

République Algérienne Démocratique et Populaire Ministère de l'Enseignement
Supérieur et de la Recherche Scientifique.

Université ABOU BEKR BELKAID de Tlemcen

Faculté De Technologie

Département Du G.E.E

MEMOIRE DE FIN D'ETUDE POUR L'OBTENTION DU DIPLOME
DE MASTER ACADEMIQUE

Spécialité : Automatique

Option : Automatique et informatique industrielle

préparé au Laboratoire d'Automatique de Tlemcen (LAT)

**Synthèse de régulateur PID pour un quadrotor et
génération de carte en fonction de l'indice de végétation**

Présenté par :

Bougrara Samira Kawther

Boutrigue Sarah

Soutenu devant le jury :

A. Hadj Abdelkader Président Professeur

B. Benyahia Examineur MCA

A. Choukchou-Braham Encadrante Professeur

Année universitaire : 2019 / 2020

Dédicaces

À l'homme de ma vie, mon exemple éternel, mon soutien moral et source de joie et de bonheur, celui qui s'est toujours sacrifié pour me voir réussir, que Dieu te garde , à toi mon père.

À la lumière de mes jours, la source de mes efforts, la flamme de mon cœur, ma vie et mon bonheur, maman que j'adore.

Aux personnes dont j'ai bien aimé la présence à ce jour, à toutes mes sœurs, **Hadjer** , **Nour El Houda** et **Maram**, à mon frère **Imad Eddine**, je dédie ce travail dont le grand plaisir leurs revient en premier lieu pour leurs conseils, aides, et encouragements.

À mes grands-parents **Belkhir** et **Kheira**, à ma grand-mère **Yamna**. Je suis ici aujourd'hui à cause de vos prières et de vos douaa.

Aux personnes qui m'ont toujours aidées et encouragées, qui étaient toujours à mes côtés, et qui m'ont accompagnées durant mon chemin d'études supérieures, mes aimables amies, collègues d'étude, et frères de cœur, toi **Leila**, **Asma** et **Bouchra**.

Bouhrara Samira Kawther

Dédicaces

Je dédie ce travail à mes chers parents BOUTRIGUE Mohammed et BENYEKHELEF Karima, qui ont toujours cru en moi et ont mis à ma disposition tous les moyens nécessaires pour que je réussisse dans mes études.

- À mon grand frère Sid-ahemed
- À ma soeur Ghizlène
- À monsieur H. Benddermel
- À mes amis
- À toute ma famille
- À mon amie Ahmadouche Ghizlène

Boutrigue Sarah

Remerciements

Nous remercions **DIEU** le tout puissant de nous avoir donné la santé et la volonté d'entamer et de terminer ce mémoire.

Tout d'abord, ce travail ne serait pas aussi riche et n'aurait pas pu être réalisé sans l'aide et les conseils de madame **Amal CHOUKCHOU BRAHAM**, nous la remercions pour la qualité de son encadrement exceptionnel, pour sa patience, sa rigueur et sa disponibilité lors de la préparation de ce mémoire.

Nos remerciements s'adressent au directeur du Laboratoire d'Automatique de Tlemcen LAT, monsieur le professeur **Lotfi BAGHLI** de nous avoir accueilli au laboratoire et de nous avoir assuré des conditions de travail confortables.

Nous sommes conscientes de l'honneur que nous a fait monsieur **A. Hadj Abdelkader** en étant président du jury et monsieur **B. Benyahia** d'avoir accepté d'examiner et d'évaluer notre travail.

Nous remercions aussi monsieur **I. Meslouli** pour son aide et son encouragement.

Nos remerciements s'adressent également à tout nos professeurs pour leur générosité et la grande patience dont ils ont su faire preuve malgré leurs charges académiques et professionnelles.

Nos profonds remerciements vont également à toutes les personnes qui nous ont aidées et soutenues de près ou de loin.

Bouhrara Samira Kawther, Boutrig Sarah

Table des matières

1	Les Drones	3
1.1	Introduction	3
1.2	Historique	3
1.2.1	Histoire militaire	4
1.2.2	Histoire civile	5
1.3	Applications des drones	5
1.4	Classification des drones	6
1.4.1	Selon leur champ d'action	6
1.4.2	Selon leur configuration aérodynamique	7
1.4.3	Selon la taille et la charge utile	9
1.5	Les Quadrirotors	11
1.5.1	Description :	11
1.5.2	Technologie	11
1.5.3	Avantage et inconvénient	12
1.5.4	Recherche dans le domaine des quadrirotors	12
1.6	Conclusion	15
	partie théorique	3
2	Modélisation et synthèse de commande pour le quadrotor	16
2.1	Introduction	16
2.2	Description générale	16
2.3	Les mouvements de quadri-rotor	17
2.3.1	Mouvement de rotation	18
2.3.2	Les translations verticales	19
2.3.3	Les translations horizontales	19
2.3.4	Le vol stationnaire	20
2.4	Modélisation dynamique du quadri-rotor	20
2.4.1	Effets physiques agissants sur la quadri-rotor	20
2.5	Repère du quadri-rotor	23
2.5.1	Repères utilisés	23
2.5.2	Matrice de rotation	23

2.6	Vitesses angulaires	26
2.7	Vitesses linéaires	26
2.8	Développement du Modèle mathématique selon Newton-Euler	27
2.9	Equations de mouvement de rotation	28
2.10	Représentation d'état du système	29
2.11	Identification des paramètres	30
2.12	Synthèse du régulateur PID	31
2.13	Contraintes de sous actionnement	32
2.14	Simulation du modèle	34
2.15	Résultat des simulations	34
2.16	Etude de la robustesse	37
2.16.1	Robustesse paramétrique	37
2.16.2	Robustesse vis-à-vis une perturbation externe	41
2.17	Conclusion	46
3	Réalisation de la carte NDVI	47
3.1	Introduction	47
3.2	NDVI (Normalized Difference Vegetation Index)	47
3.2.1	Applications du NDVI	48
3.2.2	UAV pour NDVI	49
3.3	Partie hardware	50
3.3.1	Raspberry Pi 3 modele B	50
3.3.2	Raspberry Pi caméra rev L3	50
3.3.3	Alimentation électrique USB	51
3.3.4	Adaptateur et carte microSD	51
3.3.5	Clavier et souris USB	51
3.3.6	Câble VGA / VGA et adaptateur VGA / HDMI	52
3.3.7	Monitor	52
3.3.8	Montage finale et résultat de l'alimentation	53
3.4	Partie software	54
3.4.1	Introduction sur les images	54
3.5	Raspberry Pi OS (Raspbian)	56
3.5.1	Etapas d'installation	56
3.6	OpanCV-Python	57
3.6.1	OpenCV	57
3.6.2	OpenCV-Python	57
3.7	Etapas d'installation d'OpenCV sur Raspberry Pi	57
3.8	Résultat et interprétation	60
3.9	Conclusion	62

Table des figures

1.1	Lawrence and Sperry UAV	4
1.2	Predator	4
1.3	Drone HALE Global Hawk de Northrop Grumman	6
1.4	Drone HALE Phantom Eye de Boeing	6
1.5	UAV à Voilure fixe	7
1.6	(a) Mono-rotor	8
1.7	(b) Coaxial	8
1.8	(c) Quadrotor	8
1.9	(d) Multi-rotor	8
1.10	UAV à ailes battantes	9
1.11	UAV à grande échelle	9
1.12	UAV de taille moyenne :	10
1.13	UAV à petite échelle	10
1.14	Mini UAV	10
1.15	Micro-drones	11
1.16	Le Msicopter	13
1.17	Le Quadrirotor OS4	13
1.18	Le STARMAC	14
2.1	Description générale du quadri-rotor	17
2.2	Principe de fonctionnement du quadri-rotor	17
2.3	Illustration du mouvement de roulis	18
2.4	Illustration du mouvement de tangage	18
2.5	Illustration du mouvement de lacet	19
2.6	Illustration du mouvement vertical	19
2.7	Illustration du mouvement de translation	20
2.8	Principales forces agissantes sur le drone	21
2.9	Repérage du quadri-rotor	23
2.10	Rotation autour de l'axe X (Roulis)	24
2.11	Rotation autour de l'axe Y (Tangage)	24
2.12	Rotation autour de l'axe Z (Lacet)	25
2.13	Datasheet du moteur	30

2.14	Structure PID traditionnel	31
2.15	Modèle <i>Simulink</i> du quadri-rotor	34
2.16	Tracés de x et x_d	35
2.17	Tracés de Y et Y_d	35
2.18	Tracés de Z et Z_d	36
2.19	Tracés de ϕ et ϕ_d	36
2.20	Tracés de θ et θ_d	36
2.21	Tracés de ψ et ψ_d	36
2.22	Tracés de x et x_d avec la perturbation sur M	37
2.23	Tracés de x et x_d avant la perturbation	37
2.24	Tracés de y et y_d après la perturbation sur M	38
2.25	Tracés de y et y_d avant la perturbation	38
2.26	Tracés de z et z_d après la perturbation sur M	38
2.27	Tracés de z et z_d avant la perturbation	38
2.28	Tracés de ϕ et ϕ_d après la perturbation sur M	38
2.29	Tracés de ϕ et ϕ_d avant la perturbation	38
2.30	Tracés de θ et θ_d après la perturbation sur M	39
2.31	Tracés de θ et θ_d avant la perturbation	39
2.32	Tracés de ψ et ψ_d après la perturbation sur M	39
2.33	Tracés de ψ et ψ_d avant la perturbation	39
2.34	Tracés de x et x_d après la perturbation sur M et d	39
2.35	Tracés de x et x_d avant la perturbation	39
2.36	Tracés de y et y_d après la perturbation sur M et d	40
2.37	Tracés de y et y_d avant la perturbation	40
2.38	Tracés de z et z_d après la perturbation sur M et d	40
2.39	Tracés de z et z_d avant la perturbation	40
2.40	Tracés de ϕ et ϕ_d après la perturbation sur M et d	40
2.41	Tracés de ϕ et ϕ_d avant la perturbation	40
2.42	Tracés de θ et θ_d après la perturbation sur M et d	41
2.43	Tracés de ψ et ψ_d avant la perturbation	41
2.44	Tracés de ψ et ψ_d après la perturbation sur M et d	41
2.45	Tracés de ψ et ψ_d avant la perturbation	41
2.46	Schéma global de la commande en présence de perturbation	42
2.47	Tracés de x et x_d avec un vent calme	42
2.48	Tracé de y et y_d avec un vent calme	42
2.49	Tracés de z et z_d avec un vent calme	43
2.50	Tracés de ϕ et ϕ_d avec un vent calme	43
2.51	Tracés de θ et θ_d avec un vent calme	43
2.52	Tracés de ψ et ψ_d avec un vent calme	44
2.53	Echelle de <i>Beaufort</i>	44
2.54	Tracés de x et x_d avec un vent du Petite brise	45

2.55	Tracés de y et y_d avec un vent du Petite brise	45
2.56	Tracés de z et z_d avec un vent du Petite brise	45
2.57	Tracés de ϕ et ϕ_d avec un vent du Petite brise	45
2.58	Tracés de θ et θ_d avec un vent du Petite brise	45
2.59	Tracés de ψ et ψ_d avec un vent du Petite brise	45
2.60	Tracés de ψ et ψ_d avec un vent du Petite brise	46
3.1	Comparaison entre la lumière visible réfléchie et la lumière proche infrarouge pour la végétation verte et la végétation sénescente	48
3.2	Quadricoptère avec caméra intégrée	49
3.3	Montage des éléments avec la Raspberry	53
3.4	Résultat de l'alimentation	53
3.5	54
3.6	54
3.7	54
3.8	Image originale	55
3.9	Image binaire	55
3.10	GrayScale image	55
3.11	SD Card Formatter	56
3.12	BalenaEtcher	56
3.13	Image originale	61
3.14	Résulta d'affichage	61
3.15	Image originale	61
3.16	Résultat du traitement NDVI	61
3.17	Filtre IR	62
3.18	Papier bleu transparent	62
3.19	Résultat final	62

Acronymes

UAV Unmanned Aerial Vehicles.

DAST Drones for Aerodynamic and Structural Testing.

HiMAT Highly Maneuverable Aircraft Technology.

ERAST Environmental Research Aircraft and Sensor Technology.

HALE Haute Altitude Longue Endurance.

MALE Moyenne Altitude Longue Endurance.

TUAV Tactical UAV (Medium-Range)

MUAV Mini UAV.

MAV Micro UAV.

NAV Nano Air Vehicles.

RC Radio Controlled.

ASL Laboratoire des Systèmes Autonomes.

EPFL Ecole Polytechnique Fédérale de Lausanne.

STARMAC Stanford Testbed of Autonomous Rotorcraft for Multi-Agent Control.

PID Proportionnelle, Intégrale, Dérivée.

DDL Degré De Liberté.

Notations

F_i : Force générée par chaque actionneur

ϕ : Angle de rotation créé selon X

θ : Angle de rotation créé selon Y

ψ : Angle de rotation créé selon Z

c, s, tan, atan : Cos, sin, tangente, arc-tangente

Rot_x, Rot_y, Rot_z : Matrice de rotation autour de X, Y et Z

P : Pesanteur

f_g : Force de gravité

m : Masse

g : Accélération de la pesanteur

F : Force de poussée

F_f : Force de portance

b : Coefficient de portance

ω : Vitesse angulaire

T_h : Traînée de l'hélice

d : Coefficient de drag

F_t : Traînée selon les axes

K_{ft} : Coefficient de traînée

V : Vitesse linéaire

L : Longueur du segment

M_f : Matrice de moment de portance

M_x, M_y, M_z : Moment de rotation selon les axes X, Y et Z

M_a : Moment de frottement aérodynamique

Ω : Vitesse angulaire dans le repère fixe

J_r : Inertie des rotors

\wedge : Produit vectoriel

M_{gh} : Moment gyroscopique selon les hélices

M_{gm} : Moment gyroscopique selon les mouvements du drone

J : Inertie du système

Introduction générale

Ces dernières années, un intérêt croissant a été porté à la robotique. En effet, de nombreuses industries (automobile, médicale, manufacturière, spatiale, ...), ont besoin de robots pour remplacer les hommes dans des situations dangereuses, ennuyeuses ou onéreuses. Un vaste domaine de cette recherche est consacré aux plates-formes aériennes.

Quand la réalité dépasse la fiction Voici une phrase pourtant banale qui résume très bien la grande révolution que nous avons sous les yeux. Car les drones sont parmi nous ! Plusieurs structures et configurations ont été développées pour permettre des mouvements en 3D. Par exemple, il existe des dirigeables, des avions à voilure fixe, des hélicoptères à rotor unique, des prototypes ressemblant à des oiseaux, des quadrotors. C'est alors que la réalité va véritablement dépasser la fiction.

Cette structure peut être attrayante dans plusieurs applications, notamment pour la topographie, l'imagerie, les environnements dangereux, la navigation intérieure et la cartographie. Les objectifs de ce projet de fin d'étude sont la modélisation du système, la synthèse de la commande PID, la conception du simulateur, l'étude de robustesse par rapport à un changement paramétriques et par rapport à deux formes de vent et la réalisation d'une carte NDVI. Ce travail s'inscrit dans le cadre de la continuation des travaux précédents menés au sein de l'équipe -commande- du laboratoire d'Automatique de Tlemcen.

Ce mémoire est structuré comme suit :

Chapitre 1 : Ce chapitre est consacré à la description des drones d'une manière générale et au quadrirotor en particulier. Nous donnons un bref historique des drones, leurs applications et leurs classifications.

Chapitre 02 : Dans ce chapitre une étude détaillée de modèle dynamique est présentée. Des hypothèses sont établies afin d'obtenir quelques simplifications du modèle. Nous concevons une loi de commande PID valable pour le modèle linéarisé. Des résultats en simulation sont présentés pour ce modèle afin d'illustrer les développements théoriques et enfin une analyse de robustesse est effectuée pour montrer l'efficacité du contrôleur PID.

Chapitre 3 : Dans le dernier chapitre, nous présentons de manière générale le NDVI, ses différentes applications dans différents domaines comme la cartographie de la déforestation, nous parlons aussi de la relation des drones avec le NDVI. Ensuite, nous passons à la partie matérielle dans la-

quelle nous avons décrit chaque composant utilisé dans le montage au cours de l'évolution de ce projet. Enfin, nous passons par la suite à la partie la plus difficile de ce travail dans laquelle nous présentons comment nous avons générée la carte NDVI en utilisant Raspberry pi 3 avec le Système d'exploitation Rasbian, et un code python basé sur une bibliothèque de vision par ordinateur ouverte (Open Computer Vision or OpenCV).

Nous terminons par une conclusion générale et donnons quelques perspectives.

Chapitre 1

Les Drones

Résumé

Dans ce premier chapitre nous présentons en bref l'état de l'art des drones, leur historique, leurs applications puis une classification. Parmi ces classes nous avons celle des quadri-rotors pour laquelle nous donnons une description, un développement technologique et citons également leurs avantages et inconvénients.

1.1 Introduction

Au cours des dernières années, des chercheurs et des ingénieurs de divers domaines ont travaillé intensivement pour développer des machines volantes efficaces capables d'effectuer des missions avec un minimum ou aucune intervention humaine. Ce type de véhicule est communément appelé Unmanned Aerial Vehicle (UAV) (véhicule aérien sans pilote). La théorie du contrôle, la vision par ordinateur, la mécanique, l'aérodynamique, l'automatisation et l'électronique embarquée sont quelques-uns des domaines liés au développement de ces systèmes. Ce chapitre présente tout d'abord un bref résumé des drones de manière historique, suivi d'une explication des applications et classifications les plus courantes de ce type de véhicule.

1.2 Historique

Les drones ont été fabriqués pour la première fois par "Lawrence" et "Sperry" (États-Unis) en 1916. Ils l'ont appelé "torpedo" d'aviation illustrée à la figure(1.1) et ils ont pu le faire voler sur une distance de 30 milles. Il a été rapporté que Lawrence et Sperry ont utilisé un gyroscope pour équilibrer le corps [5].



FIGURE 1.1 – Lawrence and Sperry UAV

1.2.1 Histoire militaire

Les États-Unis ont montré un grand intérêt pour développer des drones destinés à être utilisés pendant la Première Guerre mondiale et deux projets ont été financés. Elmer Sperry a d'abord développé le drone «Flying Bomb» et le deuxième projet était le «Kettering Bug» fabriqué par General Motors. Les deux projets ont été annulés et le financement a cessé car ils se sont révélés infructueux. Cela est dû au fait de l'absence des avancées technologiques requises dans les domaines des systèmes de guidage et des moteurs.

Le développement des drones a commencé à augmenter considérablement à la fin des années 1950, les États-Unis les ont déployés pendant la guerre du Vietnam pour diminuer le nombre de victimes chez les pilotes lorsque ils survolaient des territoires hostiles. Après leur succès, les États-Unis et Israël ont décidé d'investir davantage pour construire des drones plus petits et moins chers, ils ont utilisé de petits moteurs comme ceux trouvés dans les motos pour produire des drones plus petits et plus légers. De plus, une caméra vidéo a été ajoutée sur les drones pour transmettre des images à l'opérateur au sol. En 1991, les États-Unis ont largement utilisé des UAV pendant la guerre du Golfe, et le modèle le plus célèbre était le Predator illustré sur la figure(1.2). Les drones ont été utilisés de manière intensive par les États-Unis dans de nombreux conflits et guerres à la fin des années 1990 et au début des années 2000 et plus tard, les drones ont été largement utilisés dans la guerre contre l'Irak



FIGURE 1.2 – Predator

1.2.2 Histoire civile

Les utilisations des drones n'étaient pas seulement limitées à un usage militaire ; en 1969, la NASA a pris de plus en plus le souci de contrôler automatiquement un avion, les premiers essais ont été le programme PA-30, qui a réussi, mais ils avaient un pilote à bord pour prendre le contrôle de l'avion en cas de problème. D'autres programmes de recherche ont suivi le succès du programme PA-30 comme : les programmes Drones for Aerodynamic and Structural Testing (DAST) et Highly Maneuverable Aircraft Technology (HiMAT). Après cette époque, dans les années 1990, la NASA s'est associée à des entreprises industrielles pour développer un projet de recherche de neuf ans appelé "Environmental Research Aircraft and Sensor Technology Project" (ERAST). Ils ont développé plusieurs modèles de drones capables de voler à des altitudes allant jusqu'à 30 km et ont survécu à des vols jusqu'à 6 mois. Les modèles d'UAV résultants comprenaient le pathfinder, Helios, Atlas et Perseus B. Les drones développés transportaient plusieurs capteurs pour effectuer des mesures environnementales, les capteurs embarqués comprenaient une caméra, un interféromètre à balayage numérique (DASI) et un système actif de détection, de visualisation et d'évitement (DSA).

1.3 Applications des drones

En plus de l'utilisation militaire, les drones peuvent être utilisés dans de nombreuses applications civiles ou commerciales qui sont trop ennuyeuses ou trop dangereuses pour les avions pilotés. Parmi les rôles civils des drones, nous pouvons citer :

Les observations des sciences de la Terre provenant d'UAV peuvent être utilisées côte à côte avec celles acquises à partir de satellites. Ces missions comprennent :

1. Mesures des déformations de la croûte terrestre qui peuvent être des indications de catastrophes naturelles comme des tremblements de terre, des glissements de terrain ou des volcans.
2. Mesures des nuages et des aérosols.
3. Mesures de la pollution troposphérique et de la qualité de l'air pour déterminer les sources de pollution et comment les panaches de pollution sont transportés d'un endroit à un autre.
4. Mesures des épaisseurs de la calotte glaciaire et déformation de surface pour étudier le réchauffement climatique.
5. Mesures d'accélération gravitationnelle, puisque l'accélération gravitationnelle varie près de la Terre, les drones sont utilisés pour mesurer avec précision l'accélération gravitationnelle à plusieurs endroits afin de définir des références correctes.
6. Le débit de la rivière est mesuré à partir du volume d'eau qui coule dans une rivière en plusieurs points. Cela aidera dans les études mondiales et régionales sur le bilan hydrique [6].

Les drones de **recherche et sauvetage** équipés de caméras sont utilisés pour rechercher des survivants après des catastrophes naturelles comme des tremblements de terre et des ouragans ou des survivants d'épaves de navires et d'accidents d'avion [3].

Des drones **anti-incendie** équipés de capteurs infrarouges sont envoyés pour survoler des forêts sujettes aux incendies afin de le détecter à temps et de renvoyer un avertissement à la station au sol

avec l'emplacement exact du feu avant qu'il ne se propage ([6],[3]).

Les drones de **surveillance des frontières** sont utilisés pour patrouiller les frontières à la recherche d'intrus, d'immigrants illégaux ou de trafic de drogue et d'armes ([6],[3]).

Les drones de recherche sont également utilisés dans les recherches menées dans les universités pour vérifier certaines théories. De plus, les drones équipés de capteurs appropriés sont utilisés par les instituts de recherche environnementale pour surveiller certains phénomènes environnementaux comme la pollution dans les grandes villes [3].

Applications industrielles Les drones sont utilisés dans diverses applications industrielles telles que l'inspection ou la surveillance de pipelines et la surveillance des usines nucléaires [3].

Les drones de développement agricole ont également des utilisations agricoles telles que la pulvérisation des cultures [3].

1.4 Classification des drones

Il existe différentes façons de classer les drones, soit en fonction de leur champ d'action, de leur configuration aérodynamique, de leur taille et de leur charge utile, soit en fonction de leur niveau d'autonomie.

1.4.1 Selon leur champ d'action

Les drones peuvent être classés en 7 catégories différentes en fonction de leur altitude et de leur endurance maximales, comme suit([3], [14]) :

1. **Haute Altitude Longue Endurance (HALE)** : ils peuvent voler à plus de 15000 m d'altitude avec une endurance de plus de 24 heures. Ils sont principalement utilisés pour des missions de surveillance à longue distance.



FIGURE 1.3 – Drone HALE Global Hawk de Northrop Grumman



FIGURE 1.4 – Drone HALE Phantom Eye de Boeing

2. **Moyenne Altitude Longue Endurance (MALE)** : ils peuvent voler entre 5000 - 15000 m d'altitude pour un maximum de 24 heures. Les drones MALE sont également utilisés pour des missions de surveillance.
3. **UAV à moyenne portée ou tactique (TUAV)** : ils peuvent voler entre 100 et 300 m d'altitude. Ils sont plus petits et fonctionnent avec des systèmes plus simples que leurs homologues HALE et MALE.

4. **UAV à courte portée** : Ils ont une portée opérationnelle de 100 m. Ils sont principalement utilisés dans les applications civiles telles que l'inspection des lignes électriques, la pulvérisation des cultures, la surveillance du trafic, la sécurité intérieure, etc.
5. **Mini UAV (MUAV)** : Ils ont un poids d'environ 20 kg et une autonomie d'environ 30 km.
6. **Micro UAV (MAV)** : Ils ont une envergure maximale de 150 millimètre. Ils sont principalement utilisés à l'intérieur où ils doivent voler lentement et planer.
7. **Nano Air Vehicles (NAV)** : Ils ont une petite taille d'environ 10 millimètre. Ils sont principalement utilisés dans les essais pour des applications telles que la confusion radar. Ils sont également utilisés pour la surveillance à courte portée s'ils sont équipés d'une caméra tout aussi petite.

1.4.2 Selon leur configuration aérodynamique

Les drones peuvent être classés en quatre catégories principales en fonction de leur configuration aérodynamique comme suit :

1. **UAV à voilure fixe** : nécessitent une piste pour décoller et atterrir. Ils peuvent voler longtemps et à des vitesses de croisière élevées. Ils sont principalement utilisés dans des applications scientifiques telles que la reconnaissance météorologique et la surveillance de l'environnement, illustrées par la figure(1.5)

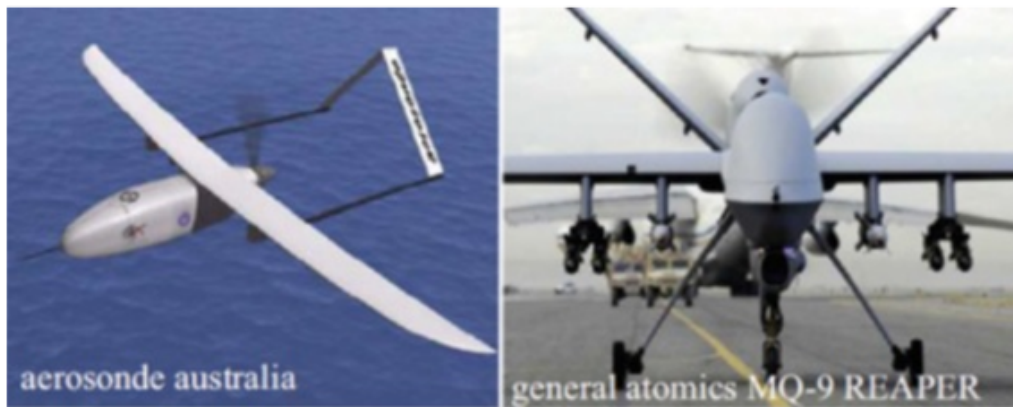


FIGURE 1.5 – UAV à Voilure fixe

2. **UAV à voilure tournante** : ils peuvent décoller et atterrir verticalement. Ils peuvent aussi planer et voler avec une grande maniabilité. Les drones à voilure tournante peuvent être classés en quatre groupes :



FIGURE 1.6 – (a) Mono-rotor



FIGURE 1.7 – (b) Coaxial



FIGURE 1.8 – (c) Quadrotor



FIGURE 1.9 – (d) Multi-rotor

- (a) **Mono-rotor** : ils ont un rotor principal sur le dessus et un autre rotor à la queue pour la stabilité, identique à la configuration de l'hélicoptère montré dans la figure (1.6)
- (b) **Coaxial** : ils ont deux rotors tournant dans des directions opposées montés sur le même arbre montré dans la figure(1.7)
- (c) **Quadrotor** : ils ont quatre rotors montés dans une configuration en croix montré dans la figure (1.8)
- (d) **Multi-rotor** : drones à six ou huit rotors ou plus. Ils sont de type agile et volent même lorsqu'un moteur tombe en panne, car il y a redondance en raison du grand nombre de rotors montré dans la figure (1.9)

l'augmentation du nombre de rotors augmente à son tour la charge utile et l'altitude maximale des drones, mais cela se fait au prix d'une augmentation de la taille et de la consommation d'énergie.

3. **UAV à ailes battantes** : ils sont inspirés des oiseaux et des insectes volants. Ces drones ont de petites ailes et ont une charge utile et une endurance extrêmement faibles. En revanche, ils ont une faible consommation d'énergie et peuvent effectuer des décollages et atterrissages verticaux. Cette classe d'UAV est toujours en cours de développement, figure(1.10)

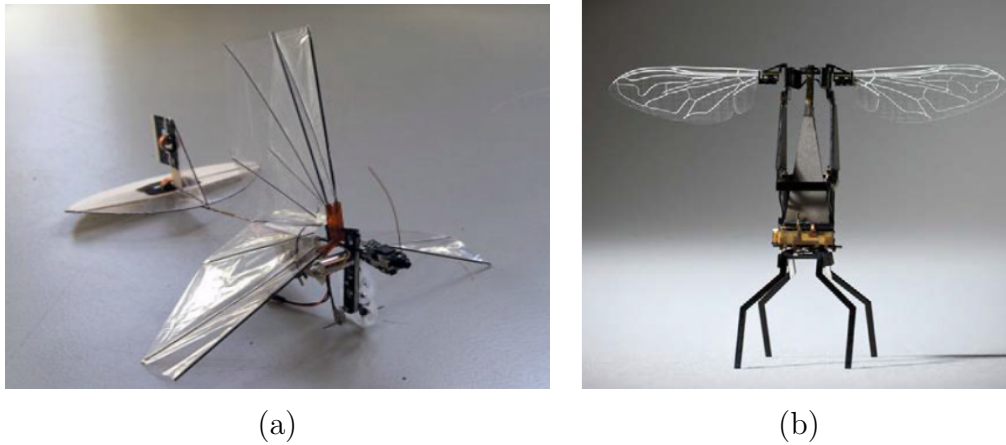


FIGURE 1.10 – UAV à ailes battantes

1.4.3 Selon la taille et la charge utile

Les drones peuvent être classés en cinq classes principales en fonction de leur taille et de leur charge utile. La figure montre des exemples de ces cinq classes comme décrit ci-dessous :

1. **UAV à grande échelle** : il s'agit de véhicules de taille normale, illustrés par la figure(1.11). Bien qu'un pilote puisse être présent à bord, le véhicule est capable de voler de façon autonome. Le pilote a pour but de faire une sauvegarde lors d'essais de vols et de manœuvres complexes. De plus, ces véhicules ont la charge utile et l'endurance les plus élevées.



FIGURE 1.11 – UAV à grande échelle

2. **UAV de taille moyenne** : ce sont les véhicules principalement utilisés pour les missions de sécurité. Ils ont une charge utile relativement élevée de 10 kg, ce qui permet d'avoir à bord des capteurs de navigation lourds et de haute qualité, réalisant ainsi des vols dépendants. La figure(1.12) montre un exemple de drone à échelle moyenne utilisé dans l'armée américaine.



FIGURE 1.12 – UAV de taille moyenne :

3. **UAV à petite échelle** : il s'agit principalement d'UAV basés sur des jouets radiocommandés (RC), ils ont une charge utile inférieure, de 2 à 10 kg, ce qui leur permet de transporter des capteurs de navigation de qualité adéquate. Un exemple de drone à petite échelle est illustré à la figure(1.13).



FIGURE 1.13 – UAV à petite échelle

4. **Mini UAV** : drones portables capables de voler à l'intérieur et à l'extérieur avec une charge utile inférieure à 2 kg, ce qui est suffisant pour transporter de petits capteurs légers. La petite taille, le faible coût et la facilité d'entretien de ces drones en font les drones de banc d'essai les plus courants dans les applications de recherche. Un exemple d'un mini UAV est montré dans la figure(1.14).



FIGURE 1.14 – Mini UAV

5. **Micro-drones** : principalement utilisés à l'intérieur en raison de leur très petite taille. Ils ont une charge utile inférieure à 100 g ce qui rend très difficile d'être équipé de capteurs de navigation et de guidage. Le défi de la recherche concernant ces micro-drones est de concevoir des capteurs de navigation et de guidage légers. Un exemple d'un micro UAV est montré dans la figure(1.15).



FIGURE 1.15 – Micro-drones

1.5 Les Quadrirotors

1.5.1 Description :

Le quadrirotor est un objet volant qui a la particularité de posséder 4 rotors placés aux extrémités d'un corps rigide en forme de croix.

L'électronique de contrôle est en général placée au centre de la croix qui constitue le centre de gravité de l'engin. Pour éviter au quadrirotor de tourner sur lui-même autour de son axe de lacet, il faut que les hélices appartenant au couple de moteur avant-arrière tournent dans un sens et que les hélices appartenant au couple gauche-droite tournent dans un autre sens. Le quadrirotor étant une configuration complètement instable, il faut développer des algorithmes permettant de contrôler chaque moteur séparément pour contrer l'inclinaison sur chaque axe et ainsi le stabiliser.

1.5.2 Technologie

Avant de commander n'importe quel système il faut bien définir ses composants pour bien pouvoir le modéliser, comme tout autre robot le quadrirotor est constitué essentiellement d'une partie mécanique qui constitue son squelette et ses muscles, plus une partie électronique rassemblant les capteurs, le calculateur et les modules de communications.

1.5.3 Avantage et inconvénient

- La conception du quadrirotor offre de réels avantages par rapport à d'autres configurations : [7]
- Sa taille réduite et sa manœuvrabilité lui 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.
 - Son décollage et atterrissage verticaux.
 - Cette configuration est commandée en variant seulement la vitesse de rotation des quatre moteurs.
 - Sa capacité de portance à cause de la présence de quatre rotors au lieu d'un qui peut être augmenté en rallongeant les pales d'un rotor ou en augmentant leur nombre, mais à cause de phénomènes aérodynamiques et d'encombrement, cela a des limites [17].
 - Réduction de l'effet gyroscopique.

1.5.4 Recherche dans le domaine des quadrirotors

Ils existent des projets qui portent sur les problèmes de la modélisation et la commande en se basant sur des plateformes commerciales comme le Draganflyer, HMX4 , UFO4,... [4], 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 [15].

Un quadrirotor consiste en une armature en croix symétrique avec des moteurs et des rotors aux extrémités de chaque tige. Les rotors diamétralement opposés tournent dans le même sens. Les rotors sont généralement non articulés.

Le projet Mesicopter (1999-2001)

Le Mesicopter, appelé également Meso-Scale, est un nano quadrirotor électrique de 40 gr et 1.5 cm d'envergure capable de voler en portant soit sa propre alimentation fournie par des batteries miniatures, soit avec une alimentation externe. Il porte des capteurs dédiés à la recherche atmosphérique ou l'exploration planétaire figure(1.16). Ce travail est une collaboration d'une équipe de chercheurs du département d'Aéronautique et d'Astronautique et le département de Mécanique à l'université de Stanford aux USA, avec l'appui des associés industriels Intel et SRI, qui est un leader dans la fabrication des batteries et des capteurs miniatures de la haute technologie, ainsi que la NASA. Ce dispositif miniature fait partie d'une classe très évoluée de robots volants très utiles dans les mesures atmosphériques [9].

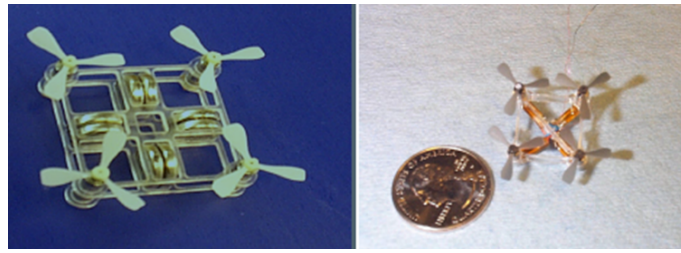


FIGURE 1.16 – Le Msicopter

Le projet OS4 (2003-2007) :

Au Laboratoire des Systèmes Autonomes (ASL) de l'Ecole Polytechnique Fédérale de Lausanne, EPFL, Suisse, de nombreux projets ont étudié des hélicoptères existants dans le commerce. Un modèle propre à l'EPFL a été développé : Omnidirectional Stationary Flying Outstretched Robot (OS4) figure(1.17).

Le développement de l'OS4 a eu pour but la réalisation d'un quadrirotor avec une pleine autonomie et capable de voler dans un environnement encombré. Dans ce projet, l'équipe a travaillé simultanément sur les aspects de la conception et de la commande, ceci a permis de simplifier la commande par changement de conception et vice-versa [1].



FIGURE 1.17 – Le Quadrirotor OS4

Le quadrirotor OS4 inclut tous les dispositifs nécessaires de l'avionique et d'énergie pour un vol entièrement autonome. Il comporte [1] :

- Une unité de mesure inertielle 3DM-GX1 comme capteur d'attitude.
- Un capteur de position basé sur la vision utilisant une caméra CCD miniature embarquée.
- Cinq capteurs à ultrason SRF10 Ultrasonic Ranger sont utilisés : Quatre pour l'évitement des obstacles et un pour la mesure de l'altitude.
- Un ordinateur embarqué Geode1200, de vitesse 266 Mhz et 128 Mo de RAM est utilisé pour l'implémentation en temps réel des lois de commande.

Le projet STARMAC 2007 :

STARMAC (Stanford Testbed of Autonomous Rotorcraft for Multi-Agent Control) est un autre projet très intéressant qui étudie la commande multi-agents à l'université de Stanford, département d'Aéronautique et d'Astronautique, USA [4]. Les quadrirotors conçus pour ce projet (figure 21) sont des plateformes autonomes dédiées pour des applications Outdoor idéales pour valider de nouveaux algorithmes de coordination multi-véhicules et répondent aux exigences suivantes [4] :

- Une manoeuvre simple et sûre quelque soit le milieu (interne ou externe).
- Une commande de position et poursuite de trajectoire en une autonomie.
- l'utilisation de plusieurs capteurs pour la perception de l'environnement.
- la communication avec d'autres plateformes et avec une station au sol.
- l'implémentation en temps réel des algorithmes de coordination multi véhicules sur des calculateurs embarqués.

Ces quadrirotors sont dotés de moteurs Brushless Axi 2208 pour la propulsion, et ils sont capables de suivre une trajectoire prescrite en utilisant des GPS, des Unités de Mesures Inertielles et des Ultrasons pour l'altitude [4].











La commande et tout calcul sont partagés en deux niveaux : une commande bas niveau qui exécute la boucle d'asservissement en temps réel et produit les signaux de commande PWM. Cela se fait sur une carte microcontrôleurs de Robostix, basée sur le processeur Atmega 128. La planification, l'estimation et la commande de haut niveau sont effectuées sur un ordinateur embarqué Crossbow Stargate 1.0



FIGURE 1.18 – Le STARMAC

Autres :

Le tableau ci dessous résume quelques travaux de recherche [1] :

Project	University	Status	Picture
Mesicopter	Stanford	Ended	
E. Altuğ's thesis	Univ. Pennsylvania	Ended	
P. Castillo's thesis	Univ. Compiègne	Ended	
A. Clifton's thesis	Univ. Vanderbilt	Ended	
P. Pounds's thesis	ANU	in progress	
N. Guenard's thesis	CEA	in progress	
Starmac	Stanford	in progress	
M. Kemper's thesis	Univ. Oldenburg	in progress	
P. Tournier's thesis	MIT	in progress	
MD4-200®	microDrones GmbH	in progress	

1.6 Conclusion

Nous avons donné dans ce chapitre un état de l'art sur les UAV, leur historique, leur classification et nous finissons par les quadri-rotor. Dans les chapitres suivants nous allons établir le modèle dynamique du quadri-rotor ainsi que la commande synthétisée pour notre système.

Chapitre 2

Modélisation et synthèse de commande pour le quadrotor

Résumée

Dans ce chapitre nous allons faire une modélisation puis une synthèse de la loi de commande par PID et l'appliquer sur le modèle simulink du quad-copter suivie d'une étude de robustesse

2.1 Introduction

La modélisation regroupe un ensemble des techniques permettant de disposer d'une représentation mathématique du système à étudier.

Le quadri-rotor 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 robuste, on doit d'abord comprendre profondément les mouvements du système et sa dynamique. Cette compréhension n'est pas nécessaire simplement pour la conception du contrôleur, mais afin de s'assurer que les simulations de l'engin dépendront un comportement aussi proche que possible de la réalité quand la commande est appliquée.

Notre but dans ce chapitre est de faire la modélisation, de décrire rapidement l'identification des paramètres d'un quadri-rotor pour lequel nous allons synthétiser un régulateur de type PID, puis de faire une simulation des trajectoires d'UAV sous cette loi de commande et nous finissons par une étude de robustesse.

2.2 Description générale

Un quadri-rotor est un robot mobile aérien à quatre rotors défini dans l'espace par 6 DDL. Ces 4 rotors sont généralement placés aux extrémités d'une croix, et l'électronique de contrôle est habituellement placée au centre de la croix. Afin d'éviter à l'appareil de tourner sur lui-même sur son axe de lacet, il est nécessaire que deux hélices tournent dans un sens, et les deux autres dans l'autre

sens. Pour pouvoir diriger l'appareil, il est nécessaire que chaque couple d'hélice tournant dans le même sens soit placé aux extrémités opposées d'une branche de la croix, figure (2.1)

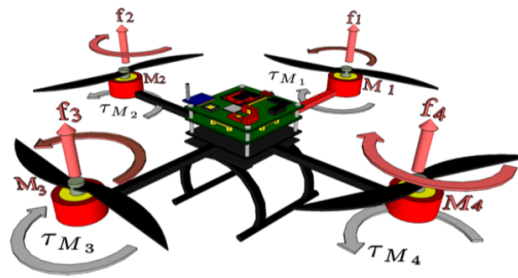


FIGURE 2.1 – Description générale du quadri-rotor

Le fonctionnement d'un quadri-rotor est assez particulier. En faisant varier astucieusement la puissance des moteurs, il est possible de le faire monter/descendre, de l'incliner à gauche/droite (roulis) ou en avant/arrière (tangage) ou encore de le faire pivoter sur lui-même (lacet), le quadri-rotor a six degrés de libertés, trois mouvements de rotation et trois mouvements de translation, ces six degrés doivent être commandés à l'aide de quatre moteurs seulement, donc c'est un système sous actionné (le nombre des entrées est inférieur au nombre des sorties) [3].

2.3 Les mouvements de quadri-rotor

Comme montré sur la figure (2.2), les moteurs avant et arrière (M_1, M_3) tournent dans le sens contraire des aiguilles d'une montre alors que les moteurs droit et gauche (M_2, M_4) tournent dans le sens des aiguilles d'une montre. Chaque actionneur produit une force F_i parallèle à son axe de rotation, ainsi qu'un couple résistant Q_i opposé au sens de rotation. La force totale ou poussée totale exercée sur l'hélicoptère (parallèle à l'axe z) est la somme des quatre forces générées par chaque actionneur ($F_T = F_1 + F_2 + F_3 + F_4$). La combinaison des forces F_i et des couples résistants Q_i donne origine aux mouvements angulaires autour des axes principaux du drone [21] :

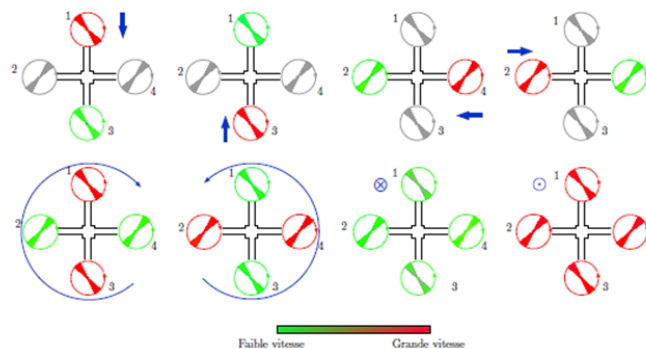


FIGURE 2.2 – Principe de fonctionnement du quadri-rotor

2.3.1 Mouvement de rotation

- **Mouvement de roulis** (ϕ) : Ce mouvement est assuré par la différence des forces (F_2, F_4) produites par les actionneurs droit et gauche. Cette différence de forces produit un couple τ_ϕ autour de l'axe x [12].

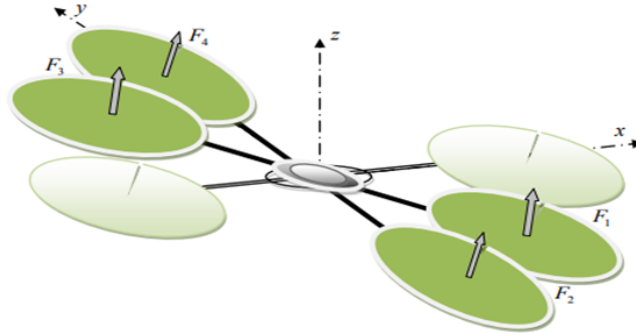


FIGURE 2.3 – Illustration du mouvement de roulis

- **Mouvement de tangage** (θ) : Ce mouvement est assuré par la différence des forces (F_1, F_3) produites par les actionneurs avant et arrière. Cette différence de forces produit un couple τ_θ autour de l'axe y [12].

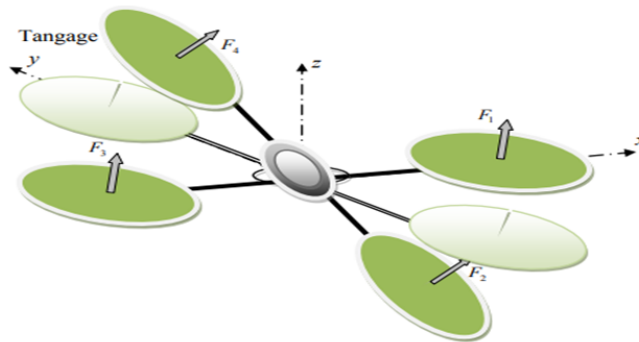


FIGURE 2.4 – Illustration du mouvement de tangage

- **Mouvement de lacet** (ψ) : Ce mouvement est assuré par la somme des couples de traînée (Q_i) produits par les quatre actionneurs. Étant donné que les sens de rotation des actionneurs (M_1, M_3) et (M_2, M_4) sont opposés, nous pouvons régler la somme des quatre couples résistants. Quand les quatre rotors tournent à la même vitesse, ils sont soumis au même couple résistant, donc la somme est nulle. Par conséquent, il n'y a pas de rotation autour de l'axe z. Par contre, si nous provoquons une différence de vitesse entre les moteurs tournant en sens opposé, les couples résistants provoquent un couple τ_ψ autour de l'axe z, provoquant ainsi la rotation de l'engin [12].

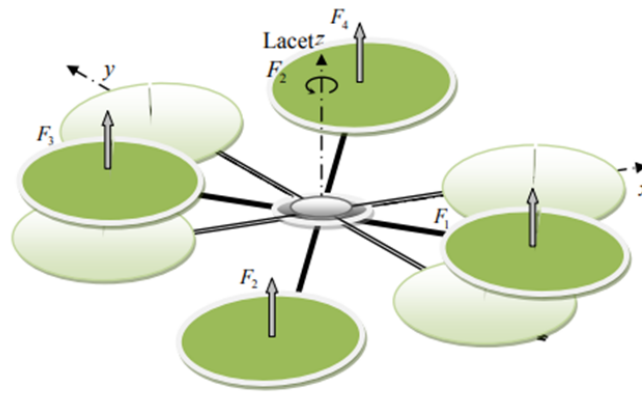


FIGURE 2.5 – Illustration du mouvement de lacet

2.3.2 Les translations verticales

Elles sont définies lorsque le quadri-rotor se déplace suivant l'axe z_n . En l'absence de perturbations, la force de poussée F_T est toujours verticale et en montée elle est toujours supérieure au poids du quadri-rotor ($F_T > mg$) tandis qu'en descente elle est inférieure ($F_T < mg$). La combinaison de l'inclinaison ($\phi \neq 0$ et /ou $\theta = 0$) du quadri-rotor et de la force de poussée F_T produit une composante de la force suivant l'axe x_n et/ou y_n . Cette force est connue comme la force de traction et celle-ci assure la translation du système dans la direction du vol souhaitée. Par conséquent, les translations verticales sont aussi définies quand le quadri-rotor se déplace dans deux directions simultanément, par exemple dans les plans $x_n z_n$ ou $y_n z_n$ [12].

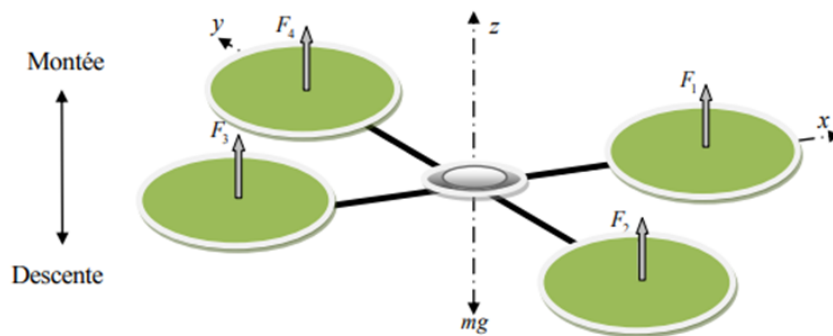


FIGURE 2.6 – Illustration du mouvement vertical

2.3.3 Les translations horizontales

Elles sont définies de façon similaire aux translations verticales mais cette fois-ci dans le plan $x_n y_n$. Lorsqu'une translation est effectuée suivant la direction x_n et la force de poussée maintient le système à une hauteur constante par rapport au sol. Le système effectue un vol connu dans la littérature sous le nom de "vol en palier".

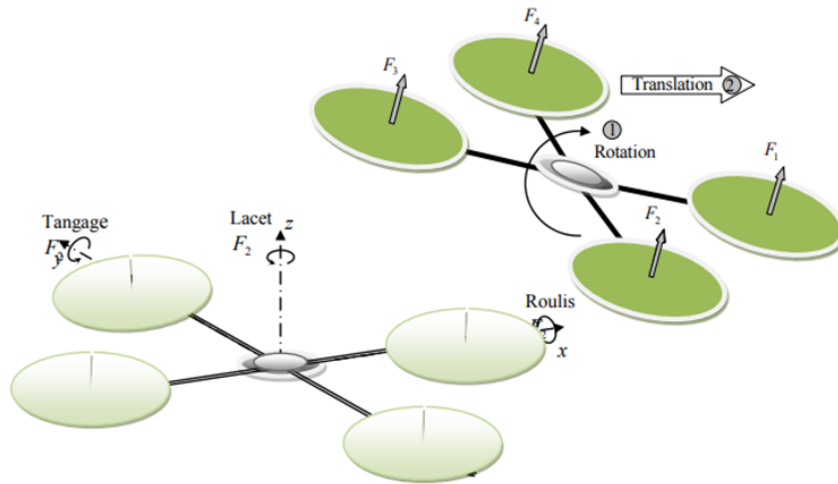


FIGURE 2.7 – Illustration du mouvement de translation

2.3.4 Le vol stationnaire

En montée verticale et après avoir franchi le seuil définissant l'effet du sol, le quadri-rotor peut rester en vol stationnaire à une certaine hauteur constante par rapport au sol en ayant une vitesse de translation nulle. La force de sustentation doit alors équilibrer le poids mg du quadri-rotor. Cette force de sustentation correspond à une force de poussée F_T qui est orientée (en l'absence de perturbation) dans la direction de l'axe z . Dans ce mode de vol, le quadri-rotor a la liberté de faire des rotations autour de l'axe z qui, dans ce cas, coïncide avec l'axe z_n du système de coordonnées inertiel [12].

2.4 Modélisation dynamique du quadri-rotor

La modélisation des robots volants est une tâche délicate puisque la dynamique du système est fortement non linéaire et pleinement couplée. Afin de pouvoir comprendre au mieux le modèle dynamique développé ci-dessous, voilà les différentes hypothèses de travail [19] :

- La structure du quadri-rotor est supposée rigide et symétrique, ce qui induit que la matrice d'inertie sera supposée diagonale.
- Les hélices sont supposées rigides pour pouvoir négliger l'effet de leur déformation lors de la rotation.

2.4.1 Effets physiques agissants sur la quadri-rotor

Les forces :

— **Le poids :**

La force de gravité est donnée par :

$$P = -mgz_0 \tag{2.1}$$

avec : m la masse totale du véhicule et g la constante d'accélération de la pesanteur

- **La force de traînée** : C'est la résultante des forces qui s'opposent au mouvement du quadri-rotor dans l'air, de même direction que le mouvement du quadri-rotor mais de sens opposé. Elle représente en quelque sorte les forces de frottement visqueux sur l'objet. Nous considérons ici, la force de traînée des hélices donnée par

$$F_t = d\omega_i^2 \tag{2.2}$$

Où d est le coefficient de drag, il dépend de la construction de l'hélice.

ω_i est la vitesse angulaire du moteur i.

- **La force de portance** : Elle est perpendiculaire à l'écoulement d'air, dirigée vers le haut c'est-à-dire qu'elle a une tendance à faire élever le quadri-rotor.

Elle représente la force totale produite par les quatre hélices, elle est donnée par :

$$F_p = \left(\sum_{i=1}^4 f_i\right)z_1 \tag{2.3}$$

Où :

F_i est la force de portance produite par la rotation de l'hélice i, elle est donnée par :

$$f_i = b\omega_i^2 \tag{2.4}$$

Avec : b coefficient de portance.

Donc la force de poussée totale est :

$$F_p = b(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2)z_1 \tag{2.5}$$

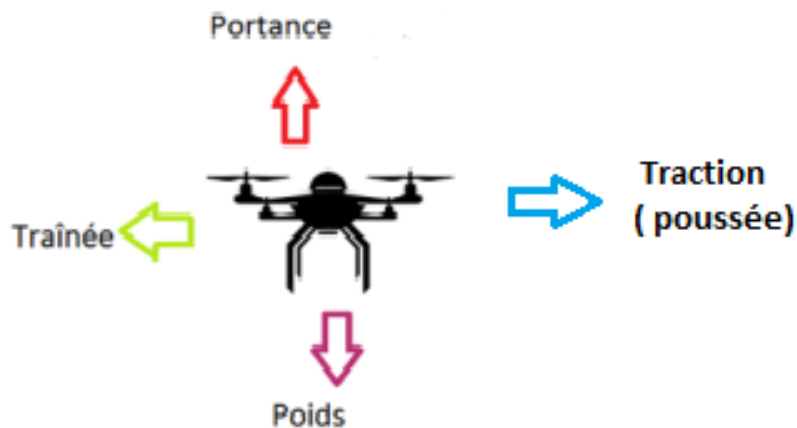


FIGURE 2.8 – Principales forces agissantes sur le drone

Les moments :

Il y a plusieurs moments agissants sur le quadri-rotor, ces moments sont dus aux forces de poussée et de traînée et aux effets gyroscopiques [12] [19].

— **Moments dus aux forces de poussée :**

- La rotation autour de l'axe x : elle est due au moment créé par la différence entre les forces de portance des rotors 2 et 4, ce moment est donné par la relation suivante :

$$M_x = l(F_4 - F_2) = lb(\omega_4^2 - \omega_2^2) \quad (2.6)$$

Avec :

l est la longueur du bras entre le rotor et le centre de gravité du quadri-rotor

- La rotation autour de l'axe y : elle est due au moment créé par la différence entre les forces de portance des rotors 1 et 3, ce moment est donné par la relation suivante :

$$M_y = l(F_3 - F_1) = lb(\omega_3^2 - \omega_1^2) \quad (2.7)$$

— **Moments dus aux forces de traînée :**

- La rotation autour de l'axe z : elle est due à un couple réactif provoqué par les couples de traînée dans chaque hélice, ce moment est donné par la relation suivante :

$$M_z = d(\omega_1^2 - \omega_2^2 - \omega_3^2 - \omega_4^2) \quad (2.8)$$

Moment résultant des frottements aérodynamiques, il est donné par :

$$M_a = K_{fa}\Omega^2 \quad (2.9)$$

avec :

K_{fa} : Le coefficient des frottements aérodynamiques et Ω est la vitesse angulaire

Effet gyroscopique :

L'effet gyroscopique se définit comme la difficulté de modifier la position ou l'orientation du plan de rotation d'une masse tournante. L'effet gyroscopique est ainsi nommé en référence au mode de fonctionnement du gyroscope, appareil de contrôle de mouvement utilisé dans l'aviation (du grec gyro qui signifie rotation et scope, observer). Dans notre cas il y a deux moments gyroscopiques, le premier est le moment gyroscopique des hélices, l'autre est le moment gyroscopique dû aux mouvements du quadri-rotor.

— **Moment gyroscopique des hélices :** il est donné par la relation suivante :

$$M_{gh} = \sum_{i=1}^4 \Omega \wedge J_r \left[0 \quad 0 \quad (-1)^{i+1} \omega_i \right]^T \quad (2.10)$$

avec : J_r est l'inertie des rotors.

— **Moment gyroscopique dû aux mouvements du quadri-rotor :** il est donné par la relation suivante :

$$M_{gm} = \Omega \wedge J\Omega \quad (2.11)$$

avec : J est l'inertie du système.

2.5 Repère du quadri-rotor

2.5.1 Repères utilisés

Un quadri-rotor nécessite deux trièdres pour le repérer dans l'espace, ces repères sont :[20]

- **Le repère terrestre** : Il est noté : $R_0(O_0, X_0, Y_0, Z_0)$. C'est un repère lié à la terre Figure(2.9) supposé immobile.
- **Le repère lié au corps du quadri-rotor** : Le repère lié au corps du quadri-rotor est noté : $R_1(O_1, X_1, Y_1, Z_1)$. C'est un repère dont l'origine O_1 coïncide avec le centre de gravité G du quadri-rotor. Donc les paramètres qui permettent de décrire le mouvement du quadri-rotor sont : $(\phi, \theta, \psi, x, y, z, V, \Omega)$ avec :
 - **(angle de roulis)** : rotation autour de X_1 $(-\pi < \phi < \pi)$
 - **(angle de tangage)** : rotation autour de Y_1 $(-2\pi < \theta < 2\pi)$
 - **(angle de lacet)** : rotation autour de Z_1 $(-\pi < \psi < \pi)$
 - x : coordonnée du centre de gravité G du quadri-rotor suivant X_0 .
 - y : coordonnée du centre de gravité G du quadri-rotor suivant Y_0 .
 - z : coordonnée du centre de gravité G du quadri-rotor suivant Z_0 .
 - Ω : $\begin{bmatrix} p & q & r \end{bmatrix}^T \in R_0$: la vitesse de rotation du quadri-rotor par rapport au repère inertiel.
 - V : $\begin{bmatrix} u & v & w \end{bmatrix}^T \in R_0$: la vitesse linéaire du quadri-rotor par rapport au repère inertiel.

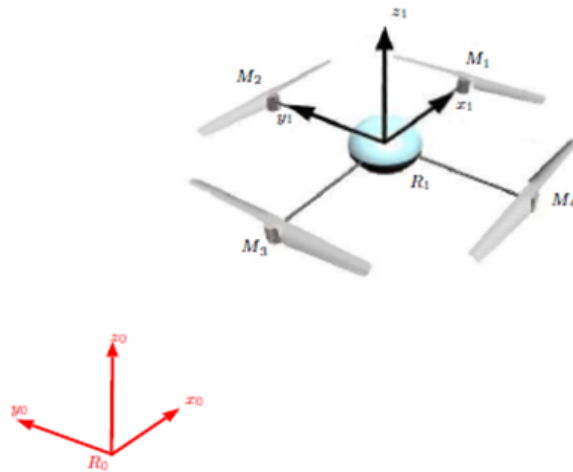


FIGURE 2.9 – Repérage du quadri-rotor

2.5.2 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_0 . Trois paramètres indépendants sont nécessaires pour décrire complètement l'orientation du repère R_1 par rapport à celle de R_0 . Le passage du repère

R_1 vers le repère R_0 se fera par trois rotations en utilisant deux repères intermédiaires R_i et R_j [11].

•**Passage du repère R_0 vers le repère R_i :**

La rotation se fait autour de l'axe $x_i = x_0$. On passe du repère R_0 vers R_i en faisant une rotation d'angle ϕ appelé angle de roulis Figure(2.10).

La représentation se fait par des figures planes, à partir desquelles nous construisons les matrices de passage, nous avons ainsi la matrice :

$$R(X_0, \phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \sin(\phi) & \cos(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \tag{2.12}$$

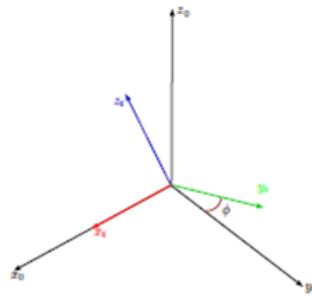


FIGURE 2.10 – Rotation autour de l'axe X (Roulis)

•**Passage du repère R_i vers le repère R_j :**

La rotation se fait autour de l'axe $y_j = y_i$. On passe du repère R_i vers le repère R_j en faisant une rotation d'angle θ appelé angle de tangage.

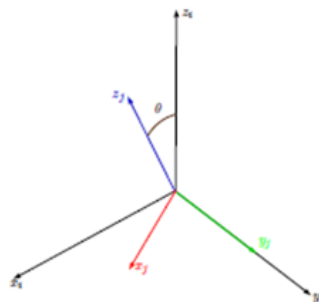


FIGURE 2.11 – Rotation autour de l'axe Y (Tangage)

Nous avons ainsi la matrice :

$$R(Y_0, \theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2.13}$$

•Passage du repère R_j vers le repère R_1 :

La rotation se fait autour de l'axe $z_1 = z_j$. On passe du repère R_j vers le repère R_1 en faisant une rotation d'angle ψ appelé angle du lacet

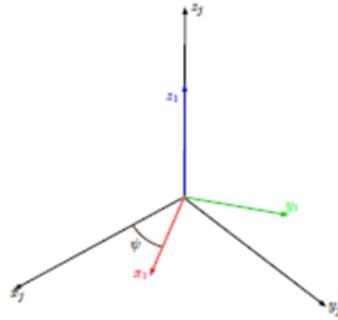


FIGURE 2.12 – Rotation autour de l'axe Z (Lacet)

Nous avons ainsi la matrice :

$$R(Z_0, \psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

Le passage du repère R_1 vers le repère R_0 ou inversement se fait par trois rotations successives de telle sorte que tous les axes de R_1 occupent des positions différentes de celle de R_0 .

La matrice de passage de R_1 vers R_0 est donnée par le produit des trois matrices successives on obtient :

$$R = R(\phi, \theta, \psi) = R(Z_0, \psi) \times R(Y_0, \theta) \times R(X_0, \phi) \tag{2.15}$$

$$R = \begin{bmatrix} c\theta c\psi & -c\phi s\psi + s\phi s\theta c\psi & s\phi s\psi + c\phi s\theta c\psi \\ c\theta s\psi & c\phi c\psi + s\phi s\theta s\psi & -s\phi c\psi + c\phi s\theta s\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \tag{2.16}$$

Avec : $C(\theta) = \cos(\theta), s(\psi) = \sin(\psi)$

On peut exprimer la force de portance dans le repère inertiel en utilisant la matrice de rotation R

$$F_f = R \times \begin{bmatrix} 0 & 0 & F_p \end{bmatrix} \tag{2.17}$$

2.6 Vitesses angulaires

Les vitesses de rotations, $\Omega_1, \Omega_2, \Omega_3$ dans le repère fixe sont exprimées en fonction des vitesses de rotations dans le repère mobile, nous avons [2] :

$$\Omega = \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + Rot_x(\phi)^{-1} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + (Rot_y(\theta)Rot_x(\phi))^{-1} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \quad (2.18)$$

En effet, la rotation en roulis a lieu lorsque les repères sont encore confondus. Puis, en ce qui concerne le tangage, le vecteur représentant la rotation doit être exprimé dans le repère fixe : il est donc multiplié par $Rot_x(\phi)^{-1}$. De même, le vecteur représentant la rotation en lacet doit être exprimé dans le repère fixe qui a déjà subies deux rotations. Nous arrivons ainsi à :

$$\begin{aligned} \Omega = \begin{bmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{bmatrix} &= \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \dot{\theta} \cos(\phi) \\ -\dot{\theta} \sin(\phi) \end{bmatrix} + \begin{bmatrix} -\dot{\psi} \sin(\theta) \\ \dot{\psi} \sin(\phi) \cos(\theta) \\ \dot{\psi} \cos(\phi) \cos(\theta) \end{bmatrix} = \begin{bmatrix} \dot{\phi} - \dot{\psi} \sin(\theta) \\ (\dot{\theta} \cos(\phi) + \dot{\psi} \sin(\phi) \cos(\theta)) \\ (\dot{\psi} \cos(\phi) \cos(\theta) - \dot{\theta} \sin(\phi)) \end{bmatrix} \\ \Omega &= \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi) \cos(\theta) \\ 0 & -\sin(\phi) & \cos(\phi) \cos(\theta) \end{bmatrix} \times \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \end{aligned} \quad (2.19)$$

Quand le quadri-rotor fait des petites rotations, on peut faire les approximations suivantes :

$$c(\phi) = c(\theta) = c(\psi) = 1, s(\phi) = s(\theta) = s(\psi) = 0$$

Donc la vitesse angulaire sera :

$$\Omega = [\dot{\phi} \quad \dot{\theta} \quad \dot{\psi}]^T$$

2.7 Vitesses linéaires

Les vitesses linéaires v_x^b, v_y^b, v_z^b dans le repère fixe en fonction des vitesses linéaires v_x^m, v_y^m, v_z^m dans le repère mobile sont données par [2] :

$$V = \begin{bmatrix} v_x^b \\ v_y^b \\ v_z^b \end{bmatrix} = R \times \begin{bmatrix} v_x^m \\ v_y^m \\ v_z^m \end{bmatrix} \quad (2.20)$$

2.8 Développement du Modèle mathématique selon Newton-Euler

En utilisant la formulation de Newton-Euler, les équations sont écrites sous la forme suivante [19] :

$$\begin{cases} \dot{\epsilon} = v \\ m\ddot{\epsilon} = F_f + F_g \\ \dot{R} = RS(\Omega) \\ J\dot{\Omega} = -\Omega \wedge J\Omega + M_f - M_a - M_{gh} \end{cases}$$

Avec :

ϵ : est le vecteur de position du quadri-rotor

m : la masse totale du quadri-rotor

Ω : La vitesse angulaire exprimée dans le repère fixe

R : la matrice de rotation

\wedge : Le produit vectoriel

J : la matrice d'inertie symétrique de dimension (3×3) , elle est donnée par :

$$J = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix}$$

$S(\Omega)$: est la matrice antisymétrique ; pour un vecteur de vélocité $\Omega = [\Omega_1 \quad \Omega_2 \quad \Omega_3]$ elle est donnée par :

$$S(\Omega) = \begin{bmatrix} 0 & -\Omega_3 & \Omega_2 \\ \Omega_3 & 0 & -\Omega_1 \\ -\Omega_2 & \Omega_1 & 0 \end{bmatrix}$$

F_f : est la force totale générée par les quatre rotors, elle est donnée par :

$$F_f = R \times \left[0 \quad 0 \quad \sum_{i=1}^4 F_i \right]^T, \text{ avec : } F_i = b\omega_i^2$$

F_g : force de gravité, elle est donnée par :

$$F_g = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix}^T$$

M_f : moment provoqué par les forces de poussée et de traînée :

$$M_f = \left[l(F_4 - F_2) \quad l(F_3 - F_1) \quad d(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \right]^T$$

Équations de mouvement de translation : d : coefficient de drag.

On a : $m\ddot{\mathbf{e}} = F_f + F_g$

On remplace chaque force par sa formule, on trouve :

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} c\phi c\psi s\theta + s\phi s\psi \\ c\phi s\theta s\psi - s\phi c\psi \\ c\phi c\theta \end{bmatrix} \times \sum_{i=1}^4 F_i - \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \quad (2.21)$$

On obtient alors les équations différentielles qui définissent le mouvement de translation :

$$\begin{cases} \ddot{x} = \frac{1}{m}(c\phi c\psi s\theta + s\phi s\psi)(\sum_{i=1}^4 F_i) \\ \ddot{y} = \frac{1}{m}(c\phi s\theta s\psi - s\phi c\psi)(\sum_{i=1}^4 F_i) \\ \ddot{z} = \frac{1}{m}(c\phi c\theta)(\sum_{i=1}^4 F_i) - g \end{cases}$$

on pose :

$U_1 = F_i$ donc :

$$U_1 = b(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2)$$

2.9 Equations de mouvement de rotation

On a :

$$J\dot{\Omega} = -M_{gm} - M_{gh} + M_f$$

On remplace chaque moment par la formule correspondant, on trouve :

$$\begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \wedge \left(\begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \right) - \begin{bmatrix} J_r g(u) \dot{\theta} \\ J_r g(u) \dot{\phi} \\ 0 \end{bmatrix} + \begin{bmatrix} lb(\omega_4^2 - \omega_2^2) \\ lb(\omega_3^2 - \omega_1^2) \\ d(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{bmatrix} \quad (2.22)$$

On obtient alors les équations différentielles définissant le mouvement de rotation :

$$\begin{cases} \ddot{\phi} = \frac{J_{yy} - J_{zz}}{J_{xx}} \dot{\theta} \dot{\phi} - \frac{J_r g(u)}{J_{xx}} \dot{\theta} + \frac{U_2}{J_{xx}} \\ \ddot{\theta} = \frac{J_{zz} - J_{xx}}{J_{yy}} \dot{\phi} \dot{\psi} + \frac{J_r g(u)}{J_{yy}} \dot{\phi} + \frac{U_3}{J_{yy}} \\ \ddot{\psi} = \frac{J_{xx} - J_{yy}}{J_{zz}} \dot{\phi} \dot{\theta} + \frac{U_4}{J_{zz}} \end{cases}$$

avec :

$$g(u) = -\omega_1 + \omega_2 - \omega_3 + \omega_4$$

$$U_2 = lb(\omega_4^2 - \omega_2^2)$$

$$U_3 = lb(\omega_3^2 - \omega_1^2)$$

$$U_4 = d(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2)$$

En conséquence, le modèle dynamique complet qui régit le quadri-rotor est le suivant :

$$\begin{cases} \ddot{\phi} = \frac{J_{yy}-J_{zz}}{J_{xx}} \dot{\theta} \dot{\phi} - \frac{J_r g(u)}{J_{xx}} \dot{\theta} + \frac{U_2}{J_{xx}} \\ \ddot{\theta} = \frac{J_{zz}-J_{xx}}{J_{yy}} \dot{\phi} \dot{\psi} + \frac{J_r g(u)}{J_{yy}} \dot{\phi} + \frac{U_3}{J_{yy}} \\ \ddot{\psi} = \frac{J_{xx}-J_{yy}}{J_{zz}} \dot{\phi} \dot{\theta} + \frac{U_4}{J_{zz}} \\ \ddot{x} = \frac{1}{m} (c\phi c\psi s\theta + s\phi s\psi) U_1 \\ \ddot{y} = \frac{1}{m} (c\phi s\theta s\psi - s\phi c\psi) U_1 \\ \ddot{z} = \frac{1}{m} (c\phi c\theta) (\sum_{i=1}^4 F_i) - g \end{cases}$$

avec :

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ 0 & -lb & 0 & lb \\ -lb & 0 & lb & 0 \\ -d & d & -d & d \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}$$

2.10 Représentation d'état du système

Pour un système physique il existe une multitude de représentations d'état, dans notre cas on choisit le vecteur d'état comme suit [19] :

$$\begin{aligned} X &= [\phi \quad \dot{\phi} \quad \theta \quad \dot{\theta} \quad \psi \quad \dot{\psi} \quad x \quad \dot{x} \quad y \quad \dot{y} \quad z \quad \dot{z}]^T \\ &= [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_8 \quad x_9 \quad x_{10} \quad x_{11} \quad x_{12}]^T \end{aligned}$$

on obtient la représentation d'état suivante :

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \frac{J_{yy}-J_{zz}}{J_{xx}} x_4 x_6 - \frac{J_r g(u)}{J_{xx}} x_4 + \frac{U_2}{J_{xx}} \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = \frac{J_{zz}-J_{xx}}{J_{yy}} x_2 x_6 + \frac{J_r g(u)}{J_{yy}} x_2 + \frac{U_3}{J_{yy}} \\ \dot{x}_5 = x_6 \\ \dot{x}_6 = \frac{J_{xx}-J_{yy}}{J_{zz}} x_2 x_4 + \frac{U_4}{J_{zz}} \\ \dot{x}_7 = x_8 \\ \dot{x}_8 = \frac{1}{m} (\cos(x_1) \sin(x_3) \cos(x_5) + \sin(x_1) \sin(x_5)) U_1 \\ \dot{x}_9 = x_{10} \\ \dot{x}_{10} = \frac{1}{m} (\cos(x_1) \sin(x_3) \sin(x_5) - \sin(x_1) \cos(x_5)) U_1 \\ \dot{x}_{11} = x_{12} \\ \dot{x}_{12} = \frac{1}{m} (\cos(x_1) \cos(x_3)) - g \end{cases}$$

2.11 Identification des paramètres

Dans cette partie, nous nous sommes inspirés d'un travail fait l'année dernière dans notre laboratoire, où ils ont appliqué différentes méthodes d'identification pour déterminer les valeurs des paramètres du modèle du quadrotor [2]. Leur procédure a permis d'identifier des valeurs qui ont donné un comportement en simulation très proche du comportement réel. Nous rappelons brièvement cette procédure et résumons les valeurs dans un tableau.

Ces valeurs seront utilisées par la suite dans la synthèse de la loi de commande stabilisante.

- **Masse totale** : pour cette étape, après avoir terminé le montage d'UAV, une balance électrique est utilisée pour mesurer la masse du drone, cette masse est égale à 1218g.

- **Coefficient de portance "b"** : le datasheet du moteur illustré sur la figure (2.13) ci-dessous, est utilisé pour calculer le coefficient de portance « b ».

SPECS for Turnigy Multistar 2216-800Kv 14Pole
Multi-Rotor Outrunner V2

Battery	Prop Size	RPM	Thrust (g)	Current (A)	Power
11.1V (3Cell)	12" Slow Flier	5,500	907	17	185
	11" thin Style	7,200	908	11.7	128
	11" Slow Flier	6,000	908	16.2	171
	10" Slow Flier	7,440	771	10.8	120
	9" Slow Flier	8,000	544	8.9	105

FIGURE 2.13 – Datasheet du moteur

- **Coefficient de drag "d"** : est calculé d'après la relation de l'aérodynamique qui relie le coefficient de traîné et le coefficient de portance.

- **La matrice d'inertie "J"** : la méthode expérimentale de l'expérience du pendule bifilaire est appliquée pour déterminer l'inertie.

Les valeurs obtenues sont résumés dans le tableau suivant [2] :

TABLE 2.1 – Paramètres obtenus par identification

Paramètre	Valeur	Unité
m	1218	g
g	9,81	$m.s^{-1}$
l	0.25	m
J_{xx}	1.49×10^{-2}	$Kg.m^2$
J_{yy}	1.49×10^{-2}	$Kg.m^2$
J_{zz}	2.94×10^{-2}	$Kg.m^2$
J_r	1.27^{-5}	$Kg.m^2$
b	15.65×10^{-5}	$N.sec^2$
d	6×10^{-4}	$N.m.sec^2$

2.12 Synthèse du régulateur PID

Dans la zone industrielle, les régulateurs les plus utilisés sont les PID. Les raisons de ce succès sont principalement trois :

- structure simple
- bonnes performances pour plusieurs processus
- accordable même sans modèle spécifique du système piloté.

En robotique, la technique PID représente les bases du contrôle. Même si de nombreux algorithmes différents offrent de meilleures performances que le PID, cette structure est souvent choisie pour les raisons exposées ci-dessus. La structure PID traditionnelle est composée de l'addition de trois contributions, Figure(2.14).

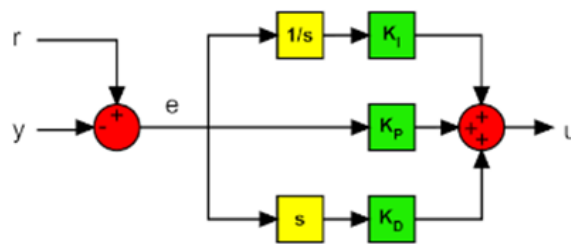


FIGURE 2.14 – Structure PID traditionnel

Les blocs «1/s» et «s» représentent les opérations d'intégration et de dérivation. donc on peut écrire l'expression du correcteur sous la forme suivante :

$$U(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}$$

Où "U" est une variable contrôlée générique, "e" est l'erreur entre la tâche et la sortie du processus, "K_P" est le coefficient proportionnel, "K_I" est le coefficient intégral et "K_D" est le coefficient dérivé. La première contribution (P) est proportionnelle à l'erreur et définit la largeur de bande proportionnelle. À l'intérieur de cet intervalle, la sortie sera proportionnelle à l'erreur tandis qu'à l'extérieur la sortie sera minimale ou maximale. La deuxième contribution (I) varie en fonction de l'intégrale de l'erreur. Même si ce composant augmente le dépassement et le temps de stabilisation, il a une propriété unique : il élimine l'erreur en régime permanent. La troisième contribution (D) varie en fonction du domaine de l'erreur. Ce composant aide à diminuer le dépassement et le temps de réponse.

Notre système est composé de 6 PID : 3 régulateurs du mouvement dans la boucle interne et 3 autres dans la boucle externe. Les sorties des régulateurs de position sont des accélérations ($\ddot{x}, \ddot{y}, \ddot{z}$).

Si on pose : $U = \ddot{x}$, et $e_x = x - x_d$ avec $x_d = constant$

L'expression générale du PID devient comme suit [2] :

$$\ddot{e}_x(t) = K_P e_x(t) + K_i \int_0^t e_x(t)(\tau) d(\tau) + K_d \frac{de_x(t)}{dt}$$

En dérivant cette équation on obtient :

$$\ddot{e}_x(t) = K_p \dot{e}_x(t) + K_i e_x(t) + K_d \ddot{e}_x(t)$$

en régime permanent : $e = 0$ avec des gains bien choisis. Ainsi, $\ddot{x} = \ddot{x}_d$ ce qui assure le comportement souhaitable.

La régulation des accélérations angulaires est faite à partir de l'équation suivante

$$U_i = K_P e_i(t) + K_i \int_0^t e_i(\tau) d\tau + K_d \frac{de_i(t)}{dt}$$

Avec : $i = \phi, \theta, \psi$

Pour simplifier notre modèle nous négligeons l'effet gyroscopique parce qu'il n'a pas une influence sur le modèle du quadri-rotor par rapport à la rotation des moteurs. Ce qui nous donne les commandes U_2, U_3, U_4 exprimées comme suit :

$$\begin{cases} U_2 = \ddot{\phi} \times J_{xx} \\ U_3 = \ddot{\theta} \times J_{yy} \\ U_4 = \ddot{\psi} \times J_{zz} \end{cases}$$

A partir d'équation dynamique de translation l'expression de U_1 est la suivante :

$$U_1 = \frac{m(\ddot{z} + g)}{\cos(\phi) + \cos(\theta)}$$

Nous pouvons déterminer les expressions de vitesse de rotation des moteurs à partir des expressions des commandes :

$$\begin{cases} \omega_1 = \sqrt{\left| \frac{U_1 b}{4} - \frac{U_3}{2lb} - \frac{U_4}{4b} \right|} \\ \omega_2 = \sqrt{\left| \frac{U_1 b}{4} - \frac{U_2}{2lb} + \frac{U_4}{4b} \right|} \\ \omega_3 = \sqrt{\left| \frac{U_1 b}{4} + \frac{U_3}{2lb} - \frac{U_4}{4b} \right|} \\ \omega_4 = \sqrt{\left| \frac{U_1 b}{4} + \frac{U_2}{2lb} + \frac{U_4}{4b} \right|} \end{cases}$$

2.13 Contraintes de sous actionnement

Nous allons exploiter les équations dynamiques de translation pour exprimer les angles ϕ_d et θ_d en fonction de ψ_d et le vecteur d'accélération $(\ddot{x}, \ddot{y}, \ddot{z})$

avec [2] :

$$U = (\ddot{x}, \ddot{y}, \ddot{z})$$

$$U = -g_z + \frac{1}{m} R \times T$$

$$T = \begin{bmatrix} 0 & 0 & \sum_{i=1}^4 F_i \end{bmatrix}^T$$

$$g_z = \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T$$

En déplaçant le vecteur de gravité g_z à gauche et en multipliant l'équation par R^T on obtient :

$$R^T(U + g_z) = \frac{1}{m}T \quad (2.23)$$

ce qui nous donne :

$$\begin{bmatrix} c\psi c\theta & s\phi s\theta c\psi - s\psi c\phi & c\phi s\theta c\psi + s\psi s\theta \\ s\psi c\theta & s\phi s\theta s\psi + c\psi c\phi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}^T \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} + g \end{bmatrix} = \frac{1}{m} \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \quad (2.24)$$

$$\ddot{x}c\psi c\theta + \ddot{y}s\psi c\theta - (\ddot{z} + g)s\theta = 0 \quad (2.25)$$

$$\ddot{x}(s\phi s\theta c\psi - s\psi c\phi) + \ddot{y}(s\phi s\theta s\psi + c\psi c\phi) + (\ddot{z} + g)s\phi c\theta = 0 \quad (2.26)$$

$$\ddot{x}(c\phi s\theta c\psi + s\psi s\theta) + \ddot{y}(c\phi s\theta s\psi - s\phi c\psi) + (\ddot{z} + g)c\phi c\theta = \frac{1}{m}T \quad (2.27)$$

En considérant que $c\theta$ est différent de 0, on divise l'équation (2.25) par $c\theta$ l'expression de θ_d est la suivante :

$$\theta_d = \arctan\left(\frac{\ddot{x}c\psi + \ddot{y}s\psi}{\ddot{z} + g}\right) \quad (2.28)$$

On soustrait (2.26) $\cos \phi$ de (2.27) $\sin \phi$ afin d'obtenir :

$$\frac{1}{m}T s\phi = \ddot{x}s\psi - \ddot{y}c\psi \quad (2.29)$$

En manipulant l'équation(2.23) nous obtenons la relation suivante :

$$(U + g_z)^T(U + g_z) = \left(\frac{1}{m}T\right)^T\left(\frac{1}{m}T\right) \quad (2.30)$$

on obtient :

$$\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} + g)^2 = \left(\frac{1}{m}T\right)^2 \quad (2.31)$$

La combinaison entre (2.29) et (2.31) nous donne l'expression de ϕ_d :

$$\phi_d = \arcsin\left(\frac{\ddot{x} \sin \psi - \ddot{y} \cos \psi}{\sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} + g)^2}}\right) \quad (2.32)$$

2.14 Simulation du modèle

Les paramètres d'identification sont classés dans le tableau suivant :
 Le modèle Simulink de la commande du quadri-rotor est le suivant :

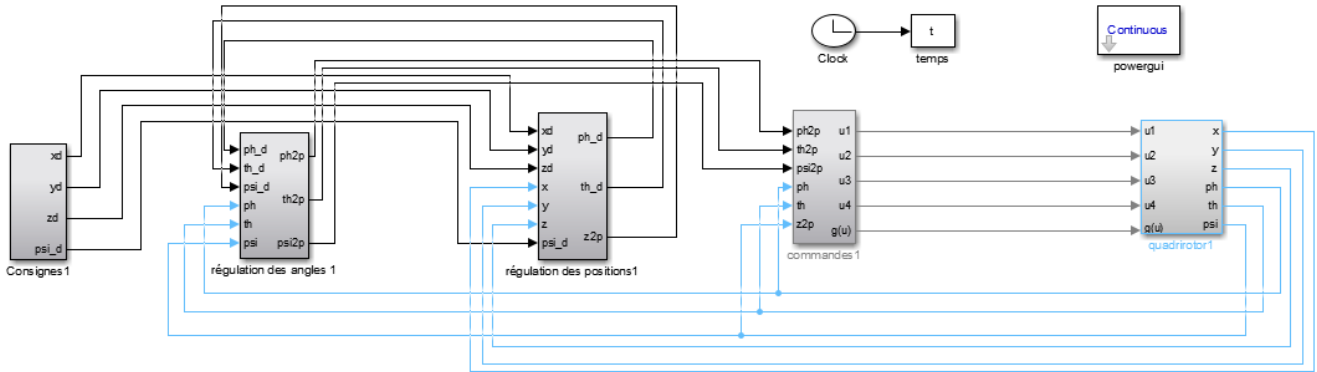


FIGURE 2.15 – Modèle *Simulink* du quadri-rotor

- Le premier bloc est le bloc des entrées contient les valeurs désirées de X, Y, Z et ψ .
- Le deuxième bloc est le bloc de régulation des angles ϕ, θ, ψ par le PID.
- Le troisième bloc nous allons trouver les consignes qui vont être comparées avec les valeurs réelles, et les réguler par un autre PID.
- Le quatrième bloc contient un couplage entre les commande U_1, U_2, U_3, U_4 et les vitesses de rotation $\omega_1, \omega_2, \omega_3, \omega_4$ à partir des équations
- Les commandes U_1, U_2, U_3, U_4 sortantes vont agir sur notre système dans le dernier bloc.

Nous avons réussi à stabiliser les trajectoires du modèle du quad-copteur et cela grâce au paramètres PID ajustés par la méthode essai-erreur.

Les gains des PID que nous avons pris sont résumés dans le tableau suivant :

TABLE 2.2 – Valeurs choisies pour les gains du PID

Paramètre	X	Y	Z	ϕ	θ	ψ
K_p	18	16.2	5	2.1	1.3	4
K_i	0	-0.0.1	0	0.01	0.01	-0.01
K_d	9	9.05	4	12	18	1.5

2.15 Résultat des simulations

Mouvements de translation :

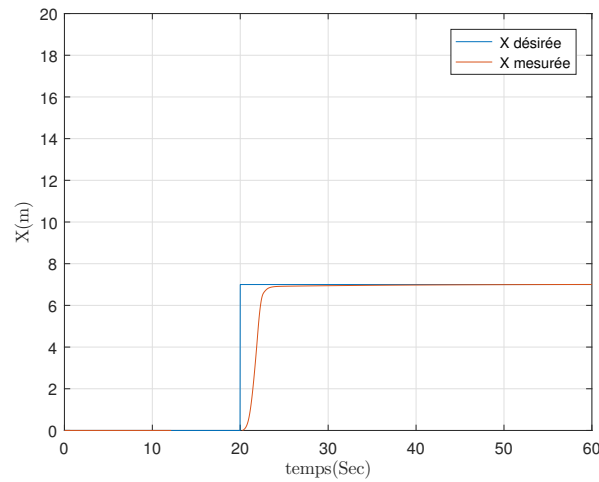


FIGURE 2.16 – Tracés de x et x_d

Nous donnons en entrée une consigne de 7m à l' instant 20, nous remarquons que la trajectoire réelle converge vers la référence en un temps de 22 sec.

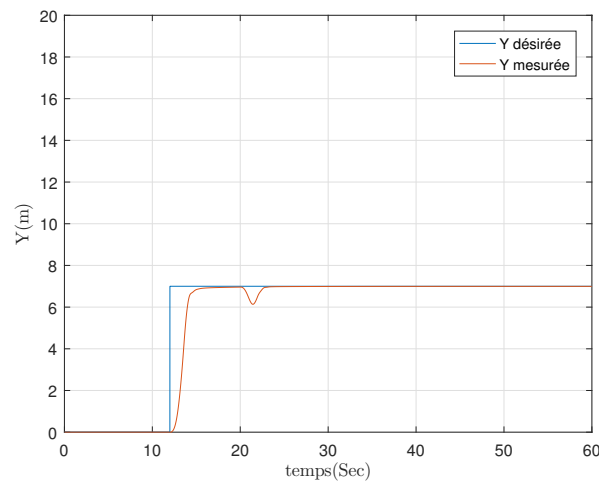


FIGURE 2.17 – Tracés de Y et Y_d

Nous donnons en entrée une consigne de 7 m à l' instant 12, nous remarquons que la trajectoire réelle converge vers la référence en temps acceptable. La perturbation dans le tracé de Y est due à l'effet de couplage suite au changement de la consigne x à l' instant 22s

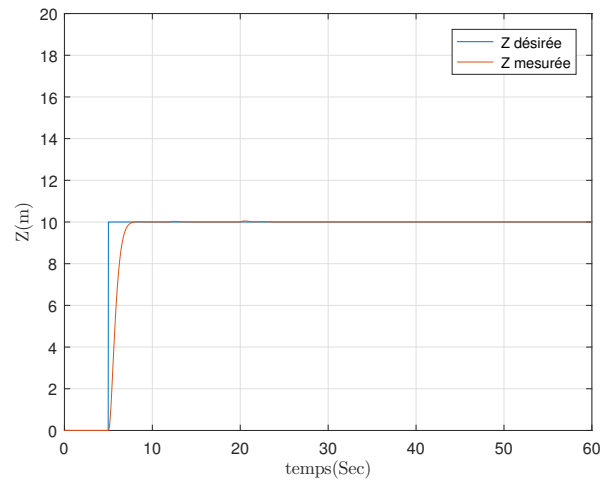


FIGURE 2.18 – Tracés de Z et Z_d

Nous donnons en entrée une consigne de 10 m à l' instant 5, nous remarquons que la trajectoire réelle converge vers la référence en un temps de 7s.

Mouvements de rotation :

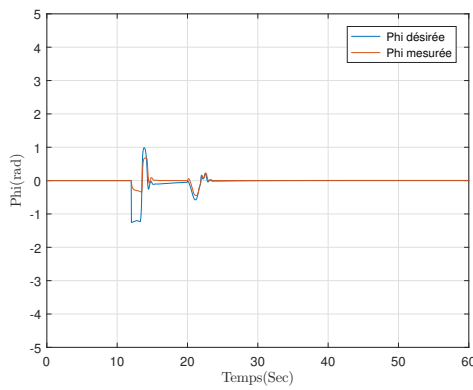


FIGURE 2.19 – Tracés de ϕ et ϕ_d

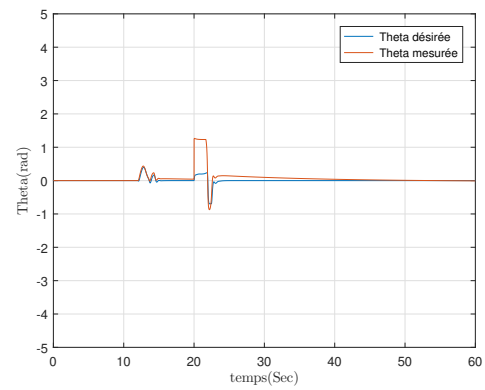


FIGURE 2.20 – Tracés de θ et θ_d

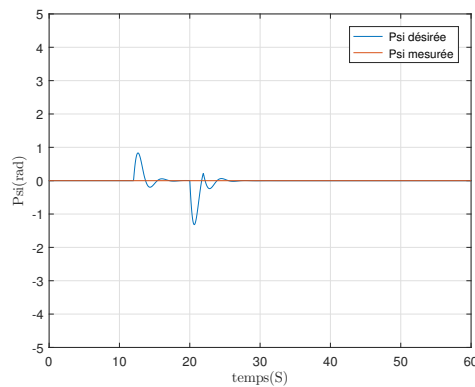


FIGURE 2.21 – Tracés de ψ et ψ_d

L'angle de lacet désiré a été fixé à 0 et ϕ_d , θ_d ont été déduit à partir des contraintes de sous-actionnement.

Nous remarquons que à chaque perturbation due aux changements de consignes, les régulateurs PID corrigent les angles et les stabilisent autour de l'origine.

2.16 Etude de la robustesse

Nous nous intéressons dans cette partie, au test de la robustesse de la commande par PID où nous allons imposer d'une part des variations paramétriques et d'autre part une force de traînée ou une résistance «vent» au mouvement de drone dans le cas de la commande du vol vertical qui entraînera une dégradation de la performance avec une consommation supplémentaire d'énergie aux niveaux des actionneurs. Pour cela nous pouvons partager notre étude de robustesse en deux parties : robustesse paramétrique et robustesse vis-à-vis d'une perturbation externe.

2.16.1 Robustesse paramétrique

Afin de tester la robustesse de la commande, nous allons changer le coefficient de traînée « d » et la masse « M ».

Nous allons effectuer deux changements paramétriques :

- sur la valeur de la masse M en considérant une variation allant jusqu'à 250% (de 1.2 à 3kg) en prévoyance d'une probable surcharge du drone lorsqu'il sera équipé avec d'autres capteurs, de caméra et de cartes électroniques supplémentaires pour effectuer par exemple le survol d'un champ agricole en vue de la mesure de l'indice de végétation NDVI.
- En plus de la variation de la masse, nous avons considéré une variation supplémentaire sur le coefficient de traînée "d" où nous avons pris un $d = 6.10^{-3}$ au lieu de 6.10^{-4} .

L'application en simulation de ces variations au quadrotor soumis à la loi de commande PID donne les résultats suivants :

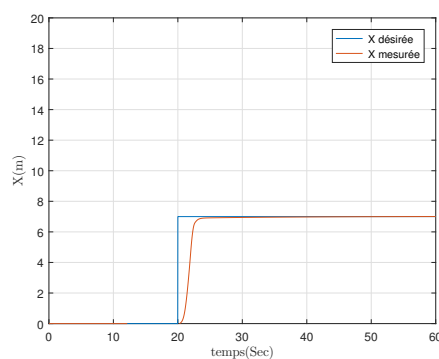


FIGURE 2.22 – Tracés de x et x_d avec la perturbation sur M

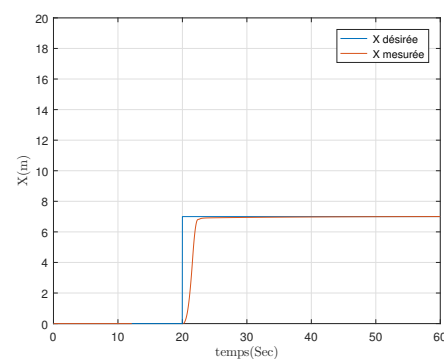


FIGURE 2.23 – Tracés de x et x_d avant la perturbation

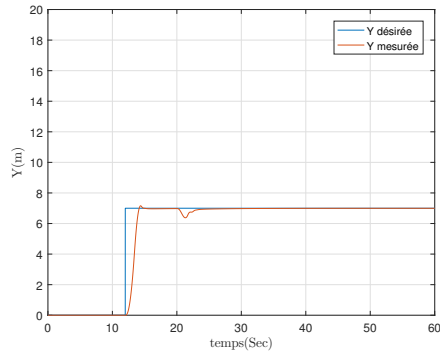


FIGURE 2.24 – Tracés de y et y_d après la perturbation sur M

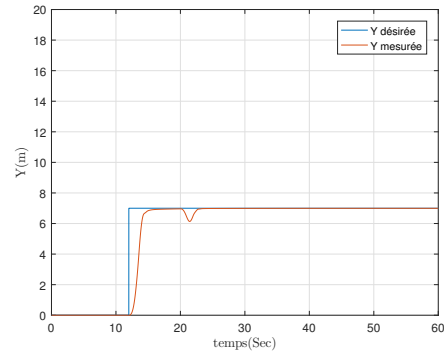


FIGURE 2.25 – Tracés de y et y_d avant la perturbation

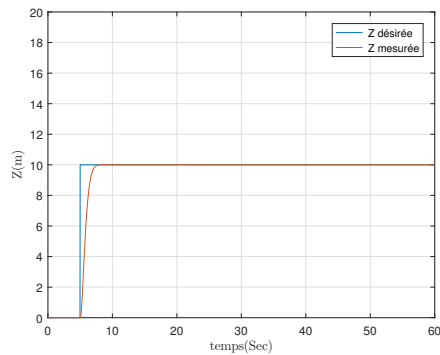


FIGURE 2.26 – Tracés de z et z_d après la perturbation sur M

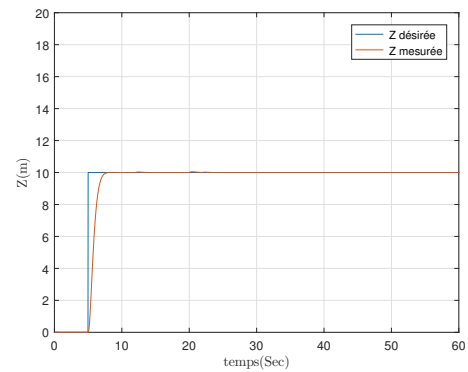


FIGURE 2.27 – Tracés de z et z_d avant la perturbation

Nous remarquons que pour la translation x et y , il y' a un léger dépassement rapidement effacé et une erreur nulle, le système suit la consigne par contre il n'y a pas une influence sur la translation z .

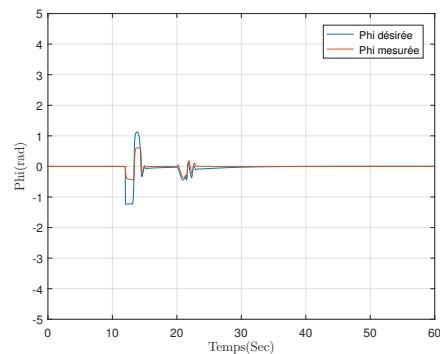


FIGURE 2.28 – Tracés de ϕ et ϕ_d après la perturbation sur M

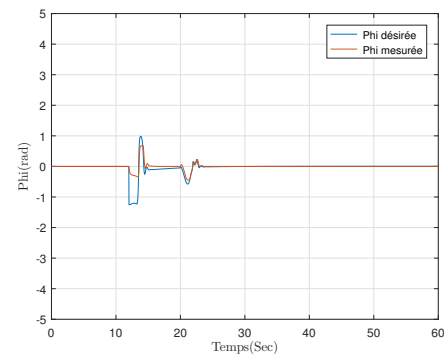


FIGURE 2.29 – Tracés de ϕ et ϕ_d avant la perturbation

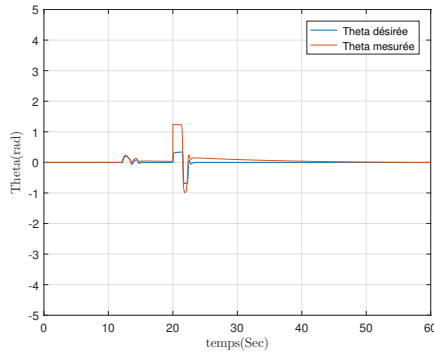


FIGURE 2.30 – Tracés de θ et θ_d après la perturbation sur M

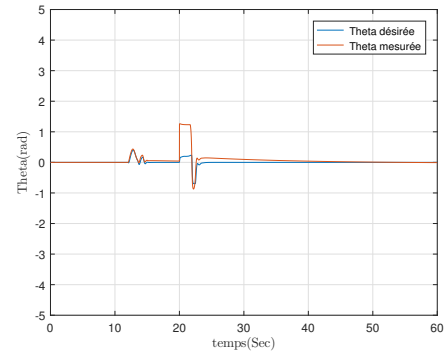


FIGURE 2.31 – Tracés de θ et θ_d avant la perturbation

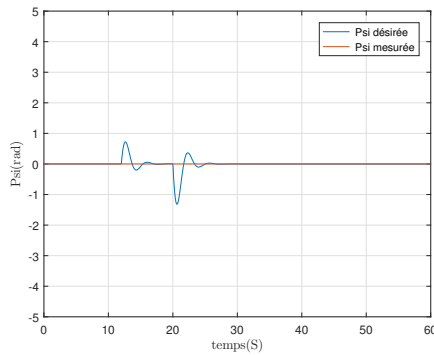


FIGURE 2.32 – Tracés de ψ et ψ_d après la perturbation sur M

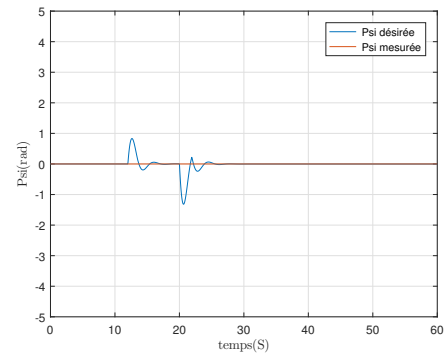


FIGURE 2.33 – Tracés de ψ et ψ_d avant la perturbation

Pour les angles de rotation nous voyons que les trajectoires de ϕ_d, θ_d, ψ_d sont modifiées, les amplitudes sont plus importantes mais convergent rapidement vers leurs consignes. Nous concluons que la commande PID est robuste par rapport à un changement important de la masse, ce qui est intéressant puisque le drone pourra supporter un poids plus important. Les figures suivantes sont relatives aux changements en même temps sur M et d :

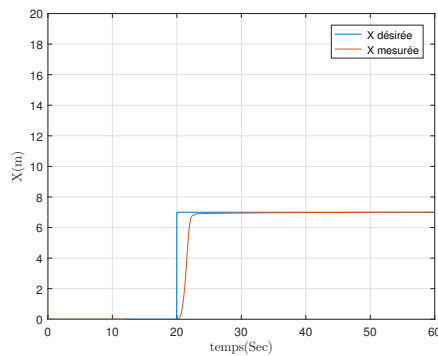


FIGURE 2.34 – Tracés de x et x_d après la perturbation sur M et d

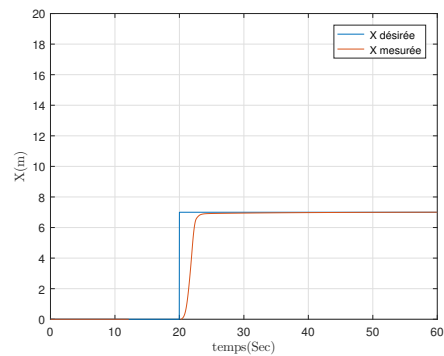


FIGURE 2.35 – Tracés de x et x_d avant la perturbation

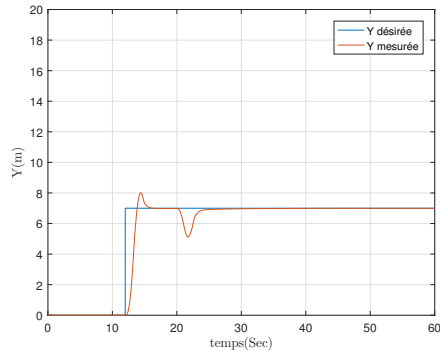


FIGURE 2.36 – Tracés de y et y_d après la perturbation sur M et d

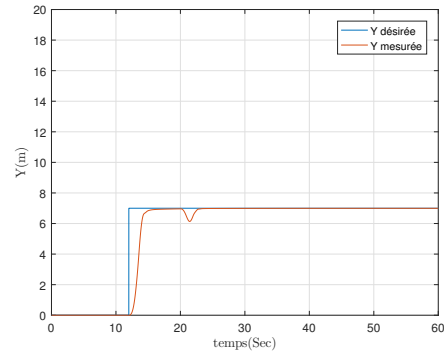


FIGURE 2.37 – Tracés de y et y_d avant la perturbation

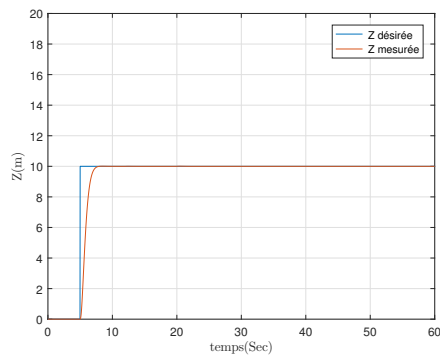


FIGURE 2.38 – Tracés de z et z_d après la perturbation sur M et d

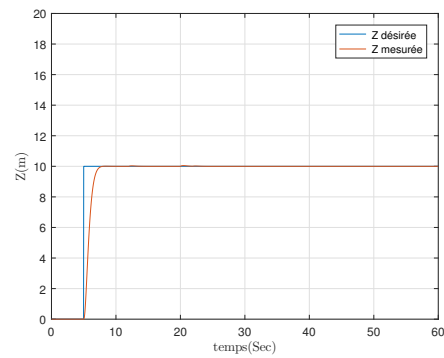


FIGURE 2.39 – Tracés de z et z_d avant la perturbation

Nous remarquons que pour la trajectoire de translation y il y'a un léger dépassement puis elle suit la référence désirée ; pas d'influence sur la trajectoire de la translation x et z .

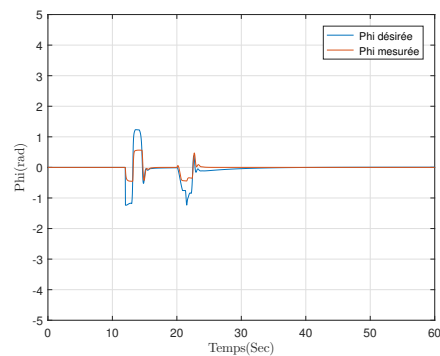


FIGURE 2.40 – Tracés de ϕ et ϕ_d après la perturbation sur M et d

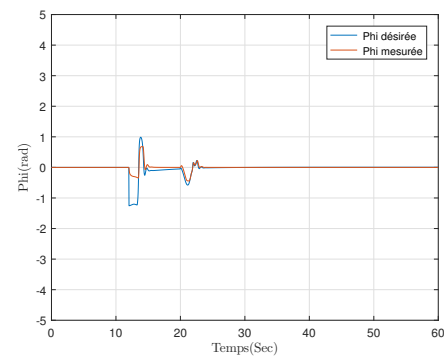


FIGURE 2.41 – Tracés de ϕ et ϕ_d avant la perturbation

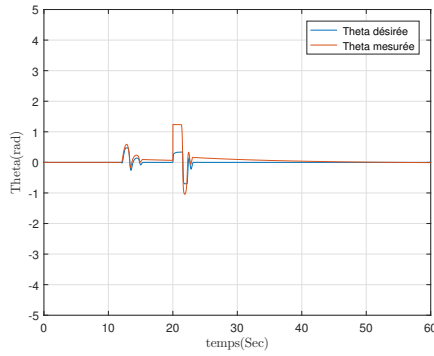


FIGURE 2.42 – Tracés de θ et θ_d après la perturbation sur M et d

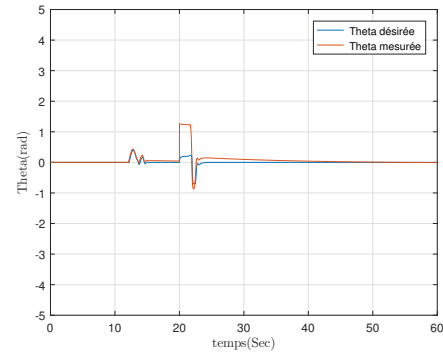


FIGURE 2.43 – Tracés de ψ et ψ_d avant la perturbation

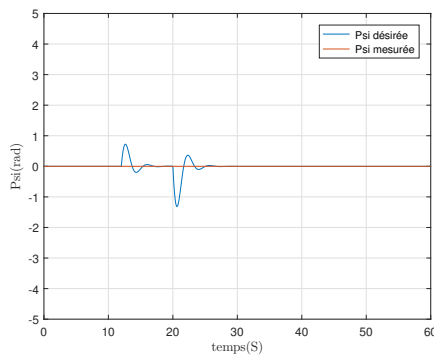


FIGURE 2.44 – Tracés de ψ et ψ_d après la perturbation sur M et d

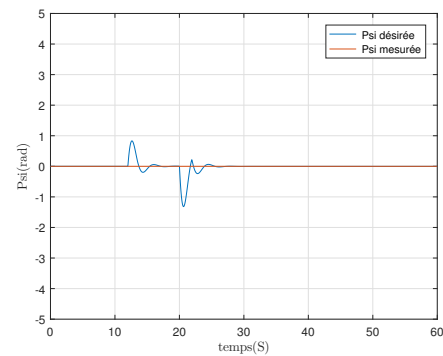


FIGURE 2.45 – Tracés de ψ et ψ_d avant la perturbation

les trajectoires de rotation ϕ_d et θ_d suivent la consigne et pour la trajectoire de rotation ψ_d il y a des oscillations puis elle suit la consigne.

Nous pouvons conclure que la commande PID est robuste pour les perturbations paramétriques.

2.16.2 Robustesse vis-à-vis une perturbation externe

Le vol des drones peut être soumis à plusieurs perturbations telles que les conditions météorologiques, les effets aérodynamiques non modélisés, la collision avec un oiseau, etc.

Pour cette étude nous nous sommes intéressés à celle de la météo où nous avons considéré le vent comme étant une perturbation externe. Nous avons ajouté un profil de vent à la simulation en faisant deux tests de simulation le premier test correspond à la situation d'un vent calme et l'autre cas d'une petite brise .

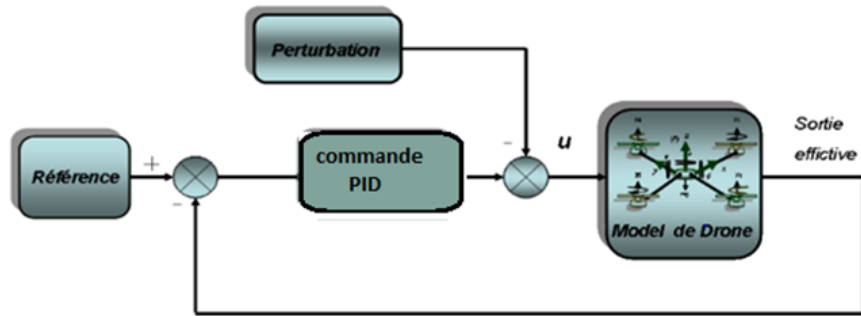


FIGURE 2.46 – Schéma global de la commande en présence de perturbation

•supposant qu'on est dans le cas d'un vent calme :

Nous ajoutons un profil de vent calme à la simulation sous forme d'un échelon fixé à 3. que signifie 3 pour notre simulation ? En météorologie, le vent est le mouvement de l'air dans le plan horizontal c'est la raison pour laquelle nous faisons une perturbation juste dans l'axe des x et non pas dans les autres axes.

Sa mesure comprend deux paramètres : sa direction et sa vitesse ou la force. L'unité internationale de la vitesse du vent est le mètre par seconde (noté m/s). En aéronautique et en météo marine, on utilise le nœud qui exprime le nombre de milles marin par heure (noté kt pour knot). L'équation suivante montre ce que représente 1 nœud en Km/h :

$$1 \text{ nœud} = 1 \text{ mille marin par heure} = 1,852 \text{ km/h} = 0,51 \text{ m/s}$$

$$\Rightarrow 3/1,852 = 1,61 \text{Kt}$$

Alors dans notre cas la vitesse du vent est 1,61Kt

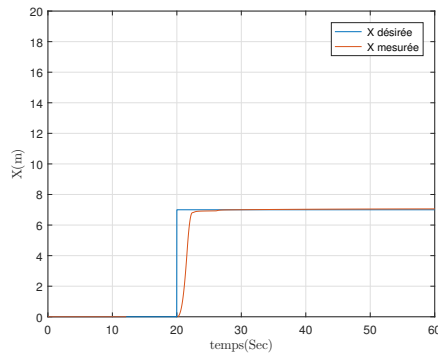


FIGURE 2.47 – Tracés de x et x_d avec un vent calme

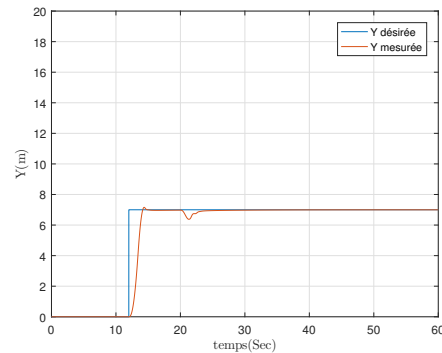


FIGURE 2.48 – Tracé de y et y_d avec un vent calme

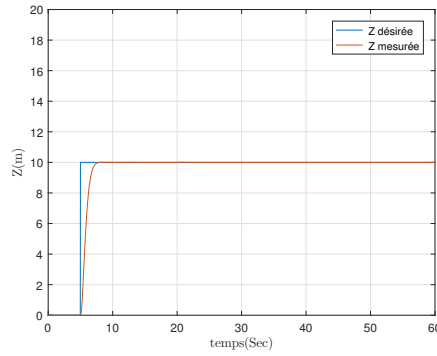


FIGURE 2.49 – Tracés de z et z_d avec un vent calme

La réponse x est très légèrement décalée de la consigne mais reste très acceptable. Par contre, pour la trajectoire y , le dépassement est plus important lié au couplage, la perturbation est par la suite parfaitement rejetée, ceci est dû au fait que le vent est directement appliqué dans la direction de x . Pas trop d'influence sur z .

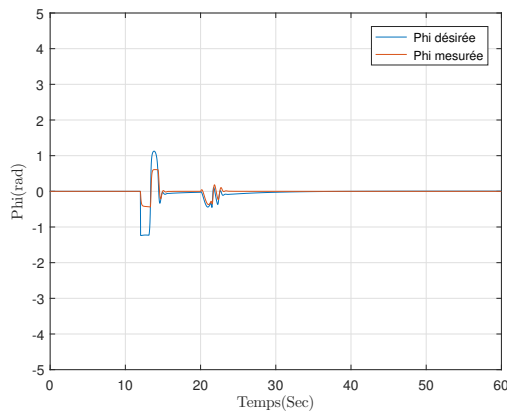


FIGURE 2.50 – Tracés de ϕ et ϕ_d avec un vent calme

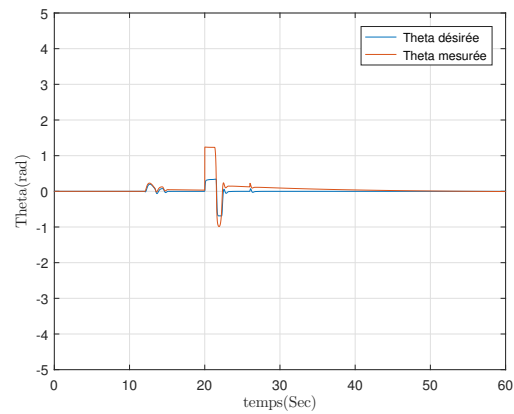


FIGURE 2.51 – Tracés de θ et θ_d avec un vent calme

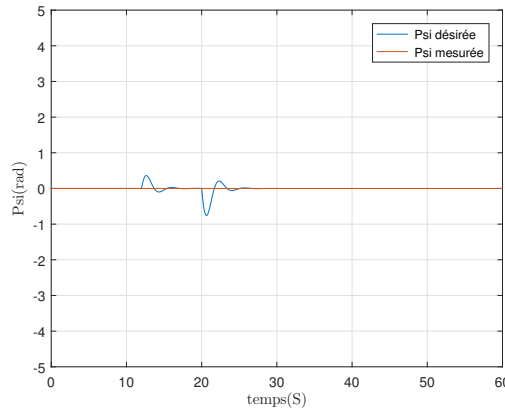


FIGURE 2.52 – Tracés de ψ et ψ_d avec un vent calme

Pas une grande influence sur les trajectoires de rotation qui arrivent à suivre les consignes désirés donc nous pouvons conclure que notre système reste assez stable dans le cas du vent calme.

- Supposant que nous sommes dans le cas de petite brise :

L'air s'écoule en général irrégulièrement entraînant une forte variabilité du vent en direction et en force. Il existe des modalités pour évaluer la vitesse du vent sans la mesurer vraiment. On utilise pour cela des échelles. Une des échelles les plus souvent utilisées est celle de *Beaufort*, qui permet d'estimer la vitesse du vent selon ses effets sur l'environnement.

Degré de l'échelle	Appellation	Effets produits par le vent	Vitesse (km/h)	Vitesse (noeuds)
0	Calme	Calme, la fumée s'élève verticalement.	0 à 1	0 à 0,54
1	Brise très légère	La direction du vent est révélée par le sens de la fumée, mais non par la girouette.	1 à 5	0,54 à 2,7
2	Brise légère	On sent le vent sur la figure. La girouette est mise en mouvement. Les feuilles bougent.	5 à 11	2,7 à 5,9
3	Petite brise	Feuilles et petites branches constamment agitées. Le vent déploie les drapeaux légers.	11 à 19	5,9 à 10,2
4	Jolie brise	Soulève la poussière et les papiers, fait mouvoir les petites branches.	19 à 28	10,2 à 15
5	Bonne brise	Les arbustes en feuilles balancent. Des vaguelettes se forment sur les lacs ou étangs.	28 à 38	15 à 20,5

FIGURE 2.53 – Echelle de *Beaufort*

Pour notre étude nous avons choisi une petite brise parce que les drones ne peuvent pas faire des

vols à plus de 24 Km/h.

Nous avons ajouté un profil de vent à la simulation sous forme d'une fonction **Random**

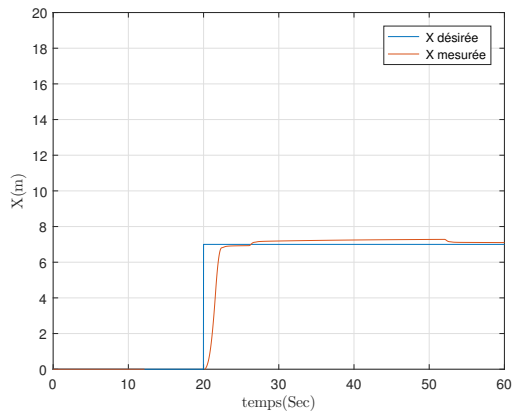


FIGURE 2.54 – Tracés de x et x_d avec un vent du Petite brise

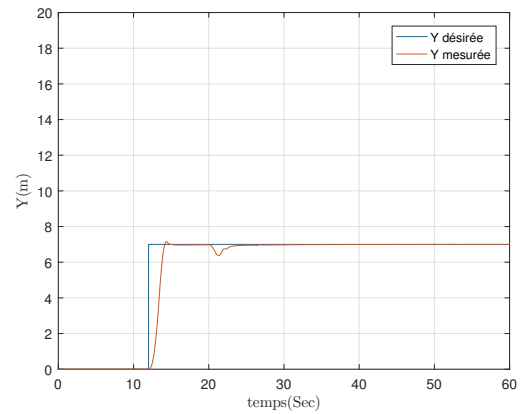


FIGURE 2.55 – Tracés de y et y_d avec un vent du Petite brise

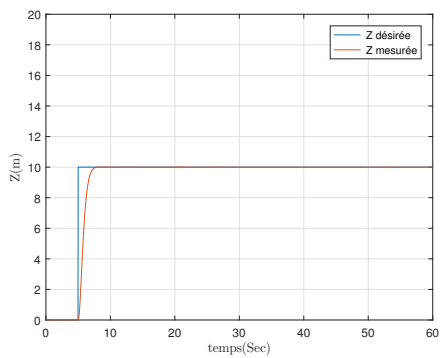


FIGURE 2.56 – Tracés de z et z_d avec un vent du Petite brise

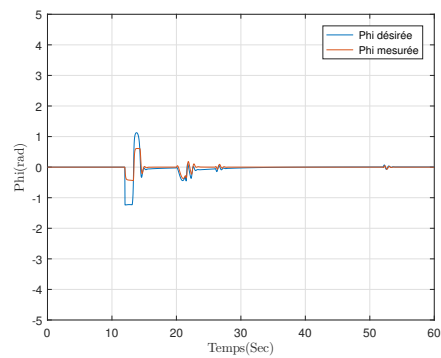


FIGURE 2.57 – Tracés de ϕ et ϕ_d avec un vent du Petite brise

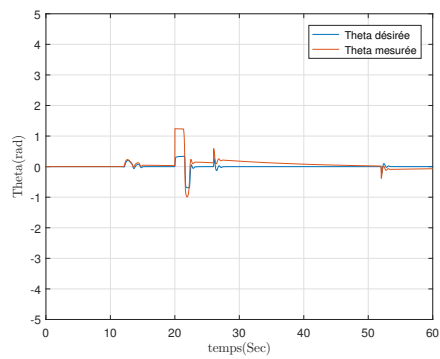


FIGURE 2.58 – Tracés de θ et θ_d avec un vent du Petite brise

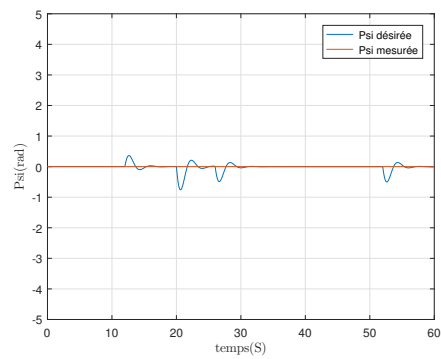


FIGURE 2.59 – Tracés de ψ et ψ_d avec un vent du Petite brise

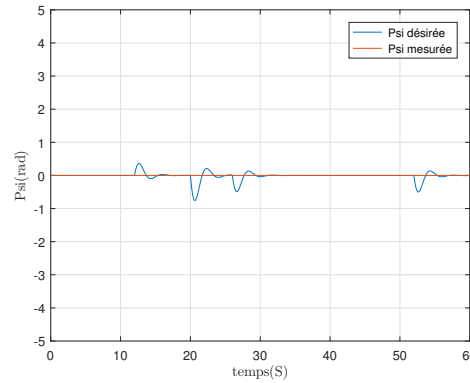


FIGURE 2.60 – Tracés de ψ et ψ_d avec un vent du Petite brise

Clairement, il apparaît qu'un vent plus important qu'un vent calme perturbe le fonctionnement du drone.

Pour une petite brise, le drone continue de voler de façon acceptable mais au delà de cette vitesse, il faut éviter de le faire voler sinon nous risquons un crash.

2.17 Conclusion

Dans ce chapitre, nous avons présenté la modélisation d'un drone quadrotor en tenant compte de toutes les forces et moments agissant sur le drone. Pour pouvoir appliquer une loi de commande linéaire, ce modèle a été simplifié.

En se basant sur les paramètres du drone identifié lors d'un travail précédent dans notre équipe, nous avons synthétisé une commande de type PID permettant de stabiliser le quadrotor et de faire du suivi de consigne.

Nous avons ensuite testé la robustesse de cette lois de commande par rapport à deux formes de vent. Dans le chapitre suivant, nous nous intéressons au calcul de l'indice de végétation NDVI des plantations.

Chapitre 3

Réalisation de la carte NDVI

3.1 Introduction

Dans ce chapitre, nous définissons le NDVI, et nous parlons de sa relation avec les drones. Ensuite, nous passons à la partie matérielle dans laquelle nous présentons les composants utilisés dans le montage. Après cela, nous citons les étapes que nous avons suivies pour installer le système d'exploitation et l'OpenCV / Python sur la carte Raspberry. Nous finissons avec des tests d'image capturées avec la caméra.

3.2 NDVI (Normalized Difference Vegetation Index)

L'indice de végétation par différence normalisée (NDVI) est l'une des techniques les plus couramment utilisées par les chercheurs pour mesurer la "verdure" des plantes ou leur activité photosynthétique. Le concept de NDVI est basé sur l'observation que la quantité de lumière réfléchie par une végétation verte saine varie en fonction de la végétation stressée ou morte et des surfaces rocheuses. La végétation considérée comme saine, ou photosynthétique, absorbe principalement la lumière rouge qui se situe dans le spectre visible tout en réfléchissant une grande partie de la lumière proche infrarouge. D'autre part, la végétation stressée ou morte a tendance à réfléchir plus de lumière rouge visible et moins de lumière proche infrarouge. Pour les surfaces non végétalisées, la réflectance est beaucoup plus uniforme dans le spectre lumineux [8]. La variation de la quantité de lumière absorbée et réfléchie de la végétation se produit parce que le pigment chlorophyllien des feuilles de plantes utilise la lumière visible, absorbée de 0,4 à 0,7 μm dans le processus de photosynthèse. La lumière proche infrarouge de 0,7 à 1,1 μm est réfléchie par coïncidence par la structure cellulaire des feuilles [2]. Par conséquent, une plante plus saine absorbe plus de lumière visible pour une activité photosynthétique accrue et a également une structure cellulaire plus dense pour réfléchir des niveaux plus élevés de lumière proche infrarouge. Une illustration de ce concept est présentée dans la Figure (3.1).

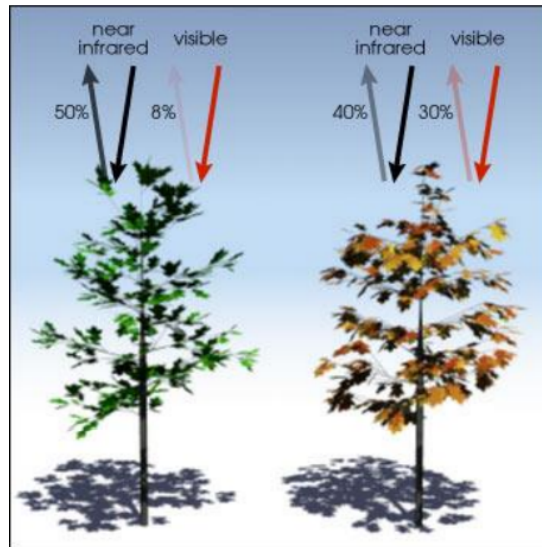


FIGURE 3.1 – Comparaison entre la lumière visible réfléchie et la lumière proche infrarouge pour la végétation verte et la végétation sénescente

La formule mathématique pour calculer le NDVI est donnée par l'équation (3.1) [18] :

$$NDVI = \frac{NIR - R}{NIR + R} \quad (3.1)$$

C'est une valeur scalaire comprise entre $[-1, 1]$. Plus cette valeur est élevée, plus la probabilité qu'elle corresponde à une végétation verte est grande [22].

3.2.1 Applications du NDVI

La déforestation s'est avérée produire intrinsèquement des effets négatifs sur les systèmes écologiques à l'échelle régionale et continentale malgré les gains du bois et d'autres industries. «Entre 2000 et 2012, la perte de forêt mondiale a augmenté d'environ $2\,000\text{ km}^2$ par an». Il existe plusieurs techniques qui utilisent différents indices de végétation (VI) pour la cartographie de la déforestation. Cependant, le NDVI est l'indices de végétation le plus fréquemment utilisé pour la cartographie de la déforestation dans les tropiques. En comparant les observations actuelles aux données capturées précédemment, les perturbations de la densité forestière peuvent être cartographiées en temps quasi réel. Le NDVI est également utilisé dans l'évaluation du couvert végétal, des stocks de carbone et de l'état des terres, ce qui peut fournir des indicateurs de résilience.

Bien qu'il ait été démontré que le NDVI est très utile pour étudier la végétation d'un point de vue écologique, il est également bénéfique pour de nombreuses applications agricoles, résidentielles et biologiques. Par exemple, le NDVI peut être utilisé pour déterminer la teneur en eau de la végétation, afin de déduire le stress hydrique pour les décisions d'irrigation et aider à estimer le rendement de cultures telles que le maïs et le soja [20]. En outre, on s'intéresse de plus en plus à la "verdure" des

quartiers, ou à l'exposition à la végétation et aux espaces verts, et à son association avec des effets positifs sur la santé physique et mentale et la réduction des risques de problèmes cardiovasculaires. "Le NDVI semble être un outil de mesure valable et pratique pour la recherche épidémiologique sur les associations entre la verdure et la santé au niveau du quartier" [10]. D'un point de vue biologique, le NDVI a récemment permis de mieux comprendre le lien entre la répartition des espèces animales et les ressources disponibles. Ces exemples ne fournissent qu'un sous-ensemble des applications du NDVI mais démontrent clairement sa polyvalence et son utilité pour résoudre des problèmes du monde réel.

3.2.2 UAV pour NDVI

Le type de plate-forme utilisée pour la télédétection dépend des exigences de résolution et des limites de coût. Les plates-formes de télédétection peuvent aller des hélicoptères et dirigeables à des altitudes relativement basses, aux aéronefs et aux ballons volant à des altitudes moyennes et élevées, en passant par des satellites à plusieurs centaines de kilomètres de distance. Les véhicules aériens sans pilote (UAV) constituent une alternative pratique à faible coût aux plates-formes conventionnelles pour la télédétection de données à haute résolution avec une flexibilité accrue des opérations [19]. Les drones font référence à une variété d'aéronefs à voilure fixe semi-autonomes ou entièrement autonomes, de quadricoptères et d'autres hélicoptères multi-rotors, de formes et de tailles variées avec des conceptions avancées pour transporter de petites charges utiles et des systèmes de contrôle de vol intégrés. «Les quadricoptères combrent un vide entre les satellites et les aéronefs lorsqu'une plate-forme de surveillance stationnaire est nécessaire pour l'observation à relativement long terme d'une zone [13].

Les drones sont également utilisés pour cartographier et surveiller l'étendue, la biomasse et la santé des zones humides de marée, des forêts, des cultures agricoles et d'autres couvertures végétales. À l'aide de types spécifiques d'imageurs, ils peuvent cartographier et classer les types de végétation côtière en utilisant simplement une caméra, un GPS et une unité de mesure inertielle (IMU) pour calculer l'orientation de l'UAV dans l'espace 3D. Un exemple de ceci est illustré par un UAV à voilure fixe et un hélicoptère travaillant ensemble dans des parcours éloignés pour localiser et pulvériser des herbicides sur les mauvaises herbes. Une autre application pratique des drones est la surveillance des estuaires à la recherche de proliférations d'algues nuisibles [18].

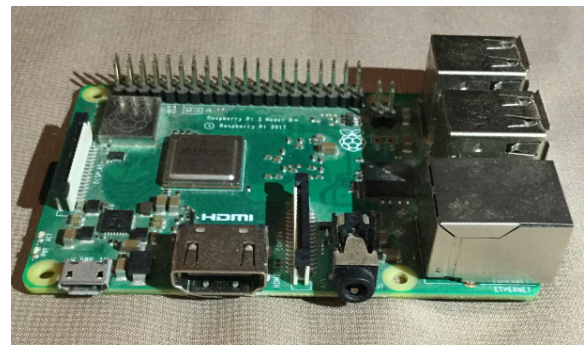
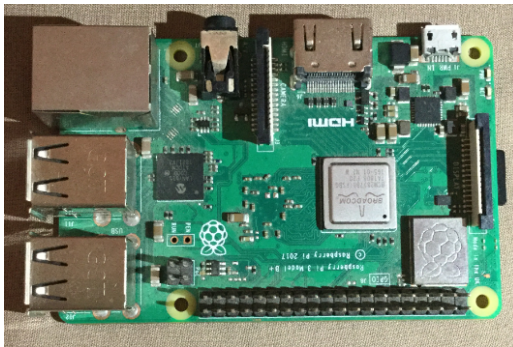


FIGURE 3.2 – Quadricoptère avec caméra intégrée

3.3 Partie hardware

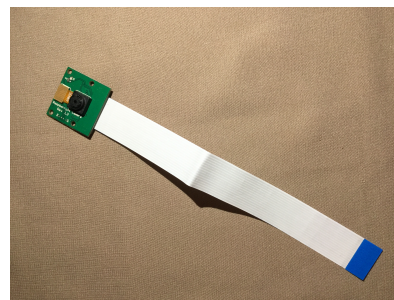
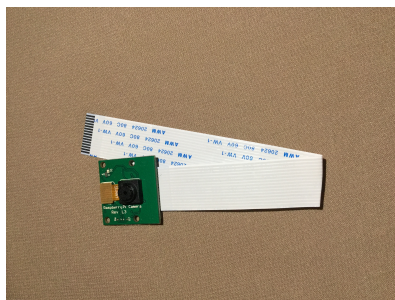
3.3.1 Raspberry Pi 3 modele B

La Raspberry pi est bien nommée comme micro-ordinateur, elle est 10 fois plus puissante que la première Pi. La Raspberry Pi 3 est équipée d'un processeur Broadcom BCM2837 ARM Cortex A53 SoC 64 bits avec 4 cœurs à 1,2 GHz, avec une augmentation des performances de 50% par rapport au Pi2. Ce nouveau processeur permettra d'utiliser la Raspberry Pi 3 pour le bureau ou la navigation sur le web. Cette troisième version est équipée d'une puce WiFi et de Bluetooth à basse consommation. Et de plus de ports USB pour la connexion de différents périphériques. Grâce à ses caractéristiques, la carte Raspberry Pi 3 fonctionne avec les systèmes d'exploitation Linux Android ou Firefox OS installé sur une carte micro SD. elle peut être utilisée comme plate-forme de développement, comme serveur web ou comme centre de médias. Sans aucune limite, la Raspberry Pi 3 B est le meilleur support pour la réalisation de notre projet.



3.3.2 Raspberry Pi caméra rev L3

La carte caméra spécifique au Raspberry Pi a été créée de manière pour se connecter à l'un des ports d'entrée vidéo du Raspberry Pi sur la carte elle-même. Cela signifie qu'il n'utilisera aucun des ports USB limités ou d'autres entrées. L'appareil photo n'est pas l'offre de résolution la plus élevée au monde, mais c'est un 5MP respectable qui peut prendre des images à 2592 x 1944 pixels et enregistrer des vidéos à 1080p à 30 fps, ce qui est plus que suffisant pour notre projet. elle est également petite, ce qui la rend aussi portable que la Raspberry Pi elle-même.



3.3.3 Alimentation électrique USB

Une alimentation électrique de qualité à (2,5A) ou (12,5W) et avec un connecteur micro USB. Le bloc d'alimentation officiel Raspberry Pi est le choix recommandé, car elle peut faire face aux demandes de puissance à commutation rapide du Raspberry Pi.



3.3.4 Adaptateur et carte microSD

la carte microSD sert de stockage permanent au Raspberry Pi, tous les fichiers que nous créons et les logiciels que nous installons, ainsi que le système d'exploitation lui-même, sont stockés sur la carte microSD. Une carte de 8 GB peut être utilisée pour démarrer, mais une carte de 16 GB offre plus de place pour se développer. Dans notre projet, nous avons utilisé un type de carte sd de classe 10 de 64 GB juste pour obtenir plus de performances. L'adaptateur est utilisé pour connecter la carte sd à notre ordinateur afin de la formater et d'y installer le système d'exploitation (plus de détails sur l'installation sont expliqués dans la section(3.5)).



3.3.5 Clavier et souris USB

Le clavier et la souris nous permettent de contrôler la Raspberry pi. Presque tous les claviers et souris filaires ou sans fil dotés d'un connecteur USB fonctionneront avec la Raspberry Pi, bien que certains claviers de style "jeu" dotés de lumières colorées puissent consommer trop d'énergie pour être utilisés de manière fiable.



3.3.6 Câble VGA / VGA et adaptateur VGA / HDMI

Pour connecter notre moniteur à notre raspberry pi, nous avons besoin de cet adaptateur car, comme il est mentionné précédemment, la Raspberry pi n'a qu'un port HDMI.



3.3.7 Monitor

Il nous permet de visualiser l'évolution de notre projet et de voir les résultats de l'exécution.



3.3.8 Montage finale et résultat de l'alimentation

Après avoir inséré la carte microSD en bas du Raspberry Pi, nous devons placer les autres éléments chacun dans son port correspondant comme indiqué sur la figure(3.3)



FIGURE 3.3 – Montage des éléments avec la Raspberry

après avoir terminé notre montage comme il est mentionné, il ne nous reste plus qu'à placer l'alimentation. Après le premier démarrage, la Raspberry Pi démarre en mode configuration.

En mode configuration, nous voulons :

- Modifier les paramètres régionaux pour qu'ils correspondent à notre emplacement, puis modifier le fuseau horaire en fonction de notre emplacement.
- Connecter à notre réseau WiFi en sélectionnant son nom et en entrant le mot de passe.
- Rechercher des mises à jour du système d'exploitation et les installer.

Après avoir fait toutes ces étapes, nous redémarrons la Raspberry Pi, et quand elle redémarrera, elle présentera notre nouvelle configuration, le résultat est montré sur la figure(3.4).



FIGURE 3.4 – Résultat de l'alimentation

Notre Raspberry est maintenant configurée et prête à être utilisée.

3.4 Partie software

3.4.1 Introduction sur les images

De quoi les images sont construites ?

Prenons l'exemple de l'affichage du chiffre 3 :

Si nous voulons afficher le numéro 3, nous prendrons un tableau de cases, où chaque case pourrait être remplie ou vide Figure(3.5)

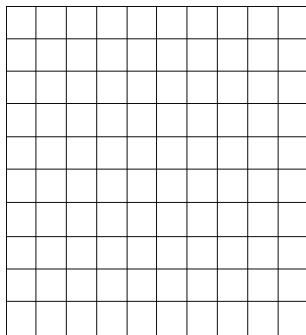


FIGURE 3.5

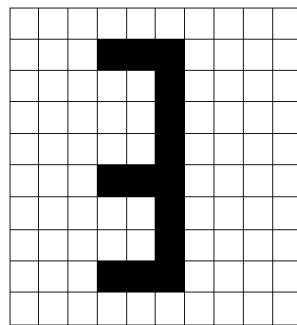


FIGURE 3.6

1	1	1	1	1	1	1	1	1	1
1	1	1	0	0	0	1	1	1	1
1	1	1	1	1	0	1	1	1	1
1	1	1	1	1	0	1	1	1	1
1	1	1	1	1	0	1	1	1	1
1	1	1	0	0	0	1	1	1	1
1	1	1	1	1	0	1	1	1	1
1	1	1	1	1	0	1	1	1	1
1	1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1

FIGURE 3.7

Donc pour écrire le numéro 3, nous colorons quelques cases pour créer cette forme, certaines cases sont noires et les autres sont blanches Figure(3.6). Nous désignons toutes les cases noires par "0" et les cases blanches par "1" Figure(3.7). Dans cet exemple, nous avons 10 cases sur 10, si nous voulons plus de détails, nous pouvons augmenter le nombre de cases.

En réalité les cases sont des pixels donc les images sont constituées d'un grand nombre de pixels. Plus le nombre de pixels d'une image est grand, plus il y a des chances que nous puissions récupérer plus de détails et d'informations.

Nombre fixe de pixels

VGA : 640 × 480

HD : 1280 × 720

FHD : 1920 × 1080

4K : 3840 × 2160

cela signifie que VGA a 640 pixels dans la largeur et 480 pixels dans la hauteur d'une image.

Image binaire :

elle n'a que deux couleurs (2 niveaux) qui sont le noir et le blanc → "0" et "1" .

Afin d'avoir plus de détails, nous pouvons avoir une image avec plus de niveaux, cela signifierait qu'au lieu d'avoir seulement "0" et "1", nous aurons une plage de valeurs.



FIGURE 3.8 –
Image originale



FIGURE 3.9 –
Image binaire

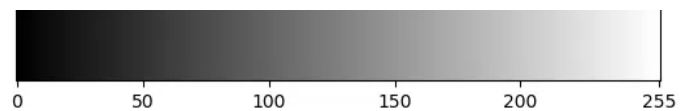


FIGURE 3.10 –
GrayScale image

Comme nous pouvons le voir, l'image binaire (3.9) n'est pas très claire, donc nous ne pouvons pas obtenir des informations intéressantes.

Alors, nous allons utiliser des valeurs de 8-bits :

$$2^8 = 256$$



Cela nous donne une résolution de 256 où "0" est le noir et 255 est le blanc. Donc maintenant nous avons 254 nuances de gris, l'image que nous avons trouvée avec cette résolution(3.10) est connue sous le nom GrayScale image.

Image en couleurs :

Pour l'image en couleurs, nous avons 3 images "GrayScale" représentant les intensités de vert, rouge et bleu.

$$\begin{array}{c} \mathbf{R} \end{array} + \begin{array}{c} \mathbf{G} \end{array} + \begin{array}{c} \mathbf{B} \end{array} = \text{une image en couleur} \\ \text{(RGB)}$$

Ajouter ces images ensemble nous donne une image colorée.
→ si nous l'appliquons à un VGA coloré, nous obtenons :

$$RGB(VGA) = 640 \times 480 \times 3$$

3.5 Raspberry Pi OS (Raspbian)

Raspbian est le système d'exploitation recommandé pour une utilisation normale sur une Raspberry Pi. C'est un système gratuit basé sur Debian, optimisé pour le matériel Raspberry Pi. Il est livré avec plus de 35000 packages : des logiciels précompilés regroupés dans un format agréable pour une installation facile sur la Raspberry Pi.

3.5.1 Etapes d'installation

Pour que notre raspberry pi soit opérationnelle, la première étape consiste à formater notre carte sd à l'aide du logiciel "SD Card Formatter".

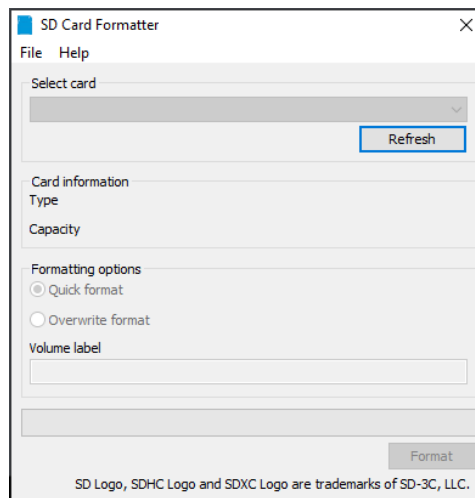


FIGURE 3.11 – SD Card Formatter

Une fois notre carte sd formatée, l'étape suivante consiste à télécharger le Raspbian à partir du site officiel "raspberry pi.org", et le mettre dans notre carte SD en utilisant "BalenaEtcher".

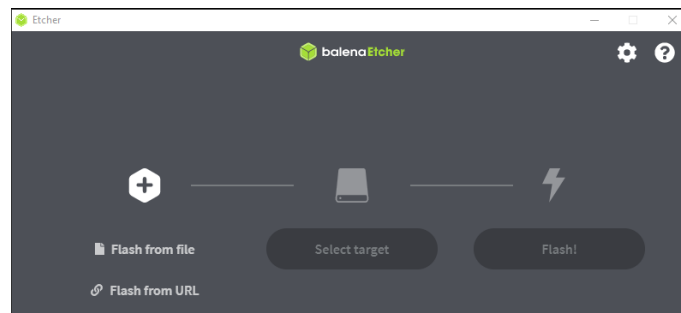


FIGURE 3.12 – BalenaEtcher

3.6 OpanCV-Python

3.6.1 OpenCV

En 2005, OpenCV a été utilisée sur Stanley, le véhicule qui a remporté le Grand Challenge 2005 de la DARPA. Plus tard, son développement actif s'est poursuivi sous le soutien de Willow Garage, avec Gary Bradsky et Vadim Pisarevsky à la tête du projet. Actuellement, OpenCV supporte de nombreux algorithmes liés à la vision par ordinateur et à l'apprentissage automatique et se développe de jour en jour.

Actuellement, OpenCV prend en charge une grande variété de langages de programmation comme C ++, Python, Java, etc, et il est disponible sur différentes plates-formes, notamment Windows, Linux, Android, etc. OpenCV-Python est l'API Python d'OpenCV. Il combine les meilleures qualités de l'API OpenCV C ++ et du langage Python [16].

3.6.2 OpenCV-Python

Python est un langage de programmation à usage général lancé par Guido van Rossum, qui est devenu très populaire en peu de temps principalement en raison de sa simplicité et de sa lisibilité du code. Il permet au programmeur d'exprimer ses idées en moins de lignes de code sans réduire la lisibilité.

Comparé à d'autres langages comme C / C ++, Python est plus lent. Mais une autre caractéristique importante de Python est qu'il peut être facilement étendu avec C / C ++. Cette fonctionnalité nous aide à écrire des codes intensifs en calcul en C / C ++ et à créer un wrapper Python pour cela afin que nous puissions utiliser ces wrappers comme modules Python [16].

OpenCV-Python est une bibliothèque de liaisons Python conçue pour résoudre les problèmes de vision par ordinateur.

3.7 Etapes d'installation d'OpenCV sur Raspberry Pi

Cette étape n'est pas facile comme il semble, nous avons été confrontés à de nombreux problèmes d'installation et de connexion qui nous ont obligés à redémarrer depuis le début à chaque fois que nous échouions. Il a donc fallu beaucoup de temps pour obtenir le résultat souhaité.

Nous allons parcourir 6 étapes pour compiler et installer OpenCV 4 sur la Raspberry Pi.

Étape 1 : Étendre le système de fichiers sur la Raspberry Pi

la première chose que nous avons faite a été d'étendre notre système de fichiers pour inclure tout l'espace disponible sur notre carte micro-SD. En tapant la commande "sudo raspi-config" sur le terminal, et puis sélectionner l'élément de menu "Advanced Options", suivi en sélectionnant "Expand filesystem". Pour finir cette étape il faut redémarrer la Raspberry en tapant la commande "sudo reboot".

Étape 2 : Installer les dépendances d'OpenCV 4 sur la Raspberry Pi

Après le redémarrage il est très important de mettre à jour le système avant toute installation. Nous faisons cela en tapant la commande "sudo apt-get update && sudo apt-get upgrade".

Puis nous installons des outils de développement y compris CMake avec la commande " sudo apt-get install build-essential cmake unzip pkg-config".

Après, nous installons une sélection de bibliothèques d'images et de vidéos en incluant les commandes suivantes l'une après l'autre :

- "sudo apt-get install libjpeg-dev libpng-dev libtiff-dev"
- "sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev"
- "sudo apt-get install libxvidcore-dev libx264-dev"

Ces commandes sont essentielles pour pouvoir travailler avec des fichiers d'images et de vidéos.

A partir de là, installons GTK, notre backend d'interface graphique : " sudo apt-get install libgtk-3-dev".

Ensuite, installons un paquet qui pourrait réduire les fâcheux avertissements de la GTK : " sudo apt-get install libcansel-gtk*".

Suivi de l'installation de deux packages contenant des optimisations numériques pour OpenCV," sudo apt-get install libatlas-base-dev gfortran".

Et enfin, installons les en-têtes de développement Python 3 : " sudo apt-get install python3-dev".

Étape 3 : Télécharger OpenCV 4 pour la Raspberry Pi

Une fois que nous avons tous ces prérequis installés, nous pouvons passer au téléchargement d'OpenCV.

Dans cette étape, nous allons télécharger opencv et opencv_contrib de type .zip et les placer dans le dossier "home/pi" :

- "cd~"
- "wget -O opencv.zip https://github.com/opencv/opencv/archive/4.0.0.zip"
- " wget -O opencv_contrib.zip https://github.com/opencv/opencv_contrib/archive/4.0.0.zip"

Et pour dézipper les archives, il suffit de taper les commandes :

- " unzip opencv.zip"
- " unzip opencv_contrib.zip"

Étape 4 : Configurer l'environnement virtuel Python 3 pour OpenCV4

Maintenant que opencv et opencv_contrib sont téléchargés et prêts à l'emploi, nous configurons notre environnement.

Nous allons installer pip, un gestionnaire de packages Python. Pour l'installer, saisissez simplement ce qui suit dans le terminal :

- " wget https://bootstrap.pypa.io/get-pip.py"

— " sudo python3 get-pip.py"

Puis, nous installons le virtualenv et le virtualenvwrapper :

— " sudo pip install virtualenv virtualenvwrapper"

— " sudo rm -rf /get-pip.py /.cache/pip"

Les environnements virtuels nous permettront d'exécuter différentes versions de logiciels Python de manière isolée sur notre système.

Pour terminer l'installation de ces outils, nous devons mettre à jour notre "~/.profile". En utilisant l'éditeur de texte du terminal "nano" et tapez : " nano ~/.profile "

puis nous ajouterons ces lignes et les enregistrerons :

— # virtualenv and virtualenvwrapper

— export WORKON_HOME=\$HOME/.virtualenvs

— export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3

— source /usr/local/bin/virtualenvwrapper.sh

Ensuite, créons le fichier ~/.profile : " source ~/.profile".

Nous sommes maintenant au point où nous pouvons créer l'environnement virtuel OpenCV 4 + Python 3 sur la Raspberry Pi en tapant : "mkvirtualenv cv -p python3".

Le premier paquet Python et le seul pré-requis OpenCV que nous installerons est NumPy.

Numpy est une bibliothèque hautement optimisée pour les opérations numériques. Elle donne une syntaxe de type MATLAB. Toutes les structures de tableaux OpenCV sont converties en tableaux Numpy. Ainsi, quelles que soient les opérations que vous pouvez effectuer dans Numpy, vous pouvez les combiner avec OpenCV, ce qui augmente le nombre d'armes dans votre arsenal. Pour l'installer, il suffit de taper la commande suivante : " pip install numpy" .

Étape 5 : CMake et compiler OpenCV 4 pour la Raspberry Pi

Pour cette étape, nous allons configurer notre compilation avec CMake, puis exécuter *make* pour compiler OpenCV.

nous allons d'abord créer un répertoire "build", en tapant les lignes suivantes :

— cd ~/opencv

— mkdir build

— cd build

puis nous lancerons CMake pour configurer le "build" OpenCV 4 :

cmake -D CMAKE_BUILD_TYPE=RELEASE \

— -D CMAKE_INSTALL_PREFIX=/usr/local\

— -D OPENCV_EXTRA_MODULES_PATH=~/.opencv_contrib/modules \

— -D ENABLE_NEON=ON \

— -D ENABLE_VFPV3=ON \

- `-D BUILD_TESTS=OFF \`
- `-D OPENCV_ENABLE_NONFREE=ON \`
- `-D INSTALL_PYTHON_EXAMPLES=OFF \`
- `-D BUILD_EXAMPLES=OFF ..`

Augmenter le SWAP sur la Raspberry Pi

Avant de commencer la compilation, nous devons augmenter l'espace d'échange. Cela nous permettra de compiler OpenCV avec les quatre cœurs du Raspberry Pi sans que la compilation ne se bloque en raison de l'épuisement de la mémoire. Pour faire ça, nous devrions taper :” `sudo nano /etc/dphys-swapfile`”, puis modifier **CONF_SWAPSIZE** :

au début c'était `CONF_SWAPSIZE=100` et nous devons le changer en `CONF_SWAPSIZE=2048`.

À partir de là, redémarrer le service d'échange :

- `”sudo /etc/init.d/dphys-swapfile stop”`
- `”sudo /etc/init.d/dphys-swapfile start”`

Compiler OpenCV 4

Nous sommes maintenant prêts à compiler OpenCV 4 : `” make -j4”`.

lorsque nous réussissons à compiler OpenCV 4 sans aucune erreur, à partir de là, nous pouvons installer OpenCV 4 avec deux commandes supplémentaires :

- `”sudo make install”`
- `”sudo ldconfig”`

Étape 6 : Lier OpenCV 4 à l'environnement virtuel Python 3

Créons un lien symbolique depuis l'installation d'OpenCV dans le répertoire système site-packages vers notre environnement virtuel :

- `”cd ~/.virtualenvs/cv/lib/python3.5/site-packages/”`
- `”ln -s /usr/local/python/cv2/python-3.5/cv2.cpython-35m-arm-linux-gnueabi.so cv2.so”`
- `cd ~`

3.8 Résultat et interprétation

Après la réussite d'installation d'OpenCV et de python, nous avons lancé un programme qui calcule l'Indice de Végétation par Différence Normalisée (NDVI).

Premier essai :

1. Après avoir lancé le programme, il affiche une fenêtre divisée en quatre cases, la première montre l'image bleue, la seconde l'image NIR, la troisième l'image verte, et la dernière l'image NDVI,

comme indiqué sur la figure(3.14). Dans ce test, nous n'avons pas pu voir le traitement qu'en temps réel, et nous n'avons pas pu enregistrer les images.

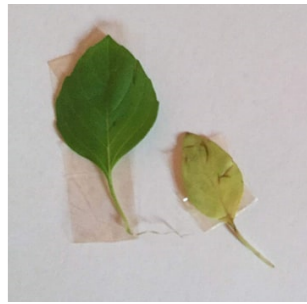


FIGURE 3.13 – Image originale

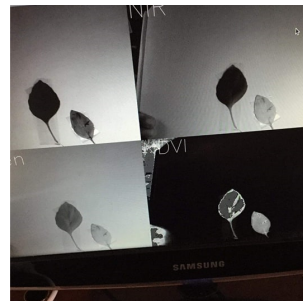


FIGURE 3.14 – Résultat d'affichage

- Après le résultat que nous avons obtenu dans le premier essai, nous avons changé le programme de manière à obtenir uniquement la fenêtre qui affiche le traitement NDVI, car c'est la partie qui nous intéresse. Nous ajoutons également une commande qui nous permet de sauvegarder les images juste en double-cliquant sur la souris.

Les résultats obtenus sont indiqués sur la figure(3.16).



FIGURE 3.15 – Image originale



FIGURE 3.16 – Résultat du traitement NDVI

Comme nous pouvons le voir, les surfaces non végétalisées sur l'image d'origine sont présentées en couleur blanche sur l'image NDVI, et les surfaces considérées comme saines sont présentées avec la couleur sombre. Ce résultat montre ce qu'on appelle un faux NDVI, car la caméra est toujours avec le filtre infrarouge.

Deuxième essai

dans ce test et pour obtenir le bon résultat :

1. Nous nous débarrassons du filtre IR de la caméra Pi comme illustré sur la Figure(3.17)

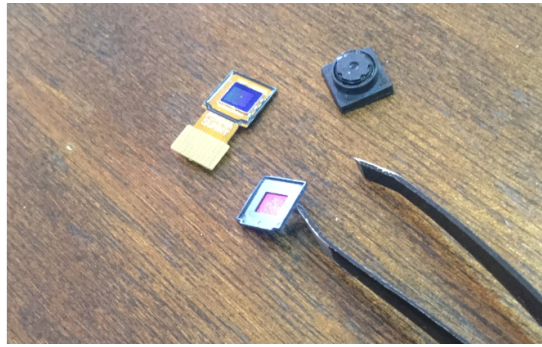


FIGURE 3.17 – Filtre IR

2. Ensuite, nous l'avons remplacé par une feuille de papier bleu transparent pour jouer le rôle d'un filtre bleu qui est placé derrière l'objectif de la caméra comme indiqué sur les figures suivantes :



FIGURE 3.18 – Papier bleu transparent

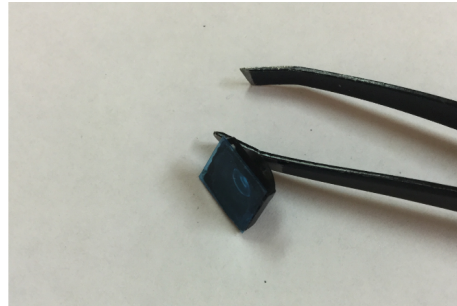


FIGURE 3.19 – Résultat final

Résultat final après avoir retiré le filtre IR :

Après avoir retiré le filtre IR, nous avons obtenu les résultats souhaités, mais malheureusement nous n'avons pas enregistré les images à cause d'un problème matériel.

3.9 Conclusion

Ce chapitre était la partie la plus importante du projet. Dans lequel nous avons défini le NDVI et comment le calculer. Puis nous avons parlé des éléments utilisés dans le montage, et de toutes les étapes que nous avons suivies pour installer OpenCV afin que nous puissions analyser nos plantes. Malgré tous les problèmes et les difficultés que nous avons rencontrés, nous avons réussi à prendre des photos et à générer la carte en fonction du NDVI.

Conclusion générale

La popularité croissante des quadrirotors, l'ambition pour la conception des drones à décollage verticale et les avancées technologiques dans ce domaine, ont fait que le quadrirotor voit un intérêt écrasant ces dernières années. En fait, la nécessité d'un système de contrôle d'attitude pour la stabilisation du quadrirotor et pouvoir prendre des images pour générer l'indice de végétation s'est avérée être les principales motivations de ce travail.

Le projet décrit tout d'abord, les drones en général spécialement les quadrirotors, leur évolution et leurs avantages et inconvénients. Par la suite nous nous sommes intéressés dans ce mémoire aux problèmes liés à la synthèse de la loi de commande PID d'un robot volant de type quadrirotor dont le but est de concevoir une approche de commande permettant le suivi d'une trajectoire désirée par un quadrirotor. Pour cela, nous avons présenté la modélisation du quadrirotor qui est une étape indispensable à la bonne compréhension des lois de la physique qui régissent notre système. L'exploitation du modèle mathématique qui définit la dynamique du quad-copter, nous permet de déduire le modèle d'état. Pour lequel nous avons synthétisé une commande type PID. Des tests de simulation ont montré l'efficacité de cette commande dans la stabilisation et le suivi de référence pour le drone. Par la suite, nous avons effectué des tests de robustesse par rapport à des changements paramétriques et des perturbations extérieures.

Pour la partie pratique de ce projet, nous avons expliqué en détail toutes les étapes nécessaires pour installer et configurer une raspberry pi, puis nous avons parlé de l'installation d'OpenCV et de python pour réaliser notre carte NDVI.

À la fin, et après toutes les difficultés auxquelles nous avons été confrontées, nous avons réussi à réaliser une carte en fonction de NDVI, ce qui pourrait être un bon point de départ pour les sciences de l'agriculture.

Nous espérons que les travaux entamés dans ce projet pourront se poursuivre en intégrant la raspberry et une caméra à un drone et en effectuant des vols sur des champs agricoles pour générer des cartes en fonction de l'indice de végétation.

Annexe : Description du code python

Nous allons expliquer le programme que nous avons utilisé pour calculer le NDVI.

La première partie du code contient les packages, puis les déclarations des fonctions dont nous avons besoin dans ce programme, qui sont :

1. **"disp_multiple"** pour combiner quatre images dans une seule fenêtre.
2. **"label"** pour donner un nom à chaque image.
3. **"contrast_stretch"** pour Effectue un simple étirement de contraste de l'image donnée.
4. **"captureFrame"** elle nous donne la possibilité de sauvegarder des images en double-cliquant sur la souris.

```
import time
import numpy as np

import cv2
import picamera
import picamera.array

def disp_multiple(im1=None, im2=None, im3=None, im4=None):
    """
    Combines four images for display.
    """
    height, width = im1.shape

    combined = np.zeros((2 * height, 2 * width, 3), dtype=np.uint8)

    combined[0:height, 0:width, :] = cv2.cvtColor(im1, cv2.COLOR_GRAY2RGB)
    combined[height:, 0:width, :] = cv2.cvtColor(im2, cv2.COLOR_GRAY2RGB)
    combined[0:height, width:, :] = cv2.cvtColor(im3, cv2.COLOR_GRAY2RGB)
    combined[height:, width:, :] = cv2.cvtColor(im4, cv2.COLOR_GRAY2RGB)

    return combined

def label(im1, text):
    """
    Labels the given image with the given text
    """
    return cv2.putText(im1, text, (0, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, 255)
```

```

def contrast_stretch(im1):
    """
    Performs a simple contrast stretch of the given image, from 5-95%.
    """
    in_min = np.percentile(im, 5)
    in_max = np.percentile(im, 95)

    out_min = 0.0
    out_max = 255.0

    out = im - in_min
    out *= ((out_min - out_max) / (in_min - in_max))
    out += in_min

    return out
def captureFrame(event,x,y,flags,frame):
    if event == cv2.EVENT_LBUTTONDOWN:
        captureFrame.i +=1
        cv2.imwrite('test'+str(captureFrame.i)+'.png',frame) #want to save frame here

```

La deuxième partie du code, contient le programme principal qui est formé de quatre parties :

1. Résolution de la caméra.

```

def run():
    with picamera.PiCamera() as camera:
        # Set the camera resolution
        x = 400
        camera.resolution = (int(1.33 * x), x)

        # Need to sleep to give the camera time to get set up properly
        time.sleep(2)

```

2. Récupérer les données de la caméra au format couleur, puis les séparer avec la commande "Split" pour obtenir les couleurs individuelles r, g, b de l'image.

```

    with picamera.array.PiRGBArray(camera) as stream:
        # Loop constantly
        while True:
            # Grab data from the camera, in colour format
            # NOTE: This comes in BGR rather than RGB, which is important
            # for later!
            camera.capture(stream, format='bgr', use_video_port=True)
            image = stream.array

            # Get the individual colour components of the image
            b, g, r = cv2.split(image)

```

3. Calcul du NDVI

```

# Calculate the NDVI

# Bottom of fraction
bottom = (r.astype(float) + b.astype(float))
bottom[bottom == 0] = 0.01 # Make sure we don't divide by zero!

ndvi = (r.astype(float) - b) / bottom
ndvi = contrast_stretch(ndvi)
ndvi = ndvi.astype(np.uint8)

```

4. donner un nom à l'image et l'afficher

```

# Do the labelling
label(b, 'Blue')
label(g, 'Green')
label(r, 'NIR')
label(ndvi, 'NDVI')

# Combine ready for display
combined = disp_multiple(b, g, r, ndvi)

#save a photo by double clicking on the mouse
cv2.setMouseCallback('image', captureFrame, ndvi)

# Display
#cv2.imshow('image', combined)
cv2.imshow('image', ndvi)
stream.truncate(0)

# If we press ESC then break out of the loop
c = cv2.waitKey(7) % 0x100
if c == 27:
    break

# Important cleanup here!
cv2.destroyAllWindows()

if __name__ == '__main__':
    captureFrame.i = 0 # instializing saving
    run()

```

Bibliographie

- [1] S. Bouabdallah. Design and control of quadrotors with application to autonomous flying. Technical report, Epfl, 2007.
- [2] N.E. Cherigui Ch.Sedini. Conception et commande d'un quadrotor uav à base d'arduino.
- [3] L. Rodolfoía G. Carrillo et al. *Quad rotorcraft control : vision-based hovering and navigation*. 2012.
- [4] H. Huang G. Hoffmann et al. Quadrotor helicopter flight dynamics and control : Theory and experiment. In *AIAA guidance, navigation and control conference and exhibit*, page 6461, 2007.
- [5] H. Maki W. Gharieb et al. Dynamic modeling and control of a quadrotor using linear and nonlinear approaches. 2014.
- [6] J. Christopher J H. Timothy et al. Civil uav capability assessment-draft version. *NASA Capability Assessment Report*, 2004.
- [7] R. AbouSleiman H. Yang et al. Implementation of an autonomous surveillance quadrotor system. In *AIAA Infotech@ Aerospace Conference and AIAA Unmanned... Unlimited Conference*, page 2047, 2009.
- [8] S. Hard. A low-cost normalized difference vegetation index (ndvi) payload for cubesats and unmanned aerial vehicles (uavs). 2018.
- [9] M. Shantz I. Kroo et al. The mesicopter : A miniature rotorcraft concept phase ii final report. Technical report, Technical report, Stanford University, 2001.
- [10] A. Vander Stoep I. Rhew et al. Validation of the normalized difference vegetation index as a measure of neighborhood greenness. *Annals of epidemiology*, 21(12) :946–952, 2011.
- [11] A. KADI. *Mecanique rationnelle* book. 2010.
- [12] H. Khebbache. *Tolérance aux défauts via la méthode backstepping des systèmes non linéaires : application système UAV de type quadrirotor*. PhD thesis, 2018.
- [13] V. Klemas. Coastal and environmental remote sensing from unmanned aerial vehicles : An overview. *Journal of coastal research*, 31(5) :1260–1267, 2015.
- [14] Arnaud Koehl. *Modélisation, observation et commande d'un drone miniature à birotor coaxial*. PhD thesis, 2012.
- [15] L. Laib and E. Maamria. Commande d'un quadrirotor.
- [16] A. Mordvintsev and K. Abid. Opencv-python tutorials documentation. *Obtenido de <https://media.readthedocs.org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals.pdf>*, 2014.

-
- [17] O. E.Holland R. De Nardi et al. Coevolutionary modelling of a miniature rotorcraft. In *10th International Conference on Intelligent Autonomous Systems (IAS10)*, pages 364–373, 2008.
- [18] G. Rabatel, N. Gorretta, et al. Getting ndvi spectral bands from a single standard rgb digital camera : a methodological approach. pages 333–342, 2011.
- [19] I.Meslouli R.M. Mesli. Realisation et pilotage d’un drone à quatre rotors.
- [20] D. Chen T. Jackson et al. Vegetation water content mapping using landsat data derived normalized difference water index for corn and soybeans. *Remote Sensing of Environment*, 92(4) :475–482, 2004.
- [21] M. Vanin. Modeling, identification and navigation of autonomous air vehicles. 2013.
- [22] J. Weier and D. Herring. Measuring vegetation (ndvi and evi). *NASA Earth Observatory*, 20, 2000.

Synthèse de régulateur PID pour un quadrotor et génération de carte en fonction de l'indice de végétation

Résumé

Ce mémoire de Master concerne d'une part la modélisation dynamique, l'identification des paramètres, la synthèse de régulateur PID, les tests de simulation et l'analyse de robustesse pour montrer l'efficacité du contrôleur PID pour la stabilisation d'un drone quadrotor. Et d'autre part, la sélection des composants, les étapes d'installation de "Rasbian" et OpenCV sur la Raspberry Pi pour la génération d'une carte basée sur l'indice de végétation NDVI qui représente un point de départ pour la prise de décision dans l'agriculture de précision.

Ce document décrit en détail la procédure pour la réussite d'un tel projet.

Mots clés

Quadrotor, PID, Raspberry Pi, OpenCV, NDVI

Synthesis of PID controller for a quadrotor and card generation according to the vegetation index

Abstract

This Master thesis concerns on one hand dynamic modeling, identification of parameters, synthesis of PID regulator, simulation tests and robustness analysis to show the efficiency of the PID controller for the stabilization of a quadrotor drone. And on the other hand, the selection of the components, the installation steps of "Rasbian" and OpenCV on the Raspberry Pi for the generation of a map based on the NDVI vegetation index which represents a starting point for the taking decision-making in precision agriculture.

This document describes in detail the procedure for the success of such a project.

Keywords

Quadrotor, PID, Raspberry Pi, OpenCV, NDVI