RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département d'Automatique

Laboratoire de Commande des Processus

Mémoire de projet de fin d'études

pour l'obtention du diplôme d'ingénieur d'état en Automatique

Commande et Navigation d'un Quadrirotor : Etude et Réalisation

Anes BENMERZOUG
Oussama MEKKID

Sous la direction de
Mr. M. TADJINE Professeur
Mr. M. CHAKIR Enseignant Chercheur

Présenté et soutenu publiquement le 21/06/2016

Composition du Jury :

Président	Mr. M.S. BOUCHERIT	Professeur	Ecole Nationale Polytechnique
Promoteurs	Mr. M. TADJINE	Professeur	Ecole Nationale Polytechnique
	Mr. M. CHAKIR	Enseignant Chercheur	Ecole Nationale Polytechnique
Examinateur	Mr. B. HEMICI	Professeur	Ecole Nationale Polytechnique

ENP 2016

ملخص

يعرض هذا العمل, التحكم و الملاحة للطائرة بدون طيار ذات رباعية المحرك, في بداية الامر قمنا بوضع نموذج للنظام و بعده تم استنباط قانون التحكم الإنزلاقي (MG)خاص, و بعد هذا قمنا بتقديم عرض و نبذة على اشكالية الأحادي التحديث و الاستحداث الآني SLAM و إقتراح خوارزمية ملائمة, و من تم استخدمنا المفعلين ROS و GAZEBO من أجل محاكاة الطريقة المقترحة, و في الأخير اشتمل عملنا على تحقيق نموذج وتطبيق قانون تحكم للطائرة و هذا بعد شرح مختلف المركبات اللازمة وكشف مختلف المعدات.

الكلمات المفتاحية : طائرة بدون طيار, التحكم الإنزلاقي, الأحادي التحديث و الاستحداث الآني SLAM , ROS و GAZEBO

Abstract

In this thesis, we focus on the control and navigation of a quadrotor UAV. First of all, we will model the system and synthesize a sliding mode control. Then, we will present the problem of monocular SLAM, its state of the art and we will propose an approach to solve it. After that, we will introduce ROS and Gazebo we will use them to simulate realistically the implementation of the sliding mode control and the proposed SLAM approach. Finally, we will make our own quadrotor. We will begin by presenting its different components, then we will identify the necessary parameters for the implementation of the control methods and we will implement them on our system.

Keywords: UAV, quadrotor, control, sliding mode, monocular SLAM, ROS, Gazebo, making, identification.

Résumé

Dans ce mémoire, nous nous intéressons à la commande et la navigation d'un drone du type quadrirotor. Dans un premier lieu, nous modélisons le système et nous synthétisons une commande par mode glissant. Ensuite, nous présenterons le problème du SLAM monoculaire et son état de l'art et nous proposerons une approche pour le résoudre. Puis, nous introduirons ROS et Gazebo que nous utiliserons pour simuler de façon réaliste l'implémentation de la commande et de l'approche de SLAM proposée. Dans un dernier lieu, nous réaliserons un quadrirotor. Nous commencerons par une présentation du matériel utilisé, nous ferons ensuite l'identification des paramètres nécessaires pour l'implémentation de commandes et nous implémenterons ces dernières sur notre quadrirotor.

Mots clés: drone, quadrirotor, commande, mode glissant, SLAM monoculaire, ROS, Gazebo, réalisation, identification.

Dédicace

Je dédie ce travail à mes très chers parents, dont le sacrifice, la tendresse, l'amour, la patience, le soutien, l'aide et l'encouragement sont l'essentiel de ma réussite. Sans eux je ne serai pas à ce stade aujourd'hui.

A mon frère et ma sœur pour leur soutien continue durant mon parcours.

A ma grande famille.

Et à tous mes amis.

Anes

Dédicace

A la mémoire de mes frères décédé à L'EPST Tlemcen, aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours eu pour vous. Allah yerhamkoum.

Je dédie ce travail à : Mes très chers parents, dont le sacrifice, la tendresse, l'amour, la patience, le soutien, l'aide et l'encouragement sont l'essentiel de ma réussite. Sans eux je ne serai pas à ce stade aujourd'hui.

Mes frères et mes sœurs pour leur soutien continue durant mon parcours.

Ma grande famille.

Mes meilleurs amis Redouane et Bilal sans oublier mes amis du marché de Medea.

Oussama

Remerciements

Nous remercions Dieu le tout puissant de nous avoir donné la force et le courage pour réaliser ce travail.

Les travaux présentés dans ce mémoire ont été effectués au sein du Laboratoire de Commande des Processus (LCP) de l'Ecole Nationale Polytechnique.

Ce travail que nous présentons a été effectué sous la direction de Monsieur M. Tadjine, professeur à l'Ecole Nationale Polytechnique, et Monsieur M. Chakir, enseignant chercheur à l'Ecole Nationale Polytechnique, qui ont suivie de très près ce travail, pour leur orientation pédagogique dans l'élaboration de ce mémoire.

Nous tenons à remercier monsieur M.S. Boucherit, professeur à l'Ecole Nationale Polytechnique, pour l'honneur qu'il nous fait de présider le jury de notre soutenance.

Que Monsieur B. Hemici, professeur à l'Ecole Nationale Polytechnique, soit convaincu de notre sincère reconnaissance pour avoir accepté d'examiner et de critiquer ce mémoire.

Nous remercions tous nos amis qui nous ont aidé dans l'élaboration de ce mémoire, en particulier Lakab Ahmed, Belkheroubi Abdesalam, Benallel Mohammed Amine, Debib Abdelhakim et Rai Amar.

Enfin, nous tenons à remercier tous les gens qui ont contribué à notre réussite tout au long de notre parcours d'étude.

Table des matières

Liste des tableaux

Liste des figures

Introduction Générale

La robotique est un domaine dont le nombre d'applications est en augmentation constante depuis son apparition. Il s'agit d'un domaine multidisciplinaire qui regroupe des aspects concernant la mécanique, l'informatique ou encore l'électronique. Cet essor découle essentiellement du besoin d'automatiser des tâches répétitives et difficiles dans tous les domaines de la vie quotidienne pour remplacer l'être humain.

Globalement, la robotique est divisée en deux grand catégorie. On trouve d'une part les robots mobiles qui représentent simplement l'ensemble des robots à base mobile (robots mobiles à roues, drones...). D'autre part, on trouve les robots fixes ou les bras manipulateurs, qui sont plus utilisés dans l'industrie (chirurgie médicale, chaine de montage...)

Dans le domaine des robots mobiles, la conception d'un drone autonome capable de naviguer dans un environnement inconnue pose de nombreux défis tant bien théoriques, que pratiques. En effet, les mesures provenant des capteurs utilisés sont entachées de bruit, la grande puissance de calcul nécessaire pour exécuter les différents algorithmes de commande et d'observation, la forte non linéarités du système et la synthèse des lois de commande en font à problème difficile à résoudre[1].

Le quadrirotor est un drone à voilure tournante de type VTOL (Vertical Take Off and Landing). Il a attiré l'attention et est devenu le sujet de recherche de plusieurs équipes et laboratoires ces dernières années.

La localisation et la cartographie (SLAM) est une étape primordiale de la navigation, elle permet de découvrir l'environnement qui entoure le robot mobile et de le positionner par rapport à son entourage[2]. Plusieurs approches ont été proposées pour résoudre le problème du SLAM.

Le premier chapitre de notre travail présente brièvement les généralités sur les quadrirotors et leurs principes de fonctionnement. Puis nous élaborons un modèle d'état qui nous permettra d'étudier l'évolution dynamique des angles, positions, vitesses de rotations et de translation du quadrirotor. Ensuite, nous appliquons une commande par mode glissant pour asservir la position et l'orientation du quadrirotor. Enfin, nous présentons les différents résultats des simulations effectuées sous MATLAB pour différents scénarios de vol.

Le second chapitre sera consacré à la navigation qui est basée sur la localisation et cartographie simultané (SLAM). Nous présentons dans un premier lieu l'historique du SLAM en général et les différents capteurs utilisés pour ses différentes approches. Puis, nous nous intéressons au SLAM monoculaire, pour lequel nous présentons l'état de l'art, ainsi que quelques uns des algorithmes d'estimation utilisés. Enfin, nous expliquons l'approche proposée pour résoudre le problème du SLAM monoculaire.

Dans le troisième chapitre, nous présentons d'une manière succincte le système d'exploitation (ROS), ces avantages, ces notions de base et son principe de fonctionnement. Ensuite, nous effectuons une implémentation de deux approches de SLAM sous ROS. La première étant l'approche proposée au chapitre précédant et la deuxième une approche basé sur l'EKF (Extended Kalman Filter). Enfin nous terminons par une étude comparative entre ces deux approches.

Le quatrième chapitre, est consacré à la réalisation de notre quadrirotor. Tout d'abord, nous commençons par une description du matériel, des logiciels, des bibliothèques et du langage de programmation utilisés. Puis, nous expliquons la méthode d'acquisition des données et les filtres utilisés pour obtenir de bonnes mesures. Ensuite, nous effectuons une identification du système étudié pour pouvoir déterminer les différents paramètres dont dépend la dynamique de ce dernier. Enfin nous implémentons une commande par PID pour l'asservissement de l'angle de lacet ψ.

Chapitre 1

Modélisation et Commande

1.1 Introduction

Le quadrirotor est une excellente plateforme pour la recherche sur le contrôle et l'observation, de par sa nature non linéaire et sous-actionnée il est idéal pour la synthèse et l'analyse des algorithmes de commande et d'observation. Il est parmi les types de drones qui ont connu un grand essor dans le monde de la recherche car il offre beaucoup d'avantages, d'intérêt et d'applications.

Dans ce chapitre, nous présentons de façon générale les quadrirotors et leur avantages. Puis, nous présentons une petite étude sur l'état de l'art et l'avancement de la recherche sur ces engins volants et nous expliquerons quelques unes de leurs applications. Ensuite, nous détaillerons la cinématique et la dynamique du quadrirotor et nous ferons la synthèse d'une technique de commande par modes glissant pour la poursuite de trajectoires. Enfin, nous simulons le système sous MATLAB et nous vérifions le bon fonctionnement et la robustesse de la commande, ainsi que la justesse de l'une des hypothèses utilisées.

1.2 Définitions

Le Drone

Un drone ou UAV (de l'anglais Unmanned Aerial Vehicle) est un appareil volant sans pilote, i.e. capable de voler et d'effectuer une mission sans présence humaine à bord de l'appareil et qui est soit contrôlé à distance depuis un autre lieu (le sol, un autre appareil volant, l'espace), soit programmé et totalement autonome[3].

Il en existe plusieurs types, comme en peut le voir sur la figure 1.1, qui peuvent être classifiés selon plusieurs critères : la taille, l'altitude, le système de contrôle, etc[4].

Le Quadrirotor

Le quadrirotor est un aéronef faisant partie de la famille des hélicoptères, plus particulièrement de la famille des multirotors. Le quadrirotor possède plusieurs caractéristiques (simplicité mécanique, décollage/atterrissage vertical, vol stationnaire, agilité) qui lui procurent plusieurs avantages opérationnels par rapport à d'autres types d'appareils. Comme son nom l'indique, il se compose de quatre moteurs, figure 1.2, qui sont situés aux extrémités de quatre bras. Son mouvement est contrôlé par la variation de la vitesse angulaire de chaque rotor pour changer la force de portance et le couple crée par chacun d'eux. Deux rotors du même axe tournent dans le sens horaire ; alors que les deux autres tournent dans le sens anti-horaire pour compenser le couple crée par ces derniers sur l'armature et

(a)

(b)

(c)

FIGURE 1.1: (a) Hummingbird, drone militaire de l'armée américaine, (b) Predator, drone militaire de l'armée américaine, (c) AR.Drone, drone civil de la société française Parrot

éviter ainsi, en considérant que les quatre rotors tournent à la même vitesse et que tous les éléments sont identiques (hélices, dimensions, équilibre des masses), qu'elle ne tourne sur lui même.

FIGURE 1.2: Modèles de quadrirotors

1.3 Avantages du quadrirotor

La conception du quadrirotor offre de réels avantages par rapport à d'autres configurations[5] :

– Leurs tailles réduites et leur manœuvrabilité leurs permettent de voler dans des environnements fermés (Indoor) ou ouverts (Outdoor) et près des obstacles à l'opposition des hélicoptères classiques

- la simplicité de sa mécanique facilite sa maintenance
- aucun embrayage n'est exigé entre le moteur et le rotor et aucune exigence n'est donnée sur l'angle d'attaque des rotors
- quatre petits rotors remplacent le grand rotor de l'hélicoptère ce qui réduit énormément l'énergie cinétique stockée et minimise les dégâts en cas d'accidents

1.4 L'état de l'art sur les quadrirotors

Au fil du temps et grâce à l'avancement des techniques de production et l'innovation connus dans la technologie des capteurs et des calculateurs numériques, qui tend particulièrement vers la miniaturisation et la haute précision, les quadrirotors ont vu le jour. Ces drones sont équipés d'une électronique de commande et des capteurs pour assurer leur autonomie et leur stabilité. Par conséquent, le problème de la commande de cette nouvelle génération d'UAV a connu un énorme progrès et de nouveaux axes de recherches ont été crées pour résoudre cette problématique[6]. Le nombre de projets portants sur le problème de la conception et la commande des quadrirotors ne cesse d'augmenter. Ils existent des projets qui portent sur le problèmes de la modélisation et la commande en se basant sur des plateformes commerciales comme le Draganflyer, HMX4 , UFO4, etc [7]. L'objectif est de doter ces quadrirotors avec plus de capteurs et d'intelligence pour réaliser un certain degré d'autonomie. Tandis que d'autres projets ont abordé le problème de la conception[8].

Coopération entre quadrirotors

Avec les progrès technologiques dans le domaine des drones, le quadrirotor s'est vu accordé un intérêt particulier pour sa maniabilité et sa capacité de charge utile. Ces atouts sont amplifiés lorsque plusieurs d'entre eux sont déployés simultanément. Dés que le nombre d'agents opérant dans le même environnement augmente, une intelligence commune est nécessaire pour optimiser leur coopération et assurer leur sécurité tout au long de l'accomplissement de leurs missions.

L'une des applications de la coopération entre les quadrirotors est le transport aérien de charges utiles par des câbles de remorquage. Il est en général utilisé dans les interventions d'urgence, dans l'industrie et dans des applications militaires pour le transport d'objets dans des environnements inaccessibles par d'autres moyens. On trouve comme exemples d'usage de remorquage aérien des missions de sauvetage où des individus sont transporté hors de lieux dangereux ou bien la livraison d'équipement lourd au sommet d'un immeuble de grande hauteur. En règle générale, le remorquage aérien est réalisée par l'intermédiaire d'un seul câble relié à une charge utile. Cependant, la contrôlabilité de la charge utile se trouve limitée de part le fait qu'il n'y ait qu'un seul point d'attache[9].

En [10], les auteurs présentent une nouvelle approche pour la manipulation et le transport aérien de charges en utilisant plusieurs quadrirotors. Ils présentent aussi les résultats des essais expérimentaux, figure 1.3, qui montrent que la coopération entre quadrirotors est une méthode efficace pour la manipulation et le transport de charges utiles qui sont au delà des capacités d'un seul quadrirotor.

Quadrirotors à pales mobiles

Les quadrirotors sont largement utilisés comme plates-formes expérimentales en raison de leur simplicité mécanique et leur robustesse inhérente. Pour les quadrirotors traditionnels à pales fixes, le contrôle de la stabilité et du vol sont effectués en contrôlant la vitesse

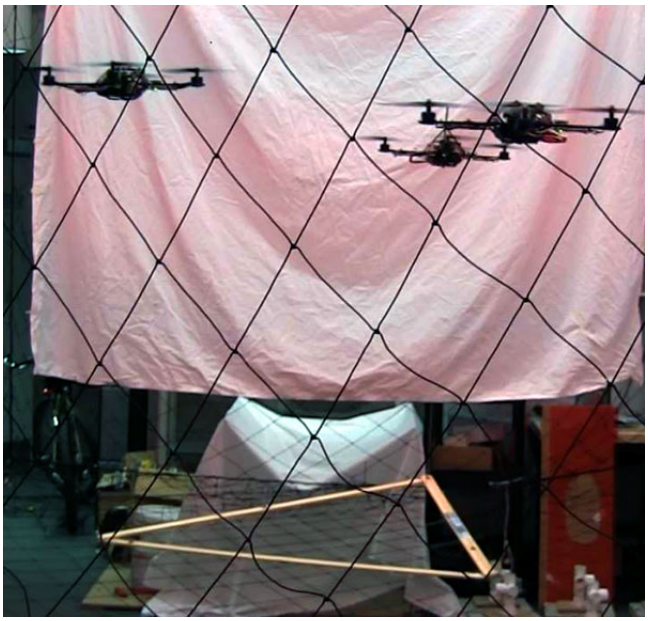


FIGURE 1.3: Coopération de trois quadrirotors pour le soulèvement d'une charge

angulaire de chacun des quatre rotors. Cependant, cela implique que la bande passante de la commande se trouve limitée par l'inertie des rotors. la bande passante du contrôleur devient un problème pour la stabilité du quadrirotor dés que sa taille augmente[11]. Les grands quadrirotors requièrent d'utiliser de grands moteurs qui ont une plus grande inertie et qui ne peuvent être commandé aussi rapidement que les petits moteurs. Un autre inconvénient des quadrirotors à pales fixes c'est qu'ils sont intrinsèquement incapables d'achever l'inversion de la poussée[12].

En [8], les auteurs ont réalisé un quadrirotor à pales mobiles, figure 1.4, en utilisant quatre servomoteurs à grande vitesse pour contrôler l'angle d'attaque des hélices de chaque rotor à l'aide d'une tige qui passe par l'axe de ce dernier qui est creux.

FIGURE 1.4: Quadrirotor à pales mobiles conçu au sein de l'Aerospace Controls Lab de MIT

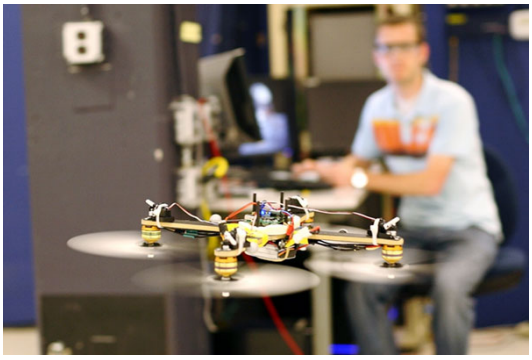


FIGURE 1.5: Quadrirotor à pales mobiles qui vole en étant renversé

Les premiers essais ont montré que le quadrirotor à pales mobiles à des performances comparables à celle du quadrirotor traditionnel à pales fixes pour la poursuite nominale de trajectoire. Mais, il présente de meilleures performances lorsqu'il s'agit d'effectuer des manœuvres plus agressives et lorsqu'il s'agit d'inversement de la poussée et de vol inversé, comme on peut le voir sur la figure 1.5.

1.5 Applications et utilisations

En raison des capacités uniques du quadrirotor telles que sa grande manœuvrabilité, sa petite taille et sa relative facilité de contrôle, il trouve de nombreux usages et applications[13].

L'agriculture

Le quadrirotor est une plateforme intéressante pour l'agriculture et la sylviculture, car il peut fournir des informations géographiques précises à propos de l'état du terrain. En [14], les auteurs proposent d'utiliser un quadrirotor pour faire l'arpentage correct et précis ainsi que la surveillance de l'état des cultures et pour réduire l'effort humain.

FIGURE 1.6: Quadrirotor Indago utilisé pour la surveillance de l'état des cultures

L'inspection des lignes électriques

L'inspection des lignes électriques à haute tension est principalement effectué par des véhicules aériens pilotés ou des patrouilles à pied. Cependant, ces méthodes d'inspection sont en quelque sorte inefficace et coûteuse. De plus, les inspections assistées par hélicoptère mettent en danger la vie humaine. Récemment, un grand nombre de recherches se sont concentrées sur l'inspection automatique des lignes électriques. En particulier, les drones sont une technologie intéressante pour l'inspection automatique, car ils permettent une inspection plus rapide que les patrouilles à pied et ont la même ou une meilleure précision que les coûteuse inspections par hélicoptère[13]. En [15], les auteurs proposent d'utiliser un quadrirotor équipé d'une caméra caméra couleur et d'une caméra TIR(Thermal Imaging Camera) pour inspecter les appareils et les composants dans les couloirs de lignes électriques.

1.6 Modélisation du Quadrirotor

Un système est une ensemble d'objets ou de phénomènes liés entre eux et isolés artificiellement du monde extérieur. La modélisation regroupe un ensemble de techniques permettant de disposer d'une représentation mathématique du système à étudier.

FIGURE 1.7: Quadrirotor équipé d'une caméra infrarouge pour la surveillance des lignes à très haute tension (UHT)

La synthèse des lois de commande d'un système dynamique nécessite une modélisation précise de ce dernier afin que le modèle puisse prévoir au mieux le comportement du système aux diverses excitations (commandes, perturbations, . . .). Ainsi, plus il est détaillé, plus il est fidèle au système. Néanmoins, cela engendre une complication de l'étude et de la synthèse d'éventuelles lois de commande. Un compromis doit être fait en adoptant des hypothèses simplificatrices afin de pouvoir répondre aux contraintes pratiques.

Le quadrirotor est classé dans la catégorie des systèmes volants les plus complexes vu le nombre des phénomènes physiques qui affectent sa dynamique. Afin de concevoir un contrôleur de vol, on doit d'abord comprendre profondément les mouvements du système et sa dynamique. Cette compréhension est nécessaire pas simplement pour la conception du contrôleur, mais afin de s'assurer que les simulations de l'engin dépeindront un comportement aussi proche que possible de la réalité quand la commande est appliquée.

1.6.1 Modes de vol

Le quadrirotor est un engin volant doté de quatre rotors placés aux extrémités de ses quatre bras. Ce sont ces quatre rotors qui fournissent la force verticale (portance) qui permet à l'appareil de s'élever. Les mouvements possibles du quadrirotor sont :

Les Gaz (mouvement vertical)

Cela représente le mouvement de montée/descente du drone. Les quatre moteurs tournent à la même vitesse, figure 1.8. On augmente la vitesse de ces derniers pour faire monter le quadrirotor tandis qu'on la diminue pour le faire descendre.

Le lacet (rotation autour de l'axe Z)

C'est une rotation autour de l'axe Z du repère liée au quadrirotor, figure 1.9. Pour faire tourner le quadrirotor dans le sens horaire (anti-horaire) on diminue la vitesse des moteurs avant et arrière (gauche et droit) ayant le sens de rotation horaire (anti-horaire), et on augmente celle des moteurs gauche et droit (avant et arrière) dont le sens de

rotation est dans le sens anti-horaire (horaire) tout en gardant la somme des vitesses inchangée, ce qui fait que le moment du lacet crée par ces derniers soit supérieur à celui créé par les deux autres sans changer de portance, roulis ou tangage.

Le roulis (rotation autour de l'axe X)

Ce mouvement se produit suite à une rotation autour de l'axe X du repère liée au quadrirotor en agissant sur les moteurs gauche et droit, figure 1.10. Par exemple, pour que le quadrirotor penche à droite, on augmente la vitesse du moteur gauche tout en diminuant celle du moteur droit.

Le tangage (rotation autour de l'axe Y)

Il permet de faire tourner le quadrirotor sur lui-même suivant l'axe Y du repère liée à ce dernier. Pour avoir un angle de tangage vers l'avant on augmente la vitesse du moteur arrière et on diminue celle du moteur avant, ce qui permet de créer un moment autour de l'axe Y tout en gardant la même force de portance et des moments de lacet et de roulis nuls, figure 1.11. On pourrait bien croire qu'on peut réaliser n'importe quelle inclinaison, mais en réalité si on dépasse une certaine limite le quadrirotor perd son équilibre et chute.

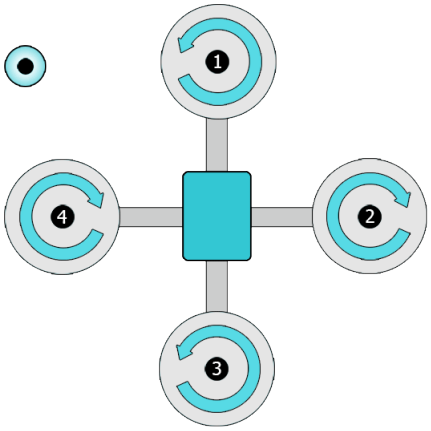


FIGURE 1.8: Les gaz

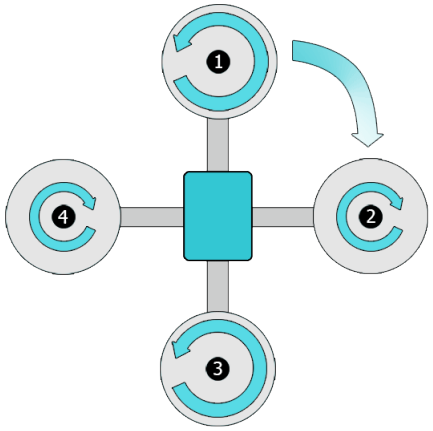


FIGURE 1.9: Le lacet

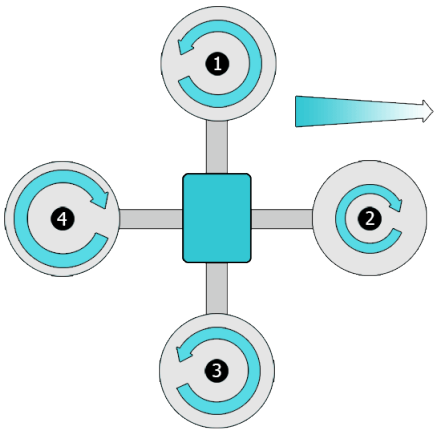


FIGURE 1.10: Le roulis

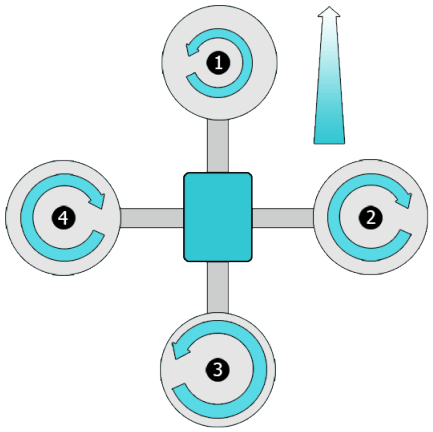


FIGURE 1.11: Le tangage

Le mouvement horizontal

Le mouvement horizontal ne peut pas être obtenu directement dans le cas du quadrirotor, c'est-à-dire qu'on ne peut pas le faire bouger en avant ou latéralement sans action sur les angles de tangage ou le roulis, car c'est un système sous-actionné. Cependant le déplacement horizontal est une conséquence de ces deux mêmes angles. En effet, si on maintient un angle de tangage donné, la force de portance normale au plan des quatre rotors aura une composante non nulle suivant l'axe Y et c'est cette composante qui permet un mouvement latéral dans sa direction. De même pour le mouvement suivant l'axe X, il suffit de maintenir un angle de roulis donné. En combinant ces mouvements on peut déplacer le quadrirotor dans toutes les directions voulues.

1.6.2 Repérage du Quadrirotor dans l'espace

Un quadrirotor nécessite deux trièdres pour être repéré dans l'espace, figure 1.12. Ces repères sont :

Le repère terrestre (inertiel ou fixe)

Il est noté $R_0(O_0, X_0, Y_0, Z_0)$. C'est un repère d'origine O_0 et d'axes X_0, Y_0 et Z_0 lié à la terre, supposée immobile.

Le repère lié au corps du quadrirotor (mobile)

Il est noté $R_1(O_1, X_1, Y_1, Z_1)$. C'est un repère d'origine O_1, qui coïncide avec le centre de gravité du quadrirotor, et d'axes X_1, Y_1 et Z_1. Ce repère peut être obtenu on effectuant trois rotations successives définies suivant la convention Z-Y-X de Tait-Bryan, qui est la plus utilisé en aéronautique, à partir du repère $R_0(O_0, X_0, Y_0, Z_0)$. Puis une translation de vecteur (x, y, z) à partir du point O_0 suivant les axes (X_0, Y_0, Z_0).

On peut en déduire donc les paramètres qui nous permettent de décrire le mouvement du quadrirotor : $(x, y, z, \phi, \theta, \psi, v_x, v_y, v_z, p, q, r)$

Où :

x (m) : la coordonnée du point O_1 (centre de gravité du quadrirotor) suivant l'axe X_0.
y (m) : la coordonnée du point O_1 (centre de gravité du quadrirotor) suivant l'axe Y_0.
z (m) : la coordonnée du point O_1 (centre de gravité du quadrirotor) suivant l'axe Z_0.
ϕ (rad) : l'angle de roulis. Rotation autour de l'axe X_1 ($-\frac{\pi}{2} < \phi < \frac{\pi}{2}$).
θ (rad) : l'angle de tangage. Rotation autour de l'axe Y' ($-\frac{\pi}{2} < \theta < \frac{\pi}{2}$).
ψ (rad) : l'angle de lacet. Rotation autour de l'axe Z_0 ($-\pi < \psi < \pi$).
v_x (m/s) : la vitesse du point O_1 suivant l'axe X_0.
v_y (m/s) : la vitesse du point O_1 suivant l'axe Y_0.
v_z (m/s) : la vitesse du point O_1 suivant l'axe Z_0.
p (rad/s) : la vitesse instantanée de rotation autour l'axe X_1.
q (rad/s) : la vitesse instantanée de rotation autour l'axe Y_1.
r (rad/s) : la vitesse instantanée de rotation autour l'axe Z_1.

Matrice de rotation

On considère que les centres O_0 et O_1 des deux repères sont confondus, ce qui signifie que le repère R_1 ne fait que des rotations par rapport au repère R_1. Trois paramètres indépendants, ϕ, θ, ψ, sont nécessaires pour décrire complètement l'attitude du repère R_1 par rapport à celle de R_0. Le passage du premier repère vers le second se fera par trois rotations successives, figure 1.13.

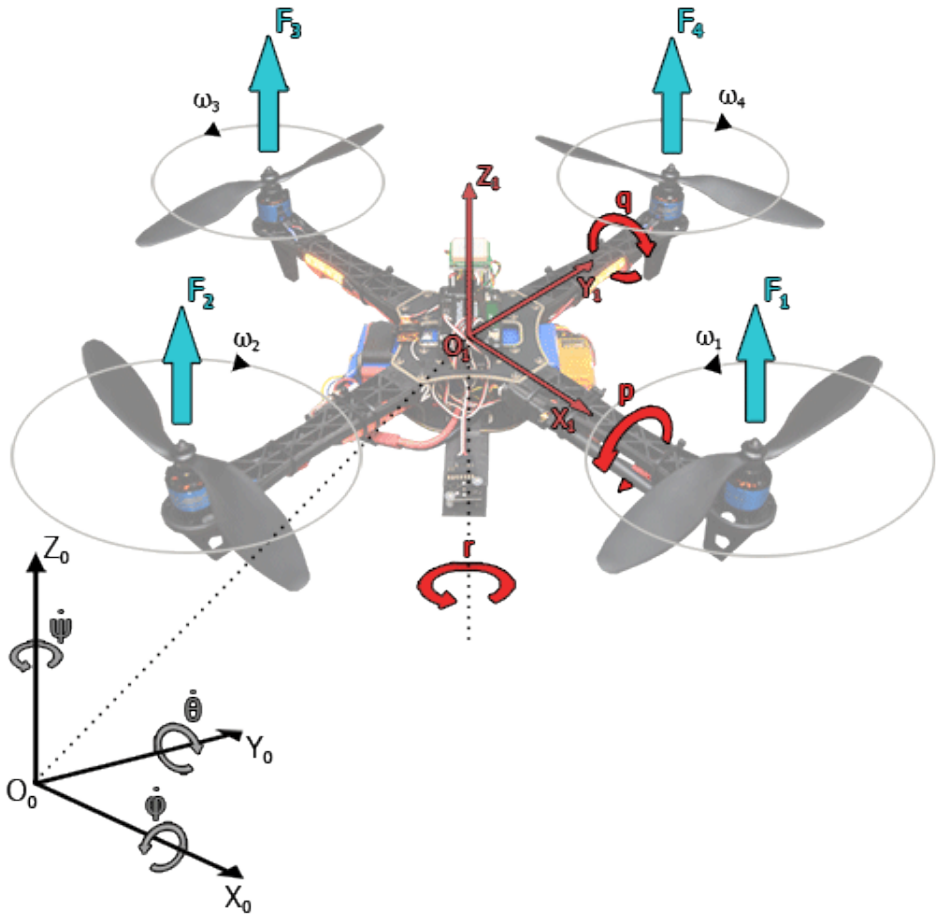


FIGURE 1.12: Repérage du quadrirotor

Première rotation

La première rotation est une rotation de ψ autour de l'axe Z_0 qui fait coïncider l'axe Y_0 avec l'axe Y' et l'axe X_0 avec l'axe X'. Sa matrice de rotation est :

$$R_\psi = \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.1}$$

Où : $c_\psi = cos(\psi)$ et $s_\psi = sin(\psi)$.

Deuxième rotation

La deuxième rotation est une rotation de θ autour de l'axe Y' qui fait coïncider l'axe X' avec l'axe X_1 et l'axe Z_0 avec l'axe Z'. Sa matrice de rotation est :

$$R_\theta = \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 1 & c_\theta \end{bmatrix} \tag{1.2}$$

Où : $c_\theta = cos(\theta)$ et $s_\theta = sin(\theta)$.

Troisième rotation

La troisième rotation est une rotation de ϕ autour de l'axe X_1 qui fait coïncider l'axe Z' avec l'axe Z_1 et l'axe Y' avec l'axe Y_1. Sa matrice de rotation est :

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix} \tag{1.3}$$

Où : $c_\phi = cos(\phi)$ et $s_\phi = sin(\phi)$.

Matrice de rotation totale

Nous obtenons enfin la matrice de rotation totale qui nous permet de passer du repère R_1 au repère R_0 en multipliant les trois matrices de rotation précédentes successivement :

$$Rot = R_\psi \cdot R_\theta \cdot R_\phi = \begin{bmatrix} c_\theta c_\psi & -c_\phi s_\psi + s_\phi s_\theta c_\psi & s_\phi s_\psi + c_\phi s_\theta c_\psi \\ c_\theta s_\psi & c_\phi c_\psi + s_\phi s_\theta s_\psi & -s_\phi c_\psi + c_\phi s_\theta s_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{1.4}$$

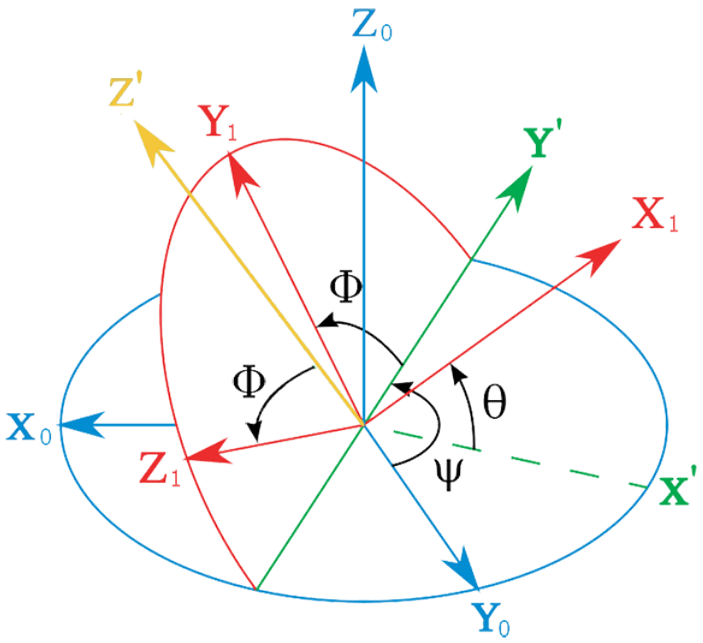


FIGURE 1.13: Passage du repère R_0 au repère R_1

1.6.3 Modèle Dynamique

Dans ce qui suit nous adoptons quelques hypothèses simplificatrices :
- La structure du quadrirotor est supposée rigide et symétrique.
- La matrice d'inertie J est supposée constante (il n'y a pas de changement de poids).
- Les forces de portance et de trainée sont supposées proportionnelles au carré de la vitesse angulaire des rotors.
- Le repère liée au corps du quadrirotor est supposé confondu avec son centre de gravité.

Et nous procédons de la même manière que dans [16, 17].

Dynamique de translation

D'après la seconde loi dynamique de Newton dans le repère inertiel :

$$m\ddot{X} = \sum F_{ext} \tag{1.5}$$

Où :

$m \in \mathcal{R}^+$, est la masse totale du quadrirotor.
$X = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathcal{R}^3$, est le vecteur de position du quadrirotor dans le repère inertiel ;
$\sum F_{ext} \in \mathcal{R}^3$, est le vecteur des forces externes totales ;

Les forces externes appliqués au quadrirotor sont :

le poids C'est la force de gravitation de la terre. Elle est donnée par :

$$P = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \tag{1.6}$$

Où : $g \in \mathcal{R}^+$, est l'accélération de la pesanteur sur terre.

la force de portance C'est la force totale générée par la rotation des hélices des quatre rotors. Elle est dirigée vers le haut, c'est-à-dire qu'elle à tendance à faire élever le quadrirotor. Elle est donnée par :

$$F_p = Rot \cdot \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \tag{1.7}$$

Où :
$T = \sum_{i=1}^{4} f_i$ est la force de portance totale des quatre hélices.
f_i est la force de portance produite par la rotation de l'hélice i, elle est donnée par :

$$f_i = b \cdot \omega_i^2 \tag{1.8}$$

Avec : $b \in \mathcal{R}^+$, le coefficient de portance.

En remplaçant l'expression des forces externes dans l'équation (1.5) nous obtenons après simplification le système d'équations différentielles suivant :

$$\begin{cases} \ddot{x} = \frac{T}{m}(c_\phi s_\theta c_\psi + s_\phi s_\psi) \\ \ddot{y} = \frac{T}{m}(c_\phi s_\theta s_\psi - s_\phi c_\psi) \\ \ddot{z} = \frac{T}{m}(c_\phi c_\theta) - g \end{cases} \tag{1.9}$$

Dynamique de rotation

D'après la seconde loi de la dynamique de Newton :

$$\frac{d(J\Omega)}{dt} = \sum \Gamma_{ext} \tag{1.10}$$

Et comme la vitesse angulaire est exprimée dans le repère lié au quadrirotor, alors :

$$\frac{d(J\Omega)}{dt} = J\dot{\Omega} + \Omega \wedge J\Omega \tag{1.11}$$

Ce qui nous donne :

$$J\dot{\Omega} = -\Omega J\Omega + \sum \Gamma_{ext} \tag{1.12}$$

Où :

$J = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix} \in \mathcal{R}^{33}$, est la matrice d'inertie du quadrirotor ;

$\sum \Gamma_{ext} \in \mathcal{R}^3$, est le vecteur des moments externes totaux ;

$\Omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \in \mathcal{R}^3$, est le vecteur des vitesses de rotations instantanées dans le repère du quadrirotor.

Les couples externes appliqués au quadrirotor sont :

Les couples aérodynamiques Ils sont produits par les forces de trainée et de poussée créées par la rotation des quatre hélices. Ils sont donnée par :

$$\begin{cases} \tau_\phi = l \cdot b(\omega_4^2 - \omega_2^2) \\ \tau_\theta = l \cdot b(\omega_3^2 - \omega_1^2) \\ \tau_\psi = k \cdot (-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{cases} \tag{1.13}$$

Où :

$l \in \mathcal{R}^+$, est la distance entre le centre de gravité du quadrirotor et l'axe de rotation de l'un des rotors ;

$b \in \mathcal{R}^+$, est le coefficient de portance ;

$k \in \mathcal{R}^+$, est la constante de trainée.

En remplaçant l'expression des couples externes dans l'équation (1.12) on obtient après simplification le système d'équations différentielles suivant :

$$\begin{cases} \dot{p} = (\frac{J_y - J_z}{J_x})qr + \frac{\tau_\phi}{J_x} \\ \dot{q} = (\frac{J_z - J_x}{J_y})pr + \frac{\tau_\theta}{J_y} \\ \dot{r} = (\frac{J_x - J_y}{J_z})pq + \frac{\tau_\psi}{J_z} \end{cases} \tag{1.14}$$

Où : $\Omega_r = \omega_1 - \omega_2 + \omega_3 - \omega_4$

Relation entre les angles d'Euler et les vitesses angulaires

Si un solide tourne à une vitesse constante, sa vitesse angulaire Ω est constante, par contre les variations des angles d'Euler seront variables car elles dépendent des angles instantanés entre les axes des deux repères. La séquence des angles d'Euler est obtenue à partir de trois rotations successives : lacet, tangage et roulis. La variation ψ nécessite deux rotations, θ nécessite une rotation et ϕ ne nécessite aucune rotation[18] :

$$\Omega = R_\phi R_\theta \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + R_\phi \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \tag{1.15}$$

Ce qui nous donne :

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & s_\phi c_\theta \\ 0 & -s_\phi & c_\phi c_\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{1.16}$$

Pour des raisons de simplification et comme la plupart des cas étudiés dans la littérature travaillent avec un modèle simplifié [19, 20, 21], on suppose que les angles de roulis et de tangage sont de faible amplitude, i.e. $|\phi| \leq \frac{\pi}{9}$ et $|\theta| \leq \frac{\pi}{9}$, ce qui nous permet d'avoir $sin(\phi) \approx \phi$, $sin(\theta) \approx \theta$, $cos(\phi) \approx 1$ et $cos(\theta) \approx 1$ et on suppose aussi que les vitesses angulaires autour des trois axes du quadrirotor sont faibles, et donc l'équation (1.16) devient :

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} - s_\theta \dot{\psi} \\ c_\phi \dot{\theta} + s_\phi c_\theta \dot{\psi} \\ -s_\phi \dot{\theta} + c_\phi c_\theta \dot{\psi} \end{bmatrix} \approx \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{1.17}$$

Et l'équation (1.14) devient :

$$\begin{cases} \ddot{\phi} = (\frac{J_y - J_z}{J_x})\dot{\theta}\dot{\psi} + \frac{\tau_\phi}{J_x} \\ \ddot{\theta} = (\frac{J_z - J_x}{J_y})\dot{\phi}\dot{\psi} + \frac{\tau_\theta}{J_y} \\ \ddot{\psi} = (\frac{J_x - J_y}{J_z})\dot{\phi}\dot{\theta} + \frac{\tau_\psi}{J_z} \end{cases} \tag{1.18}$$

Dynamique des rotors

Les quadrirotors sont généralement équipés de moteurs brushless ou BLDC, qui sont des moteurs sans balais et qui requièrent donc que chaque phase soit alimentée par une source externe durant la bonne période de rotation. Pour ce faire, des circuits de puissances appelés ESC (Electronic Speed Controllers) sont utilisés pour faire la fonction d'onduleur et de circuit de commande. La paire : moteur BLDC et ESC, est alimenté par une tension continue et se comporte donc comme un moteur à courant continu, d'où le nom "Brushless DC" (Moteur à courant continu sans balai).

Dans ce qui suit, nous allons considérer la paire, moteur BLDC et ESC, comme étant équivalente à un moteur à courant continu avec balai et nous notons V_{EQ} la tension efficace appliquée par l'ESC aux bornes du moteur BLDC, figure 1.14.

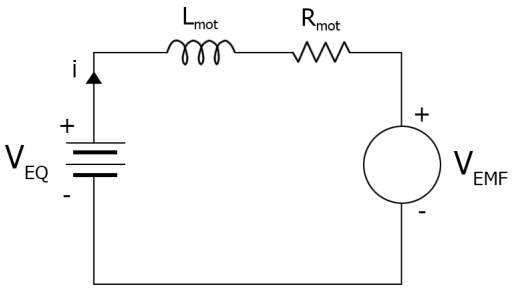


FIGURE 1.14: Circuit électrique équivalent

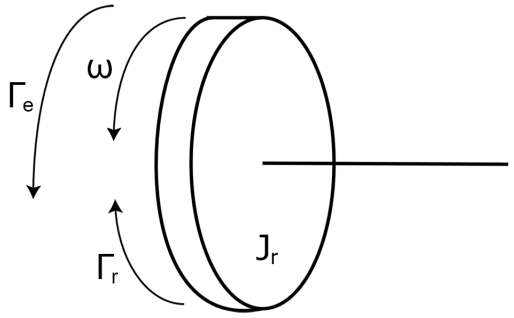


FIGURE 1.15: Partie mécanique du moteur

En utilisant la loi des mailles on obtient :

$$V_{EQ} = R_{mot}i + L_{mot} \cdot \frac{di}{dt} + V_{EMF} \tag{1.19}$$

Avec :
$V_{EMF} = K_e \cdot \omega$, la force contre-électromotrice,
ω, la vitesse angulaire du rotor,
K_e, la constante électrique du moteur,
L_{mot}, l'inductance équivalente du moteur,
R_{mot}, la résistance équivalente du moteur.
Pour les moteurs de petites tailles, comme ceux utilisés pour les quadrirotors, l'inductance est très petite et donc la partie électrique est beaucoup plus rapide que la partie mécanique. L'équation (1.19) devient :

$$V_{EQ} = R_{mot}i + K_e\omega \tag{1.20}$$

Pour la partie mécanique, figure 1.15, nous utilisons la seconde loi de Newton et nous obtenons :

$$J_r\dot{\omega} = \Gamma_e + \Gamma_r \tag{1.21}$$

Avec :
$\Gamma_e = K_e i$, le couple électrique fournie par le moteur,
$\Gamma_r = -f_r\omega$, le couple dû au frottement de l'air.
L'équation (1.21) devient alors :

$$J_r\dot{\omega} = K_e i - f_r\omega \tag{1.22}$$

En utilisant la transformée de Laplace sur les équations (1.20) et (1.22) et en simplifiant nous obtenons :

$$\frac{\omega(s)}{V_{EQ}(s)} = \frac{1}{\frac{R_{mot} \cdot J_r}{K_e}s + f_r + K_e} \tag{1.23}$$

La tension équivalente V_{EQ} est la tension générée par l'ESC en fonction du signal de commande u_{PWM} qui représente la largeur du signal PWM (Pulse Width Modulation ou Modulation à Largeur d'Impulsion (MLI) en Français) reçu en entrée. La relation entre V_{EQ} et u_{PWM} est donnée par :

$$V_{EQ} = u_{PWM}V_B \tag{1.24}$$

Avec :
V_B, la tension disponible en entrée.
L'expression de u_{PWM} est donnée par :

$$u_{PWM}(t) = \frac{P(t) - P_{max}}{P_{max} - P_{min}} \tag{1.25}$$

P, la largeur de l'impulsion à l'instant t,
P_{max}, la largeur maximale de l'impulsion,
P_{min}, la largeur minimale de l'impulsion.
L'équation (1.23) devient enfin :

$$\frac{\omega(s)}{u_{PWM}(s)} = \frac{K}{Ts + 1} \tag{1.26}$$

Avec : $K = \frac{V_B}{f_r + K_e}$, et $T = \frac{R_{mot} J_r}{K_e(f_r + K_e)}$.

Du fait de la taille relativement petite des moteurs, nous négligeons dans ce qui suit leur dynamique par rapport à celle du quadrirotor. Nous obtenons enfin :

$$\frac{\omega}{u_{PWM}} = K \tag{1.27}$$

Effets aérodynamiques et incertitudes

Il y a beaucoup d'effets aérodynamiques et gyroscopiques associés à un quadrirotor qui modifient le modèle présenté ci-dessus. La plupart de ces effets provoquent seulement des perturbations mineures et ne justifient pas d'être prisent en compte, même si elles sont importantes pour le la conception d'un système complet. Le battement des hélices et la traînée induite, cependant, sont des effets fondamentaux qui sont d'importance significative dans la compréhension de la stabilité naturelle des quadrirotors. Ces effets sont particulièrement importants car ils induisent des forces dans le plan x-y du quadrirotor, ses directions sous-actionnées, qui ne peuvent pas être facilement dominé par une commande à gain élevé.

le frottement de l'air Le châssis du quadrotor ainsi que les hélices offrent une résistance à l'air. Celle-ci génère une force de friction qui s'oppose au mouvement linéaire et rotatif du quadrirotor. Cette force est proportionnelle au carré de la différence entre la vitesse du quadrirotor et celle du vent et elle dépend de la géométrie du quadrirotor. Son expression est donnée par[22, 23] :

$$F_r = \begin{bmatrix} -A_x|\dot{x} - w_x|(\dot{x} - w_x) \\ -A_y|\dot{y} - w_y|(\dot{y} - w_y) \\ -A_z|\dot{z} - w_z|(\dot{z} - w_z) \end{bmatrix} \tag{1.28}$$

Où :
$A_x \in \mathcal{R}^+$, est le coefficient de frottement visqueux suivant l'axe X_0 ;
$A_y \in \mathcal{R}^+$, est le coefficient de frottement visqueux suivant l'axe Y_0 ;
$A_z \in \mathcal{R}^+$, est le coefficient de frottement visqueux suivant l'axe Z_0 ;
$w_x \in \mathcal{R}^+$, est la vitesse du vent suivant l'axe X_0 ;
$w_y \in \mathcal{R}^+$, est la vitesse du vent suivant l'axe Y_0 ;
$w_z \in \mathcal{R}^+$, est la vitesse du vent suivant l'axe Z_0.

Le moment gyroscopique Il se crée dans les systèmes physiques en mouvement avec des parties rotatoires et tend à résister aux mouvements du quadrirotor. L'expression générale de ce moment est donnée par :

$$\tau_{gyro} = \sum_{i=1}^{4} \Omega \wedge J_r \begin{bmatrix} 0 \\ 0 \\ (-1)^i \omega_i \end{bmatrix} \tag{1.29}$$

La matrice d'inertie de chaque rotor est supposé diagonale :

$$J_r = \begin{bmatrix} J_{rx} & 0 & 0 \\ 0 & J_{ry} & 0 \\ 0 & 0 & J_{rz} \end{bmatrix} \tag{1.30}$$

On obtient enfin :

$$\tau_{gyro} = \sum_{i=1}^{4} [(-1)^i \omega_i J_{rz}] \Omega \wedge \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{1.31}$$

battement d'hélice L'effet de battement d'hélice (ou "blade flapping" en anglais) est créé lorsque cette dernière se déplace horizontalement. Ce déplacement crée une différence de vitesse, et donc de poussée, entre la partie de d'hélice qui attaque le flot d'air par rapport à celle qui se retire du flot d'air. Cette différence de poussée entre ces éléments cause le plan de l'hélice à s'incliner, ce qui change la direction du vecteur de poussée[22, 23].

l'effet de sol L'effet de sol est créé lorsqu'une surface, qui est suffisamment près de l'hélice, perturbe le flot d'air généré par celle-ci en plus d'améliorer la poussée de l'hélice. A basse vitesse, cet effet peut être modélisé par[24, 23] :

$$\frac{|T_{ES}|}{|T_0|} = \frac{1}{1 - (\frac{r}{4h})^2} \tag{1.32}$$

Où :

$T_{ES} \in \mathcal{R}^+$, est la poussée générée par l'hélice avec l'effet de sol ;

$T_0 \in \mathcal{R}^+$, est la poussée générée par l'hélice sans l'effet de sol ;

$r \in \mathcal{R}^+$, est le rayon de l'hélice ;

$h \in \mathcal{R}^+$, est la hauteur de l'hélice par rapport au sol ;

On remarque que la poussé augmente grâce à l'effet de sol, mais cette effet diminue considérablement même à de faibles hauteurs. L'augmentation de la poussé n'est que de 7% lorsque la hauteur est égale au rayon de l'hélice[24].

Dans ce qui suit, nous traitons tout ces effets ainsi que toutes les variations due au simplifications que nous avons effectué et au éventuelle erreurs de modélisation comme étant des incertitudes et des perturbations et nous les regroupons dans le vecteur $\rho_d \in \mathcal{R}^4$ que nous ajoutons au équations (1.9) et (1.18) pour qu'elle deviennent :

$$\begin{cases} \ddot{x} = \frac{T}{m}(c_\phi s_\theta c_\psi + s_\phi s_\psi) + \rho_{dx} \\ \ddot{y} = \frac{T}{m}(c_\phi s_\theta s_\psi - s_\phi c_\psi) + \rho_{dy} \\ \ddot{z} = \frac{T}{m}(c_\phi c_\theta) - g + \rho_{dz} \end{cases} \tag{1.33}$$

$$\begin{cases} \ddot{\phi} = (\frac{J_y - J_z}{J_x})\dot{\theta}\dot{\psi} + \frac{\tau_\phi}{J_x} + \rho_{d\phi} \\ \ddot{\theta} = (\frac{J_z - J_x}{J_y})\dot{\phi}\dot{\psi} + \frac{\tau_\theta}{J_y} + \rho_{d\theta} \\ \ddot{\psi} = (\frac{J_x - J_y}{J_z})\dot{\phi}\dot{\theta} + \frac{\tau_\psi}{J_z} + \rho_{d\psi} \end{cases} \tag{1.34}$$

1.6.4 Modèle d'état

Pour mettre les équations du quadrirotor sous forme d'état, on choisit comme vecteur d'état :

$$X = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \phi & \theta & \psi & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T \tag{1.35}$$

Et les commandes :

$$\begin{cases} U_1 = T = b(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ U_2 = \tau_\phi = l \cdot b(\omega_4^2 - \omega_2^2) \\ U_3 = \tau_\theta = l \cdot b(\omega_3^2 - \omega_1^2) \\ U_4 = \tau_\psi = k \cdot (-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{cases} \tag{1.36}$$

La représentation d'état obtenue est la suivante :

$$\begin{cases} \dot{x}_1 = x_4 \\ \dot{x}_2 = x_5 \\ \dot{x}_3 = x_6 \\ \dot{x}_4 = \frac{U_1}{m}(c_{x_7} s_{x_8} c_{x_9} + s_{x_7} s_{x_9}) + \rho_{dx} \\ \dot{x}_5 = \frac{U_1}{m}(c_{x_7} s_{x_8} s_{x_9} - s_{x_7} c_{x_9}) + \rho_{dy} \\ \dot{x}_6 = \frac{U_1}{m}(c_{x_7} c_{x_8}) - g + \rho_{dz} \\ \dot{x}_7 = x_{10} \\ \dot{x}_8 = x_{11} \\ \dot{x}_9 = x_{12} \\ \dot{x}_{10} = (\frac{J_y - J_z}{J_x}) x_{11} x_{12} + \frac{U_2}{J_x} + \rho_{d\phi} \\ \dot{x}_{11} = (\frac{J_z - J_x}{J_y}) x_{10} x_{12} + \frac{U_3}{J_y} + \rho_{d\theta} \\ \dot{x}_{12} = (\frac{J_x - J_y}{J_z}) x_{10} x_{11} + \frac{U_4}{J_z} + \rho_{d\psi} \end{cases} \tag{1.37}$$

Comme on peut le remarquer, le système est naturellement divisé en deux sous-systèmes couplés : un sous-système translationnel et un sous-système rotationnel.

1.7 Synthèse de la Commande par Mode Glissant

Le quadrirotor à fait l'objet de beaucoup de recherches dans le domaine de la commandes, ainsi plusieurs lois de commandes ont été proposées. Parmi ces dernières on trouve : la commande PID[25], la commande LQR[26], le backstepping[27, 28], la commande par mode glissant[29].

Nous avons opté pour la commande par mode glissant, car elle offre de bonnes performances et qu'elle est insensible aux perturbations externes[30].

D'après les équations (1.33) et (1.6.3), le modèle dynamique du quadrirotor peut se mettre sous la forme :

$$\ddot{\xi} = f + U + \rho_d \tag{1.38}$$

Avec :

$$\xi = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} \tag{1.39}$$

$$f = \begin{bmatrix} 0 \\ 0 \\ -g \\ (\frac{J_y - J_z}{J_x})x_{11}x_{12} \\ (\frac{J_z - J_x}{J_y})x_{10}x_{12} \\ (\frac{J_x - J_y}{J_z})x_{10}x_{11} \end{bmatrix} \tag{1.40}$$

$$U = \begin{bmatrix} \frac{U_1}{m}(c_{x_7}s_{x_8}c_{x_9} + s_{x_7}s_{x_9}) \\ \frac{U_1}{m}(c_{x_7}s_{x_8}s_{x_9} - s_{x_7}c_{x_9}) \\ \frac{U_1}{m}(c_{x_7}c_{x_8}) \\ \frac{U_2}{J_x} \\ \frac{U_3}{J_y} \\ \frac{U_3}{J_z} \end{bmatrix} \tag{1.41}$$

$$\rho_d = \begin{bmatrix} \rho_{dx} \\ \rho_{dy} \\ \rho_{dz} \\ \rho_{d\phi} \\ \rho_{d\theta} \\ \rho_{d\psi} \end{bmatrix} \tag{1.42}$$

On introduit la commande virtuelle $v = f + U$, ce qui nous donne :

$$\ddot{\xi} = v + \rho_d \tag{1.43}$$

1.7.1 Surface de glissement

Soit l'erreur de poursuite :

$$\epsilon(t) = \xi(t) - \xi_d(t) \tag{1.44}$$

Où $\xi_d = \begin{bmatrix} x_d & y_d & z_d & \phi_d & \theta_d & \psi_d \end{bmatrix}^T$ est le vecteur de position et d'attitude désirés. Puisque chaque état est d'ordre relatif 2 par rapport à sa commande, on choisit une surface de slotine d'ordre 2[31] :

$$S_i(t) = \dot{\epsilon}_i(t) + \lambda_i \epsilon_i(t) \quad i = 1, 2 \dots 6 \tag{1.45}$$

Où les $\lambda_i \in \mathcal{R}^+$ sont des constantes de réglages.

1.7.2 Commande virtuelle

L'objectif de commande est de forcer le système sur la surface de glissement et de l'empêcher d'en sortir, i.e. :

$$S_i = 0 \quad i = 1, 2 \dots 6 \tag{1.46}$$

La dynamique de la surface (1.45) est :

$$\dot{S}_i(t) = \ddot{\epsilon}_i(t) + \lambda_i \dot{\epsilon}_i(t) \quad i = 1, 2 \dots 6 \tag{1.47}$$

Où encore :

$$\dot{S}_i(t) = v_i(t) + \rho_{d_i}(t) - \ddot{\xi_{d_i}}(t) + \lambda_i(\dot{\xi}_i(t) - \dot{\xi_{d_i}}(t)) \quad i = 1, 2 \dots 6 \tag{1.48}$$

On désire avoir la dynamique suivante :

$$\dot{S}_i(t) = -K_i sign(S_i(t)) - Q_i S_i(t) \quad i = 1, 2 \dots 6 \tag{1.49}$$

Où les $K_i \in \mathcal{R}^+$ et les $Q_i \in \mathcal{R}^+$ sont des constantes de réglages et la fonction sign(.) est définie comme suit[31] :

$$sign(x) = \begin{cases} 1 & \text{si} \quad x > 0 \\ 0 & \text{si} \quad x = 0 \\ -1 & \text{si} \quad x < 0 \end{cases} \tag{1.50}$$

En égalant (1.49) et (1.48) on obtient après développement :

$$v_i(t) = \hat{u}_i - K_i sign(S_i(t)) - Q_i S_i(t) \quad i = 1, 2 \dots 6 \tag{1.51}$$

Avec :

$$\hat{u}_i = \ddot{\xi_{d_i}}(t) - \lambda_i(\dot{\xi}_i(t) - \dot{\xi_{d_i}}(t)) \quad i = 1, 2 \dots 6 \tag{1.52}$$

Et on obtient enfin les six commandes virtuelles suivantes :

$$\begin{cases} v_x = \ddot{x_d} - \lambda_1(x_7 - \dot{x_d}) - K_1 sign(S_1) - Q_1 S_1 \\ v_y = \ddot{y_d} - \lambda_2(x_8 - \dot{y_d}) - K_2 sign(S_2) - Q_2 S_2 \\ v_z = \ddot{z_d} - \lambda_3(x_9 - \dot{z_d}) - K_3 sign(S_3) - Q_3 S_3 \\ v_\phi = \ddot{\phi_d} - \lambda_4(x_{10} - \dot{\phi_d}) - K_4 sign(S_4) - Q_4 S_4 \\ v_\theta = \ddot{\theta_d} - \lambda_5(x_{11} - \dot{\theta_d}) - K_5 sign(S_5) - Q_5 S_5 \\ v_\psi = \ddot{\psi_d} - \lambda_6(x_{12} - \dot{\psi_d}) - K_6 sign(S_6) - Q_6 S_6 \end{cases} \tag{1.53}$$

1.7.3 Commande réelle

La commande obtenue en (1.51) est une commande virtuelle. Nous allons maintenant déterminer les commandes réelles U_1, U_2, U_3, U_4 à partir de $v = \begin{bmatrix} v_x & v_y & v_z & v_\phi & v_\theta & v_\psi \end{bmatrix}^T$

D'après l'équation (1.38), on a :

$$\begin{bmatrix} (c_{x_7} s_{x_8} c_{x_9} + s_{x_7} s_{x_9}) \\ c_{x_7} s_{x_8} s_{x_9} - s_{x_7} c_{x_9}) \\ (c_{x_7} c_{x_8}) \end{bmatrix} \frac{U_1}{m} = \begin{bmatrix} v_x \\ v_y \\ v_z + g \end{bmatrix} \tag{1.54}$$

$$\begin{bmatrix} (\frac{J_y - J_z}{J_x}) x_{11} x_{12} \\ (\frac{J_z - J_x}{J_y}) x_{10} x_{12} \\ (\frac{J_x - J_y}{J_z}) x_{10} x_{11} \end{bmatrix} + \begin{bmatrix} \frac{U_2}{J_x} \\ \frac{U_3}{J_y} \\ \frac{U_3}{J_z} \end{bmatrix} = \begin{bmatrix} v_\phi \\ v_\theta \\ v_\psi \end{bmatrix} \tag{1.55}$$

Dans l'équation (1.54), nous avons trois équations pour déterminer la commande U_1. Pour cela, on remarque que :

$$\begin{bmatrix} (c_{x_7} s_{x_8} c_{x_9} + s_{x_7} s_{x_9}) \\ c_{x_7} s_{x_8} s_{x_9} - s_{x_7} c_{x_9}) \\ (c_{x_7} c_{x_8}) \end{bmatrix} \frac{U_1}{m} = Rot \cdot \frac{U_1}{m} \tag{1.56}$$

Or on sait que, d'après les propriétés des matrices de rotation[32] :

$$||Rot \cdot \frac{U_1}{m}|| = ||\frac{U_1}{m}|| \tag{1.57}$$

Ce qui nous donne enfin en prenant le module des deux côté de l'équation (1.54) :

$$U_1 = m\sqrt{v_x^2 + v_y^2 + (v_z + g)^2} \tag{1.58}$$

Dans l'équation (1.55), nous obtenons facilement les trois commandes U_2, U_3, U_4 :

$$\begin{cases} U_2 = J_x v_\phi - (J_y - J_z) x_{11} x_{12} \\ U_3 = J_y v_\theta - (J_z - J_x) x_{10} x_{12} \\ U_4 = J_z v_\psi - (J_x - J_y) x_{10} x_{11} \end{cases} \tag{1.59}$$

1.7.4 Orientation désirée

L'angle de lacet désiré ϕ_d est donné par l'opérateur ou le générateur de trajectoire, tandis que les angles de roulis et tangage désirés ϕ_d et θ_d sont générés à partir de l'équation (1.54) :

$$\begin{cases} m(v_x c_\psi + v_y s_\psi) = s_\theta U_1 \\ m(v_x s_\psi - v_y c_\psi) = s_\phi c_\theta U_1 \\ m(v_z + g) = c_\phi c_\theta U_1 \end{cases} \tag{1.60}$$

On tire de l'équation (1.60) :

$$\begin{cases} \phi_d = \arctan(\frac{v_x s_{\psi_d} - v_y c_{\psi_d}}{v_z + g}) \\ \theta_d = \arctan(\frac{v_x c_{\psi_d} + v_y s_{\psi_d}}{\sqrt{(v_x s_{\psi_d} - v_y c_{\psi_d})^2 + (v_z + g)^2}}) \end{cases} \tag{1.61}$$

1.7.5 Vitesses angulaire des moteurs

On remarque que les commandes de l'équation (1.36) peuvent se mettre sou forme matricielle :

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ 0 & -l \cdot b & 0 & l \cdot b \\ -l \cdot b & 0 & l \cdot b & 0 \\ -k & k & -k & k \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{1.62}$$

Et de là on tire l'expression de la vitesse angulaire de chaque rotor :

$$\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \begin{bmatrix} \frac{1}{4b} & 0 & -\frac{1}{2l \cdot b} & -\frac{1}{4k} \\ \frac{1}{4b} & -\frac{1}{2l \cdot b} & 0 & \frac{1}{4k} \\ \frac{1}{4b} & 0 & \frac{1}{2l \cdot b} & -\frac{1}{4k} \\ \frac{1}{4b} & \frac{1}{2l \cdot b} & 0 & \frac{1}{4k} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{1.63}$$

1.7.6 Analyse de stablité

Comme on peut le voir, la dynamique présenté dans l'équation (1.43) est sous une forme totalement découplée, ce qui veut dire que la commande obtenu en (1.51) à été synthétisée indépendamment pour chaque élément de ξ (chaque degré de liberté). Pour étudier la stabilité de chacune de ses commandes, on considère les fonctions de Lyapunov suivantes :

$$V_i = \tfrac{1}{2} S_i^2 > 0, \quad i = 1, 2, \ldots, 6 \tag{1.64}$$

En dérivant chacune de ses fonctions de Lyapunov et en considérant les équations (1.48) et (1.51) on obtient :

$$\dot{V}_i = S_i(-K_i sign(S_i) - Q_i S_i - \rho_{d_i}), \quad i = 1, 2, \ldots, 6 \tag{1.65}$$

On suppose maintenant qu'il existe des constantes positives γ_i telles que :

$$|\rho_{d_i}| < \gamma_i, \quad i = 1, 2, \ldots, 6 \tag{1.66}$$

Ce qui nous donne :

$$\dot{V}_i < S_i(-K_i sign(S_i) - Q_i S_i - \gamma_i), \quad i = 1, 2, \ldots, 6 \tag{1.67}$$

En choisissant les constantes K_i et Q_i telles que :

$$\begin{cases} K_i > \gamma_i, \\ Q_i > 0, \quad i = 1, 2, \ldots, 6 \end{cases} \tag{1.68}$$

On aura enfin :

$$\begin{gathered} \dot{V}_i < S_i(-K_i sign(S_i) - Q_i S_i - \gamma_i) = -Q_i S_i^2 - S_i(K_i sign(S_i) - \gamma_i) < 0 \\ i = 1, 2, \ldots, 6 \end{gathered} \tag{1.69}$$

Car :

$$-Q_i S_i^2 < 0, \quad \forall Q_i > 0 \tag{1.70}$$

Et :

$$\begin{cases} S_i > 0 \implies sign(S_i) = 1 \\ \implies -S_i(K_i sign(S_i) - \gamma_i) = -S_i(K_i - \gamma_i) < 0 & \forall K_i > \gamma_i \\ \\ S_i < 0 \implies sign(S_i) = -1 \\ \implies -S_i(K_i sign(S_i) - \gamma_i) = -S_i(-K_i - \gamma_i) = S_i(K_i + \gamma_i) < 0 & \forall K_i > \gamma_i > 0 \end{cases} \tag{1.71}$$

En respectant ses conditions, les V_i seront définie négatives et la stabilité de chaque élément de ξ (chaque degré de liberté) ainsi que celle du système en sa globalité sera garantie.

1.7.7 Phénomène de chattering

La commande par mode glissant contient en général deux termes : un terme discontinu qui assure l'invariance ou la robustesse et qui permet de forcer l'état du système à rester sur la surface de glissement, et un deuxième terme, appelé commande équivalente, qui permet d'assurer la convergence du système vers la surface de glissement. Un régime glissant idéal requiert une commande pouvant commuter à une fréquence infinie. Évidemment, pour une utilisation pratique, seule une commutation à une fréquence finie est possible, ce qui cause un retard entre la mesure de la sortie et le calcul de la commande. Cela conduit le système à quitter la surface de glissement sans que la commande puisse réagir[33]. Ainsi, durant le régime glissant, les discontinuités de la commande peuvent entrainer des oscillations en haute fréquence de la trajectoire du système autour de la surface de glissement,ce phénomène est appelé "broutement" ou "chattering". En conséquence, les performances et la robustesse du système sont dégradées et cela peut même conduire à l'instabilité[33].

De nombreuses solutions existent pour réduire ou éliminer ce phénomène. L'une des solutions consiste à remplacer la fonction discontinu $sign(.)$ de la loi de commande par une fonction continue. Parmi les fonctions les plus utilisées on trouve :

La fonction saturation

$$v_1 = sat(x, \delta) = \begin{cases} sign(\delta) & si \quad |x| \geq \delta \\ \frac{x}{\delta} & si \quad |x| < \delta \end{cases} \tag{1.72}$$

La fonction pseudo-signe

$$v_2 = psigne(x, \delta) = \frac{x}{|x| + \delta} \tag{1.73}$$

La fonction arctangente

$$v_3 = \frac{2}{\pi} arctan(\frac{x}{\delta}) \tag{1.74}$$

La fonction tangente hyperbolique

$$v_4 = tanh(\frac{x}{\delta}) \tag{1.75}$$

Avec $\delta \in \mathcal{R}^+$, une constante de réglage.

La fonction que nous avons choisi pour réduire la chattering des commandes générées est la fonction $pseudo - signe(.)$ de par sa simplicité.

1.8 Simulation

Dans ce qui suit nous allons simuler la dynamique du quadrirotor lorsqu'on lui applique la commande synthétisé en (1.58) et (1.59). Les paramètres physiques du quadrirotor simulé et de la commande sont disponible en annexe, table 1.1, 1.2, 1.3 et 1.4.

Pour la première simulation, nous avons donné comme référence un point de coordonnées $\{2, -2, 3\}$ et comme perturbation un vent constant de vitesse égale à $0.5m/s$. Nous pouvons voir les résultats de la simulation sur les figures 1.16, 1.17, 1.18 et 1.19. On constate que le quadrirotor atteint rapidement le point de référence malgré la présence de perturbations dues au frottement avec l'air et au vent.

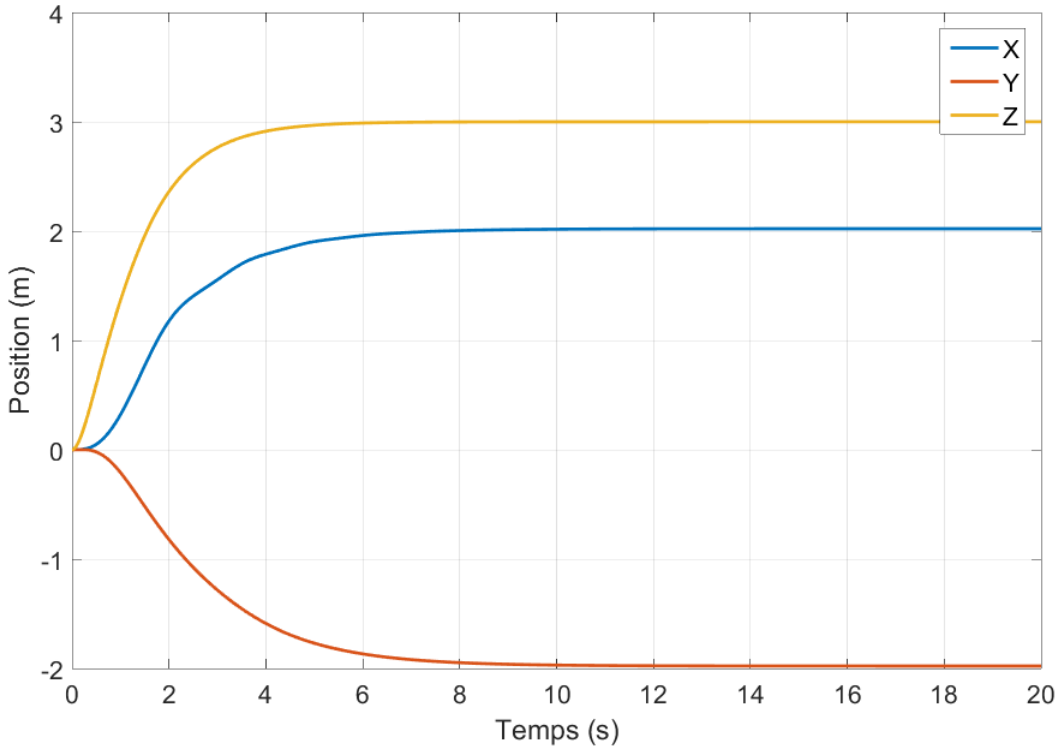


FIGURE 1.16: Évolution de la position du quadrirotor en fonction du temps

Nous avons aussi vérifié l'hypothèse énoncée dans l'équation 1.17 comme on peut le voir sur les figures 1.20, 1.21 et 1.22.

Pour la deuxième simulation, nous avons donné comme référence les sommets d'un carré centré à l'origine et dont la longueur de ses quatre côtés est égale à quatre. Nous pouvons voir les résultats de la simulation sur les figures 1.23, 1.24, 1.25 et 1.26. On constate que le quadrirotor atteint rapidement les points de référence malgré la présence de perturbations dues au frottement avec l'air et au vent.

Nous remarquons une légère ondulation sur les courbes de la figure 1.23 et sur la courbe 1.26 à chaque fois que le quadrirotor se dirige vers le sommet suivant du carré. Cela est dû au fait, qu'à chaque fois qu'il se dirige vers un nouveau sommet, il s'oriente aussi vers se dernier et le changement d'orientation se produit en même temps que le changement de position, ce qui fait qu'il s'éloigne un relativement peu de la trajectoire avant d'y revenir.

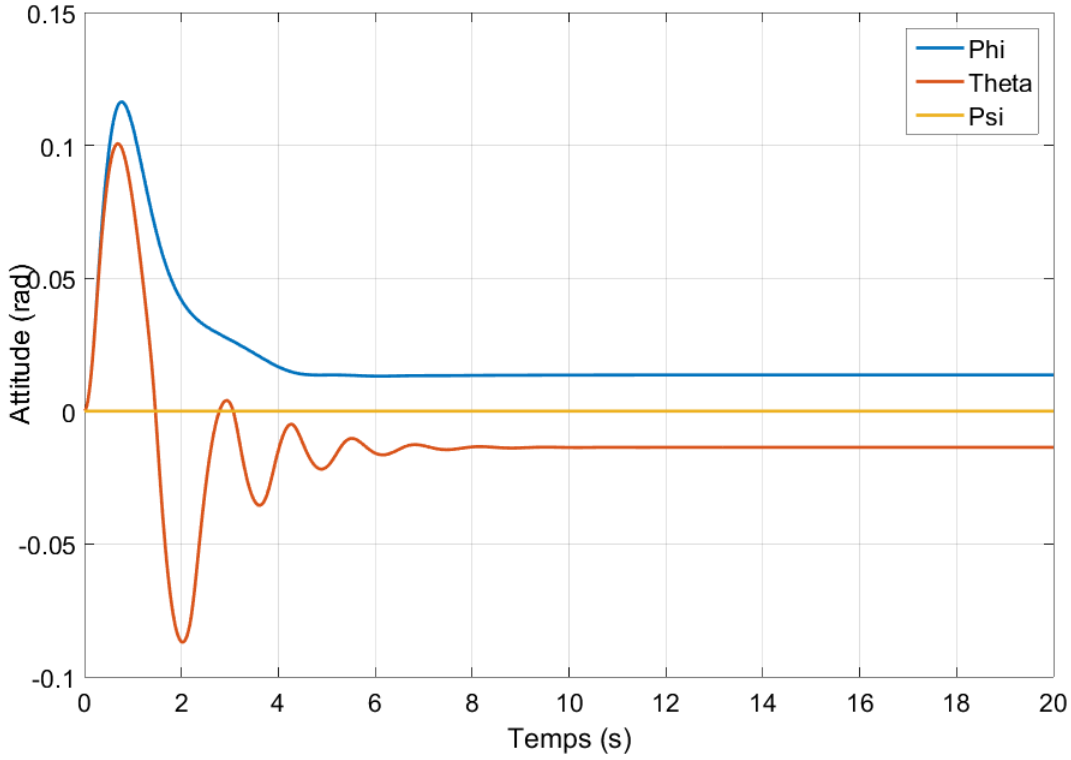


FIGURE 1.17: Évolution de l'attitude du quadrirotor en fonction du temps

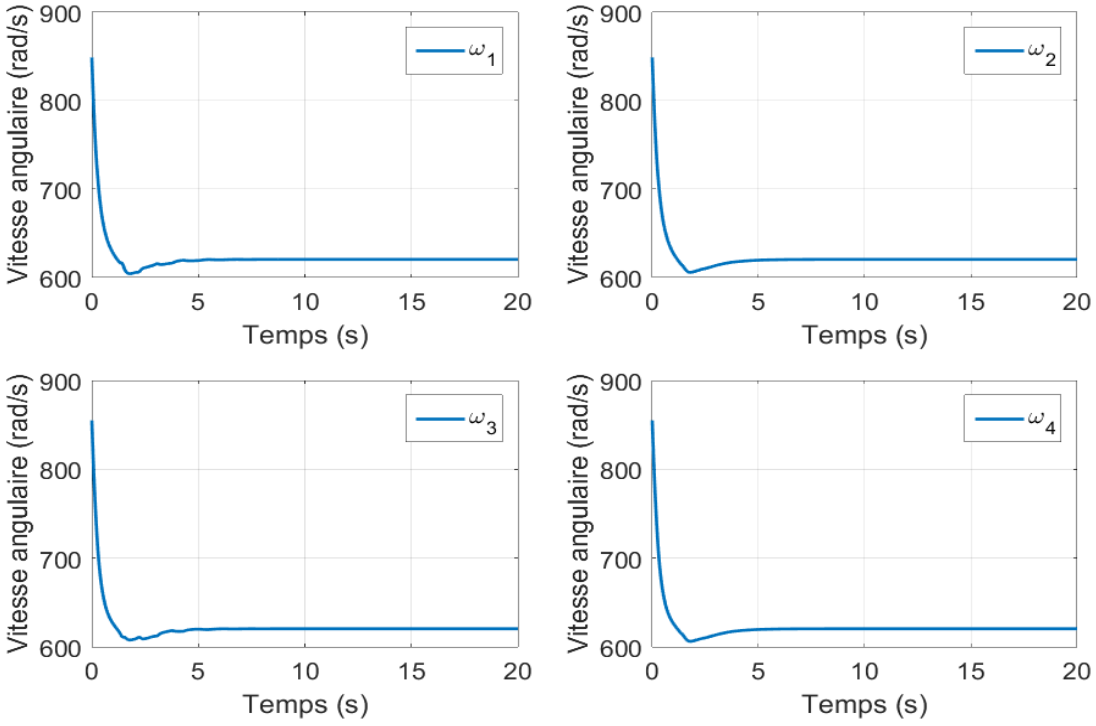


FIGURE 1.18: Évolution de la vitesse angulaire des quatre rotors du quadrirotor en fonction du temps

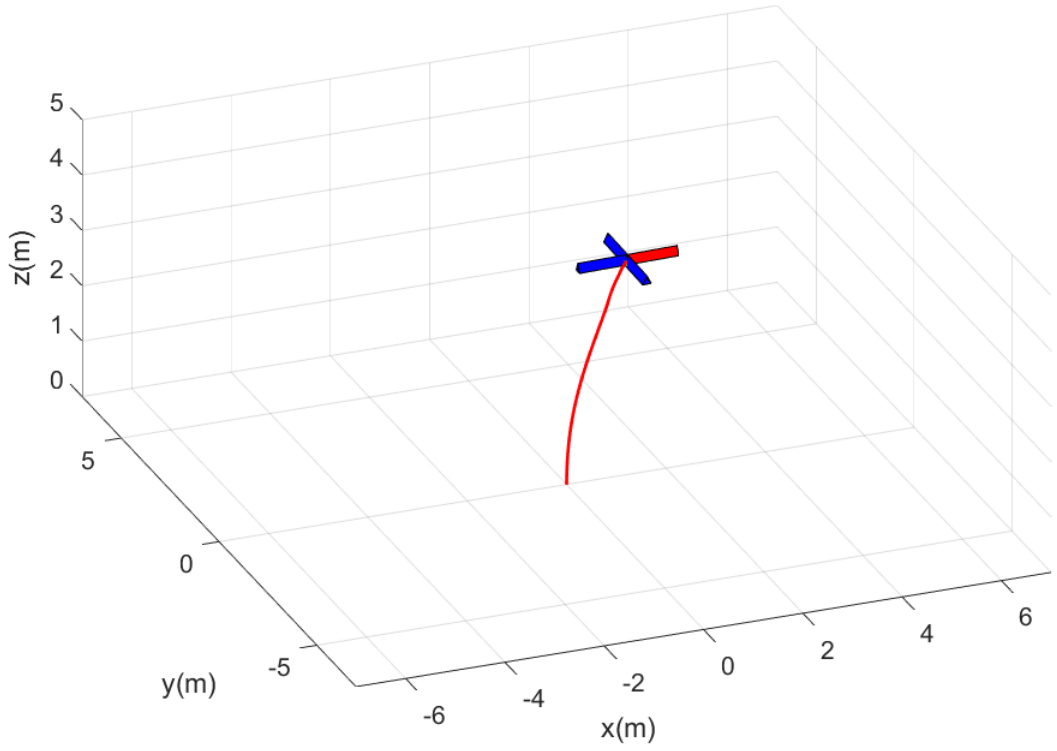


FIGURE 1.19: Évolution en 3D du quadrirotor en fonction du temps

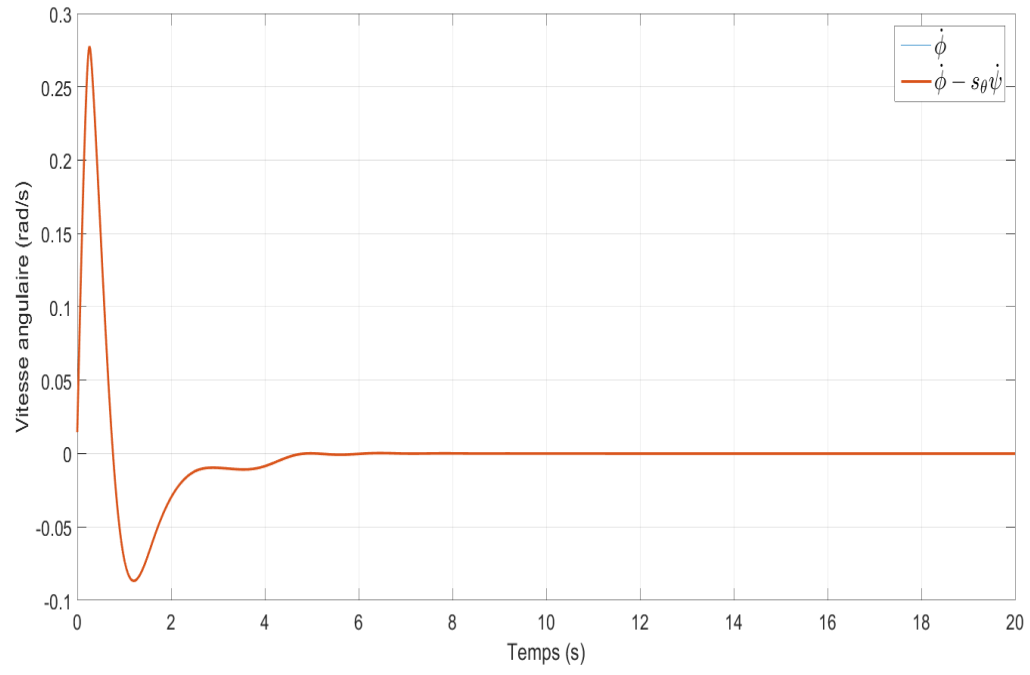


FIGURE 1.20: Vérification de la validité de l'approximation pour $\dot{\phi}$

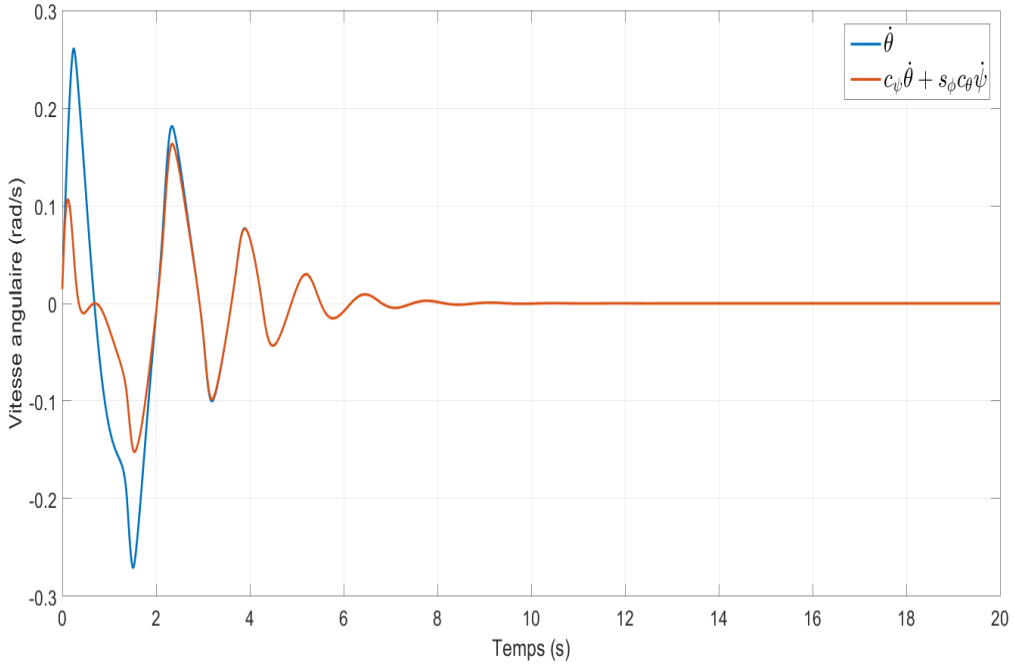


FIGURE 1.21: Vérification de la validité de l'approximation pour $\dot{\theta}$

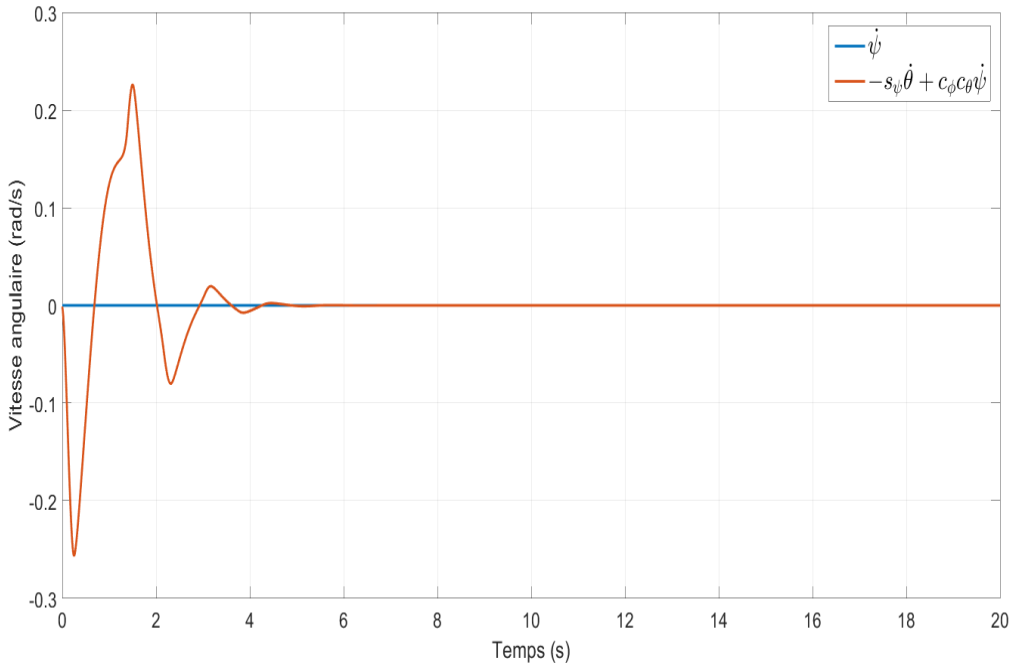


FIGURE 1.22: Vérification de la validité de l'approximation pour $\dot{\psi}$

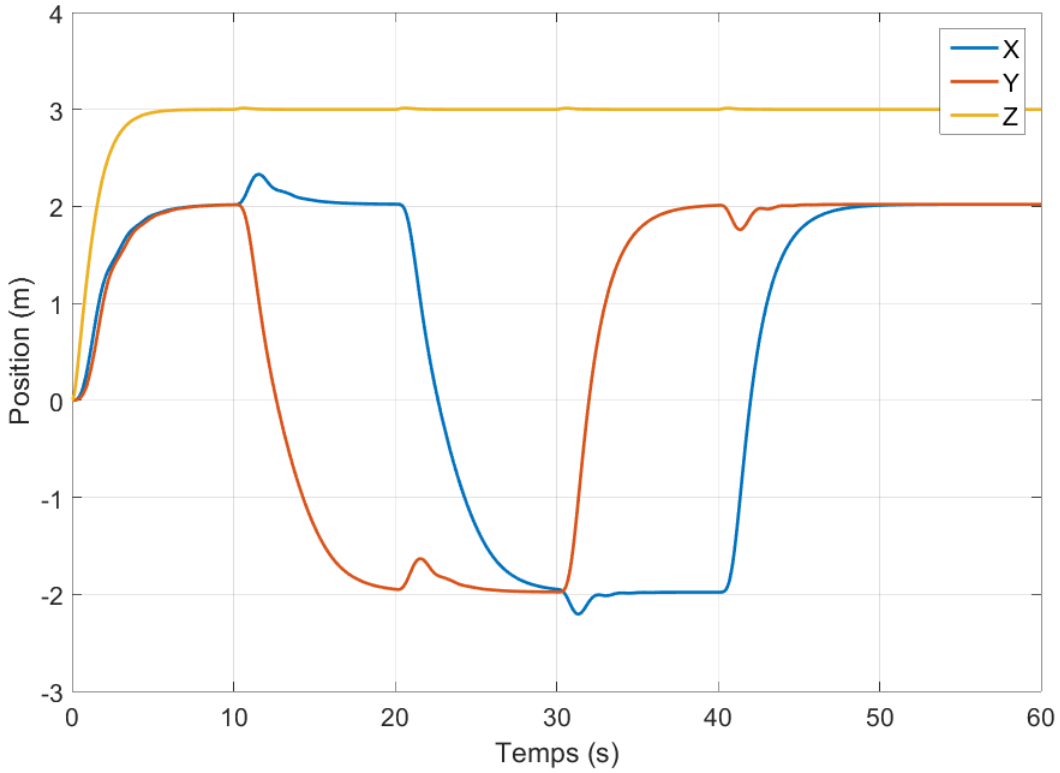


FIGURE 1.23: Évolution de la position du quadrirotor en fonction du temps

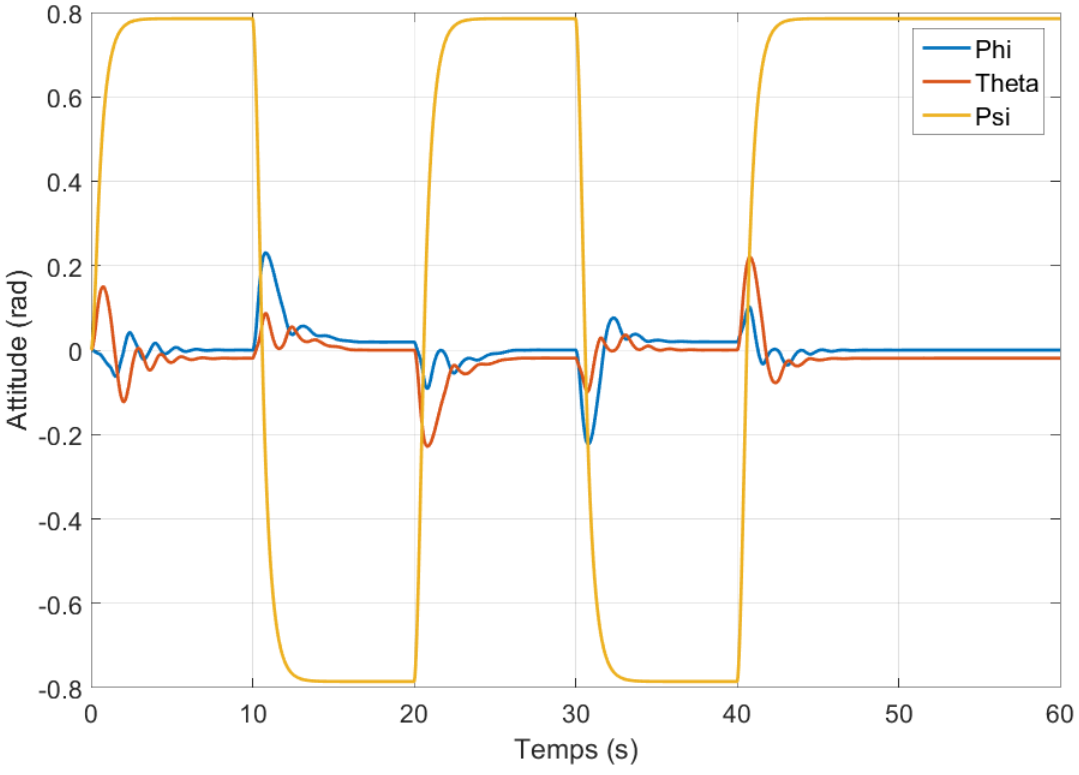


FIGURE 1.24: Évolution de l'attitude n du quadrirotor en fonction du temps

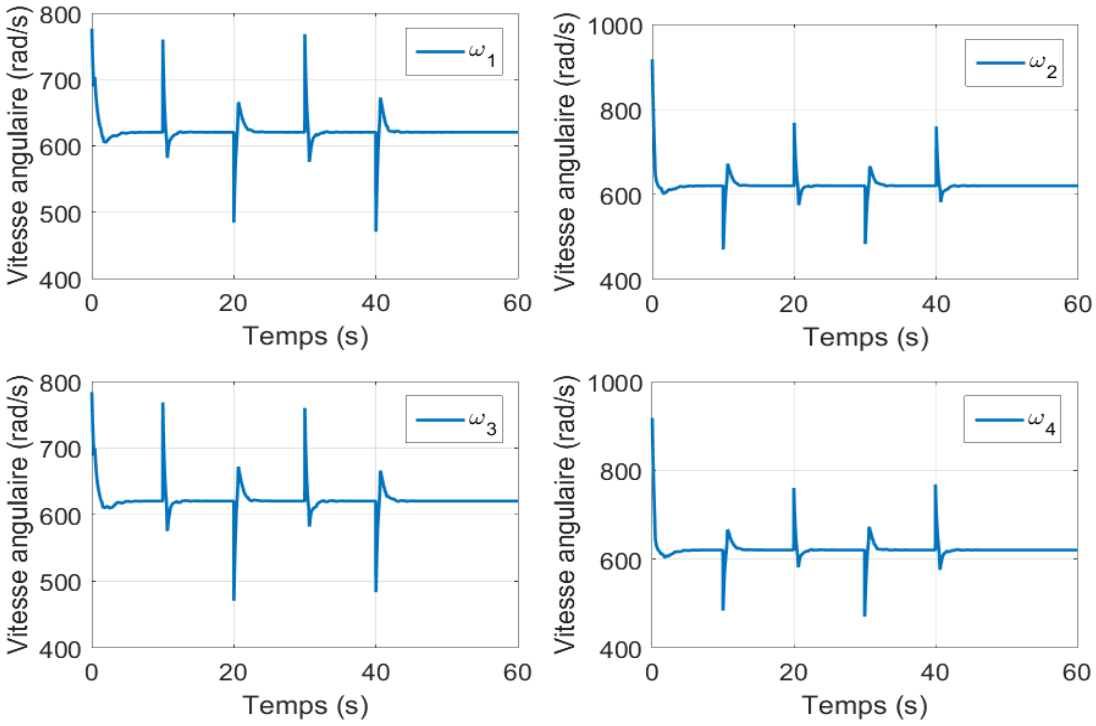


FIGURE 1.25: Évolution de la vitesse angulaire des quatres rotors du quadrirotor en fonction du temps

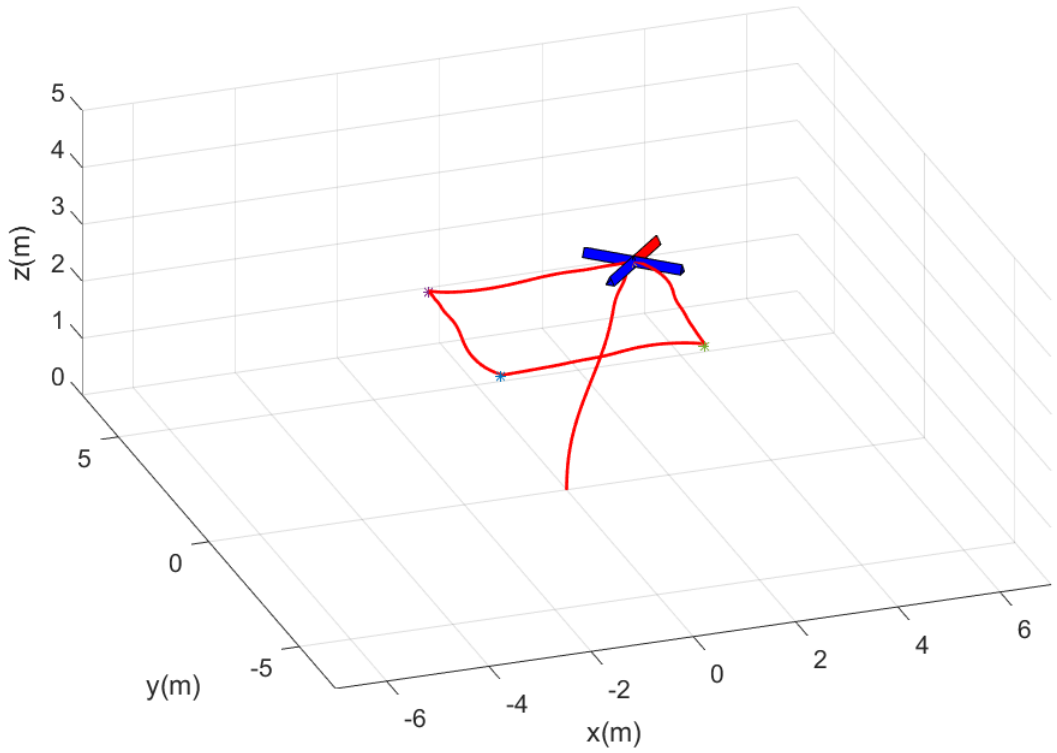


FIGURE 1.26: Évolution en 3D du quadrirotor en fonction du temps

1.9 Conclusion

Dans ce chapitre, nous avons définit brièvement ce qu'est un drone et ce qu'est un quadrirotor et nous avons fait une étude sur l'état de l'art de ce dernier et nous avons présenté quelques unes de ces applications. Puis, nous avons décrit les mouvements de base du quadrirotor et nous l'avons modélisé grâce au formalisme de Newton en faisant quelques hypothèses simplificatrices. Ce modèle a été ensuite utilisé pour synthétiser une commande par mode glissant dont nous avons étudier la stabilité et la robustesse tout en réduisant le phénomène de chattering qu'introduit la fonction $sign(.)$. Nous avons enfin simuler le système ainsi que la commande et nous avons vérifié nos hypothèses.

Chapitre 2

Navigation

2.1 Introduction

La cartographie et la localisation simultanée (SLAM) en temps réel et la reconstruction de scènes en 3D sont des sujets de recherche de plus en plus populaires. Deux des raisons majeures de cela sont leur utilisation dans la robotique, notamment pour la navigation des drones (UAV)[34, 35, 36], et leur utilisation dans les applications de réalité augmenté ou virtuelle qui touchent de plus en plus de domaines tels que les jeux vidéo, l'éducation par le jeu, le cinéma, etc.

Dans ce chapitre, nous allons commencer par définir ce qu'est le SLAM. Ensuite, nous présenterons succinctement sont historique. Puis, nous présenterons les différents capteurs utilisés pour chaque approche du SLAM. Après, nous nous intéresserons de plus près au SLAM Monoculaire et nous présenterons l'état de l'art sur le sujet, ainsi que quelques uns des algorithmes de fusion de données les plus utilisés. Enfin, nous expliquerons l'approche proposée, que nous implémenterons au chapitre suivant.

2.2 Definition

Simulataneous Localization And Mapping (SLAM) ou Localisation et Cartographie Simultanées est un processus par lequel un robot mobile peut construire une carte de son environnement (une carte composées d'amers, qui sont des points de repères supposés statiques) et en même temps utiliser cette carte pour déduire son emplacement[37]. Initialement, la carte et la position du robot sont inconnus. Tout deux sont estimés au fur et à mesure que le robot prend des mesures de son environnement[38].

On note :

x_t, le vecteur d'état décrivant la position et l'orientation du robot à l'instant t,

u_t, la commande appliquée au robot à l'instant *t-1*,

m_i, la position de l'amer i dans l'environnement, qui est supposée invariable dans le temps,

z_t^i, l'observation à l'instant t de l'amer i.

$X_{0:k} = \{x_0, x_1, \ldots, x_k\}$, l'historique du vecteur d'état du robot,

$U_{0:k} = \{u_0, u_1, \ldots, u_k\}$, l'historique des commandes appliquées au robot,

$M = \{m_0, m_1, \ldots, m_k\}$, l'ensemble des amers,

$Z_{0:k} = \{z_0, z_1, \ldots, z_k\}$, l'historique des observations.

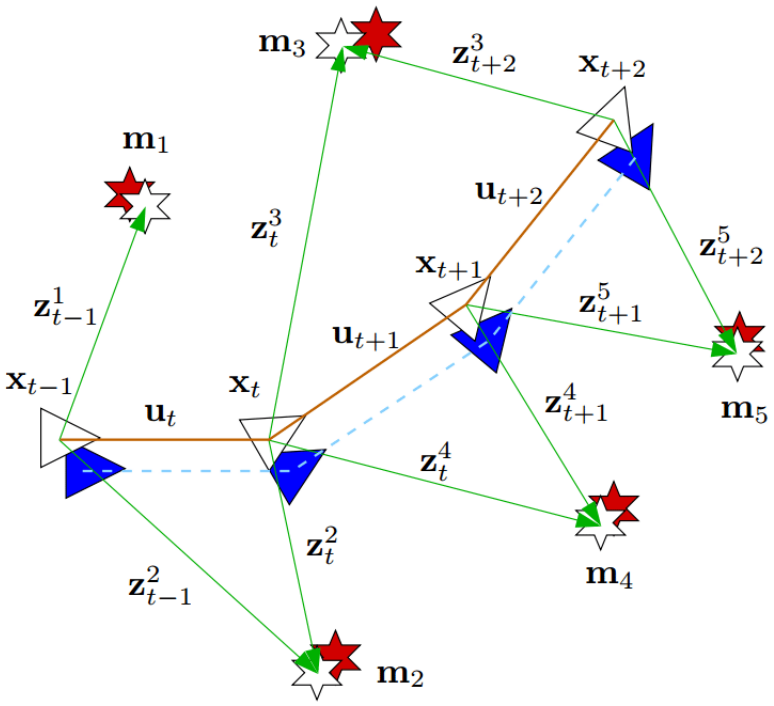


FIGURE 2.1: Notation du problème de SLAM

Le problème du SLAM est posé de manière probabiliste, l'objectif étant de déterminer à chaque instant t, la position du robots x_t et de l'ensemble de tous les amers observés M connaissant l'ensemble des observations $Z_{0:t}$ et l'ensemble des commandes envoyées au robot $U_{0:t}$ et l'état initial x_0 :

$$P(x_t, M|Z_{0:t}, U_{0:t}, x_0) \tag{2.1}$$

Plusieurs approches ont été explorées pour résoudre ce problème d'estimation. Elles se différencient principalement par le choix des hypothèses complémentaires imposées au problème ou les approximations choisies pour le rendre tractable.

2.3 Historique

La genèse du problème de SLAM probabiliste a eu lieu à la conférence IEEE Robotics and Automation en 1986, tenue à San Francisco. Ce fut un temps où les méthodes probabilistes n'étaient qu'au début d'introduction dans la robotique et l'intelligence artificielle. Un certain nombre de chercheurs avait travaillé sur l'application des méthodes de la théorie d'estimation sur les problèmes de cartographie et de localisation.

Le résultat de cette conférence était une reconnaissance du fait que la cartographie probabiliste cohérente était un problème fondamental dans la robotique avec les principaux problèmes conceptuels et informatique qui doivent être pris en compte. Au cours des années suivantes, un certain nombre de documents clés ont été produits. Les travaux [39] et [40] ont établit une base statistique pour décrire les relations entre les points de repère et pour manipuler l'incertitude géométrique. Un élément clé de ce travail était de montrer qu'il doit y avoir une forte corrélation entre les estimations des emplacements des différents points de repère dans une carte et qu'en effet, ces corrélations grandiraient avec les observations successives.

Au même moment, [41] ont entrepris les premiers travaux sur la navigation visuelle, et [42]et [43] les premiers travaux sur la navigation à base de sonar des robots mobiles à l'aide des filtres de Kalman. Ces deux axes de recherches ont beaucoup en commun et ont donné lieu à d'autres travaux. Parmi ces travaux, on note [44], où les auteurs ont montré que quand un robot mobile se déplace dans un environnement inconnu en prenant des observations relatives des points de repères, leurs estimations sont toutes nécessairement corrélés entre eux en raison de l'erreur commune dans l'estimation de l'emplacement du véhicule[45]. L'implication de cela était profonde. Une solution complète et consistante au problème combiné de la localisation et de la cartographie nécessite un état mixte composée de la pose du véhicule et des positions des points de repères, pour être mis à jour selon chaque observation d'un point de repère. A son tour, cela nécessite l'utilisation d'un vecteur d'état énorme pour l'estimateur (de l'ordre du nombre des points de repères maintenus sur la carte) avec un temps de calcul proportionnel au carré du nombre de points de repère.

Fondamentalement, ce travail n'a pas pris en considération les propriétés de convergence de la carte ou son comportement à l'état d'équilibre. En effet, il était largement admis à l'époque que les erreurs d'estimation de la carte ne convergent pas mais présentent un comportement de marche aléatoire avec une croissance d'erreur sans bornes. Ainsi, étant donné le temps de calcul du problème de cartographie et sans connaissance du comportement de convergence de la carte, les chercheurs se sont plutôt concentrés sur une série d'approximations à la solution du problème de cartographie qui assume ou même force les corrélations entre les points de repères pour être minimisées ou éliminés, de façon à réduire le filtre complet à une série de point de repère découplé aux filtres de véhicule. Ainsi, pour ces raisons, les travaux théoriques sur le problème combiné de la localisation et de la cartographie sont venus à un arrêt temporaire, avec parfois des travaux sur soit sur la cartographie ou la localisation comme problèmes distincts.

La percée conceptuelle est venue avec le fait que le problème combiné de la cartographie et la localisation, une fois formulé comme un problème d'estimation unique, était en réalité convergent. Plus important encore, il a été reconnu que les corrélations entre les points de repère, que la plupart des chercheurs ont tenté de minimiser, étaient en fait la partie critique du problème et que, au contraire, plus elles augmentent, mieux est la solution. La structure du problème de SLAM, le résultat de la convergence et l'acronyme "SLAM" sont présentés pour la première fois en 1995 dans le symposium international de la recherche en robotique. La théorie essentielle sur la convergence et la plupart des premiers résultats ont été développés dans [46]. Plusieurs groupes travaillant déjà sur la cartographie et la localisation, notamment au MIT [47], Zaragoza [48, 49], la ACFR à Sydney [50, 51] et d'autres [52, 53], ont commencé à travailler sérieusement sur les applications du SLAM dans les environnements intérieurs, extérieurs et sous-marins.

A cette époque, les travaux ont porté sur l'amélioration des performances des calculs et sur la résolution des problèmes d'association des données ou fermeture de boucle. Le symposium international sur la recherche en robotique de 1999 a été un point de rencontre important où la première session de SLAM s'est tenue et où un degré de convergence entre les méthodes de SLAM basés sur le filtre de Kalman et les méthodes probabilistes de localisation et de cartographie introduites en [51] a été achevé. Le IEEE ICRA Workshop de l'an 2000 a attiré une quinzaine de chercheurs qui se sont concentré sur des questions telles que la complexité algorithmique, l'association de données et les défis de mise en œuvre pratique.

L'intérêt pour le SLAM a connu et connaît toujours une croissance exponentielle, et des ateliers continue de se tenir à la fois à ICRA et à IROS.

2.4 Capteurs

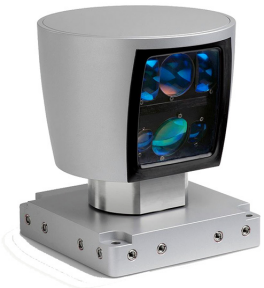
L'un des critères les plus importants pour un robot autonome est la capacité à percevoir son environnement. Les capteurs du robot traduisent les conditions de l'environnement en des signaux appropriés pour le traitement. Le choix du capteur à utiliser est crucial car il affecte la qualité et la quantité d'information mise à la disposition du robot et détermine ainsi quelle approche de SLAM est la plus appropriée pour être utilisée. Nous pouvons voir quelques exemples de capteurs utilisés pour le SLAM sur les figures 2.2, **??**.

(a)

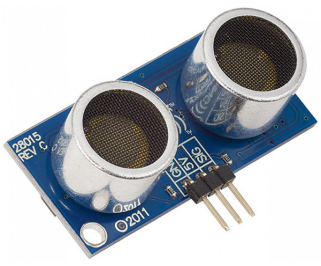
(b)

(c)

(d)



(e)



(f)

FIGURE 2.2: Exemples de capteurs utilisés pour le SLAM (a) caméra monoculaire, (b) caméra stéréo, (c) caméra omnidirectionnelle, (d) caméra RGB-D, (e) télémètre laser, (f) télémètre sonar

2.5 SLAM Monoculaire

Le principe de la localisation par vision monoculaire d'un véhicule est identique pour la grande majorité des méthodes. La première position de la caméra est inconnue et est

donc fixée arbitrairement. Par la suite, le déplacement en trois dimensions de la caméra dans l'environnement qui l'entoure va se traduire visuellement dans l'image par un déplacement en deux dimensions des éléments d'intérêt de l'image. Ces éléments d'intérêt sont généralement des zones ou plus couramment des points de l'image qu'il est possible de suivre au fil du flux vidéo, c'est à dire entre les différentes images. A partir de l'observation du déplacement de ces éléments d'intérêt, il est possible d'inférer le mouvement de la caméra.

2.5.1 Etat de l'art

Différentes questions sont impliqués dans le problème du SLAM visuel et de nombreuses approches différentes existent afin de les résoudre. Nous allons présenter quelque uns des travaux majeurs qui ont été réalisés.

MonoSLAM (2003)

MonoSLAM[54], est le premier algorithme de SLAM Monoculaire à être capable de fonctionner en temps-réel. Le concept clé de MonoSLAM est l'utilisant d'une carte probabiliste des caractéristiques de l'environnement, qui représente à un instant donné la position et l'orientation estimée de la caméra et de tout les points d'intérêts, qui sont regroupées dans un vecteur d'état, ainsi que l'incertitude de ces estimées, qui sont regroupées dans une matrice de covariance, figure 2.4. Le vecteur d'état et la matrice de covariance augmentent à chaque nouvelle observation et les estimées sont misent à jour grâce à un filtre de Kalman étendu (EKF). C'est là que réside son inconvénient majeure. Le temps de calcul est proportionnel au carré du nombre de points de d'intérêts, ce qui veut dire que l'algorithme devient rapidement inutilisable pour de grands environnements.

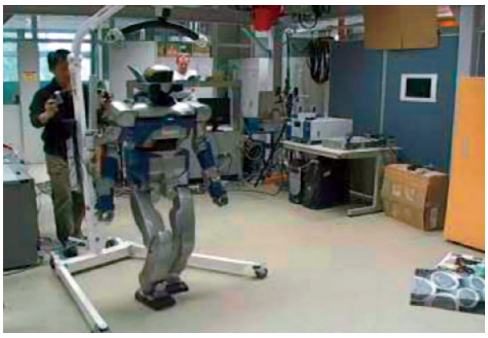


FIGURE 2.3: Robot HRP-2 qui marche d'une manière autonome en utilisant MonoSLAM

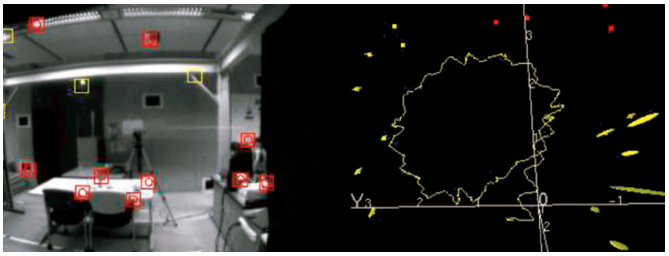


FIGURE 2.4: Extrait de MonoSLAM, en jaune la trajectoire estimée du robot HRP-2

Parallel Tracking and Mapping - PTAM (2007)

PTAM[55] est un algorithme d'estimation de la position et de l'orientation de la caméra dans un environnement inconnue. Bien que cela ai déjà été tentée par l'adaptation des algorithmes de SLAM développés pour l'exploration robotique. PTAM propose une solution simple, efficace, et l'entrée de bas niveau à ce problème de SLAM. Plutôt que de compter sur des descripteurs de couleur, des marqueurs placés à des endroits spécifiques dans l'environnement, et les modèles de CAO de l'environnement existant, PTAM utilise l'algorithme Fast[56] (Features from Accelerated Segment Test) qui est un algorithme de détection de caractéristiques qui sont rapides à détecter et relativement peu chéres à traiter.

PTAM exécute la détection de la position et de l'orientation de la caméra et la cartographie sur l'environnement en parallèle réduisant ainsi les coûts de calcul et lui permettant de fonctionner en temps-réel. Les points forts de PTAM sont aussi ses limites : C'est un système relativement simple, de sorte qu'il n'extrait aucun sens structurel de l'environnement reconstruit autre que les points détectés par l'algorithme Fast. Il est également limité par la nécessité d'effectuer l'initialisation stéréo d'une manière très spécifique qui nécessite l'intervention de l'utilisateur.

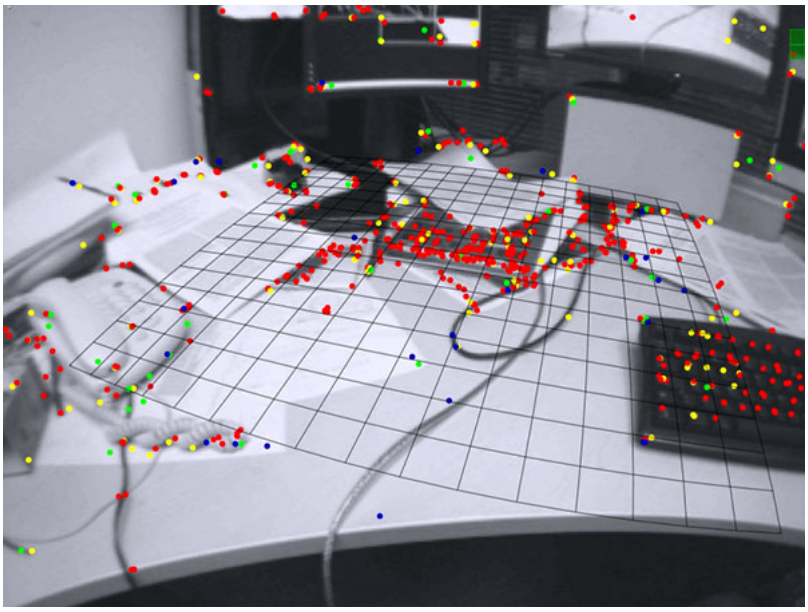


FIGURE 2.5: Exemple d'usage de PTAM

Cette algorithme peut être résumé en ces cinq points :
- Le suivi et la cartographie sont séparés et fonctionnement en parallèle dans deux fils d'exécution
- La cartographie est basée sur des images clés, qui sont traités par lots en utilisant l'ajustement de faisceaux (ou Bundle Adjustment[57])
- La map est initialisée en utilisant une paire d'images grâce à l'algorithme des 5 points (5-point Algorithm)
- De nouvelles amers sont initialisées à l'aide d'une fouille epipolaire
- Un grand nombre d'amers sont cartographiés

Dense Tracking And Mapping - DTAM (2011)

DTAM[58] est un algorithme qui permet le suivi de la position et de l'orientation de la caméra et la reconstruction de l'environnement en temps réel. Il ne repose pas sur l'extraction de caractéristiques à partir de l'image, mais sur des méthodes denses qui utilisent tout les pixels de l'image. Comme une seule caméra RGB main vole au-dessus d'une scène statique, nous estimons cartes de profondeur texturés détaillées à des images clés sélectionnés pour produire une mosaïque de surface avec des millions de sommets. Nous utilisons les centaines d'images disponibles dans un flux vidéo pour améliorer la qualité d'un terme de données photométriques simple, et de minimiser une énergie globale spatialement régularisée fonctionnelle dans un cadre novateur d'optimisation non-convexe. Entrelacé, nous suivons 6DOF le mouvement de la caméra précisément par frame-rate l'alignement d'image entier contre l'ensemble du modèle dense. Nos algorithmes sont très parallelisable partout et DTAM réalise des performances en temps réel en utilisant le courant matériel produit GPU

FIGURE 2.6: Exemple d'une scène reconstruite en 3D à l'aide de DTAM

Large-Scale Direct monocular SLAM - LSD-SLAM (2014)

LSD-SLAM[59] est une technique de SLAM monoculaire directe, c'est-à-dire, qu'au lieu d'utiliser des points d'intérêts dispersés dans l'environnement, il opère directement sur les intensités d'image à la fois pour le suivi et la cartographie. La caméra est suivie à l'aide d'alignement d'image directe, tandis que la géométrie est estimée sous la forme de cartes de profondeur semi-denses obtenue par filtration sur de nombreuses comparaisons faites pixel par pixel.

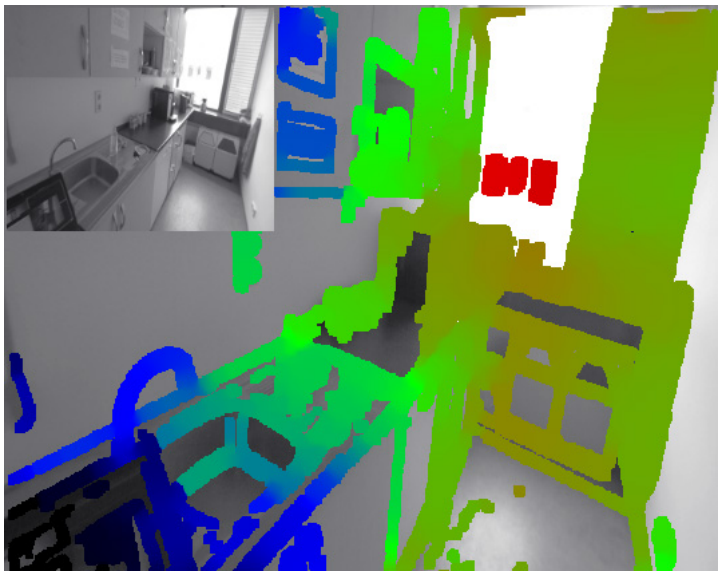


FIGURE 2.7: Exemple d'usage de LSD-SLAM

ORB-SLAM (2015)

ORB-SLAM[60] est un algorithme de SLAM visuel qui utilise le détécteur de zones d'intérêts ORB[61]. Il est capable de calculer en temps réel la trajectoire de la caméra et une reconstruction 3D clairsemée de la scène dans une grande variété d'environnements, allant de petites séquences tenues à la main d'un bureau à une voiture conduite autour de plusieurs pâtés de maisons.

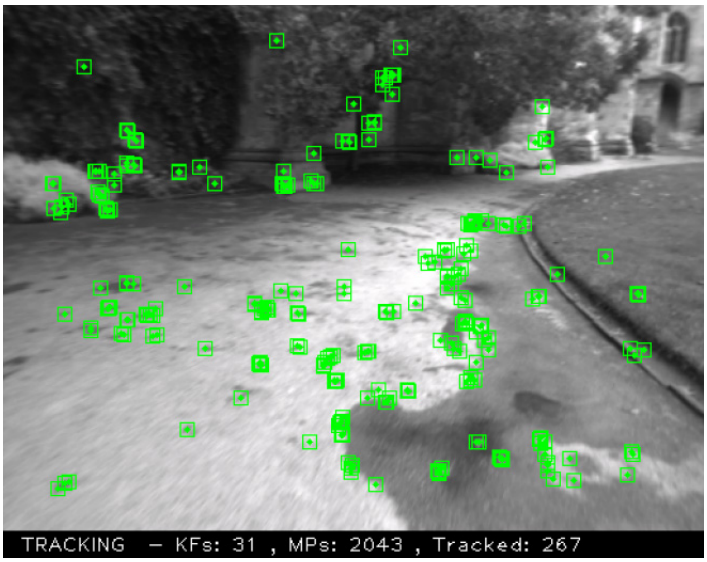


FIGURE 2.8: Exemple d'usage d'ORB-SLAM

Comme on peut le voir sur la figure 2.9, il est constitué de trois modules qui s'exécutent en parallèle :

Suivi Ce fil d'exécution localise la position et l'orientation de la caméra pour chaque nouvelle image et décide de l'ajout ou non de cette dernière dans une base d'images clés.

Cartographie locale Ce fil d'exécution traitent les images clés et exécute un ajustement de faisceaux local pour obtenir une reconstruction de l'environnement optimale.

Fermeture de boucles Ce fil d'exécution tente de trouver des boucles fermées à chaque nouvelle image clé.

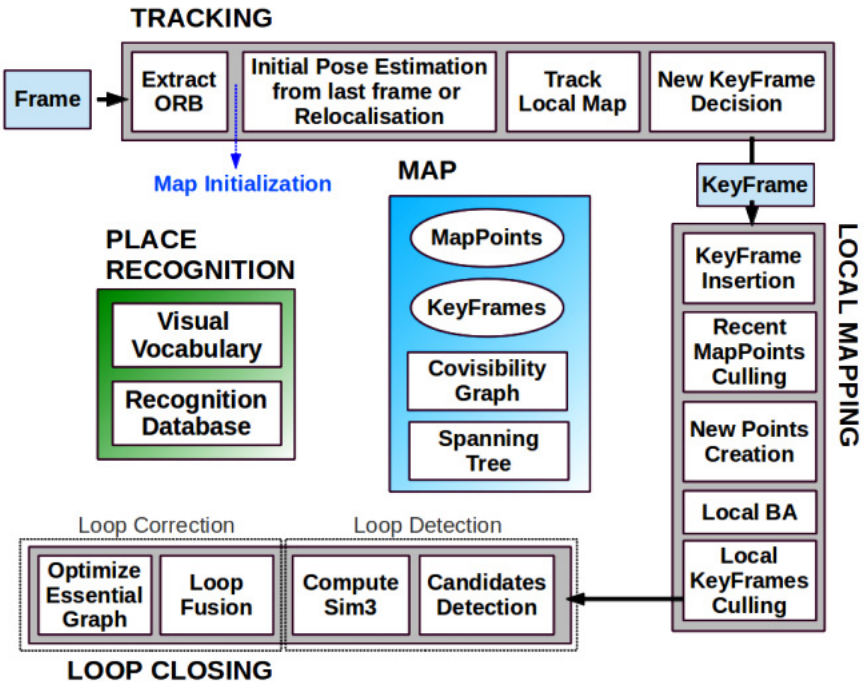


FIGURE 2.9: Schéma des différents modules d'ORB-SLAM

2.5.2 Inconvénients

Parmi les inconvénients du SLAM monoculaire se trouve le manque de robustesse face aux mouvements brusques et l'ambiguïté de l'échelle. C'est pour cela que dés les années 90, des chercheurs ont proposés de combiner les données des caméras avec celles des centrales inertielles et d'autres capteurs[62] pour exploiter leurs caractéristiques complémentaires[63] :

- Les centrales inertielles sont incapables de distinguer entre un changement d'inclinaison et l'accélération que subit le corps auxquels elles sont attachées, à cause du principe d'équivalence d'Einstein[64].
- Les centrales inertielles ont une grande incertitude pour des mouvements à faible vitesse et une moindre incertitude pour des mouvements à grande vitesse.
- Les caméras peuvent suivre des points d'intérêts avec une très grande précision pour des mouvements à faible vitesse. Cette précision diminue avec l'augmentation de la vitesse.
- Dans une image projective, celle d'une caméra, on ne peut différencier entre un mouvement de translation et un mouvement de rotation[63].
- Pour une caméra, un objet proche avec une vitesse relative faible est semblable à un objet lointain avec une grande vitesse relative[63].

2.6 Algorithmes d'estimation

Les algorithmes de fusion de données et d'estimation peuvent être réparties en deux catégories :

2.6.1 Estimation par lots

Les méthodes d'estimation par lots, ou Batch Estimation en Anglais, sont des techniques qui consistent à optimiser un ensemble de paramètres, trajectoires et positions de points d'intérêts en utilisant toutes les données de la caméra et de la centrale inertielle, afin de minimiser l'erreur de reprojection. L'algorithme classiquement utilisé est la méthode des moindres carrés non-linéaires de Levenberg-Marquardt[65].

Les reconstructions par estimation par lots sont généralement plus précises que les méthodes récursives, mais elles souffrent d'une complexité de calcul plus importante, qui est proportionnel au cube du nombre de paramètres à optimiser. On peut dès lors distinguer les méthodes hors-ligne (très lente) qui utilisent un maximum de paramètres pour la reconstruction 3D et l'estimation de la trajectoire et les méthodes en-ligne, qui réduisent le nombre de paramètres à optimiser pour traiter le flux de données en temps réel.

Les méthodes hors-ligne sont utilisés en fin de trajectoire, après avoir fini l'acquisition de toutes les données. Elles sont généralement utilisées pour la reconstruction de modèles 3D texturés de l'environnement.

Les méthodes en-ligne, contrairement aux méthodes hors-ligne où l'ensemble des images et des données inertielles est connu à l'avance, traitent le flux des données au fur et à mesure qu'elles arrivent. L'idée est d'utiliser une approche incrémentale reconstruisant petit à petit l'environnement et la trajectoire du système. Cette approche est le plus souvent utilisée pour résoudre le problème du SLAM, on peut citer les travaux effectués en [66, 67, 68, 69].

2.6.2 Estimation récursive

Les méthodes d'estimation récursive sont des techniques qui consistent à optimiser la trajectoire et la position des points d'intérêts en n'utilisant à chaque fois que l'état précédant et les nouvelles données visuelles et inertielles.

EKF

Le filtre de Kalman est un filtre à réponse impulsionnelle infinie qui estime les états d'un système dynamique à partir d'une série de mesures incomplètes ou bruitées. Le filtre a été nommé d'après le mathematicien et informaticien américain d'origine hongroise Rudolf Kalman.

Kalman propose une solution récursive au filtrage des données linéaires. Cette méthode, améliorée ensuite par Kalman lui-même et Bucy[70], ouvre de nouvelles pistes de recherche dans le domaine de la navigation autonome des robots mobiles. L'approche de base du filtre est basée sur un cycle récursif nécessitant trois hypothèses pour assurer un fonctionnement, prouvé théoriquement, optimal :

- Un modèle d'évolution linaire du système
- Une relation linaire entre l'état et les mesures
- Un bruit blanc gaussien

Pour un système de la forme :

$$\begin{cases} x_k = f(x_{k-1}, u_k) + w_k \\ z_k = h(x_k) + v_k \end{cases} \tag{2.2}$$

Où :
x, est la vecteur d'état,
u, est la commande,
z, est la mesure,
w et v, sont des bruits gaussiens à moyennes nulles et de variances σ_w et σ_v respectivement.

l'estimation se fait suivant deux étapes distinctes :

Prédiction

$$\begin{cases} \hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k) \\ P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \end{cases} \tag{2.3}$$

Où :

$$F_k = \frac{\partial f}{\partial x}|_{\hat{x}_{k-1|k-1}, u_k} \tag{2.4}$$

Correction

$$\begin{cases} \hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - h(\hat{x}_{k|k-1})) \\ P_{k|k} = (I - K_k H_k) P_{k|k-1} \end{cases} \tag{2.5}$$

Où :

$$H_k = \frac{\partial h}{\partial x}|_{\hat{x}_{k|k-1}} \tag{2.6}$$

Et :

$$\begin{cases} S_k = H_k P_{k|k-1} H_k^T + R_k \\ K_k = P_{k|k-1} H_k^T S_k^{-1} \end{cases} \tag{2.7}$$

Filtre Particulaire

Le filtre particulaire, aussi connu sous le nom de méthode de Monté-Carlo séquetielle, sont des techniques sophistiquées d'estimation de modèles fondées sur la simulation.

Les filtres particulaires sont généralement utilisés pour estimer des réseaux bayésiens et constituent des méthodes 'en-ligne' analogues aux méthodes de Monte-Carlo par chaînes de Markov qui elles sont des méthodes 'hors-ligne' (donc à posteriori) et souvent similaires aux méthodes d'échantillonnage séquentiel par importance (SIS).

S'ils sont conçus correctement, les filtres particulaires peuvent être plus rapides que les méthodes de Monte-Carlo par chaînes de Markov. Ils constituent souvent une alternative aux filtres de Kalman étendus avec l'avantage qu'avec suffisamment d'échantillons, ils approchent l'estimé Bayésien optimal. Ils peuvent donc être rendus plus précis que les filtres de Kalman. Les approches peuvent aussi être combinées en utilisant un filtre de Kalman comme une proposition de distribution pour le filtre particulaire.

2.7 Approche proposée

Pour l'approche proposée, nous nous sommes principalement inspirés des travaux effectués en [71, 72]. Nous avons choisi d'utiliser un système faiblement couplé comprenant une caméra monoculaire, une centrale inertielle et un sonar. Les principales différences entre l'approche proposée et celle faite en [71, 72] sont dans l'algorithme de SLAM (ORB-SLAM au lieu de PTAM) et dans l'observateur utilisé (SASMO au lieu du EKF).

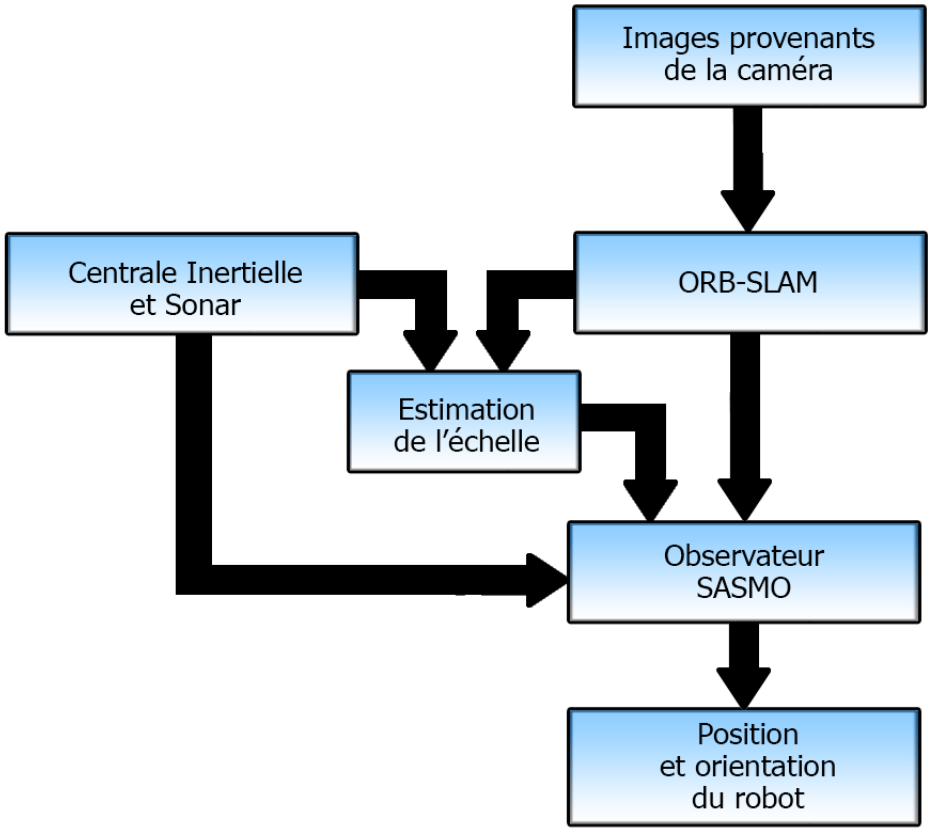


FIGURE 2.10: Schéma décrivant l'approche proposée

Nous utilisons les capteurs suivants :

Caméra monoculaire Elle est fixée à l'avant du quadrirotor pour voir la scène, sans être gênée par les hélices du quadrirotor. Les images reçus seront traitées par ORB-SLAM.

Son modèle d'observation est le suivant :

$$h_{ORB-SLAM} = \begin{bmatrix} \lambda x & \lambda y & \lambda z & \phi & \theta & \psi \end{bmatrix}^T \tag{2.8}$$

Avec :

λ, l'échelle des estimations d'ORB-SLAM.

Centrale inertielle Elle est fixée au centre de gravité du quadrirotor pour qu'elle puisse mesurer les accélérations linéaires et les vitesses angulaires sans erreurs. Nous l'utilisons pour effectuer deux types de mesures.

Le premier type de mesures consiste à intégrer les accélérations linéaires pour obtenir les vitesses de translation et à utiliser directement les mesures des vitesses angulaires.

Le modèle d'observation est le suivant :

$$h_{IMU-1} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T \tag{2.9}$$

Le deuxième type de mesures consiste à utiliser le fait que lorsque le quadrirotor est au repos, l'accéléromètre mesure seulement l'accélération due à la gravité et comme on peut le voir sur la figure 2.11 nous pouvons l'utiliser pour mesurer les angles de roulis et tangage comme décrit dans l'équation(2.10).

$$\begin{aligned} \phi &= atan(\tfrac{a_y}{a_z}) \\ \theta &= atan(-cos(\phi) \cdot \tfrac{a_x}{a_z}) \end{aligned} \tag{2.10}$$

Avec :

a_x, a_y et a_z les mesures filtrées de l'accéléromètre suivant les axes X, Y et Z, respectivement.

Cependant, les mesures des accélérations n'étant pas parfaites, à cause du bruit et des fortes vibrations engendrés par la rotation des quatre rotors, nous ne pouvons les utiliser directement. C'est pour cela que nous utilisons un filtre passe-bas de Butterworth[73] pour les filtrer et ne laisser que les composantes statiques qui représentent l'accélération due à la gravité.

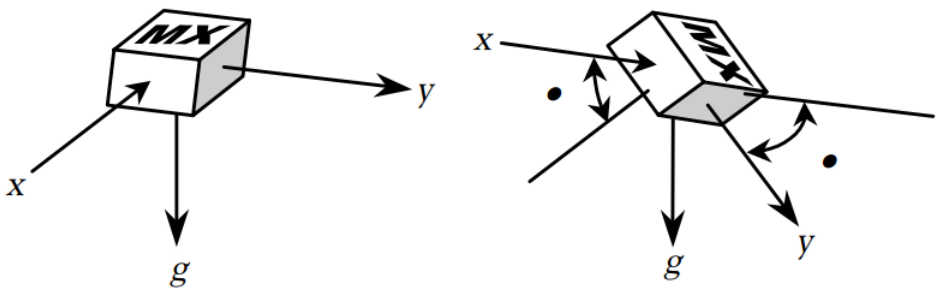


FIGURE 2.11: Mesure des angles en utilisant la gravité

Le modèle d'observation est le suivant :

$$h_{IMU-2} = \begin{bmatrix} \phi & \theta \end{bmatrix}^T \tag{2.11}$$

Sonar ultrason Il est fixé sur la châssis du quadrirotor et est dirigé vers le sol pour qu'il puisse mesurer la hauteur à laquelle se trouve le drone.

Le modèle d'observation est le suivant :

$$h_{Sonar} = z \tag{2.12}$$

L'approche proposée peut être divisée en trois parties, figure 2.10 :
- ORB-SLAM
- SASMO (Stochastic Adaptative Sliding Mode Observer)
- Estimateur de l'échelle

2.7.1 ORB-SLAM

Nous utilisons l'algorithme de SLAM Monoculaire open-source ORB-SLAM pour estimer la position et l'orientation du quadrirotor, ainsi que le position des points d'intérêts dans l'environnement. Cependant, les estimations données par cette algorithme ne sont pas métrique, c'est-à-dire, que l'échelle de ces estimations n'est identique à l'échelle réelle. Pour résoudre ce problème, nous combinons ces données avec celles de la centrale inertielle et du sonar pour estimer l'échelle et pouvoir les fusionner à l'aide de l'observateur.

2.7.2 SASMO (Stochastic Adaptative Sliding Mode Observer)

Nous utilisons un observateur glissant stochastique et adaptatif pour pouvoir observer l'état du quadrirotor et pour fusionner les différentes mesures des différents capteurs.

Nous allons utiliser l'observateur décrit en [74].

Soit un système non-linéaire non-autonome décrit l'équation différentielle d'Itô[75] :

$$\begin{cases} dx_t = (Ax_t + h(x_t)u_t)dt + f(x_t, \zeta_t)dt + g_1(x_t, t)dv_t \\ dy_t = Cdx_t + g_2(x_t, t)dw_t \end{cases} \tag{2.13}$$

Où :

$x_t \in \mathcal{R}^n$, est le vecteur d'état,

$u_t \in \mathcal{R}^p$, est la vecteur de commande,

$\zeta_t \in \mathcal{R}^n$, est le vecteur des perturbations déterministes non mesurables,

$A \in \mathcal{R}^{nn}$, est la matrice de transition qui représente la partie linéaire du système,

$C \in \mathcal{R}^{nn}$, est la matrice de d'observation,

$h : \mathcal{R}^n \to \mathcal{R}^{np}$, est une fonction qui de commande,

$f : \mathcal{R}^n\mathcal{R}^n \to \mathcal{R}^n$, est une fonction qui représente les non-linéarités et les incertitudes du systèmes,

$g_1 : \mathcal{R}^n\mathcal{R}^+ \to \mathcal{R}^{nv}$, est une fonction qui représente l'intensité du bruit du processus,

$g_2 : \mathcal{R}^n\mathcal{R}^+ \to \mathcal{R}^{nw}$, est une fonction qui représente l'intensité du bruit de mesure,

$dv_t \in \mathcal{R}^v$, est un processus stochastique à moyenne nulle et variance dt qui représente l'incrément d'un processus de Weiner $v_t \in \mathcal{R}^v$[76],

$dw_t \in \mathcal{R}^w$, est un processus stochastique à moyenne nulle et variance dt qui représente l'incrément d'un processus de Weiner $w_t \in \mathcal{R}^w$[76].

On admettant les hypothèses suivantes :
- La paire (A, C) est observable. Ce qui implique l'existence d'un gain $K \in \mathcal{R}^{nn}$ tel que les valeurs propres de la matrice $A_0 = A - KC$ soient à valeur réelle négative.

– La fonction f est séparable, i.e. $f = f_1 + f_2$.
– La fonction f_1 est Lipschitzienne :

$$||f_1(x_1) - f_1(x_2)|| \leq l||x_1 - x_2|| \quad \forall x_1, x_2 \in \mathcal{R}^n \tag{2.14}$$

où $l \in \mathcal{R}^+$ est une constante qui doit être déterminée.
– f_2 est une fonction incertaine, non mesurable, déterministe et bornée qui peut être modélisée comme suit :

$$f_2 = C^T \zeta(t, x_t, u_t) \leq \bar{\zeta} \tag{2.15}$$

On peut démontrer que l'observateur suivant est stable en probabilité[77] :

$$d\hat{x}_t = (A\hat{x}_t + h(\hat{x}_t)u_t + f_1(\hat{x}_t))dt + K(Ce_t)dt + S(\hat{x}_t, y_t, \phi_t, \hat{\eta}_t)dt \tag{2.16}$$

Et qu'il reconstruit l'état du système à partir des mesures en utilisant le gain adaptatif glissant donné par :

$$S = P^{-1}C^T(\sum_{i=1}^{N} \hat{\eta}_i \Gamma_i(e))\frac{\gamma Ce_t}{||\gamma Ce_t| - \phi_t|} \tag{2.17}$$

L'algorithme d'adaptation est basé sur l'espérance de l'erreur d'estimation :

$$d\hat{\eta}_i = \gamma_0 E||Ce_i||^i dt \tag{2.18}$$

Cette observateur nous donne une estimation telle que l'erreur quadratique moyenne soit exponentiellement bornée.

$$\begin{cases} \lim\limits_{t \to +\infty} Sup||e_t||^2 \leq \lambda_{min}^{-1}(P) B_1(\frac{\lambda_{min}(Q)}{\lambda_{max}(P)} - l) \\ si \qquad\qquad l\lambda_{max}(P) < \lambda_{min}(Q) \end{cases} \tag{2.19}$$

Où :

$e_t \in \mathcal{R}^n$, est l'erreur d'observation définie comme $e_t = x_t - \hat{x}_t$,

$\Gamma_i(e) : \mathcal{R}^+ \to \mathcal{R}^+$, est définie comme $\Gamma_i(e) = \frac{E||Ce||^i}{||Ce||}$,

$\phi_t : \mathcal{R}^+ \to \mathcal{R}^+$, est définie comme $\phi_t = \gamma exp(-\beta t)$,

$B_1 \in \mathcal{R}+$, est définie comme $B_1 = \beta_1^2 \lambda_{max}(P) + \beta_2 \lambda_{max}(K^T P K)$,

$\gamma, \gamma_0, \beta, \beta_1, \beta_2 \in \mathcal{R}^+$, sont des constantes de réglage,

$P = P^T, Q = Q^T \in \mathcal{R}^{nn}$, sont des matrices définies positives telles que :

$$(A - KC)^T P + P(A - KC) = -2Q \tag{2.20}$$

2.7.3 Estimateur de l'échelle

L'un des inconvénients majeurs des algorithmes de SLAM visuel monoculaire est que l'échelle, $\lambda \in \mathcal{R}$, de la carte obtenue ne peut être déterminée sans utiliser d'autres capteurs ou d'informations sur des objets présents dans la scène. Pour pouvoir utiliser un algorithme

de SLAM visuel monoculaire dans la navigation il est nécessaire d'estimer la valeur de cette échelle.

Pour cela nous utilisons un algorithme basé sur la maximum de vraisemblance comme dans [72].

D'un côté, nous utilisons les mesures d'altitude données par le sonar que nous dérivons pour obtenir la vitesse d'ascension/descente , $\dot{z}^i_{sonar}$, et les mesures de l'accéléromètre que nous intégrons pour obtenir les vitesses horizontales, $\dot{x}^i_{accel}$ et $\dot{y}^i_{accel}$. Nous regroupons ces mesures dans le vecteur X^i, avec i qui représente le numéro de la mesure. De l'autre côté, nous utilisons les estimations de position données par ORB SLAM que nous dérivons pour obtenir les vitesses de translation, $\dot{x}^i_{orb}$, $\dot{y}^i_{orb}$ et $\dot{z}^i_{orb}$. Nous regroupons ces mesures dans le vecteur Y^i

Nous supposons que les deux mesures sont indépendantes et distribuées selon une loi normale :

$$\begin{aligned} X^i &\sim \mathcal{N}(\mu_i, \sigma_X^2) \\ Y^i &\sim \mathcal{N}(\lambda\mu_i, \sigma_{orb}^2) \end{aligned} \tag{2.21}$$

Pour estimer l'échelle λ, nous utilisons la méthode du maximum de vraisemblance. Le logarithme de la vraisemblance des paramètres μ_i et λ et donnée par :

$$ln\mathcal{L}(\mu_1, \mu_2, \ldots, \mu_N, \lambda) \propto \frac{1}{2}\sum_{i=1}^{N}\left(\frac{||X^i - \mu_i||^2}{\sigma_X^2} + \frac{||Y^i - \lambda\mu_i||^2}{\sigma_{orb}^2}\right) \tag{2.22}$$

la valeur optimale de l'échelle, $\lambda*$, et des valeurs moyennes de l'altitude, μ_i*, est obtenue lorsque la vraisemblance atteint son maximum, i.e. :

$$\begin{cases} \frac{\partial ln\mathcal{L}(\mu_1,\mu_2,\ldots,\mu_N,\lambda)}{\partial\mu_1} = 0 \\ \frac{\partial ln\mathcal{L}(\mu_1,\mu_2,\ldots,\mu_N,\lambda)}{\partial\mu_2} = 0 \\ \vdots \\ \frac{\partial ln\mathcal{L}(\mu_1,\mu_2,\ldots,\mu_N,\lambda)}{\partial\mu_N} = 0 \\ \frac{\partial ln\mathcal{L}(\mu_1,\mu_2,\ldots,\mu_N,\lambda)}{\partial\lambda} = 0 \end{cases} \tag{2.23}$$

En prenant la dérivée par rapport à μ_i, on peut calculer la valeur optimale de cette dernière en fonction de λ :

$$\begin{aligned} \frac{\partial ln\mathcal{L}(\mu_1,\mu_2,\ldots,\mu_N,\lambda)}{\partial\mu_1} \propto \frac{\mu_i - X^i}{\sigma_X^2} + \frac{\lambda^2\mu_i - \lambda Y^i}{\sigma_{orb}^2} = 0 \\ \implies \mu_i = \frac{\lambda\sigma_X^2 Y^i + \sigma_{orb}^2 X^i}{\lambda^2\sigma_X^2 + \sigma_{orb}^2} \end{aligned} \tag{2.24}$$

En prenant cette fois-ci la dérivée par rapport à λ et en remplaçant les μ_i par leurs expressions nous obtenons :

$$\frac{\partial ln\mathcal{L}(\mu_1, \mu_2, \ldots, \mu_N, \lambda)}{\partial\lambda} \propto \sum_{i=1}^{n}\frac{(\lambda X^i - Y^i)^T(\sigma_{orb}^2 X^i + \lambda\sigma_X^2 Y^i)}{(\lambda^2\sigma_X^2 + \sigma_{orb}^2)^2} = 0 \tag{2.25}$$

Ce qui nous donne, après la résolution de l'équation, la valeur optimale de l'échelle, λ^* :

$$\lambda^* = \frac{s_{xx} - s_{yy} + \sqrt{(s_{xx} - s_{yy}^2 + 4s_{xy}^2)}}{2\sigma_{orb}^{-1}\sigma_X s_{xy}} \tag{2.26}$$

Avec :
$s_{xx} := \sigma_X^2 \sum_{i=1}^{N} (Y^i)^2$
$s_{yy} := \sigma_{orb}^2 \sum_{i=1}^{N} (X^i)^2$
$s_{xy} = \sigma_X \sigma_{orb} \sum_{i=1}^{N} X^i \cdot Y^i$

2.8 Conclusion

Dans ce chapitre, nous avons définit ce qu'est le SLAM. Puis, nous avons présenté son historique. Ensuite, nous avons présenté les différents capteurs utilisés pour les différentes approches du SLAM. Après, nous nous sommes intéressés de plus près au SLAM visuel monoculaire. Nous avons présentés l'état de l'art sur le sujet, les inconvénients qu'apporte l'usage de la caméra comme seul capteur. Puis, nous avons présenter les algorithmes de fusion de données visuelles-inertielles. Enfin, nous avons expliqué l'approche de SLAM visuel monoculaire que nous avons proposé et que nous allons implémenter au chapitre suivant.

Chapitre 3

Commande et Navigation sous ROS

3.1 Introduction

L'écriture de logiciels pour les robots est difficile, d'autant plus que l'ampleur et la portée de la robotique continue de croître. Différents types de robots peuvent avoir un matériel différent, ce qui rend réutilisation du code non trivial. En plus de cela, la taille du code nécessaire peut être intimidant, car il doit contenir une pile profonde à partir de logiciels de niveau pilote et continue à travers la perception, le raisonnement abstrait et au-delà. Puisque l'expertise requise est bien au-delà des capacités de tout chercheur individuel, les logiciels robotiques doivent également soutenir les efforts d'intégration à grande échelle.

Pour relever ces défis, de nombreux chercheurs ont déjà créé une grande variété de frameworks et de logiciels pour gérer la complexité et faciliter le prototypage rapide de logiciels pour des expériences. De cet effort résulte les nombreux systèmes logiciels robotiques utilisés actuellement dans le milieu de la recherche et de l'industrie[78].

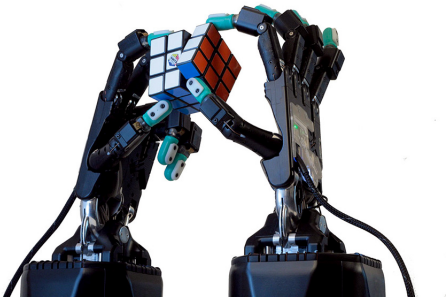
Nous entamons ce chapitre par une introduction sur ROS, sur ces concepts fondamentaux et sur ses différents outils. Nous enchainerons ensuite sur l'implémentation de la commande synthétisée au premier chapitre sous ROS. Puis, nous nous intéresserons à l'implémentation l'approche de SLAM proposée au deuxième chapitre sous ROS et Gazebo. Nous finirons enfin par une comparaison entre notre approche et une autre avec l'EKF.

3.2 ROS

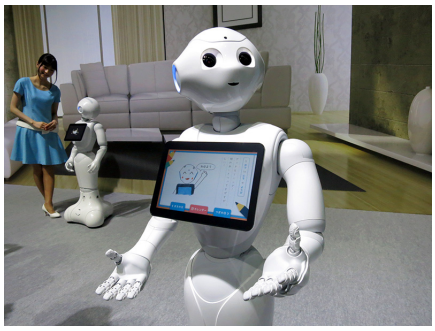
ROS[79] (Robot Operating System) est un meta-système d'exploitation, quelque chose entre le système d'exploitation et le middleware, pour robots né d'une collaboration entre le milieu industriel et celui de la recherche. Il fournit des services proches d'un système d'exploitation (abstraction du matériel, gestion de la concurrence, des processus...) mais aussi des fonctionnalités de haut niveau (appels asynchrones, appels synchrones, base centralisée de données, système de paramétrage du robot...). Il est composé de bibliothèques réutilisables qui sont conçus pour fonctionner de façon indépendante. Les bibliothèques sont enveloppés d'une couche de passage de messages mince qui leur permet d'être utilisés par et d'utiliser d'autres nœuds de ROS. Les messages sont transmis entre pairs et ne sont pas basées sur un langage de programmation spécifique. Ils peuvent être écrits en C++, Python, C, LISP, Octave ou tout autre langue pour lequel un wrapper ROS est disponible.

ROS peut être séparé en trois groupes :
- les outils utilisés pour créer, lancer et distribuer des logiciels basés sur ROS (roscore, roslaunch, catkin)
- les clients ROS pour des langages : roscpp (C++) et rospy (python)
- les packages contenant des programmes pour ROS utilisant un ou plusieurs clients ROS

Les outils et les principaux clients ROS (roscpp et rospy) sont publiés sous les termes de la licence BSD[80]. De nombreux packages sont publiés pour ROS avec diverses licences open source (BSD[80], MIT[81]). Ces packages permettent de lancer des applications, des algorithmes ou encore des programmes pour interfacer ROS avec des robots. L'outil de simulation Gazebo[82] est directement intégré à ROS.



(a) Shadow Hand, main robotique humanoide de la société The Shadow Robot Company



(b) Pepper, robot humanoïde développé par la société Aldebaran

(c) AscTec Hummingbird, quadrirotor de la société Ascending Technologies

FIGURE 3.1: Exemples de robots utilisant ROS

3.2.1 Concepts fondamentaux

Les concepts fondamentaux de ROS sont : le master, les nœuds, les topics, les messages, les services et le serveur de paramètres.

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, figure 3.2.

Le Master comprend une sous-partie très utilisée qui est le serveur de paramètres. Celui-ci, 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.

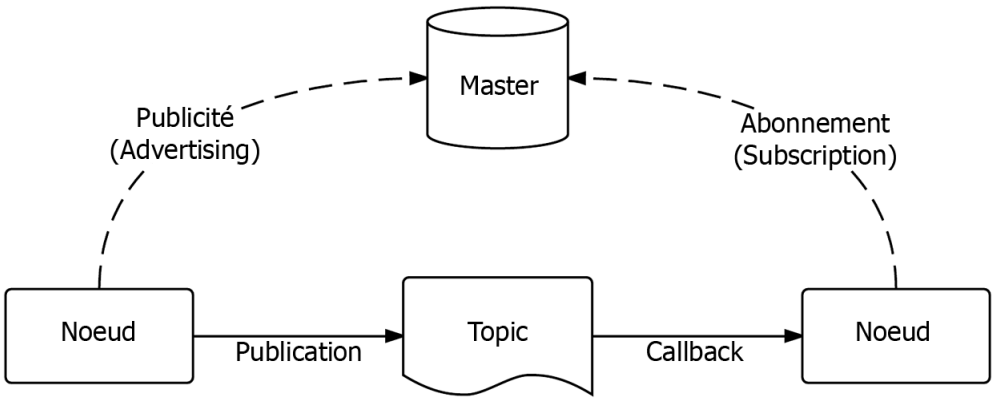


FIGURE 3.2: Schéma de fonctionnement du master

Nœuds

Dans ROS, un nœud est une instance d'un exécutable. Un nœud peut correspondre à un capteur, un moteur, un algorithme de traitement, de surveillance, etc. Chaque nœud qui se lance se déclare au Master. On retrouve ici l'architecture microkernel où chaque ressource est un nœud indépendant.

Topics

L'échange d'information s'effectue soit de manière asynchrone via un topic ou de manière synchrone via un service.

Un topic est un système de transport d'information basé sur le principe de l'abonnement/publication (subscribe / publish). Un ou plusieurs nœuds pourront publier de l'information sur un topic, sous forme de message, et un ou plusieurs nœuds pourront lire cette 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 est toujours structuré de la même manière.

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, etc) et de messages (le message est une structure récursive). Par exemple un nœud représentant un servomoteur d'un 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 ...

Services

Le topic est un mode de communication asynchrone permettant une communication many-to-many. Le service en revanche répond à une autre nécessité, celle d'une communication synchrone entre deux nœuds. Un nœud fait appel à un service par exemple pour lire la valeur d'un capteur à l'instant où il en a besoin, contrairement aux messages qui arrivent à une fréquence bien définie.

Serveur de paramètres

Le serveur de paramètres permet de définir et de gérer les paramètres globaux du robot. Les paramètres peuvent être des entiers, des flottants, des booléens, des strings, des listes et dictionnaires. Le serveur de paramètres est une mémoire partagée. Chaque nœud peut lire et modifier ces paramètres

3.2.2 Outils utiles dans ROS

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. Parmi les plus utilisés on trouve :

Gazebo

Gazebo[82] est un simulateur physique multi-robots pour les environnements en 3D. Il est capable de simuler une population de robots, de capteurs et d'objets de manière précise et efficace dans des environnements complexes d'intérieur et d'extérieur. Il génère des signaux réalistes et des interactions physiques plausibles entre les différents objets.

Parmi ces caractéristiques clés on trouve :

- La présence de plusieurs moteurs physiques
- Un large choix de modèles de robots et d'environnement, avec la possibilité de créer ses propres modèles.
- Un large éventail de capteurs
- La présence d'interfaces de programmatiques et graphiques pratiques

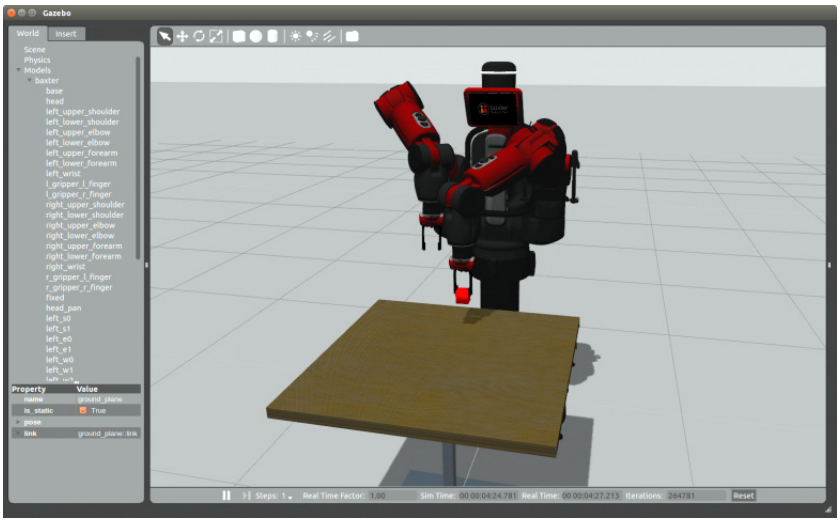


FIGURE 3.3: Exemple de simulation du robot Baxter[83] sous Gazebo

Rviz

Rviz[84] est un système de visualisation 3D (contrairement à Gazebo, il n'inclut pas de moteur physique).

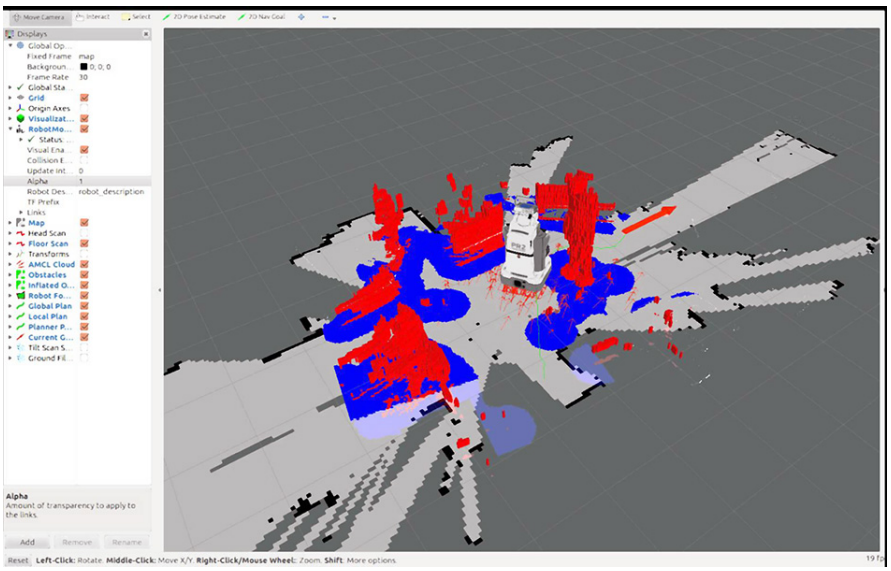


FIGURE 3.4: Exemple de visualisation de la simulation du robot PR2[85] simulé sous Rviz

RQT-Graph

RQT-Graph est un outil graphique qui permet de visualiser les différents noeuds et topics en cours d'exécution, ainsi que les différentes liaisons entre eux.

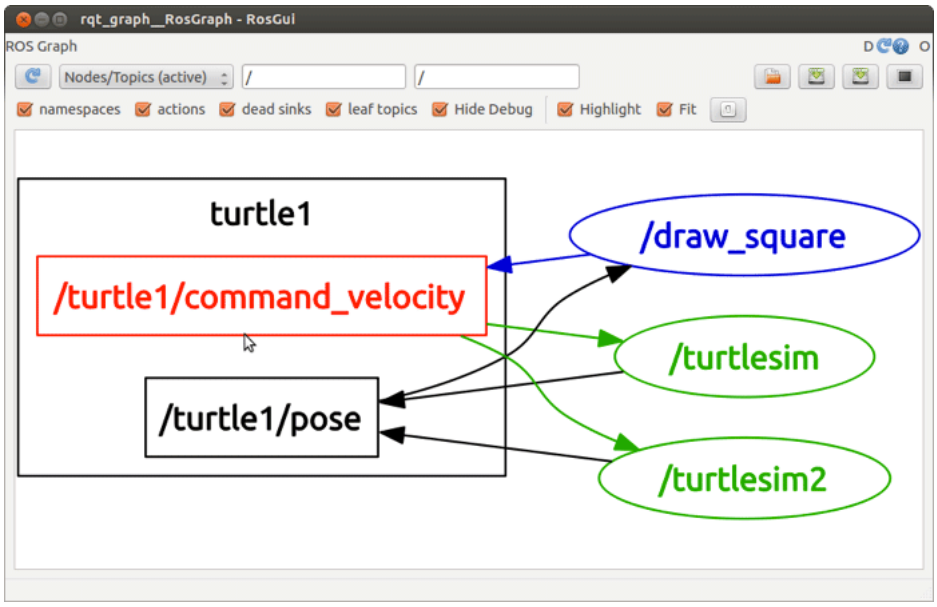


FIGURE 3.5: Exemple de visualisation de noeuds et de topics à l'aide de RQT-Graph

3.3 Implémentation de la commande

Dans cette partie, nous allons nous intéresser à l'implémentation de la commande synthétisée au premier chapitre sous ROS.

Pour cela, nous avons crée un nœud ROS sous Simulink nommé "Quadrirotor_Simulink_-Ros" qui contient le modèle dynamique du quadrirotor. Ce nœud publie sur le topic "/pose_-quadrirotor" le vecteur d'état du quadrirotor et s'abonne sur le topic "/commande_quadrirotor" sur lequel sont publiées les commandes des quatre rotors du quadrirotor. Nous avons aussi créer un autre nœud ROS nommé "Commande_Par_Mode_Glissant", mais cette fois-ci en utilisant le langage C++. Ce nœud publie sur le topic "/commande_quadrirotor" et s'abonne sur le topic "/pose_quadrirotor", figure 3.6, les ellipses représentent des nœuds et les rectangles des topics. Nous utilisons deux machines lors de notre simulation chacune faisant fonctionner un des deux nœuds. La première machine joue le rôle du quadrirotor. Sur cette dernière nous lançons le nœud "Quadrirotor_Simulink_Ros". La deuxième machine joue le rôle d'une station de calcul fixe. Nous lançons sur cette dernière le nœud "Commande_Par_Mode_Glissant".

Pour la première simulation, nous avons donné comme référence un point de coordonnées $\{1, -1, 2\}$. Nous pouvons voir les résultats de la simulation sur les figures 3.7, 3.8 et 3.9.

Nous remarquons que les courbes de la figure 3.7 présentent plus d'ondulations que celles de la figure 1.16 obtenue au premier chapitre. Cela peut s'expliquer par le fait qu'un temps de latence ait été introduit par l'utilisation de deux machines différentes pour effectuer cette simulation. Ce temps de retard introduit un retard de phase de la commande par rapport au système et pourrait déstabiliser le système s'il n'est pas compensé. Malgré cela, le quadrirotor atteint bien le point de référence et possède un comportement relativement bon.

Pour la deuxième simulation, nous avons donné comme référence les sommets d'un carré centré à l'origine et de longueur de côté égale à 2. Nous pouvons voir les résultats de la simulation sur les figures 3.10, 3.11 et 3.12.

Nous constatons que le quadrirotor suit bien la trajectoire de référence malgré la présence du même phénomène de retard présent dans la première simulation.

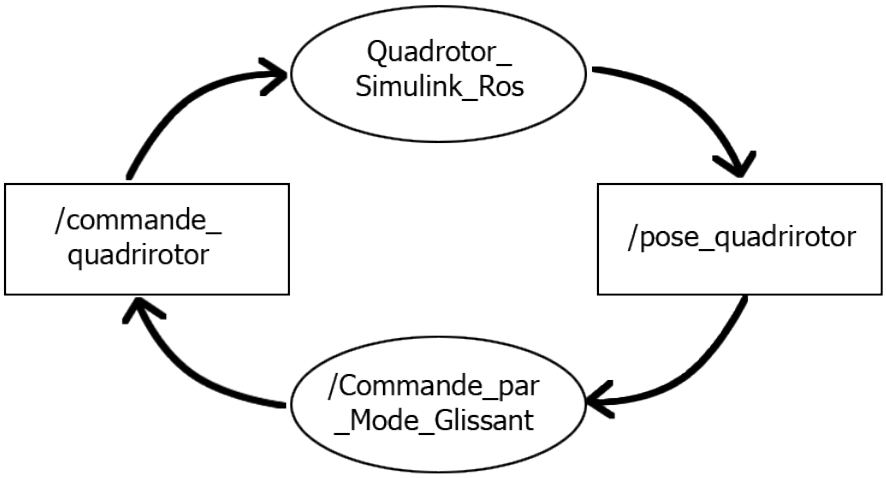


FIGURE 3.6: Schéma des noeuds et topics de la commande

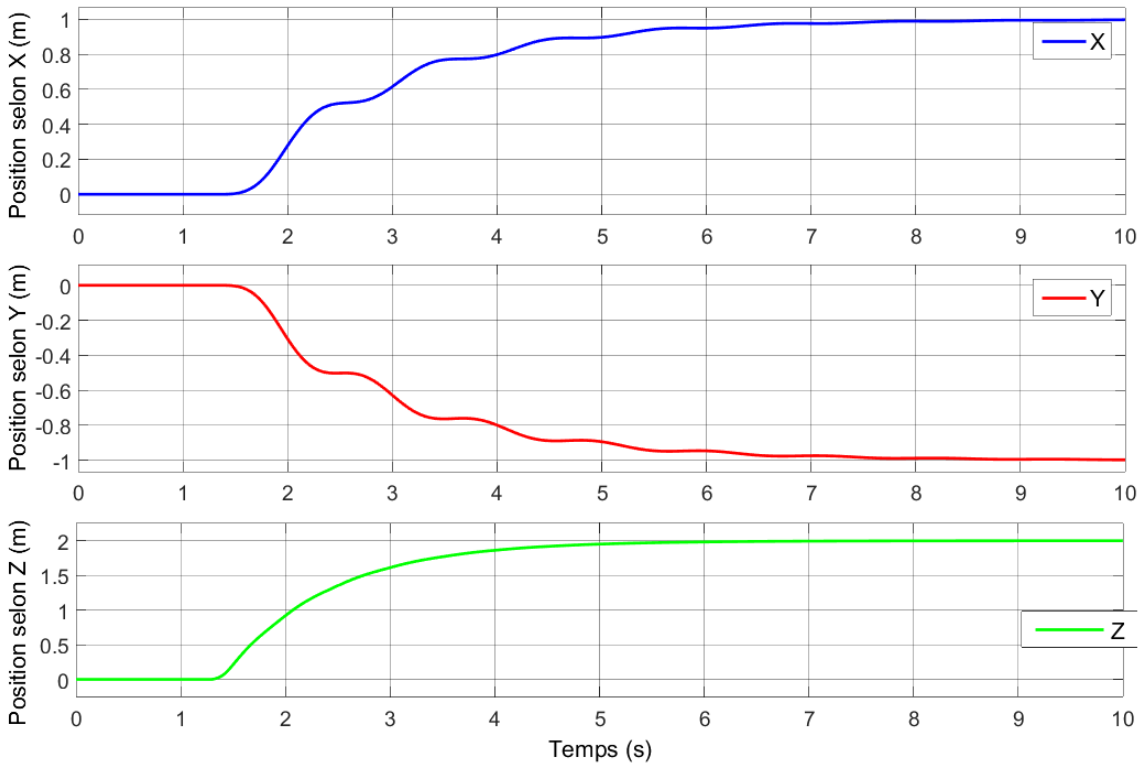


FIGURE 3.7: Evolution de la position du quadrirotor en fonction du temps

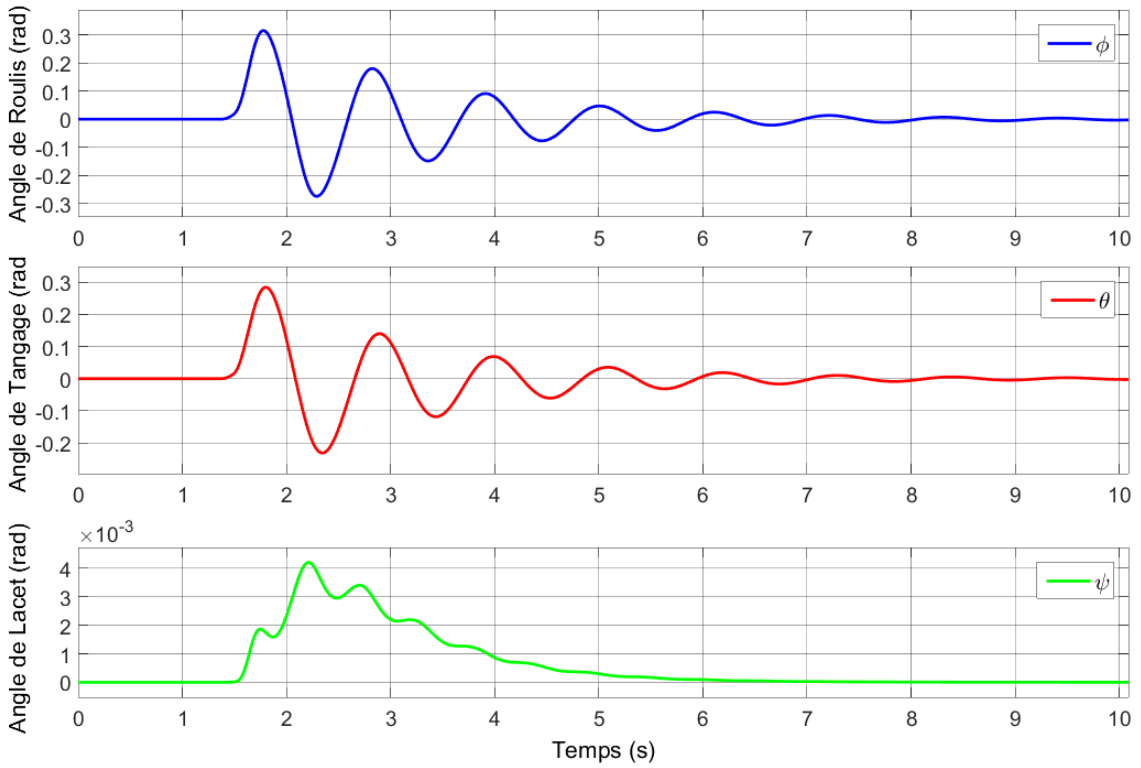


FIGURE 3.8: Evolution de l'attitude du quadrirotor en fonction du temps

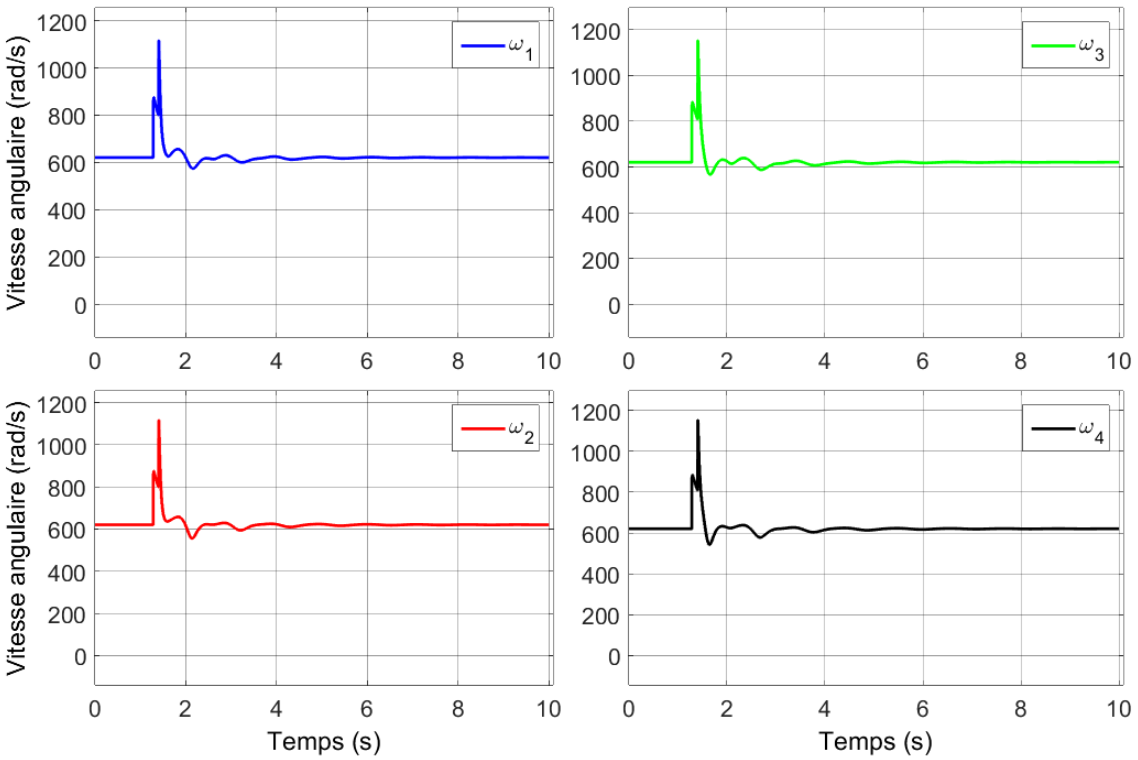


FIGURE 3.9: Evolution de la vitesse angulaire des quatre rotors du quadrirotor en fonction du temps

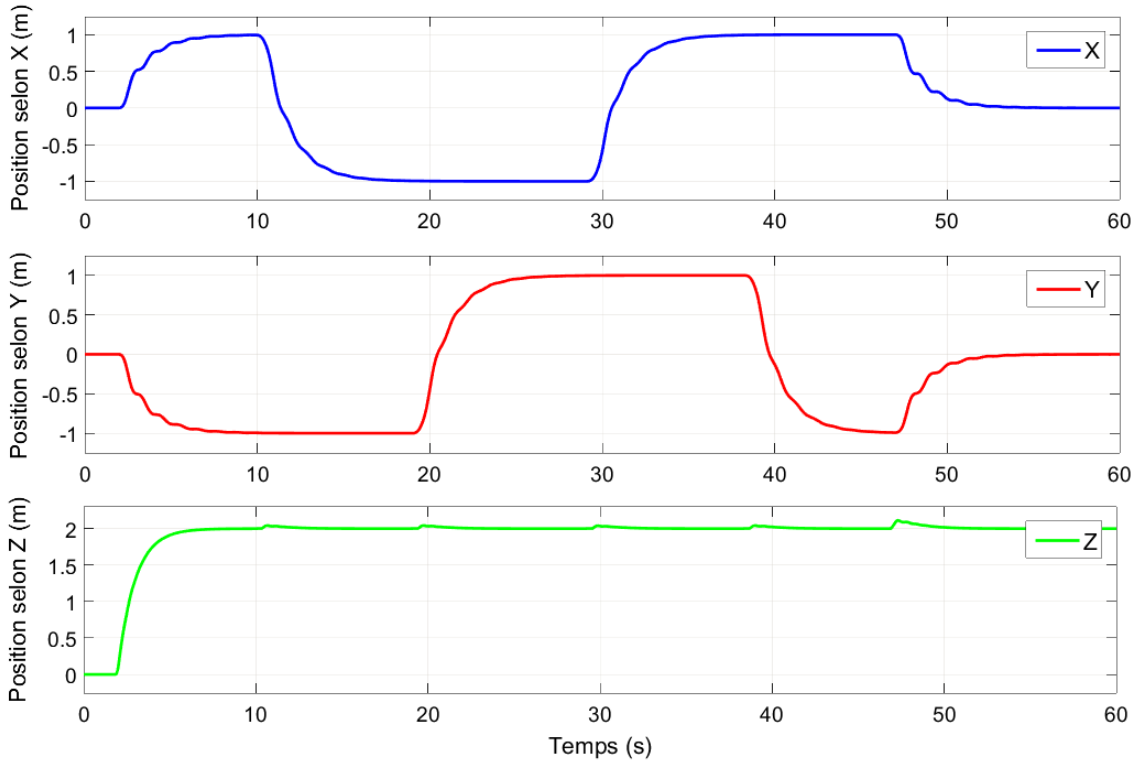


FIGURE 3.10: Evolution de la position du quadrirotor en fonction du temps

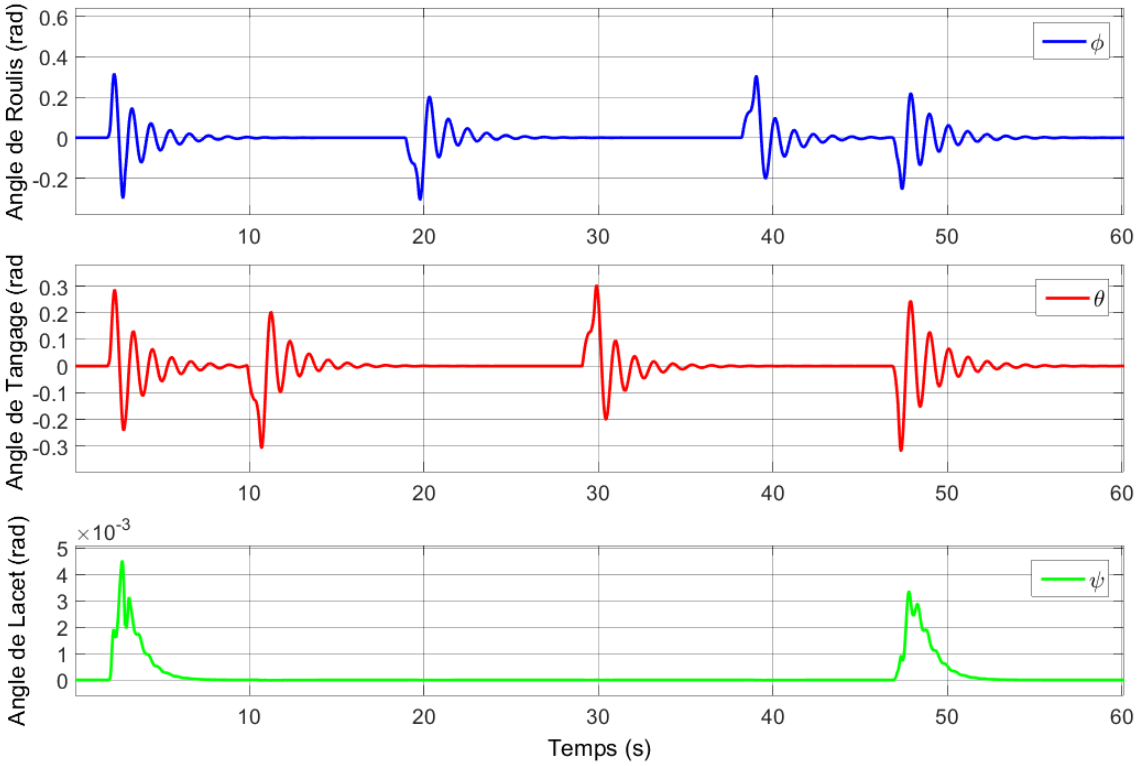


FIGURE 3.11: Evolution de l'attitude du quadrirotor en fonction du temps

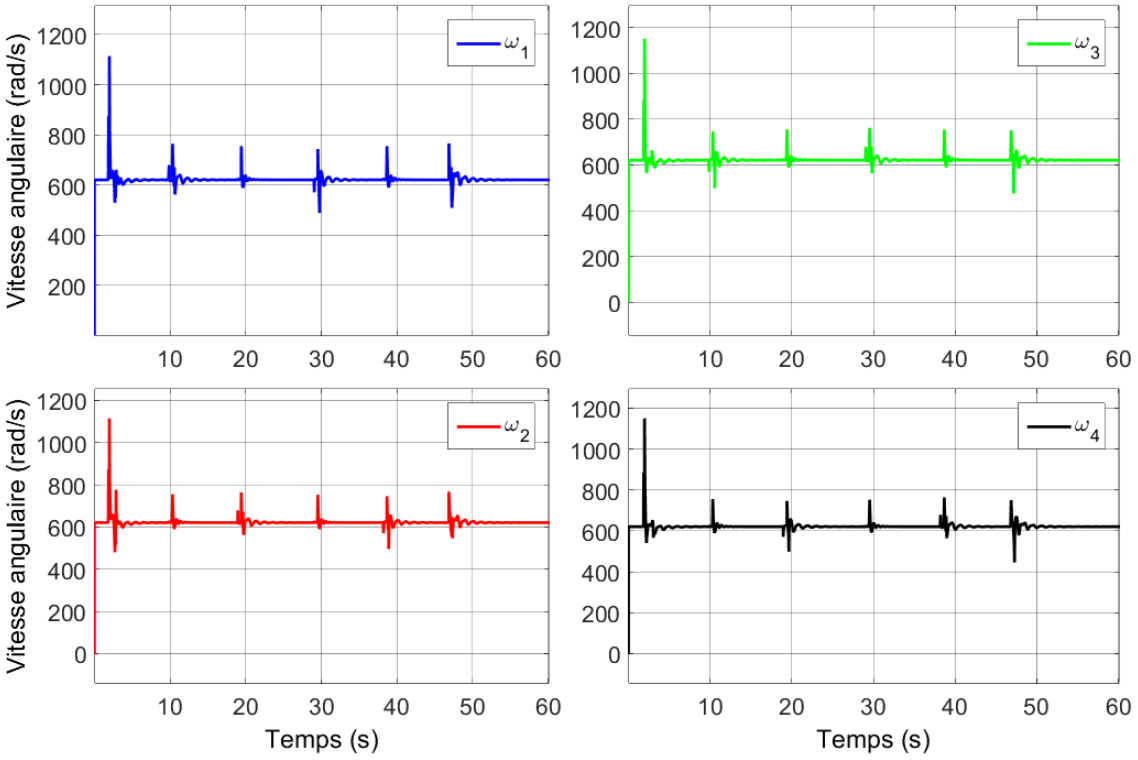


FIGURE 3.12: Evolution de la vitesse de rotation des quatres rotors du quadrirotor en fonction du temps

3.4 Implémentation de l'approche de SLAM

Dans cette section nous allons nous intéresser à l'implémentation de notre approche de SLAM décrite au troisième chapitre.

Pour cela, nous utilisons l'outil de simulation Gazebo, car il offre beaucoup d'avantages par rapport à Matlab. Il nous permet de faire des expériences et des essais physiques réalistes facilement et rapidement et avec de bon graphismes, contrairement à Matlab.

Pour le modèle du quadrirotor, nous utilisons le package "Hector Quadrotor"[86]. Ce dernier contient un modèle de quadrirotor, figure 3.13, équipé de différents capteurs tels qu'un caméra monoculaire, un télémètre laser, une centrale inertielle, un sonar et un altimètre. Ce package contient aussi des interfaces de commandes pour interagir avec le modèle du quadrirotor.

Pour commander ce dernier, nous utilisons l'interface de commandes en vitesse qui s'abonne sur le topic "/command/twist" qui contient des références de vitesses de translation et de rotation à suivre, car on n'a pas pu trouver certains paramètres nécessaires pour le commander directement en utilisant les signaux PWM des 4 rotors. Nous avons implémenter dans un nœud sous ROS nommé "sliding_mode_velocity_controller" une commande par mode glissant pour asservir la position et l'orientation du quadrirotor en utilisant l'interface de commande en vitesse. Ce nœud s'abonne sur le topic "/reference" pour obtenir le point de référence à atteindre. Il calcul la commande par mode de glissement et la publie sur le topic "/command/twist" comme on peut le voir dans la figure 3.14.

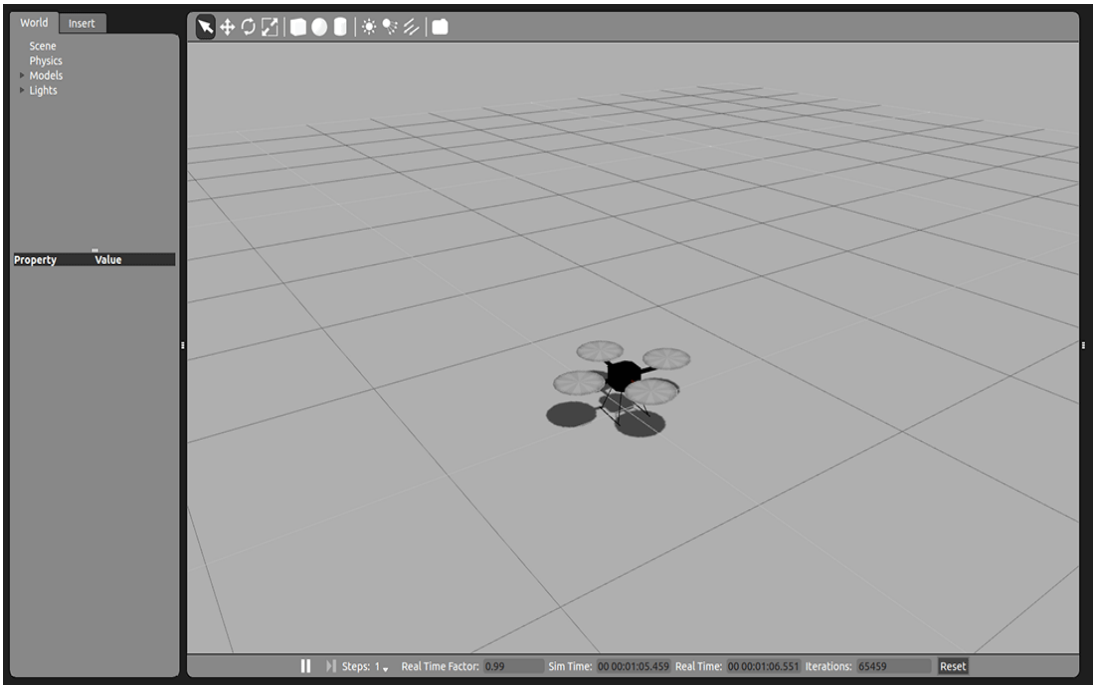


FIGURE 3.13: Modèle du quadrirotor du package "Hector Quadrotor" sous Gazebo

Pour notre approche de SLAM, nous avons décider d'implémenter deux versions différentes : une version avec l'observateur glissant SASMO et une autre avec l'EKF, qui est l'algorithme le plus utilisé, pour pouvoir faire une comparaison entre les deux.

Nous commençons par créer un environnement en 3D sous Gazebo pour pouvoir y faire évoluer le quadrirotor, figure 3.15.

Nous utilisons le code open-source de ORB-SLAM pour créer un noeud sous ROS, nommé "orb_slam", qui va traiter les images provenant de la caméra attachée au qua-

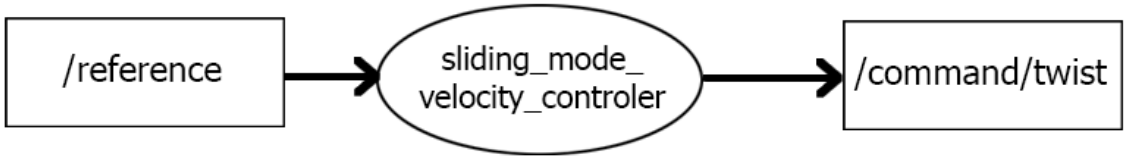


FIGURE 3.14: Schéma de la commande

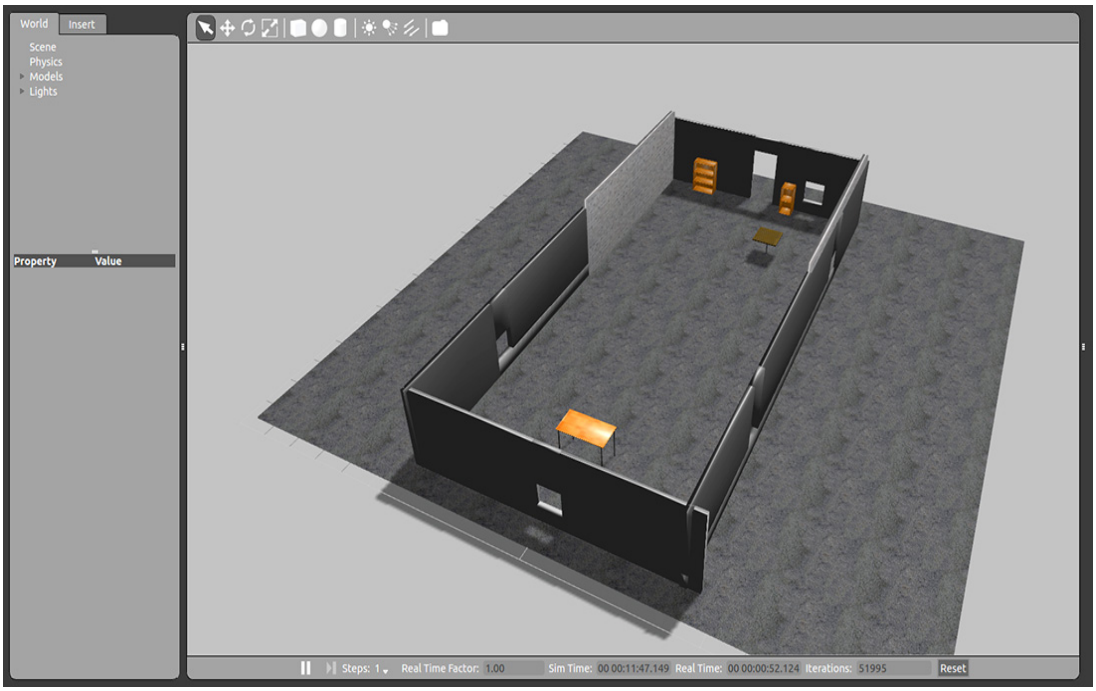


FIGURE 3.15: Environnement 3D de la simulation sous Gazebo

drirotor, présentes sur le topic "/front_cam/camera/image", et qui va détecter les points d'intérêts, figure 3.16 et publier ensuite la position de ces derniers ainsi que celle de la caméra et son orientation définies par rapport à un repère dont l'origine est fixée au point où se trouvait la caméra lorsque l'algorithme s'est initialisé, c'est-à-dire, lorsqu'il a pris deux images clés assez distantes l'une de l'autre pour pouvoir exécuter l'initialisation par la méthode des 5 points[60].

Nous créons aussi un autre nœud sous ROS, nommé "tf_ORB_to_World", qui modifie les positions et les orientations données par ORB-SLAM pour qu'elles aient la même origine et la même orientation que celle du repère fixe lié à l'environnement. Ce nœud publie cette estimation sur le topic "orb_estimated_pose". Nous créons enfin un nœud qui implémente l'algorithme d'estimation, nommé "ekf_slam" dans le cas du EKF et "sasmo_slam" dans le cas de l'observateur glissant SASMO, qui reçoit les estimations de la position et de l'attitude donnée par ORB-SLAM, les mesures de l'accélération linéaire et de la vitesse angulaire provenant de la centrale inertielle, la mesure de l'altitude provenant du sonar et les mesures des commandes et qui nous donne l'estimation finale de la position et de l'attitude du quadrirotor, comme on peut le voir sur la figure 3.17.

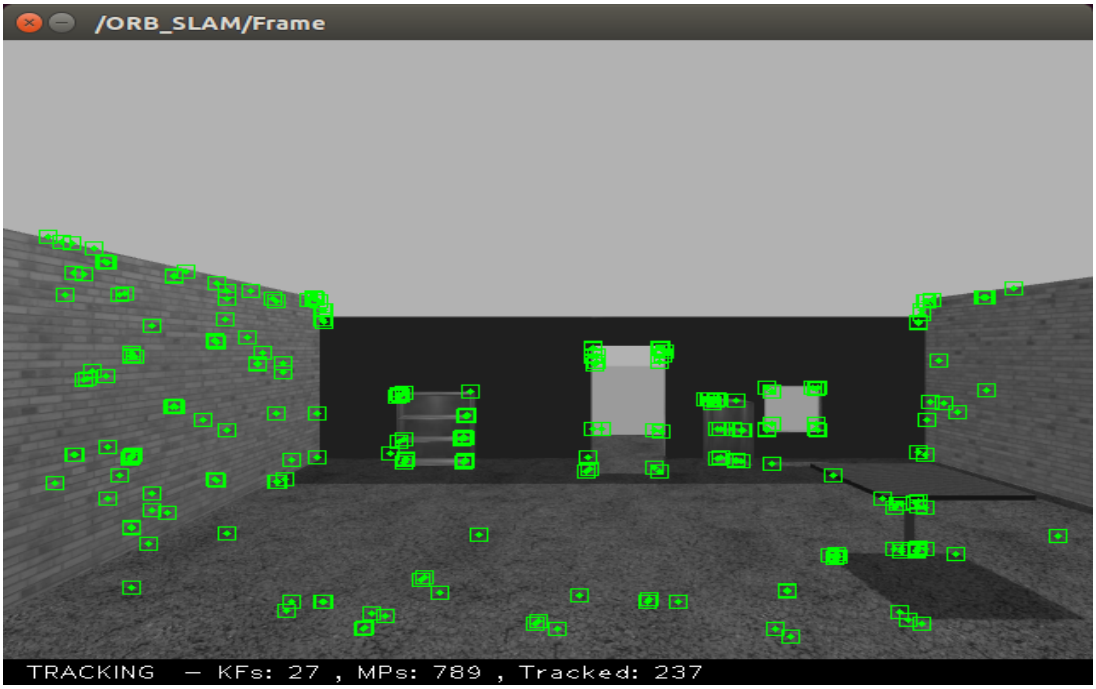


FIGURE 3.16: Détection des points d'intérêts par orb_slam

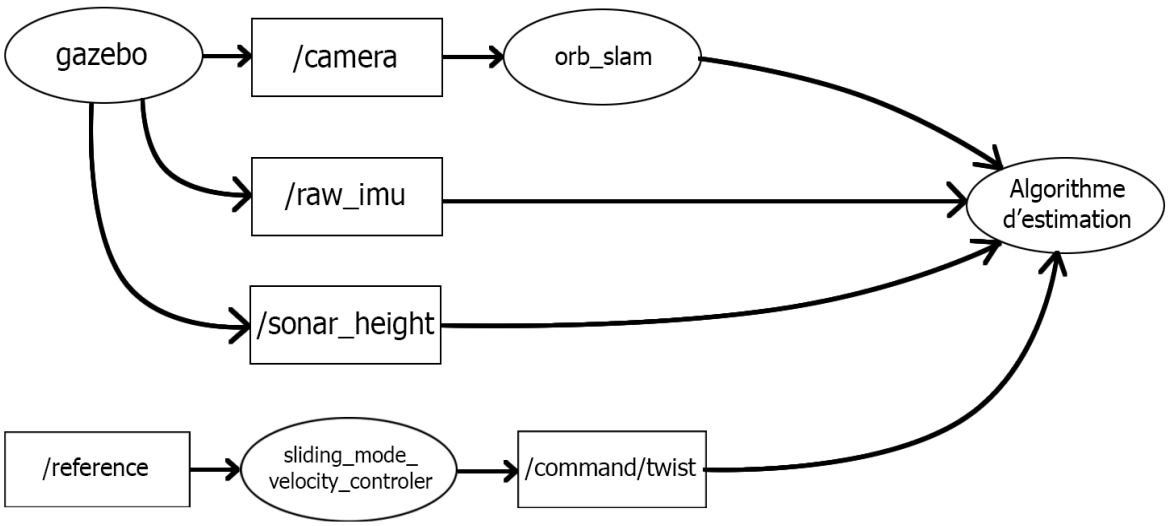


FIGURE 3.17: Les nœuds et les topics utilisés pour réaliser notre algorithme de SLAM

Pour la simulation, nous procédons de la manière suivante :

- Nous commençons par lancer Gazebo ainsi que tout les nœuds décrits précédemment.
- Ensuite, nous attendons que l'algorithme ORB-SLAM se lance, cela prend en moyenne 12 secondes.
- Après cela, nous faisons suivre au quadrirotor des mouvements de translation pour que ORB-SLAM puisse initialiser la position et l'orientation de la caméra, ainsi que la carte et les premiers points d'intérêt.
- Ensuite, nous faisons suivre au quadrirotor d'autres mouvements de translation pour pouvoir faire l'estimation de l'échelle, car elle dépend des mouvements du quadrirotor.
- Enfin, après la convergence de l'échelle, nous donnons au quadrirotor comme référence les sommets d'un carré centré à l'origine et de longueur de côté égale à 4.

3.4.1 Approche SASMO

Nous pouvons voir les résultats de la simulation sur les figures 3.18, 3.19, 3.20, 3.21, 3.22 et 3.23.

Nous constatons qu'entre le début de la simulation et l'initialisation d'ORB-SLAM, l'estimation de la position suivant X (respectivement Y), augmente (respectivement diminue) malgré que la position du quadrirotor est fixe. Cela est dû aux bruits qui entache les mesures provenant de l'accéléromètre de la centrale inertielle.

Nous remarquons aussi qu'à l'instant où ORB-SLAM s'initialise, les estimations données par l'observateur glissant divergent, puis convergent de nouveaux après quelques instants. Cela peut s'expliquer par le fait que l'initialisation de l'erreur d'estimation à l'instant d'initialisation d'ORB-SLAM introduit un grand gain de glissement qui diminue rapidement dés que l'erreur se stabilise.

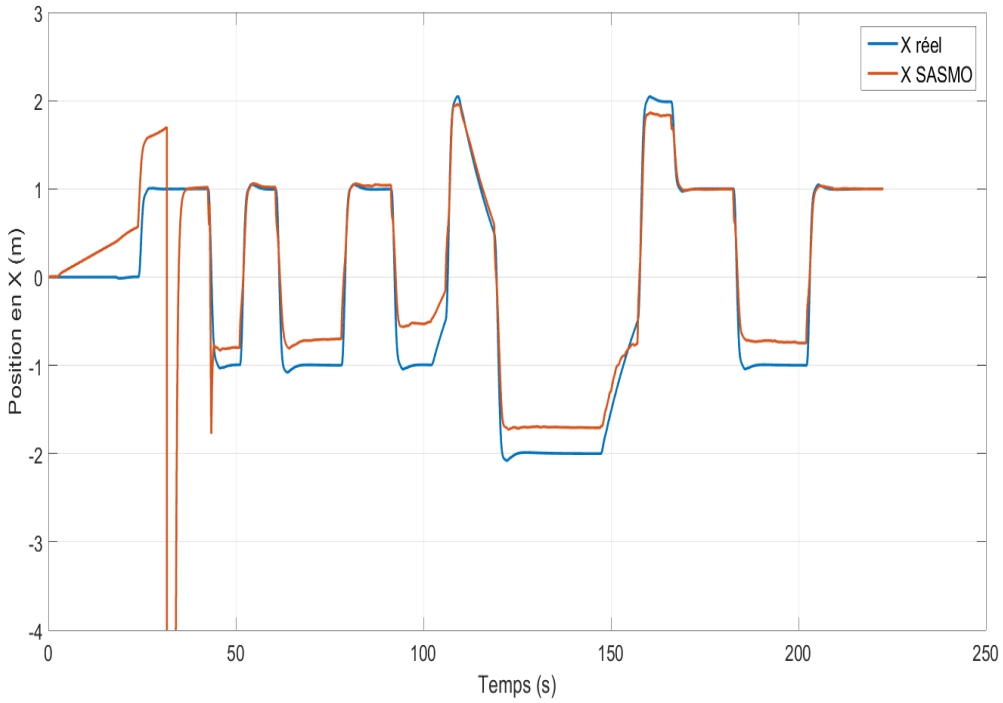


FIGURE 3.18: Evolution de la position suivant X du quadrirotor en fonction du temps

Après que l'échelle se soit stabilisée, nous constatons que les estimations des positions suivent bien l'allure des position réelles. Il persiste néanmoins une erreur statique qui est due au fait que l'échelle ait convergé vers une fausse valeur. Pour ce qui est de l'attitude, les estimations des angles de Roulis et de Tangages sont assez proches de angles réels. Cependant, pour l'estimation de l'angle de Lacet, nous remarquons qu'il persiste de faibles erreurs statiques.

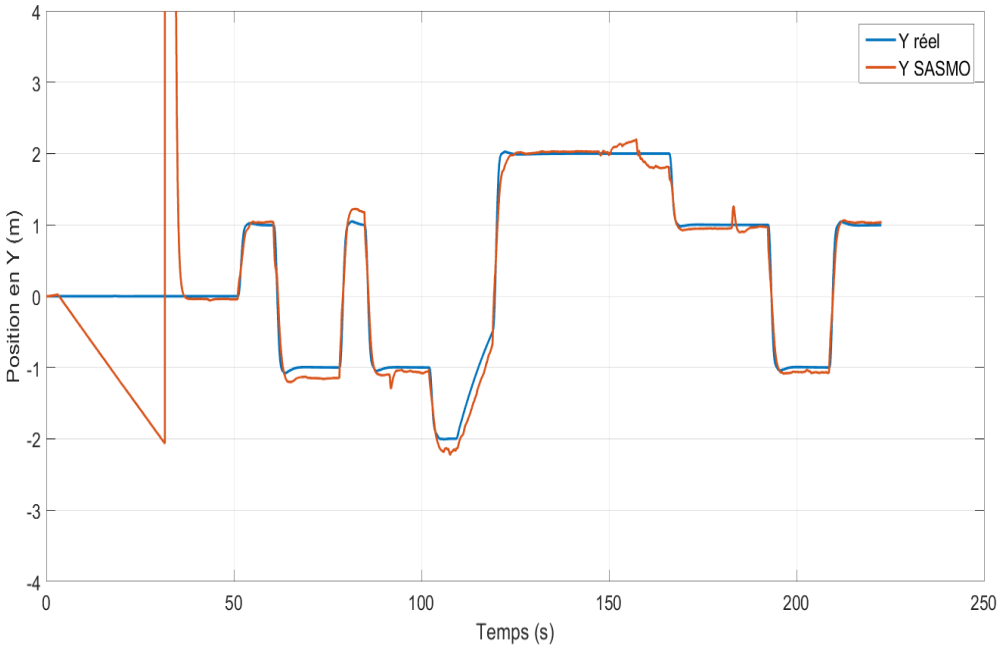


FIGURE 3.19: Evolution de la position suivant Y du quadrirotor en fonction du temps

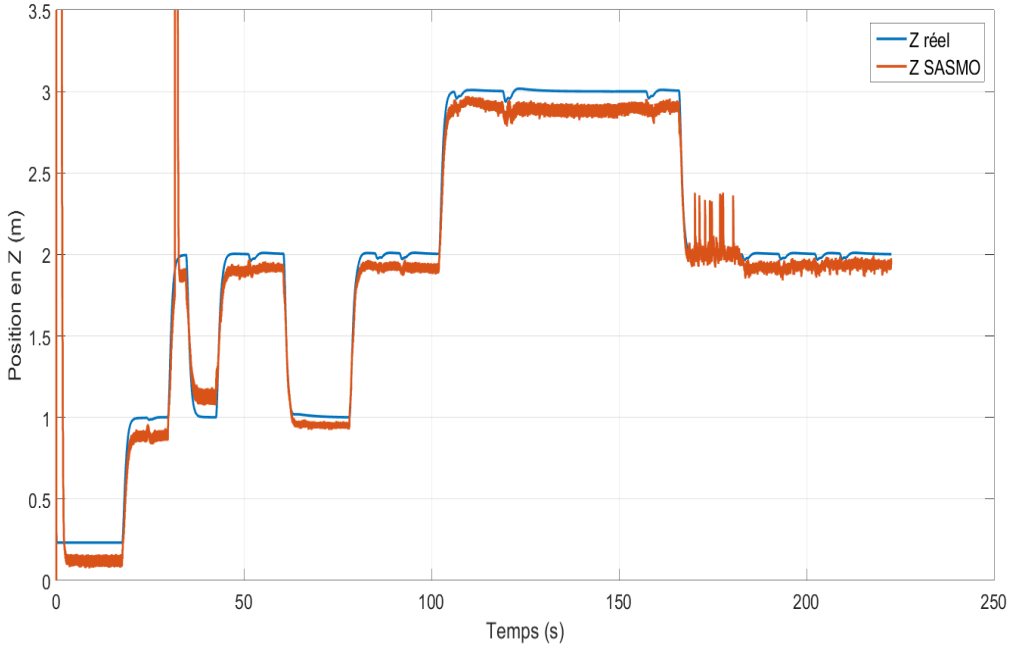


FIGURE 3.20: Evolution de la position suivant Z du quadrirotor en fonction du temps

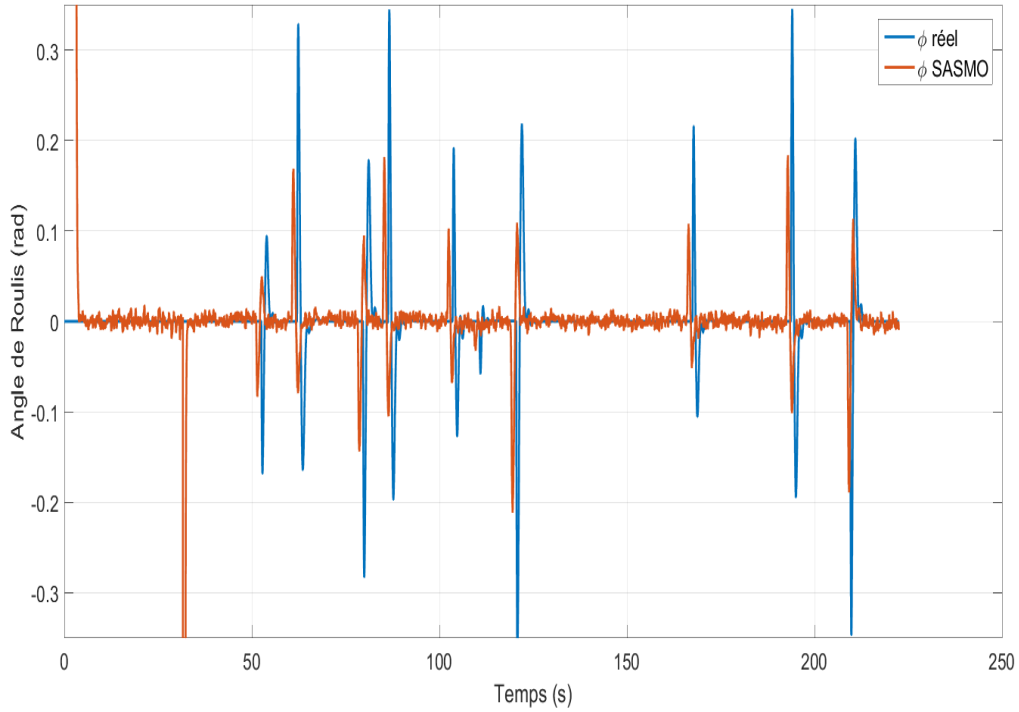


FIGURE 3.21: Evolution de l'angle de Roulis du quadrirotor en fonction du temps

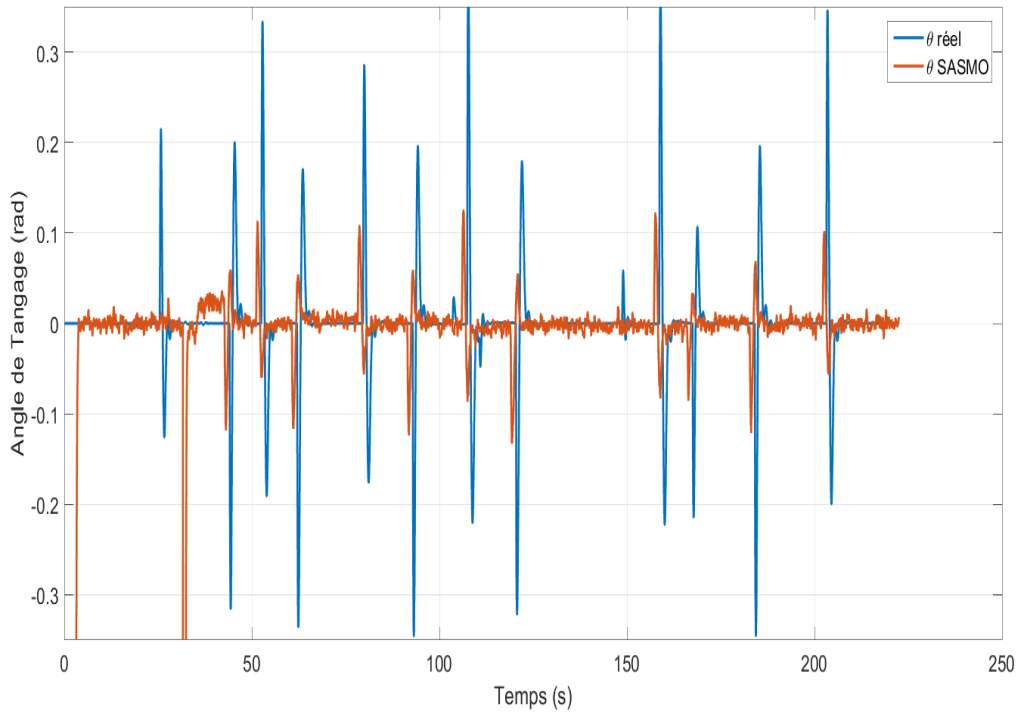


FIGURE 3.22: Evolution de l'angle de Tangage du quadrirotor en fonction du temps

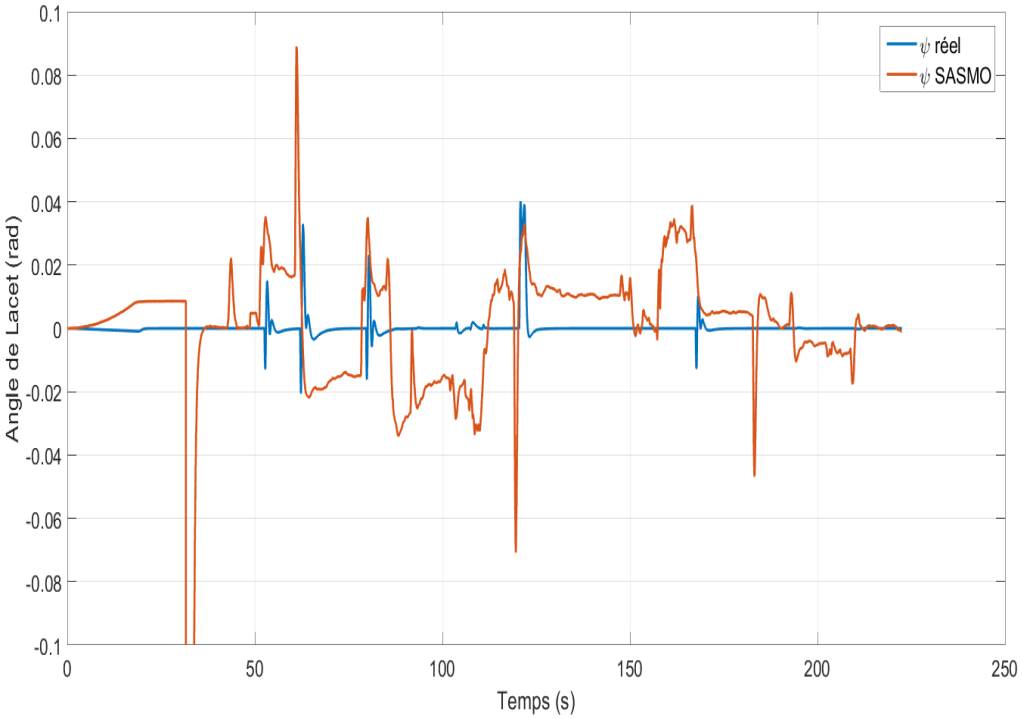


FIGURE 3.23: Evolution de l'angle de Lacet du quadrirotor en fonction du temps

3.4.2 Approche EKF

Nous pouvons voir les résultats de la simulation sur les figures 3.24, 3.25, 3.26, 3.27, 3.28 et 3.29.

Nous constatons qu'entre le début de la simulation et le lancement d'ORB-SLAM, l'estimation de la position suivant X et Y ne change pas contrairement à l'approche précédente.

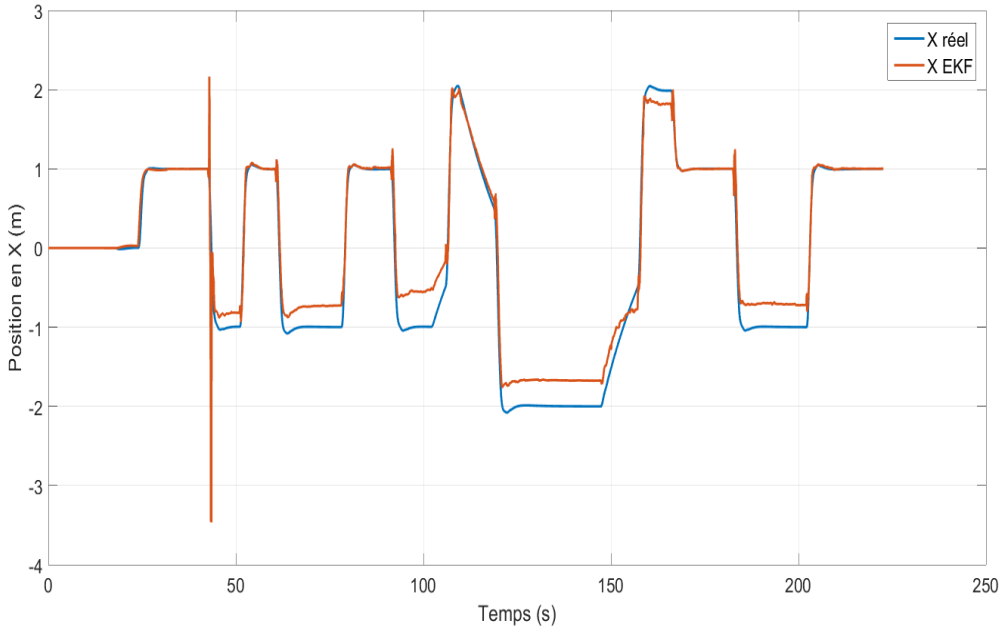


FIGURE 3.24: Evolution de la position en X du quadrirotor en fonction du temps

Nous remarquons aussi qu'à l'instant où ORB-SLAM s'initialise, les estimations données par l'EKF ne divergent contrairement à l'approche précédente.

Après que l'échelle se soit stabilisée, nous constatons que les estimations des positions suivent bien l'allure des position réelles. Il persiste néanmoins une erreur statique qui est due au fait que l'échelle ait convergé vers une fausse valeur.

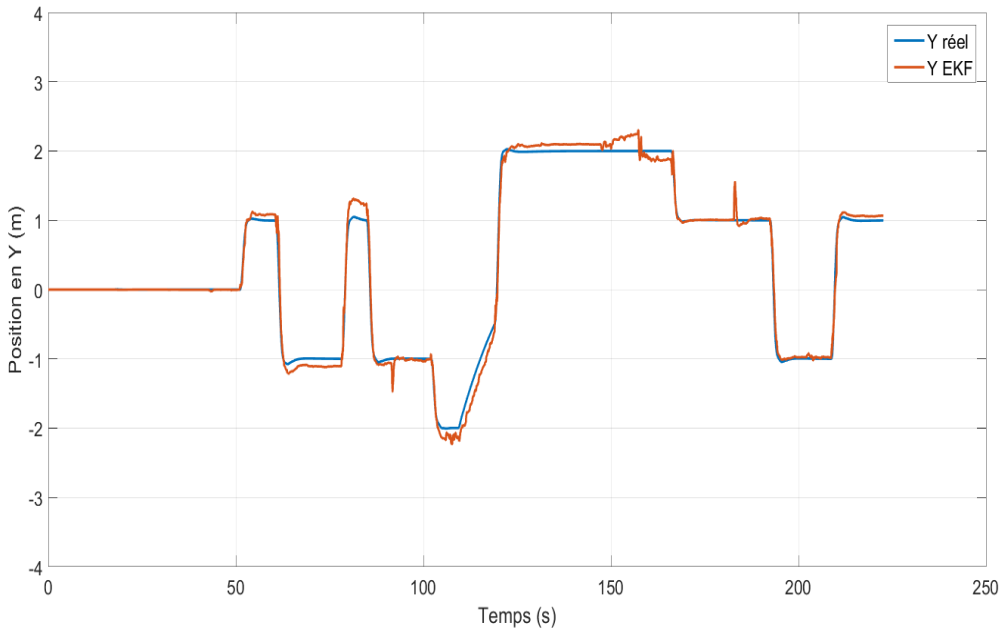


FIGURE 3.25: Evolution de la position en Y du quadrirotor en fonction du temps

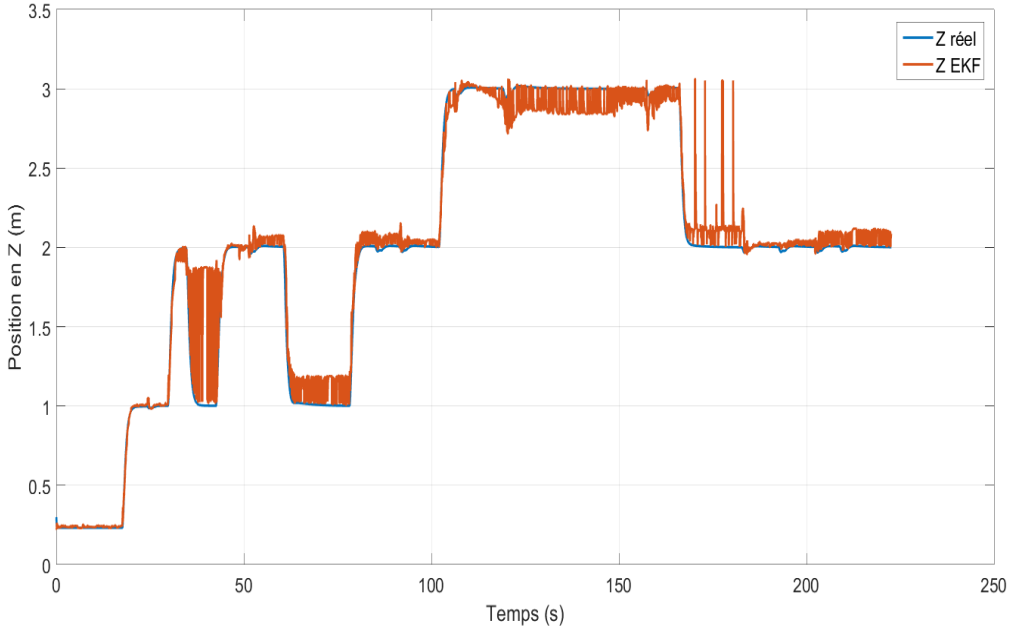


FIGURE 3.26: Evolution de la position en Z du quadrirotor en fonction du temps

Nous constatons aussi que l'estimation de l'altitude est entachée de beaucoup d'oscillations, contrairement à l'approche précédente, et cela à cause de la différence entre la

mesure donnée par ORB-SLAM avec correction de l'échelle et le sonar. Pour ce qui est de l'attitude, les estimations des angles de Roulis et de Tangages sont très bruitées et ne suivent pas l'allure des angles réels, contrairement à l'approche précédente. Cependant, pour l'estimation de l'angle de Lacet, nous remarquons que cette dernière ne suit pas bien la valeur réelle et qu'elle diverge à la moitié de la simulation.

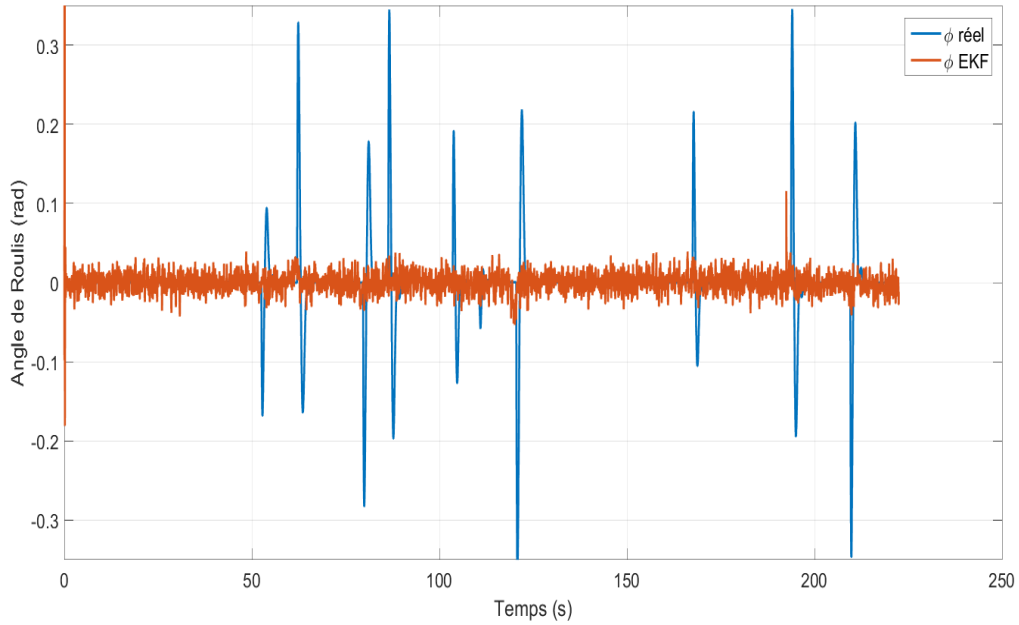


FIGURE 3.27: Evolution de l'angle de Roulis du quadrirotor en fonction du temps

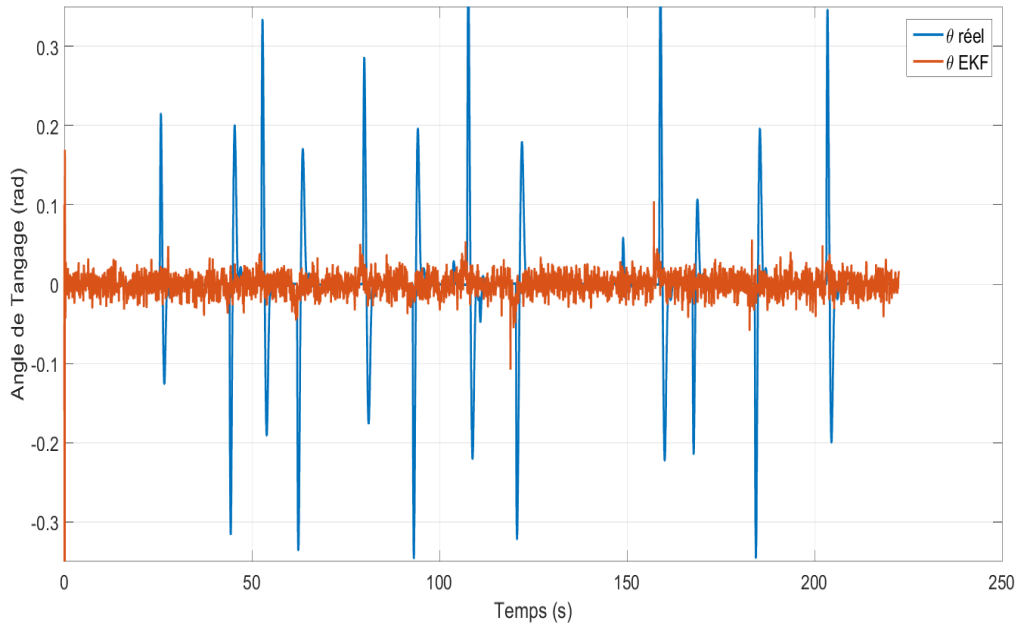


FIGURE 3.28: Evolution de l'angle de Tangage du quadrirotor en fonction du temps

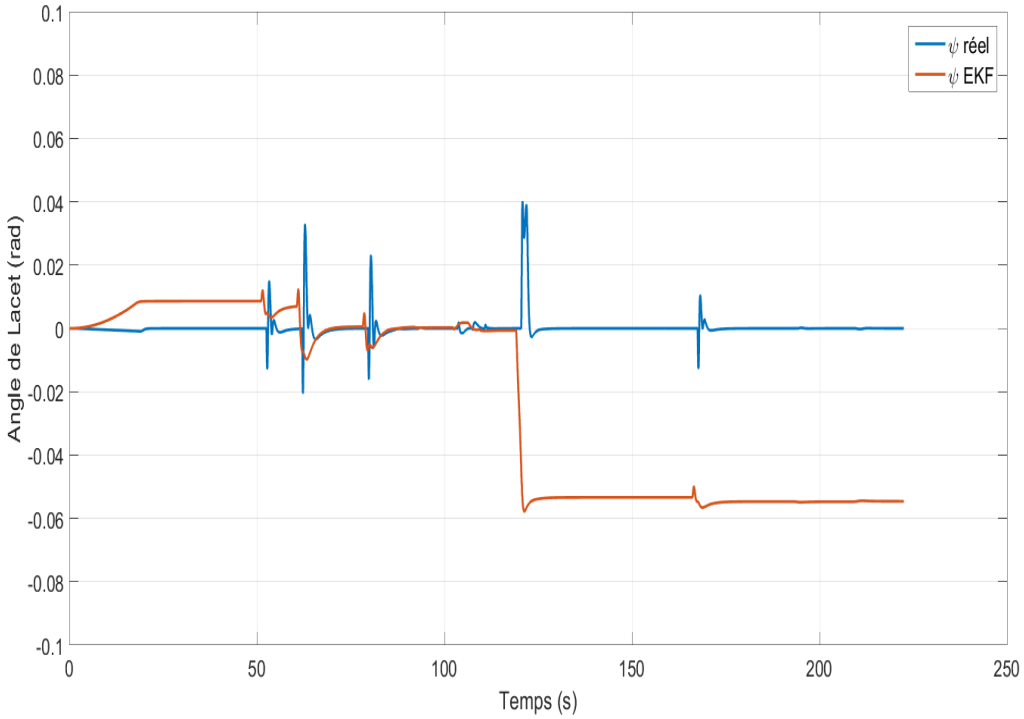


FIGURE 3.29: Evolution de l'angle de Lacet du quadrirotor en fonction du temps

3.4.3 Carte de l'environnement

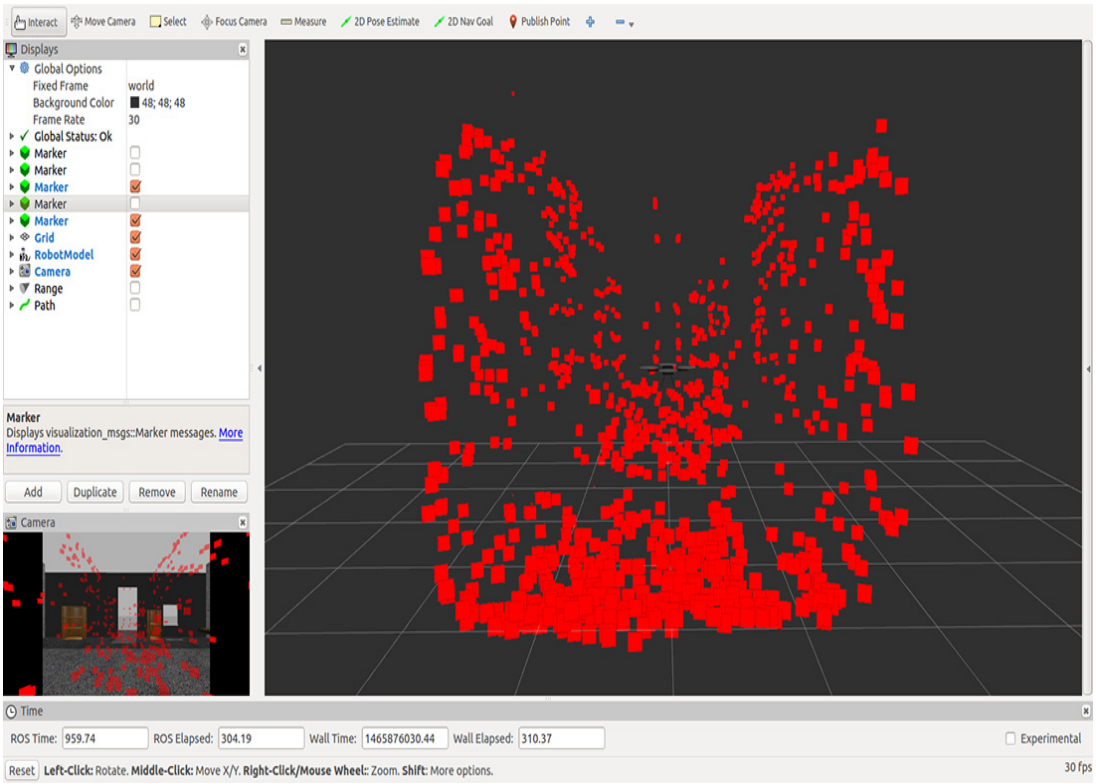


FIGURE 3.30: Vue de dos de la carte des amères générée par notre approche

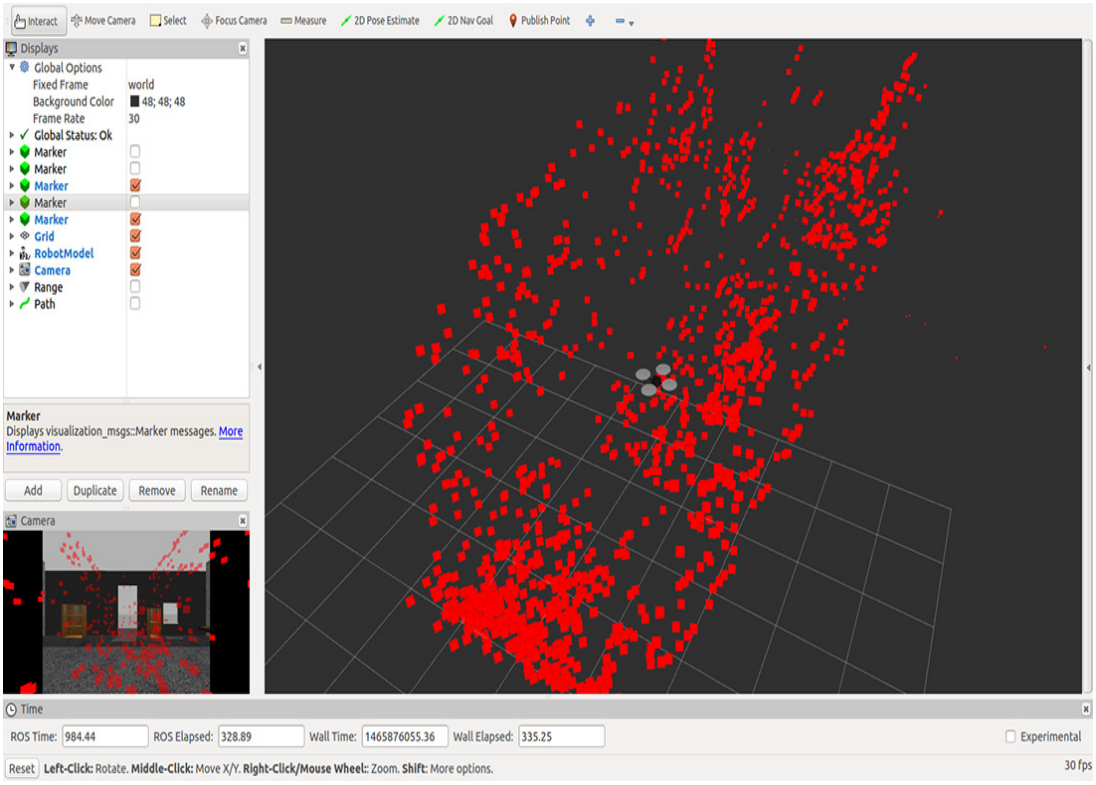


FIGURE 3.31: Vue en perspective de la carte des amères générée par notre approche

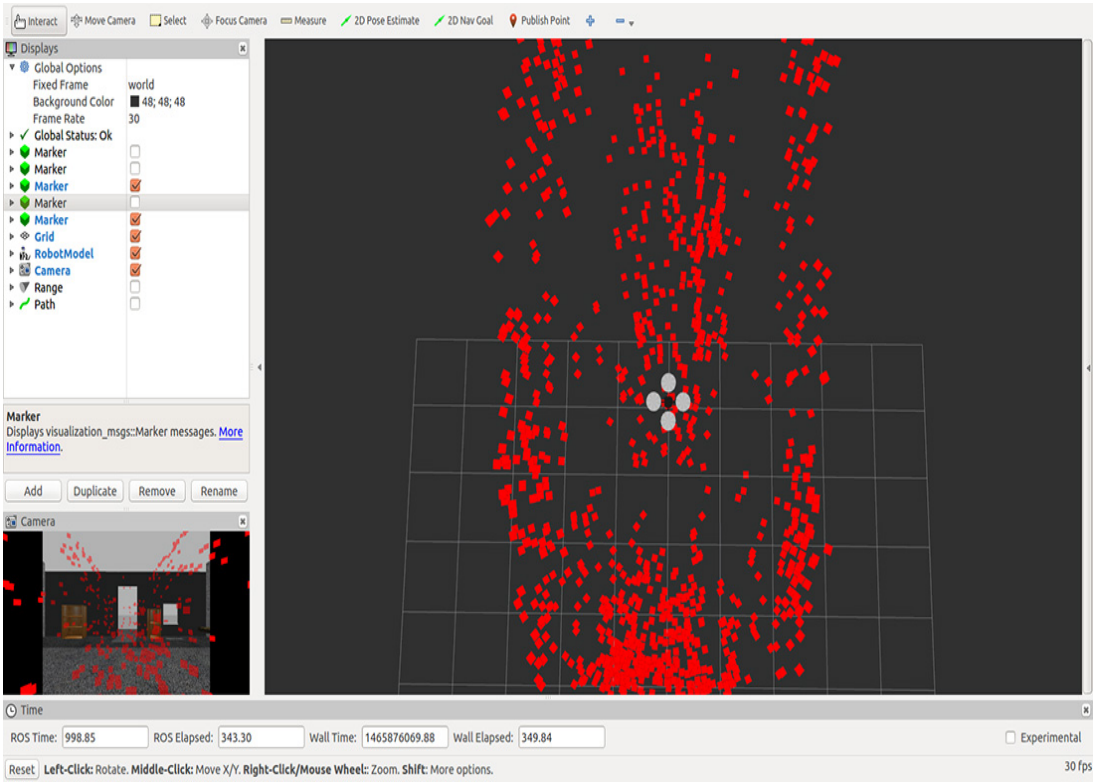


FIGURE 3.32: Vue de dessus de la carte des amères générée par notre approche

La carte donnée par notre approche est indépendante de l'algorithme d'estimation utilisé. Elle ne dépend que des estimations d'ORB-SLAM et de l'estimation de l'échelle. Comme nous pouvons le voir, figures 3.30, 3.31 et 3.32, la carte est fidèle à l'environnement 3D crée et utilisé pour notre simulation.

3.5 Conclusion

Dans ce chapitre, nous avons présenté ROS, ses notions de bases et nous avons expliqué en quoi réside son importance. Nous avons ensuite implémenté sous ce dernier la commande par mode glissant synthétisée au chapitre 1 et nous avons vu les bons résultats qu'elle offre. Puis, nous avons implémenté deux algorithmes de SLAM sous ROS et Gazebo. L'un avec l'observateur par mode de glissement "SASMO" et l'autre avec l'EKF. Enfin, nous avons simulé et présenté les résultats et fait une comparaison qualitative entre les deux. Nous avons comme conclusion que les résultats donnée par l'EKF et le SASMO sont assez proches les uns des autres.

Chapitre 4

Mise en Œuvre Pratique

4.1 Introduction

Dans ce chapitre, nous allons nous intéresser à la réalisation d'un quadrirotor et à la mise en œuvre pratique de commandes. Nous allons tout d'abord commencer par une présentation de ce dernier. Nous allons ensuite présenter le matériel utilisé pour la réalisation ainsi que les logiciels, bibliothèques et le langage de programmation utilisés. Puis, nous expliquerons la méthode d'acquisition des données des capteurs ainsi que les problèmes liés à ces derniers et la manière dont nous les avons résolus. Après cela, nous passerons à l'identification des paramètres du système, qui est une étape clés pour l'implémentation des commandes. Enfin, nous finirons par la mise en œuvre pratique d'une commande par PID pour l'asservissement de l'angle de lacet ψ.

4.2 Présentation du quadrirotor

FIGURE 4.1: Notre quadrirotor

Comme on peut le voir sur la figure 4.1 notre quadrirotor n'est pas en forme de croix comme ceux modélisés sous Matlab ou Gazebo aux chapitres précédents. Il est en forme de H et cela pour que la caméra qui est fixée dessus ne soit pas gênée par les hélices dans

son champ de vision. Ce changement de forme implique quelques changements au niveau des repères du quadrirotor comme on peut le voir sur la figure 4.2.

Cela implique aussi quelques changements aux niveau des commandes de l'équation(1.36) qui deviennent :

$$\begin{cases} U_1 = T = b(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ U_2 = \tau_\phi = \frac{L_1}{2} \cdot b(-\omega_1^2 - \omega_2^2 + \omega_3^2 + \omega_4^2) \\ U_3 = \tau_\theta = \frac{L_2}{2} \cdot b(-\omega_1^2 + \omega_2^2 + \omega_3^2 - \omega_4^2) \\ U_4 = \tau_\psi = k \cdot (-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{cases} \tag{4.1}$$

Et l'équation des vitesses angulaires de rotors en fonctions des commandes(1.63) devient :

$$\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \begin{bmatrix} \frac{1}{4b} & -\frac{1}{2L_1 \cdot b} & -\frac{1}{4L_2 \cdot b} & -\frac{1}{4k} \\ \frac{1}{4b} & -\frac{1}{2L_1 \cdot b} & \frac{1}{2L_2 \cdot b} & \frac{1}{4k} \\ \frac{1}{4b} & \frac{1}{2L_1 \cdot b} & \frac{1}{2L_2 \cdot b} & -\frac{1}{4k} \\ \frac{1}{4b} & \frac{1}{2L_1 \cdot b} & -\frac{1}{2L_2 \cdot b} & \frac{1}{4k} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{4.2}$$

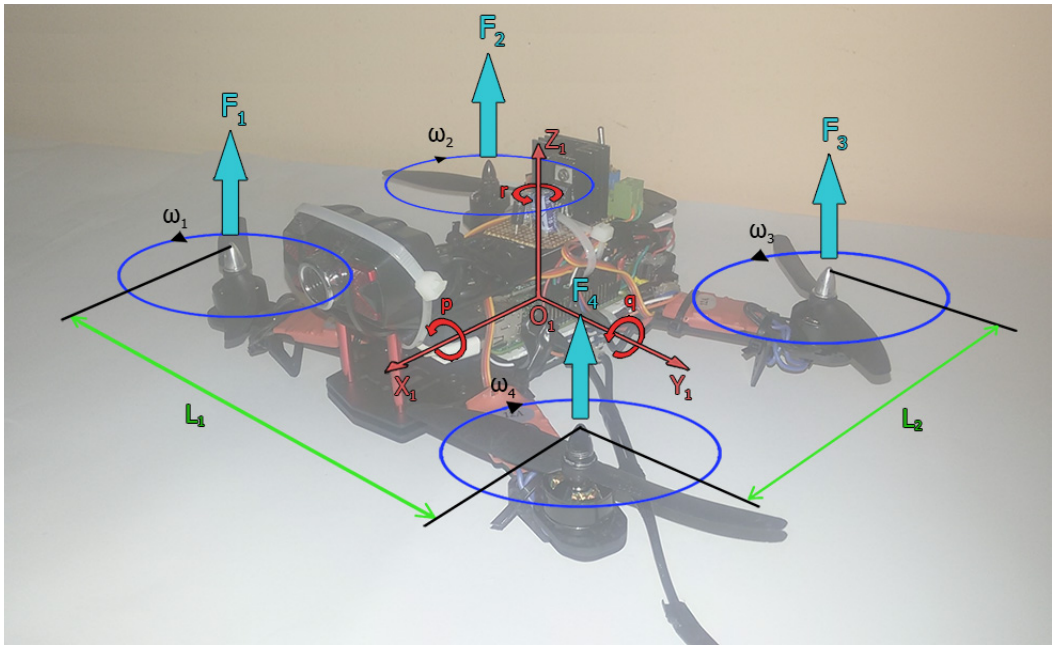


FIGURE 4.2: Repère du quadrirotor

4.3 Le matériel

Pour la réalisation de notre quadrirotor, nous avons utilisé des composants disponibles au grand public :

- Un Raspberry Pi 2
- Un châssis en fibre de carbone
- Quatre moteurs BLDC MT1806 2280kv et ESC 12A Simonk
- Un contrôleur de signaux PWM PCA9685
- Quatre hélices de diamètre égale à 12.7 cm
- Une carte de répartition de l'alimentation
- Un régulateur de tension 5v LM2576
- Une centrale inertielle MPU6050

- Un sonar ultrason HC-SR04
- Un récepteur WIFI TL-WN722N
- Une webcam

4.3.1 Raspberry Pi

Le Raspberry Pi[87] est un nano-ordinateur monocarte mis au point par le Raspberry Pi Foundation dans un but éducatif. il permet l'exécution de plusieurs variantes du système d'exploitation libre GNU/Linux et des logiciels compatibles. Il en existe plusieurs versions dont la version 2, figure 4.3, que nous utilisons dans notre réalisation. Le cœur de l'ordinateur est un processeur ARM Cortex-A7 cadencé à 900MHz et de nombreux périphériques. Elle peut être directement être connectée à une IHM classique, souris/clavier/écran HDMI ou vidéo composite, cependant comme tout ordinateur Linux, le Raspberry Pi peut intégrer ses propres outils de développement et une interface homme-machine reposant sur le SSH et est donc contrôlable depuis un autre ordinateur par Ethernet ou WIFI. Le Raspberry Pi est aussi dotée d'un connecteur GPIO (General Purpose Input Output, littéralement Entrée/Sortie pour un Usage Général) qui possède des broches d'entrées/sorties binaires, des broches pour des protocoles de communications tels que l'UART, l'I2C et le SPI et des broches d'alimentation 5v et 3v3.

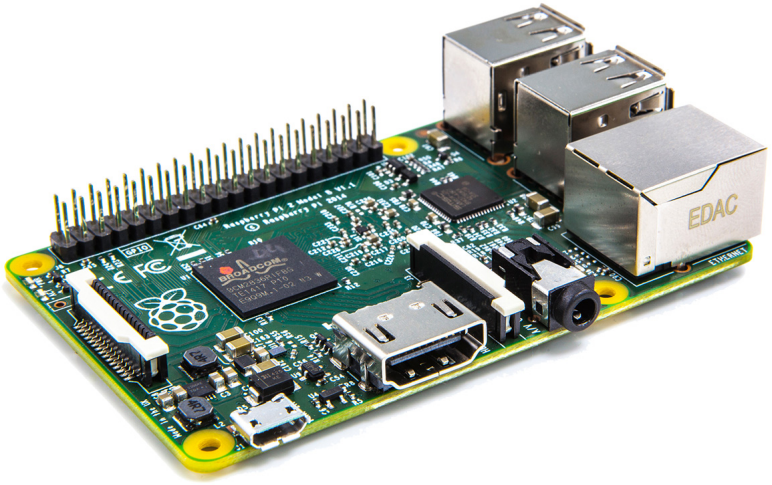


FIGURE 4.3: Raspberry Pi version 2

4.3.2 Le Châssis

C'est la structure sur laquelle se fixe toutes les autres parties du quadrirotor. Le choix de cette influe grandement sur les performances, la robustesse et la stabilité du système.

Pour avoir de bonnes caractéristique de vol, il faut choisir un châssis qui soit symétrique et qui offre le moins de déformation et de flexion possibles.

Dans notre projet, nous avons choisit d'utiliser un châssis en fibre de carbone en forme de H avec des dimension égales à : $22cm18cm7cm$, figure 4.4.

4.3.3 Moteur Brushless (BLDC)

Le moteur brushless (BLDC) se comporte comme un moteur à courant continu traditionnel. Il présente des caractéristiques semblables à celles des moteurs à courant continu et alternatif sans les inconvénients : une forte dynamique de vitesse et d'accélération sans

FIGURE 4.4: Le châssis de notre quadrirotor

l'usure mécanique des moteurs courant continu ; la commutation électronique se substituant à la commutation mécanique. Ils entrent dans la catégorie des moteurs synchrones, ce qui signifie que le champ magnétique créé par le stator et celui généré par le rotor tournent à la même fréquence.

Les moteurs BLDC sont largement utilisés dans l'industrie, l'aérospatiale, l'automobile et dans le modélisme ou ils sont utilisés pour faire se mouvoir des modèles réduits d'avions, d'hélicoptères ainsi que de petits drones tels que les quadrirotors.

Le choix d'un moteur BLDC doit être fait en prenant en compte l'usage que l'on en fera et les différentes caractéristiques qu'il possède.

Kv : la constante de vitesse par volt, elle est indiquée en RPM (rotations par minute) par volt, à vide. Cela signifie donc qu'un moteur de 1000 Kv tourne à 1000 tours/min à 1 Volt, à vide. Dans notre cas, nous ne faisons pas tourner les moteurs à vide mais avec une hélice. Pour maintenir le nombre de tours par minute il faut compenser la résistance aérodynamique de l'hélice. Ceci nécessité plus de puissance.

De manière générale, plus un moteur tourne vite moins il a de couple, et inversement. Un moteur avec un Kv inférieur, pourra donc supporter de plus grandes hélices qu'un moteur à Kv élevé.

Dans notre réalisation, Nous allons utiliser un moteur BLDC MT1806, figure 4.5, ayant entre 2280 Kv avec une masse de 18g pour des hélices de diamètre égales à 12.7 cm.

4.3.4 Electronic Speed Controller (ESC)

Le contrôle d'un moteur BLDC doit donc obligatoirement se faire grâce à un circuit électronique auxiliaire. En effet, c'est le circuit de commande qui va exciter de façon successive les différentes bobines du stator. Pour créer un champ magnétique tournant, le circuit de commande devra exciter les bobines dans un ordre approprié (séquence de commutations) et cela au moment opportun.

C'est l'ESC ou variateur électronique qui délivre le courant et l'ampérage au moteur.

FIGURE 4.5: Moteur BLDC MT1806

Son rôle est de réguler la tension pour donner plus ou moins de vitesse au moteur, mais aussi de laisser passer l'ampérage dont il aura besoin.

Pour un quadrirotor, nous avons besoin d'utiliser quatre ESC, un pour chacun des quatre moteurs BLDC.

Comme chaque ESC est alimenté par l'alimentation principale, le connecteur unique de cette dernière doit en quelque sorte être réparti entre les quatre ESC. Pour ce faire, une carte de répartition de l'alimentation est utilisée, figure 4.6. Cette carte divise les bornes positives et négatives de la batterie principale en quatre.

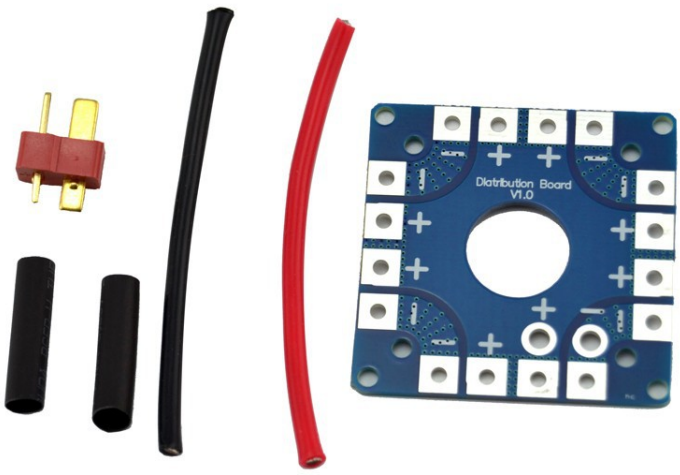


FIGURE 4.6: Carte de répartition de l'alimentation

Dans notre projet, Nous utilisons quatre ESC 12A SimonK, figure 4.7. Ces derniers possèdent les caractéristiques suivnates :

- 11A max de chargement en continu
- Des pics de courant pouvant aller jusqu'à 16A pendant une durée de 1 secondes
- Des dimensions égales à : $5cm5cm0.2cm$

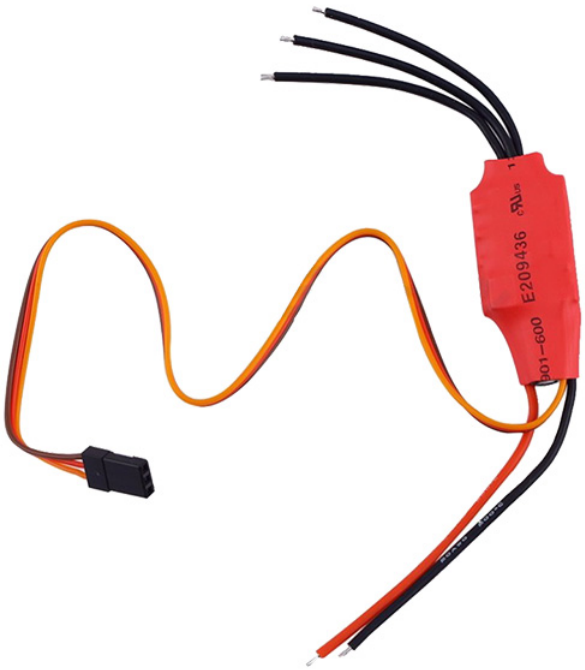


FIGURE 4.7: ESC 12A SimonK

– Une masse égale à 5.5 g.

4.3.5 Contrôleur de signaux PWM

Les ESC que nous utilisons sont commandés à l'aide de signaux PWM, mais puisque le Raspberry Pi ne possède pas de sortie PWM nous devons utiliser un circuit externe pour fournir ces signaux. Pour cela, nous utilisons le contrôleur de signaux PWM PCA9685, figure 4.8. Ce dernier communique avec le Raspberry Pi grâce au protocole de communication I2C

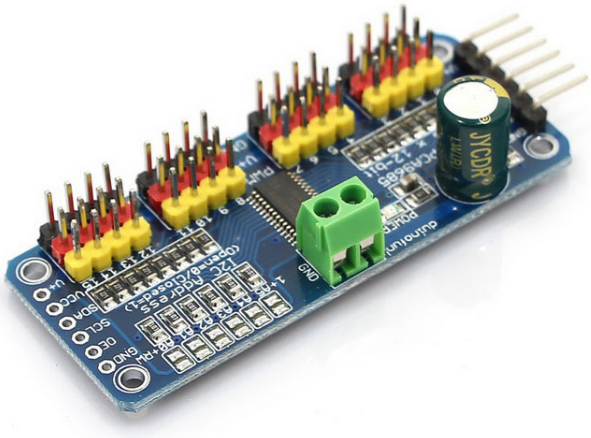


FIGURE 4.8: PCA9685

4.3.6 Les hélices

Les hélices sont les extensions de la voile, et réalise une propulsion vélique : en tournant, l'hélice mobilise la force de portance de l'air mis en mouvement et la transfère au corps sur lequel elle est attachée. Cette force de portance est une composante de la réaction du fluide, qui agit perpendiculairement au plan tangent des pales de l'hélice.

Elles doivent être adaptées à la taille du quadrirotor et aussi aux moteurs qu'on a choisi. Un quadrirotor utilise deux hélices qui tournent dans le sens horaire (CW) et deux autres hélices qui tournent dans le sens anti-horaire (CCW).

Pour notre projet, nous avons choisit des hélices de diamètre égale à 12.7 cm, figure 4.9.

FIGURE 4.9: Les Hélices

4.3.7 Le régulateur de tension 5V

L'alimentation principal du quadrirotor est une alimentation fixe de 12v. Elle est suffisante pour alimenter les quatre moteurs BLDC et leurs variateurs électroniques. Mais, la tension qu'elle fournie est trop grande pour alimenter le Raspberry Pi et les autres composants qui requièrent une tension de 5v. Pour résoudre ce problème, nous utilisons un régulateur de tension 5v à base du circuit intégré LM2576-5.0 qui peut fournir des courants allant jusqu'à 3A. Ce régulateur est un convertisseur DC-DC à découpage, ce qui lui confère un rendement relativement élevé. Sur la figure 4.10 on peut voir le circuit de régulation.

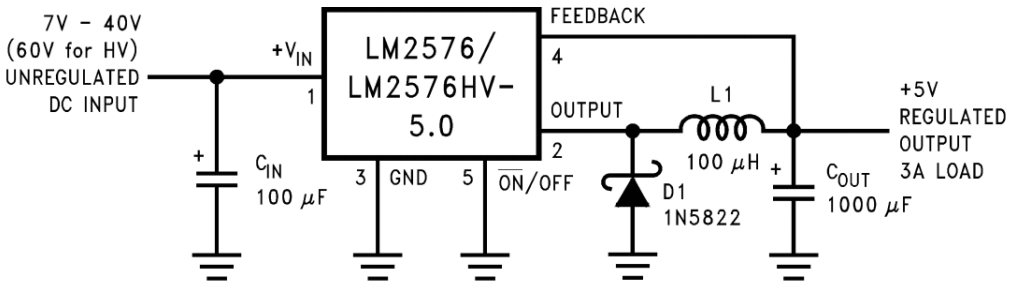


FIGURE 4.10: Schéma du circuit de régulation 5v

4.3.8 Centrale Inertielle

Une centrale inertielle ou IMU (acronyme Anglais de Inertial Measurement Unit) est un dispositif électronique qui mesure la force spécifique, la vitesse angulaire et dans

certains modèles le champ magnétique qui l'entoure en combinant des accéléromètres, des gyroscopes et des magnétomètres dans un seul et même circuit intégré.

L'avantage principal d'un centrale inertielle est la fréquence élevé d'arrivage des mesures.

Dans notre projet, nous utilisons la centrale inertielle MPU6050, 4.11 qui combine trois accéléromètres qui nous donnent l'accélération linéaire selon 3 axes orthogonaux avec trois gyroscopes qui nous donnent la vitesse angulaire autour de ses 3 axes.

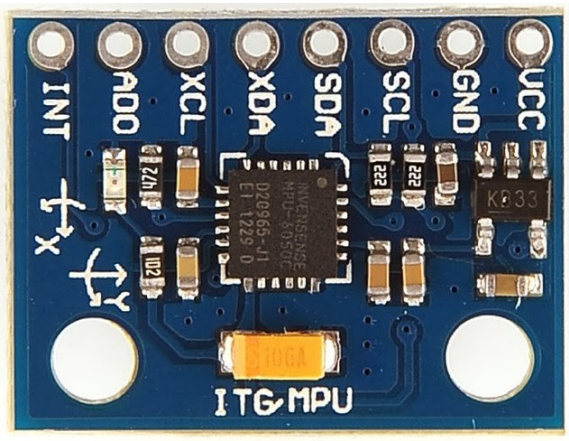


FIGURE 4.11: MPU6050

4.3.9 Sonar Ultrason

Le sonar (acronyme Anglais de SOund Navigation And Ranging) est un dispositif qui a été initialement développé pour la navigation sous-marine[88]. Il utilise des ondes sonores qu'il émet à intervalles réguliers de courtes durée et à une fréquence données, qui est supérieure à 20 kHz dans le cas d'ondes ultrasonores, pour pouvoir mesurer la distance entre un objet de l'environnement qui les reflète et lui même. La distance étant déterminée par le temps de vol (TOF, acronyme Anglais de Time Of Flight), qui est le temps mis pour parcourir un aller-retour entre le sonar et l'objet qui reflète les ondes, des ultrasons et non par leur intensité.

Pratiquement tous les matériaux réfléchissant le son peuvent être détectés, quelle que soit leur couleur. Même les objets transparents ou films minces ne posent aucun problème à un capteur à ultrasons.

Dans notre projet, nous utilisons le sonar à ultrasons HC-SR04, figure 4.12, qui possède une portée de 3 mètres et émet des ondes ultrasonores à une fréquence de 40 kHz.

FIGURE 4.12: HC-SR04

4.3.10 Récepteur WIFI

Un récepteur Wifi est un support ayant l'allure d'une clé USB classique, sauf qu'il permet d'envoyer et de recevoir des données via un réseau WIFI. Conformes aux normes IEEE 802.11n, il permet de connecter le Raspberry Pi avec l'ordinateur ou le réseaux WIFI sans fil.

Dans notre projet, nous avons choisit le récepteur WIFI TL-WN722N, figure 4.13, à gain élevé 150Mbps et une fréquence de 2.4 GHz. Ce taux d'émission et de réception relativement élevé permet d'envoyer des commandes en temps réelle et de recevoir des informations sur les capteur et même des vidéo de la caméra rapidement. Le TL-WN722N comprend une antenne externe à gain élevé de 4 dBi que l'on peut tourner et ajuster dans différentes directions pour la faire fonctionner dans des environnements variés. Elle produit une meilleure performance que l'antenne interne.

Figure 4.13: TL-WN722N

4.4 Logiciels et langage de programmation

Le cœur de notre quadrirotor se trouve être le Raspberry Pi. Il fonctionne sous Rasbian, qui est un système d'exploitation libre et gratuit fondé sur GNU/Linux/Debian et optimisé pour fonctionner sur un Raspberry Pi.

Le langage de programmation que nous avons utilisé pour réaliser toutes les manipulations et les commandes est Python[89], qui est un langage de programmation objet, multi-paradigme et multiplateformes. Il a l'avantage d'être simple et bien adapté aux débutants. Il n'en reste pas moins un langage très populaire chez les programmeurs expérimentés[90].

L'interfaçage avec le contrôleur de signaux PWM PCA9685 se fait à l'aide de la bibliothèque open-source Adafruit PWM Servo Driver Library[91]. Cette dernière utilise le protocole I2C et nous permet de définir la fréquence et la largeur en μs du signal PWM pour chaque ESC.

L'interfaçage avec la centrale inertielle MPU6050 se fait en utilisant le bus I2C du Raspberry Pi. Nous accédons aux mesures de l'accéléromètre et du gyroscope en faisant une lecture dans des registres de 8 bit (deux registres pour chaque mesure, i.e. 16 bits par mesure).

L'interfaçage avec le sonar ultrason HC-SR05 se fait à l'aide de la bibliothèque RPi.GPIO[92]. Cette dernière en charge l'interfaçace avec les broches d'entrée/sortie. Elle nous permet de les mettre soit en mode entrée, soit en mode sortie. Dans ce dernier mode, nous pouvons

mettre les sorties à 1, i.e. 3.3v, ou à 0, 0v. Le HC-SR05 possède deux broches, TRIG et ECHO, qui nous servent à prendre des mesures. La première, TRIG, est la gâchette qui lorsqu'elle reçoit un front montant enclenche l'envoi de huit ondes ultra-sonores à une fréquence de 40kHz qui seront reflétés par les objets de l'environnement et captés par le récepteur. Une fois les ondes captées, la broche ECHO se met au niveau haut (5v) pendant une durée égale à celle prise par les ondes ultra-sonores pour sortir de l'émetteur et revenir vers le récepteur.

4.5 Mesures

Dans la réalité, toute mesure est entaché de bruits et de biais. C'est pour cela qu'avant d'utiliser toute mesure, il est nécessaire d'effecteur à un pré-traitement pour les rendre utilisables.

Dans notre cas, nous utilisons trois capteurs : un accéléromètre, un gyroscope et un sonar ultrason, et tout les trois possèdent des caractéristiques bien propres que nous devons prendre en compte avant de mettre en œuvre des algorithmes de commande et d'estimation.

4.5.1 Accéléromètre

Le MPU6050 contient un accéléromètre qui mesure les accélérations suivant trois axes mutuellement orthogonaux. Son modèle de mesure est défini comme suit[93, 94, 95] :

$$\begin{cases} a = a_r + g + b_a + \eta_a \\ \dot{b}_a = \eta_{b_a} \end{cases} \tag{4.3}$$

Où :

a_r, est l'accélération propre du système,

g, est l'accélération due à la gravité,

b_a, est un biais,

η_a, η_b, sont des bruit blancs gaussiens à moyenne nulle.

Comme nous pouvons le voir sur la figure 4.14, les mesures de l'accéléromètre sont bruitées et biaisées, malgré qu'elles aient étaient prises lorsque le système était immobile. Pour avoir des mesures plus précises, nous devons les filtrer et enlever le biais.

Nous commençons tout d'abord par identifier le biais que nous supposons constant. Nous prenons les mesures effectuées et nous calculons leur moyenne. Comme le bruit des capteurs est à moyenne nulle, la moyenne des mesures sera égale au biais. Les résultats obtenus sont résumés dans le tableau 1.5.

Nous utilisons ensuite sur ces mesures, après avoir enlevé le biais et l'effet de la gravité, un filtre médian pour réduire les pics et le bruit présents dans les mesures sans trop altérer le signal et pouvoir ainsi les intégrer pour estimer les vitesses de translation. C'est un filtre numérique souvent utilisé pour la réduction de bruit. L'idée principale de ce filtre est de remplacer chaque entrée par la valeur médiane de son voisinage. Comme nous pouvons le constater sur la figure 4.15, il y a bien une réduction dans le bruit et le nombre de pics.

Nous passons les mesures brutes sans biais par un filtre passe-bas de Butterworth d'ordre 4 de fréquence de coupure égale à 5 Hz pour pouvoir obtenir les composantes

du vecteur de gravité dans le repère local lié au système et estimer ainsi l'orientation de ce dernier. Comme nous pouvons le voir sur la figure 4.16, il ne reste du signal que sa composante continue, qui représente la gravité, et de faibles perturbations.

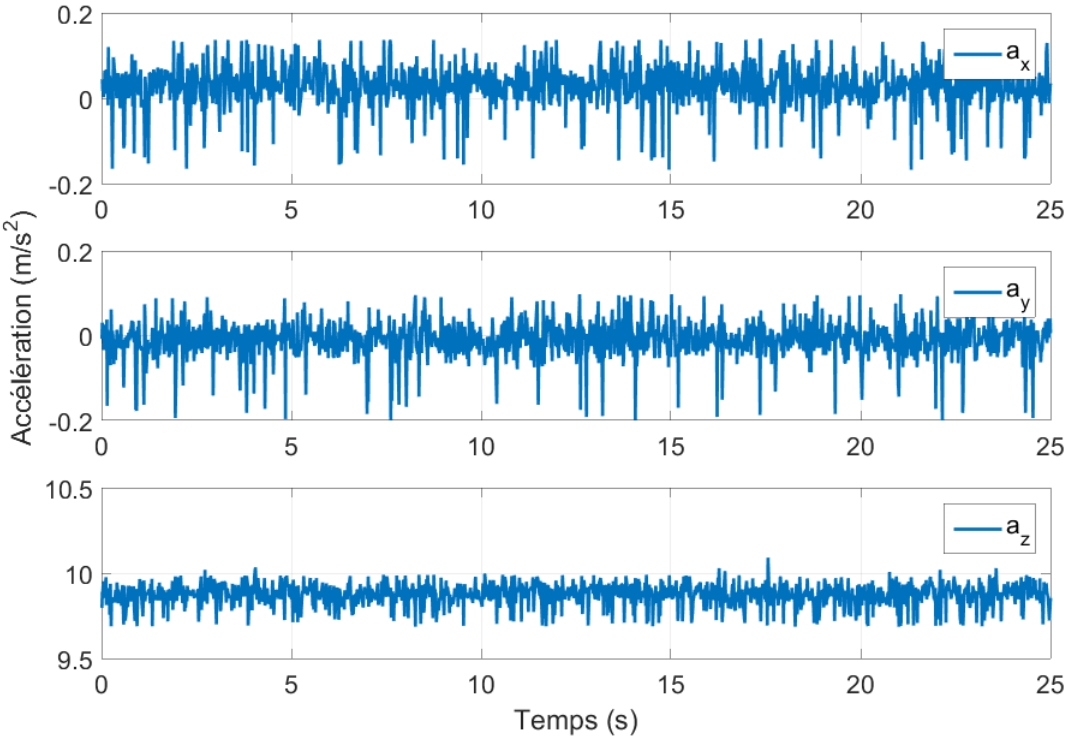


FIGURE 4.14: Mesures brutes de l'accéléromètre

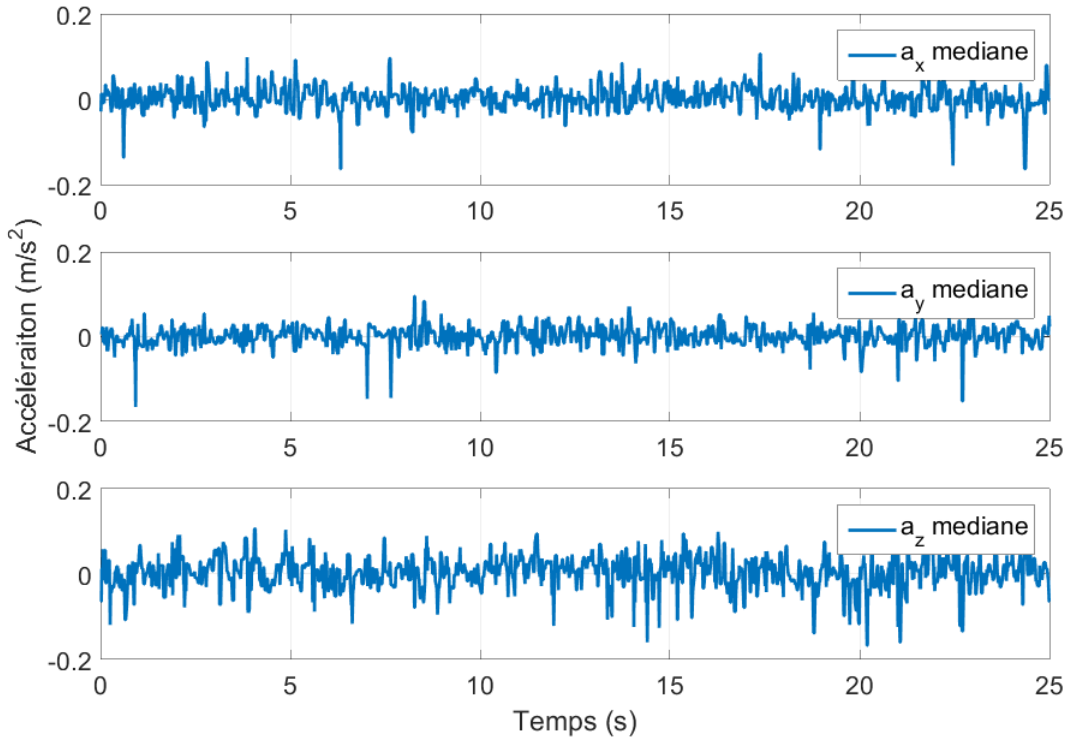


FIGURE 4.15: Mesures filtrées de l'accéléromètre par le filtre médian

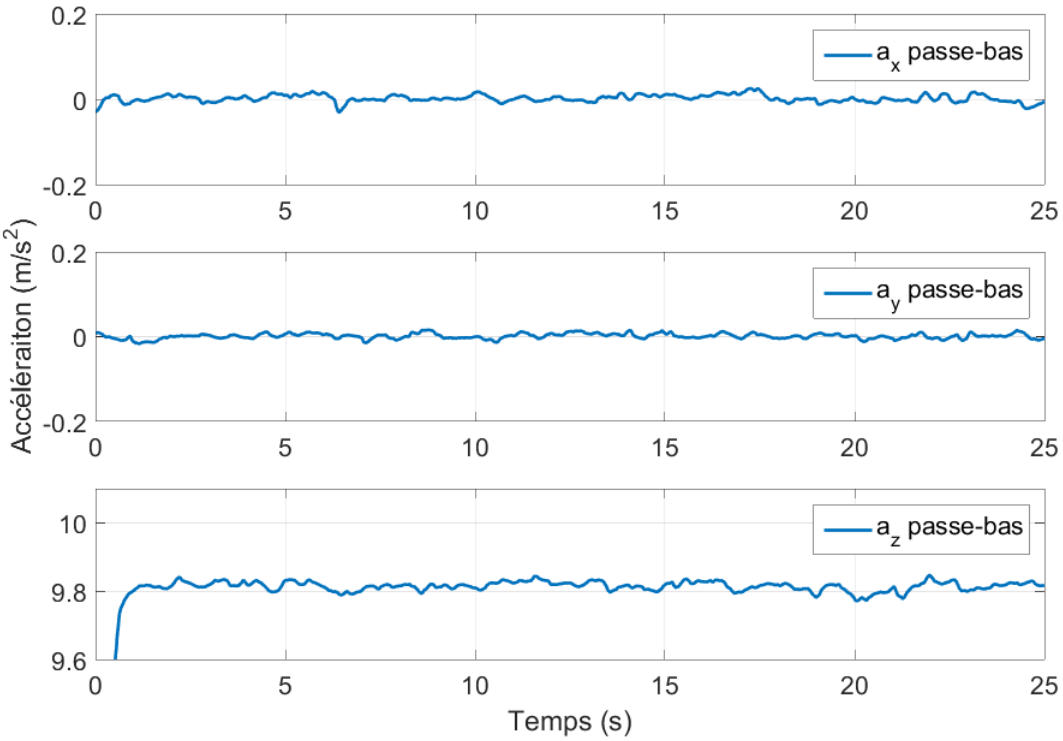


FIGURE 4.16: Mesures filtrées de l'accéléromètre par le filtre passe-bas

4.5.2 Gyroscope

Le MPU6050 contient un gyroscope qui mesure les vitesse angulaires autours des trois axes X, Y et Z. Son modèle de mesure est défini comme suit[93, 94, 95] :

$$\begin{cases} \omega = \omega_r + b_\omega + \eta_\omega \\ \dot{b}_\omega = \eta_{b_\omega} \end{cases} \tag{4.4}$$

Où :

ω_r, est la vitesse angulaire du système,

b_ω, est un biais,

η_ω, η_b, sont des bruit blancs gaussiens à moyenne nulle.

Nous commençons tout d'abord par identifier le biais que nous supposons constant. Nous prenons les mesures effectuées et nous calculons leur moyenne. Comme le bruit des capteurs est à moyenne nulle, la moyenne des mesures sera égale au biais. Les résultats obtenus sont résumés dans le tableau 1.5.

Nous utilisons ensuite sur ces mesures, après avoir enlevé le biais, un filtre médian pour réduire les pics et le bruit présents dans les mesures sans trop altérer le signal. Comme nous pouvons le constater sur la figure 4.18, il y a une grande réduction dans le bruit et le nombre de pics.

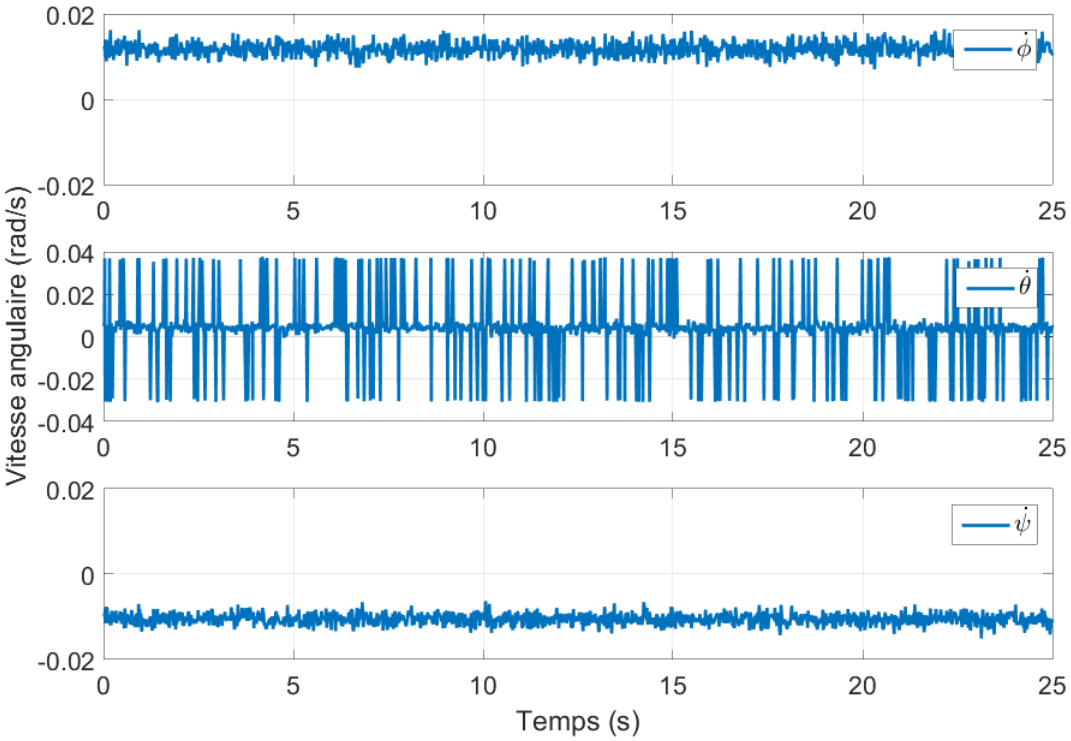


FIGURE 4.17: Mesures brutes du gyroscope

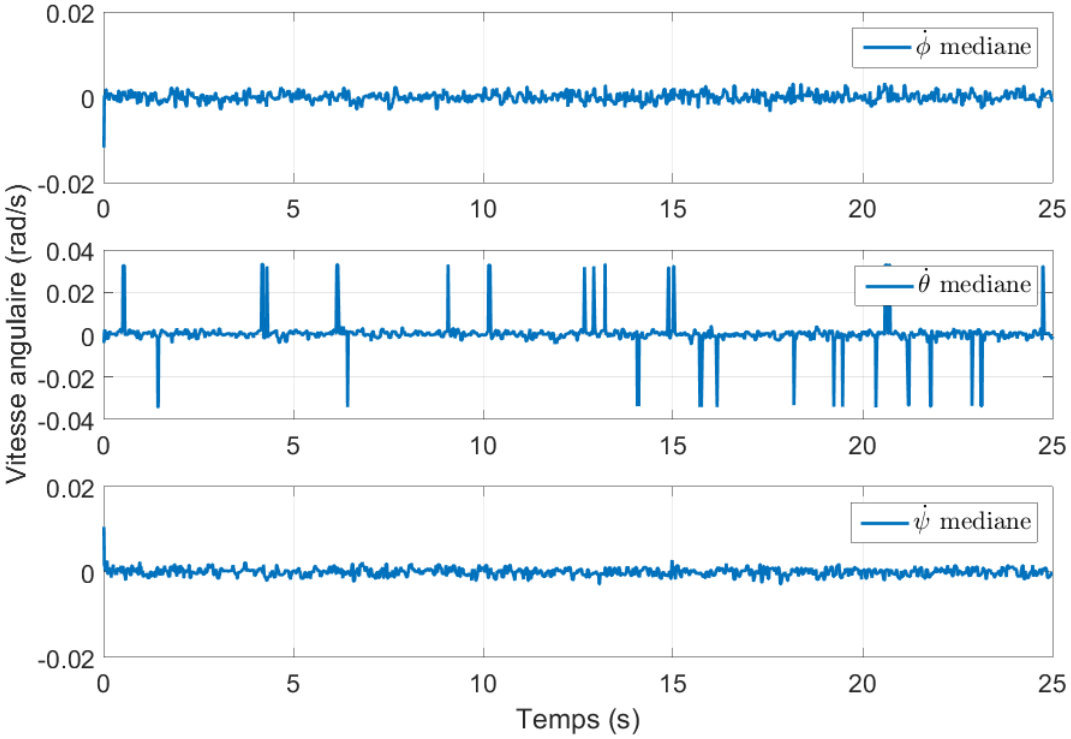


FIGURE 4.18: Mesures filtrés du gyroscope par le filtre médian

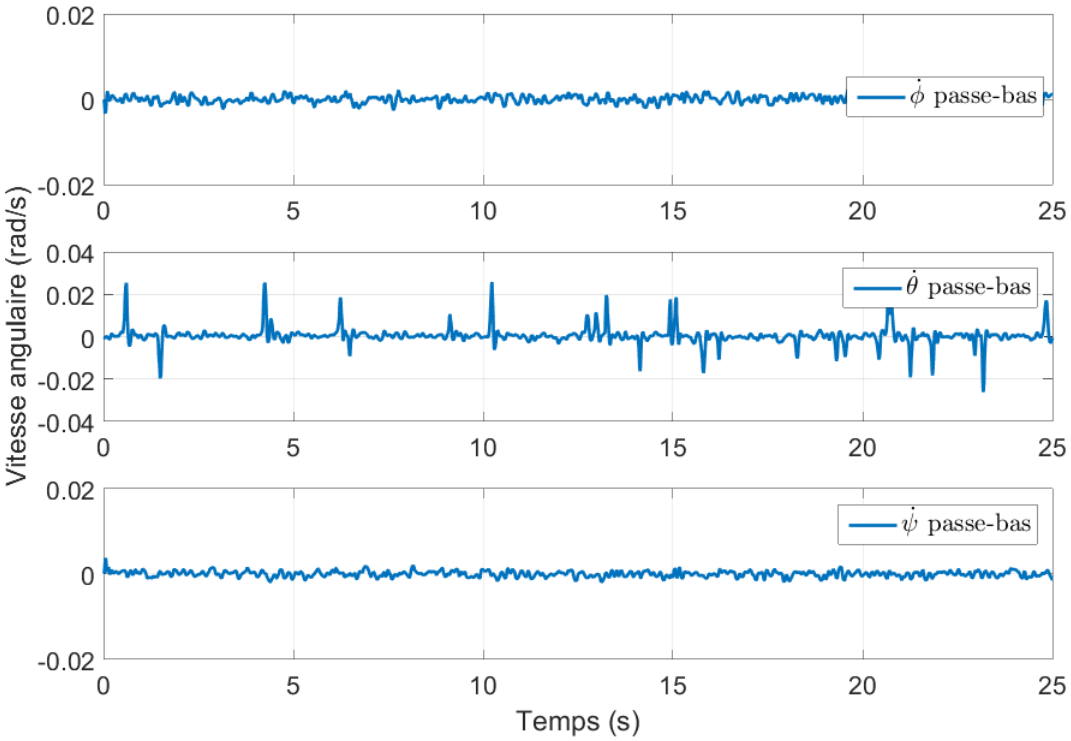


FIGURE 4.19: Mesures filtrées du gyroscope par le filtre passe-bas

Enfin, nous passons les mesures filtrées par un filtre passe-bas de Butterworth d'ordre 4 de fréquence de coupure égale à 25 Hz pour réduire d'avantage le bruit et les pics. Comme nous pouvons le constater sur la figure 4.19, il y a bien une atténuation des bruits et des pics.

4.5.3 Sonar ultrason

En utilisant le temps mis par les ondes ultra-sonores pour sortir de l'émetteur et revenir vers le récepteur et la vitesse du son nous pouvons calculer la distance du sonar ultrason jusqu'à l'objet qui a reflété les ondes ultra-sonores, qui est égale à la moitié de la distance traversée, en utilisant l'équation suivante :

$$d = v_{son}\frac{\Delta t}{2} \tag{4.5}$$

Où :

$v_{son} = 340m/s$, est la vitesse du son,

Δt, la durée mise par les ondes ultra-sonores pour faire un aller-retour.

Mais comme tout mesure est entachées de bruits et d'erreurs, cette dernière devient :

$$d = v_{son}\frac{\Delta t}{2} + \eta_d \tag{4.6}$$

Où :

η_d, est un bruit blanc gaussien de moyenne nulle.

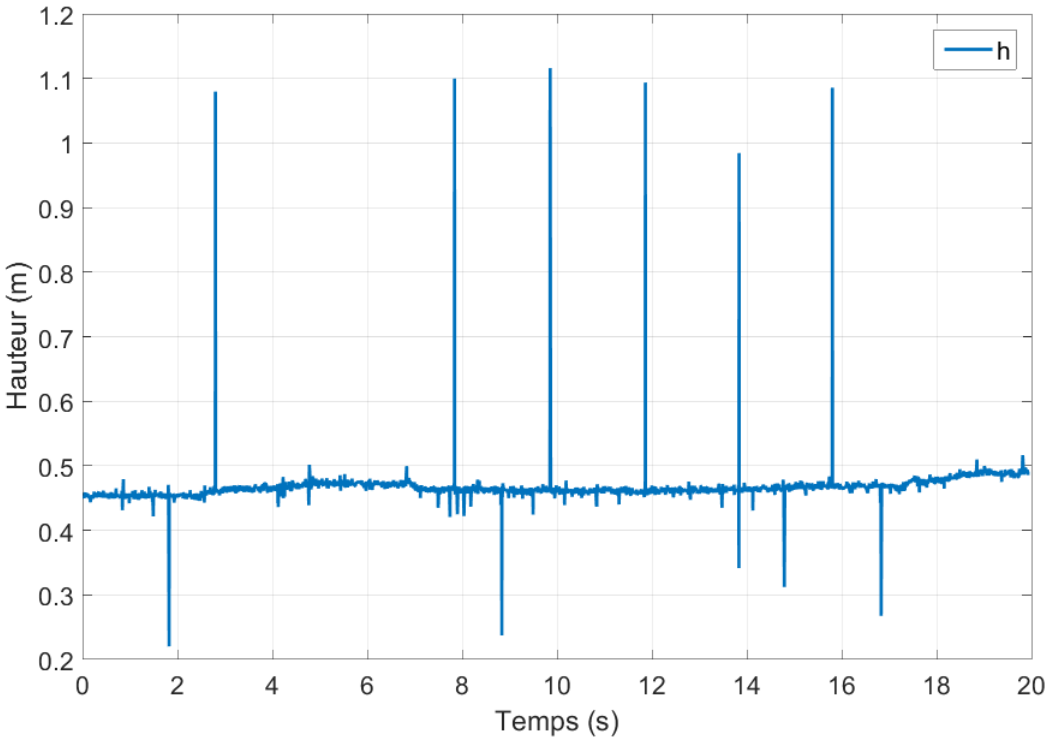


FIGURE 4.20: Mesures brutes du sonar

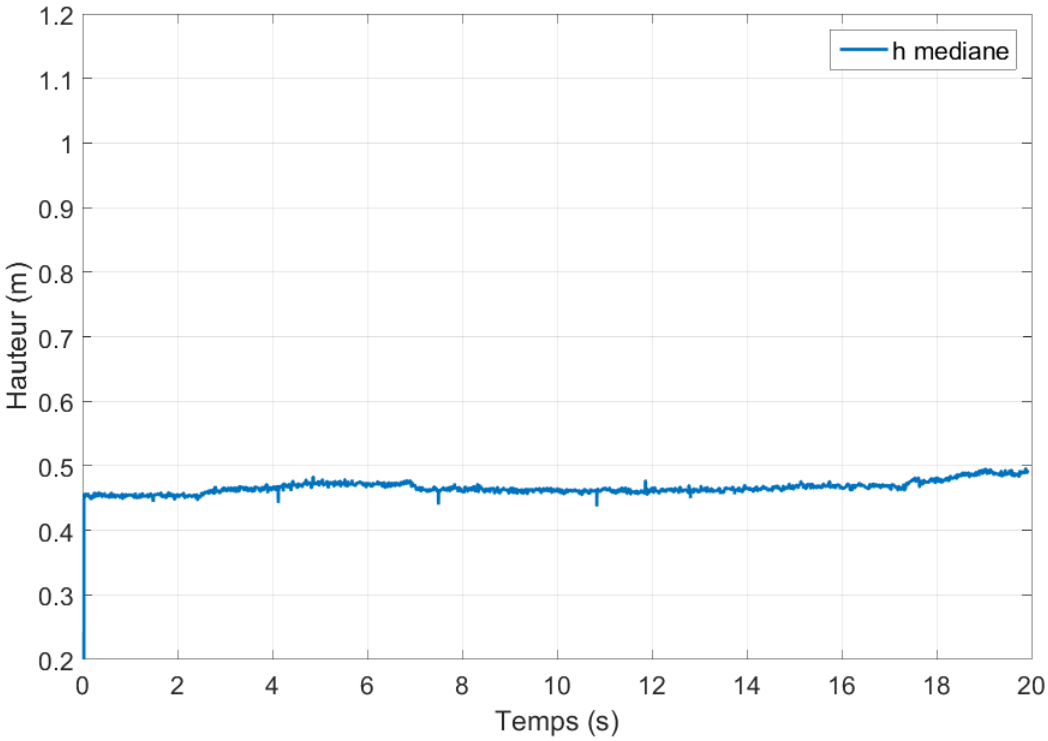


FIGURE 4.21: Mesures filtrés du sonar par le filtre médian

Comme nous pouvons le voir sur la figure 4.20, le signal mesuré présente des pics ainsi que du bruit.

Pour les atténuer nous utilisons un filtre médian. Comme nous pouvons le constater sur la figure 4.21, les pics ont disparus et le bruit a été atténué.

4.6 Identification des paramètres

Pour pouvoir implémenter des algorithmes de commande et d'observation sur le quadrirotor nous devons connaître les différents paramètres qui régissent sa dynamique.

Les paramètres que nous devons identifier sont :
- La masse du quadrirotor
- Les longueurs L_1 et L_2, figure 4.2
- Le moment d'inertie du quadrirotor
- Les coefficients de portance et de trainée

4.6.1 La masse

A l'aide d'une balance numérique, nous avons pesé le quadrirotor et obtenu le résultat suivant :

$$m = 506g \tag{4.7}$$

4.6.2 Les longueurs L_1 et L_2

A l'aide d'une règle, nous avons mesuré les deux longueurs L_1 et L_2 et avons obtenu les mesures suivantes :

$$L_1 = 20.1cm \tag{4.8}$$

$$L_2 = 15.6cm \tag{4.9}$$

4.6.3 Le moment d'inertie

Le moment d'inertie d'un solide décrit le comportement dynamique d'un corps en rotation autour d'un axe défini. Il a le même rôle dans la dynamique de rotation que la masse dans la dynamique de translation. Le moment d'inertie dépend de la distribution des masses du solide par rapport à un axe de rotation. Pour un solide qui subit des mouvements de rotation dans l'espace, le moment d'inertie peut être décrit par une matrice symétrique de dimensions 33.

$$J = \begin{bmatrix} J_{XX} & J_{XY} & J_{XZ} \\ J_{YX} & J_{YY} & J_{YZ} \\ J_{ZX} & J_{ZY} & J_{ZZ} \end{bmatrix} \tag{4.10}$$

Pour le même solide, différents axes de rotation peuvent avoir des moments d'inertie différents et on peut on trouver une infinité.

Généralement la matrice d'inertie est calculée par des méthode mathématique à l'aide de son expression mais pour des solides ayant des formes géométriques compliquées, il est préférable d'utiliser un logiciel qui calcule ce dernier directement ou bien de procéder par un processus d'identification.

Dans notre cas, puisque nous n'avions pas le matériel suffisant pour identifier le moment d'inertie et que la géométrie du quadrirotor est assez compliquée pour être modélisée correctement avec des logiciels, nous n'avons pas pu identifier son moment d'inertie.

4.6.4 La poussée

La poussée est la force aérodynamique qui est produite dans le sens des hélices. Il est nécessaire pour surmonter les forces de traînée et de poids, et ainsi soutenir le vol vers l'avant du quadrirotor.

La poussée d'un rotor est proportionnelle au carré de la vitesse de rotation de ce dernier. Le facteur de proportionnalité est "b", le facteur de poussée :

$$T = b\omega^2 \tag{4.11}$$

Dans notre cas, puisque nous commandons les moteurs directement à l'aide de signaux PWM et que l'on a pas en notre possession de capteurs de vitesse angulaire assez petits et précis pour nos moteurs, nous nous sommes intéressés à la relation entre ces signaux et la poussée. Nous modélisons cette relation comme un polynôme du premier ordre comme suit :

$$T = a \cdot u + b \tag{4.12}$$

Où u est la largeur d'impulsion PWM en μs.

Pour l'identification, nous avons alourdi le quadrirotor pour l'empêcher de s'envoler et nous l'avons mis sur une balance que nous avons taré, comme on peut le voir sur la figure 4.22.

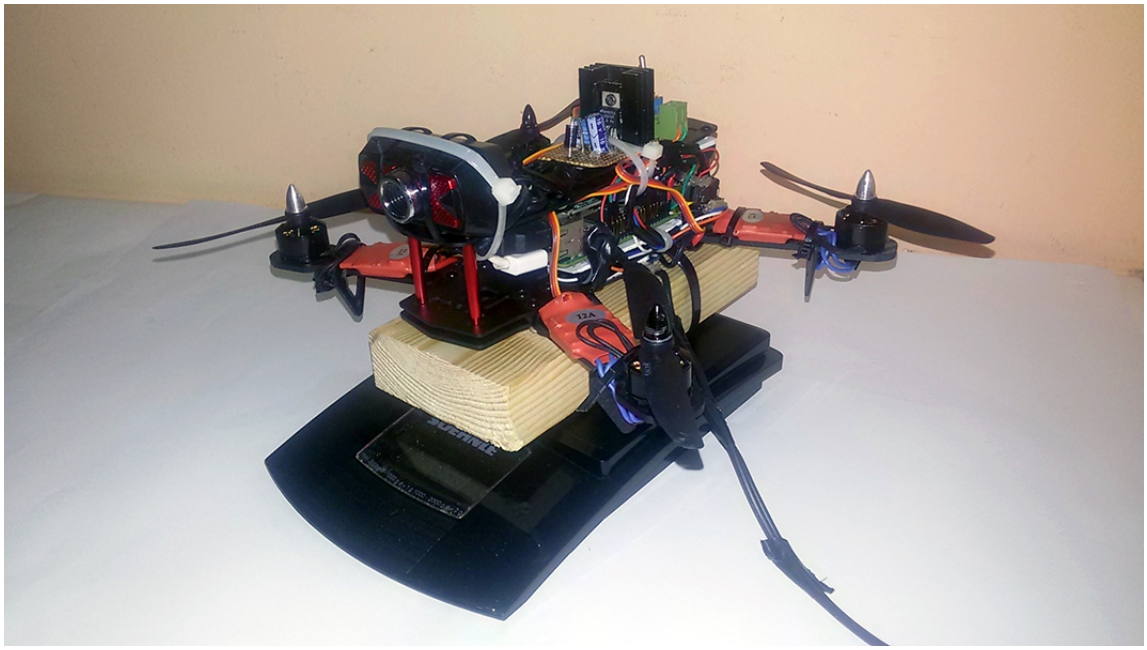


FIGURE 4.22: Montage réalisé pour l'identification de la poussée

Puis, nous avons fait tourner les quatre rotors à la même vitesse et nous avons à chaque fois pris note de la valeur affichée par la balance et la largeur d'impulsion (PWM) en μs utilisée pour commander les moteurs.

Les résultats obtenus sont résumés dans le tableau 1.6 et la figure 4.23.

4.6.5 Le moment de trainée

Le moment de trainée est le moment généré par la rotation d'une hélice et qui fait tourner le quadrirotor dans le sens opposé à la rotation de cette dernière.

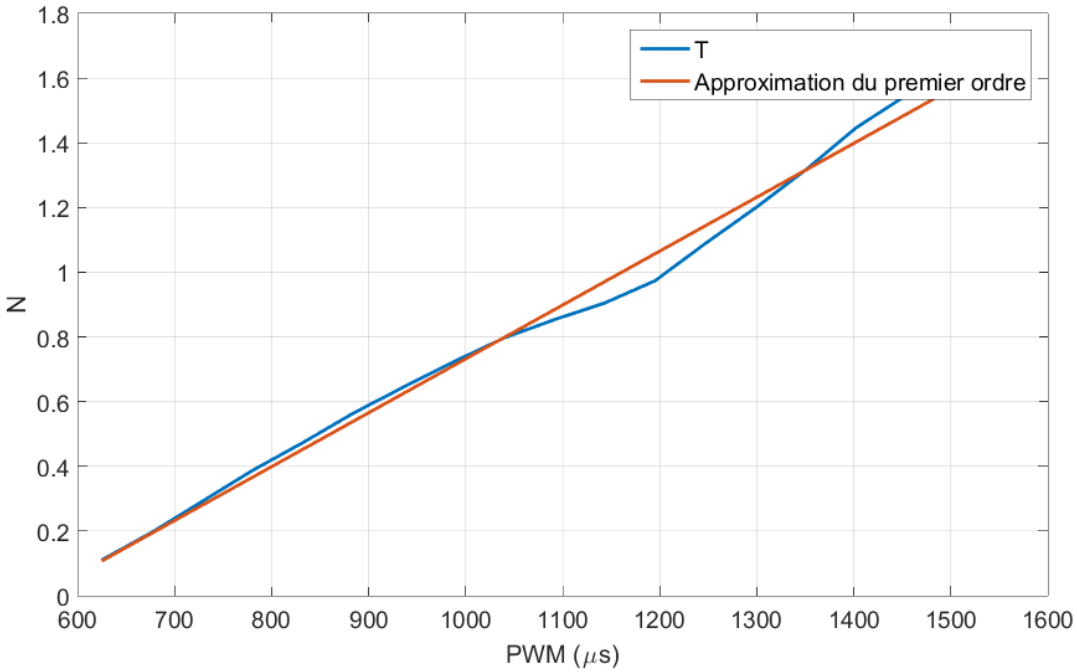


FIGURE 4.23: Relation entre la poussée et le PWM

Ce moment est proportionnel au carré de la vitesse de rotation du rotor. Le facteur de proportionnalité est "k", le coefficient de trainée :

$$\tau = k \cdot \omega^2 \tag{4.13}$$

Dans notre cas, puisque nous commandons les moteurs directement à l'aide de signaux PWM et que l'on a pas en notre possession de capteurs de vitesse angulaire assez petits et précis pour nos moteurs, nous nous sommes intéressés à la relation entre ces signaux et le moment de trainée. Nous modélisions cette relation comme un polynôme du premier ordre comme suit :

$$\tau = c \cdot u + d \tag{4.14}$$

Où u est la largeur d'impulsion PWM en μs.

Pour l'identification, nous avons monté l'un des moteurs verticalement sur axe qui peut effectuer des mouvements de rotation autour d'un axe qui est perpendiculaire au sol et nous avons monté une balance verticalement et nous l'avons taré, comme on peut le voir sur la figure 4.24.

Puis, nous avons fait tourner le moteur dans le sens horaire et de ce fait l'axe à tourné dans le sens opposé et à appliqué un moment sur la balance. Nous avons noté à chaque fois la valeur affichée par la balance et la largeur d'impulsion (PWM) en μs utilisée pour commander le moteur.

Les résultats obtenus sont résumés dans le tableau 1.7 et la figure 4.25.

4.7 Implémentation des commandes

Dans cette partie nous allons nous intéresser à la commande du quadrirotor. Nous commençons par commander son orientation suivant à chaque fois l'un des axes du repère

FIGURE 4.24: Montage réalisé pour l'identification du moment de trainée

lié à ce dernier, puis nous commandons son altitude tout en essayant de le stabiliser dans les airs.

4.7.1 Commande de l'angle de lacet ψ

L'angle de lacet, ψ, ou l'angle de rotation autour de l'axe Z_1 lié au quadrirotor est très important pour notre système car il représente l'orientation de l'avant du quadrirotor sur lequel est fixée la caméra.

Pour pouvoir limiter les mouvements du quadrirotor à un seul degré de liberté, qui est la rotation autour de l'axe Z_1, nous avons réalisé le montage présenté sur la figure 4.26.

Il s'agit d'une plateforme sur laquelle se trouve un axe qui peut tourner librement. Nous avons attaché sur ce dernier un support sur lequel nous avons posé et attaché le quadrirotor.

Pour le modèle de ce système, nous prenons seulement l'équation qui régit la dynamique de rotation autour de l'axe Z_1 depuis l'équation (1.6.3) et nous annulons tout les autres états car le quadrirotor étant fixé, il ne peut se mouvoir suivant les autres axes.

On obtient donc l'équation dynamique suivante :

$$\ddot{\psi} = \frac{\tau_\psi}{J_z} + \rho_{d\psi} \tag{4.15}$$

Où :

$\tau_\psi = K(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)$, est la commande appliqué,

$\rho_{d\psi}$, est une variable qui regroupe les perturbations et les erreurs de modélisation.

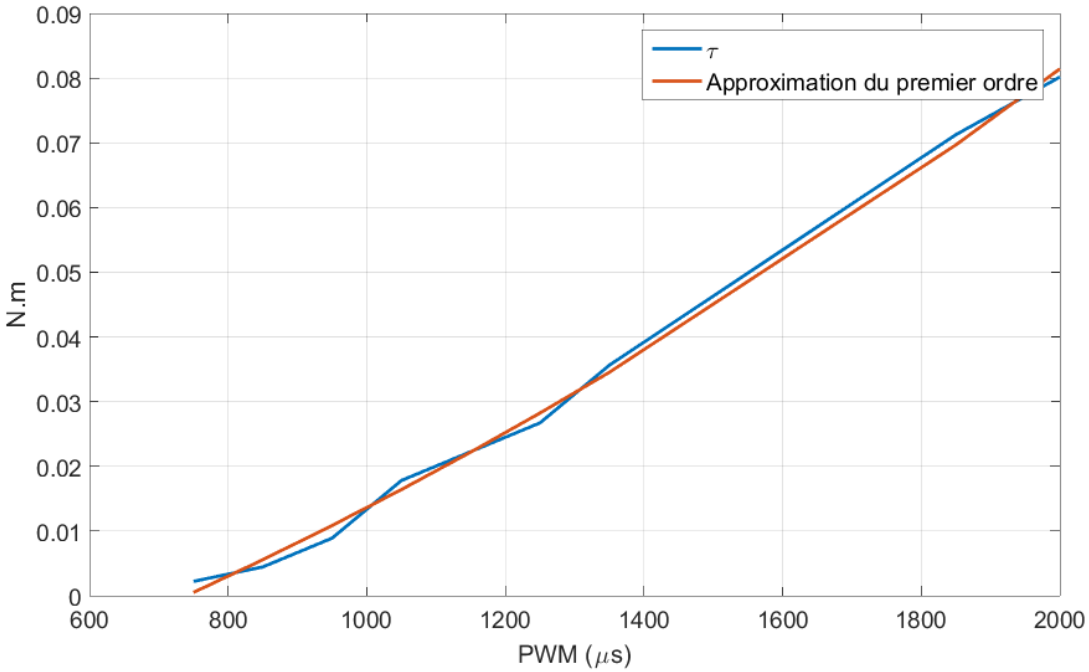


FIGURE 4.25: Relation entre le moment de trainée et le PWM

Commande par PID

Nous commençons par implémenter une commande PID classique. Pour cela nous définissons l'erreur sur l'angle de lacet comme suit :

$$e_\psi = \psi_d - \psi \tag{4.16}$$

Où ψ_d, est l'angle de lacet désiré.
Puis, nous définissons la commande comme suit :

$$\tau_\phi = K_P e_\psi + K_I \int e_\psi dt + K_D \frac{de_\psi}{dt} \tag{4.17}$$

Avec :

K_P, le gain de l'action proportionnelle,

K_I, le gain de l'action intégrale,

K_D, le gain de l'action dérivée.

Mais puisque la commande va être appliqué discrètement la commande devient :

$$\tau_{\phi_{k+1}} = K_P e_{\psi_k} + K_I \sum_{i=1}^{k} e_{\psi_i} \Delta t + K_D \frac{e_{\psi_k} - e_{\psi_{k-1}}}{\Delta t} \tag{4.18}$$

Avec Δt, le temps d'échantillonnage que l'on suppose fixe.

Commande par mode glissant

Nous implémentons ensuite une commande par mode glissant. Pour cela nous définissons l'erreur sur l'angle de lacet comme suit :

FIGURE 4.26: Montage réalisé pour la commande de l'angle de lacet ψ

$$e_\psi = \psi - \psi_d \tag{4.19}$$

Où ψ_d, est l'angle de lacet désiré.

Puis, nous utilisons la commande synthétisée dans l'équation (1.59), avec $U_4 = \tau_\psi$, et qui est définie comme suit :

$$\tau_\psi = J_z(\ddot{\psi}_d - \lambda(\dot{\psi} - \dot{\psi}_d) - Ksign(S) - QS) \tag{4.20}$$

Avec :

$S = \dot{\psi} + \lambda(\psi - \psi_d)$, la surface de glissement,

λ, le paramètre de réglage de la surface,

K, le gain de la fonction $sign(.)$,

Q, le gain de l'action proportionnelle.

Mais puisque dans notre cas nous une référence constante, i.e. $\ddot{\psi} = \dot{\psi} = 0$, la commande devient :

$$\tau_\psi = J_z(-\lambda\dot{\psi} - Ksign(S) - QS) \tag{4.21}$$

4.7.2 Commande de l'angle de roulis ϕ

L'angle de roulis, ψ, ou l'angle de rotation autour de l'axe X_1 lié au quadrirotor est très important pour notre système car une inclinaison de ce dernier permet au quadrirotor de se mouvoir suivant ce même axe.

Pour pouvoir limiter les mouvements du quadrirotor à un seul degré de liberté, qui est la rotation autour de l'axe X_1, nous avons réalisé le montage présenté sur la figure 4.27.

Il s'agit d'une plateforme sur laquelle se trouve un axe qui peut tourner librement. Nous avons attaché sur ce dernier un support sur lequel nous avons posé et attaché le quadrirotor.

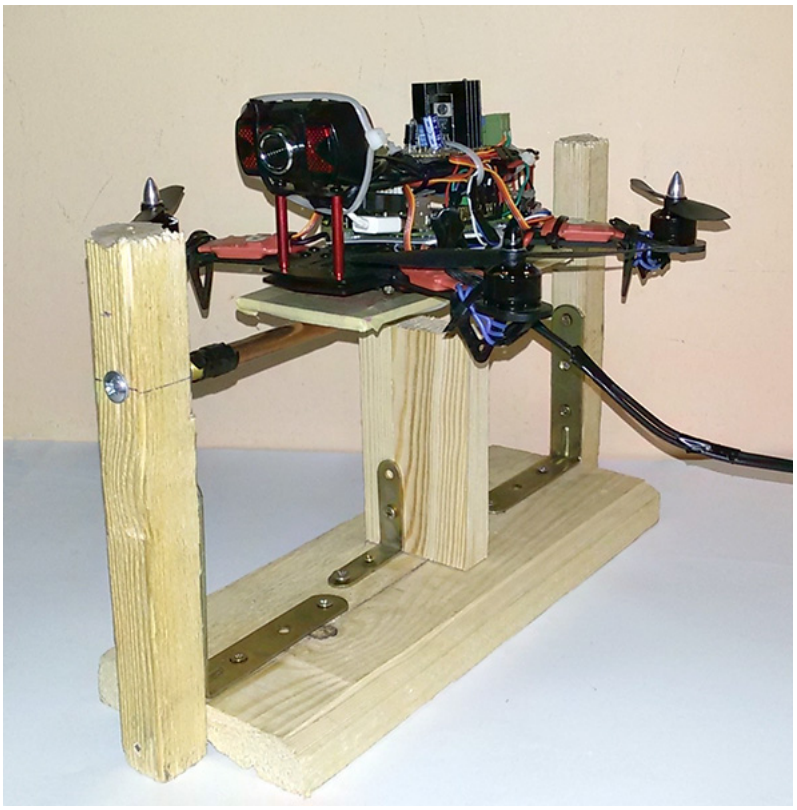


FIGURE 4.27: Montage réalisé pour la commande de l'angle de roulis ϕ

La dynamique du quadrirotor sur ce montage est différente de celle exprimée dans les équations (1.33) et (1.6.3).

Le schéma de la figure 4.28 résume les différentes forces qui régissent sa dynamique.

Pour la modéliser, nous utilisons la seconde loi dynamique de Newton :

$$J\ddot{\phi} = \sum \Gamma_{ext} \tag{4.22}$$

Avec :

$J = J_x + mh_0^2$, le moment d'inertie du quadrirotor autour de l'axe de rotation du montage.

h_0, la distance suivante l'axe Z_1 entre l'axe de rotation du montage et le centre de gravité du quadrirotor.

m, la masse du quadrirotor.

En remplaçant l'expression des moments externes appliqués au système et après simplification nous obtenons l'équation différentielle suivante :

$$\ddot{\phi} = 4\frac{m_0 g h_0}{J} sin(\phi) + \frac{\tau_\phi}{J} + \rho_{d\phi} \tag{4.23}$$

Avec :

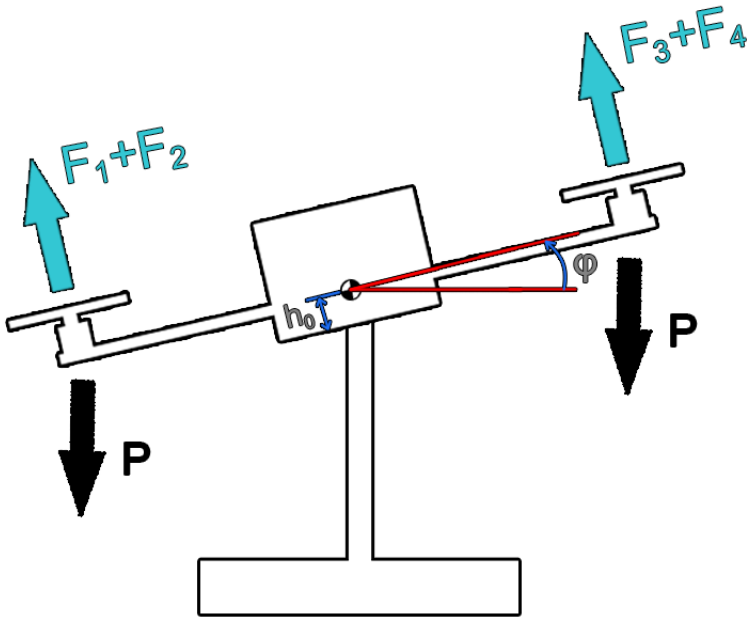


FIGURE 4.28: Schéma simplifiée de la dynamique du montage

$\tau_\phi = \frac{L_1}{2}(-F_1 - F_2 + F_3 + F_4)$, la commande appliquée.

$F_i = b \cdot \omega_i^2$ pour $i = 1, 2, 3, 4$.

m_0, la masse de l'un des quatre moteurs.

Pour des raisons de simplification, nous posons $F_1 = F_2$ et $F_3 = F_4$, ce qui nous permet d'écrire :

$$\tau_\phi = L_1(-F_1 + F_4) = L_1 \cdot \Delta F \tag{4.24}$$

Commande par PID

Nous commençons par implémenter une commande PID classique. Pour cela nous définissons l'erreur sur l'angle de roulis comme suit :

$$e_\phi = \phi_d - \phi \tag{4.25}$$

Où ϕ_d, est l'angle de roulis désiré.
Puis, nous définissons la commande comme suit :

$$\tau_\phi = K_P e_\phi + K_I \int e_\phi dt + K_D \frac{de_\phi}{dt} \tag{4.26}$$

Avec :

K_P, le gain de l'action proportionnelle,

K_I, le gain de l'action intégrale,

K_D, le gain de l'action dérivée.

Mais puisque la commande va être appliquée discrètement la commande devient :

$$\tau_{\phi_{k+1}} = K_P e_{\phi_k} + K_I \sum_{i=1}^{k} e_{\phi_i} \Delta t + K_D \frac{e_{\phi_k} - e_{\phi_{k-1}}}{\Delta t} \tag{4.27}$$

Avec Δt, le temps d'échantillonnage que l'on suppose fixe.

Commande par mode glissant

Nous implémentons ensuite une commande par mode glissant. Pour cela nous définissons l'erreur sur l'angle de roulis comme suit :

$$e_\phi = \phi - \phi_d \tag{4.28}$$

Où ϕ_d, est l'angle de roulis désiré.
Puis, nous utilisons la commande synthétisée dans l'équation (1.59), avec $U_4 = \tau_\phi$, et qui est définie comme suit :

$$\tau_\phi = J_x(\ddot{\phi_d} - \lambda(\dot{\phi} - \dot{\phi_d}) - Ksign(S) - QS) \tag{4.29}$$

Avec :

$S = \dot{\phi} + \lambda(\phi - \phi_d)$, la surface de glissement,

λ, le paramètre de réglage de la surface,

K, le gain de la fonction $sign(.)$,

Q, le gain de l'action proportionnelle.

Mais puisque dans notre cas nous une référence constante, i.e. $\ddot{\phi} = \dot{\phi} = 0$, la commande devient :

$$\tau_\phi = J_x(-\lambda\dot{\phi} - Ksign(S) - QS) \tag{4.30}$$

4.7.3 Commande de l'altitude Z

L'altitude, z, est très importante pour tout système aérien, car elle représente la hauteur à laquelle il vol. Pour notre système, la montée et la descente se font en augmentant, diminuant, la vitesse de rotations des quatre rotors respectivement. Mais cela ne suffit pas, car comme tout système physique réel, notre quadrirotor n'est pas parfait. Son centre de gravité ne coïncide pas avec son centre géométrique et cela influe sur sont comportement dans les airs. C'est pour cela que devons réguler les angles de roulis, de tangage et de lacet pour qu'il reste stable autour du point d'origine, c'est-à-dire, $\phi = 0$, $\theta = 0$ et $\psi = 0$.

Le schéma de la figure 4.29 présente de manière générale l'algorithme de la commande de l'altitude.

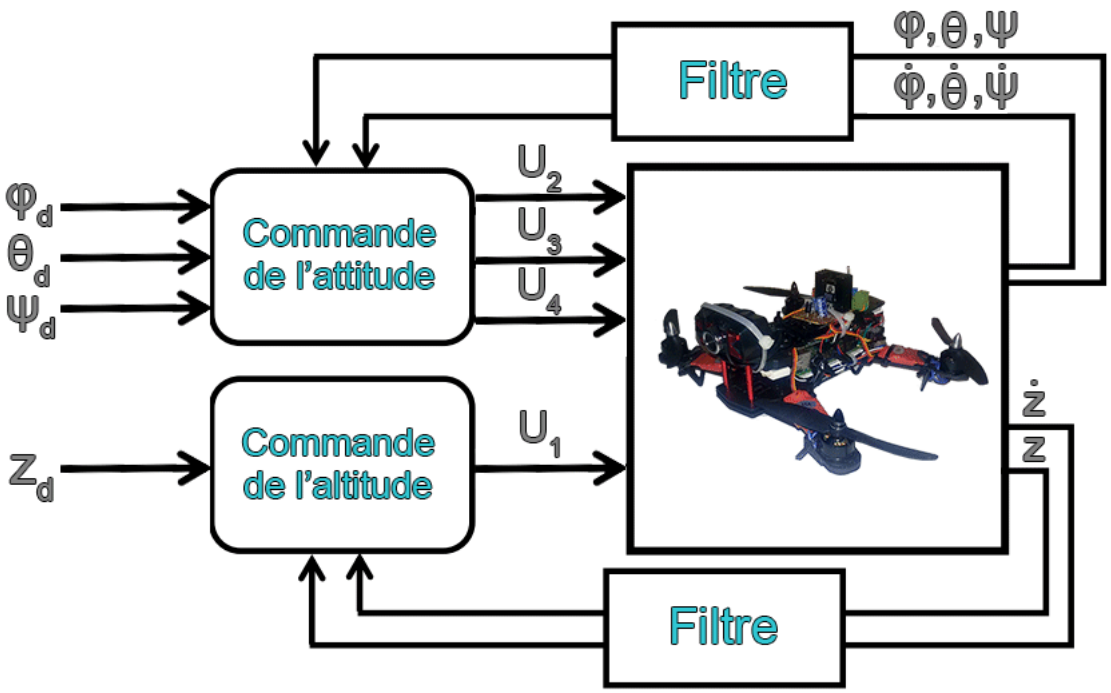


FIGURE 4.29: Schéma général de la commande de l'altitude

Pour la modélisation de la dynamique du système, nous choisissons le vecteur d'état suivant :

$$X = \begin{bmatrix} z \\ \dot{z} \\ \phi \\ \theta \\ \psi \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{4.31}$$

Et nous obtenons, en prenant l'équation (1.37) et ne prenant pas en compte les équations des positions horizontales, x et y, nous obtenons le système d'équations suivant :

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \frac{U_1}{m}(c_{x_3}c_{x_4}) - g + \rho_{dz} \\ \dot{x}_3 = x_6 \\ \dot{x}_4 = x_7 \\ \dot{x}_5 = x_8 \\ \dot{x}_6 = (\frac{J_y - J_z}{J_x})x_7x_8 + \frac{U_2}{J_x} + \rho_{d\phi} \\ \dot{x}_7 = (\frac{J_z - J_x}{J_y})x_6x_8 + \frac{U_3}{J_y} + \rho_{d\theta} \\ \dot{x}_8 = (\frac{J_x - J_y}{J_z})x_6x_7 + \frac{U_4}{J_z} + \rho_{d\psi} \end{cases} \tag{4.32}$$

Comme commande nous utilisons celle qui a été synthétisée au premier chapitre et dans l'expression est donnée par :

$$\begin{cases} U_1 = m(v_z + g) \\ U_2 = J_x v_\phi - (J_y - J_z)x_8x_9 \\ U_3 = J_y v_\theta - (J_z - J_x)x_7x_9 \\ U_4 = J_z v_\psi - (J_x - J_y)x_7x_8 \end{cases} \tag{4.33}$$

Avec :

$$\begin{cases} v_z = \ddot{z}_d - \lambda_z(x_2 - \dot{z}_d) - K_z sign(S_z) - Q_z S_z \\ v_\phi = \ddot{\phi}_d - \lambda_\phi(x_6 - \dot{\phi}_d) - K_\phi sign(S_\phi) - Q_\phi S_\phi \\ v_\theta = \ddot{\theta}_d - \lambda_\theta(x_7 - \dot{\theta}_d) - K_\theta sign(S_\theta) - Q_\theta S_\theta \\ v_\psi = \ddot{\psi}_d - \lambda_\theta(x_8 - \dot{\psi}_d) - K_\theta sign(S_\theta) - Q_\theta S_\theta \end{cases} \tag{4.34}$$

4.8 Résultats de la mise en oeuvre des commandes

4.8.1 Commande de l'angle de lacet ψ

Nous avons simulé le système pour une référence de $\psi_d = 30^\circ$ et avons obtenus les résultats de la figure 4.30.

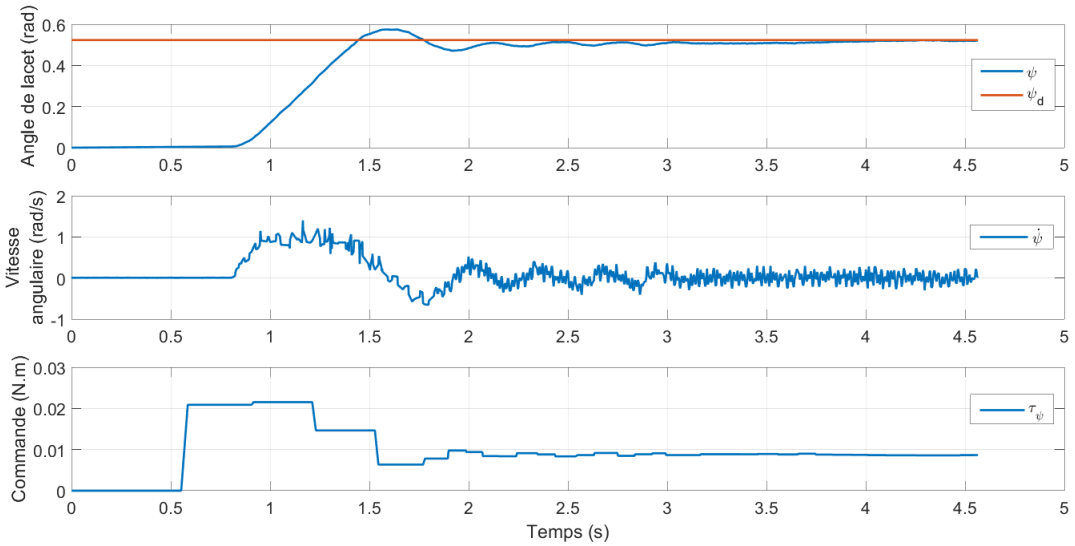


FIGURE 4.30: Résultats de la commande par PID de l'angle de lacet ψ

On constate que le système atteint bien la référence malgré les bruits qui entachent les mesures et la nature échantillonnée de la commande.

Malheureusement, à cause du manque de temps, de la mauvaise qualité des capteurs utilisé et de la faible puissance de calcul du Raspberry Pi nous n'avons pas pu implémenté toutes les commandes sur notre quadrirotor.

4.9 Conclusion

Dans ce chapitre, nous nous sommes intéressé à la réalisation d'un quadrirotor et à l'implémentation de commandes sur ce dernier. Nous avons commencé par une présentation du quadrirotor que nous avons réalisé et des quelques modifications que nous avons fait pour le repérage et les commandes. Ensuite, nous avons présenté le matériel utilisé pour la réalisation ainsi que les logiciels, bibliothèques et le langage de programmation utilisés. Puis, nous avons expliqué la méthode d'acquisition des données et les filtres utilisés pour obtenir de bonnes mesures. Après cela, nous avons procédé à l'identification de quelques paramètres nécessaires pour l'implémentation des commandes. Enfin, nous avons implémenté une commande par PID pour l'asservissement de l'angle de lacet ψ.

Conclusion Générale et Perspectives

Le quadrirotor est l'un des mini drones les plus populaires de part sa relative simplicité de fabrication et sa dynamique. Par conséquent, il a attiré l'attention et est devenu le sujet de beaucoup de recherches ces dernières années. C'est un système complexe, non linéaire, multi-variables, instable et présente une dynamique fortement couplée, ce qui fait de sa commande un grand défi.

Nous avons commencé ce document par une brève introduction sur les quadrirotors, où nous avons définit ce qu'ils sont, présenté leurs avantages, l'état de l'art et donné quelques exemples d'applications et d'utilisations. Nous avons également modélisé sa dynamique et nous avons synthétisé une commande par mode glissant pour le stabiliser et le piloter et nous l'avons enfin simulé pour nous assurer de sa justesse et de sa robustesse.

Nous avons ensuite fait une introduction au problème de SLAM et présenté l'historique de ce dernier. Puis, nous avons présenté les différents capteurs utilisés pour le résoudre. Ensuite, nous nous sommes intéressé au problème du SLAM monoculaire pour lequel nous avons présenté l'état de l'art et avons proposée une approche pour le résoudre.

Après cela, nous avons présenté ROS et Gazebo et expliqué en quoi consiste leurs notions de bases et leur importance. Ces deux outils ont été utilisé pour faire une simulation réaliste de l'implémentation de la commande et de notre approche de SLAM sur un modèle de quadrirotor en 3D.

Finalement, nous avons réalisé un quadrirotor et nous avons implémenté une commande par PID pour l'asservissement de l'angle de lacet ψ. Nous avons commencé par une présentation de ce dernier et du matériel, des logiciels, des bibliothèques et des langages de programmation utilisés. Ensuite, nous avons expliqué la méthode d'acquisition des données et les filtres utilisés pour obtenir de bonnes mesures. Puis, nous avons procédé à l'identification des paramètres nécessaires pour l'implémentation des différents algorithmes de commande et d'observation. Enfin, nous avons implémenté une commande par PID pour l'asservissement de l'angle de lacet ψ.

Dans le perspective d'une continuité de ce travail, nous proposons :
- La modélisation de tout les phénomènes qui régissent la dynamique du quadrirotor.
- La création d'un modèle de quadrirotor sous Gazebo identique à celui réalisé pour pouvoir faire des essais plus approfondis sans risquer d'endommager ce dernier.
- L'utilisation d'autres méthodes plus performante pour l'estimation de l'échelle.
- L'amélioration du quadrirotor réalisé en vue d'obtenir de meilleures performances et une meilleure robustesse.
- L'utilisation de meilleurs capteurs que ceux utilisés et d'un langage de programmation plus performant tel que le C ou le C++.

Bibliographie

[1] D. GHEORGHIŢĂ et al. « Quadcopter control system ». In : *System Theory, Control and Computing (ICSTCC), 2015 19th International Conference on*. 2015, p. 421–426. DOI : 10.1109/ICSTCC.2015.7321330.

[2] Strasdat H., Montiel J.M.M. et Davi-son A.J. « Real-time monocular slam : Why filter ? » In : *IEEE Int. Conf. Robotics and Automation (ICRA)*. 2010.

[3] ICAO (International Civil Aviation ORGANIZATION). *Circular 328 AN/190 : Unmanned Aircraft Systems*. 2011.

[4] George J. Vachtsevanos (eds.) KIMON P. VALAVANIS. *Handbook of Unmanned Aerial Vehicles*. 1re éd. Springer Netherlands, 2015. ISBN : 978-90-481-9706-4,978-90-481-9707-1.

[5] Rami ABOUSLEIMAN et al. *The Oakland University Unmanned Aerial Quadrotor System*. AUVSI UAS Student Competition, 2008.

[6] Andrew ZULU et Samuel JOHN. « A Review of Control Algorithms for Autonomous Quadrotors ». In : *Open Journal of Applied Sciences* 4 (2014), p. 547–556. DOI : 10.4236/ojapps.2014.414053.

[7] Gabriel M. HOFFMANN et al. « Quadrotor helicopter flight dynamics and control: Theory and experiment ». In : *In Proc. of the AIAA Guidance, Navigation, and Control Conference*. 2007.

[8] B. MICHINI et al. « Design and flight testing of an autonomous variable-pitch quadrotor ». In : *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. 2011, p. 2978–2979. DOI : 10.1109/ICRA.2011.5980561.

[9] Richard M. MURRAY. « TRAJECTORY GENERATION FOR A TOWED CABLE SYSTEM USING DIFFERENTIAL FLATNESS ». In : *IFAC World Congress*.

[10] Nathan MICHAEL, Jonathan FINK et Vijay KUMAR. « Cooperative manipulation and transportation with aerial robots ». In : *Autonomous Robots* 30.1 (2011), p. 73–86. ISSN : 1573-7527. DOI : 10.1007/s10514-010-9205-0. URL : http://dx.doi.org/10.1007/s10514-010-9205-0.

[11] P. POUNDS et R. MAHONY. « Design principles of large quadrotors for practical applications ». In : *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*. 2009, p. 3265–3270. DOI : 10.1109/ROBOT.2009.5152390.

[12] Mark CUTLER et al. « Comparison of Fixed and Variable Pitch Actuators for Agile Quadrotors ». In : *AIAA Guidance, Navigation, and Control Confrence*. Portland, Oregon, 2011.

[13] P. DAPONTE et al. « Metrology for drone and drone for metrology: Measurement systems on small civilian drones ». In : *Metrology for Aerospace (MetroAeroSpace), 2015 IEEE*. 2015, p. 306–311. DOI : 10.1109/MetroAeroSpace.2015.7180673.

[14] Parth N. Patel et al. « Quadcopter for Agricultural Surveillance ». In : *Advance in Electronic and Electric Engineering* (2013), p. 427–432. ISSN : 2231-1297.

[15] L. F. Luque-Vega et al. « Power line inspection via an unmanned aerial system based on the quadrotor helicopter ». In : *MELECON 2014 - 2014 17th IEEE Mediterranean Electrotechnical Conference*. 2014, p. 393–397. DOI : 10.1109/MELCON.2014.6820566.

[16] Randal Beard. « Quadrotor Dynamics and Control Rev 0.1 ». In : (2008).

[17] Paul Pounds et al. « Design of a four-rotor aerial robot ». In : *Proceedings of the 2002 Australasian Conference on Robotics and Automation (ACRA 2002)*. Australian Robotics & Automation Association. 2002, p. 145–150.

[18] Matko Orsag et Stjepan Bogdan. « Influence of Forward and Descent Flight on Quadrotor Dynamics ». In : Dr. Ramesh Agarwal. *Recent Advances in Aircraft Technology*. 2012. ISBN : 978-953-51-0150-5. URL : http://www.intechopen.com/books/recent-advances-in-aircraft-technology/influence-of-forwardand-descent-flight-on-quadrotor-dynamics.

[19] Y. C. Choi et H. S. Ahn. « Nonlinear Control of Quadrotor for Point Tracking: Actual Implementation and Experimental Tests ». In : *IEEE/ASME Transactions on Mechatronics* 20.3 (2015), p. 1179–1192. ISSN : 1083-4435. DOI : 10.1109/TMECH.2014.2329945.

[20] S. H. Dolatabadi et M. J. Yazdanpanah. « MIMO sliding mode and backstepping control for a quadrotor UAV ». In : *2015 23rd Iranian Conference on Electrical Engineering*. 2015, p. 994–999. DOI : 10.1109/IranianCEE.2015.7146356.

[21] C. Wu. « Robust output feedback position control for quadrotor based on disturbance observer ». In : *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2015, p. 446–451. DOI : 10.1109/ROBIO.2015.7418808.

[22] R. Mahony, V. Kumar et P. Corke. « Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor ». In : *IEEE Robotics Automation Magazine* 19.3 (2012), p. 20–32. ISSN : 1070-9932. DOI : 10.1109/MRA.2012.2206474.

[23] Guillaume Charland-Arcand. « Contrôle non linéaire par backstepping d'un hélicoptère de type quadrotor pour des applications autonômes ». Maîtrise en génie électrique. École de Technologie Supérieure, Université du Québec, 2014.

[24] Matthew Rich. « Model development, system identification, and control of a quadrotor helicopter ». MASTER OF SCIENCE. Iowa State University, 2012.

[25] Samir Bouabdallah, Andre Noth et Roland Siegwart. « PID vs LQ control techniques applied to an indoor micro quadrotor ». In : *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*. T. 3. IEEE. 2004, p. 2451–2456.

[26] Ian D Cowling, James F Whidborne et Alastair K Cooke. « Optimal trajectory planning and LQR control for a quadrotor UAV ». In : *UKACC International Conference on Control*. 2006.

[27] Samir Bouabdallah. « Design and control of quadrotors with application to autonomous flying ». Thèse de doct. Ecole Polytechnique Federale de Lausanne, 2007.

[28] Tarek Madani et Abdelaziz Benallegue. « Backstepping control for a quadrotor helicopter ». In : *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE. 2006, p. 3255–3260.

[29] Mehmet Önder Efe. « Robust low altitude behavior control of a quadrotor rotorcraft through sliding modes ». In : *Control & Automation, 2007. MED'07. Mediterranean Conference on*. IEEE. 2007, p. 1–6.

[30] Y. Li et S. Song. « A survey of control algorithms for Quadrotor Unmanned Helicopter ». In : *Advanced Computational Intelligence (ICACI), 2012 IEEE Fifth International Conference on*. 2012, p. 365–369. DOI : 10.1109/ICACI.2012.6463187.

[31] P. Kachroo et M. Tomizuka. « Chattering reduction and error convergence in the sliding-mode control of a class of nonlinear systems ». In : *IEEE Transactions on Automatic Control* 41.7 (1996), p. 1063–1068. ISSN : 0018-9286. DOI : 10.1109/9.508917.

[32] Giovanni Costa et Gianluigi Fogli. « The rotation group ». In : *Symmetries and Group Theory in Particle Physics*. 2012. Chap. 2. ISBN : 978-3-642-15481-2. DOI : 10.1007/978-3-642-15482-9.

[33] Vincent Bregeault. « Quelques contributions à la théorie de la commande par modes glissants ». Thèse de doct. Ecole Centrale de Nantes (ECN)(ECN)(ECN)(ECN), 2010.

[34] Markus Achtelik et al. « Onboard IMU and monocular vision based control for MAVs in unknown in-and outdoor environments ». In : *Robotics and automation (ICRA), 2011 IEEE international conference on*. IEEE. 2011, p. 3056–3063.

[35] Jakob Engel, Jürgen Sturm et Daniel Cremers. « Camera-based navigation of a low-cost quadrocopter ». In : *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE. 2012, p. 2815–2821.

[36] Christian Forster, Matia Pizzoli et Davide Scaramuzza. « SVO: Fast semi-direct monocular visual odometry ». In : *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, p. 15–22.

[37] Hugh Durrant-Whyte et Tim Bailey. « Simultaneous localization and mapping: part I ». In : *Robotics & Automation Magazine, IEEE* 13.2 (2006), p. 99–110.

[38] Josep Aulinas et al. « The SLAM problem: a survey. » In : *CCIA*. Citeseer. 2008, p. 363–371.

[39] Randall C Smith et Peter Cheeseman. « On the representation and estimation of spatial uncertainty ». In : *The international journal of Robotics Research* 5.4 (1986), p. 56–68.

[40] H. F. Durrant-Whyte. « Uncertain geometry in robotics ». In : *IEEE Journal on Robotics and Automation* 4.1 (1988), p. 23–31. ISSN : 0882-4967. DOI : 10.1109/56.768.

[41] Nicholas Ayache et Olivier D Faugeras. « Building, registrating, and fusing noisy visual maps ». In : *The International Journal of Robotics Research* 7.6 (1988), p. 45–65.

[42] James L Crowley. « World modeling and position estimation for a mobile robot using ultrasonic ranging ». In : *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*. IEEE. 1989, p. 674–680.

[43] Raja Chatila et Jean-Paul Laumond. « Position referencing and consistent world modeling for mobile robots ». In : *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*. T. 2. IEEE. 1985, p. 138–145.

[44] Randall Smith, Matthew Self et Peter Cheeseman. « Estimating uncertain spatial relationships in robotics ». In : *Autonomous robot vehicles*. Springer, 1990, p. 167–193.

[45] John J Leonard et Hugh F Durrant-Whyte. « Simultaneous map building and localization for an autonomous mobile robot ». In : *Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on*. Ieee. 1991, p. 1442–1447.

[46] Michael Csorba. « Simultaneous localisation and map building ». Thèse de doct. University of Oxford, 1997.

[47] John J Leonard et Hans Jacob S Feder. « A computationally efficient method for large-scale concurrent mapping and localization ». In : *ROBOTICS RESEARCH-INTERNATIONAL SYMPOSIUM-*. T. 9. Citeseer. 2000, p. 169–178.

[48] José A Castellanos, Juan D Tardós et Günther Schmidt. « Building a global map of the environment of a mobile robot: The importance of correlations ». In : *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*. T. 2. IEEE. 1997, p. 1053–1059.

[49] José A Castellanos et al. « Experiments in multisensor mobile robot localization and map building ». In : *Proc. 3rd IFAC Sym. Intell. Auton. Vehicles* (1998), p. 173–178.

[50] J Guivant, EM Nebot et Stephan Baiker. « Localization and map buiding using laser range sensors in outdoor applications ». In : *Journal of Robotics Systems* 17.10 (2001).

[51] Sebastian Thrun, Wolfram Burgard et Dieter Fox. « A probabilistic approach to concurrent mapping and localization for mobile robots ». In : *Autonomous Robots* 5.3-4 (1998), p. 253–271.

[52] Kok Seng Chong et Lindsay Kleeman. « Feature-based mapping in real, large scale environments using an ultrasonic array ». In : *The International Journal of Robotics Research* 18.1 (1999), p. 3–19.

[53] Matthew Deans et Martial Hebert. « Experimental comparison of techniques for localization and mapping using a bearing-only sensor ». In : *Experimental Robotics VII*. Springer, 2001, p. 395–404.

[54] Andrew J Davison et al. « MonoSLAM: Real-time single camera SLAM ». In : *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 29.6 (2007), p. 1052–1067.

[55] Georg Klein et David Murray. « Parallel tracking and mapping for small AR workspaces ». In : *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE. 2007, p. 225–234.

[56] Miroslav Trajković et Mark Hedley. « Fast corner detection ». In : *Image and vision computing* 16.2 (1998), p. 75–87.

[57] Bill Triggs et al. « Bundle adjustment—a modern synthesis ». In : *Vision algorithms: theory and practice*. Springer, 1999, p. 298–372.

[58] Richard A Newcombe, Steven J Lovegrove et Andrew J Davison. « DTAM: Dense tracking and mapping in real-time ». In : *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, p. 2320–2327.

[59] Jakob Engel, Thomas Schöps et Daniel Cremers. « LSD-SLAM: Large-scale direct monocular SLAM ». In : *Computer Vision–ECCV 2014*. Springer, 2014, p. 834–849.

[60] Raul Mur-Artal, JMM Montiel et Juan D Tardos. « ORB-SLAM: a versatile and accurate monocular SLAM system ». In : *Robotics, IEEE Transactions on* 31.5 (2015), p. 1147–1163.

[61] Ethan Rublee et al. « ORB: an efficient alternative to SIFT or SURF ». In : *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, p. 2564–2571.

[62] Thierry Viéville et Olivier D Faugeras. « Cooperation of the inertial and visual systems ». In : *Traditional and non-traditional robotic sensors*. Springer, 1990, p. 339–350.

[63] Peter Corke, Jorge Lobo et Jorge Dias. « An introduction to inertial and visual sensing ». In : *The International Journal of Robotics Research* 26.6 (2007), p. 519–535.

[64] A DÉCOUVRIR ÉGALEMENT. « La théorie de la relativité restreinte et générale ». In : (1990).

[65] Jorge J Moré. « The Levenberg-Marquardt algorithm: implementation and theory ». In : *Numerical analysis*. Springer, 1978, p. 105–116.

[66] Gabe Sibley et al. « Vast-scale outdoor navigation using adaptive relative bundle adjustment ». In : *The International Journal of Robotics Research* (2010).

[67] Kurt Konolige et al. « View-based maps ». In : *The International Journal of Robotics Research* (2010).

[68] Frank Dellaert et Michael Kaess. « Square Root SAM: Simultaneous localization and mapping via square root information smoothing ». In : *The International Journal of Robotics Research* 25.12 (2006), p. 1181–1203.

[69] Paul Furgale, Timothy D Barfoot et Gabe Sibley. « Continuous-time batch estimation using temporal basis functions ». In : *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, p. 2088–2095.

[70] Rudolph E Kalman et Richard S Bucy. « New results in linear filtering and prediction theory ». In : *Journal of basic engineering* 83.1 (1961), p. 95–108.

[71] Jakob Engel, Jürgen Sturm et Daniel Cremers. « Accurate figure flying with a quadrocopter using onboard visual and inertial sensing ». In : *IMU* 320 (2012), p. 240.

[72] Jakob Engel, Jürgen Sturm et Daniel Cremers. « Scale-aware navigation of a low-cost quadrocopter with a monocular camera ». In : *Robotics and Autonomous Systems* 62.11 (2014), p. 1646–1656.

[73] D Gordon E Robertson et James J Dowling. « Design and responses of Butterworth and critically damped digital filters ». In : *Journal of Electromyography and Kinesiology* 13.6 (2003), p. 569–573.

[74] Reza Raoufi et Hamid Khaloozadeh. *Stochastic/Adaptive Sliding Mode Observer for Noisy Excessive Uncertainties Nonlinear Systems*. Liege, Belgium : 15th PSCC, 2005.

[75] Bernt Øksendal. *Stochastic differential equations*. Springer, 2003.

[76] Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

[77] Fakhreddin Abedi, Wah June Leong et Sarkhosh Seddighi Chaharborj. « A notion of stability in probability of stochastic nonlinear systems ». In : *Advances in Difference Equations* 2013.1 (2013), p. 1–8.

[78] James Kramer et Matthias Scheutz. « Development environments for autonomous mobile robots: A survey ». In : *Autonomous Robots* 22 (2007), p. 132.

[79] *ROS*. 27 mai 2016. URL : http://www.ros.org/.

[80] *Licence BSD*. 27 mai 2016. URL : http://www.freebsd.org/copyright/license.html.

[81] *Licence MIT*. 27 mai 2016. URL : https://opensource.org/licenses/MIT.

[82] *Simulateur Gazebo*. 28 mai 2016. URL : http://gazebosim.org/.

[83] *Robot Baxter*. 10 juin 2016. URL : http://www.rethinkrobotics.com/baxter.

[84] *Rviz*. 28 mai 2016. URL : http://wiki.ros.org/rviz.

[85] *Robot PR2*. 10 juin 2016. URL : https://www.willowgarage.com/pages/pr2/overview.

[86] *Package Hector Quadrotor*. 11 juin 2016. URL : http://wiki.ros.org/hector_quadrotor.

[87] *Raspberry Pi*. 17 juin 2016. URL : https://www.raspberrypi.org/.

[88] Robert J Urick. *Principles of underwater sound for engineers*. Tata McGraw-Hill Education, 1967.

[89] *Python, langage de programmation*. 24 juin 2016. URL : https://www.python.org.

[90] *TIOBE Index pour Juin 2016*. 24 juin 2016. URL : http://www.tiobe.com/tiobe_index.

[91] *Bibliothèque Python pour le PCA9685*. 24 juin 2016. URL : https://github.com/adafruit/Adafruit-PWM-Servo-Driver-Library.

[92] *RPi.GPIO, bibliothèque pour python*. 24 juin 2016. URL : https://pypi.python.org/pypi/RPi.GPIO.

[93] Johannes Meyer et al. « Comprehensive simulation of quadrotor uavs using ros and gazebo ». In : *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer. 2012, p. 400–411.

[94] Alex G Quinchia et al. « A comparison between different error modeling of MEMS applied to GPS/INS integrated systems ». In : *Sensors* 13.8 (2013), p. 9549–9588.

[95] Robert Grover Brown et Patrick YC Hwang. « Introduction to random signals and applied Kalman filtering: with MATLAB exercises and solutions ». In : *Introduction to random signals and applied Kalman filtering: with MATLAB exercises and solutions, by Brown, Robert Grover.; Hwang, Patrick YC New York: Wiley, c1997*. 1 (1997).

[96] LAIB Khaled et MAAMRIA Djameleddine. « Commande d'un quadrirotor ». Projet de fin d'études, Ingéniorat. Ecole Nationale Polytechnique, Alger, juin 2011.

Annexe A

Paramètres utilisés

Les paramètres physique utilisés pour les simulations du quadrirotor sont résumés dans le tableau suivant[96] :

Paramètre	**Symbole**	**Valeur**	**Unité**
Masse du quadrirotor	m	0.650	kg
Distance entre le centre d'un moteur et le centre de gravité	l	0.230	m
Moment d'inertie du quadrirotor par rapport à son axe 'X'	J_x	$7.510^{-}3$	$kg.m^2$
Moment d'inertie du quadrirotor par rapport à son axe 'Y'	J_y	$7.510^{-}3$	$kg.m^2$
Moment d'inertie du quadrirotor par rapport à son axe 'Z'	J_z	$1.310^{-}2$	$kg.m^2$
Moment d'inertie du rotor par rapport à son axe 'Z'	J_r	610^{-5}	$kg.m^2$
Coefficient de portance	b	$3.1310^{-}5$	$N.s^2$
Coefficient de trainée	k	$7.510^{-}5$	$N.m.s^2$
Constante de gravité	g	9.81	$m.s^{-2}$

TABLE 1.1: Les paramètres physiques utilisés pour le système

Les gains utilisés dans les surfaces pour la commande par mode glissant synthétisé au premier chapitre sont résumés dans le tableau suivant :

Symbole	**Valeur**
λ_1	0.7
λ_2	0.7
λ_3	1.0
λ_4	2.8
λ_5	2.8
λ_6	2.0

TABLE 1.2: Les paramètres des surfaces de commande

Les gains de la fonction $sign(.)$ pour la commande par mode glissant synthétisé au premier chapitre ainsi que le paramètre de la fonction $pseudo-signe(.)$ sont résumés dans le tableau suivant :

Symbole	**Valeur**
K_1	0.5
K_2	0.5
K_3	1.0
K_4	7.0
K_5	7.0
K_6	3.0
δ	0.05

TABLE 1.3: Les gains des fonctions $sign(.)$ et le paramètre de lissage

Les gains proportionnels pour la commande par mode glissant synthétisé au premier chapitre sont résumés dans le tableau suivant :

Symbole	**Valeur**
Q_1	1.0
Q_2	1.0
Q_3	2.5
Q_4	2.0
Q_5	2.0
Q_6	2.0

TABLE 1.4: Les gains de l'action proportionnelle

Les biais de l'accéléromètre et du gyroscope sont résumés dans le tableau suivant :

Symbole	**Valeur**
b_{ax}	0.0287
b_{ay}	-0.0091
b_{az}	0.0644
$b_{\omega x}$	0.0118
$b_{\omega y}$	0.0039
$b_{\omega z}$	-0.0106

TABLE 1.5: Les biais de l'accéléromètre et du gyroscope

Les coefficients de l'approximation du premier ordre de la relation entre la poussée et le signal de commande PWM sont résumés dans le tableau suivant :

Coefficient	Valeur
a	0.0017
b	-0.9335

TABLE 1.6: Coefficients de l'approximation de la poussée

Les coefficients de l'approximation du premier ordre de la relation entre le moment de trainée et le signal de commande PWM sont résumés dans le tableau suivant :

Coefficient	Valeur
c	$1.236210^{-}8$
d	$3.0737e10^{-}5$

TABLE 1.7: Coefficients de l'approximation du moment de trainée