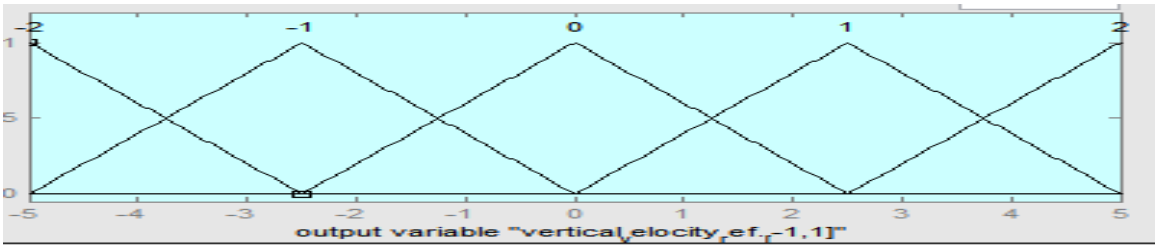


Figure V.4. L'ensemble flou de la variable de sortie de la vélocité verticale.

La figure V.5, présente L'entrée de la référence pour l'altitude.

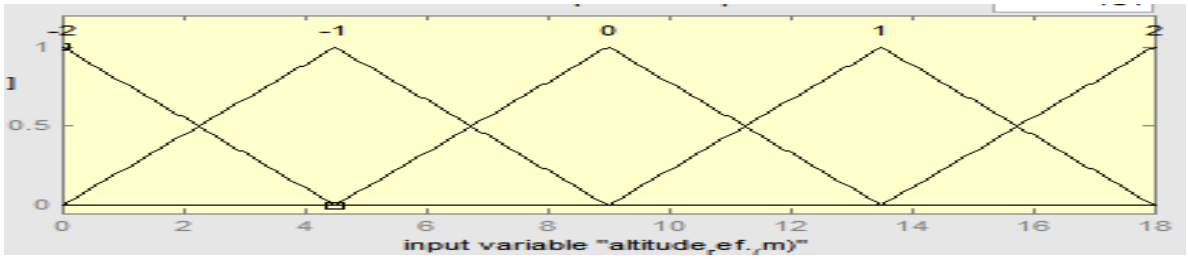


Figure V.5. Fonctions d'appartenance de la référence de l'altitude.

La figure suivante, illustre L'entrée de l'altitude.

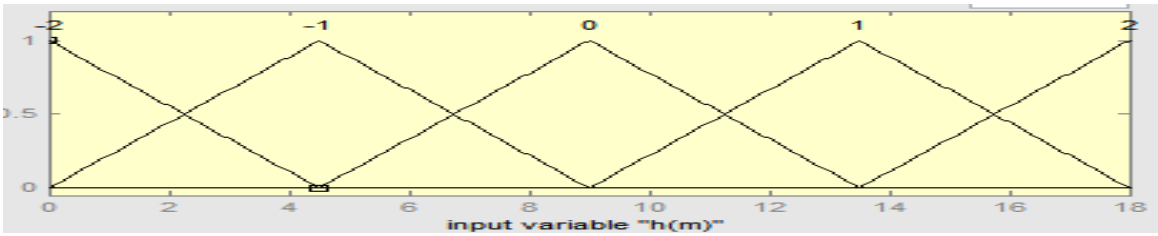


Figure V.6. Fonctions d'appartenance de l'altitude.

Le tableau suivant montre les 25 règles établies pour contrôler l'altitude du quadrotor suivant les axes **Z, Y** et **X** respectivement **t**els que les variables linguistiques sont comme suit :
Z (0): null, TP (2) : Très positif, P (1): Positif, TN (-2): Très négatif, N (-1): Négatif
Les entées sont les Références des vélocités et les vélocités

Tableau V.1. Les règles floues pour l'altitude.

Cmd / Réf	**TN**	**N**	**Z**	**P**	**TP**
TN	Z	N	TN	N	TN
N	P	Z	N	TN	TN
Z	TP	P	Z	N	TN
P	TP	TP	P	Z	N
TP	TP	P	TP	P	Z

La lecture du tableau se traduit comme suit :

Comme exemple on prend la référence de la vélocité de la première ligne (Très négative) et la valeur de la vélocité de la première colonne (Très négative) :
Si (Réf est Très négatif) *et* (Cmd est Très négatif) *donc* (la sortie nulle)
Et ainsi les autres règles sont déduites ...
Si (Réf est Négatif) *et* (Cmd est Très négatif) *donc* (la sortie est Positive)
Si (Réf est Nul) *et* (Cmd est Très négatif) *donc* (la sortie Z)….

La figure (V.7) présente les règles établies pour le contrôleur d'élévation

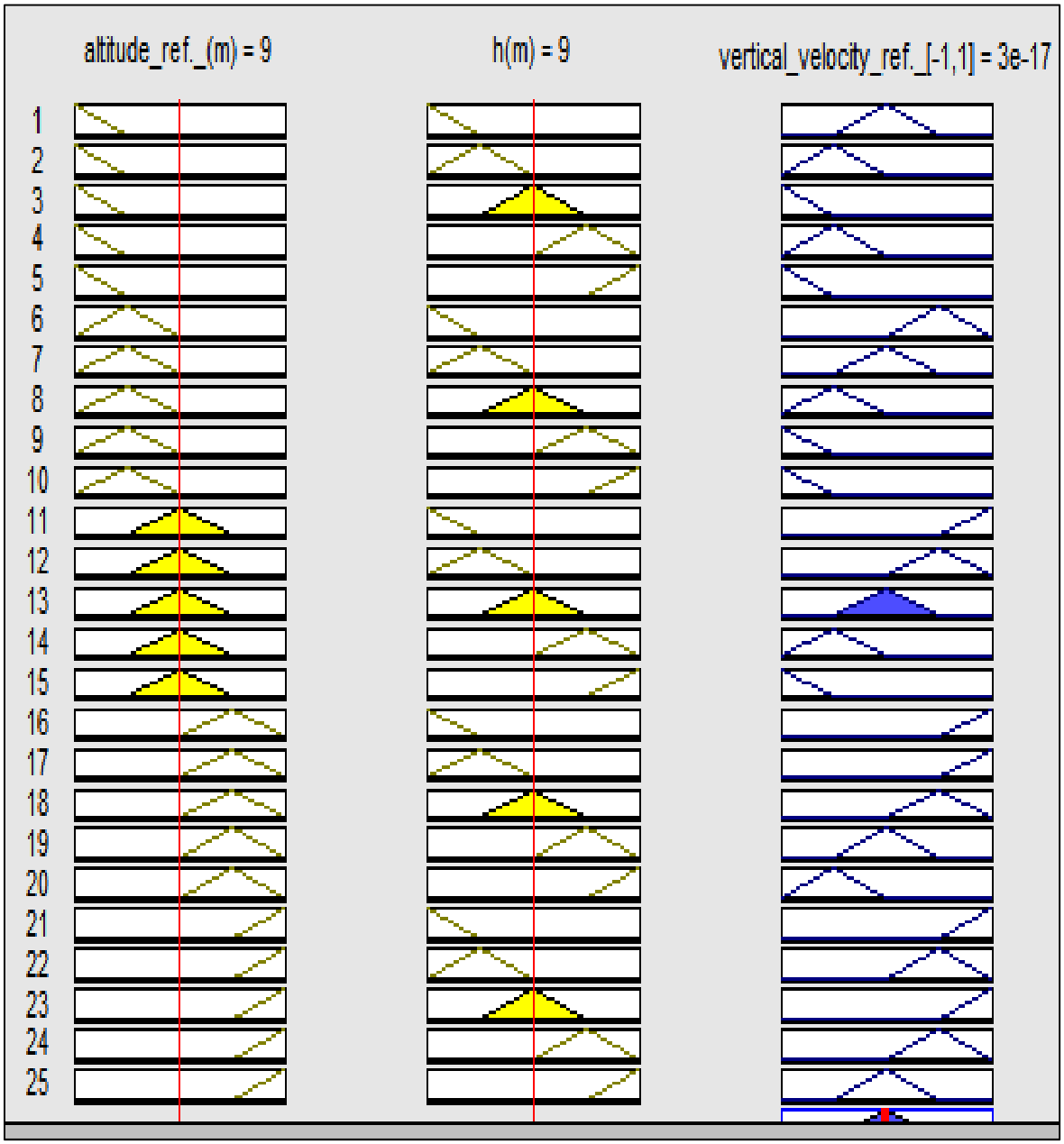


Figure V.7.Règles établies pour le contrôleur d'élévation.

La surface des règles floues pour le contrôleur d'Elévation est schématisée sur la figure suivante :

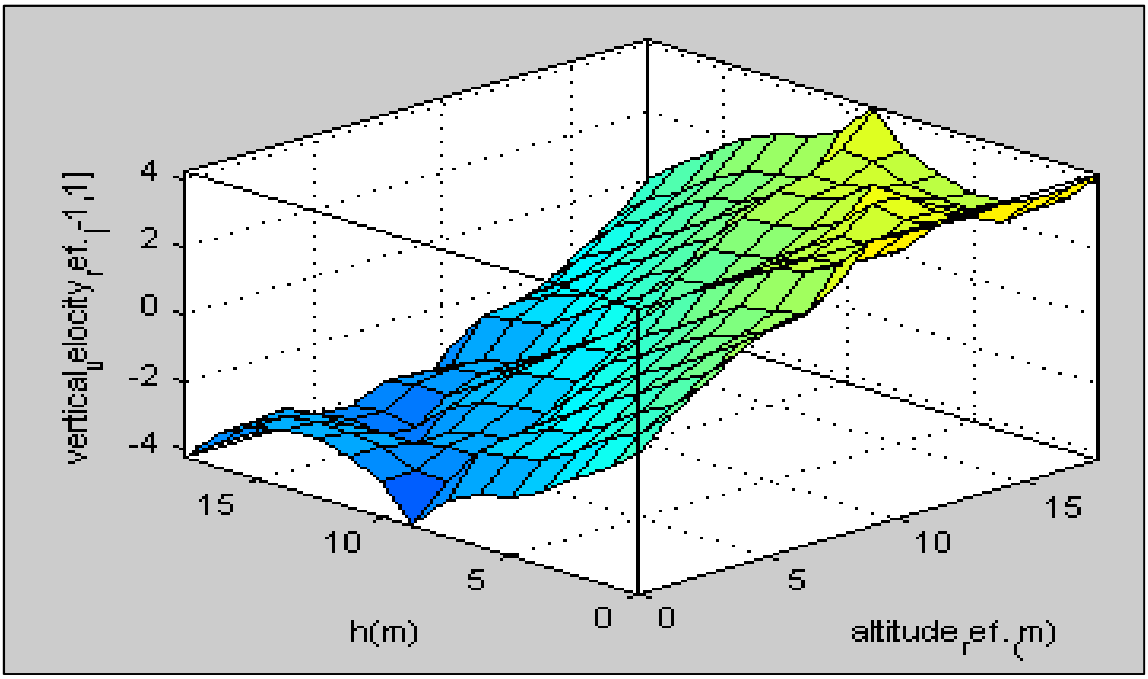


Figure V.8.La surface des règles floues pour le contrôleur d'Elévation.

Pitch_FLC: Ce contrôleur dispose de cinq fonctions d'appartenance pour chaque entrée et sortie, pour les entrées nous avons utilisé une plage de [-20 20] et la plage de sortie est de [-18 18]. Toutes les fonctions d'appartenance sont de type triangulaire.

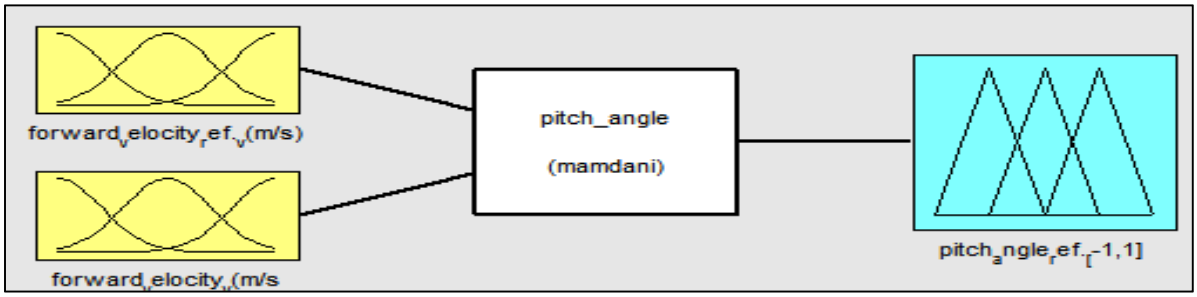


Figure V.9.Fonctions d'appartenance Tangage.

La figure suivante présente la sortie de l'angle de tangage.

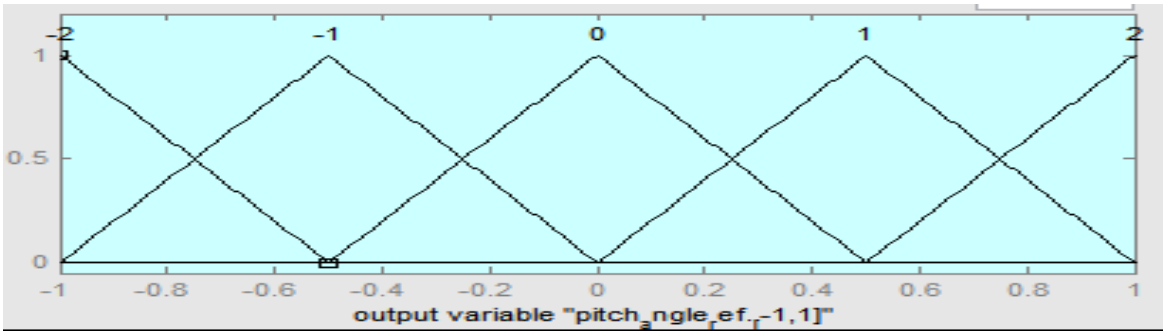


Figure V.10.Fonctions d'appartenance de l'angle de tangage.

L'entrée de la référence de la vélocité de l'avancement :

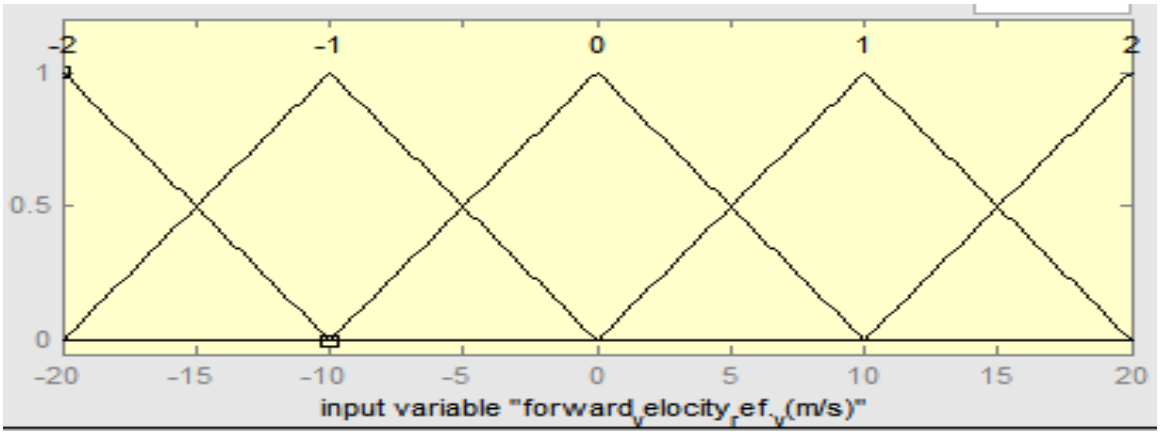


Figure V.11.Fonctions d'appartenance de la référence de tangage.

L'entrée de la vélocité de l'avancement :

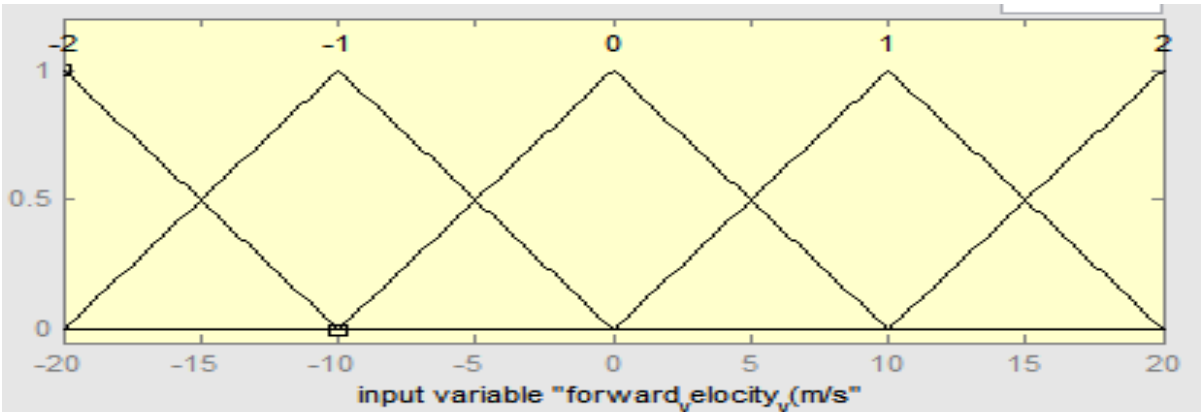


Figure V.12.Fonctions d'appartenance de tangage.

Le tableau suivant résume les 25 règles établies pour contrôler le tangage du quadrotor suivant les axes **Z, Y** et **X** respectivement **t**el que les variables linguistiques sont comme suit :

Tableau V.2. Règles floues pour le tangage.

Cmd / Réf	**TN**	**N**	**Z**	**P**	**TP**
TN	Z	N	TN	N	TN
N	P	Z	N	TN	TN
Z	TP	P	Z	N	TN
P	TP	TP	P	Z	N
TP	TP	P	TP	P	Z

La figure (V.13) présente les règles établies pour le contrôleur de Tangage

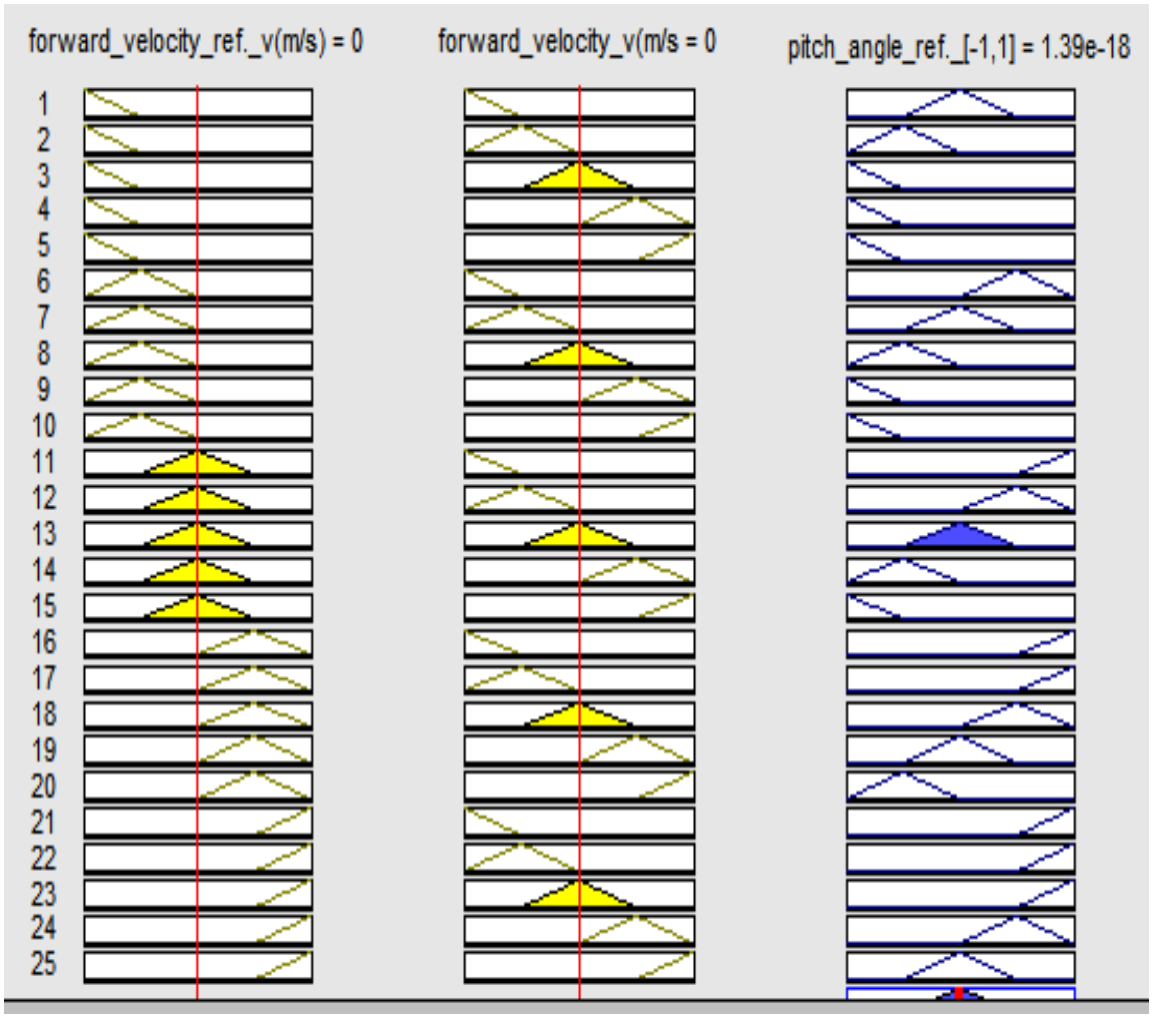


Figure V.13.Règles établies pour le contrôleur de tangage.

La surface des règles floues pour le contrôleur de tangage est présentée sur la figure suivante.

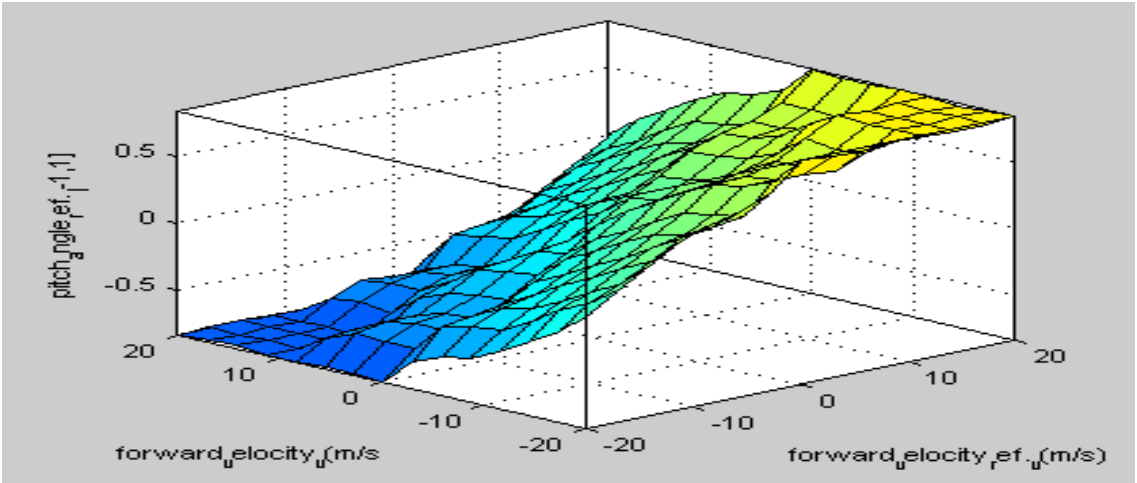


Figure V.14.La surface des règles floues pour le contrôleur de tangage.

Roll_FLC: Ce contrôleur dispose de cinq fonctions d'appartenance pour chaque entrée et sortie. Une de [-20 20] est attribuées aux entrées, par contre la sortie à une plage de [-18 18]. Toutes les fonctions d'appartenance sont de type triangulaire

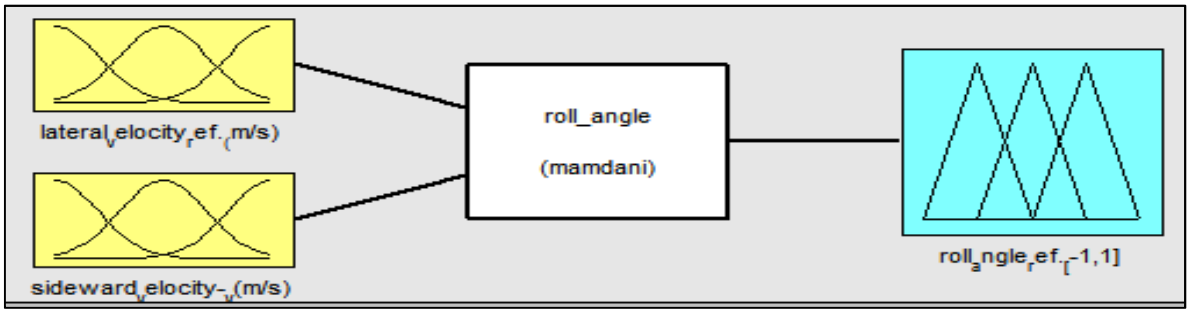


Figure V.15.Fonctions d'appartenance Roulis.

La sortie de l'angle de roulis :

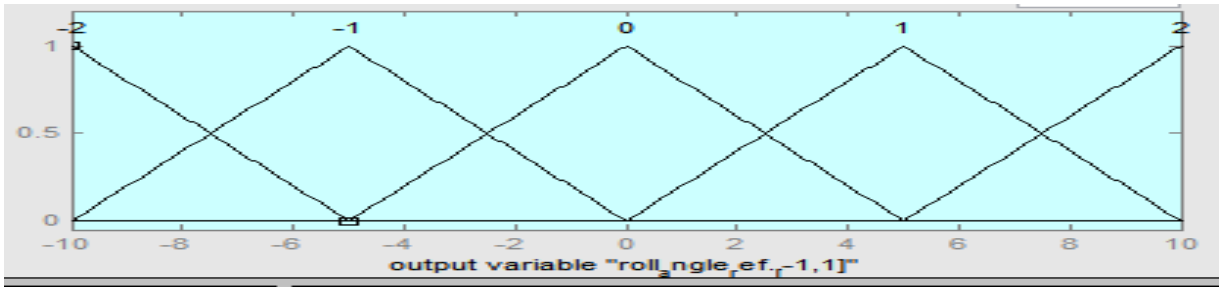


Figure V.16.Fonctions d'appartenance de l'angle de roulis.

L'entrée de la référence latérale : de la vélocité

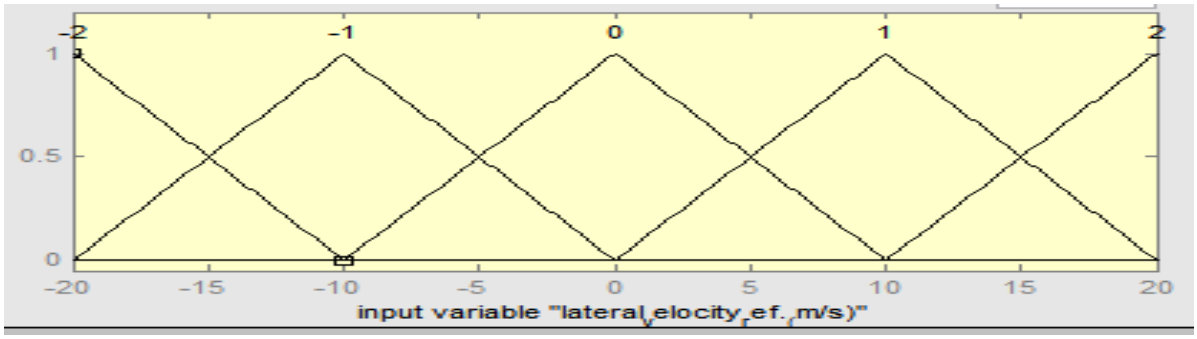


Figure V.17.Fonctions d'appartenance de la référence de roulis.

L'entrée de la vélocité latérale :

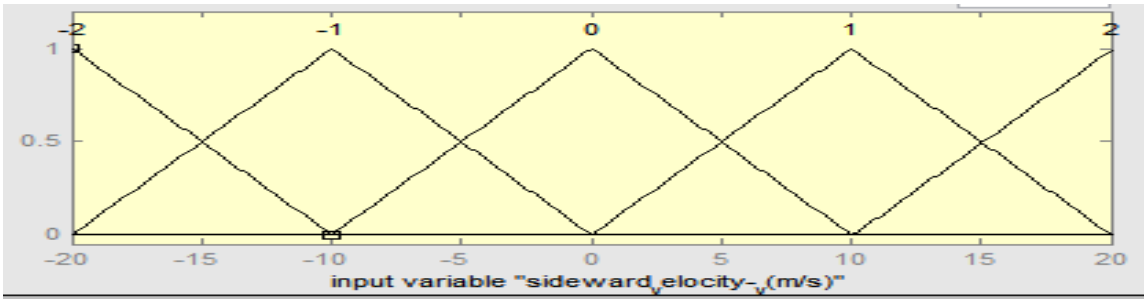


Figure V.18.Fonctions d'appartenance de roulis.

Le tableau suivant montre les 25 règles établies pour contrôler le roulis du quadrotor suivant

les axes **Z, Y** et **X** respectivement **t**el que les variables linguistiques sont comme suit :
Z (0): nul, TP (2) : Très positif, P (1): Positif, TN (-2): Très négatif, N (-1): Négatif

Tableau V.3. Les règles floues pour le correcteur de roulis

Réf \ Cmd	**TN**	**N**	**Z**	**P**	**TP**
TN	Z	N	TN	N	TN
N	P	Z	N	TN	TN
Z	TP	P	Z	N	TN
P	TP	TP	P	Z	N
TP	TP	P	TP	P	Z

La figure V.19 suivante présente les règles établies pour le contrôleur de Roulis.

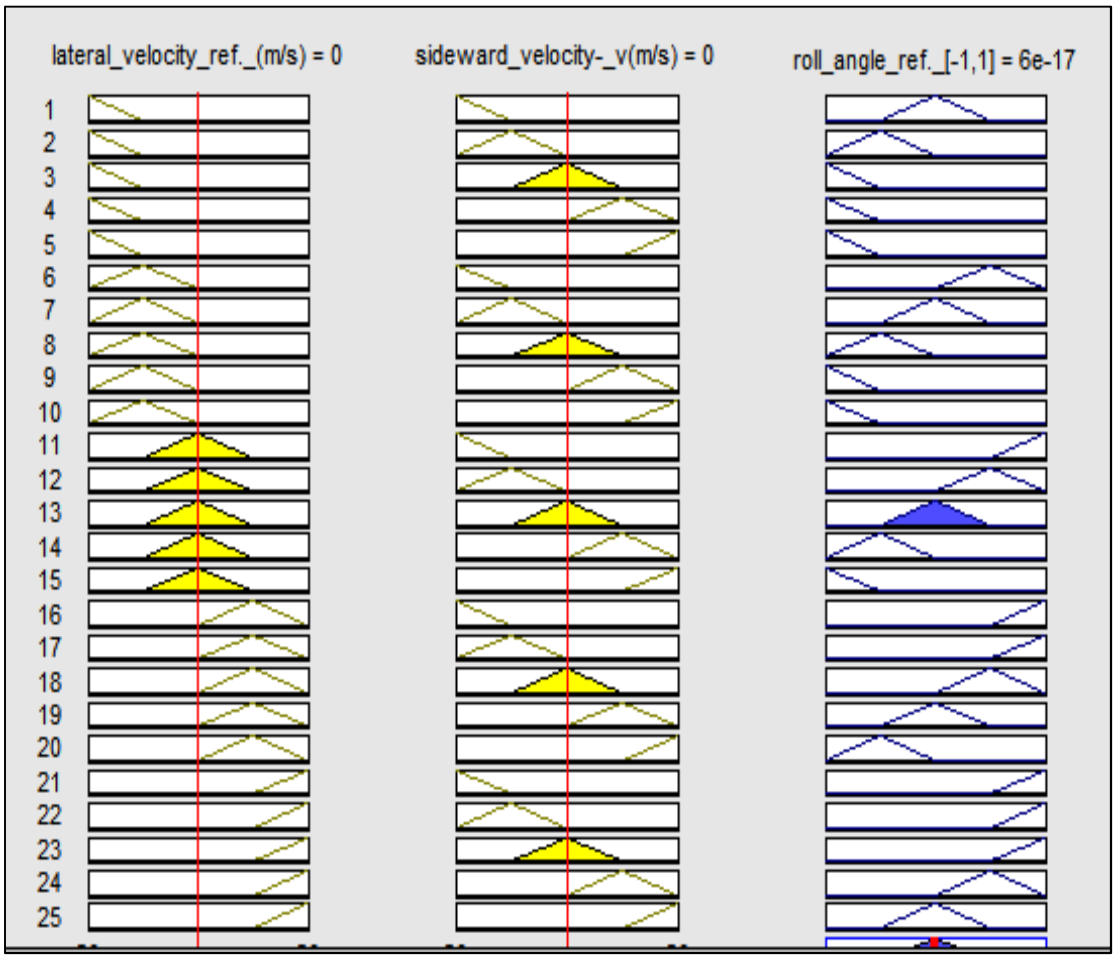


Figure V.19. Règles établies pour le contrôleur de roulis.

La surface des règles floues pour le contrôleur de Roulis :

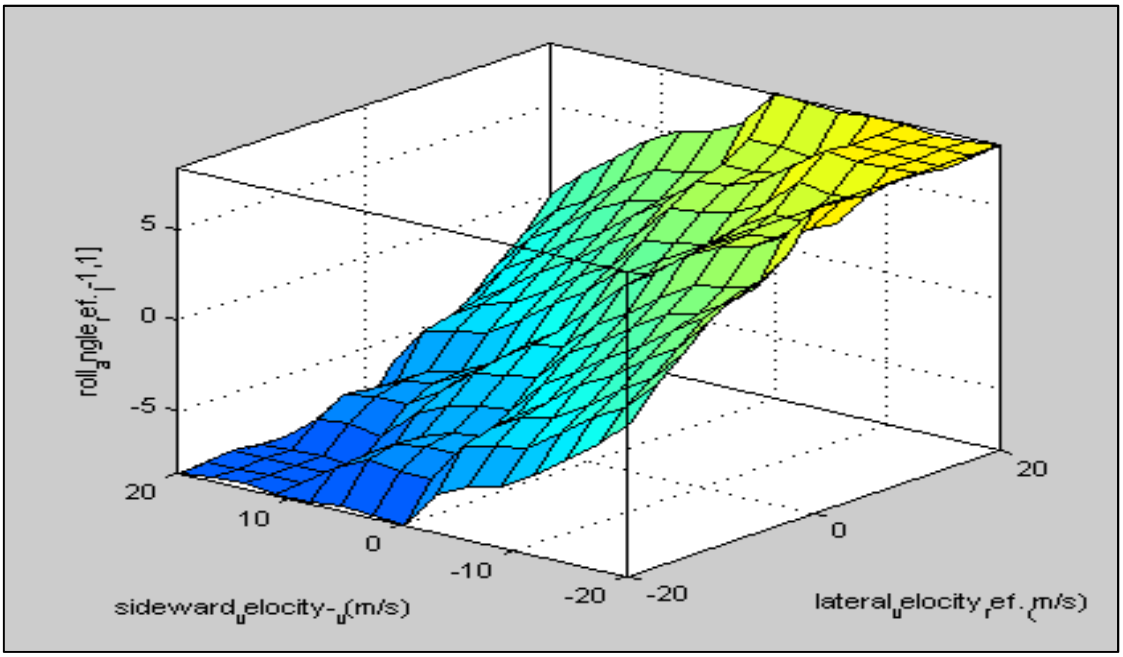


Figure V.20.La surface des règles floues pour le contrôleur de roulis.

Yaw_FLC: Ce contrôleur dispose de sept fonctions d'appartenance pour les entrées et sorties. Les entrées disposent d'une plage de [0 π] et la sortie de [-5 5]. Toutes les fonctions d'appartenance sont de type triangulaire.

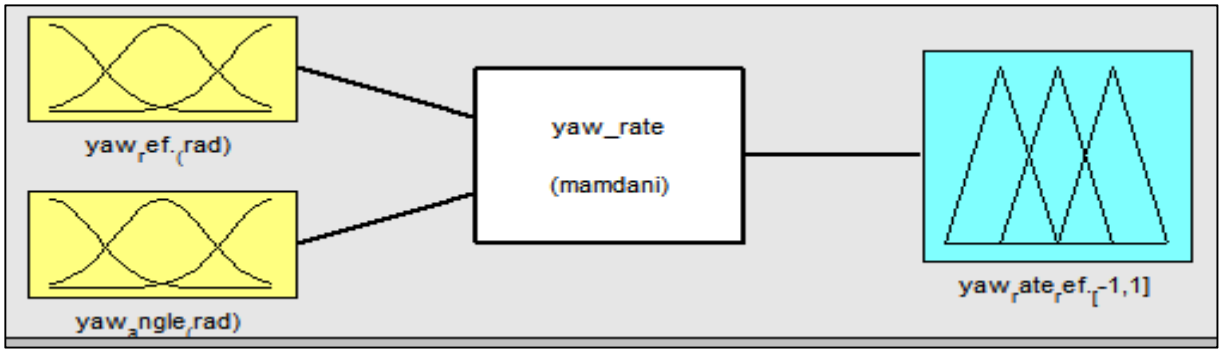


Figure V.21 : Fonctions d'appartenance Lacet.

La sortie de l'angle de Lacet :

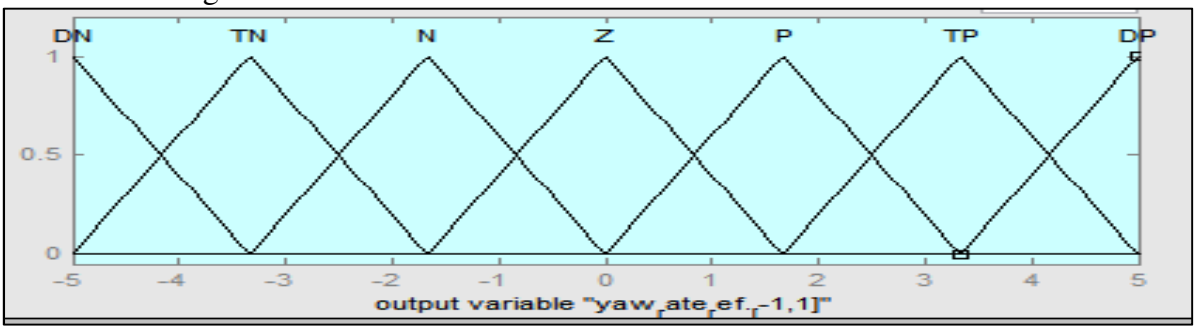


Figure V.22. Fonctions d'appartenance de l'angle de lacet.

L'entrée de la référence de l'angle de lacet :

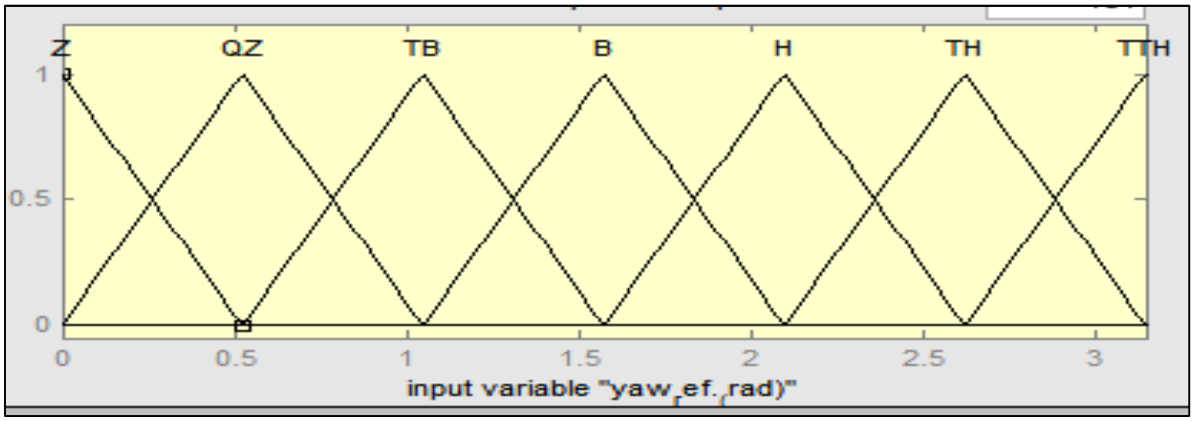


Figure V.23. Fonctions d'appartenance de la référence de lacet.

L'entrée de l'angle de lacet :

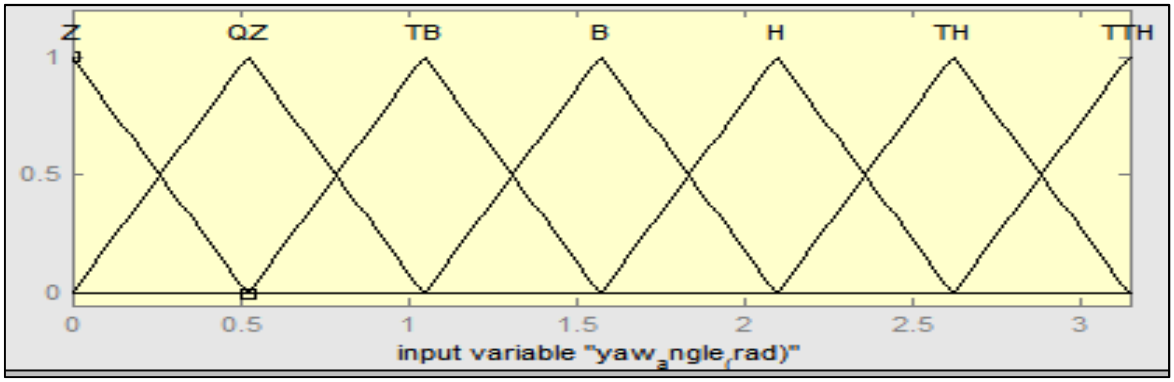


Figure V.24. Fonctions d'appartenance de lacet.

Le tableau suivant montre les 49 règles établies pour contrôler le lacet du quadrotor autour de l'axe Z :

Tableau V.4. La table d'inférence pour le contrôleur du lacet.

Yaw-vel / Yaw-vel-ref	Z	QZ	TB	B	H	TH	TTH
Z	Z	Z	N	TN	DN	DN	DN
QZ	Z	Z	N	N	TN	DN	DN
TB	P	Z	N	N	TN	TN	DN
B	N	Z	Z	N	N	TN	DN
H	N	N	N	Z	Z	TN	DN
TH	TN	N	N	Z	TN	TN	DN
TTH	DN	TN	TN	N	N	Z	Z

Les règles floues pour le correcteur de Lacet sont :

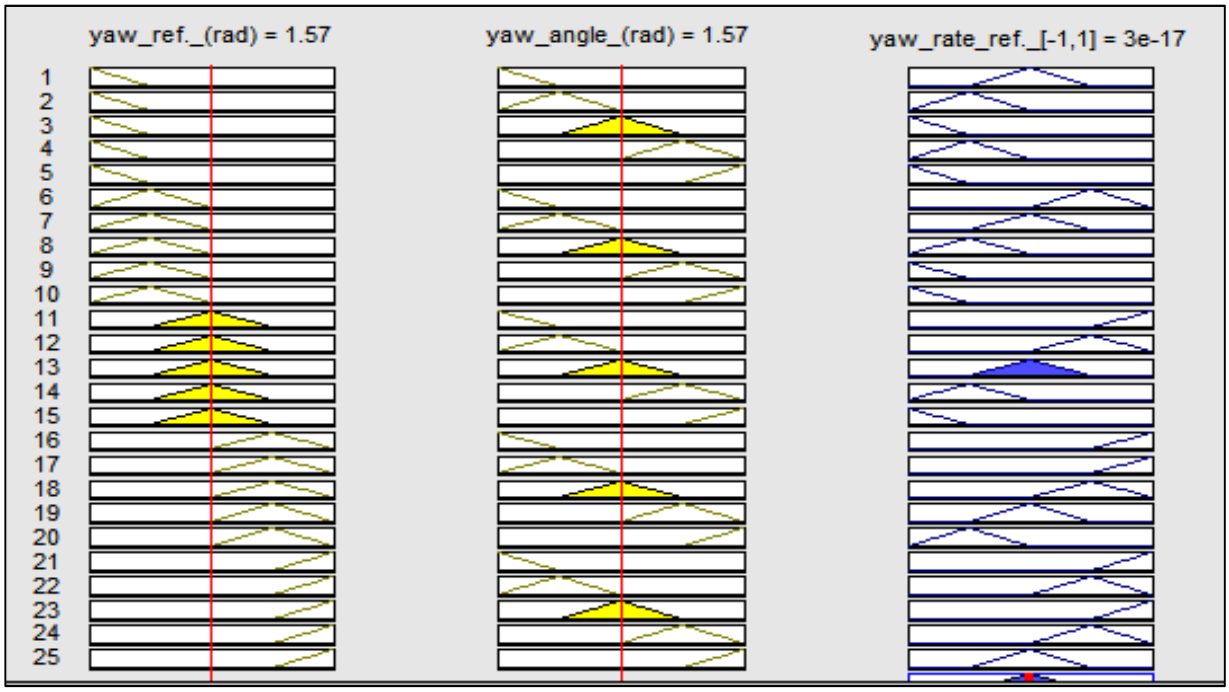


Figure V.25. Règles établies pour le contrôleur de roulis.

La surface des règles floues pour le contrôleur de Lacet :

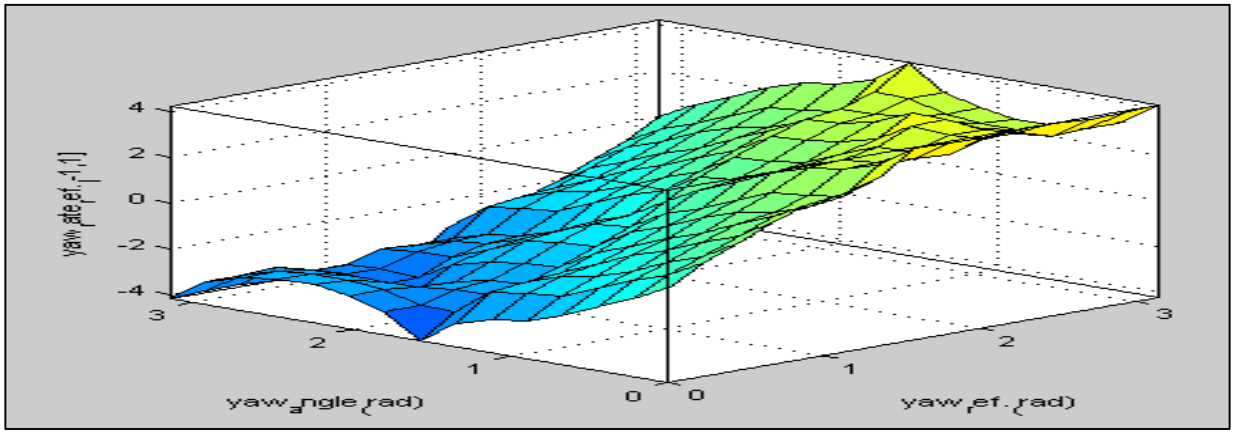


Figure V.26. La surface des règles floues pour le contrôleur de Lacet.

V.3. RESULTATS

V.3.1. Etude de Simulation

Afin de rendre la simulation similaire au fonctionnement du drone réel, des temps de retard ont été ajoutés. La durée d'échantillonnage utilisée dans la simulation est de 0,065 seconde pour obtenir l'état du système. Un délai de 0,26 seconde est ajouté compte tenu du temps nécessaire à la communication entre le drone et l'ordinateur. Un temps de simulation de 0,005 s est utilisé pour l'exécution de la simulation.

Il est important de noter que le système est considéré comme idéal, de sorte que pour cette mise en œuvre, les effets de l'air, de l'inertie et des changements d'intensité lumineuse dans l'environnement sont négligés.

V.3.2. Etude sur Plateforme réelle

Dans le cas de la mise en œuvre sur la plate-forme réelle, nous avons fixé une durée d'échantillonnage générale de 0,065 s.

Pour le contrôle de la plate-forme réelle, il était nécessaire de prendre en compte les changements d'intensité lumineuse dans l'environnement et les caractéristiques du sol dans la zone de vol. Cela est dû au fait que le quadrotor AR.Drone, dispose de certaines fonctions établies dans l'API pour son emplacement, il utilise aussi son caméra inférieure, pour savoir s'il se déplace dans une direction quelconque.

Considérant ce qui précède, si la surface ne présente pas de variations ou si en raison de l'intensité de la lumière, elle ne parvient pas à détecter les variations, le système ne détectera pas certains déplacements dans lesquels les variations d'angles sont très faibles et la caméra ne détecte pas de variation entraînant le maintien de la plate-forme, ce qui constitue un problème d'instabilité dans le système.

Dans cette mise en œuvre, des caractéristiques telles que l'aérodynamique du système ou les effets de l'air n'ont pas été prises en compte. De plus, il existe des variations ou de petites oscillations dans les mouvements effectués. C'est parce que le système ne sait plus où il se situe en raison de très petites perturbations telles que celles mentionnées au paragraphe précédent.

V.3.3. Les résultats

Dans cette section, nous présentons certains des résultats obtenus avec les différentes trajectoires utilisées pour valider notre approche. Tout d'abord nous affichons les résultats de la simulation puis les résultats obtenus à partir de la plate-forme réelle. Dans les deux cas, les résultats sont générés à partir de trajectoires simples dans lesquelles il y a plusieurs mouvements simultanément car ils utilisent plusieurs variables de contrôle.

Pour chaque trajectoire, il existe un script spécifique dans lequel les points sont établis, grâce auxquels le système, au moyen d'un contrôle intelligent à base de logique floue, peut se déplacer pour se rapprocher du chemin sélectionné. Notons que plus le nombre de points existant dans la référence est grand, plus le suivi se rapproche de la courbe des résultats désirés.

En analysant la trajectoire d'un cercle à titre d'exemple, avec un plus grand nombre de points, le résultat du suivi ressemblera davantage au cercle de référence, tandis que si le nombre de points est petit, la trajectoire aura tendance à ressembler à un cercle polygone par exemple, avec 6 points, cela ressemblera à un hexagone ou, avec 8 points, à un octogone.

Tous les déplacements effectués utilisent des mesures absolues par rapport au point initial. Dans les graphiques illustrant la mise en œuvre dans la plate-forme réelle, il est nécessaire de considérer que les déplacements initiaux et finaux du système ne font pas partie de la trajectoire ou du mouvement. Et dans la simulation, il n'y a que des déplacements initiaux, mais ils ne font pas non plus partie de la trajectoire considérée.

V.3.3.1. Trajectoire carrée

Les paramètres de la trajectoire carrée sont indiqués sur le tableau suivant.

Tableau V.5. Coordonnées désirés de la trajectoire carrée.

X	0	5	5	0	0	0	0	0	0
Y	0	0	5	5	2	1.5	1	0.5	0
Z	2	2	2	2	2	1.5	1	0.5	0
Lacet	0	0	0	0	0	0	0	0	0
Temps	3	3	3	3	3	3	3	3	3

Vous trouverez ci-dessous les résultats de simulation obtenus pour les mouvements de trajectoire carrée.

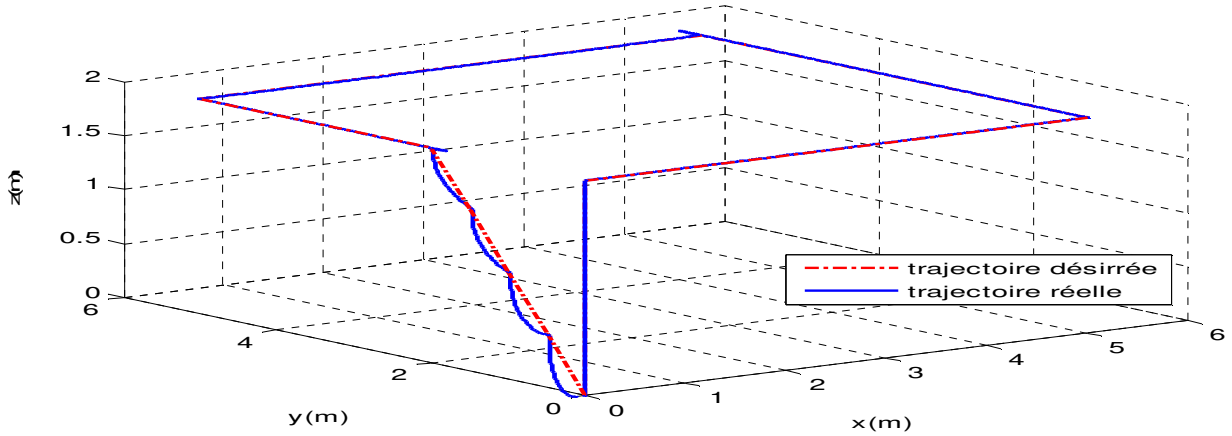
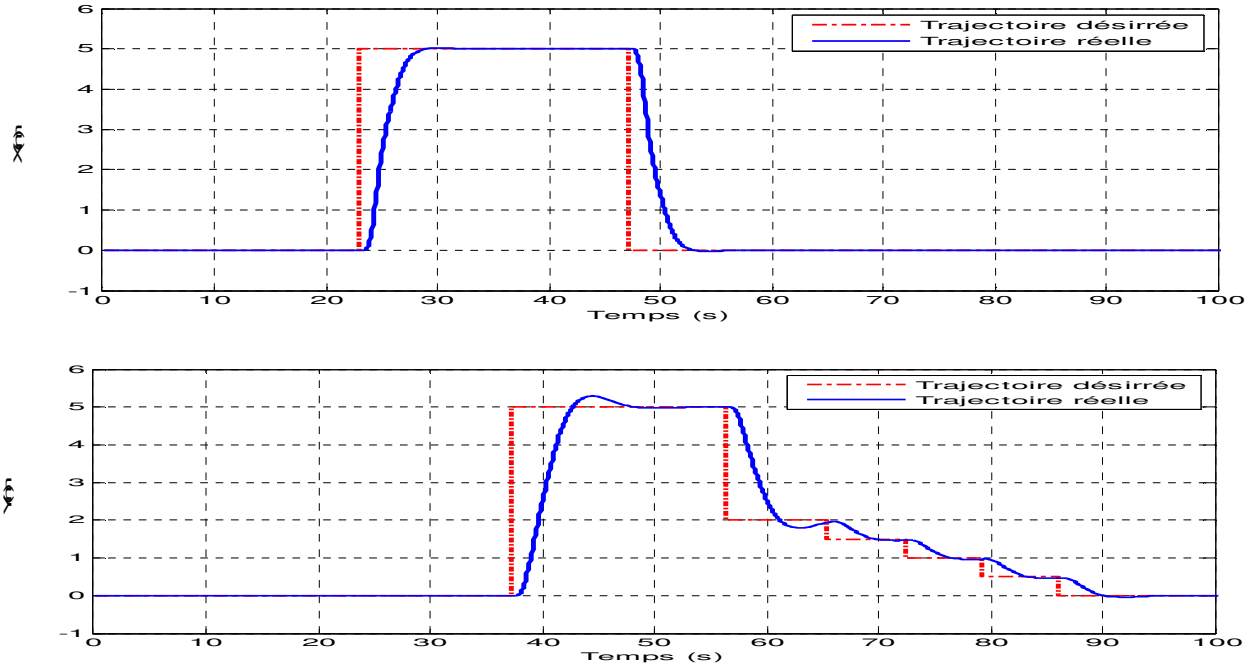


Figure V.27. Trajectoire carrée en 3D (Résultat de simulation).



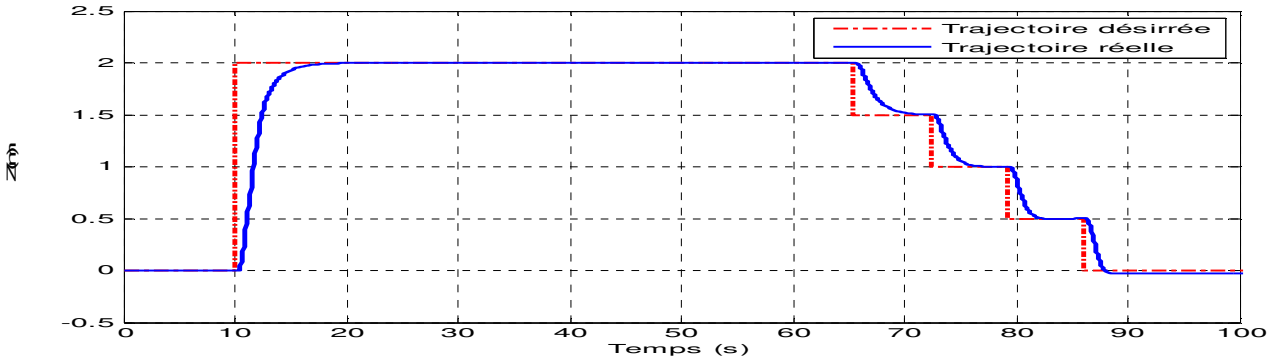


Figure V.28. Courbes temporelles pour trajectoire carrée (Résultat de simulation).

On observe dans chacune des figures deux graphes, dont le premier (figure V.27) est le graphe du déplacement du système en 3D et le second (figure V.28) est le graphe des axes correspondant au mouvement par rapport au temps.

Pour les déplacements en Z, on observe à la figure V.27 que le système monte à 2 mètres comme point de départ, une fois cette hauteur atteinte, le quadrotor se déplace de 5 mètres sur X puis il reste dans cette position pendant une durée de 3 secondes.

Pour les déplacements en Y, des tests similaires à ceux des déplacements en X ont été réalisés la quadrotor monte à 2 mètres d'altitude comme point de départ, une fois cette hauteur atteinte, il avance de 5 mètres en Y puis il reste dans cette position pendant 3 secondes avant de commencer la procédure d'atterrissage. Les tests d'altitude peuvent être observés à la Figure V.28 où on peut voir que le système monte et descend, en modifiant sa hauteur pour passer par les points indiqués sur l'axe des Z. Le système a démarré de 0 à 2 m, par la suite il commence la descente par paliers de 0,5 m, puis il revient à son point de départ.

À la suite des trajectoires, dans lesquelles des mouvements de trajectoire carrée ont été effectués mais maintenant dans la plate-forme réelle AR.Drone, les graphiques suivants ont été obtenus.

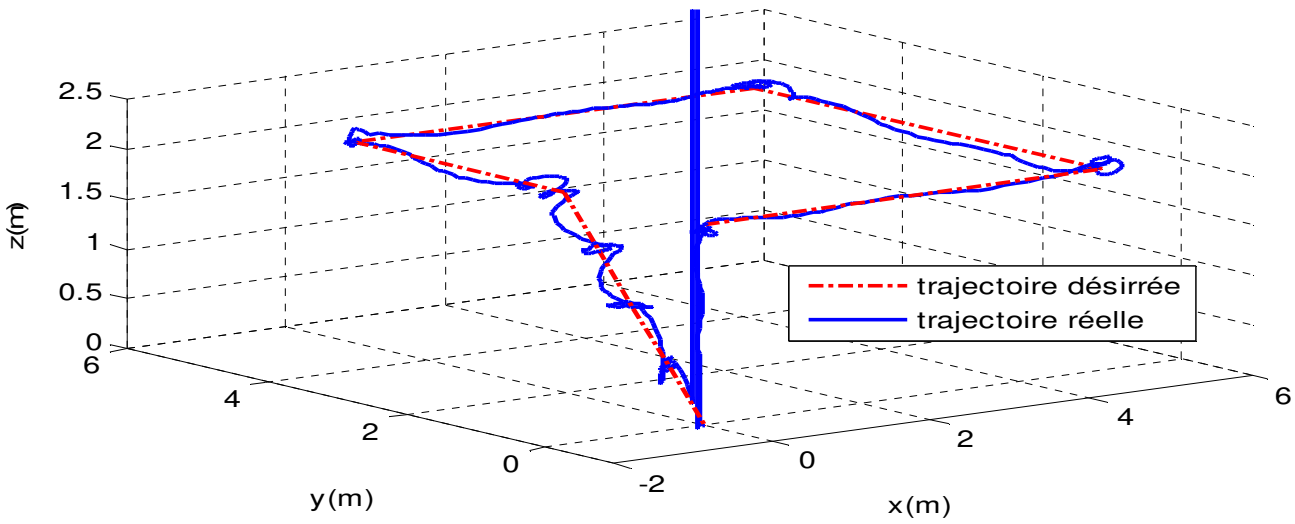


Figure V.29. Trajectoire carrée en 3D (Test Réel).

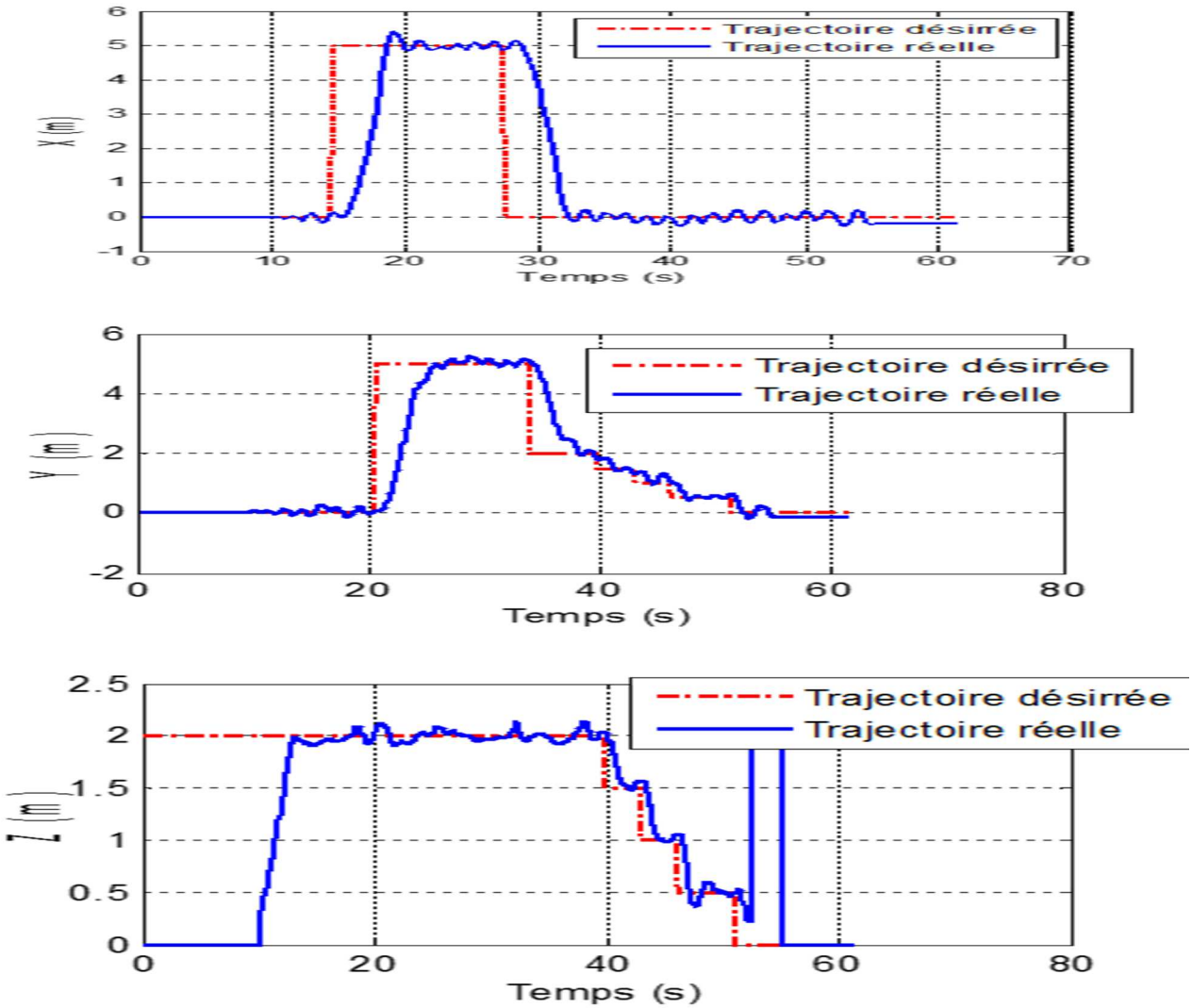


Figure V.30. Courbes temporelles pour trajectoire carrée (Test réel).

Pour observer la réponse des déplacements en X en fonction du temps pour l'AR.Drone, on peut voir sur la figure V.29 que le système monte à 2 mètres de hauteur comme point de départ, une fois sur cette hauteur atteinte, il avance de 5 mètres sur l'axe des X pendant 6 secondes, puis il revient à son de son point de départ.

Pour les déplacements en Y, le quadrotor monte à 2 mètres de hauteur comme point de départ et il se déplace de 5 mètres en Y puis il commence la descente par pas de 0,5 mètre pour revenir à la fin au point de départ (voir figure V.29). Dans les deux déplacements, sur les axes X et Y, on peut voir que le système passe d'un point à un autre sur des points établis dans le tableau de waypoint, en maintenant sa hauteur approximative à un mètre.

Sur la plate-forme réelle, un test d'élévation similaire a été effectué, dans lequel le quadrotor devait passer par 4 points d'altitude : de 2 m, à 1,5 m puis et 1 m et en dernier à 0,5 m en maintenant une période de 3 secondes pour chaque palier (voir figure V.30).

Dans les graphiques des mouvements pour la trajectoire carrée, on peut voir que le quadrotor parvient à atteindre les points spécifiques décrits dans le tableau du Waypoint, même si au fil

des points, il a des mouvements saccadés et perd momentanément sa position. Ces mouvements saccadés sont considérés comme dus à des agents externes tels que ceux mentionnés ci-dessus qui contrairement à la simulation, sont omis et les performances du système produisent une trajectoire proche de la référence.

V.3.3.2. Trajectoire circulaire avec 16 points

Les paramètres de la simulation sont indiqués sur le tableau suivant :

Tableau V.6. Coordonnées désirés de la trajectoire circulaire avec 16 points.

X	0	4	3.7	2.8	1.5	0	-1.5	-2.8	-3.7	-4	-3.7	-2.8
Y	0	0	1.5	2.8	3.7	4	3.7	2.8	1.5	0	-1.5	-2.8
Z	2	2	2	2	2	2	2	2	2	2	2	2
Lacet	0	0	0	0	0	0	0	0	0	0	0	0
Temps	3	3	3	3	3	3	3	3	3	3	3	3

X	-1.5	0	1.5	2.8	3.7	2.7	1.8	0.9	0
Y	-3.7	-4	-3.7	-2.8	-1.5	-1.1	-0.75	-0.4	0
Z	2	2	2	2	2	1.5	1	0.5	0
Lacet	0	0	0	0	0	0	0	0	0
Temps	3	3	3	3	3	3	3	3	3

Le modèle obtenu a été utilisé pour identifier le système, afin que le quadrotor puisse suivre une trajectoire sous forme d'un cercle avec 2 mètres de hauteur, avec un nombre de points suffisant pour former une allure de cercle. Dans la figure V.31, vous pouvez observer le suivi de trajectoire circulaire en simulation avec 16 points.

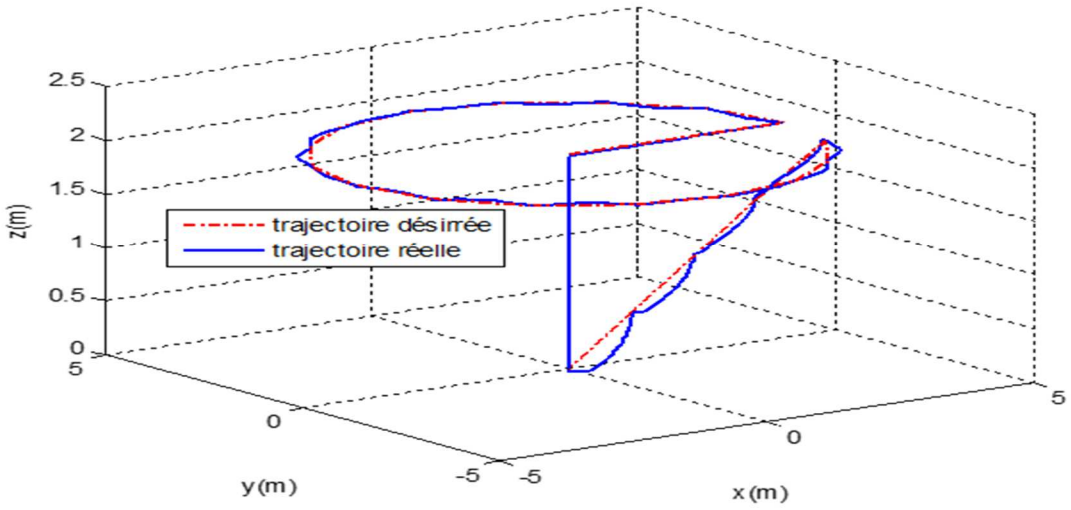


Figure V.31. Courbe circulaire en 3D (test de simulation).

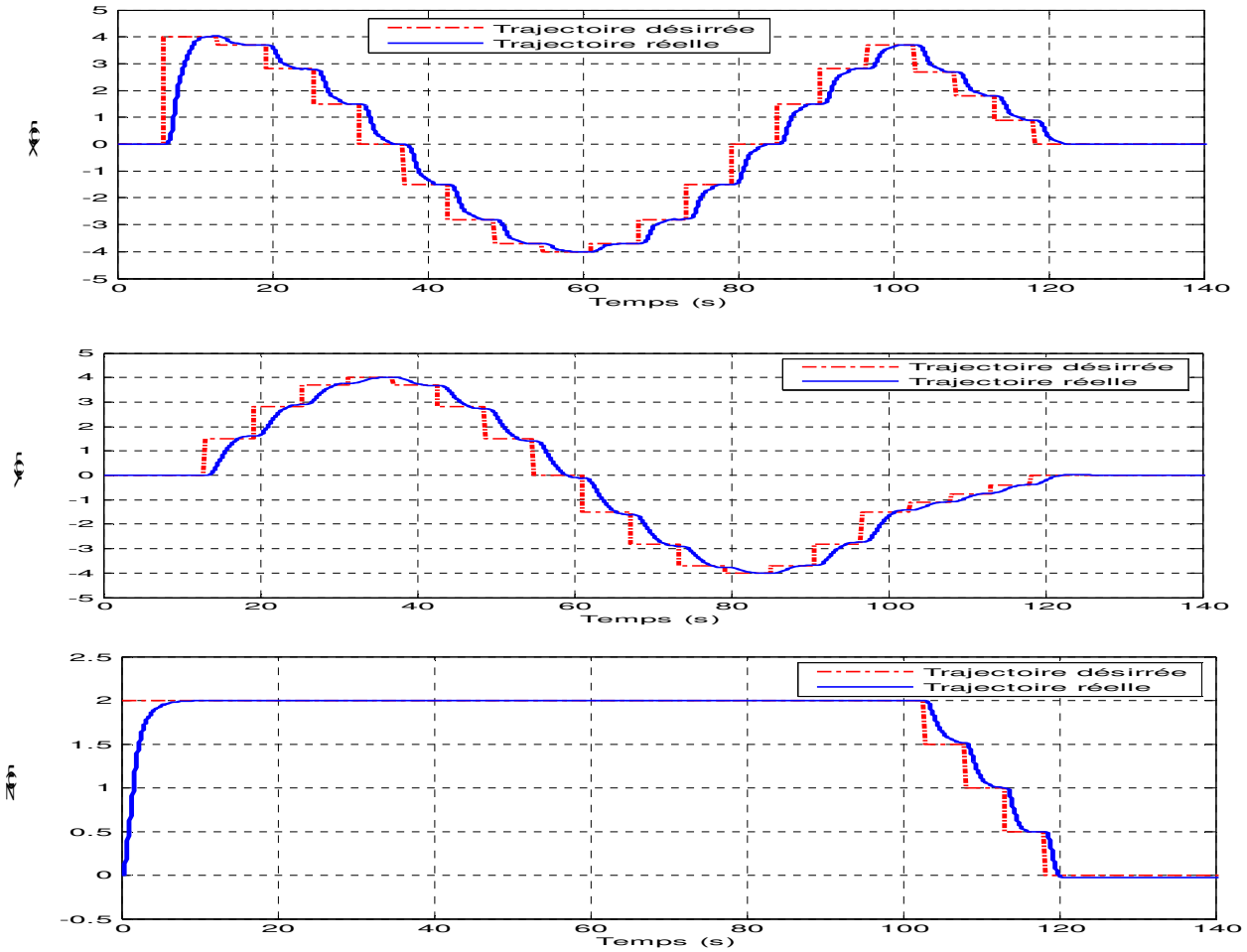


Figure V.32. Courbes Circulaire temporelles (test de simulation).

Dans ce test, une trajectoire circulaire constituée de 16 points a été imposée au quadrotor, on constate d'après les résultats obtenus que le drone a respecté le suivi de trajectoire.

Dans les graphiques de simulation pour la trajectoire circulaire, nous pouvons constater que plus le nombre de points de référence est grand, plus la réponse du système se traduit par une trajectoire plus proche de celle souhaitée. Par exemple, dans les figures V.31 en 3D, on peut voir que la réponse du système était très similaire à la trajectoire de référence. Cependant, il convient de noter que le comportement attendu du quadrotor est qu'il passe par les points de référence imposés.

Pour observer la réponse des déplacements en X en fonction du temps, on peut voir sur la figure V.32 que le système monte à 2 mètres de hauteur comme point de départ, une fois sur cette hauteur, il réalise un cercle en avançant de 3 mètres sur l'axe des X, puis il revient à son point de départ.

Pour les déplacements en Y, le l'AR.Drone monte à 2 mètres de hauteur comme point de départ et il se déplace de 1,5 mètres en Y puis il commence à modifier son déplacement pour former un cercle (voir figure V.32). Dans les deux déplacements, sur les axes X et Y, on peut voir que le système passe d'un point à un autre sur des points établis dans le tableau de waypoint, en maintenant sa hauteur approximative à 2 mètres. Pour valider le comportement du système dans

l'environnement réel sur l'AR.Drone et avec les mêmes points utilisés dans la simulation, nous avons imposé au quadrotor les mêmes valeurs des waypoints, les graphiques suivants ont été obtenus :

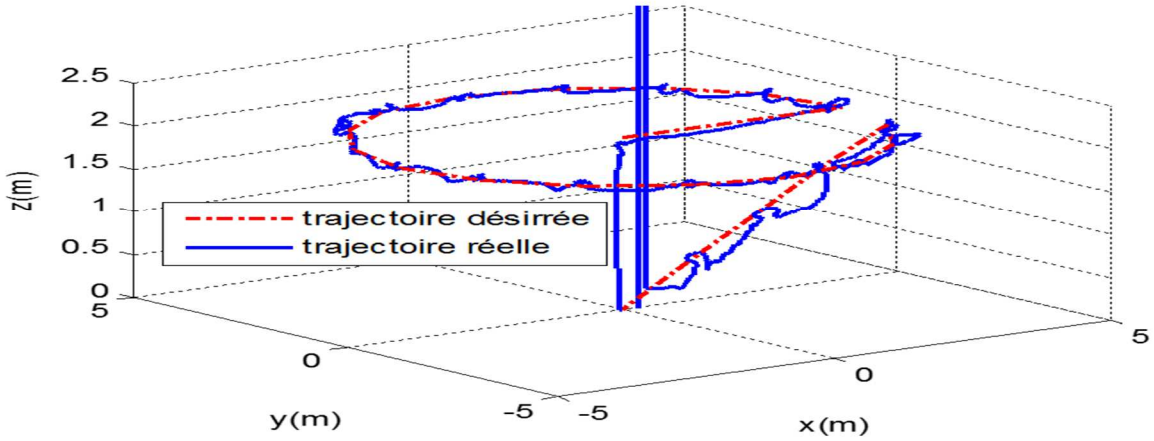


Figure V.33. Courbe Circulaire en 3D (Test Réel).

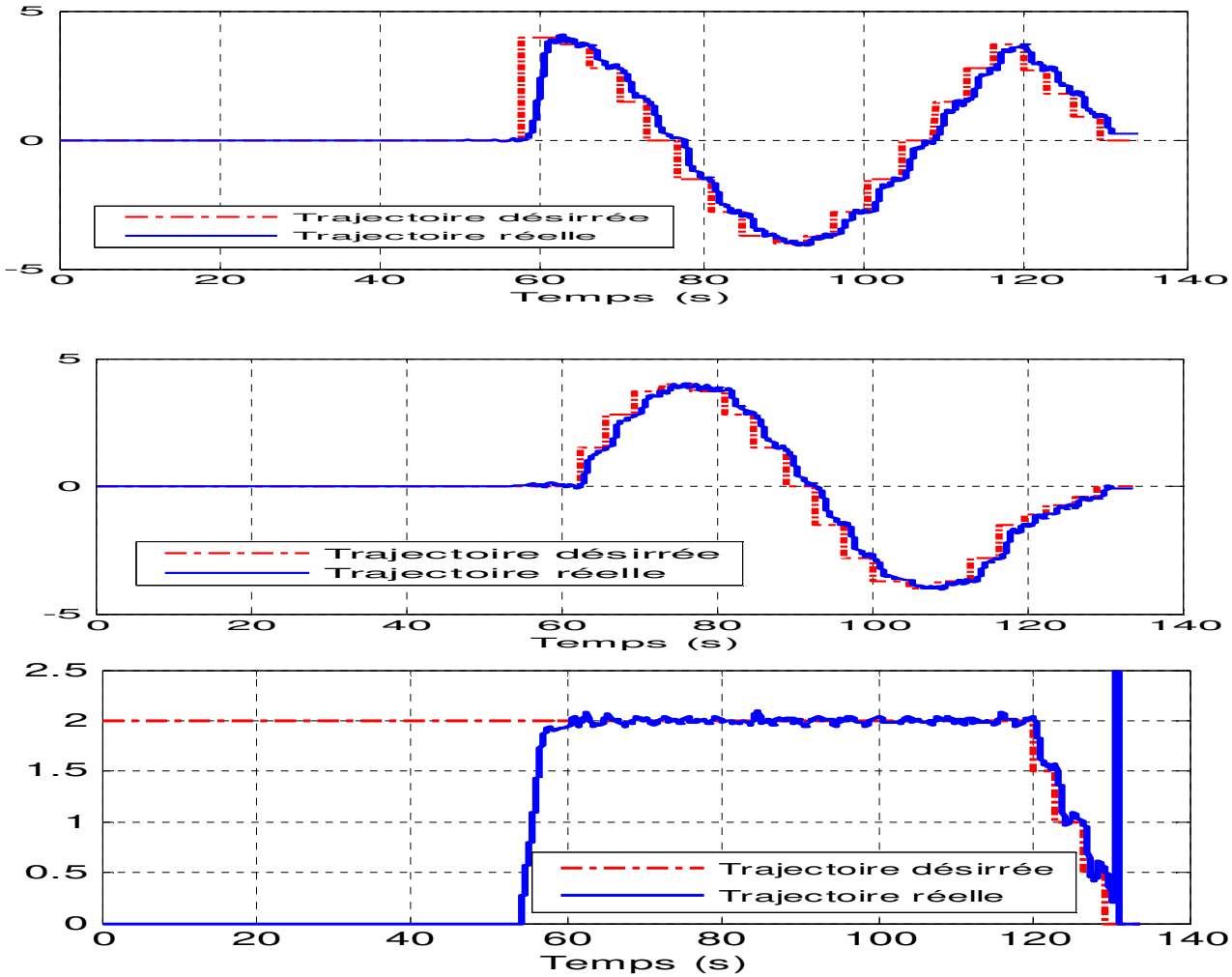


Figure V.34. Courbes Circulaire temporelles (Test réel).

Le test pratique pour la trajectoire circulaire prouve une bonne poursuite, les résultats sont convaincants.

Pour valider le comportement du système avec la même trajectoire et les mêmes nombres de points, des tests ont été effectués sur des trajectoires circulaires.

Sur la courbe des trajectoires en forme de cercles en 3D (voir figure V.33), on peut voir que le quadrotor s'est déplacé sur toute la trajectoire en passant par les points de référence. Nous remarquons aussi une petite marge d'erreur pour chacune des variables de références. Ce qui explique pourquoi parfois le système ne passe pas exactement par le point, mais compte tenu des erreurs de mesure lors de la détection, le quadrotor passe à proximité des points établis dans le tableau de waypoint.

Bien que dans les graphiques illustrant le suivi des trajectoires par la plate-forme réelle, il semble que celle-ci se déplace de manière aléatoire en raison des oscillations. De telles oscillations se produisent principalement dans le plan XY car, comme on peut le voir à la figure V.34, le système est capable de maintenir sa hauteur et en même temps de faire des déplacements dans le plan.

V.3.3.3. Trajectoire spirale

Les paramètres de simulation sont introduits dans l'ANNEXE 3.

Un waypoint a été utilisé afin que le quadrotor puisse suivre une trajectoire de forme spirale avec 7 mètres de hauteur, constituée d'une géométrie complexe. On observe dans chacune des figures deux graphes, dont le premier (figure V.35) est le graphe du déplacement du quadrotor en 3D et le second (figure V.36) est le graphe correspondant au mouvement des axes par rapport au temps.

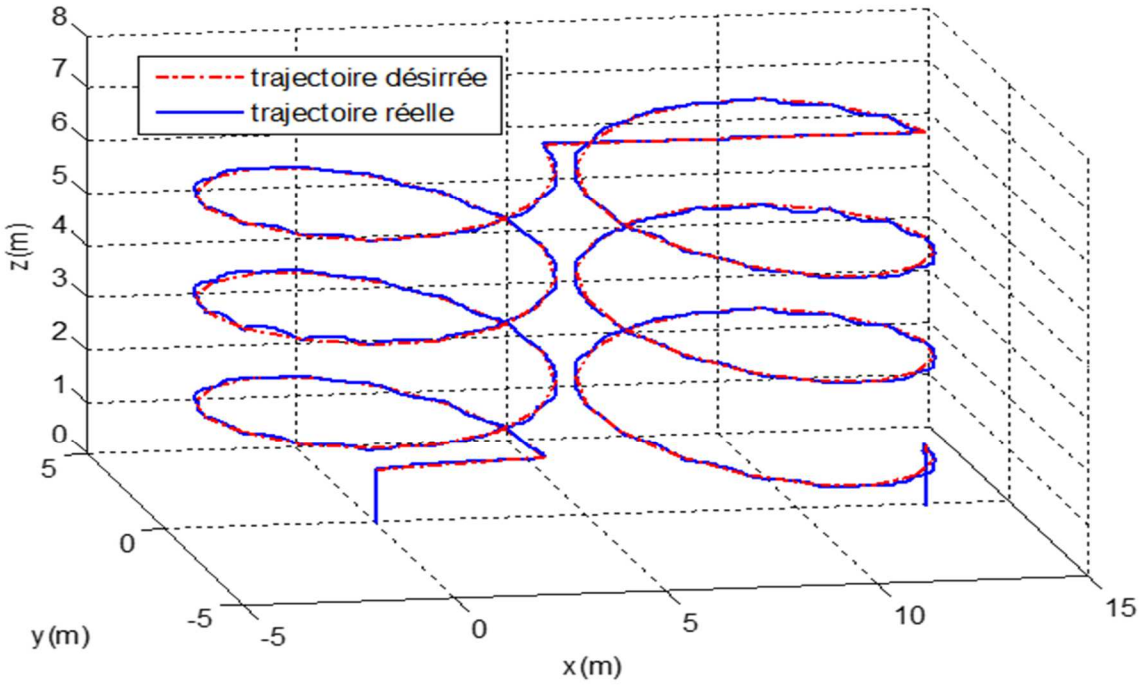


Figure V.35. Courbe Spirale en 3D (Test de simulation).

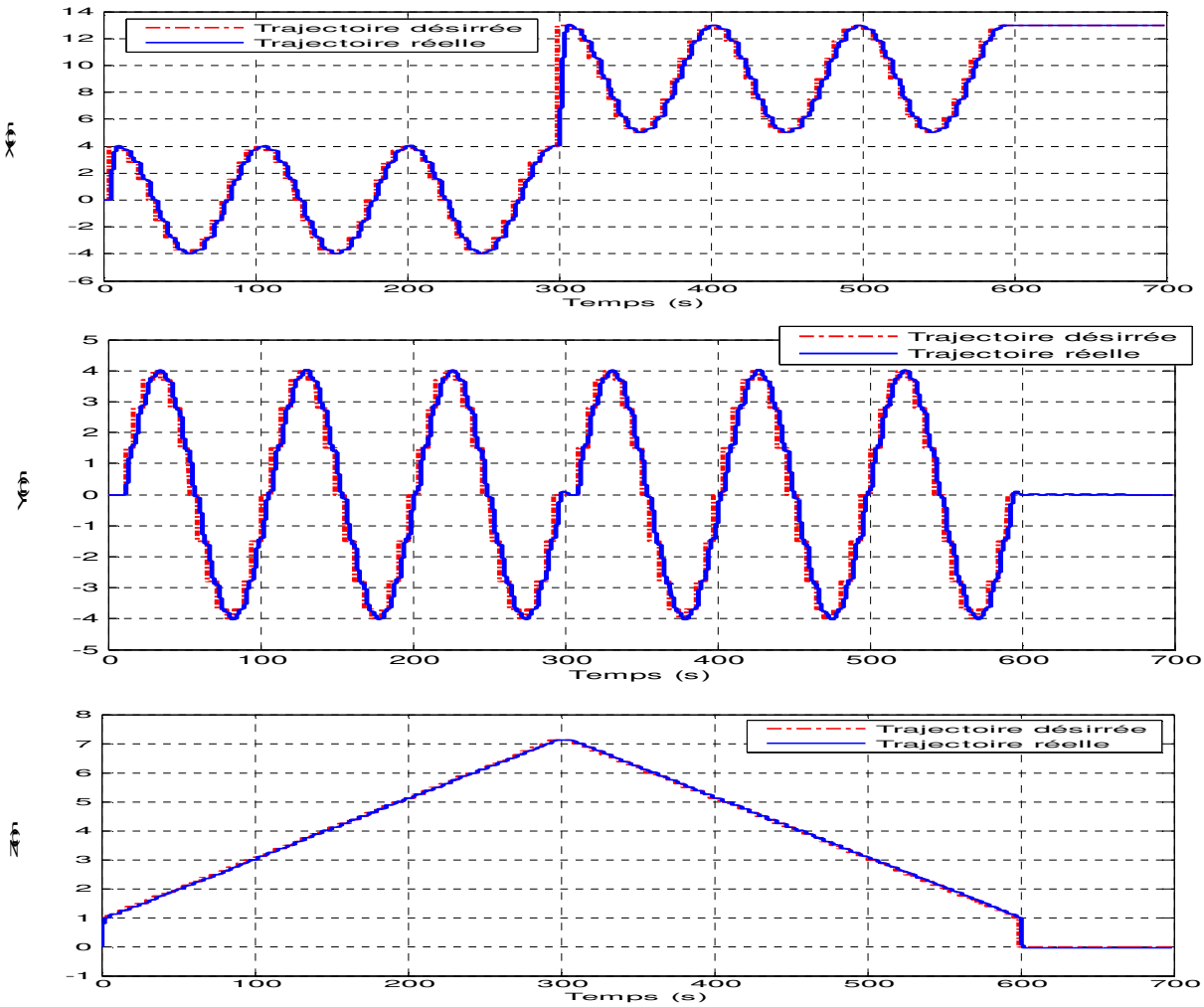


Figure V.36. Courbes Spirale temporelles (Test de simulation).

En analysant le résultat obtenu à partir de la figure V.35, nous pouvons observer que le quadrotor a suivi la trajectoire établie comme référence avec sucées. Pour les déplacements en Z, on observe à la figure en 3D que le système monte à 7 mètres comme point de départ, une fois cette hauteur atteinte, le quadrotor fait des déplacements suivant les axes X et Y pour former une géométrie spirale.

Pour la forme spirale le quadrotor a dû faire plusieurs manœuvres suivant les trois axes X, Y et Z. La réponse du contrôleur flou est illustrée dans la figure V.36 où nous pouvons remarquer que le drone a suivi la trajectoire imposée dans les trois axes X, Y et Z. d'après les résultats de simulation nous pouvons dire que les résultats sont très convaincants malgré la complexité de la trajectoire.

Pour valider le comportement du quadrotor avec la même trajectoire de forme spirale, des tests ont été à nouveau effectués pour valider les résultats de simulation. Les graphiques suivants ont été obtenus :

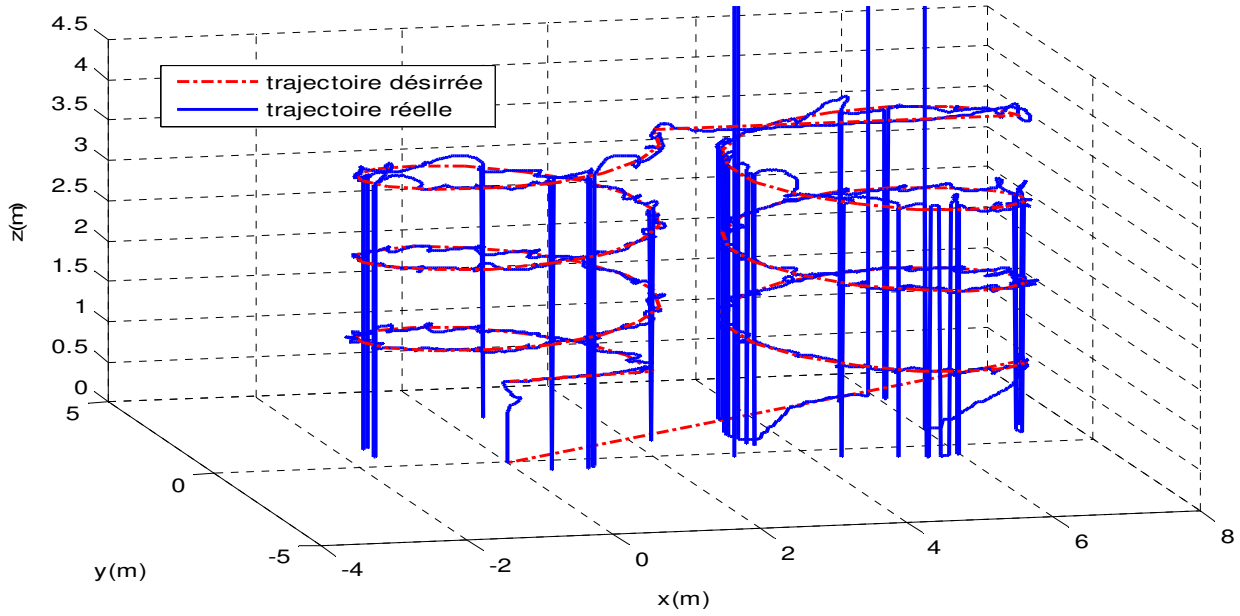


Figure V.37. Courbe Spirale en 3D (Test réel).

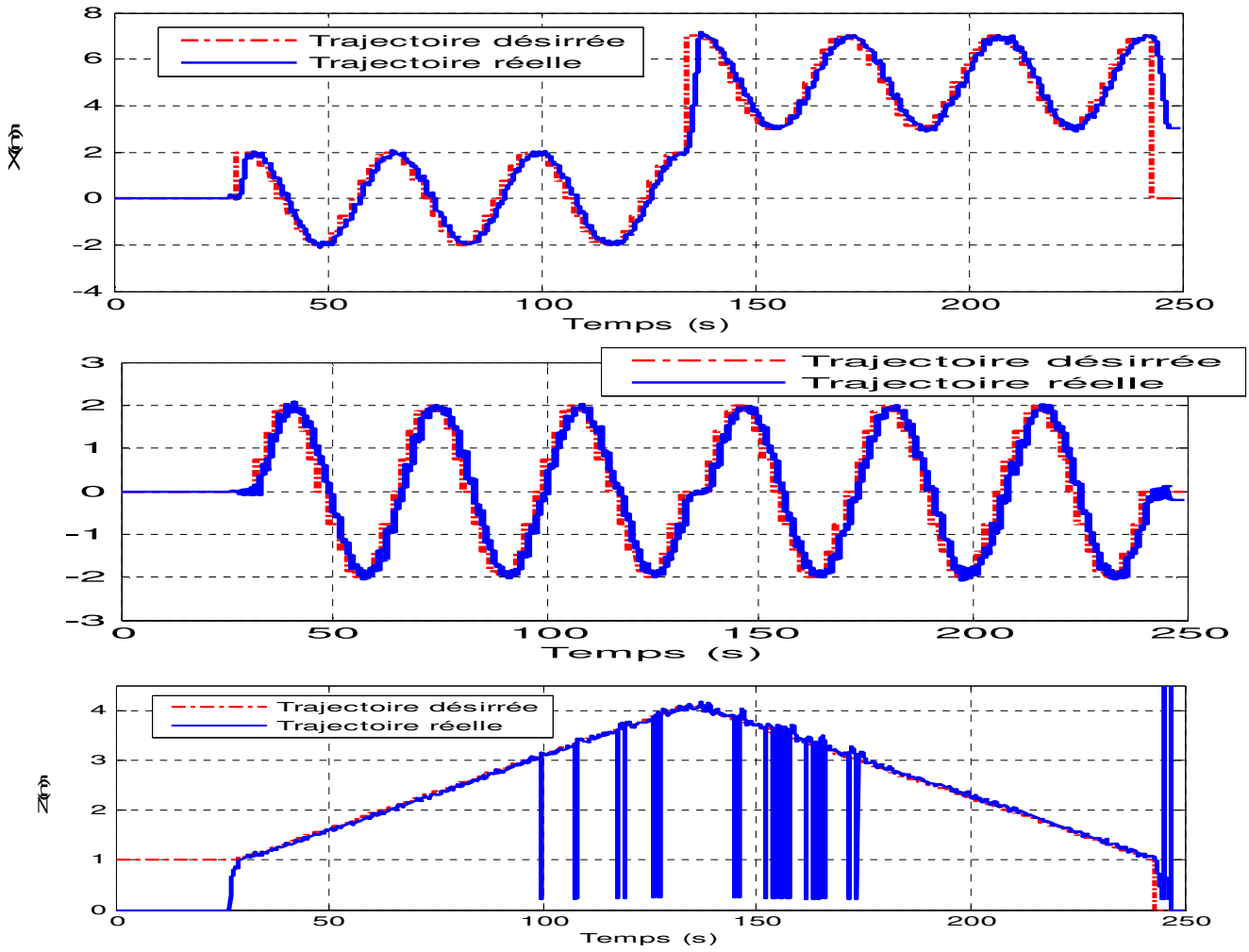


Figure V.38. Courbes Spirale temporelles (Test réel).

La figure V.37 montre que le système est capable de suivre la trajectoire la plus complexe considérée dans ce travail, une spirale. Dans cette figure, on comprend que plus le quadrotor est en hauteur, plus le mouvement est difficile. En effet, l'utilisation de l'hélice fournit moins de poussée au système, cet aspect influence également la stabilité et les oscillations de toutes les trajectoires effectuées. On constate aussi que la complexité augmente lors d'une spirale, car dans cette trajectoire, les trois axes étaient constamment modifiés, mais le système a néanmoins fait le suivi correctement selon le tracé en 3D.

Même s'il est vrai que certaines perturbations sont observées, le système reste robuste dans le suivi des trajectoires spirales en utilisant l'AR.Drone. Pour les mêmes données utilisées en simulation nous avons obtenu des résultats pratiques très similaire à la trajectoire de référence, ceci est visible à la figure V.38.

V.4. CONCLUSION

L'objectif principal de ce chapitre était de développer un contrôleur flou pour un quadrotor, doté d'un accéléromètre et d'un gyroscope à trois axes, ainsi que d'un capteur à ultrasons, dans le but de suivre une trajectoire de référence définie d'avance.

Le module de communication a été conçu, avec lequel il était possible de contrôler la plate-forme réelle AR.Drone 2.0. Ce module est basé sur des blocs de communication inclus dans le kit de développement de Simulink V1.1.

En général, nous concluons que, malgré les problèmes rencontrés dans le développement, le contrôle d'un quadrotor pour le suivi de trajectoire est possible à l'aide d'un contrôleur à base de logique floue.

Conclusion Générale

CONCLUSION GENERALE

La détection d'objets est l'un des domaines importants de la vision par ordinateur. Le cœur de la détection d'objet est la reconnaissance précise d'un objet dans les images. Des applications telles que la recherche ou la reconnaissance d'images utilisent une méthode de détection d'objet comme partie principale. Aujourd'hui, le problème de détection d'objet est toujours classé comme problème ouvert en raison de la complexité de l'image ou de l'objet lui-même.

Les véhicules aériens sans pilote (UAV) ont une large gamme d'applications - militaires et civiles. Bien qu'ils soient capables d'exécuter automatiquement une partie, voire la totalité de la mission, ils n'ont pas toute l'autonomie. Le niveau d'autonomie peut être augmenté de manière substantielle au moyen de systèmes d'évitement d'obstacles et suivi de cible.

L'AR.Drone de Parrot sera utilisé comme plate-forme dans cette thèse. Il s'agit d'un véhicule aérien de type quadrotor bon marché doté de capacités impressionnantes. Il offre un contrôle de vol stable et diffuse la vidéo en direct sur un système distant à partir des caméras frontale et inférieure intégrées. Notre intention est d'exploiter la capacité de la vidéo en direct et la plate-forme AR.Drone pour la navigation en utilisant la vision par ordinateur afin de suivre une trajectoire prédéfinie d'avance et le suivi de cible en utilisant des algorithmes intelligents. La localisation et les commandes basées sur la vision seront transmises à un système intelligent pour contrôler la position et le mouvement de l'AR.Drone.

Dans le chapitre trois de cette thèse, nous avons conçu des contrôleurs flous pour contrôler un quadrotor afin de suivre une cible et de réaliser un évitement d'obstacle. Pour la conception des contrôleurs nous avons utilisés la combinaison des outils open source FuzzyLite, ROS Gazebo avec Tum_ARdrone Simulator. Pour évaluer le succès ou l'échec des algorithmes implémentés, le drone devrait être capable de naviguer avec succès d'un point de départ vers une cible, tout en évitant les obstacles le long du chemin. De plus, le drone devrait suivre la cible pendant qu'il se déplace dans l'environnement de test. Les résultats de la simulation et les tests réels sur l'AR.Drone ont montrés des performances acceptables du contrôleur flou. Lors d'études de simulation et d'essais en vol nous constatons que le système permettant d'éviter les obstacles visuels peut être utilisé dans les applications réelles.

Le suivi de cible est d'un grand intérêt pour des applications dans divers domaines des Drones, tels que le système de surveillance, le divertissement, pour n'en citer que quelques-unes. Le système de vision artificielle est basé sur des techniques informatiques intelligentes inspirées du modèle neuronal biologique de l'être humain. Bien connues sous le nom de réseaux de neurones artificiels (RNA), ces techniques se caractérisent principalement par leur capacité à apprendre par l'expérience, et à s'adapter aux conditions défavorables et à être tolérantes au bruit. Ces fonctionnalités font de l'ANN un excellent domaine d'application et le conduisent à résoudre des problèmes réels tels que la classification, l'identification et le traitement d'images numériques. Pour suivre une cible donnée, deux tâches doivent être implémentées. La première consiste à reconnaître la cible parmi de nombreux types d'objets. Une autre consiste à déterminer la pose relative de la cible, telle que la distance et l'angle relatifs. Les difficultés de repérage d'objets résultent d'un certain nombre de :

- Manœuvres rapides de la cible ;

- Bruit dans les images et communication entre la caméra et le récepteur ;

- Plusieurs tâches de commutation telles que l'évitement d'obstacles lors du suivi d'une cible ;

- La nécessité de suivre la cible lorsque l'UAV est proche de la cible ;

- Traitement de données en temps réel.

Dans le chapitre quatre de cette thèse, nous avons présenté la formulation d'un réseau neuronal permettant à un véhicule aérien sans pilote (UAV) de type quadrotor de suivre une cible. L'application potentielle réside dans des missions telles que la surveillance et la reconnaissance. Les principales contributions de ce chapitre sont les suivantes :

- La détection d'objet basée sur ANN est extrêmement exigeante en termes de calcul, nécessitant généralement des unités de traitement graphique (GPU) haut de gamme et trop de puissance. Dans notre application, des expériences sur des données réelles montrent les résultats prometteurs obtenus sur un processeur standard et ouvrent la voie à de futures recherches dans cette direction ;

- Le suivi de cible est basé sur un réseau de neurone simple MLP pouvant être facilement généralisé pour suivre tout motif visuel dans des scènes dynamiques ;
- Ce chapitre vise à utiliser un mécanisme de contrôle visuel pour contrôler un quadrotor à la recherche d'une cible. La nature non linéaire du quadrotor, d'une part, et la difficulté d'obtenir un modèle exact pour celui-ci, d'autre part, constituent deux défis sérieux dans la conception d'un contrôleur pour ce drone. Outre les deux problèmes mentionnés, un autre problème qui apparaît en raison de la nature d'une cible est l'incertitude qui existe dans l'image de la cible. En utilisant un réseau neuronal artificiel, le contrôleur neuronal MLP a été conçu pour un quadrotor à la recherche d'une cible.

Dans le chapitre cinq, un contrôleur flou est proposé pour stabiliser le fonctionnement d'un quadrotor dans le suivi de trajectoire. Le contrôleur flou est réalisé pour traiter les signaux du quadrotor et générer une réponse. Pour cela, 4 pilotes ont été développés, dont les entrées sont : la hauteur et les trois angles d'orientation pour chacun. Pour la mise en œuvre du contrôleur flou sur la plate-forme réelle et dans la simulation, nous avons utilisé la boîte à outils Toolbox de Fuzzy Logic et Real-Time Windows Target de Matlab / Simulink. D'après les expériences qui ont été effectuées, le drone semble être capable de suivre une trajectoire donnée avec différentes positions initiales.

En général, nous concluons que, malgré les problèmes rencontrés dans le développement de cette thèse, le contrôle d'un quadrotor est possible à l'aide des contrôleurs intelligents. Compte tenu de l'évolution de ce projet et en fonction des expériences réalisées et des résultats obtenus, certaines tâches possibles sont considérées pour poursuivre le développement de ce travail. Certains d'entre eux sont énumérés ci-dessous :

- Ajouter un autre type de capteurs pour mesurer la position qui permettrait d'obtenir un contrôle plus précis, car la localisation serait précise ;
- Améliorez l'estimation de la position du système pour obtenir un contrôle avec une erreur minimale ;
- Rapprocher les trajectoires au moyen d'un algorithme permettant aux mouvements d'être plus fidèles à la trajectoire de référence souhaitée ;
- Étendre cette recherche à l'étude de la planification de trajectoire, permettant au système de planifier et de suivre une trajectoire de manière autonome.

BIBLIOGRAPHIE

[1] JR Wilson. “UAV worldwide roundup 2009”. Aerospace America, 47(4), 2009.

[2] Richard P Schwing. “Unmanned aerial vehicles-revolutionary tools in war and peace”, Technical report, DTIC Document, 2007.

[3] Zak Sarris and STN ATLAS. Survey of UAV applications in civil markets (june 2001). In The 9 th IEEE Mediterranean Conference on Control and Automation (MED'01), 2001.

[4] Farid Kendoul. “Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems”, Journal of Field Robotics, 29(2):pp. 315-378, 2012.

[5] Valavanis, Kimon P., ed. “Advances in Unmanned Aerial Vehicles: State of the Art and the Road to Autonomy”, Dordrecht: Springer, 2007.

[6] C. Eck, Navigation Algorithms with Applications to Unmanned Helicopters. PhD thesis, ETHZ, 2001.

[7] R.V. Jategaonkar. “Flight vehicle system identification: a time domain methodology”. American Institute of Aeronautics and Astronautics, 2006.

[8] L. Beji K. M. Zemalache and H. Marref. “Control of an under-actuated system: Application to a four rotors rotorcraft”. IEEE International Conference on Robotics and Biomimetics, pages 404 – 409, 2005.

[9] A. Mokhtari and A. Benallegue. “Dynamic feedback controller of euler angles and wind parameters estimation for a quadrotor unmanned aerial vehicle”. Proceedings of the 2004 IEEE International Conference on Robotics and Automation, pages 2359 – 2366, 2004.

[10] R. Lozano P. Castillo and A. Dzul. “Stabilization of a mini rotorcraft with four rotors”. IEEE Control Systems Magazine, pages 45 – 55, 2005. 2

[11] A. Abichou L. Beji and K. M. Zemalache. “Smooth control of an x4 bidirectional rotors flying robot”. Fifth International Workshop on Robot Motion and Control, pages 181 – 186, 2005.

[12] I.D. Cowling, O.A. Yakimenko, J.F. Whidborne, and A.K. Cooke. “A prototype of an autonomous controller for a quadrotor uav”. In European Control Conference, pages 1–8, 2007.

[13] B. Whitehead and S. Bieniawski. “Model Reference Adaptive Control of a Quadrotor UAV”. In Guidance Navigation and Control Conference 2010, Toronto, Ontario, Canada, 2010. AIAA.

[14] H. Nijmeijer and A. van der Schaft. “Nonlinear Dynamical Control Systems”. Springer-Verlag, 1990.

[15] A. Palomino S. Salazar-Cruz and R. Lozano. “Trajectory tracking for a four-rotor mini-aircraft”. Proceedings of the 44th IEEE Conference on Decision and control, and the European Control Conference 2005, pages 2505 – 2510, 2005.

[16] P. Pounds, R. Mahony, and P. Corke. “Modelling and control of a quadrotor robot”. In Australasian conference on robotics and automation 2006, Auckland, NZ, 2006.

[17] W. Zeng, B. Xian, C. Diao, Q. Yin, H. Li, and Y. Yang. “Nonlinear adaptive regulation control of a quadrotor unmanned aerial vehicle”. In Control Applications (CCA), 2011 IEEE International Conference on, pages 133 –138, 2011.

[18] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar. Design, “modeling, estimation and control for aerial grasping and manipulation”. In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, pages 2668 –2673, sept. 2011.

[19] T. Bresciani, “Modelling, Identification and Control of a Quadrotor Helicopter”, Master’s thesis, Lund University, Sweden, 2008.

[20] H.K. Khalil and JW Grizzle. “Nonlinear systems”, volume 3. Prentice Hall New Jersey, 2002.

[21] Parrot ardrone 2.0 web. http://ardrone2.parrot.com/

[22] P. Benavidez, J. Lambert, A. Jaimes, and M. Jamshidi, « Landing of an ARDrone 2.0 quadrotor on a mobile base using fuzzy logic », International Journal of Complex Systems-Computing, Sensing and Control, Vol 1, N° 1-2, pp. 5-25, 2013.

[23] T. Krajrik, V. Vonasek, D. Fiser, and J. Faigl,” AR.Drone as a robotic platform for research and education”, in International Conference on Research and Education in Robotics, Prague, Springer 2011.

[24] Willow Garage, Documentation Robot Operating System, available: http://www.ros.org/wiki/

[25] K. Alisher, K. Alexander, and B. Alexandr, “Control of the mobile robots with ROS in robotics courses” 25th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2014.

[26] A. H. N. Koenig, “Gazebo – ROS wiki”, available: http://www.ros.org/wiki/gazebo.

[27] M.A. Olivares-Mendez, S. Kannan, and H. Voos, “V-REP &Ros tested for design, test, and tuning of a quadrotor vision based fuzzy control system for autonomous landing”, 2014 International Micro Air vehicle Conference and Competition (IMAV 2014).

[28] J. Rada-Vilela. (2013). fuzzylite - A Fuzzy Logic Control Library and Application in C++Available: http://code.google.com/p/fuzzylite/

[29] “AR Drone Simulink Development-Kit V1.1 - File Exchange - MATLAB Central.” [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/43719-ar-drone-simulink-development-kit-v1-1. [Accessed: 16-Sep-2015].

[30] D. Escobar Sanabria and P. J. Mosterman. Ar drone simulink development-kit v1.1, website, http://www.mathworks.com/matlabcentral/ leexchange/43719-ar-drone-simulinkdevelopment- kit-v1-1, 2014.

[31] W. Yeugkwon, K. Seyoug, and K. Eunjin, “Real-Time visual tracking and remote control application for an aerial surveillance robot”, Journal of Automation and Control Engineering, Vol 3, N° 6, Decmeber 2015.

[32] L. Mejias, S. McNamara, J. Lai, and J. Ford, “Vision-Based detection and tracking of aerial targets for UAV collision avoidance”, in Intelligent Robot and Systems (IROS), 2010 IEEE/RSJ International Conference, pp. 87-92. 2010.

[33] P. Linci, A. Vinyojita Mohanraj; “Target Identification and Tracking Using UAV”; International Journal of Engineering research & Technology. Vol 4, issue 4, pp. 353-356, April 2015.

[34] K.E. Wenzel, A. Masselli, and A. Zell, “Automatic take off, tracking and landing of a miniature UAV on a moving carrier vehicle”, Journal of Intelligent & Robotic Systems, Vol. 61, pp. 221-238, 2011.

[35] A. Prayitno, V. Indrawati, and G. Utomo, “Trajectory tracking of AR.Drone quadrotor using fuzzy logic controller”, TELEKOMNIKA, Vol. 12, N° 4, pp. 819-828, 2014.

[36] P. Benavidez, J. Lambert, A. Jaimes, and M. Jamshidi, « Landing of an ARDrone 2.0 quadrotor on a mobile base using fuzzy logic », International Journal of Complex Systems-Computing, Sensing and Control, Vol 1, N° 1-2, pp. 5-25, 2013.

[37] F. Arifin, R. Arifandi Daniel, and D. Widiyanto,” Autonomous detection and tracking of an object autonomously using AR.Drone quadrotor”, 12 Journal of Computer Science and Information, Vol 7, pp. 11-17, 2014

[38] J. W. Tweedate, “Fuzzy control loop in an autonomous landing system for unmanned air vehicles”, WCCI 2012 IEEE World Congress on Computational Intelligence, June 10-15, 2012, Brisbane, Autralia.

[39] K.M. Zemalache, and H. Maaref, “Controlling a drone: comparison between a based model method and a fuzzy inference system”, Applied Soft Computing, Vol. 9, pp. 404-418, September 2009.

[40] L. Doitsidis, K. Valavanis, N. Tsourveloudis, M. Kontitis, « A framework for fuzzy logic based UAV navigation and control », In, 2004 IEEE International Conference on Robotics and Automation, 2004, Proceeding ICRA’04,Vol.4, pp. 4041-4046, 2004.

[41] K. Boudjit, C. Larbes, “Detection and Target Tracking with a Quadrotor Using Fuzzy Logic”, 8th International Conference on Modelling Identification and Control (ICMIC 2016), November 2016, Alger, Algeria.

[42] M.T. Hagan, H.B. Demuth, M. Beale, «Neural Network Design», PWS Publishing Compagny, 1995.

[43] Simon Haykin, «Neural Networks : A Comprehensive Foundation», IEEE Press, 1994.

[44] J.A. Freeman, D.M. Skapura, «Neural Networks: Algorithms, Applications, and Programming Techniques», Addison-Wesley, 1992.

[45] R.P. Lippmann, «An Introduction to Computing with Neural Nets», IEEE ASSP Magazine, pp. 4-22, avril 1987.

[46] R.V. Badu, S. Suresh; and All, “Online Adaptive radial Function Networks for Robust Object Tracking”; Computer Vision and Image Understanding, pp. 297-310, 2010.

[47] H. K. Kavith, S.C.P. Kumar, “Study on Object Tracking Using Neural Network Functions”, International Journal of Advanced Research in Computer and Communication Engineering,Vol 3, issue 5, pp. 6424-6427, May 2014.

[48] T. Dierks, S. Jagannathan, “Output Feedback Control of a Quadrotor UAV Using Neural Networks”, IEEE transactions on neural Networks, Vol 21, N°1, pp. 50-66, January 2010.

[49] R. L. Galvez, E. P. Dadios, A. A. Bandala, “Predicting the Motion of Quadrotor Using neural Network”, 8th IEEE International Conference Humoid, Nanotechnology Information Technology Communication and Control, Environment and Management (HNICEM) 9-12 December 2015, Philippines.

[50] L. Jangwon, J. Wang, D. Crandall, and all, “Real-Time Object Detection for Unmanned Aerial vehicles Based on Cloud-Based Convolutional Neural Networks”, Journal Concurrency and Computation : Practice and Experience, Vol 29, issue 6, 2017.

[51] M. Shirzadeh, A. Amirkhani, A. Jalali, M.R. Mosari, “An Indirect Adaptive Neural Control of a Visual-Based Quadrotor Robot for Pursing a Moving Target”, ISA Transactions, Elsevier, October 2015.

[52] A. D. Mengistu, D. M. Alemayehu, “Robot for Visual Object Tracking Based on Artificial Neural Network”, International Journal of Robotics Research and Development (IJRRD), vol 6, issue 1, February 2016.

[53] W. Yang, Z. Jin, C. Thiem, and all, “Autonomous Target Tracking of UAVs Based on Lower-Power neural Network Hardware” machine Intelligence and Bio-inspired Computation: Theory and Applications VIII, Proc of SPIE, Vol 9119, 2014.

[54] L. Wang, W. Ouyang, X. Wang, and all; “Visual tracking with Fully Convolutional Networks” 2015 IEEE International Conference on Computer Vision (ICCV), December 2015, USA.

[55] F. Garcia and E. Araujo, “Visual Multi-Target Tracking by Using Modified Kohonen Neural Networks,” in International Joint Conference on Neural Network. IEEE, 2008, pp. 4162–4167.

ANNEXES

ANNEXE 1

Paramètres physiques du quadrotor AR.Drone

Le tableau 1 présente les paramètres physiques de l'AR.Drone 2.0 nécessaires à la simulation et aux tests expérimentaux utilisés dans le présent document.

Tableau 1. Paramètres physiques du quadrotor AR.Drone 2.0.

Symbole	Définition	Valeur	Unité
m	Quadrotor Poids	335	Gr
L	Longueur de bras	18	Cm
I_z	Moment d'inertie autour de l'axe z	4.7 x 10^{-3}	Kg.m^2
I_x	Moment d'inertie autour de l'axe x	1.8 x 10^{-3}	Kg.m^2
I_y	Moment d'inertie autour de l'axe y	1.8 x 10^{-3}	Kg.m^2
b	Facteur de poussée de l'hélice	5.7231 x 10^{-6}	N.S^2
d	Facteur de traînée de l'hélice	1.7169 x 10^{-7}	N.m.S^2
J_r	Inertie du Rotor	1.85 x 10^{-5}	Kg.S^2

ANNEXE 2

Cadre de transformation ARDrone-Marqueur

En fait, l'axe de la caméra ne correspond pas à l'axe du drone. Pour mieux montrer la nécessité d'une conversion, la figure 1 compare les deux systèmes de références, où X_D, Y_D et Z_D désignent l'axe du drone et X_C, Y_C et Z_C indiquent l'axe de la caméra.

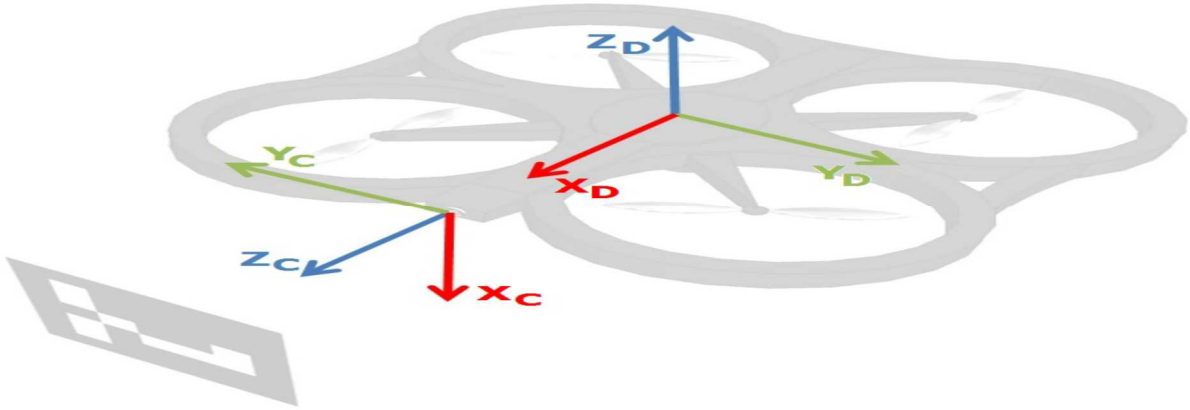


Figure 1. Représentation de l'axe de la caméra du drone.

Il est clair qu'une rétro-translation est nécessaire. En particulier deux rotations sont nécessaires pour convertir l'axe de la caméra en axe de drone, comme illustré à la figure 2. Dans la figure 3 les rotations sont décrites plus en détail.

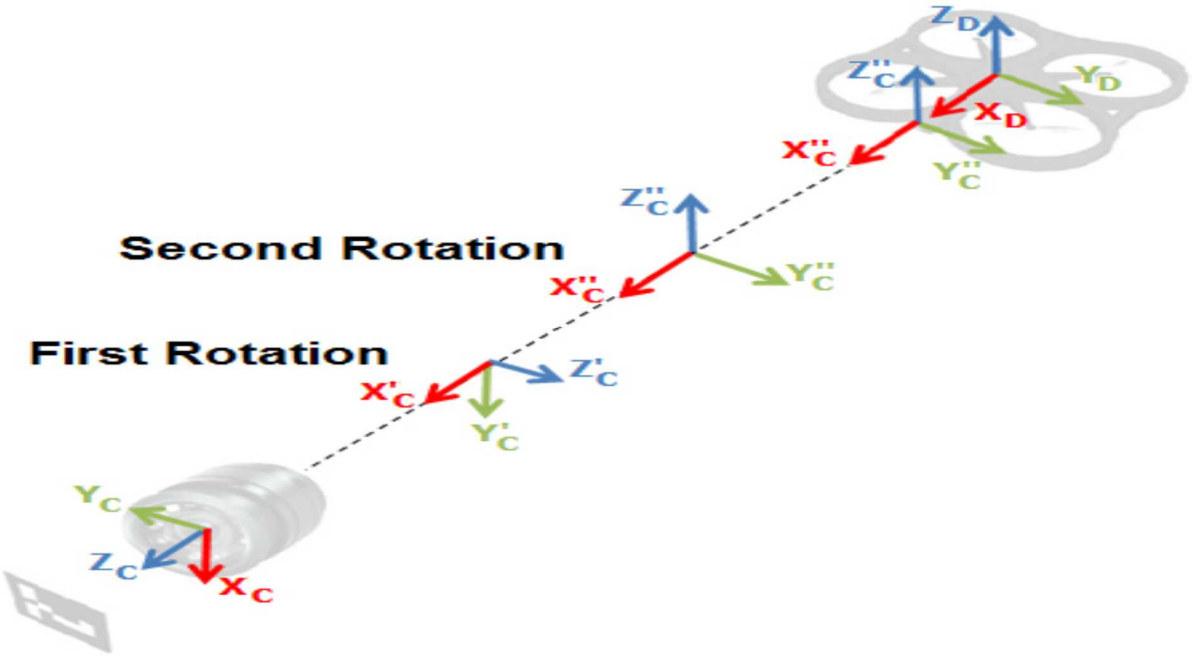


Figure 2. Retro-translation de l'axe de la caméra.

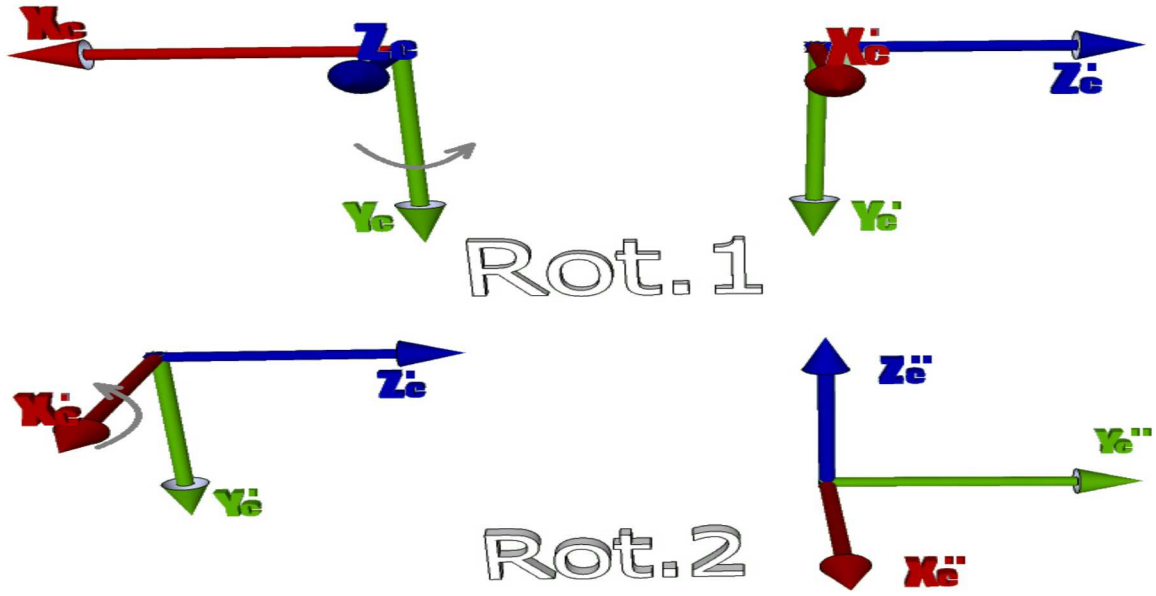


Figure 3. Détail des rotations de l'axe de la caméra.

La première rotation définit les axes X'_C, Y'_C *et* Z'_C à travers la matrice de rotation suivante :

$$R_{(Z_D,\ 90°)} = \begin{bmatrix} \cos(90°) & -\sin(90°) & 0 & 0 \\ \sin(90°) & -\cos(90°) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

La deuxième rotation définit les axesX''_C, Y''_C et Z''_C à travers la matrice de rotation suivante :

$$R_{(X_D,\ 90°)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(90°) & -\sin(90°) & 0 \\ 0 & \sin(90°) & \cos(90°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

Pour compléter le chevauchement des axes il faut une translation de 20 cm du système de référence de la caméra au centre du Drone. Ainsi, la matrice complète de retro-translation décrit la relation entre le système de référence de la caméra et celui du drone et se traduit comme suit :

$$H = R_{(Z_D}, 90°)\, R_{(X_D}, 90°)\, T_{(X_D,\ 20cm)} \quad (3)$$

Par exemple, pour représenter la position d'un point générique QC dans le système de référence de la caméra dans le système de référence du drone, il est nécessaire de multiplier son vecteur de position dans le système de référence de la caméra par la matrice de retro-translation H, pour obtenir le QC, qui définit la position du point dans le système de référence du drone:

$$Q_D = H \,.\, Q_C \quad (4)$$

Néanmoins, cette simple conversion décrit la direction des mouvements qui doivent être exécuté par le drone, mais ne définit pas la vraie torsion à envoyer au drone. Les modules de vitesse dépendent de la loi de contrôle.

ANNEXE 3

Données de la trajectoire spirale

X	0	4	3.7	2.8	1.5	0	-1.5	-2.8	-3.7	-4	-3.7	-2.8	-1.5
Y	0	0	1.5	2.8	3.7	4	3.7	2.8	1.5	0	-1.5	-2.8	-3.7
Z	1	1.125	1.25	1.375	1.5	1.625	1.75	1.875	2	2.125	2.25	2.375	2.5
Lacet	0	0	0	0	0	0	0	0	0	0	0	0	0
Temps	3	3	3	3	3	3	3	3	3	3	3	3	3

X	0	1.5	2.8	3.7	4	3.7	2.8	1.5	0	-1.5	-2.8	-3.7	-4
Y	-4	-3.7	-2.8	-1.5	0	1.5	2.8	3.7	4	3.7	2.8	1.5	0
Z	2.625	2.75	2.875	3	3.125	3.25	3.375	3.5	3.625	3.75	3.875	4	4.125
Lacet	0	0	0	0	0	0	0	0	0	0	0	0	0
Temps	3	3	3	3	3	3	3	3	3	3	3	3	3

X	-3.7	-2.8	-1.5	0	1.5	2.8	3.7	4	3.7	2.8	1.5	0	-1.5
Y	-1.5	-2.8	-3.7	-4	-3.7	-2.8	-1.5	0	1.5	2.8	3.7	4	3.7
Z	4.25	4.375	4.5	4.625	4.75	4.875	5	5.125	5.25	5.375	5.5	5.625	5.75
Lacet	0	0	0	0	0	0	0	0	0	0	0	0	0
Temps	3	3	3	3	3	3	3	3	3	3	3	3	3

X	-2.8	-3.7	-4	-3.7	-2.8	-1.5	0	1.5	2.8	3.7	4	13	12.7
Y	2.8	1.5	0	-1.5	-2.8	-3.7	-4	-3.7	-2.8	-1.5	0	0	1.5
Z	5.875	6	6.125	6.25	6.375	6.5	6.625	6.75	6.875	7	7.125	7.125	7
Lacet	0	0	0	0	0	0	0	0	0	0	0	0	0
Temps	3	3	3	3	3	3	3	3	3	3	3	3	3

X	11.8	10.5	9	7.5	6.2	5.3	5	5.3	6.2	7.5	9	10.5	11.8
Y	2.8	3.7	4	3.7	2.8	1.5	0	-1.5	-2.8	-3.7	-4	-3.7	-2.8
Z	6.875	6.75	6.625	6.5	6.375	6.25	6.125	6	5.875	5.75	5.625	5.5	5.375
Lacet	0	0	0	0	0	0	0	0	0	0	0	0	0
Temps	3	3	3	3	3	3	3	3	3	3	3	3	3

X	12.7	13	12.7	11.8	10.5	9	7.5	6.2	5.3	5	5.3	6.2	7.5
Y	-1.5	0	1.5	2.8	3.7	4	3.7	2.8	1.5	0	-1.5	-2.8	-3.7
Z	5.25	5.125	5	4.875	4.75	4.625	4.5	4.375	4.25	4.125	4	3.875	3.75
Lacet	0	0	0	0	0	0	0	0	0	0	0	0	0
Temps	3	3	3	3	3	3	3	3	3	3	3	3	3

X	9	10.5	11.8	12.7	13	12.7	11.8	10.5	9	7.5	6.2	5.3	5
Y	-4	-3.7	-2.8	-1.5	0	1.5	2.8	3.7	4	3.7	2.8	1.5	0
Z	3.625	3.5	3.375	3.25	3.125	3	2.875	2.75	2.625	2.5	2.375	2.25	2.125
Lacet	0	0	0	0	0	0	0	0	0	0	0	0	0
Temps	3	3	3	3	3	3	3	3	3	3	3	3	3

X	5.3	6.2	7.5	9	10.5	11.8	12.7	13	13	13
Y	-1.5	-2.8	-3.7	-4	-3.7	-2.8	-1.5	0	0	0
Z	2	1.875	1.75	1.625	1.5	1.375	1.25	1.125	1	0
Lacet	0	0	0	0	0	0	0	0	0	0
Temps	3	3	3	3	3	3	3	3	3	0