Commande et localisation embarquée d'un drone aérien en utilisant la vision

Thèse présentée pour l'obtention du grade de Docteur de l'UTC

Guillaume SANAHUJA

Jury :

Yasmina BESTAOUI	Rapporteur
Nicolas PETIT	Rapporteur
Philippe BONNIFAIT	Examinateur
Simon LACROIX	Examinateur
Rogelio LOZANO	Examinateur
Nicolas MARCHAND	Examinateur
Pedro CASTILLO	Directeur de thèse
Isabelle FANTONI	Directeur de thèse

Soutenue le 29 janvier 2010

Université de Technologie de Compiègne

REMERCIEMENTS

Cette thèse a été réalisée au sein du laboratoire Heudiasyc (HEUristique et DIAgnostique des SYstèmes Complexes) de l'Université de Technologie de Compiègne, grâce à une allocation de recherche MENRT. Je tiens donc d'abord à remercier l'ensemble du personnel de ce laboratoire (afin de n'oublier personne), chacun ayant à sa façon contribué à ce travail.

Je remercie plus particulièrement mes directeurs de thèse Pedro Castillo, Chargé de Recherches CNRS à l'Université de Technologie de Compiègne, et Isabelle Fantoni, Chargée de Recherches CNRS à l'Université de Technologie de Compiègne. J'ai fortement apprécié leur aide et dévouement, ainsi que la confiance qu'ils m'ont accordée, du début à la fin.

J'adresse mes vifs remerciements à Yamsina Bestaoui, Maître de Conférences à l'Université d'Evry, ainsi qu'à Nicolas Petit, Professeur à l'École des Mines de Paris, pour avoir accepté d'être rapporteurs de mon travail. Leurs remarques et conseils m'ont été précieux afin d'améliorer cette thèse.

Je remercie également Philippe Bonnifait, Professeur à l'Université de Technologie de Compiègne, de m'avoir fait l'honneur d'accepter la présidence de ce jury, ainsi que Rogelio Lozano, Directeur de Recherches CNRS à l'Université de Technologie de Compiègne, pour avoir participé au jury. Tous deux m'ont par ailleurs soutenu et appuyé tout au long de cette thèse à Heudiasyc.

Mes remerciements vont aussi vers Simon Lacroix, Directeur de Recherches CNRS au LAAS, et Nicolas Marchand, Chargé de Recherches CNRS au GIPSA-Lab, eux aussi ayant accepté de faire parti de mon jury et m'ayant apporté des remarques constructives.

Je voudrais remercier notamment Nathalie Alexandre, Gildas Bayard, Corine Boscolo, Magali Collignon, Thomas Heurtel, Céline Ledent, Isabelle Martin, Eliane Mercier, Thierry Monglon, Véronique Moisan et Sabine Vidal des services administratifs et techniques du laboratoire pour leur aide tout au long de cette thèse.

Je tiens à saluer les doctorants de l'équipe que j'ai côtoyés et avec qui j'ai pu collaborer : David Lara, Juan Escareno, Luis-Rodolfo Garcia-Carrillo, Octavio Garcia-Salazar, Jose-Ernesto Gomez-Balderas, Jose-Alfredo Guerrero-Mata, Farid Kendoul, Laura-Elena Munoz-Hernandez, Hugo Romero, Eduardo Rondon, Jose-Luis Rullan-Lara, Sergio Salazar, Anand Sanchez, Duc-Anh Ta et Liu Xiaojie. Merci à eux pour les moments partagés et pour m'avoir permis de découvrir la culture mexicaine. Je remercie aussi les étudiants que j'ai eu l'occasion d'encadrer lors de ces travaux : Jérémy Brunet, Olivier Hababou, Eduardo Campos Mercado, Sharaku Gil Koshishi, Julien Lancry, José Miguel Muñoz, Adrian Partearroyo et Mihaita-Laurentiu Popa.

Merci également à Pedro Garcia et Angel Valera de la *Universidad Politécnica de Valencia*, pour m'avoir accueilli dans leur laboratoire lors de mes séjours en Espagne et avec qui j'espère continuer la collaboration.

Enfin, je voudrais remercier ma famille et mes amis pour leur aide et leur soutien durant cette thèse. Merci surtout à Valentina Espina qui m'a soutenu quotidiennement, même dans les moments difficiles, et sans qui je n'aurai pas pu finir cette thèse.

Table des matières

Résumé

Ce travail de thèse porte sur l'obtention de lois de commandes non linéaires pour la stabilisation d'un drone, ainsi que sur la localisation en utilisant la vision. Il comporte une grande partie expérimentale, avec notamment la réalisation de plateformes.

Une étude bibliographique des différents types de drones et de leurs lois de commandes a d'abord été menée. Ainsi plusieurs types de lois de commandes (linéaires et non linéaires) ont pu être testées et comparées à l'aide de *Matlab xPC Target*, sur une plateforme de type avion *PVTOL* (*Planar Vertical Take Off and Landing*) en temps réel. Une loi de commande stabilisant un système à n intégrateurs, basée sur des fonctions de saturations où chaque état est séparé a ensuite été proposée. Sa stabilité étant démontrée par une analyse de Lyapunov.

Un prototype de type quadrirotor a alors été construit. En tirant partie de l'expérience précédente de l'équipe sur ce type de plateforme, un grand soin a été apporté à sa réalisation ; tant dans les parties mécanique, électronique qu'informatique. Parallèlement, un simulateur pour quadrirotor a été développé. Celui-ci permet de faire tourner le programme du quadrirotor sur un ordinateur grâce à un modèle dynamique. Ainsi, le programme peut être *débuggé* et testé facilement avant le vol. De plus, le drone simulé est équipé de caméras virtuelles, fournissant alors des images d'un environnement 3D sous *OpenGL* aux algorithmes de vision à tester.

Des méthodes de traitement d'images ont ensuite été étudiées. Afin de répondre au critère de calcul embarqué, nous avons d'abord étudié une méthode légère de calcul de flux optique à une dimension, permettant à un robot mobile *e-puck* d'effectuer de l'évitement d'obstacles. Puis, des solutions basées sur la stéréovision mais utilisant des pointeurs lasers ont été proposées. La première technique permet d'estimer l'attitude à partir de trois pointeurs, la seconde permet d'effectuer du suivi de mur grâce à un laser ligne.

Enfin, un schéma de contrôle utilisant un prédicteur et un observateur a été

étudié. Cette combinaison prend en compte les éventuels retards dans la boucle de commande, ceux-ci pouvant venir par exemple du temps de traitement vidéo. De plus, le schéma proposé permet d'utiliser efficacement un capteur ayant une période d'échantillonnage élevée (*i.e.* une caméra) dans une loi de commande plus rapide.

Chapitre 1

Introduction

1.1 Histoire

Un drone aérien ou *UAV* (*Unmanned Air Vehicule* en anglais) ou plus simplement drone, est un engin volant sans pilote. Les premiers essais d'avions sans pilote datent de 1897. Le français Octave Détable invente en effet un système de stabilité automatique basé sur une voilure à cônes divergents et l'utilise sur un planeur. Ce principe intéresse alors fortement le capitaine Max Boucher qui veut l'installer sur un avion motorisé. Les essais sont interrompus par la guerre et reprennent en 1917 où le premier vol sans pilote d'un avion motorisé a lieu. L'avion, de type Voisin (voir figure 1.1), est équipé du stabilisateur qu'a breveté Détable. Après avoir décollé tout seul, il parcourt environ 1 km avant de se poser faute de carburant. Max Boucher poursuit ses études et améliore son système (stabilisation automatique, vol télécommandé, etc) jusqu'en 1924 où le projet s'arrête par manque de crédits. Malgré la démonstration d'un vol autonome de 180 km, l'armée française ne trouve alors pas d'intérêt à cette nouvelle technologie.

Le concept est repris dans les années 1930 afin de construire des avions télépilotés, servant de cibles pour les entrainements de défense anti-aérienne. Plus tard, des bombes volantes ou missiles (type *V1* ou *V2*) sont conçus par l'armée allemande. Même si les missiles ne font aujourd'hui plus partie de la famille des drones (car non récupérables), ces inventions (bien que meurtrières) représentent une certaine avancée dans le domaine. C'est enfin pendant la guerre froide que les drones ont pris leur essor, les États-Unis développant alors des avions sans pilotes capables de faire de la reconnaissance sans engager de vies humaines sur le terrain ennemi.

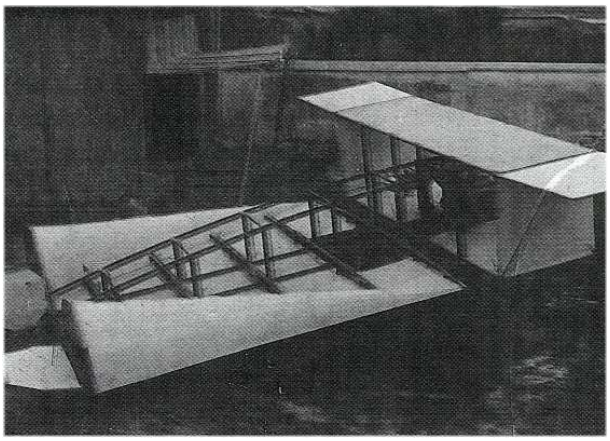


Figure 1.1 – Le premier avion motorisé sans pilote. Source : site *Aéroplanes Détable* [ita].

De nos jours, les drones sont devenus beaucoup plus accessibles. En effet, la technologie ayant évoluée, les systèmes se sont miniaturisés et les coûts ont baissé. De nombreuses équipes de recherche travaillent maintenant sur des drones miniatures (en général électriques), et la production scientifique sur le sujet est donc abondante. De nouvelles applications s'ouvrent alors au drone qui ne sont plus qu'à finalités militaires mais aussi civiles : surveillance des feux de forêt, surveillance du trafic autoroutier, inspection des ouvrages d'arts, inspection des lignes électrifiées, etc. Malgré tout, ces applications restent dans le domaine de l'observation ; le transport de marchandises ou de personnes par exemple n'étant pas d'actualité.

1.2 Problématique

La problématique principale des drones est la stabilisation. En effet, l'engin étant autonome il doit être capable de mesurer son orientation, et éventuellement sa position, afin de pouvoir voler de manière stable pour réaliser des applications de surveillance. Ce problème se divise donc en deux parties distinctes.

D'une part l'estimation de l'orientation et de la position. Il s'agit bien d'une estimation car aucun capteur ne délivre une telle information brute. Il faut donc avoir recours à différents capteurs afin de fusionner les données. Dans le cas de l'orientation, ces capteurs sont généralement de types inertiels (accéléromètres et gyromètres), aidés de magnétomètres pour le cap. Pour la position, les systèmes les

plus communs sont le *GPS*, les télémètres ou les caméras. Position et orientation peuvent d'ailleurs être estimées conjointement afin d'améliorer la précision. Celle-ci est l'aspect le plus important de l'estimation ; un système n'étant utile que s'il est précis, et exempt de dérives. Un autre enjeu important est la fréquence de l'estimateur. Les drones pouvant avoir une dynamique rapide, les informations d'orientation et de position doivent pourvoir être fournies à une cadence élevée, en minimisant les retards. Enfin, la clé de cette estimation passe par le filtrage des capteurs et de leur bruit. En effet, une fois embarqués dans le drone ceux-ci sont alors sujets à de nombreuses perturbations : vibrations mécaniques, champs magnétiques des moteurs électriques, accélérations latérales, canyons urbains pour le *GPS*, conditions d'éclairages changeantes pour la caméra, etc.

D'autre part, une fois l'orientation et la position estimées, un autre problème se présente. Ces informations doivent être utilisées afin de stabiliser efficacement le drone. Or le modèle de celui-ci est généralement non linéaire et peut présenter de forts couplages entre les différents états. Des lois de commandes non linéaires doivent donc être conçues, souvent en passant par quelques simplifications du modèle. Certains phénomènes aérodynamiques sont alors négligés, des équations linéarisées, et des variables découplées. Les contraintes du système réel doivent aussi être prises en compte, et notamment le fait que les actionneurs soient limités. Les lois de commandes utilisées doivent donc pouvoir être bornées. De plus, les lois doivent si possible être conçues et réglées de manière à optimiser l'autonomie du drone.

Cette thèse s'inscrit donc dans ces deux problématiques. Tout d'abord, l'objectif est d'étudier l'estimation de la localisation du drone en utilisant la vision. Celle-ci peut en effet compléter les capteurs habituellement employés mais qui possèdent certains désavantages. Les centrales inertielles miniatures par exemple sont relativement onéreuses et ont l'inconvénient d'avoir des dérives. Le GPS quant à lui ne fournit pas toujours des informations suffisamment précises à l'échelle d'un drone, de plus les phénomènes de canyons urbains peuvent rendre les données GPS inutilisables dans certains milieux. Les systèmes de vision sont en effet de plus en plus populaires en robotique, car les informations que peut délivrer une caméra sont beaucoup plus nombreuses et variées que tout autre type de capteur. Notamment, ce capteur est très adapté dans des environnements peu ou pas connus (tâches d'exploration). Cependant cette quantité d'information demande des ressources supplémentaires pour être traitée. La vision pourrait donc être utilisée pour détecter des repères dans l'environnement du véhicule de manière à estimer sa position et/ou son orientation, ou pour calculer le flux optique et en déduire les vitesses linéaires de déplacement. Par ailleurs, afin d'obtenir un drone de grande autonomie, l'objectif de la thèse est de pouvoir embarquer le traitement lié à la vision. Enfin, afin de répondre à la problématique de la stabilisation des drones,

le but de la thèse est aussi de proposer des lois de commandes (non linéaires) qui devront être validées sur une plateforme expérimentale.

1.3 Organisation de la thèse

La suite de ce rapport est organisée de la façon suivante. D'abord un état de l'art (chapitre 2) sur les différentes configurations de drones est dressé. Le but est de privilégier une configuration qui servira par la suite aux essais expérimentaux. La configuration à retenir devra donc être mâture (sa stabilisation ne doit pas être un problème encore ouvert) et propice aux essais de vision. Puis, la plateforme choisie et construite est analysée en détails dans le chapitre 3. Son modèle dynamique est donc présenté, ainsi que tous ses organes principaux. Le chapitre suivant (4) traite des lois de commandes. En se basant sur un modèle simplifié du drone choisi, plusieurs lois (linéaires ou non) sont étudiées. Une loi de commandes stabilisant un système à n intégrateurs en cascade est alors proposée, en utilisant des fonctions de saturations. Des notions de visions (chapitre 5) sont ensuite présentées : modèle et calibration de la caméra, flux optique et stéréovision. Des applications basées sur ces deux techniques sont alors proposées, en tenant compte des restrictions du matériel embarqué. Enfin, le dernier chapitre (6) introduit un prédicteur et un observateur afin d'améliorer l'algorithme de commande basé sur la vision. Le schéma proposé permet de prendre en compte le retard de la mesure introduit par la caméra ainsi que sa cadence irrégulière.

Les conclusions sur ces travaux de thèses sont données dans la partie 6.5. Les perspectives ainsi que les travaux futurs y sont alors discutés.

Chapitre 2

État de l'art des différents types de drones

Ce chapitre expose les différentes configurations possibles de drones, afin d'étudier leurs caractéristiques, avantages et inconvénients. Le but étant ici de choisir la configuration la mieux adaptée à la partie expérimentale de cette thèse. La liste dressée dans ce chapitre n'est pas exhaustive, elle représente néanmoins les drones les plus courants. Il est à noter par ailleurs que certaines des configurations existent aussi à plus grande échelle et peuvent être "habitées". Enfin certaines applications de vision seront aussi présentées dans ce chapitre, avec leurs plateformes respectives.

Il existe de nombreuses classifications des drones, en fonction de leur taille, de leur rayon d'action, de leur charge utile, du type de configuration... La classification retenue ici est fonction de la configuration. Il existe en effet quatre grandes familles de drones : ceux à voilure fixe, ceux à voilure tournante, ceux à ailes battantes et les plus légers que l'air. Les drones à voilures fixes sont les avions, dont les ailes assurent la portance. Les drones à voilures tournantes sont de type hélicoptère. Ce sont alors les pales des hélices qui assurent la portance. Cette famille peut elle-même se diviser en plusieurs types de configurations suivant le nombre de rotors. Les drones à ailes battantes sont des drones reproduisant les mouvements des oiseaux ou des insectes volants. Enfin les drones plus légers que l'air sont du type dirigeable, montgolfière ou ballons sondes.

2.1 L'avion

L'avion (voir figure 2.1) est un drone à voilure fixe, la propulsion étant assurée par un ou plusieurs moteurs. L'avion est donc une configuration économique, car l'énergie dépensée ne sert qu'à le faire avancer, et non pas à le porter. Sa vitesse d'avance peut d'ailleurs être très rapide. C'est donc une configuration à privilégier pour des missions de longue durée ou de grande distance.

Figure 2.1 – L'avion *Lord 400*. Source : site *Newpower Modélisme* [itk].

Les différentes gouvernes permettent de contrôler les moments de tangage, roulis et lacet. Les avions nécessitent en principe une piste pour pouvoir décoller ou atterrir. Cependant les petits modèles peuvent être lancés à la main (tel un avion en papier), en mettant préalablement un niveau de gaz suffisant. La récupération peut aussi se faire à la main, mais est plus délicate. L'utilisation de ce type de drone nécessite donc presque toujours un pilote expérimenté, à moins que le drone soit parfaitement autonome et capable de décoller et d'atterrir en un point précis, grâce à un système de positionnement ; ce qui reste rarement le cas.

Les applications de vision sur les avions sont assez nombreuses. Citons par exemple celle originale de l'*EPFL*, [ZKB+07]. Leur avion est ultra léger (10 g) et capable de faire de l'évitement d'obstacle grâce au flux optique, voir figure 2.2. Le flux optique est calculé grâce à un capteur optique linéaire de 102 pixels (*TSL3301* de *Taos*), pouvant fonctionner jusqu'à 1000 Hz, couplé avec un gyromètre. L'avion possède ainsi deux systèmes de vision : l'un frontal et l'autre pointant vers le bas. Le traitement des données provenant des capteurs est effectué dans le microcontrôleur embarqué, ce qui est encore assez rare dans ce domaine. L'ensemble du système

est donc simple et efficace. La résolution du capteur est toutefois faible, ce qui nécessite un environnement maitrisé ayant un fort contraste.

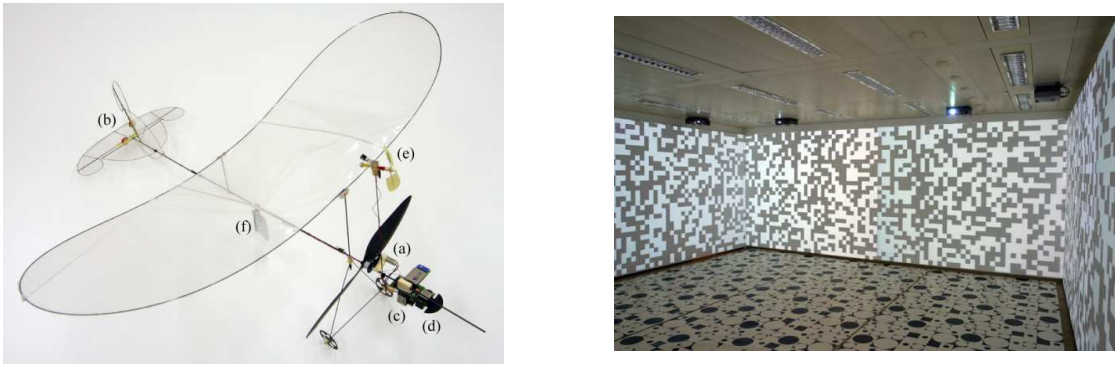


Figure 2.2 – Prototype d'avion ultra léger de l'*EPFL* et sa salle de vol. Source : [ZKB+07].

Les applications de vision les plus courantes sur les avions sont cependant les suivis de cibles : véhicule se déplaçant au sol, route... Par exemple l'application développée par [RKS06] permet de suivre une route, après une phase d'apprentissage, grâce à un système de vision embarqué fonctionnant à 5 Hz. L'une des principales difficultés pour un avion est de pouvoir suivre une route, même si son rayon de courbure est relativement élevé. En effet, lorsque l'avion vire, l'objet suivi a tendance à sortir du champ de vision de la caméra. Ainsi, dans [RKS06] l'avion doit voler relativement haut (100 m) pour éviter ce genre de problèmes. Une autre solution est d'avoir une caméra ayant deux degrés de liberté afin de l'orienter en fonction de la situation, ce qui peut tout de même poser des problèmes d'encombrement ou de poids. Au contraire, [Egb07] utilise une caméra fixe mais prend en compte la manœuvrabilité de l'appareil dans la loi de commande afin de garder la cible dans une zone spécifique de la caméra et de ne pas la perdre.

2.2 Le monorotor

Le monorotor est un drone possédant un seul rotor. Contrairement à l'avion, qui lui aussi peut ne posséder qu'un seul rotor, c'est une configuration à voilure tournante et donc volant verticalement. Par ailleurs un avion bien conçu peut lui aussi effectuer le vol vertical ; c'est la configuration la plus simple du monorotor. Les mouvements de tangage, roulis et lacet sont alors obtenus grâce aux ailerons. D'autres configurations possèdent un ou deux degrés de liberté sur le rotor. C'est la cas du monorotor représenté figure 2.3, possédant un rotor pivotant sur un axe.

Ce degré de liberté permet de contrôler le moment de roulis, alors que les deux ailerons contrôlent à la fois le moment de tangage (déflexion parallèle) et de lacet (déflexion différentielle). L'intérêt de cette configuration est que la commande de roulis garde l'avion vertical, tout en ayant une mécanique simple. L'ajout d'un deuxième degré de liberté serait en effet plus compliqué mécaniquement. Notons que cette configuration a l'inconvénient d'avoir une prise au vent importante. Le vol en extérieur en présence de rafales de vent est donc plus difficile à stabiliser.

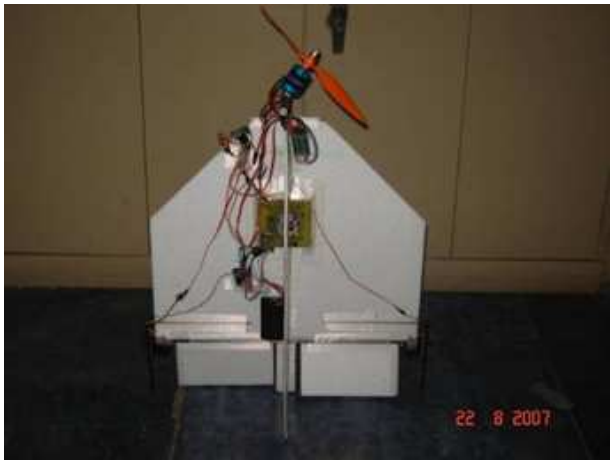


Figure 2.3 – Le monorotor. Source : [GSEL08].

Les applications de vision sur cette plateforme sont très rares, voire inexistantes.

2.3 Le birotor

Le birotor est un drone possédant deux rotors. La configuration de ce type la plus connue est l'hélicoptère "classique" (par abus de langage, toutes les configurations à voilures tournantes peuvent être appelées hélicoptère). La figure 2.4 montre le *Yamaha Rmax*, l'un des hélicoptères thermiques le plus utilisé pour des applications drones ; sa charge utile étant en effet de près de 30 kg.

L'hélicoptère classique possède donc un rotor principal assurant la poussée verticale, et un rotor de queue aussi appelé anti-couple. Celui-ci permet de compenser le couple créé par le rotor principal et évite à l'hélicoptère de tourner sur lui même. Les pales de chacun des rotors sont à pas variable ; c'est en effet l'incidence des pales qui permet de déplacer l'engin, les vitesses de rotation des rotors restant

Figure 2.4 – Hélicoptère *Yamaha Rmax*. Source : site *Yamaha* [itw].

quant à elles constantes. La mécanique d'un tel engin est alors relativement complexe. Cela peut donc poser problème pour une plateforme expérimentale en cas de chute de l'engin et de rupture d'un des éléments du plateau cyclique ; les réparations pouvant alors être difficiles.

Les autres configurations à deux rotors sont les tandems (voir figure 2.5) et les contra rotatifs (voir figure 2.6). La disposition de leurs rotors permet d'annuler le couple de lacet et les hélices peuvent être à pas variables ou à pas fixes. Dans le cas de pas fixes, se sont des gouvernes qui assurent les déplacements. Ce type de configuration, plus simple mécaniquement est donc plus adapté aux plateformes expérimentales. De plus, les couples de lacet s'annulant par eux mêmes sans utilisation de rotor anti-couple, il n'y a ainsi pas d'énergie dépensée inutilement. Cependant les applications de vision sont beaucoup plus rares que pour les hélicoptères classiques.

L'hélicoptère classique ayant une charge utile importante (dans sa version thermique), il est très populaire dans les applications de vision. En effet, cela permet d'emporter des ordinateurs et des capteurs puissants sans avoir à se soucier des contraintes de poids. Cependant, ce type d'hélicoptère évolue rarement en intérieur, du fait de sa taille et des gaz d'échappement qu'il faudrait alors évacuer. Les applications de vision se font donc principalement en extérieur, où les conditions d'éclairage ne sont pas toujours favorables et souvent changeantes. L'hélicoptère électrique quant à lui serait susceptible de voler en intérieur mais sa charge utile est beaucoup moins intéressante.

Ainsi, parmi les applications de vision sur les hélicoptères classiques, beaucoup ressemblent à celles des avions (suivi de cible, de route, etc), l'hélicoptère ayant cependant l'avantage d'être beaucoup plus manœuvrable qu'un avion et les problèmes de pertes de cible sont moins fréquents. Les applications de décollage ou

Figure 2.5 – Birotor en configuration tandem.

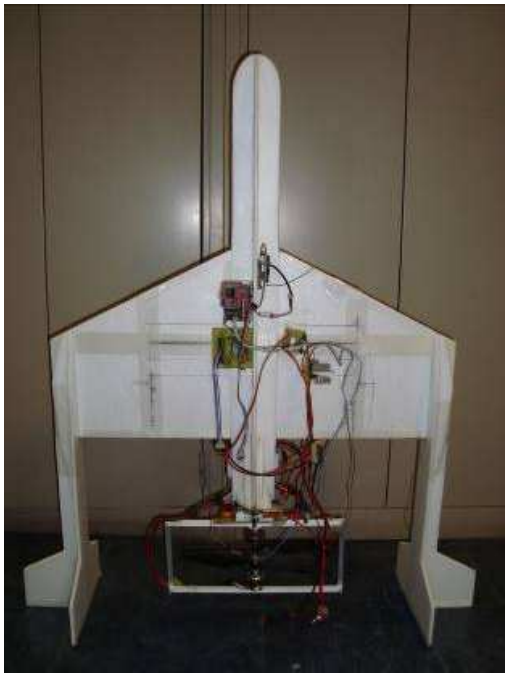


Figure 2.6 – Birotor en configuration contra rotatif.

atterrissage automatiques sont quant à elles plus développées que pour l'avion, car plus faciles à réaliser. Par exemple, [SMS03] présente un système d'atterrissage automatique utilisant un motif au sol (voir figure 2.7). Une caméra sans fil pointant vers le bas est installée dans un hélicoptère thermique, permettant de reconnaitre et de localiser le marqueur. Cependant, l'altitude est estimée grâce à un sonar et à un *GPS*. D'autres travaux, tels que [YNSC07], présentent un système de localisation par stéréovision. Dans ce cas, le marqueur au sol n'est plus nécessaire et l'altitude est directement estimée grâce aux caméras. Ce travail propose d'ailleurs deux lois de commandes pour l'atterrissage en fonction de l'altitude afin de prendre en compte l'effet de sol.

2.4 Le trirotor

Le trirotor (voir figure 2.8) est une configuration à trois rotors. Étant donné que les couples créés par les moteurs ne peuvent pas s'annuler par eux mêmes, comme dans le cas des birotors de type contra rotatifs ou tandems, il est nécessaire d'avoir

Figure 2.7 – Hélicoptère et marqueur au sol. Source : [SMS03].

au moins un degré de liberté sur un moteur afin de contrôler l'angle de lacet. Cependant, il existe toujours un certain couplage. Dans [SL05] seul le rotor de queue est inclinable ; le bras de levier de ce rotor est alors plus important que ceux des rotors frontaux. Le désavantage de cette configuration est son asymétrie qui peut poser problème pour son équilibrage, notamment pour emporter une charge utile. Plus récemment, dans [ESGL08], cette configuration a été améliorée en rendant les trois rotors mobiles ; la configuration devenant ainsi symétrique.

Une autre possibilité est de ne pas contrôler l'angle de lacet. C'est le cas du *Vectron* de *BlackHawk* (voir figure 2.9) ; il s'agit en fait d'un jouet dont l'effet gyroscopique provoqué par la rotation permet de le stabiliser. Une force vectorielle permet ensuite de le déplacer. En effet le *Vectron* est piloté par une télécommande infrarouge et est équipé de plusieurs récepteurs afin de connaitre la position de la télécommande à tout instant. Il peut alors adapter la vitesse des moteurs pour fournir un contrôle vectoriel du drone. Dans [RLP05], le *Vectron* a été équipé d'un inclinomètre afin d'aider à la stabilisation. Étant donné que le corps du drone tourne sur lui même et que son angle de lacet est connu par rapport à la télécommande, un seul capteur est nécessaire pour contrôler les angles de roulis et tangage.

L'avantage des configurations à trois rotors est leur relative simplicité. En effet les hélices sont ici à pas fixe et le drone ne possède pas d'ailerons pour la stabilisation (réduisant aussi la prise au vent de l'engin) ; seul un degré de liberté sur un moteur est nécessaire. De plus c'est une configuration économique, car seuls trois moteurs sont utilisés, contrairement aux configurations qui suivent (voir sections 2.5, 2.6 et 2.7) et ayant les mêmes caractéristiques (hélice à pas fixe, sans ailerons). L'autonomie est donc augmentée, mais au détriment de la charge utile. Ainsi le trirotor est moins adapté pour embarquer des systèmes de vision. De manière

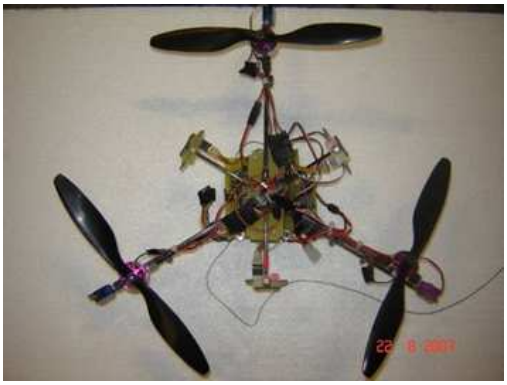


Figure 2.8 – Trirotor.

Figure 2.9 – Le *Vectron*. Source : site *Vectron* [itr]

générale, ce drone est peu utilisé par les équipes de recherche, les applications de traitement d'image sont donc très rares voire inexistantes.

2.5 Le quadrirotor

Le quadrirotor (parfois appelé quadrotor) est sans doute le prototype le plus utilisé. Il a été inventé par *DraganFly*, sous le nom de *X4* du fait de sa structure en *X* ayant un rotor à l'extrémité de chacun de ses quatre bras. La dernière version du *X4* (non sortie à ce jour) est montrée sur la figure 2.10. Selon le constructeur, cette version a été prévue pour les équipes de recherche, donnant accès aux capteurs et au contrôle des moteurs. Le succès du *X4* a rendu très populaire cette configuration, d'autres compagnies commercialisent maintenant des quadrirotors tels que *Xufo* ou *Mikrokopter* (ce dernier ayant l'avantage d'être *open source*[1]). De nombreux amateurs se construisent aussi leur propre plateforme. En effet la forme de la structure lui permet de ne pas avoir d'autre actionneur que ses quatre moteurs, ce qui la rend mécaniquement très simple et facile à construire. Les moments de roulis et tangage sont obtenus en jouant sur les vitesses de rotation de deux moteurs opposés ; de plus, deux des quatre rotors ont une hélice à pas inverse ce qui permet d'annuler le couple de lacet (contrairement au trirotor).

Le quadrirotor étant très populaire, de nombreuses applications de vision ont été développées sur cette plateforme. Ainsi, dans [HRHM08], un algorithme d'atter-

1. Le *Xufo* possède lui aussi depuis peu (18 février 2010) un *firmware open source*, voir [tpltea].

Figure 2.10 – Le dernier *X4* de *Draganfly*, sortie prévue en automne 2009. Source : site *Draganfly* [itc].

rissage automatique est proposé. Celui-ci utilise le flux optique divergeant provenant de la caméra (sans fil) regardant au sol et les données des gyromètres. Il est alors montré qu'en assurant un flux optique divergeant constant, l'atterrissage peut se faire en douceur. Par ailleurs, le calcul du flux optique permet aussi de stabiliser les deux autres vitesses translationnelles (suivant les axes x et y). Sur le même principe, [HHMR09] utilise le flux optique divergeant pour faire du suivi de terrain. Ainsi, la caméra regardant vers le bas permet de garder le drone à une altitude constante par rapport au sol même si celui-ci est en pente. Une caméra placée latéralement pourrait alors aussi permettre de suivre un mur. Les vidéos de ces deux expériences sont disponibles sur internet (voir [dedsdtpfo]) et montrent la fiabilité de l'algorithme. Leur seul inconvénient étant que le calcul n'est pas embarqué mais déporté. Par ailleurs, la stabilité en orientation est confiée aux capteurs inertiels.

L'*AR.Drone* de *Parrot* (voir figure 2.11) quant à lui possède deux caméras : une frontale et une verticale. La première permet de retransmettre la vidéo au sol (sur un *IPhone*, servant aussi à piloter l'engin) ; la deuxième (voir figure 2.12) est couplée aux capteurs inertiels (accéléromètres et gyromètres) afin d'estimer les vitesses latérales de déplacement. Cependant, peu de détails sont donnés sur la nature de l'algorithme utilisé (sans doute un type de flux optique, avec un filtre de Kalman). Par contre le constructeur spécifie que ces calculs se font en embarqué, sur un processeur *ARM9* 32 bits à 468 MHz ; sûrement directement fabriqué par *Parrot*. La caméra verticale est de faible résolution (176x144 pixels), afin de permettre un traitement à 60 images par secondes (données constructeur) ; les vidéos en vol du drone sont disponibles sur le site de *Parrot* ([idldt]) et montrent une bonne stabilité.

Figure 2.11 – L'*AR.Drone*. Source : [idldt].

Figure 2.12 – Vue de dessous de l'*AR.Drone* ; la caméra verticale est repérée par un cercle rouge. Source : [idldt].

2.6 L'hexarotor X6

Figure 2.13 – Le *X6* de *Draganfly*. Source : site *Draganfly* [itc].

L'hexarotor, ou *X6*, est le dernier drone de *DraganFly*. Il est composé de trois bras, tel que le tritor vu précédemment (voir section 2.4), à la différence que chacun des bras comporte deux rotors contra-rotatifs (voir figure 2.13). Cette configuration permet ainsi d'annuler le couple de lacet, et d'éviter de rajouter des actionneurs sur

les bras pour contrôler l'angle de lacet comme vu dans la section 2.4. Les six rotors permettent au *X6* d'emporter une charge utile assez importante, notamment une caméra ou un appareil photo numérique. Le *X6* étant en effet destiné à la surveillance ou pour réaliser des prises de vues aériennes pour des films. Contrairement au futur *X4*, cette plateforme n'est pas ouverte ; les informations des capteurs ne sont donc pas directement récupérables. Cela rend difficile la possibilité d'ajouter ses propres algorithmes sur la plateforme ; les applications de traitement d'image sont ainsi inexistantes sur le *X6*.

2.7 L'octarotor

L'octarotor est un prototype (voir figure 2.14) d'hélicoptère à huit rotors. Il a été conçu au laboratoire *Heudiasyc* [RSSL07], [RSS+07]. Ce drone possède quatre rotors principaux tout comme le quadrirotor. Ceux-ci assurent la portance et permettent de contrôler les moments de tangage, roulis et lacet. Les quatre autres rotors permettent de contrôler les déplacements latéraux. Cette configuration permet alors de découpler la dynamique de rotation de la dynamique de translation, ce qui n'est pas le cas dans les autres configurations présentées.

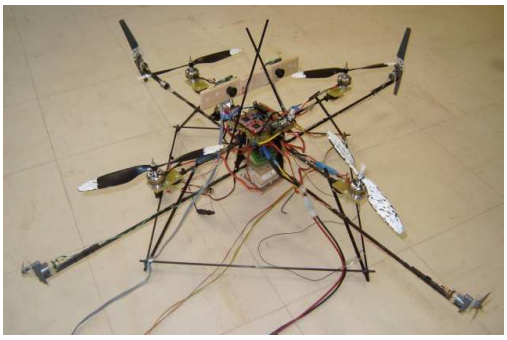


Figure 2.14 – Octarotor.

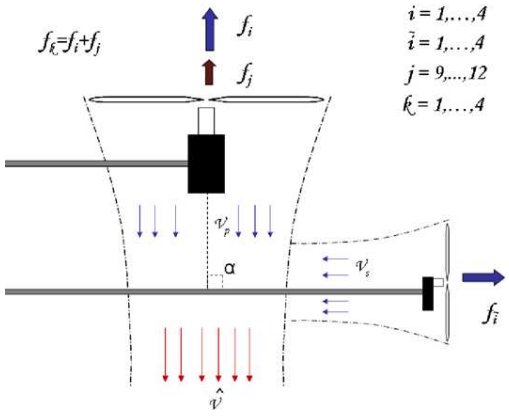


Figure 2.15 – Précompensateur. Source : [RSSL07].

Cependant, cette configuration présente quelques inconvénients. En effet, la présence des rotors latéraux à proximité des rotors principaux perturbent ces derniers. Ainsi, le flux d'air généré par un rotor latéral augmente la portance du rotor principal, provoquant un déséquilibre au niveau des couples de tangage

ou de roulis ; les deux dynamiques ne sont donc pas tout à fait découplées. Un pré-compensateur a donc été introduit dans la loi de commande pour réduire ce phénomène, voir figure 2.15. Celui-ci permet d'adapter la vitesse de rotation du rotor principal en fonction de celle du rotor latéral. Un autre inconvénient de la plateforme est son poids, dû à l'ajout des moteurs latéraux, réduisant significativement son autonomie et sa charge utile. Le prototype ne peut en effet voler que quelques minutes ; la plupart des essais ont d'ailleurs été faits avec une alimentation de laboratoire et non une batterie.

L'octarotor a été développé dans le but de tester des algorithmes de vision pour asservir la position et/ou la vitesse du drone. Deux applications ont ainsi été développées, l'une basée sur la stéréovision ([Rom08]) et l'autre sur le flux optique ([SRL08]). Dans les deux cas, la stabilisation en orientation est faite grâce à la centrale inertielle embarquée, alors que l'algorithme de vision permet de se positionner par rapport à une cible (stéréovision) ou de contrôler les vitesses de déplacement latéral (flux optique). Des essais ont aussi été conduits en utilisant les deux techniques de vision ([Rom08]).

Enfin notons que *Draganfly* a aussi prévu de sortir une configuration à huit rotors en 2010 ; de rares photos sont pour l'instant disponibles (telle que celle présentée figure 2.16) sur le site internet du fabricant. Cette photo montre que le *X8* sera en fait une configuration de type *X4*, avec deux moteurs contra rotatifs sur chaque bras. Cette configuration ne semble pas apporter un grand intérêt par rapport au *X4* ; contrairement à la configuration *X6* qui améliorait grandement le trirotor. L'intérêt annoncé du *X8* est sa charge utile qui devrait être plus importante.

2.8 Les convertibles

Les drones convertibles sont des engins volants capables d'effectuer le vol horizontal et le vol vertical. Ils tirent ainsi parti des avantages des configurations à voilure fixe et à voilure tournante. En effet l'hélicoptère a la capacité de décoller et d'atterrir sans piste, ce qui le rend plus indépendant ; il peut aussi effectuer le vol stationnaire pour surveiller une zone par exemple. L'inconvénient de ce type de configuration est que la portance est assurée par les parties "tournantes", ce qui nécessite beaucoup d'énergie. Sur les drones à voilure fixe la portance est assurée par les ailes, ce qui rend le vol beaucoup plus économique. Les convertibles sont donc capables de réaliser les deux types de vol mais surtout d'effectuer la transi-

Figure 2.16 – Le *X8* de *Draganfly*, sortie prévue en 2010. Source : site *Draganfly* [itc].

tion, c'est à dire le passage d'une configuration à l'autre. Cette étape est cruciale et difficile à effectuer. De nombreux travaux traitent de ce sujet, voir par exemple [GO05], ou [SC01]. Les drones convertibles sont généralement de type monorotor ou birotor, tels que présentée dans les sections 2.2 et 2.3. La figure 2.17 montre par exemple le *V-Bat* de *MLB Company*.

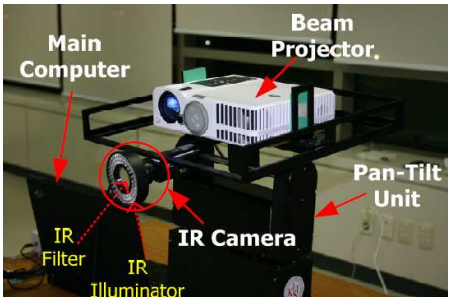
Figure 2.17 – Drone convertible *V-Bat* de *MLB Company*. Source : site *MLB Company* [itj].

2.9 Les dirigeables

Le dirigeable (voir figure 2.18) est un drone de la catégorie plus léger que l'air, contrairement aux plus lourds que l'air présentés précédemment. Les dirigeables ont l'avantage de pouvoir faire du vol stationnaire et d'avoir une charge utile importante. De plus, ils ont une grande autonomie car la portance est assurée par l'enveloppe et ne nécessite donc aucune énergie. Le déplacement est ensuite assuré grâce à plusieurs rotors orientables par rapport au corps du ballon. Cependant, ces plateformes sont très encombrantes et requièrent un hangar de stockage. Par ailleurs elles nécessitent plusieurs personnes pour pouvoir les manipuler sur le terrain notamment lors des opérations d'amarrage. Un autre inconvénient du dirigeable est sa forte prise au vent due à la taille de l'enveloppe.

Figure 2.18 – Dirigeable du *LSC* (Laboratoire des Systèmes Complexes, Université d'Evry). Source : site *LSC* [idth].

Une application de vision originale pour un dirigeable est proposée par [SSK+06]. Un vidéoprojecteur est utilisé pour dessiner une image sur l'enveloppe (voir figure 2.19). Une caméra placée à l'extérieur permet alors de calculer la position et l'orientation du drone par rapport à un repère fixe. Les calculs de vision sont donc ici déportés. L'ensemble caméra et projecteur est par ailleurs placé sur une tourelle de type "pan-tilt" afin de pouvoir projeter l'image sur le dirigeable quand ce dernier se déplace. L'inconvénient de ce système est cependant qu'il est fortement dépendant de la station sol et du système de vision fixé au sol.



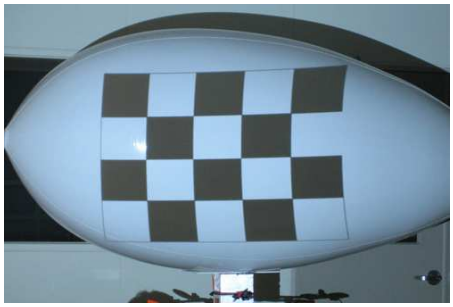


Figure 2.19 – Le dirigeable et son dispositif de vision. Source : [SSK+06].

2.10 Les drones à ailes battantes

Ce type de drone cherche à imiter les oiseaux, mais reste peu développé, le contrôle et la stabilisation d'un tel engin étant encore un domaine ouvert (voir par exemple [Rak06], [RMP08]). Cette configuration a généralement une charge utile très faible. En effet, les plateformes actuelles sont de type microdrone, voir figure 2.20, et de plus leur niveau d'autonomie est relativement faible. Par ailleurs le seul oiseau capable de faire le vol stationnaire est le colibri (oiseau mouche), mais les plateformes actuelles ne sont pas encore réellement capables de l'imiter. Cependant, les drones à ailes battantes ont l'avantage de pouvoir décoller et atterrir verticalement, d'avoir une consommation électrique relativement faible et d'avoir une grande manœuvrabilité. Ces plateformes se rapprochent donc des configurations convertibles au niveau des avantages. L'atout du drone à ailes battantes (que l'on ne retrouve sur aucune autre plateforme), est l'absence de rotor ce qui le rend beaucoup moins dangereux. Ces drones ont donc un énorme potentiel, et risquent de susciter un intérêt de plus en plus fort.

L'*ONERA (Office National d'Etudes et de Recherches en Aérospatiales*) s'y intéresse notamment depuis 2002 et le lancement du projet *REMANTA* (*REsearch programme on Microvehicle And New Technologies*) d'un micro véhicule à ailes battantes. Ce projet étant vaste, il a été découpé en trois domaines de recherches :

- dynamique de vol et commande,
- aérodynamique à bas Reynolds,
- matériaux légers, structures et actionneurs.

Des avancées étant en effet nécessaires dans chacun de ces domaines, d'autant que le drone visé devait avoir une envergure de 15 à 20 cm, une vitesse en vol de 10 m/s, un poids de 50 à 100 g et une fréquence de battement de 30 à 40 Hz. Les autres thèmes (tels que l'énergie, les capteurs, etc) n'ont pas été étudiés dans ce projet car le but n'était pas d'arriver à un prototype de drone mais plutôt d'enrichir

les connaissances scientifiques et techniques sur le sujet. Ayant pris fin en 2006, le projet a permis entre autres d'écrire un modèle de simulation nommé *OSCAB* (*Outil de Simulation de Concept d'Ailes Battantes*, voir [RMO04] et [ROM07]), ainsi que de montrer la contrôlabilité du système (voir [ROM06]). Notons que des prototypes d'ailes et d'actionneurs ont aussi été construits et testés.

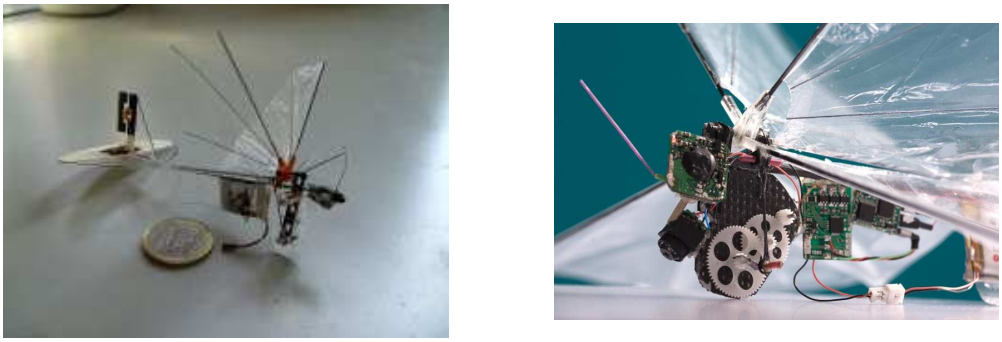


Figure 2.20 – Les drones à aile battante *DelFly*. Source : site internet *DelFly* [idta].

Comme il peut être vu sur la figure 2.20, le drone de *DelFly* possède une caméra embarquée. Celle-ci permet d'envoyer la vidéo à la station sol qui détermine alors le flux optique rotationnel afin que le drone suive la commande imposée par le pilote. Le drone ne possède pas d'autres capteurs pour aider à sa stabilisation. C'est une des rares applications de vision sur les drones à ailes battantes.

2.11 Conclusion, choix d'une plateforme

Il existe donc de nombreux types de plateformes différentes ; afin d'effectuer les expériences de vision une plateforme devait être retenue ou privilégiée. Les configurations type avion ont été écartées car leur mise en œuvre est trop compliquée : ils nécessitent un terrain pour décoller, une fois en vol il faut ensuite savoir le faire atterrir et éviter les crashs. De plus les avions ne sont pas capables de vol stationnaire et ont un mouvement d'avance rapide, certains algorithmes de vision par amers visuels ne sont donc pas adaptés. Par ailleurs les avions évoluent en extérieur ce qui peut rendre les traitements vidéos plus compliqués (changement d'éclairage...). Les convertibles en mode avion ont ainsi les mêmes inconvénients que ceux cités ci-dessus. Une plateforme de type convertible n'est donc intéressante (dans le cadre de cette thèse) qu'en mode hélicoptère. De même les dirigeables n'ont pas été retenus pour leur difficulté de mise en œuvre ; le laboratoire n'étant pas du tout équipé pour. Enfin les drones à ailes battantes semblent nécessiter encore beaucoup trop

de travail au niveau de la stabilité avant de pouvoir développer des applications de vision. Le choix s'est donc porté sur une plateforme de type hélicoptère électrique, capable de vol stationnaire et en intérieur. Parmi ces plateformes, la plus simple est le quadrirotor. En effet ce drone ne nécessite pas d'autres actionneurs que ses quatre rotors principaux ; il est donc facile à maintenir et faire évoluer.

Chapitre 3

Présentation de la plateforme

Ce chapitre présente d'abord les équations dynamiques du quadrirotor retenu comme plateforme pour cette thèse. Il présente ensuite en détail chacun des organes du drone et le choix de certaines solutions technologiques. Notons toutefois que ces éléments ne sont pas nécessairement spécifiques au quadrirotor et peuvent être utilisés pour d'autres configurations de drones.

3.1 Modèle dynamique

3.1.1 Préliminaires

Afin d'écrire les équations dynamiques du drone, définissons d'abord la matrice de rotation permettant de passer d'un système de coordonnées fixes $R_f(x_f, y_f, z_f)$ au système de coordonnées du drone $R_b(x_b, y_b, z_b)$. Cette matrice est aussi appelée matrice des cosinus directeurs, ou DCM(*Direct Cosine Matrix*). Le passage d'un repère à l'autre s'effectue en trois étapes :

- rotation de ϕ autour de x_f, avec $\phi \in [-\pi, \pi]$,
- rotation de θ autour de y_f, avec $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$,
- rotation de ψ autour de z_f, avec $\psi \in [-\pi, \pi]$.

Les angles ϕ, θ, ψ sont appelés angles d'Euler. Ils représentent respectivement les angles de roulis, tangage et lacet. Ils sont aussi souvent appelés par leurs noms

anglais, soit *roll*, *pitch* et *yaw*. Les matrices de rotation correspondantes sont donc :

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{3.1a}$$

$$R_\theta = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3.1b}$$

$$R_\psi = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1c}$$

En notant $c\alpha = cos(\alpha)$ et $s\alpha = sin(\alpha)$, on obtient :

$$R_{fb} = R_\psi R_\theta R_\phi = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & c\psi c\phi s\theta + s\psi s\phi \\ s\psi c\theta & s\phi s\theta s\psi + c\psi c\phi & s\theta s\psi c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{3.2}$$

En notant $R_{bf} = {R_{fb}}^{-1} = {R_{fb}}^T$, la relation entre les coordonnées V_f d'un vecteur dans le repère fixe et ses coordonnées V_b dans le repère mobile est donnée par :

$$V_f = R_{fb} V_b \tag{3.3a}$$

$$V_b = R_{bf} V_f \tag{3.3b}$$

Il est à noter que la représentation par les angles d'Euler est simple et explicite, cependant elle présente une singularité lorsque $\theta = \pm\frac{\pi}{2}$. Il faut donc veiller à ne pas se retrouver dans cette configuration. Dans le cas du quadrirotor, si cette configuration venait à arriver, alors l'hélicoptère n'aurait plus de portance et tomberait. On peut donc considérer que l'orientation du drone reste limitée et que la singularité n'arrivera pas. Ce n'est cependant pas le cas de toutes les configurations ; celles évoluant dans tout l'espace devront choisir une représentation plus adaptée, telle que les quaternions (voir section B.4).

Par ailleurs, pour éviter les accidents, une sécurité a été ajoutée dans le programme du drone afin de limiter l'amplitude des angles. Ainsi si $|\theta| > 45°$ ou si $|\phi| > 45°$ alors les moteurs se coupent automatiquement car cette configuration n'est pas censée arriver en bon fonctionnement. En effet, si les angles sont aussi importants, c'est que le drone a un problème et va s'écraser, il est donc préférable de couper les moteurs pour limiter les dégâts.

3.1.2 Obtention des couples et forces

Il est ici supposé par simplicité que les pales des rotors sont indéformables et que la force de trainée est négligeable, en considérant par exemple que le drone effectue un vol stationnaire ou quasi-stationnaire. Un modèle plus complet des forces et moments engendrés par l'hélice est par exemple donné dans [BMSP09]. Les auteurs de cet article prennent notamment en compte la flexibilité des rotors et montrent (en écrivant le modèle du quadrirotor) que cette hypothèse est importante si l'on veut modéliser proprement les effets aérodynamiques de trainée. De manière générale, ces effets induisent un couplage entre vitesses latérales et angulaires du drone. Par ailleurs, l'effet de sol n'est pas pris en compte dans notre étude, le drone étant supposé voler à une altitude suffisante.

Les différents couples et forces dus aux moteurs pris en compte dans cette étude sont donc représentés sur la figure 3.1. Ainsi, chacun des quatre moteurs M_i produit une force f_i et un couple τ_i sur l'axe z_b. La poussée totale est donc $u = f_1 + f_2 + f_3 + f_4$. Le couple τ_{x_b} autour de l'axe x_b est obtenu par la différence de forces $f_2 - f_4$ et le couple τ_{y_b} autour de l'axe y_b par la différence de forces $f_1 - f_3$. Enfin le couple τ_{z_b} autour de l'axe z_b est obtenu par la somme des couples produits par les moteurs $\tau_1 + \tau_3 - \tau_2 - \tau_4$; M_1 et M_3 tournant en effet dans le sens positif alors que M_2 et M_4 tournent en sens inverse.

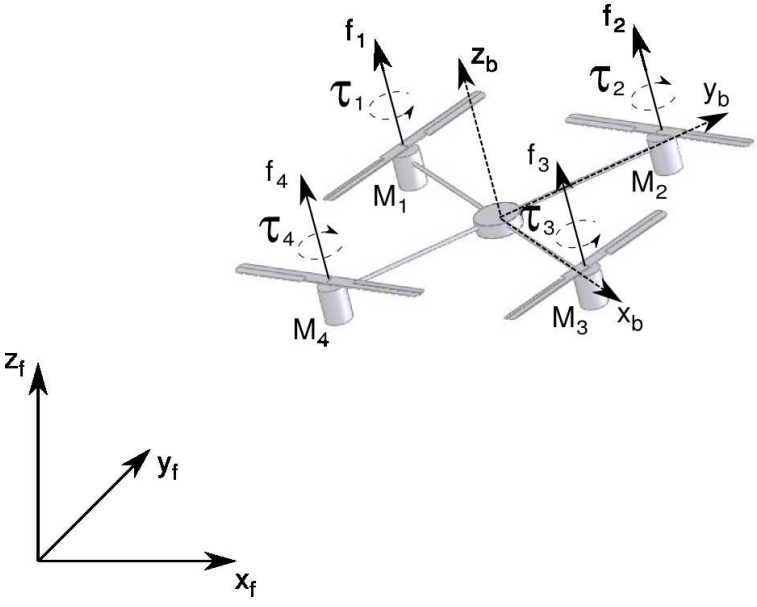


Figure 3.1 – Schéma du quadrirotor.

Dans son propre repère (x_b, y_b, z_b), ces forces et couples s'expriment :

$$F_b = \begin{bmatrix} 0 \\ 0 \\ u \end{bmatrix} \tag{3.4a}$$

$$\tau_b = \begin{bmatrix} \tau_{x_b} \\ \tau_{y_b} \\ \tau_{z_b} \end{bmatrix} = \begin{bmatrix} l(f_2 - f_4) \\ l(f_1 - f_3) \\ \tau_1 + \tau_3 - \tau_2 - \tau_4 \end{bmatrix} \tag{3.4b}$$

où l représente la longueur d'un bras du drone. Par ailleurs, la force de poussée f_i produite par le moteur M_i est proportionnelle à sa vitesse de rotation ω_i au carré :

$$f_i = k_f {\omega_i}^2 \tag{3.5}$$

et le couple τ_i produit par le moteur M_i est proportionnel à sa vitesse de rotation ω_i au carré :

$$\tau_i = k_\tau {\omega_i}^2 \tag{3.6}$$

les constantes k_f et k_τ étant supposées identiques pour tous les moteurs.

3.1.3 Équations dynamiques

Dans cette section nous obtenons le modèle dynamique du quadrirotor. Celui-ci est soumis à la poussée principale, aux trois couples et à son poids.

Les coordonnées généralisées du quadrirotor sont :

$$q = (x, y, z, \phi, \theta, \psi) \tag{3.7}$$

où $\xi = [x, y, z]^T$ représente la position du centre de gravité du quadrirotor par rapport au repère fixe, et $\eta = [\phi, \theta, \psi]^T$ sont ses angles d'Euler.

On définit le lagrangien par :

$$L(q, \dot{q}) = T_{trans} + T_{rot} - U \tag{3.8}$$

où T_{trans}, T_{rot} et U sont respectivement l'énergie cinétique de translation, l'énergie cinétique de rotation et l'énergie potentielle de pesanteur ; leur expression est :

$$T_{trans} = \frac{m}{2} \dot{\xi}^T \dot{\xi} \tag{3.9a}$$

$$T_{rot} = \frac{1}{2} \Omega_b^T I_b \Omega_b \tag{3.9b}$$

$$U = mgz \tag{3.9c}$$

où m est la masse du drone, g est la norme du vecteur gravité, Ω_b est le vecteur de la vitesse angulaire et I_b la matrice d'inertie. Notons que Ω_b et I_b sont exprimés dans le repère du drone. Le vecteur rotation peut aussi s'écrire :

$$\Omega_b = R_{bf}\Omega_f = R_{bf}\dot{\eta} \tag{3.10}$$

d'où :

$$T_{rot} = \frac{1}{2}\dot{\eta}^T R_{fb} I_b R_{bf} \dot{\eta} \tag{3.11a}$$

$$= \frac{1}{2}\dot{\eta}^T J \dot{\eta} \tag{3.11b}$$

avec :

$$J = R_{fb} I_b R_{bf} \tag{3.12}$$

Le modèle dynamique est alors obtenu avec l'équation d'Euler-Lagrange :

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \begin{bmatrix} F_f \\ \tau_b \end{bmatrix} \tag{3.13}$$

Le lagrangien n'ayant pas de couplage entre $\dot{\xi}$ et $\dot{\eta}$ (voir (3.8), (3.9) et (3.11)), l'équation d'Euler-Lagrange peut alors être divisée en une partie translationnelle et une partie rotationelle. L'équation du mouvement de translation est donnée par :

$$\frac{d}{dt}\frac{\partial L_{trans}}{\partial \dot{\xi}} - \frac{\partial L_{trans}}{\partial \xi} = F_f \tag{3.14}$$

soit :

$$m\ddot{\xi} + \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} = R_{fb} F_b \tag{3.15}$$

L'équation du mouvement de rotation est donnée par :

$$\frac{d}{dt}\frac{\partial L_{rot}}{\partial \dot{\eta}} - \frac{\partial L_{rot}}{\partial \eta} = \tau_b \tag{3.16a}$$

$$\frac{d}{dt}\left(\dot{\eta}^T J \frac{\partial \dot{\eta}}{\partial \dot{\eta}}\right) - \frac{1}{2}\frac{\partial}{\partial \eta}\left(\dot{\eta}^T J \dot{\eta}\right) = \tau_b \tag{3.16b}$$

$$J\ddot{\eta} + \dot{J}\dot{\eta} - \frac{1}{2}\frac{\partial}{\partial \eta}\left(\dot{\eta}^T J \dot{\eta}\right) = \tau_b \tag{3.16c}$$

Soit $V(\eta, \dot{\eta})$ le vecteur de Coriolis :

$$V(\eta, \dot{\eta}) = \dot{J}\dot{\eta} - \frac{1}{2}\frac{\partial}{\partial \eta}(\dot{\eta}^T J \dot{\eta}) \tag{3.17}$$

d'où :

$$J\ddot{\eta} = \tau_b - V(\eta, \dot{\eta}) \tag{3.18}$$

Définissons le changement de variable :

$$\tilde{\tau}_b = \begin{bmatrix} \tilde{\tau}_\phi \\ \tilde{\tau}_\theta \\ \tilde{\tau}_\psi \end{bmatrix} = J^{-1}\Big(\tau_b - V(\eta, \dot{\eta})\Big) \tag{3.19}$$

Les équations (3.15) et (3.18) donnent alors la dynamique complète :

$$\begin{aligned}
m\ddot{x} &= u(\cos\psi\cos\phi\sin\theta + \sin\psi\sin\phi) && \text{(3.20a)}\\
m\ddot{y} &= u(\sin\theta\sin\psi\cos\phi - \cos\psi\sin\phi) && \text{(3.20b)}\\
m\ddot{z} &= u\cos\theta\cos\phi - mg && \text{(3.20c)}\\
\ddot{\phi} &= \tilde{\tau}_\phi && \text{(3.20d)}\\
\ddot{\theta} &= \tilde{\tau}_\theta && \text{(3.20e)}\\
\ddot{\psi} &= \tilde{\tau}_\psi && \text{(3.20f)}
\end{aligned}$$

Notons que les détails du calcul de $V(\eta, \dot{\eta})$ peuvent se trouver dans [Gol80] par exemple.

3.1.4 Entrées de commandes

Le quadrirotor ne possède que quatre entrées de commandes, celles de ses moteurs, et le système a six degrés de liberté ; c'est donc un système sous actionné. Une première approche peut consister à contrôler l'altitude et les angles d'Euler du drone ; les entrées de commandes sont alors u_z, $\tilde{\tau}_\phi$, $\tilde{\tau}_\theta$ et $\tilde{\tau}_\psi$ avec :

$$u_z = u\cos\theta\cos\phi - mg \tag{3.21}$$

Nous proposons alors de répartir ces entrées de commande sur chacun des moteurs de la façon suivante :

$$\omega_1 = \sqrt{\frac{u}{4k_f} + \frac{\tau_{y_b}}{2lk_f} + \frac{\tau_{z_b}}{4k_\tau}} \tag{3.22a}$$

$$\omega_3 = \sqrt{\frac{u}{4k_f} - \frac{\tau_{y_b}}{2lk_f} + \frac{\tau_{z_b}}{4k_\tau}} \tag{3.22b}$$

$$\omega_2 = \sqrt{\frac{u}{4k_f} + \frac{\tau_{x_b}}{2lk_f} - \frac{\tau_{z_b}}{4k_\tau}} \tag{3.22c}$$

$$\omega_4 = \sqrt{\frac{u}{4k_f} - \frac{\tau_{x_b}}{2lk_f} - \frac{\tau_{z_b}}{4k_\tau}} \tag{3.22d}$$

avec, d'après (3.21) et (3.19) :

$$u = \frac{u_z + mg}{\cos\theta\cos\phi} \tag{3.23a}$$

$$\begin{bmatrix} \tau_{x_b} \\ \tau_{y_b} \\ \tau_{z_b} \end{bmatrix} = J \begin{bmatrix} \tilde{\tau}_\phi \\ \tilde{\tau}_\theta \\ \tilde{\tau}_\psi \end{bmatrix} + V(\eta, \dot{\eta}) \tag{3.23b}$$

Ainsi, en utilisant les équations (3.22), on obtient :

$$\sum_{i=i}^{4} f_i = k_f \sum_{i=i}^{4} \omega_i^2 = u \tag{3.24a}$$

$$l(f_2 - f_4) = lk_f(\omega_2^2 - \omega_4^2) = \tau_{x_b} \tag{3.24b}$$

$$l(f_1 - f_3) = lk_f(\omega_1^2 - \omega_3^2) = \tau_{y_b} \tag{3.24c}$$

$$\tau_1 + \tau_3 - \tau_2 - \tau_4 = k_\tau(\omega_1^2 + \omega_3^2 - \omega_2^2 - \omega_4^2) = \tau_{z_b} \tag{3.24d}$$

Les résultats des équations (3.24) montrent que la répartition proposée permet bien d'obtenir les couples et forces voulues.

3.1.5 Modèle simplifié

Le modèle dynamique (3.20) obtenu précédemment est non linéaire et contient de nombreux termes de couplage entre les différents états. Cependant, le quadrirotor est souvent étudié dans le cas du vol stationnaire ou quasi stationnaire (déplacements latéraux très lents par faibles changements d'orientations). Dans ce cas, le modèle (3.20) peut être linéarisé autour de la position d'équilibre ($\phi = 0, \theta = 0, \psi = 0$). Le vecteur de Coriolis $V(\eta, \dot{\eta})$ est alors nul et $J = I_b$. De

plus la matrice d'inertie I_b peut être considérée comme diagonale :

$$I_b = \begin{bmatrix} I_{b_{xx}} & 0 & 0 \\ 0 & I_{b_{yy}} & 0 \\ 0 & 0 & I_{b_{zz}} \end{bmatrix} \tag{3.25}$$

Il s'ensuit le modèle simplifié suivant :

$$\begin{aligned} m\ddot{x} &= u(\theta + \psi\phi) && \text{(3.26a)} \\ m\ddot{y} &= u(\theta\psi - \phi) && \text{(3.26b)} \\ m\ddot{z} &= u - mg && \text{(3.26c)} \\ I_{b_{xx}}\ddot{\phi} &= \tau_{x_b} && \text{(3.26d)} \\ I_{b_{yy}}\ddot{\theta} &= \tau_{y_b} && \text{(3.26e)} \\ I_{b_{zz}}\ddot{\psi} &= \tau_{z_b} && \text{(3.26f)} \end{aligned}$$

En pratique, l'angle ψ est facile à stabiliser et peut souvent être considéré comme nul. Cela permet de découpler les dynamiques latérales x et y :

$$\begin{aligned} m\ddot{x} &= u\theta && \text{(3.27a)} \\ m\ddot{y} &= -u\phi && \text{(3.27b)} \\ m\ddot{z} &= u - mg && \text{(3.27c)} \\ I_{b_{xx}}\ddot{\phi} &= \tau_{x_b} && \text{(3.27d)} \\ I_{b_{yy}}\ddot{\theta} &= \tau_{y_b} && \text{(3.27e)} \\ I_{b_{zz}}\ddot{\psi} &= \tau_{z_b} && \text{(3.27f)} \end{aligned}$$

Le modèle obtenu est alors complètement découplé et linéaire. Si l'on cherche à contrôler l'altitude et les angles d'Euler, les entrées de commandes sont alors u_z, τ_{x_b}, τ_{y_b} et τ_{z_b} avec :

$$u_z = u - mg \tag{3.28}$$

La répartition des entrées de commande sur chacun des moteurs se faisant toujours selon les équations (3.22).

3.2 Structure mécanique

La structure mécanique est très importante lors de la réalisation du quadrirotor. C'est en effet celle-ci qui va positionner les quatre moteurs. Dans l'idéal ceux-ci

doivent être parfaitement coaxiaux, placés à 90° et à égale distance du centre géométrique ; les hélices quant à elles doivent donc être dans le même plan. Par ailleurs la structure doit être suffisamment rigide pour ne pas trop se déformer lorsque les moteurs sont en fonctionnement. Cependant, la rigidité ne doit pas se faire au détriment du poids qui est aussi un critère important lors de la conception du drone ; le poids influant directement sur l'autonomie du drone. Dans un premier temps, nous avons donc travaillé avec les structures du commerce, principalement du *DraganFly*. Cette société commercialise des drones de type quadrirotor mais vend aussi séparément les éléments de sa structure. Celle-ci est composée de quatre tubes en carbone reliés au centre du drone par une croix en plastique (cf figure 3.2). Originellement prévue pour des moteurs à courant continu, nous l'avons équipée de moteurs *brushless*, plus adaptés (voir section 3.3). Cependant, bien que légère, la structure n'est pas assez rigide et a tendance à se déformer en vol. En effet elle n'a pas été prévue pour les nouveaux moteurs, plus puissants que ceux d'origine, et pour le surpoids engendré par notre système embarqué. Le constructeur a cependant amélioré cette structure en ajoutant quatre autres tubes de carbone reliant les moteurs deux à deux (cf figure 3.3). Cette structure a néanmoins le désavantage d'être relativement onéreuse. Enfin, si elle se casse suite à une chute il n'est pas évident de la réparer et il faut parfois recommander des pièces. La dernière version du *X4* (voir figure 2.10), n'est pas encore disponible à la vente, mais sa structure semble plus rigide et prévue pour des moteurs *brushless*. Cependant il n'est pas encore sûr que cette structure puisse s'acheter seule.

Figure 3.2 – Structure du *DraganFly*. Source : site *DraganFly* [itc].

Figure 3.3 – Structure rigidifiée du *DraganFly*. Source : site *DraganFly* [itc].

Depuis peu de nouveaux drones "commerciaux" sont apparus, tels que le *Xufo* et le *Mikrokopter*. Les deux fabricants proposent eux aussi d'acheter le drone en

pièces détachées et vendent donc la structure seule. Le *Xufo* (cf figure 3.4) possède une structure en carbone. Contrairement au *DraganFly*, il s'agit ici de plaques et non de tubes. Cela lui confère une meilleure rigidité, au détriment du poids même si l'ensemble reste suffisamment léger car la structure du *Xufo* est plus petite ; l'écartement entre moteurs étant de 34 cm contre 40 cm pour les autres structures. Le *Mikrokopter* quant à lui possède une structure en aluminium. Il s'agit de quatre profilés de section carrée, reliés entre eux par deux plaques en plastique (cf figure 3.5), prenant les profilés en sandwich. La rigidité de l'ensemble est bonne, mais le tout reste un peu plus lourd qu'une structure en carbone. Cette structure est cependant moins onéreuse.

Figure 3.4 – Structure du *Xufo*. Source : manuel de l'utilisateur du *Xufo* [itv].

Figure 3.5 – Structure du *Mikrokopter*. Source : site internet *Mikrokopter* [ith].

Afin d'obtenir une structure adéquate, nous effectuons en général des transformations sur les structures décrites ci dessus. C'est le cas par exemple de la structure représentée figure 3.6, sur laquelle il a été rajouté des profilés d'aluminium pour améliorer la rigidité. Le carbone étant dur à travailler et à assembler, nous avons aussi réalisé entièrement une structure en aluminium. Celle-ci est inspirée du *Mikrokopter*, mais par souci de rigidité elle est constituée de seulement deux profilés en aluminium (et non de quatre). Les profilés ont alors chacun une rainure en leur milieu pour permettre leur assemblage, cf figure 3.7.

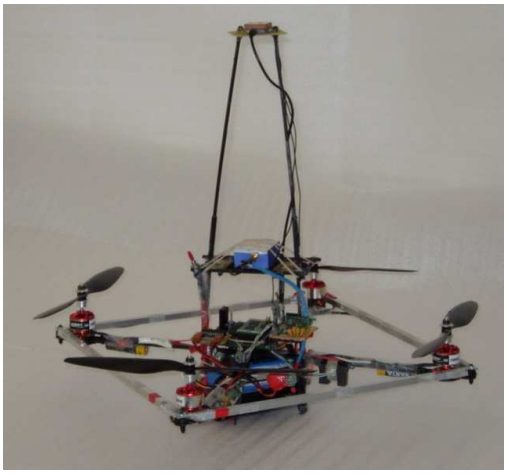


Figure 3.6 – Structure modifiée du *X4*.

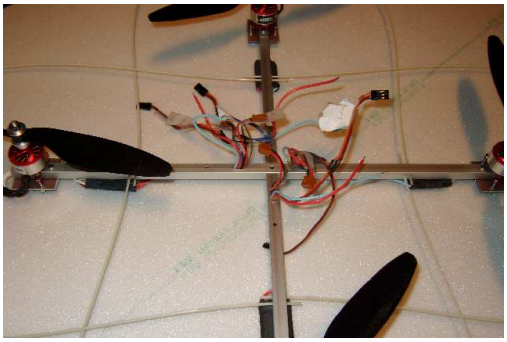


Figure 3.7 – Nouvelle structure du quadrirotor.

3.3 Motorisation

3.3.1 Types de moteurs

Il existe différents types de moteurs : électriques, thermiques, pneumatiques, hydrauliques... Seuls les moteurs électriques et thermiques sont utilisés pour la propulsion d'engins volants de type drone. Dans le cadre de cette thèse nous ne nous intéresserons qu'aux moteurs électriques. En effet les moteurs thermiques sont bien trop encombrants et trop lourds pour les drones que nous utilisons. Ils sont aussi plus puissants et donc plus dangereux. Il sera donc seulement décrit ici les différents types de moteurs électriques.

Il existe en effet différentes technologies de moteurs électriques. Ceux-ci peuvent être regroupés en trois grandes catégories : les moteurs pas à pas, les moteurs à courant continu et les moteurs à courant alternatif.

Les moteurs pas à pas sont utilisés dans des applications nécessitant un contrôle de vitesse ou de position. Cependant ce contrôle se fait en boucle ouverte. En effet, des impulsions sont envoyées au moteur afin que celui-ci tourne d'un certain nombre de pas mais rien ne garantit que le moteur soit arrivé à la position demandée. Le nombre de pas par tour permet alors de caractériser la résolution du moteur. L'une des applications principale de ce type de moteur est l'imprimante. Le pas à pas est donc rarement utilisé dans le monde du modélisme, où le contrôle

de position n'est pas une nécessité. Par ailleurs la complexité du moteur et de son électronique associée peut engendrer un surcoût inutile.

Les moteurs à courant continu quant à eux sont très simples d'utilisation, il suffit de faire varier leur tension d'alimentation pour changer la vitesse. Un moteur à courant continu peut aussi bien fonctionner en mode moteur qu'en mode générateur, afin de produire de l'énergie, mais cette dernière caractéristique n'est pas utile dans notre application. Il est constitué d'un stator et d'un rotor. Le stator crée un champ magnétique fixe grâce à un bobinage ou à des aimants permanents. Le rotor est constitué d'au moins deux bobinages, parcourus par un courant changeant de sens grâce au système balais/collecteurs. L'avantage de ce type de moteur est la facilité à changer sa vitesse, son couple ou son sens de rotation. Par contre, le système de balais représente de nombreux inconvénients. En effet la liaison balais/collecteurs s'effectue par frottements, donc plus la vitesse de rotation est élevée plus il y a des frottements et plus les balais s'usent. Par ailleurs, les moteurs de petite taille (tels que ceux utilisés dans le drone) ne sont pas prévus pour pouvoir changer les balais endommagés.

Les moteurs à courant alternatif se décomposent en deux familles, les moteurs synchrones et les moteurs asynchrones. Un moteur synchrone a une vitesse de rotation proportionnelle à la fréquence du courant qui le traverse (d'où son nom). En mode générateur, le courant généré est proportionnel à la vitesse de rotation. Ces moteurs sont généralement triphasés, et de forte puissance. On en trouve en effet dans le *TGV*, dans les centrales électriques... Il faut noter que le moteur sans balais (dit *brushless*) est un type de moteur synchrone ; une description plus précise de ce moteur sera faite au paragraphe suivant. Le moteur asynchrone quant à lui a donc une vitesse de rotation pouvant être différente de la fréquence d'alimentation. En effet, un champ tournant est créé dans le stator à la vitesse de synchronisme ; le rotor est alors mis en rotation. Cependant celui-ci ne peut pas tourner à la vitesse de synchronisme, il apparait alors un glissement (en général inférieur à 10 %). Plus ce glissement est grand, plus le moteur a un mauvais rendement.

Les moteurs *brushless* sont ainsi des moteurs à courant alternatif synchrone, possédant trois phases. Cependant, ils sont généralement associés à une électronique de commande (appelée *driver*), les rendant semblables à un moteur courant continu pour l'utilisateur. Leur avantage est de ne pas avoir de balais ; cette pièce d'usure étant éliminée, le rendement est alors amélioré car les frottements ne proviennent plus que des roulements. Le système balais/collecteurs permettant de faire tourner le champ magnétique est ici remplacé par une électronique de commande, le *driver*. Ce dernier permet de toujours maintenir le champ magnétique du stator (composé de bobines) orthogonal à celui du rotor (composé d'aimants permanents). Pour cela il faut cependant connaitre la position du rotor par rapport aux bobines

du stator. Ces moteurs peuvent donc être équipés d'un capteur à effet *Hall* ou d'un codeur incrémental pour détecter la position. Une autre possibilité est de la détecter de façon *sensorless* (sans capteur), c'est ce que font la plus grande partie des *drivers* pour moteurs *brushless*, économisant ainsi le prix d'un capteur. Le principe du détecteur de position *sensorless* est assez simple : étant donné que pour faire tourner le moteur seules deux de ses phases sont simultanément alimentées, il est possible de mesurer la tension induite sur la dernière phase par rapport au neutre. Cette tension s'annule quand le rotor est exactement en face de cette phase (voir figure 3.8).

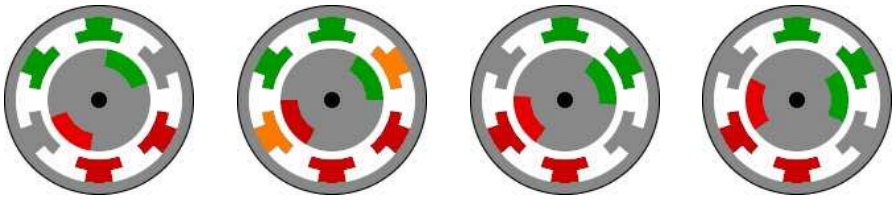


Figure 3.8 – Principe de rotation du moteur *brushless*. Le stator est à l'extérieur et le rotor à l'intérieur. Deux phases sont simultanément alimentées (rouge et vert), la troisième étant mesurée (gris). Lorsque le moteur est à la seconde position, la tension induite dans la troisième phase (orange) par rapport au neutre est nulle ; le *driver* commute alors les phases. Source : [itl].

Ce sont ces dernier moteurs, les *brushless*, qui ont été retenus pour la plateforme expérimentale. Ces moteurs sont en effet de plus en plus utilisés en modélisme et se trouvent facilement dans le commerce. Leur électronique de commande s'achète elle aussi, simplifiant ainsi l'électronique de puissance à concevoir sur le drone. Ces moteurs sont aussi plus puissants et plus réactifs. Enfin il est possible de mesurer la vitesse de rotation du moteur sans achat d'un capteur supplémentaire, ce qui est intéressant pour asservir les moteurs en vitesse.

3.3.2 *Driver*

Les moteurs retenus nécessitent donc une électronique de contrôle (*driver*), afin d'alimenter correctement ses phases. Les *drivers* sont commandés en *MLI* (Modulation de Largeur d'Impulsion) ou *PWM* (*Pulse Width Modulation*) en anglais. Ce signal de commande est semblable à celui utilisé par les radiocommandes de modélisme (voir section 3.6) puisque les *drivers* sont faits pour être branchés sur un récepteur radio. En utilisant le standard des récepteurs radios, les *drivers* sont commandés avec un signal de 50 Hz. Cependant ils acceptent en général des fréquences plus élevées. Les *drivers* utilisés sur la plateforme (*Phœnix 25* de *Castle Creation*)

acceptent un signal allant jusqu'à 500 Hz. En pratique, le signal de commande a été fixé à 200 Hz ; ce qui est largement suffisant par rapport à la dynamique du moteur. D'un point de vue utilisateur, les *drivers* permettent de commander le moteur en puissance. Ainsi à *PWM* donné, la vitesse de rotation du moteur dépendra de la charge sur son axe. En effet, une expérience simple à réaliser consiste à donner une consigne fixe de *PWM* au moteur (via la radiocommande par exemple), puis à perturber la vitesse de rotation du moteur. En serrant le rotor du moteur entre ses doigts par exemple, on constate que sa vitesse va diminuer et que le *driver* ne tentera pas de corriger ce changement.

3.3.3 Asservissement de vitesse

Comme il a été vu précédemment, les *drivers* mesurent la position du rotor du moteur afin d'ajuster le flux magnétique tournant et d'éviter les décrochements. Cependant, le *driver* n'est pas conçu pour délivrer cette information. Il a donc été recréé un montage permettant la mesure *sensorless* de la position du moteur sur la carte embarquée au drone, à partir des phases du moteur. Étant donné que notre microcontrôleur n'a pas à commuter les phases, le circuit de mesure compare la tension d'une seule phase par rapport au neutre. Si la tension est positive il envoie un signal logique 1 au microcontrôleur, si la tension est négative il envoie un signal logique 0. Le temps entre deux fronts montants successifs du signal équivaut alors au temps mis par le rotor pour faire un tour. La mesure de ce temps par le microcontrôleur permet donc d'estimer la vitesse de rotation. En pratique, cette mesure se fait grâce à une entrée de type *input capture*. Celle-ci est associée à un compteur de temps (*timer*) ; lorsqu'elle détecte un front montant, la valeur du *timer* est gardée en mémoire et une interruption est générée.

Cela permet donc de contrôler la vitesse de rotation de chacun des moteurs et de l'asservir à la vitesse désirée. Cette étape améliore alors la stabilisation du drone. En effet en observant la figure 3.9, il apparait que la vitesse de rotation du moteur n'est pas proportionnelle à la consigne *PWM*. De plus, des essais ont montré qu'à *PWM* fixé, la vitesse dépend aussi du niveau de charge de la batterie. Enfin, la figure 3.10 montre qu'à consigne *PWM* donnée, chaque moteur (ou chaque couple moteur/*driver*) a une vitesse de rotation différente Un contrôle en boucle ouverte basé sur le *PWM* n'est donc pas du tout efficace.

La réponse fréquentielle du moteur a ensuite été analysée dans le but d'identifier le modèle du moteur. Pour cela, la courbe 3.9 a d'abord été utilisée afin d'avoir la relation entre le *PWM* et la vitesse réelle en régime permanent. Cela permettra par la suite de commander le moteur directement avec une vitesse de consigne (v_c), et d'utiliser un modèle du moteur ayant un gain statique unitaire. Notons que la

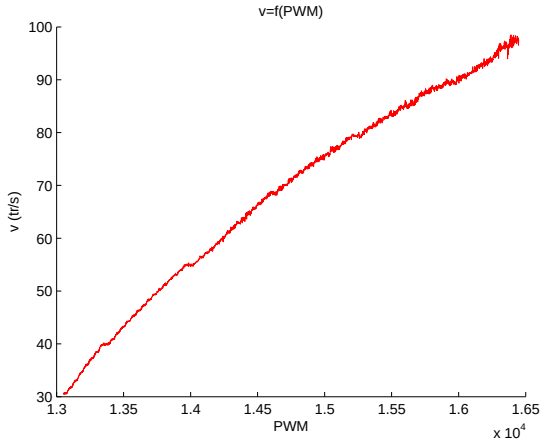


Figure 3.9 – Vitesse de rotation du moteur en fonction du PWM. Lors de cet essai, la consigne PWM a été incrémentée suffisamment lentement pour que le moteur atteigne à chaque fois sa vitesse nominale.

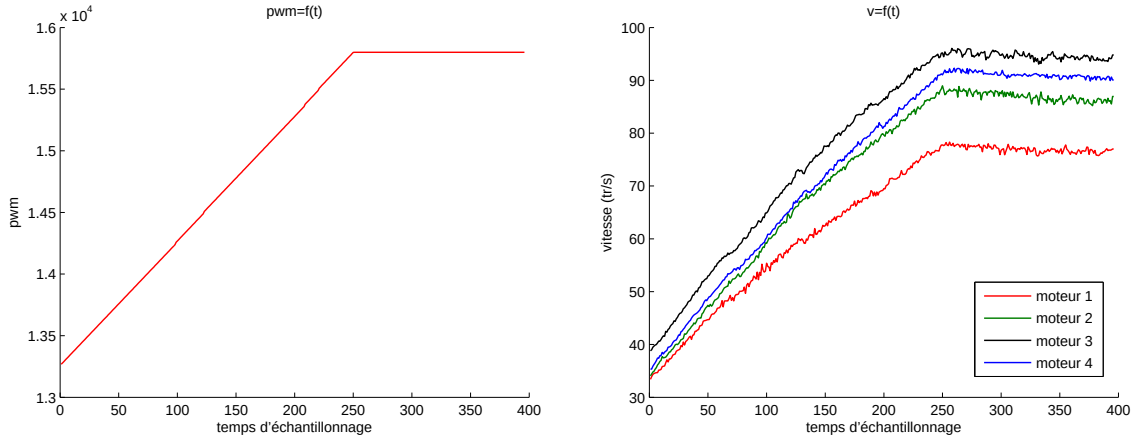


Figure 3.10 – Consigne PWM et vitesses de rotations des quatre moteurs correspondantes.

courbe 3.9 a été obtenue avec une alimentation de laboratoire afin de garder une tension constante pour tous les essais et d'éviter les problèmes liés à la charge de la batterie vus précédemment. Ainsi, à l'aide de *Matlab*, un polynôme $P(v_c)$ approximant au mieux (au sens des moindres carrés) le PWM a été cherché. La figure 3.11, montre les différences obtenues pour des ordres de 1, 2, et 3. Un ordre 3 a été retenu car passé cet ordre, les différences ne sont plus significatives :

$$PWM = 1,6005.10^{-3}{v_c}^3 + 2,0939.10^{-2}{v_c}^2 - 27.151v_c + 1,2157.10^4 \qquad (3.29)$$

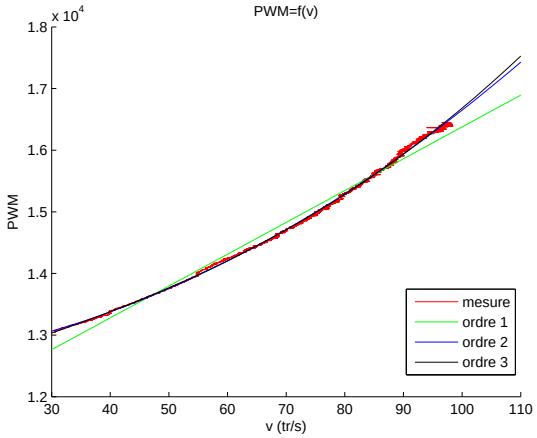


Figure 3.11 – Comparaison des polynômes approchant la mesure $PWM = f(v_c)$.

La figure 3.12 montre ainsi deux essais fréquentiels. La vitesse commandée est une sinusoïde d'amplitude 30 tr/s et de fréquence $0,67$ ou 4 Hz. Ces graphes montrent le déphasage et l'atténuation de la réponse lorsque la fréquence augmente. La table 3.1 résume le gain et le déphasage du moteur à différentes fréquences. Notons cependant que les mesures à hautes fréquences sont assez difficiles à réaliser car la période du PWM commandant le moteur est fixée à 5 ms (voir section 3.3.2), et il s'agit aussi de la période à laquelle sont récupérées les mesures. Le diagramme de Bode correspondant est donné à titre indicatif à la figure 3.13 ; les mesures à hautes fréquences n'étant pas précises. Ce diagramme laisse cependant supposer un modèle d'ordre 2.

Une loi de commande de type PI a alors été intégrée afin d'asservir la vitesse de rotation du moteur. Cependant, cette loi prend directement en entrée une vitesse de consigne en tr/s et donne un PWM en sortie. Le polynôme (3.29) n'est donc plus utilisé. Cela permet d'économiser le calcul de 4 polynômes de ce type dans le microcontrôleur, sachant que le terme intégrale compensera l'erreur. La figure 3.14 montre la réponse du moteur en boucle fermée, pour une vitesse de consigne à la même fréquence que les essais en boucle ouverte réalisés précédemment (figure 3.12). La boucle de commande est donc performante à basse fréquence ; par contre si la fréquence devient trop élevée, le correcteur PI n'est plus suffisant et il apparait un déphasage. Le terme intégrale a par ailleurs aussi tendance à faire dépasser la consigne dans ce cas. Cependant ces résultats sont satisfaisants pour notre application et améliorent grandement la réponse des moteurs. En effet l'amplitude de 30 tr/s des essais est beaucoup plus importante que les amplitude réelles en fonctionnement du drone.

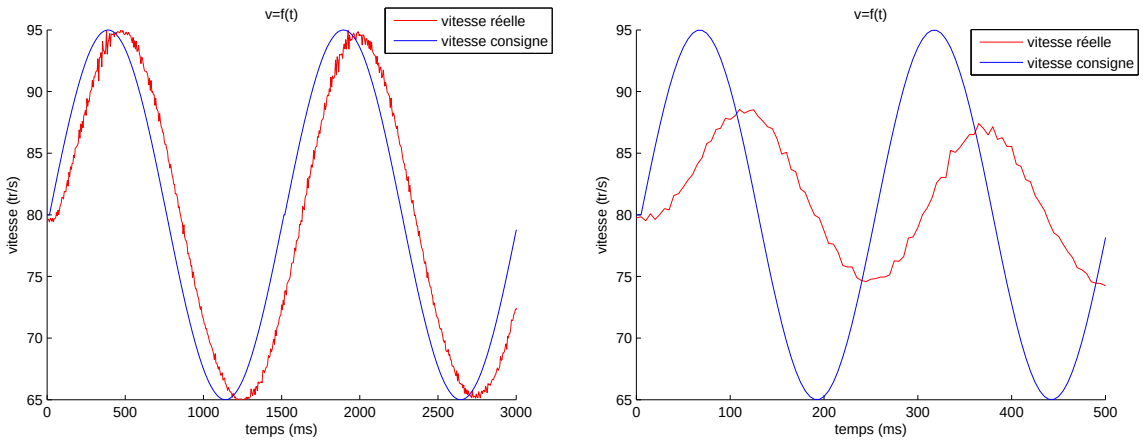


Figure 3.12 – Vitesses de rotations réelles (en boucle ouverte) et de consigne en fonction du temps. Les oscillations de la figure de gauche sont à 0,67 Hz, et à 4 Hz pour la figure de droite.

Fréquence (Hz)	Gain (dB)	Déphasage (degré)
0.5000	0	-17.5500
0.5556	-0.0192	-19.5000
0.6231	-0.2018	-21.8692
0.7117	-0.4372	-25.6228
0.8299	-0.8221	-31.3693
1.1050	-1.1559	-37.7901
1.6529	-2.5042	-52.0661
2.4691	-4.2736	-68.8889
3.2787	-6.1973	-76.7213
4.0000	-7.8853	-75.6000
6.6667	-11.6198	-108.0000
10.0000	-15.2675	-108.0000
20.0000	-20.7354	-108.0000
25.0000	-25.1328	-180.0000

Table 3.1 – Gain et déphasage du moteur en fonction de la fréquence.

Malgré tout, la solution choisie n'est pas la plus efficiente car le microcontrôleur principal mesure la vitesse des quatre moteurs alors que cela est déjà fait

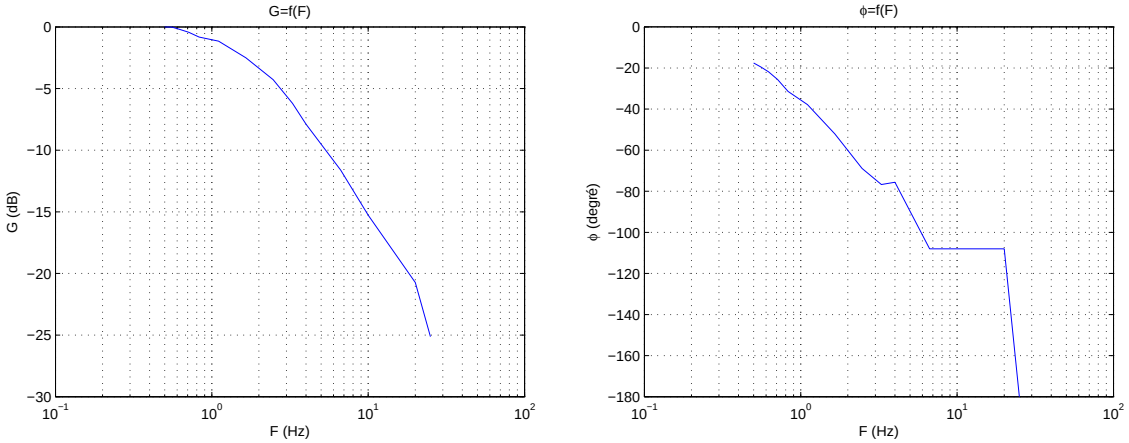


Figure 3.13 – Diagramme de Bode de la réponse du moteur.

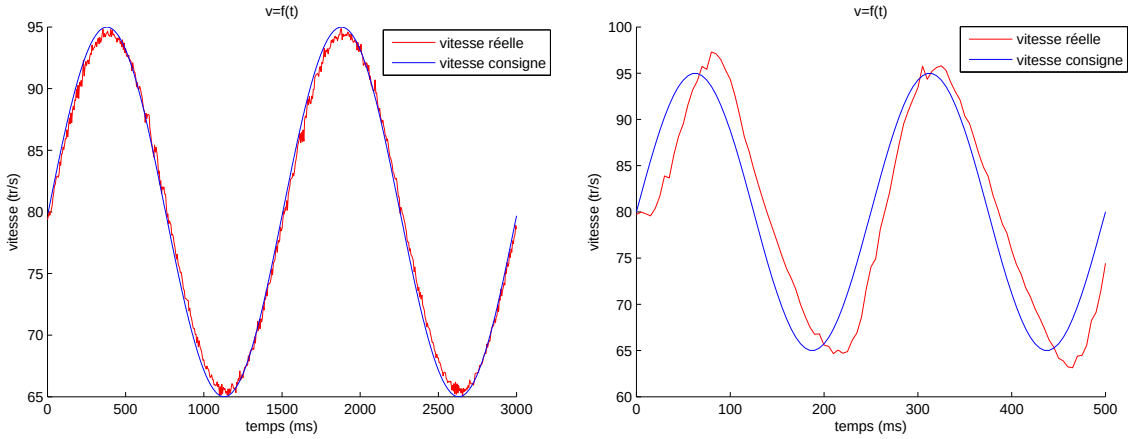


Figure 3.14 – Vitesses de rotations réelles (en boucle fermée) et de consigne en fonction du temps. Les oscillations de la figure de gauche sont à 0,67 Hz, et à 4 Hz pour la figure de droite.

dans chacun des *drivers*, et il rajoute une couche d'asservissement en plus de celle déjà effectuée dans le *driver*. La meilleure solution serait d'avoir des *drivers* contrôlables en vitesse, or à notre connaissance il n'en existe actuellement pas dans le commerce. De plus, ils sont en général conçus et vendus indépendamment des moteurs. Le *driver* utilisé n'est donc pas optimisé pour le moteur du drone ; ses performances et sa consommation ne sont donc pas optimales. Les réglages possibles d'un *driver* étant d'ailleurs très limités. Cependant, il est envisageable d'en développer un répondant au cahier des charges. En effet, le *driver* "idéal" devrait être capable d'asservir la vitesse de rotation du moteur en fonction d'une consigne

de vitesse reçue du microcontrôleur principal via un bus de communication (*I2C*, *Inter Integrated Circuit*, ou *SPI*, *Serial Peripheral Interface*, par exemple), et les gains de sa loi de commande devraient pouvoir être paramétrables (via le même bus de communication). Or de nombreux fabricants de microcontrôleurs ont des notes d'applications décrivant comment piloter un moteur *brushless*, et bien souvent fournissent le code associé, ce qui devrait rendre la tâche plus aisée. Il est aussi possible d'acheter chez certains fabricants des kits comprenant une carte de développement, un programmateur pour le microcontrôleur et un moteur *brushless* afin de s'initier au contrôle du moteur. La création d'un tel *driver* fait donc partie des travaux futurs de cette thèse.

3.4 Énergie

Étant donné que les moteurs retenus sont électriques, cette section portera sur les batteries électriques. Il existe un grand nombre de technologies de batteries différentes. Citons par exemple :

- **les batteries au plomb** (Pb), très répandues dans les automobiles ou les motos.
- **les batteries nickel-cadmium** ($NiCd$) **et nickel-métal-hydrure** ($Ni\text{-}MH$), utilisées pour les piles rechargeables.
- **les batteries lithium-ion** ($LiIon$), utilisées dans les batteries plus récentes telles que celles des téléphones portables, des ordinateurs portables, des lecteurs *MP3*, etc.
- **les batteries lithium-polymère** ($LiPO$), répandues dans le modélisme.
- **les batteries lithium-fer-phosphate** ($LiFePO_4$), apparues en 2007 elle se sont rapidement popularisées dans le domaine du modélisme.

Une batterie est caractérisée par sa tension et sa charge. La tension est exprimée en volts et correspond à la tension nominale de la batterie. La charge (C) correspond à la capacité de la batterie, elle est exprimée en mAh ; c'est la quantité moyenne d'électricité qu'elle peut débiter en une heure. Une batterie est en fait un pack, composé de plusieurs éléments (ou *cells* en anglais). Chaque élément a une tension nominale ; celle-ci dépendant de la technologie. Par exemple un élément $NiCd$ a une tension nominale de 1.2 V, un élément Pb 2.1 V et un élément $LiPO$ 3.7 V. La tension de la batterie est alors égale à la somme de la tension de chacun de ses éléments (s'ils sont montés en série), sachant que les éléments peuvent être chargés à une tension légèrement supérieure à la tension nominale. Il est à noter que chaque technologie possède son propre moyen de charge (tension, ampérage) à respecter.

Seules les batteries ayant un rapport puissance/poids important nous intéressent ici ; les batteries de type Pb, $NiCd$ et $NiMH$ ne seront donc pas traitées. En effet, les batteries au lithium proposent des capacités équivalentes aux batteries traditionnelles mais pour le tiers du poids. Le paragraphe suivant présente donc les batteries de type $LiPo$ utilisées sur le drone car répandues dans le monde du modélisme.

Le principal avantage des batteries $LiPo$ est donc leur faible poids, ce qui les rend parfaitement adaptées à notre application, mais aussi parce qu'elles ne présentent pas d'effet mémoire. Cependant, leurs principaux inconvénients se situent au niveau de leur capacité de charge, de leur fragilité et de leur prix. En effet, le taux de charge de ces batteries est de $1C$; donc il faut toujours une heure pour la charger. Le taux de décharge maximum est quant à lui limité à 15 fois la capacité ($15C$) pour des batteries traditionnelles. Cependant, pour une batterie de 2400 mAh, cela équivaut à 36 A (2400 mA $*15$) en pointe et pendant 4 minutes (60/15) ce qui est rarement le cas sur le drone. En général, une batterie de 2400 mAh permet un temps de vol d'au moins 6 minutes, ce qui laisse supposer une consommation d'environ 24 A (soit une décharge à $10C$). Les batteries $LiPo$ sont fragiles mécaniquement et électriquement. En effet un choc sur la batterie peut provoquer un court circuit interne, et la batterie peut alors prendre feu (même plus de 10 minutes après le choc). La batterie peut aussi prendre feu après une mauvaise charge ; il faut donc veiller à toujours la charger au bon voltage (4.2 V par élément) et à la bonne capacité. Il est conseillé d'effectuer la charge sur une surface à l'épreuve du feu et d'éloigner toute matière inflammable. Les chargeurs sont d'ailleurs souvent vendus avec une sonde de température à placer entre deux éléments afin d'éviter toute surchauffe du pack. De plus, lors de la charge d'une batterie, il est fortement conseillé d'utiliser un équilibreur de charge. Celui-ci permet de charger les éléments indépendamment en contrôlant leur tension. En effet, si les éléments sont chargés en série, la tension ne sera régulée que sur l'ensemble de la batterie et il est possible d'obtenir par exemple un élément chargé à 4.1 V et un autre à 4.3 V (ce qui risque de l'abimer) tout en ayant une tension moyenne correcte de 4.2 V. Il faut aussi veiller à ne pas trop décharger une batterie $LiPo$; si un élément tombe en dessous de 2.5 V, il devient alors impossible de le charger et donc inutilisable (rendant alors souvent le pack entier inutilisable). Les contrôleurs pour moteurs *brushless* permettent d'ailleurs de couper le moteur si la tension d'alimentation devient trop faible, pour éviter de décharger en dessous du seuil critique.

Les batteries $LiFePO_4$ ne sont apparues que très récemment (2007) dans le commerce, leur découverte datant de 1996 (voir [PNG97]). Les éléments se caractérisent par une tension nominale de 3.3 V et une tension de charge de 3.6 V.

Leur principal avantage est la capacité de charge qui peut être de $2C$ et même plus suivant les modèles, et la capacité de décharge pouvant atteindre $50C$. La capacité de charge apporte donc beaucoup de confort car le temps de charge des $LiPO$ est très long par rapport au temps de décharge et donc au temps de vol du drone. La capacité de décharge est aussi beaucoup plus importante, cependant comme vu précédemment la décharge de $15C$ du $LiPO$ est déjà suffisante. L'inconvénient de cette technologie est que son rapport puissance/poids reste en dessous des $LiPO$: de 130 à 200 Wh/kg pour les $LiPO$ et de 90 à 110 Wh/kg pour les $LiFePO_4$.

3.5 Capteurs

3.5.1 Centrale inertielle

La centrale inertielle est sans aucun doute l'élément le plus critique du drone. La stabilisation dépend énormément de sa qualité. Elle fournit en effet les informations inertielles à la loi de commande : orientation (angles d'Euler, matrice de rotation ou quaternions), vitesse de rotation. Une centrale inertielle est principalement composée d'accéléromètres, de gyromètres, de magnétomètres et d'un capteur de température (pour compenser les mesures en fonction de la température). Les paragraphes suivants présentent rapidement chacun de ces capteurs puis quelques méthodes de fusion de données afin d'estimer l'attitude. Enfin, la centrale inertielle utilisée sur la plateforme expérimentale est présentée.

Accéléromètre

Un accéléromètre mesure la grandeur $\overrightarrow{A}$ d'un corps auquel il est attaché :

$$\overrightarrow{A} = \overrightarrow{\gamma} - \overrightarrow{g} \tag{3.30}$$

où $\overrightarrow{\gamma}$ est l'accélération absolue et $\overrightarrow{g}$ le champ de pesanteur. La grandeur $\overrightarrow{A}$ peut être appelée ([Rad00]) :

- lecture (ou mesure) accélérométrique,
- accélération non gravitationnelle,
- effort massique non gravitationnel.

En effet, l'accéléromètre peut être schématisé comme un système masse-ressort, voir figure 3.15. Le principe fondamental de la dynamique appliqué à ce système donne :

$$\overrightarrow{F}_r + m\overrightarrow{g} = m\overrightarrow{\gamma} \tag{3.31}$$

où $\overrightarrow{F}_r$ est la force produite par le ressort et m la masse de la partie mobile. D'où :

$$\overrightarrow{F}_r = m\left(\overrightarrow{\gamma} - \overrightarrow{g}\right) = m\overrightarrow{A} \tag{3.32}$$

L'accélération non gravitationnelle est donc proportionnelle à la force de rappel du ressort, elle-même proportionnelle au déplacement de la masse.

La détection du déplacement peut se faire de différentes façons, citons par exemple les accéléromètres à détection piézoélectrique, à détection piézorésistive, à jauge de contrainte/extensométrie (proche du type piézorésistif, dans son principe), à détection capacitive, à détection inductive (ou réluctance variable), à détection optique, à poutre vibrante, à ondes de surface... La sortie de ce capteur peut ensuite être numérique ou analogique. Dans le premier cas, la valeur de l'accélération peut directement être lue par un microcontrôleur (via un bus *SPI* ou *I2C*). Dans le second cas, le capteur génère une tension proportionnelle à la mesure, et il faut utiliser un convertisseur analogique-numérique pour pouvoir l'exploiter.

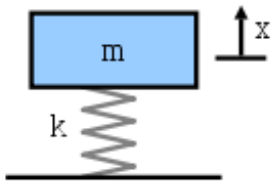


Figure 3.15 – Schéma de principe d'un accéléromètre. Source : *Wikipedia* [itt].

Les accéléromètres de type *MEMS* (*MicroElectroMechanical Systems*) sont de moins en moins coûteux car de plus en plus utilisés du fait de leur taille miniature et de leur faible poids. Ils se retrouvent en effet dans de nombreux objets de la vie quotidienne : appareils photos, téléphones portables, lecteurs *MP3* et *MP4*, manettes de jeu vidéo... Ce sont d'ailleurs des capteurs de type *MEMS* que l'on retrouve dans les centrales inertielles miniatures fréquemment utilisées dans les drones.

Les accéléromètres sont généralement caractérisés par leur plage de mesure, leur bande passante, leur précision, leur sensibilité et leur nombre d'axes. Il est en effet intéressant d'avoir un maximum d'axes (3) sur le même composant, afin d'assurer l'orthogonalité de chacun des axes.

Gyromètre

Le premier gyroscope a été inventé par l'allemand Johann Bohnenberger en 1817. Cependant, il a été rendu populaire par Léon Foucault qui l'améliora et s'en

servi pour démontrer la rotation de la Terre en 1852 (voir [Fou52]). C'est d'ailleurs Foucault qui donna le nom de gyroscope à cet instrument (voir figure 3.16). Celui-ci est composé d'un disque en rotation très rapide dont l'axe peut prendre n'importe quelle orientation grâce au système de cardans (*gimbals*). La conservation du moment angulaire impose que le disque garde une orientation quasi fixe, quelque soit le mouvement de la plateforme sur laquelle il est monté. Le dispositif permet donc de mesurer l'orientation du corps auquel il est attaché. Les centrales inertielles dites à plateforme utilisent ce type de gyroscope pour maintenir l'orientation des accéléromètres fixe. Bien que précis, les gyroscopes *gimbal* souffrent de problèmes de complexité dûs à leur mécanique et du phénomène de *gimbal lock* (lorsque deux axes de cardans sont confondus). Les centrales inertielles à composants liés (*strapdown*) n'ont pas ce type d'inconvénient car elles ne comportent plus cette plateforme.

Tout comme pour les accéléromètres, la technologie *MEMS* s'est rapidement démocratisée dans les solutions *strapdown* pour sa simplicité, sa taille réduite et son faible coût . Un tel gyromètre mesure la vitesse de rotation d'un corps auquel il est attaché. Les gyromètres *MEMS* utilisent en fait un gyroscope vibrant, contrairement aux technologies *solidstate* utilisées dans les gyromètres laser ou à fibre optique. Un gyromètre vibrant peut être représenté par un diapason (voir figure 3.17) dont les oscillations sont entretenues par effet piézoélectrique. Si le diapason est en rotation par rapport à son axe longitudinal, des vibrations orthogonales sont alors générées par les accélérations de Coriolis. Celles-ci peuvent être détectées par effet piézoélectrique. Les amplitudes des vibrations sont alors proportionnelles à la vitesse de rotation appliquée. La sortie du capteur est ensuite de type numérique ou analogique.

Les gyromètres sont généralement caractérisés par leur résolution, leur biais, leur sortie à vitesse nulle et leur facteur d'échelle. Tout comme les accéléromètres, les gyromètres peuvent comporter plusieurs axes dans le même composant. Actuellement, jusqu'à deux axes peuvent être intégrés sur une même puce. Certains fabricants peuvent aussi vendre un *package* contenant 3 axes, mais il s'agit en fait de plusieurs composants déjà alignés. Des *packages* peuvent aussi regrouper accéléromètres et gyromètres, tel que le *ADIS 16355* d'*Analog Devices* (voir figure 3.18).

Magnétomètre

Le magnétomètre mesure le champ magnétique. Dans l'idéal il ne mesure que le champ magnétique terrestre, ce qui lui permet de donner une information de cap, non disponible avec les accéléromètres. Cependant le champ magnétique n'est

Figure 3.16 – Gyroscope inventé par Léon Foucault, et construit par Dumoulin-Froment en 1852. Source : Wikipedia [itt].

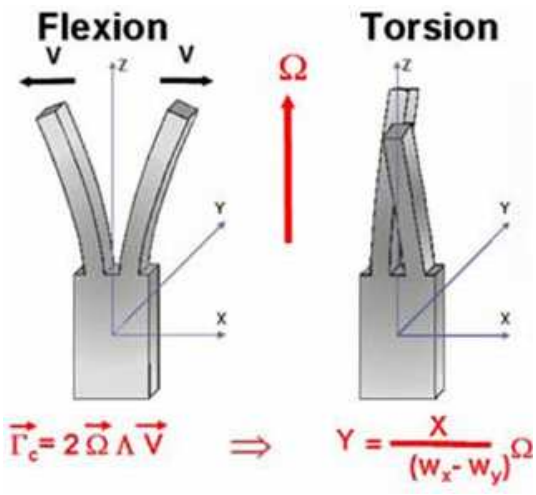


Figure 3.17 – Principe du gyromètre vibrant. Source : site internet de l'*Onera* [idl].

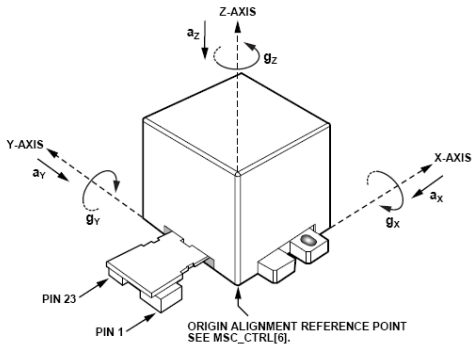


Figure 3.18 – Capteur inertiel 3 axes *ADIS 16355* d'*Analog Devices* : accéléromètre 3 axes et gyromètre 3 axes. Source : site internet *Analog Devices* [id].

pas constant à la surface de la Terre, et dépend donc de l'endroit où il est mesuré.

Malgré tout, dans nos applications, le déplacement total est suffisamment faible pour supposer que le champ magnétique reste constant. La principale source d'erreur dans la mesure vient donc des perturbations magnétiques. Celles-ci peuvent venir du drone en lui même (moteurs, câbles, éléments ferromagnétiques tels que les vis...) ou de l'environnement. Les perturbations produisant un champ magnétique fixe sont relativement faciles à filtrer (cas des éléments ferromagnétiques de la structure), alors que les perturbations variables avec le temps ou la position posent plus de problèmes (cas des moteurs dont le champ produit est fonction de la puissance consommée). En intérieur, le champ magnétique est perturbé par la structure des bâtiments, les équipements électriques et les ordinateurs entre autres. Les travaux de [VMP07a] montrent notamment l'effet de ces perturbations sur le champ magnétique mesuré dans un bureau (voir figure 3.19), ce qui conduit à une faux calcul de l'angle de lacet par exemple (voir figure 3.20).

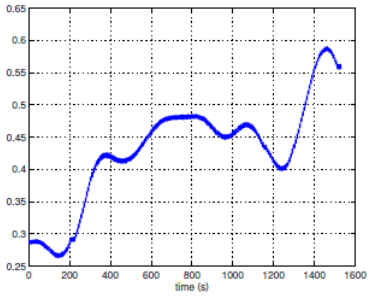


Figure 3.19 – Variation de la norme du champ magnétique lors d'un déplacement horizontal de 2.4 m dans un bureau. Source : [VMP07a].

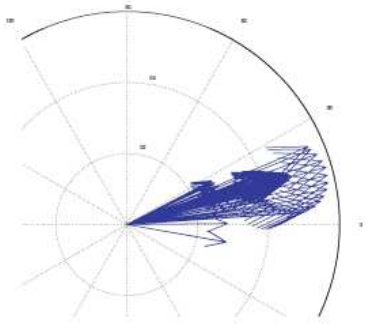


Figure 3.20 – Variation de la projection du champ magnétique lors d'un déplacement horizontal de 2.4 m dans un bureau. L'angle de lacet est fortement perturbé. Source : [VMP07a].

Les quatre technologies principales de magnétomètres sont les capteurs à bobine inductive, les capteurs à effet Hall, les capteurs à porte de flux et les capteurs magnétorésistifs. C'est cette dernière catégorie qui est principalement utilisée dans les capteurs *MEMS*. Ceux-ci utilisent donc des alliages ($Ni_{81}Fe_{19}$ ou $Ni_{65}Fe_{15}Co_{20}$ par exemple) dont la résistance varie avec le champ magnétique qui lui est appliqué.

Fusion de données

En combinant les données des trois types de capteurs précédents, il est possible d'estimer l'attitude. En effet, ces trois types de données sont nécessaires et com-

plémentaires. Par exemple, le gyromètre peut fournir par intégration l'orientation mais seulement sur un temps très court ; sur une période prolongée le résultat sera de plus en plus erroné dû à l'intégration du bruit de mesure. Les accéléromètres et magnétomètres peuvent quant à eux mesurer très précisément l'orientation, mais uniquement dans certaines conditions. Des mouvements imposés à l'accéléromètre lui feront mesurer la gravité ainsi que sa propre accélération, faussant ainsi l'estimation de l'orientation. Par ailleurs, les magnétomètres sont très sensibles aux perturbations magnétiques locales, notamment à proximité d'objets ferromagnétiques.

Les travaux sur le sujet sont nombreux, et différentes approches sont proposées. La plus simple consiste à utiliser un filtre complémentaire (voir [BK97]). Dans ce cas, il s'agit de tirer parti des avantages des accéléromètres à basse fréquence et des gyromètres à haute fréquence. En effet, le filtre consiste à ajouter l'angle estimé par un accéléromètre (utilisé comme inclinomètre) filtré par un passe bas à l'angle estimé du gyromètre (par intégration) filtré par un passe haut. L'estimation donnée par l'accéléromètre étant en effet moins bonne dans le cas de mouvements rapides et l'estimation donnée par le gyromètre étant quant à elle mauvaise à basse fréquence dû à l'intégration du biais. Ce filtre est schématisé sur la figure 3.21 dans le cas d'un pendule. Les résultats montrent que cette méthode est plutôt efficace, sauf si la dynamique de l'appareil devient trop importante. Dans ce cas, les accéléromètres ne fournissent plus une bonne estimation du vecteur gravité. C'est le cas par exemple pour un avion devant effectuer des virages serrés. Ainsi, il a été proposé dans [EKMH07] un filtre complémentaire non linéaire augmenté par un modèle dynamique du premier ordre de l'avion. La vitesse de l'air (mesurée par un capteur de pression) ainsi que la vitesse de braquage de l'avion sont nécessaires afin d'enlever le biais sur la mesure du vecteur gravité.

Le filtre complémentaire est donc assez simple et peut donner de bons résultats sous certaines conditions. Sa simplicité permet par exemple de le réaliser avec des composants analogiques, c'est le cas de [GERLon] où le filtrage est réalisé avec des composants analogiques afin de stabiliser un avion convertible. Cependant le filtre complémentaire tel quel ne fournit pas d'estimation du biais des gyromètres.

Une autre technique d'estimation couramment utilisée est le filtre de Kalman [Kal60]. Celui-ci est un filtre récursif se basant sur l'état précédent et la mesure actuelle pour estimer l'état actuel. Le calcul s'effectue alors en deux étapes : la prédiction et la mise à jour. La première phase estime l'état actuel en utilisant l'état estimé précédent et le modèle du système. La seconde phase va corriger l'estimation prédite grâce aux mesures effectuées. Les filtres de Kalman peuvent être continus ou discrets. Seul le domaine discret est intéressant ici, car les traitements

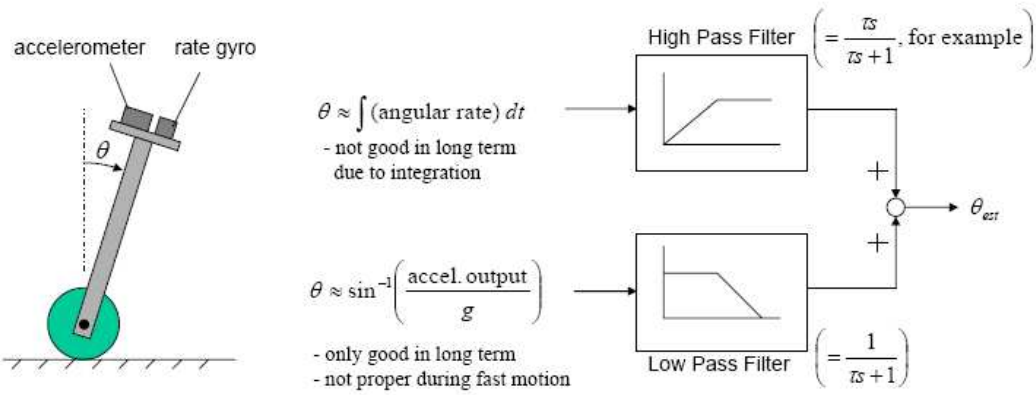


Figure 3.21 – Filtre complémentaire permettant d'estimer l'angle θ d'un pendule. Source : [PH04].

s'effectuent sur un ordinateur ou un microcontrôleur. Le filtre peut ensuite être linéaire (Kalman classique) ou non. Dans le second cas, les plus populaires sont les filtres de Kalman étendus (*EKF*, *Extended Kalman Filter*) et sans parfum [1] (*UKF*, *Unscented Kalman Filter*). De nombreux travaux utilisent donc cette approche. Dans tous les cas, l'utilisation du filtre de Kalman suppose que l'erreur dans le modèle d'état et dans le modèle de mesure est un bruit blanc gaussien de moyenne nulle. La plus grande partie des travaux utilise les quaternions comme représentation de l'attitude. Des filtres additifs (*AEKF*, *Additive Extended Kalman Filter*) ont été utilisés par [BIDM91] et [Cho03], alors que des filtres multiplicatifs (*MEKF*, *Multiplicative Extended Kalman Filter*) ont été utilisés par [LMS82] ou [Mar03b]. Cependant, l'hypothèse sur les bruits n'est pas toujours vraie, ce qui mène à l'utilisation de filtres sans parfum (voir [CM03]), mais les calculs apportés par cette méthode ne sont pas toujours réalisables en temps réel sur les systèmes embarqués. Les travaux de [VBMP08] utilisent une approche différente. Le but est ici d'utiliser un *EKF* pour estimer les différents états d'un hélicoptère classique en utilisant un modèle de l'engin et différents capteurs (inertiels, *GPS* et baromètre). L'objectif des auteurs est de montrer qu'un modèle précis est au moins aussi important que des capteurs de qualité pour obtenir une bonne estimation des états. Ainsi, des capteurs d'entrée de gamme seraient suffisants, ce qui pourrait réduire significativement le coût des drones et donc accélérer leur développement et utilisation. Le modèle proposé tient compte de nombreux effets aérodynamiques et permet alors

1. Parfois appelé sans odeur ou sans saveur. Originellement, il fut nommé *New Filter* (nouveau filtre) par ses auteurs [JUDW95], ceux-ci l'ayant ensuite rebaptisé *Unscented Filter*, [JU97]. Cette terminologie permet par ailleurs de définir l'*Unscented Transform* (transformation sans parfum), utilisée par l'*UKF* mais pouvant aussi servir hors du contexte du filtre.

d'améliorer l'étape de prédiction du filtre de Kalman, pourvu que les différents paramètres soient bien identifiés. Les résultats obtenus sont convaincants, même lorsque certains capteurs sont volontairement mis hors service.

Une autre approche est proposée par [GC08], en utilisant un observateur non linéaire afin d'estimer l'attitude et le biais des gyromètres. L'emploi d'un tel observateur est motivé par le fait que les *EKF* habituellement utilisés sont basés sur une approximation linéaire du modèle non linéaire de mesure ; la convergence globale n'est donc pas assurée. La robustesse de l'observateur proposé par rapport au bruit et à des perturbations est montrée sur des données simulées et réelles.

Les capteurs inertiels peuvent aussi servir à estimer les vitesses de déplacement latéral, voire la position. Cependant, l'approche la plus classique consiste en une double intégration (voir [Fau71] ou [GWA01]), qui n'est pas viable à long terme à cause du biais dans les capteurs. Pour résoudre ce problème, d'autres mesures (*GPS*, odomètres) doivent être considérées. Une nouvelle approche a été proposée par [VMP07a], en se basant sur les perturbations magnétiques de l'environnement. En supposant celles-ci invariantes avec le temps (mais variables avec la position), les équations de Maxwell permettent alors de relier la vitesse au champ magnétique. Les auteurs proposent donc d'utiliser une simple centrale inertielle (accéléromètre 3 axes, gyromètre 3 axes et magnétomètre 3 axes) et un *EKF* pour estimer entre autres la position et la vitesse de déplacement dans un plan. Des résultats expérimentaux montrent l'intérêt de considérer ces perturbations magnétiques car la vitesse estimée ne dérive alors quasiment plus, contrairement à la méthode classique, ce qui améliore nettement l'estimation de la position. Notamment, un essai à long terme (30 min) et faible déplacement (2 m) montre une erreur de position 90 cm ce qui est très peu. Les expériences montrent aussi que dans un environnement peu perturbé (en extérieur par exemple), cette méthode n'apporte évidement pas d'amélioration dû au manque de perturbations magnétiques. Les mêmes auteurs ont ensuite amélioré le dispositif expérimental ([VMP07b]). Le système utilise maintenant une centrale inertielle à laquelle sont ajoutés trois magnétomètres tri-axes afin de former un trièdre orthogonal. Cette configuration permet de mesurer les dérivées partielles spatiales, alors qu'avant leur dynamique était assimilée à un passe bas ayant un bruit blanc. Des résultats convaincants en 3D sont montrés, et l'amélioration par rapport au système précédent est notable. Cependant ce système pourrait ne pas fonctionner à bord d'un drone à cause du champ magnétique variable des moteurs par exemple.

Centrale inertielle utilisée

La centrale inertielle équipant le quadrirotor est une *IG-500N* de *SBG Systems* (voir figure 3.22). Elle a la particularité de posséder, en plus des capteurs inertiels présentés précédemment, un *GPS* (voir section 3.5.2) et un baromètre (voir section 3.5.3). Ces deux capteurs permettant d'obtenir une meilleur précision de mesure grâce à un filtre de Kalman étendu. Cette centrale a été choisie pour son rapport qualité/prix et pour le fait qu'elle embarque un *GPS*.

Figure 3.22 – La centrale inertielle *IG-500N* de *SBG Systems*. Source : site internet *SBG Systems* [itn].

L'*IG-500N* est capable de fournir, via un port série, les données inertielles filtrées ainsi que la position (au format *WGS84*, voir section 3.5.2, et si le *GPS* est disponible) à une fréquence de 100 Hz. L'orientation peut être fournie sous forme d'angles d'Euler, de matrice de rotation *DCM* ou de quaternions. Nous avons donc choisis les angles d'Euler, afin d'utiliser le modèle du quadrirotor obtenu à la section 3.1.3. Les autres données intéressantes délivrées par l'*IG-500N* sont les valeurs des gyromètres et des accéléromètres. Celles-ci sont exprimées dans le repère R_b du drone, mais peuvent facilement s'exprimer dans le repère fixe R_f grâce aux angles d'Euler et à l'équation (3.3). Cependant, comme il a été vu avec le modèle simplifié du quadrirotor (section 3.1.5), dans le cas d'un vol stationnaire ou quasi stationnaire les angles d'Euler sont très faibles. Les repères R_b et R_f sont alors presque alignés et les dérivées des angles d'Euler ($\dot{\phi}$, $\dot{\theta}$ et $\dot{\psi}$) peuvent être approchées par les valeurs des gyromètres. Cette approximation sera donc faite par la suite lors de l'implémentation des lois de commandes sur le quadrirotor.

3.5.2 Systèmes de positionnement par satellite

Le *GPS* (*Global Positioning System*) est actuellement le seul système de positionnement par satellite (ou *GNSS*, *Global Navigation Satellite Systems*) totalement opérationnel. Développé aux États-Unis, il couvre depuis 1995, et grâce à une

flotte de 24 satellites, l'ensemble de la surface de la Terre. A cette époque, la précision est alors d'une centaine de mètres dans le plan horizontal. En effet, le système développé par le département de la défense américaine n'est qu'en partie accessible aux civils. Certains signaux envoyés par les satellites étant volontairement dégradés (en particulier les informations concernant l'horloge), alors que d'autres signaux codés (réservés aux militaires) ne le sont pas. Le 1^{er} mai 2000, Bill Clinton décide de rendre accessible à tous cette technologie et ne restreint plus les émissions des satellites, ramenant la précision à 15 m dans le plan horizontal. Le *GPS* s'est alors énormément et rapidement démocratisé, notamment pour l'aide à la conduite avec les *GPS* de navigation. Ces derniers permettent aux automobilistes de se localiser sur une carte routière et de les conduire à la destination voulue. Enfin, le *GPS* utilise le système géodésique *WGS84* (voir[NIM97]) pour exprimer ses coordonnées.

Des systèmes concurrents au *GPS* existent aussi. Dans un contexte de Guerre Froide, la Russie a développé à partir de 1976 son système *GLONASS* (*GLObal NAvigation Satellite System*). Les 24 satellites nécessaires à la couverture mondiale sont en orbite en 1995, mais la chute de l'Union Soviétique entraina aussi la chute des crédits alloués au système de positionnement. En 2000, seuls six satellites sont encore opérationnels ; en 2001 un nouveau programme est lancé pour améliorer le système. Des satellites de nouvelle génération sont alors lancés peu à peu, portant à 21 le nombre de satellites actuellement [2] opérationnels. Les informations à jour sur ce système (encore en évolution) peuvent se trouver sur le site de l'agence spatiale russe dédié à *GLONASS*, [idlsrdàt]. Le système ne couvre donc pas actuellement l'ensemble de la surface terrestre, mais il couvre tout le territoire russe ; sachant que la dernière génération de satellites lancés permet une précision de 15 m dans le plan horizontal. Tout comme le *GPS*, un signal de meilleure précision est réservé aux militaires russes. Le 18 mai 2007, Vladimir Poutine rend officiellement accès aux signaux de navigation civils de *GLONASS*. Initialement, le système géodésique utilisé était le système russe *PZ-90* dont les paramètres de transformation avec le *WGS84* n'étaient pas connus précisément. Dans un soucis d'interopérabilité, *GLONASS* a adopté en septembre 2007 le système *PZ-90.02* compatible avec le système américain. Les autres systèmes de positionnement actuellement en développement sont *Galileo* (Union Européenne), *Compass Navigation Satellite System* (Chine) et *Indian Regional Navigational Satellite System* (Inde).

De nombreux travaux visant à améliorer la précision de la position existent. La première approche utilise le fait que les deux informations du *GPS* (position et vitesse) proviennent de deux sources distinctes :

2. Au 30 mars 2010, le dernier lancement date du 2 mars 2010.

- mesure de pseudodistance et/ou de phase pour la position,
- mesure de fréquence Doppler pour la vitesse.

Un filtre de Kalman peut donc utiliser les mesures de vitesse pour améliorer l'estimation de la position. L'autre technique courante est le *map matching*. Celle-ci consiste à supposer que le récepteur *GPS* se trouve sur une route (dont la position est parfaitement connue) afin de recaler sa position.

D'autres méthodes existent afin d'améliorer la précision du *GPS*. Ainsi le *DGPS* (*Differential Global Positioning System*) ou *GPS* différentiel, utilise les données d'une station fixe afin d'estimer les erreurs de mesure. En effet, la station fixe étant censée être proche du *GPS* dont on cherche à améliorer la précision, elle reçoit les mêmes erreurs de mesure. Connaissant exactement sa position, la station estime les erreurs et les envoie (par radio ou par satellite) au récepteur *GPS* afin qu'il corrige lui même sa position. La précision peut alors être de l'ordre de 3 à 5 m. Notons que la technique de *DGPS* a été développée au début des années 1990 afin de tenter d'améliorer le signal volontairement dégradé fournit par le *GPS*. Rapidement, le *DGPS* est devenu aussi précis que le *GPS* codé, mais moins pratique car il nécessite une station fixe. Le *RTK* (*Real Time Kinematic*), ou cinématique temps réel est une autre technique utilisant les mesures de phase des ondes porteuses et une station de référence afin d'atteindre une position de l'ordre du centimètre.

L'utilisation du *GPS* sur une drone miniature tel que celui utilisé pour cette thèse reste donc un défi. En effet, l'envergure de la plateforme est de 70 cm ce qui est bien inférieur à la précision du *GPS* classique. De plus, contrairement à ce qui est fait pour les automobiles, le drone n'évolue pas nécessairement sur une route, donc les algorithmes de *map matching* pour recaler la position ne sont pas possibles. Il est donc nécessaire d'utiliser d'autres méthodes, telles que le *DGPS* ou le *RTK* présentés précédemment par exemple. Cependant, la qualité des récepteurs et antennes *GPS* embarquables dans les drones est bien inférieure à ce qui est employé couramment pour atteindre les précision citées auparavant. La précision obtenue sera sans aucun doute bien mois bonne.

Une autre solution, proposée d'ailleurs par certains fabricants de centrales inertielles, consiste à coupler les données du *GPS* avec les données inertielles. Cette fusion de données se fait généralement à l'aide d'un filtre de Kalman. Ainsi, suivant le fabricant de la centrale que nous utilisons, la précision de la position est de 2 m. En effet, les mesures du *GPS* permettent d'estimer l'accélération de la centrale afin d'améliorer la mesure du vecteur gravité servant à calculer l'attitude.

3.5.3 Autres capteurs

Télémètres

En plus des capteurs élémentaires présentés auparavant (centrale inertielle et *GPS*), le drone peut emporter d'autres capteurs complémentaires pour aider à sa navigation. Il s'agit principalement de télémètres. Ceux-ci peuvent être de type infrarouge (voir figure 3.23), ultra son (voir figure 3.24) ou laser (appelés *Lidar*, pour *Light Detection and Ranging*). La précision et la portée dépendent du type de capteur utilisé et du modèle. Les télémètres laser ayant les meilleures performances mais étant aussi les plus onéreux et les plus encombrants. Par ailleurs, certains *Lidar* permettent d'obtenir une mesure à balayage. C'est le cas notamment du laser *Hokuyo URG-04LX* (voir figure 3.25), offrant des mesures sur 240° pour une distance allant jusqu'à 4 m.

Figure 3.23 – Télémètre infrarouge *Sharp GP2Y0A02YK*. Source : site internet *Robotshop* [itm].

Figure 3.24 – Télémètre ultrason *Devantech SRF08*. Source : site internet *Robotshop* [itm].

Figure 3.25 – Laser *Hokuyo URG-04LX*. Source : site internet *Hokuyo* [itg].

Il est à noter que la réponse du capteur infrarouge *GP2Y0A02YK* de *Sharp* (l'un des plus utilisé car peu coûteux) n'est pas bijective ; la courbe de la sortie analogique en fonction de la distance est représentée sur la figure 3.26. Cela

peut donc poser problèmes pour mesurer de petites distances, notamment en s'approchant du sol. Ce type de capteur n'est donc pas conseillé pour effectuer des tâches d'atterrissage ou de décollage automatique.

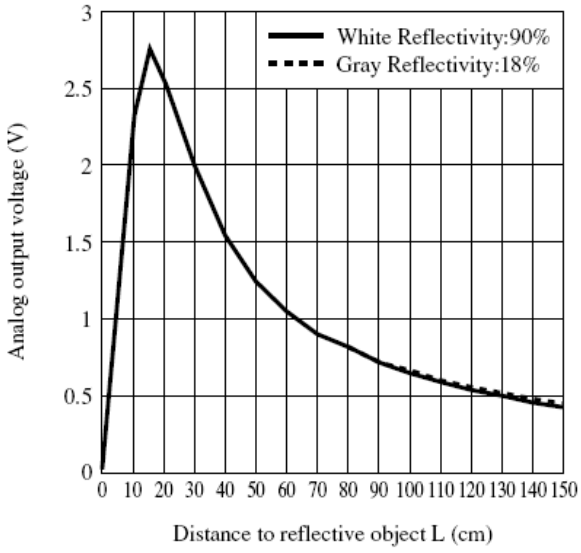


Figure 3.26 – Courbe du télémètre infrarouge *GP2Y0A02YK* : tension (V) en fonction de la distance (cm) à l'objet. Source : *datasheet Sharp GP2Y0A02YK* [ito].

Ces capteurs peuvent servir à détecter les obstacles ou à mesurer la distance au sol. Dans ce dernier cas, ils sont en général associés à un accéléromètre afin de pouvoir estimer aussi la vitesse de déplacement suivant l'axe vertical. Un filtre de Kalman (voir [Kal60]) peut par exemple être utilisé. Considérons l'état suivant :

$$X_k = \begin{bmatrix} z_k \\ v_k \\ a_k \end{bmatrix} \tag{3.33}$$

où z, v et a représentent respectivement l'altitude, la vitesse et l'accélération. La mesure étant effectuée par l'accéléromètre (a^m) et le télémètre (z^m), le vecteur de mesure peut s'écrire :

$$Z_k = \begin{bmatrix} z_k^m \\ a_k^m \end{bmatrix} \tag{3.34}$$

En supposant que l'accélération varie peu, que le process est entaché d'un bruit W_k blanc gaussien de matrice de covariance Q_k^W et que la mesure est entachée d'un bruit V_k blanc gaussien de matrice de covariance Q_k^V, alors on obtient le système

linéaire suivant :

$$X_{k+1} = FX_k + W_k \tag{3.35a}$$
$$Z_k = HX_k + V_k \tag{3.35b}$$

avec,

$$F = \begin{bmatrix} 1 & T & 0 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \tag{3.36a}$$
$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.36b}$$

où T est la période d'échantillonnage. Supposons que les matrices de covariances sont constantes : $Q_k^W = Q^W$ et $Q_k^V = Q^V$, que la condition initiale X_0 (de moyenne $\overline{X}_0$) est gaussienne de matrice de covariance Q_0^X et que les bruits W_k, V_k et la condition initiale X_0 sont mutuellement indépendants. Le filtre de Kalman appliqué à ce système se calcul alors en deux étapes.

Prédiction :

$$\hat{X}_k^- = F\hat{X}_{k-1} \tag{3.37a}$$
$$P_k^- = FP_{k-1}F^T + Q^W \tag{3.37b}$$

Correction :

$$\hat{X}_k = \hat{X}_k^- + K_k\Big(Z_k - H\hat{X}_k^-\Big) \tag{3.38a}$$
$$P_k = \Big(I - K_kH\Big)P_k^- \tag{3.38b}$$

avec :

$$K_k = P_k^-H^T\Big(HP_k^-H^T + Q^V\Big)^{-1} \tag{3.39}$$

le filtre est initialisé avec les valeurs suivantes :

$$\hat{X}_0^- = \overline{X}_0 \tag{3.40a}$$
$$P_0^- = Q_0^X \tag{3.40b}$$

Ce filtre permet alors d'estimer position, vitesse et accélération suivant l'axe z à partir des mesures de l'accéléromètre et d'un télémètre.

Baromètres

Un capteur barométrique (voir figure 3.27) peut aussi être intégré au drone afin de mesurer la pression et donc de calculer l'altitude (en le couplant par exemple avec un accéléromètre, tel que vu ci-dessus). Ce capteur doit alors être suffisamment précis pour capter de faibles variations de pression. Cependant, les capteurs offrant une telle précision ont souvent un rang très limité ; réduisant aussi la plage d'altitudes de mesure. Enfin, la précision du capteur le rend très sensible aux perturbations atmosphériques, notamment aux rafales de vent pouvant alors fausser la mesure. Un filtre en mousse peut être placé sur le capteur afin d'éviter ce genre de perturbations ; il joue alors le rôle d'un passe bas.

Figure 3.27 – Capteur de pression. Source : site *Freescale* [idtb].

Caméras

Enfin, une (ou plusieurs) caméras peuvent aussi être embarquées dans le drone. Le premier but est alors de retransmettre les images au sol, cependant les caméras peuvent aussi être utilisées comme capteur car leur large champ de vision permet d'avoir un grand nombre d'informations sur l'environnement du drone. Un étude plus approfondie de ce type de capteur (calibration, utilisation, etc) est faite au chapitre 5. Les solutions de traitement vidéo quant à elles sont traitées à la section 3.9.

3.6 Communication air/sol

Le drone, bien qu'autonome, doit toujours garder une liaison avec le sol. En effet, cela permet avant tout de pouvoir reprendre la main en cas de problèmes. Ainsi si le comportement du drone devient instable, l'opérateur au sol peut couper

les gaz ou revenir en mode manuel pour éviter tout accident. Cette liaison permet aussi d'effectuer les décollages et atterrissages en mode manuel, et de trimer les angles en cas de nécessité.

Figure 3.28 – Radiocommande et récepteur radio. Source : site internet *Futaba* [idtc].

A l'origine, cette liaison unidirectionnelle était effectuée grâce à une radiocommande (voir figure 3.28) issue du modélisme. Celle-ci est associée à un récepteur radio intégré au drone. La liaison sans fils s'effectue à 41 MHz, un quartz permettant de changer légèrement la fréquence pour éviter les perturbations avec d'autres radiocommandes. La portée est d'environ 200 m. Ce type de radio possède en général de quatre à six canaux. Le signal de sortie du récepteur est représenté sur la figure 3.29, il s'agit d'un signal *PPM* (*Pulse Position Modulation*). La valeur de chacun des canaux est en fait codée par la durée entre deux fronts montants successifs ; cette durée étant comprise entre 1 et 2 ms. La durée des états hauts est fixe, de l'ordre de 0.5 ms. Un état bas de plus de 5 ms indique le début d'une nouvelle trame. Généralement la durée totale d'une trame est de 20 ms, c'est à dire une fréquence de 50 Hz. Afin de récupérer les valeurs des canaux, une simple interruption de type *input capture* sur le microcontrôleur suffit. Cette interruption est en effet associée à un *timer*, et permet donc de calculer le temps séparant deux fronts montants successifs. La précision de la mesure dépend alors de la précision du *timer* utilisé.

Cependant, même si la fréquence utilisée est réservée pour ce type d'applications, le signal reçu comporte du bruit. Il suffit en effet d'approcher l'antenne de la radio d'une masse métallique pour voir les valeurs des canaux changer. Les principaux inconvénients de ce mode de communication sont donc son aspect unilatéral et le fait que la communication ne possède aucun outil de contrôle de type

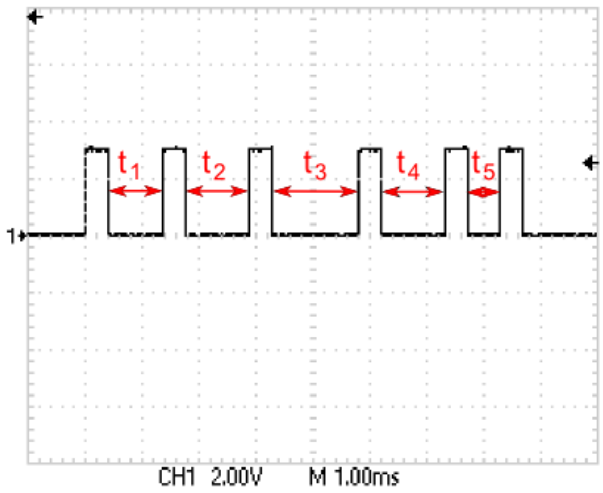


Figure 3.29 – Trame d'une radiocommande 5 canaux vue à l'oscilloscope.

checksum. Cependant elle reste très simple à mettre en place et offre un moyen de commande léger, la radiocommande, facilement transportable. Des versions plus évoluées sont maintenant disponibles sur le marché, utilisant une fréquence de 2.4 GHz et un mode de communication numérique. Celles-ci n'ont pas été testées dans le cadre de cette thèse mais semblent avoir moins de bruit. En effet, les radios à 2.4 GHz utilisent soit la technologie *FHSS* (*Frequency Hopping Spread Spectrum*) de saut de fréquence, soit le *DSSS* (*Direct Sequence Spread Spectrum*) d'étalement direct. Ainsi, même s'il y a plusieurs utilisateurs sur la même fréquence, les données restent garanties. De plus ces technologies permettent d'envoyer plus de trames par seconde que les radiocommandes classiques ; les mauvaises trames peuvent donc être éliminées tout en assurant un débit de 50 Hz au récepteur. Les récepteurs 41 MHz quant à eux sont capables de repérer des trames erronées, mais ils se contentent alors d'utiliser la dernière trame valide reçue.

Figure 3.30 – Modem *XbeePro*. Source : site internet *SparkFun* [itp].

La complexité du drone évoluant, cette liaison unilatérale s'est retrouvée insuffisante. En effet la radiocommande ne permet que d'envoyer des consignes de pilotage, ce qui reste limité. Ainsi, une liaison numérique par modem sans fils a été mise en place. Cette liaison étant bilatérale, elle offre donc de nombreuses possibilités. Associée à une interface homme machine sur ordinateur (voir section 3.10.1), elle permet d'envoyer au drone des consignes de haut niveau sur l'orientation (via un *joystick*), la position, etc... Il est aussi possible de modifier en temps réel les gains utilisés pour les lois de commande. De même, le drone peut envoyer toutes les informations (*log*) des capteurs au sol. Le modem retenu pour le drone est un *XBeePro* de chez *Maxstream*, voir figure 3.30. En effet ce modem est très populaire et est habituellement utilisé au laboratoire. Il a pour avantage d'être relativement bon marché, mais surtout de posséder un mode "transparent". Placé dans ce mode, le modem gère automatiquement les transmissions et il suffit de le brancher au port série d'un ordinateur ou d'un microcontrôleur ; la communication est alors tout de suite opérationnelle. Cependant, ce mode "transparent" reste très limité. En effet, lorsqu'une donnée arrive sur la broche de réception du modem, elle est mise dans un *buffer* d'entrée. La donnée sera alors transmise à l'autre modem si l'une des conditions suivantes est satisfaite :

- si le *buffer* est plein (sa taille étant paramétrable),
- ou si un délai (lui aussi paramétrable) s'est écoulé sans que le modem ne reçoive de données.

Une trame *XBee* contenant les données du *buffer* est alors envoyée et le modem récepteur envoi un *ACK* (*acknowledgment*) pour signaler qu'il a bien reçu la trame. Si l'émetteur ne reçoit pas cet *ACK* après un certain temps, il réitère la transmission (jusqu'à six fois). Lorsque le récepteur reçoit une trame, il la transmet via sa broche de sortie. Cependant, si la trame n'a pas pu être reçue elle est perdue. Dans ce cas, le modem n'avertit pas le microcontrôleur (ou l'ordinateur) car en mode transparent aucune donnée autre qu'une donnée envoyée par l'émetteur ne sort du modem. Il apparaît donc nécessaire de vérifier les données reçues grâce à un *checksum* par exemple, car des données peuvent être perdues. Par ailleurs, une trame *XBee* perdue peut correspondre à plusieurs trames de la communication ordinateur-microcontrôleur. De plus, étant donné que le modem renvoie les données non reçues, il est possible que les trames n'arrivent pas dans l'ordre où elles ont été envoyées, voir figure 3.31. Enfin, ce mode n'assure donc pas qu'une donnée envoyée arrive à destination. Cela peut donc être problématique, car certaines trames doivent être reçues, tel que des ordres d'arrêt des moteurs en cas de problèmes. En général, les trames sont perdues à cause de collisions. En effet, la liaison radio fréquence n'est pas bidirectionnelle. Un seul des deux modems ne peut donc transmettre à la fois, sinon il y a collision et la trame est perdue. Pour éviter cela, les modems possèdent des algorithmes anti-collisions plus ou moins

évolués. Cependant, plus les fréquences d'envoi sont élevées, plus le risque de collisions est élevé. Ainsi, dans l'application de station sol environ 10% des paquets sont perdus. La figure 3.32 montre qu'à période d'envoi de la station sol fixée, plus il est demandé au drone d'envoyer des données, plus les trames sont perdues.

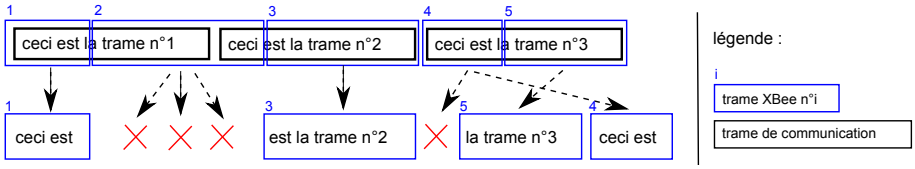


Figure 3.31 – Schéma d'une communication *XBee*. La trame *XBee* n ° 2 a été envoyée trois fois sans succès et est perdue. La trame *Xbee* n ° 4 n'a pas été reçue la première fois ; elle a été reçue à la seconde tentative mais entre temps le trame *XBee* n ° 5 a été reçue.

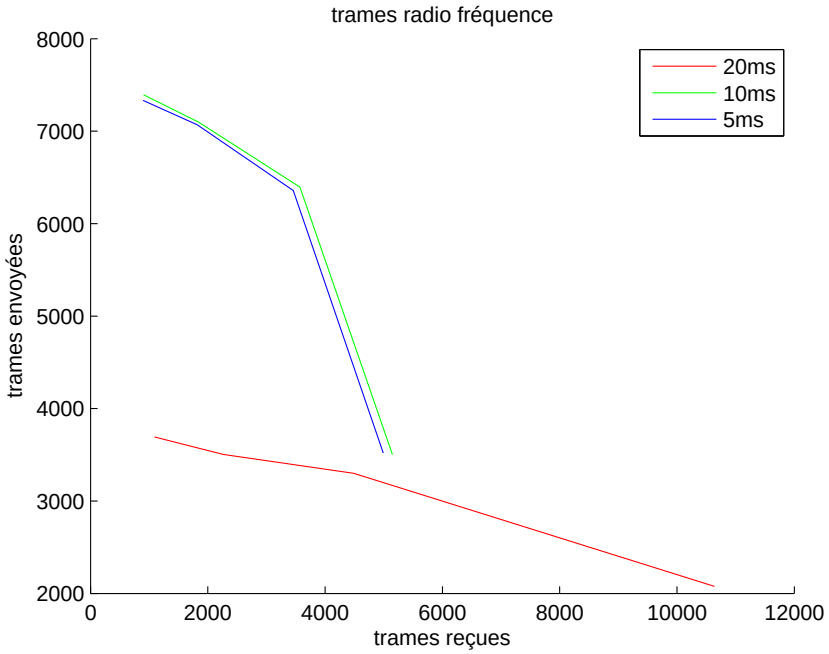


Figure 3.32 – Trames effectivement envoyées en fonction des trames effectivement reçues (point de vue station sol), sur un intervalle de 2 minutes. Chaque courbe correspond à une période d'envoi différente. Les courbes montrent que plus le microcontrôleur émet, plus les trames sont perdues. Les différentes courbes montrent aussi que l'ordinateur n'est pas capable d'envoyer à une période inférieure à 10 ms.

Le modem possède cependant un autre mode, dit *API* (*Application Programming Interface*), beaucoup plus complet mais aussi plus complexe. Dans ce mode,

l'utilisateur envoie les données au modem sous une syntaxe bien précise. Cela permet entre autres de former une trame qui sera envoyée en totalité dans une seule trame *XBee*, et de recevoir l'*ACK* ou le non-*ACK* (*NACK*). En cas de *NACK*, le logiciel peut décider de renvoyer la trame si elle est importante. Le protocole de communication inclut par ailleurs un *checksum* et donne pour chaque trame reçue la puissance du signal.

3.7 Microcontrôleur

Le microcontrôleur est le "cerveau" du drone. C'est lui qui va recevoir les informations des différents capteurs, les filtrer, effectuer les calculs liés aux lois de commandes et piloter les actionneurs. Choisir un microcontrôleur n'est pas chose aisée, car il existe de nombreux fabricants et chacun d'eux possède une large gamme. Le choix du microcontrôleur dépend de l'application ; les critères peuvent être la taille du bus de données, la fréquence d'horloge, le nombre de périphériques, le nombre d'entrées-sorties, la taille mémoire, le *package*, le prix... Il faut aussi prendre en compte ce qui est autour du microcontrôleur, c'est à dire la disponibilité d'outils de développement (gratuits ou non), les outils de programmation (libres ou non), la possibilité d'avoir des échantillons (*samples*), la présence d'une communauté travaillant sur cette architecture... Ainsi, des microcontrôleurs de fabricants différents peuvent s'avérer à peu près équivalents techniquement et le choix reste à l'utilisateur en fonction de ses habitudes.

Pour choisir le microcontrôleur de la plateforme, un cahier des charges englobant différentes architectures possibles a donc été défini :

- contrôle par radiocommande et/ou modem.
- moteurs *brushless* avec ou sans contrôle de vitesse.
- centrale inertielle du commerce par port série, ou capteurs inertiels (analogiques ou numériques).
- télémètres analogiques ou numériques (pour mesurer la distance au sol ou aux obstacles).
- possibilité de communiquer avec une autre carte effectuant les calculs de vision embarquée.

Pour répondre à cela, les périphériques suivants sont donc nécessaires :

- 4 *PWM* pour les moteurs,
- 4 *input capture* pour les capteurs de vitesse,
- 1 *input capture* pour la radiocommande,
- 1 port série pour le modem,
- 1 port série pour la centrale inertielle,

- 9 entrées analogiques pour les capteurs inertiels,
- 3 entrées analogiques pour les autres capteurs analogiques (baromètre, télémètres),
- 1 bus *SPI* pour les capteurs numériques (inertiels, baromètre, télémètres),
- 1 bus *I2C* pour les capteurs numériques (inertiels, baromètre, télémètres),
- 1 port de communication pour recevoir les données du système de vision embarqué (port série, bus *I2C* ou *SPI*).

Il faut noter que dans le cas des capteurs numériques le bus de communication peut être *SPI* ou *I2C*, suivant le constructeur. Par ailleurs, la disponibilité en *samples* (échantillons), l'existence d'outils de développements gratuits et d'outils de programmation libres faisaient aussi parti des critères de sélection.

Enfin, il est assez difficile d'évaluer la puissance de calcul nécessaire pour le logiciel embarqué (celui-ci pouvant d'ailleurs évoluer et se complexifier). Cependant, l'équipe a souvent travaillé avec un microcontrôleur 8 bits *Rabbit* 3400, fonctionnant à 29 MHz. Ce microcontrôleur semblait suffisant dans bien des cas en termes de calculs, mais il ne répond pas au cahier des charges ci-dessus. Nous avons donc cherché un microcontrôleur au moins aussi puissant.

Notre choix s'est ainsi arrêté sur le *MCF51QE128* de *Freescale*. Il a été choisi pour ses caractéristiques suivantes :

- 32 bits, 50 MHz, 128 Ko de mémoire *flash*, 16 Ko de mémoire *ram*,
- 2 *timers* 6 canaux et 1 *timer* 3 canaux de résolution 32 bits (pouvant servir aux *PWM* et aux *input capture*),
- 2 ports série,
- 2 bus *I2C*,
- 2 bus *SPI*,
- 20 entrées analogiques (*ADC*, *Analog to Digital Converter*),
- outil de développement gratuit (*CodeWarrior for Microcontrollers*), mais limité à la compilation de programmes de 96 Ko (le programme développé fait actuellement environ 65 Ko),
- outils de programmation libres (*USBDM* et *OSBDM* largement décrits sur les forums, voir [dal]),
- disponibilité d'une carte de développement peu onéreuse : la *DEMOQE128*.

Le laboratoire a acquis la carte de développement citée ci-dessus afin de débuter avec le microcontrôleur avant de passer à l'application réelle. Cette carte permet par ailleurs de programmer le *MCF51QE128* sur une carte fille, nous évitant ainsi d'acquérir/fabriquer un outil de développement supplémentaire.

3.8 Électronique embarquée

Une carte électronique (voir figure 3.33) a été conçue et réalisée afin de réunir les différents éléments vu précédemment autour du microcontrôleur, et d'être la plus polyvalente possible. Le cahier des charges est donc assez similaire à celui établi pour le microcontrôleur (voir section 3.7). Cependant, pour éviter le bruit électronique induit par les moteurs, les alimentations de puissance et de commande ont été totalement découplées. Cela permet d'avoir notamment des mesures moins bruitées sur les capteurs inertiels (gyromètres et accéléromètres). Le seul inconvénient à ce découplage est que le drone nécessite alors deux batteries distinctes (11.1 V, 2400 mAh pour la puissance, et 7.4 V, 400 mAh pour la commande). La batterie supplémentaire étant de faible capacité, le surpoids engendré reste toutefois raisonnable (25 g en plus des 230 g). Enfin, le cahier des charges prévoyant de pouvoir faire la fusion de données des capteurs inertiels dans le microcontrôleur (afin de se passer d'une centrale inertielle "commerciale"), il a été décidé d'utiliser deux microcontrôleurs *Freescale MCF51QE128*. En effet, lors de la conception de la carte il était difficile d'estimer quelle serait la puissance de calcul nécessaire pour tout effectuer. Cette solution permet de décharger complètement les calculs inertiels sur un autre microcontrôleur (dorénavant nommé $\mu C2$) qui peut être vu comme un simple capteur pour le microcontrôleur principal (dorénavant nommé $\mu C1$).

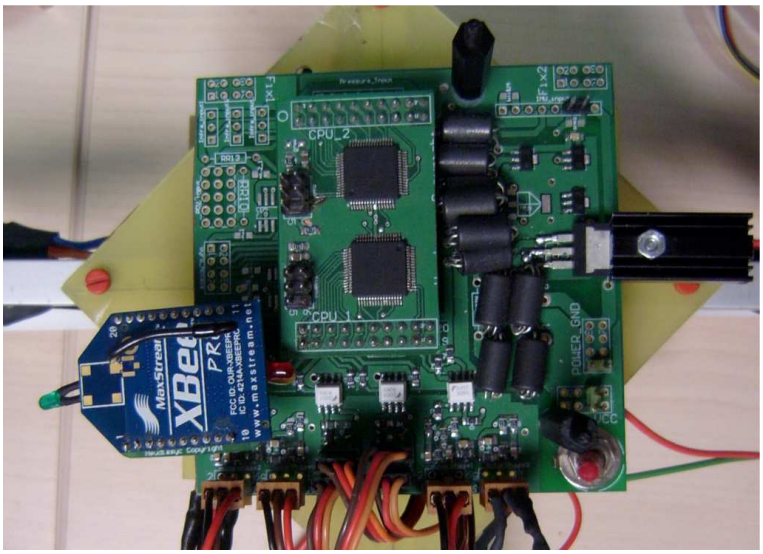


Figure 3.33 – Carte électronique.

Toutes les lois de commandes (y compris celles des moteurs) sont donc calculées

dans *µC1*. Celui-ci est relié au modem *XBeePro* par un port série, au récepteur radio et à *µC2* par un port *SPI*. Les capteurs quant à eux sont reliés à *µC2*. Ce dernier peut alors soit fusionner les données des capteurs inertiels pour fournir à *µC1* une estimation de l'orientation, soit lui donner directement les informations de la centrale inertielle. Enfin, un port de communication *SPI* est disponible sur *µC2* afin de recevoir les informations d'un autre capteur (système de vision, baromètre, etc...). Les informations venant de ce dernier capteur peuvent ainsi être fusionnées avec les autres données avant d'être envoyées à *µC1*. L'architecture complète est représentée sur la figure 3.34.

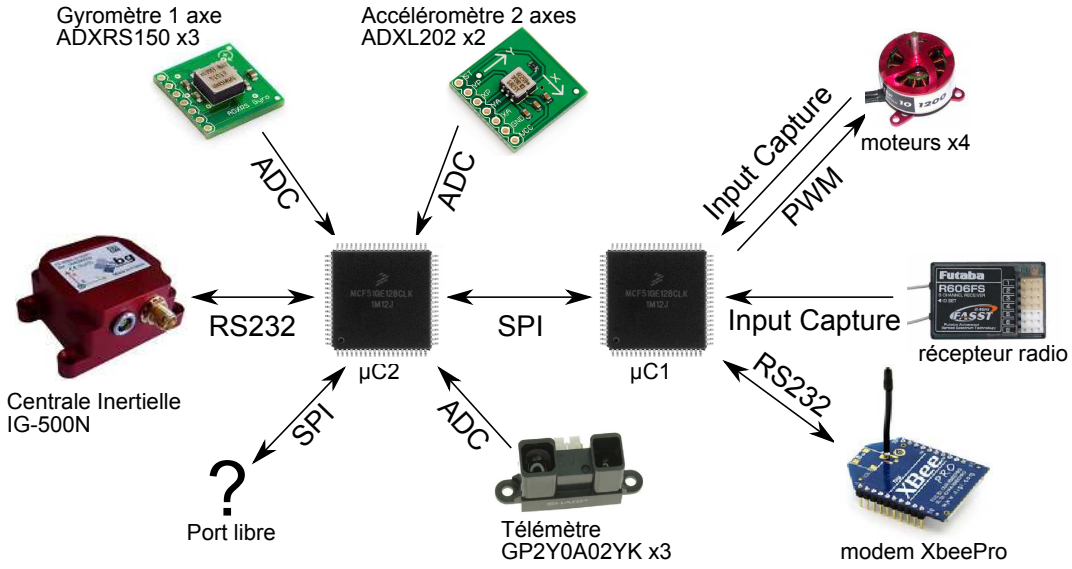


Figure 3.34 – Schéma de l'architecture embarquée. Sources : [itp], [ito], [idtb], [itn] et [itk].

L'architecture a été représentée avec tous ses périphériques possibles. Cependant, la récepteur radio n'a pas été utilisé car le modem offrait plus de possibilités. De même la fusion des données inertielles n'a pas été faite (cela reste un des travaux futurs) et seule la centrale inertielle a été utilisée.

3.9 Solutions de traitement vidéo

Les calculs de vision peuvent être embarqués ou déportés. En général, ils sont déportés lorsqu'il n'est pas possible de les embarquer. En effet ceux-ci peuvent être assez complexes ; c'est le cas du flux optique (voir section 5.2.1) ou de la stéréovision (voir section 5.2.2) par exemple. Effectuer ces types de traitements en temps

réel nécessite donc une puissance de calcul conséquente, que les ordinateurs récents sont capables de délivrer. Cependant les cartes électroniques "embarquables" ayant une puissance à peu près équivalente à un ordinateur sont encombrantes, lourdes, consomment beaucoup d'énergie et nécessitent d'être refroidies. Or ces points ne sont pas compatibles avec la plateforme retenue pour nos essais (voir section 2.11) ; notre quadrirotor étant de puissance modérée pour éviter d'être trop dangereux, il ne peut donc pas emporter une charge utile importante. Les solutions embarquées envisageables doivent donc être bien plus légères et peu consommatrices en énergie, mais aussi de puissance de calcul plus faible. Cette section présente donc des solutions embarquées compatibles avec la plateforme puis introduit quelques remarques sur les solutions déportées. Dans le premier cas, nous nous sommes intéressés ici qu'aux solutions dites "sur étagère" (*off-the-shelf*), c'est à dire des cartes du commerce prêtes à l'emploi. Des solutions spécifiques plus performantes existent sans aucun doute mais elles nécessitent alors beaucoup plus de temps à développer et à réaliser. Il est à noter que la technologie évoluant rapidement, la liste dressée dans cette section est représentative de l'existant au moment de la rédaction, mais est sujette à évoluer.

3.9.1 Solutions embarquées "clé en main"

Une solution "clé en main" est un système regroupant à la fois la caméra et le processeur de calcul. Tout est donc réuni et le système est prêt à l'usage. Par exemple, l'*HeliCommand 3A* (voir figure 3.35) fabriqué par *Captron* est un module destiné aux hélicoptères et s'interfaçant entre le récepteur radio et les actionneurs (moteurs et servomoteurs). Ce module a la particularité d'avoir (entre autres) un capteur *CCD* permettant de calculer le flux optique (voir section 5.2.1). En le pointant vers le sol, le module est alors capable d'estimer les vitesses de déplacement latérales afin de stabiliser l'hélicoptère. Le gain de cette action est configurable via la radiocommande ou depuis un ordinateur. Peu d'informations sont données sur la manière dont fonctionne ce module, notamment il n'est pas dit si les gyromètres embarqués permettent de compenser le flux optique rotationnel. L'*HeliCommand 3A* n'ayant pas de capteur d'altitude, le flux optique calculé dépend donc de la hauteur et l'action de contrôle aussi : il est recommandé de voler jusqu'à 3 m par temps calme et à 1 m en cas de vent. Une version plus évoluée, l'*HeliCommand Profi* (voir figure 3.36) est aussi disponible, intégrant plus de capteurs. Notamment, des capteurs d'altitude (baromètre et télémètre infrarouge) ont été ajoutés, permettant de voler plus haut (30 m) et sans doute d'avoir une mesure de flux optique moins dépendante de la hauteur. Destinés aux débutants voulant apprendre à piloter un hélicoptère ou aux professionnels cherchant une plateforme

stable pour des prises de vues aériennes, ces modules sont donc intéressants et montrent de très bonnes performances. Cependant, ce type de système n'est pas assez ouvert et modulaire pour pouvoir s'intégrer facilement dans notre drone.

Figure 3.35 – L'*HeliCommand 3A*.
Source : site *HeliCommand* [itf].

Figure 3.36 – L'*HeliCommand Profi*.
Source : site *HeliCommand* [itf].

Les solutions "clé en main" présentées par la suite sont donc plus "adaptées" (même si parfois moins puissantes) car elle permettent plus de réglages ou autorisent la programmation. C'est le cas de la *CMUCam*, développée par la *Carnegie Mellon University*. La première version de la *CMUCam* (voir figure 3.37) a été réalisée en 2002, voir [RRN02], il s'agit d'un des premiers systèmes de vision embarqué *low cost* commercialisé. Cette carte permet de faire de la reconnaissance de couleur et de fournir, via un port série, la position et la dimension dans l'image d'un rectangle englobant la couleur recherchée. Ce traitement se fait sur une image de 143x80 pixels et à 17 images par seconde. Ces performances sont aujourd'hui obsolètes, mais à sa sortie la *CMUCam* représentait une grande avancée dans le monde de la robotique et surtout pour la robotique ludique. La *CMUCam* a ensuite évolué et la version 3 (voir figure 3.38) est sortie en 2007. Cette version est maintenant *open source* et programmable. Le kit de développement fourni permet de réaliser facilement ses propres programmes de vision. Elle offre ainsi beaucoup plus de possibilités et est plus puissante que la première version ; elle est basée sur un *ARM7TDMI* (32 bits, 60 MHz) et un capteur *Omnivision OV6620* (352x288 pixels, 26 images par seconde). Cependant les performances de ce système restent encore limitées. Ainsi l'algorithme de reconnaissance de visage de [VJ04] a été adapté par [Goe] pour la *CMUCam3*, mais il nécessite 5 à 6 secondes pour s'exécuter. L'algorithme *Polly* ([Hor93]) a aussi été implémenté sur la *CMUCam3* [RGGN07]. Cet algorithme fournit des informations visuelles basées sur la couleur pour la navigation, il a été utilisé sur le robot *Polly* afin d'effectuer

des tours du laboratoire *MIT AI*. Sur la *CMUCam3*, cet algorithme fonctionne à 4 images par seconde sur des images de 176x144 pixels. La *CMUCam3* est donc une solution intéressante mais pas assez performante pour les applications de cette thèse.

Figure 3.37 – La *CMUCam*. Source : site *CMUCam* [idlt].

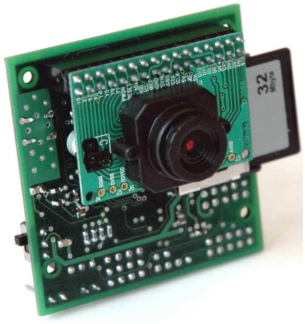


Figure 3.38 – La *CMUCam3*. Source : site *CMUCam* [idlt].

Une autre solution très similaire à la *CMUCam3* est la *Pob-Eye* (voir figure 3.39). Elle utilise aussi un microprocesseur de type *ARM7TDMI* (32 bits, 60 MHz) ; les images capturées sont par contre de résolution inférieure : 120x88 pixels. Les performances sont donc relativement similaires à celles de la *CMUCam3*. Ce système est lui aussi très orienté robotique ludique, et le fabricant propose de nombreuses autres cartes pouvant s'interfacer facilement avec la *Pob-Eye* dans le but de fabriquer un robot complet. Dans la même lignée, la *NXTCam-v2* (voir figure 3.40) est une caméra prévue pour la gamme *NXT* de *Legos* programmables. Cette caméra se configure au préalable sur un ordinateur pour chercher des lignes ou des objets de couleurs (jusqu'à 8 différents), puis se connecte au *Lego* afin de donner les positions correspondantes. Elle est donc moins souple (car non reprogrammable) que les *CMUCam3* et *Pob-Eye*, mais le traitement s'effectue à 30 images par seconde avec une résolution de 88x144 pixels. Si elle est originalement prévue pour fonctionner avec des *Lego*, la communication s'effectue par bus compatible *I2C* donc cette caméra peut être utilisée dans d'autres environnements.

Enfin, la dernière solution "clé en main" présentée ici est le *MBS270v1* (voir figure 3.41), fabriqué par *Mobisense Systems*. C'est de loin la solution la plus puissante, mais aussi la moins accessible. La carte est basée sur un microprocesseur *Intel PXA270* à 520 MHz ; elle est équipée de 64 Mo de mémoire *SDRAM* et 32 Mo de mémoire *Flash*. Cette famille de microprocesseur a été largement utilisée dans les terminaux mobiles de types *PDA* (*Personal Digital Assitant*). Le *PXA270* a la particularité de posséder un port *Quick Capture* qui permet de l'interfacer avec une

Figure 3.39 – La *PobEye*. Source : site *POB-Technology* [idte].

Figure 3.40 – La *NXTCam-v2*. Source : site *Mindsensors* [iti].

caméra numérique. Cela donne accès à l'image par *DMA* (*Direct Memory Access*), c'est à dire sans consommer de ressources processeur. Ce bus fonctionne jusqu'à 25 MHz en 8 bits, soit un débit de 200 Mbits/s. Un capteur *Omnivision OV7620* (640x480 pixels, 30 images par seconde) est ainsi connecté au *Quick Capture*. La carte tourne sous *Linux* avec un noyau 2.4. Utiliser cette carte nécessite donc de bonnes connaissances en *Linux* afin d'avoir une chaine de développement croisée fonctionnelle sur un ordinateur hôte et d'être capable de recompiler un noyau pour y ajouter de nouvelles fonctionnalités (périphériques *USB*, etc). Cependant une telle carte est non seulement capable d'effectuer les traitements vidéos, mais elle pourrait aussi servir pour la loi de commande du drone. Le *PXA270* possède en effet des sorties *PWM* pour piloter les moteurs et des ports de communication pour le relier avec une centrale inertielle ou un modem par exemple. Le *MBS270* est la carte qui a été retenue pour effectuer les calculs de vision embarqué dans le cadre de cette thèse. Ainsi, les algorithmes présentés aux sections 5.3.2 et 5.3.3 peuvent fonctionner à près de 30 Hz. Il est à noter que le *MBS270* n'a été utilisé que pour les calculs de vision, la loi de commande étant assurée par l'architecture présentée dans la section 3.7 ; le *MBS270* étant alors connecté sur le port *SPI* disponible. Cela permet de garder une architecture simple et à bas coût pour la stabilisation du drone ; toutes les applications ne nécessitant d'ailleurs pas un système de vision. Notons enfin que *Mobisense Systems* propose depuis peu le *MBS270v2*. Cette version est basée sur le même microprocesseur, cependant elle est plus petite et plus légère. Le capteur caméra est plus rapide (640x480 pixels, 60 images par seconde), et il est possible de connecter jusqu'à quatre capteurs caméra. Toutefois, une seule image peut être acquise à la fois. La distribution *Linux* intègre maintenant un noyau 2.6.

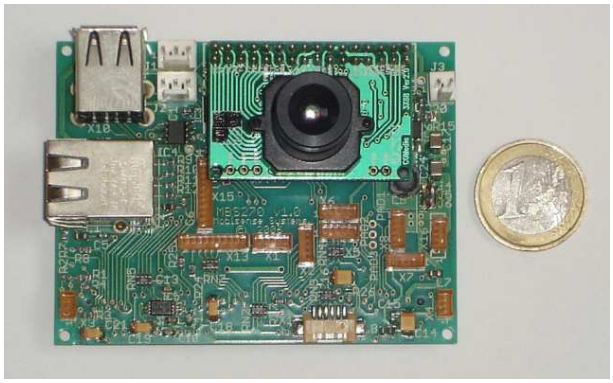


Figure 3.41 – Le *MBS270v1*. Source : site *Mobisense Systems* [idtd].

3.9.2 Autres solutions embarquées

D'autres solutions non "clé en main" existent aussi, c'est à dire ne possédant pas de caméra mais pouvant en accueillir une. Il faut alors distinguer deux types de systèmes : ceux possédant une interface dédiée pour la capture d'images (type *Quick Capture* vu précédemment) et ceux n'en possédant pas, auquel il faudra relier une caméra via un port *USB*.

Dans le premier cas, on retrouve ainsi des solutions basées sur le même microprocesseur que le *MBS270*, ou des versions plus récentes (*PXA310* et *PXA320*). C'est le cas par exemple des modules *Colibri* de *Toradex* (voir figure 3.42). Ceux-ci comportent le microprocesseur (*PXA270*, *310* ou *320*) et la mémoire. Ils doivent alors se connecter à une carte faite par l'utilisateur grâce à un connecteur *SoDIMM*. Afin de l'utiliser dans des applications de vision il faut donc faire une carte accueillant le module et la caméra. Par ailleurs, le module est fourni avec une distribution *Linux* pré-installée mais le driver pour la caméra est à programmer. Notons que le *MBS270* présenté précédemment utilise en fait un module *Colibri*. La gamme *Gumstix Verdex* (voir figure 3.43), utilise aussi un processeur *PXA270* ; l'avantage de cette carte est son encombrement qui est très réduit. Plusieurs connecteurs permettent de brancher des cartes d'extensions pour le *Gumstix*. Aucune des cartes d'extensions proposées par le constructeur ne permet actuellement de brancher une caméra, mais un des connecteurs véhicule les signaux nécessaires. C'est donc à l'utilisateur de se créer une carte d'extension pour caméra et d'écrire le driver adéquat.

Récemment, en août 2008, est apparue une carte de développement à base d'*OMAP 3530* : la *Beagle Board* (voir figure 3.44). L'*OMAP 3530* est un pro-

Figure 3.42 – Le module *Colibri*.
Source : site *Toradex* [itq].

Figure 3.43 – Le *Gumstix Verdex*.
Source : site *Gumstix* [ite].

cesseur construit par *Texas Instrument*, il a la particularité d'avoir un corps *ARM Cortex A8* à 600 MHz (et même 720 MHz pour sa dernière version), un corps *DSP TMS320C64x+* à 430 MHz ainsi qu'un accélérateur graphique *POWERVR SGX*. Ce processeur prévu pour des applications de téléphonie mobile, ou de terminaux multimédia mobiles semble donc très intéressant car il comporte un corps *DSP* spécialisé dans le traitement vidéo. Par ailleurs il possède une interface vidéo dédiée, l'*ISP* (*Image Signal Processing*). Ce bus de données peut fonctionner jusqu'à 130 MHz en 8 bits, c'est à dire un débit de 1.04 Gbits/s. En comparaison le port *USB* ne permet qu'un débit maximum de 480 Mbits/s. La *Beagle Board* a ainsi été la première carte de développement *low cost* (149 $) construite autour de l'*OMAP 3530*. Elle a rapidement suscité un grand intérêt dans la communauté *Linux* embarqué, et de nombreuses autres cartes similaires sont apparues. Le principal inconvénient de la *Beagle Board* est qu'elle ne propose pas le port *ISP*. Toutes les entrées sorties de l'*OMAP 3530* ne sont en effet pas présentes sur la carte, et les concepteurs ont fait le choix de ne pas rendre disponible l'*ISP* pour une raison de coût semble t'il (la *Beagle Board* se veut d'être vendue à moins de 150 $). Les concepteurs ont tout de même annoncé officieusement que l'*ISP* serait disponible sur une future version de la *Beagle Board*, en mai 2010. Cependant, certaines des cartes similaires intègrent des connecteurs pour l'*ISP*. C'est le cas par exemple des *Gumstix Overo Water* et *Fire* (voir figure 3.45), mais aussi du *DevKit8000* (voir figure 3.46) vendu par *Embest*. La présence d'un connecteur pour l'*ISP* ne signifie évidement pas que la caméra sera supportée nativement, il faut bien sur faire le driver approprié. Beaucoup de projets sur *internet* visent à utiliser une caméra sur une plateforme *OMAP 3530* et à porter les librairies *OpenCV* sur cette architecture, en tirant parti des instructions du corps *DSP*. Cependant la plupart de ces projets utilisent une *WebCam* par simplicité, ce qui n'est pas la meilleure solution pour les raisons citées précédemment. Le projet *PixHawk* ([idpt]) du *Computer Vision and Geometry Lab* de l'université de Zurich est plus ambitieux et veut utiliser le port *ISP*

des *Gumstix Overo Fire* pour sa plateforme de type coaxiale.

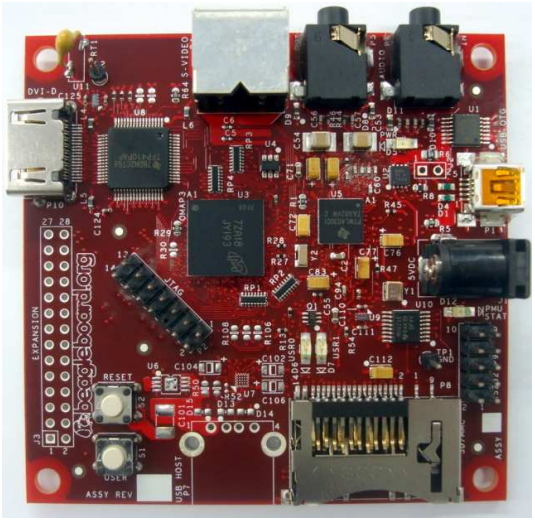


Figure 3.44 – La *Beagle Board*. Source : site *Beagle Board* [itb].

Figure 3.45 – Le *Gumstix Overo Water*. Source : site *Gumstix* [ite].

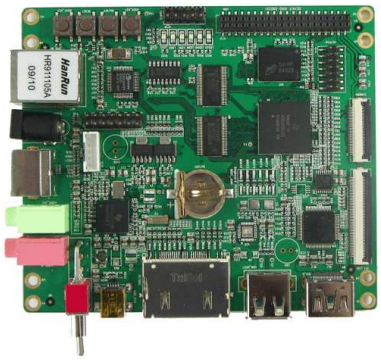


Figure 3.46 – Le *DevKit8000*. Source : site *Embest* [itd].

Les solutions ne comprenant pas d'interface dédiée pour une caméra sont en général basées sur des architectures *x86* ou *x86-64* équipant déjà la plupart des ordinateurs, ce qui facilite grandement la portabilité des programmes informatiques. Le format standard le plus petit actuellement est le *Pico-ITX*[3], lancé par *Via* en 2007. Ainsi, la toute nouvelle[4] carte *Epia-P820* (voir figure 3.47) de *Via* intègre un

3. *Pico-ITX* : 100x72 mm, *Nano-ITX* : 120x120 mm, *Mini-ITX* : 170x170 mm.
4. Cette carte a été présentée le 5 janvier 2010.

processeur *Nano U2500* 64 bits à 1.2 GHz ; celui-ci étant spécialement conçu pour les applications embarquées et ne nécessite pas de refroidissement actif (un radiateur suffit). La carte possède un emplacement *SODIMM* pouvant accueillir jusqu'à 2 Go de mémoire *RAM*, ainsi que différents connecteurs dont l'*USB*. Par contre, aucune mémoire *flash* n'est installée ; il faut donc connecter un périphérique *IDE*, *SATA* ou *USB* pour stocker le système d'exploitation.

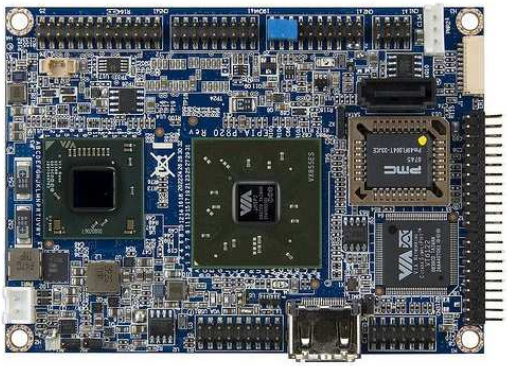


Figure 3.47 – La carte *Epia-P820*. Source : site *Via* [its].

Dans le domaine des cartes *Pico-ITX*, le principal concurrent de *Via* est *Intel* avec ces processeurs *Atom*. Ceux-ci sont devenus très populaires car ils équipent la quasi totalité des *netbooks*, les rendant très compétitifs niveau prix. Ainsi, le module *Robin* (voir figure 3.48) de *Toradex* comprend un processeur *Atom Z530* à 1.6 GHz, 512 Mo de mémoire *RAM* et 2 Go de mémoire *flash*. Ce module est prévu pour se connecter sur une carte *Daisy* (voir figure 3.49) afin d'avoir accès aux différents connecteurs, notamment de type *USB*. Les versions *Zxxx* des processeurs *Atom* étant conçues pour les applications embarquées, elle ne nécessitent donc pas non plus de refroidissement actif.

En terme de performances, *Via* annonce que ses processeurs *Nano* sont 20 % plus rapides que les *Atom* d'*Intel* à fréquences égales. Cependant la carte *Epia-P820* embarque un processeur moins rapide que le module *Robin*, rendant les performances des deux solutions sans doutes très comparables. Chez les deux fabricants, des modèles de processeurs plus rapides existent (*Nano* série 3000, ou *Atom* double cœur) mais ceux-ci ne sont pas encore disponibles sur des cartes de type *Pico-ITX*. La table 3.2 compare les différentes solutions évoquées, en terme de taille, poids et puissance consommée (en charge). Il apparait clairement que les solutions *ARM* sont beaucoup plus économes en énergie mais aussi plus légère, notamment par l'absence de radiateur de refroidissement. La carte *Epia-P820* est par ailleurs de loin la plus consommatrice.

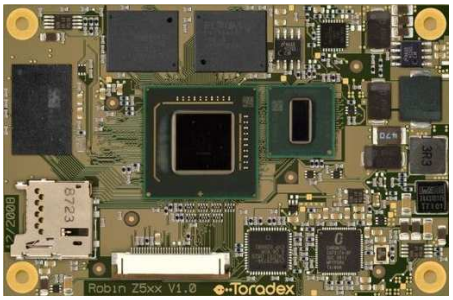


Figure 3.48 – Le module *Robin*. Source : site *Toradex* [itq].

Figure 3.49 – La carte *Daisy*. Source : site *Toradex* [itq].

Carte	Dimensions	Poids	Consommation	Prix
MBS270v1	62x80 mm	62 g	1 W	250 €
Gumstix Overo Summit Pack	39x80 mm	23 g	1 W	259 $
Epia-P820+radiateur	100x72 mm	123 g	15 W	192 €
Robin+*Daisy*+radiateur	100x72 mm	210 g	6.5 W	279 €

Table 3.2 – Comparatif des diverses solutions présentées. Données constructeur.

3.9.3 Solutions déportées

Il a été vu jusqu'ici les solutions embarquables dans le drone les plus puissantes du moment. Cependant, même en choisissant la solution la plus performante, il est difficile de savoir si les algorithmes préalablement développés sur un ordinateur fonctionneront suffisamment rapidement. Ainsi les algorithmes peuvent tourner à une fréquence trop faible pour stabiliser le drone. Dans ce cas, il faut déporter les calculs et les effectuer sur un ordinateur au sol. Cette solution permet d'utiliser des algorithmes sans trop se soucier de la puissance de calcul. Cependant elle apporte d'autres problèmes. En effet, l'utilisation de caméras filaires n'est pas envisageable car le drone aurait alors un rayon d'action très limité et les câbles déstabiliseraient trop l'engin. Il faut alors envoyer la vidéo par une liaison sans fils au sol. Cette liaison doit donc être de bonne qualité pour qu'il y ait un minimum de bruit sur l'image à traiter. Or une grande partie des systèmes de vidéo sans fils utilisent des fréquences en 2.4 GHz, déjà occupées par les modems radios (voir section 3.6), ou par le *WiFi* de l'université. Il faut donc choisir un système de transmission radio à 1.2 GHz ou 5.8 GHz pour éviter au maximum les perturbations. Par ailleurs, si l'on veut utiliser deux caméras sans fils dans le drone (voir section 5.3.1) ou faire voler

deux drones ayant des caméras sans fils, il faudra encore une fois faire attention aux interférences. Ensuite, les données calculées au sol doivent être renvoyées au drones, via le modem radio. Comme vu précédemment (section 3.6), cette liaison peut comporter des erreurs de transmission ou du retard.

Toutefois, l'utilisation d'un ordinateur fixe comme unité de calcul offre beaucoup plus de possibilités. Les processeurs (ou *CPU*, *Central Processing Unit*) sont en effet de plus en plus performants. Leur puissance a tendance à stagner, cependant les constructeurs déjouent ce problème en intégrant plusieurs cœurs à un même *CPU*. Les algorithmes doivent donc être pensés de façon à effectuer des calculs parallèles. De même, depuis quelques années, la puissance des cartes graphiques et de leur processeur (ou *GPU*, *Graphics Processing Unit*) a considérablement augmenté. L'architecture de ces derniers permet des calculs hautement parallélisable ; un algorithme bien optimisé pour cette architecture sera alors calculé beaucoup plus rapidement que sur un *CPU* classique. Ainsi, l'algorithme *FOLKI* développé par [BC05] permet de calculer le flux optique par une technique de recalage de fenêtre de type Lucas et Kanade (voir section 5.2.1) itératif et multirésolution. Une autre caractéristique importante de *FOLKI* est sa structure hautement parallèle qui profite pleinement des architectures de calcul modernes. Le temps de traitement dépend alors de la taille de la fenêtre choisie et du nombre d'itérations effectuées à chaque niveau. Cependant, [CPB+09] montre que pour des réglages correspondants à une bonne estimation du flux optique, le temps de traitement de l'algorithme est de 32 ms pour des images en *full HD* c'est à dire en 1920x1080 pixels. L'algorithme est donc plus rapide que le flux vidéo et démontre l'intérêt d'utiliser des architectures massivement parallèles, car sur des images de haute résolution (2048x4096 pixels) le traitement sur *GPU* est 100 fois plus rapide que sur *CPU*.

3.9.4 Conclusions

Les solutions embarquées seraient donc idéales, car elles évitent les problèmes de communications sans fils et de bruit. Il n'y alors pas de problèmes de transmission sur de longues distances ou de possibilités de brouillage des signaux par un éventuel ennemi. Par ailleurs, les technologies évoluent sans cesse et la puissance des solutions embarquées est de plus en plus importante ; laissant supposer que les algorithmes de vision lourds seront bientôt possible à traiter sur des cartes de faible dimension et consommation.

Parmi les solutions embarquées présentées, celles ayant une interface dédiée pour la caméra sont intéressantes pour notre application. En effet en l'absence de cette interface, la caméra (généralement une *WebCam*) doit être branchée sur un

port *USB*. Cependant, le port *USB* peut consommer des ressources processeur et surtout la plupart des *WebCam* délivrent des images compressées, car elles sont destinées à des applications internet où les débits sont limités. Ainsi le processeur devra préalablement décompresser les images avant de pouvoir les traiter. Cette opération est transparente pour l'utilisateur car c'est le *driver* qui s'en charge, mais elle reste coûteuse en ressources. Les solutions actuelles de type *x86* ne possèdent pas cette interface dédiée mais peuvent compenser cette lacune grâce à une puissance processeur relativement élevée. Cependant la consommation électrique est souvent importante. L'architecture *OMAP 3530* semble donc allier de nombreux avantages grâce à son interface *ISP* et à son *DSP* pouvant prendre en charge le traitement d'image, en maintenant une consommation très réduite. Cependant cette architecture n'a pu être utilisée dans le cadre de cette thèse car elle n'est apparue que récemment et qu'elle nécessite encore de nombreux développements (optimisation des algorithmes pour profiter des instructions *DSP*, driver pour la caméra via *ISP*, etc). L'utilisation d'une telle carte fait donc partie des travaux futurs de cette thèse[5], l'objectif étant de porter une bibliothèque de vision telle qu'*OpenCV* sur l'architecture *OMAP 3530* en tirant profit au maximum du *DSP*.

3.10 Informatique

3.10.1 Station sol

Les premiers drones réalisés au laboratoire étaient relativement simples et n'avaient qu'une radiocommande pour les contrôler. Celle-ci permettait en effet de les trimer et éventuellement de les piloter. Cependant cette liaison s'est vite avérée insuffisante. Une liaison par modem a donc été mise au point (voir section 3.6). Cette liaison a d'abord été mise en place pour éviter les problèmes de bruits dans la radiocommande, puis elle a évolué avec la réalisation de la station sol. Celle-ci a été écrite en *Visual Basic* (*VB*) par simplicité, voir une copie d'écran sur la figure 3.50.

La station a été faite de façon à être la plus complète et la plus évolutive possible. Ainsi, elle permet d'afficher à l'aide de graphiques tous les états du drone (s'ils sont disponibles) : commande du *joystick*, angles d'euler, vitesses angulaires, vitesses de rotations des moteurs (ainsi que leur consigne), position *GPS*, vitesses

5. Ces travaux seront effectués à partir de février 2010 dans le cadre d'un stage à l'*Instituto Universitario de Automática e Informática Industrial* de l'Université Polytechnique de Valencia, Espagne ; grâce à une bourse octroyée par le gouvernement espagnol.

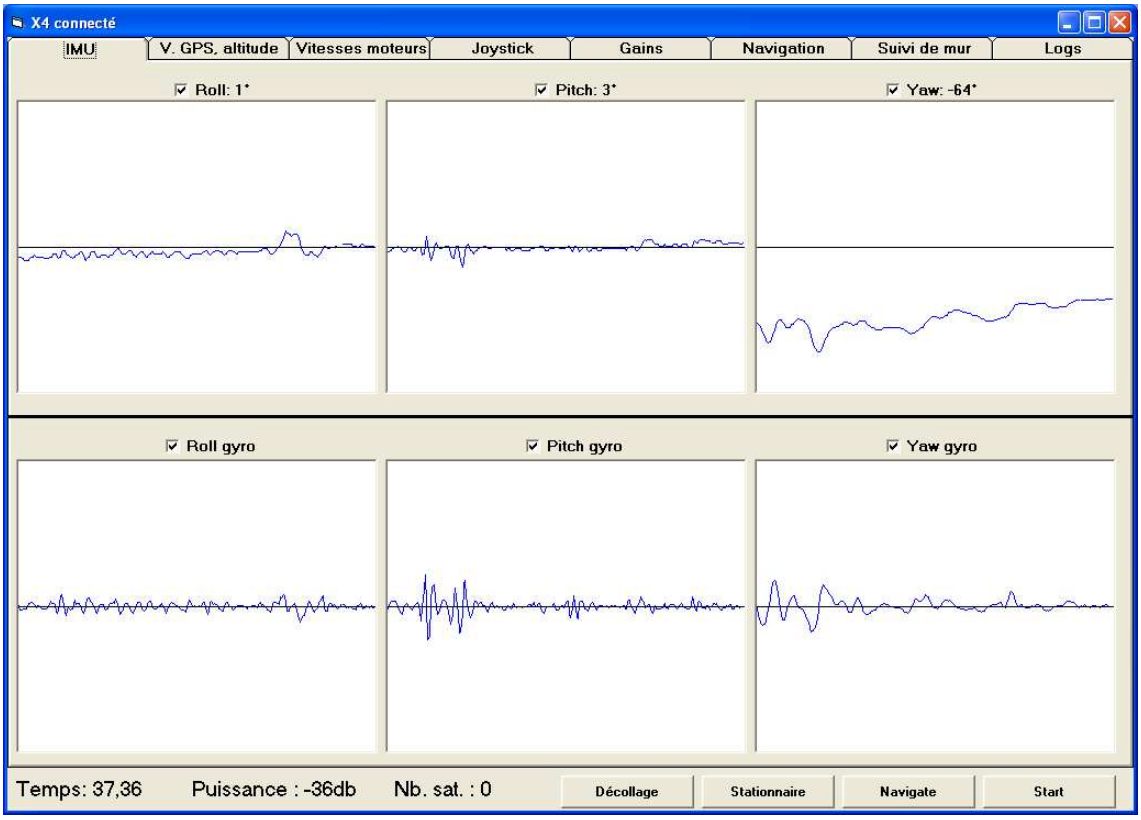


Figure 3.50 – Copie d'écran de la station sol. Les graphes représentent les données inertielles.

de déplacements latérales, altitude, données provenant du système de vision. Le drone envoie aussi des informations supplémentaires dites de *debug* afin de savoir où le microcontrôleur se trouve dans le programme (notamment dans les machines à états). Toutes ces données sont par ailleurs sauvegardées et horodatées dans un fichier lisible par *Matlab* afin de pouvoir redessiner les graphes de chaque essai. De plus, il est possible de choisir sur la station sol quelles données doit envoyer le drone, afin de ne pas récupérer tous les états. En effet, suivant l'essai il n'est pas toujours nécessaire de récupérer tous les *logs*. Cela permet alors d'alléger la communication car comme il a été vu à la section 3.6, si trop de données transitent le risque de collisions de paquets augmente. De même, sur la station sol l'utilisateur peut choisir s'il veut afficher un graphe ou non. Le programme étant écrit en *VB*, l'affichage se révèle relativement gourmand en ressources processeur. Si l'ordinateur doit aussi effectuer d'autres calculs (de vision par exemple) il peut être judicieux d'alléger au maximum la consommation de la station sol ; sachant que les données mêmes si elles ne sont pas affichées seront tout de même sauvegardées.

La station permet aussi de changer à la volée tous les gains des lois de commandes embarquées, et de les sauvegarder dans la mémoire non volatile du microcontrôleur (*flash*). En effet, lors de l'implémentation d'une nouvelle loi de commande, il est toujours nécessaire de régler les gains manuellement en faisant des essais et en constatant le comportement du drone. Le fait de pouvoir les changer instantanément apporte ainsi un grand confort et permet de vraiment ajuster les gains, en essayant différentes valeurs rapidement. Sans la station sol, il fallait auparavant arrêter le drone et reprogrammer le microcontrôleur pour changer les gains ; cette opération étant longue et fastidieuse.

Enfin, la station sol ayant été développée pour le modem *XBee* présenté dans la section 3.6, elle tire profit de son mode *API* (tout comme le programme embarqué dans le drone). Ainsi, elle contrôle chaque trame envoyée et reçue afin de compter le nombre de trames perdues et de calculer le rapport trames perdues sur trames totales aussi bien en émission qu'en réception. Ces deux indicateurs permettent de se rendre compte de la qualité de la communication et d'ajuster les réglages (fréquence d'émission de la station sol et du microcontrôleur, type de données émises) afin d'arriver au meilleur compromis sur le nombre de données effectivement transmises par unité de temps. Par ailleurs, grâce au mode *API*, le logiciel sait exactement quelle trame a été perdue et peut donc la renvoyer s'il s'agit d'un message "prioritaire". Les seuls messages non prioritaires sont les données émises régulièrement : données du *joystick* envoyées par la station sol et *logs* envoyés par le drone. Toutes les autres trames (changement des gains de contrôle, ordre de démarrage/arrêt des moteurs, coordonnées des points de passage *GPS*...) sont considérées comme prioritaires et sont donc renvoyées si la trame n'est pas reçue.

3.10.2 Drone

Le programme du drone a été écrit en *C/C++*. Le compilateur utilisé (*CodeWarrior*) est fourni avec un *debuggeur*, cependant cela nécessite de brancher un câble entre le microcontrôleur et l'ordinateur. Le *debug* ne peut donc pas être fait en vol, ce qui réduit beaucoup son intérêt ; seuls de petits éléments pouvant être testés au sol. Cette limitation empêche donc de développer des programmes trop complexes car il serait impossible de les *debugger* en cas de soucis.

Pour palier à ce problème, le code (sur chaque microcontrôleur) a été découpé en deux parties bien distinctes (voir figure 3.51) : d'une part toutes les fonctions intrinsèquement dépendantes du matériel utilisé, c'est à dire toutes les entrées-sorties des microcontrôleurs, les interruptions et les accès à la mémoire *flash*, d'autre part

toutes les fonctions indépendantes du matériel. Il s'agit ici des algorithmes de contrôle, des fonctions lisant et écrivant dans les *buffers* de communications... Ce découpage permet d'avoir un code facilement portable sur une autre architecture. En effet, dans ce cas il suffit d'adapter le code de la partie dépendante du matériel. Cela permet donc aussi d'exécuter le programme sur un ordinateur, et de faire du *debug* sur la partie indépendante.

La partie dépendante a alors été adaptée pour l'ordinateur et reliée à un modèle du drone (il s'agit de celui présenté dans la section 3.1.3, mais en temps discret) pour pouvoir simuler sa dynamique en fonction des entrées de commandes. Ce modèle calcule donc la position et l'orientation du drone en fonction des consignes des moteurs, et fournit à la partie indépendante les données de la centrale inertielle et du *GPS*. Enfin, la station sol est reliée, via un port série virtuel, à un émulateur de modem *XBee* (lui aussi écrit en *C/C++*). En effet la station sol ne peut pas être reliée directement à la partie dépendante car le *XBee* est configuré en mode *API* (voir partie 3.6) et donc possède son protocole particulier. L'émulateur s'occupe donc de fournir les *ACK* et de transférer les messages d'un port série virtuel à l'autre. Par simplicité, et contrairement au cas du modem réel, aucune donnée n'est perdue par l'émulateur. Cette fonctionnalité n'a pas été jugée nécessaire mais serait facilement implémentable.

Cependant, sur un ordinateur la taille du code est quasi illimitée, alors que la taille maximale du code sur le microcontrôleur choisi est de 128 Ko (mais limitée à 96 Ko par le compilateur). Le simulateur ne permet donc pas de vérifier cela, mais si le code écrit est trop important, le compilateur du microcontrôleur le dira. De même, la vitesse d'exécution du programme n'est pas la même sur ordinateur que sur le microcontrôleur. En effet, la période d'échantillonnage est fixée dans les deux cas à 5 ms (200 Hz), mais l'ordinateur peut effectuer plus de calculs que le microcontrôleur en 5 ms. Ainsi, à chaque fin de boucle le microcontrôleur attend que 5 ms se soient écoulées depuis la dernière itération grâce à un *timer*. Si la boucle a pris plus de 5 ms, un *flag* d'*overflow* est placé sur le *timer* ce qui permet de savoir que les calculs demandés sont trop longs. Un message est alors envoyé à la station sol, pour avertir que la période d'échantillonnage ne peut être tenue. Il faut alors modifier le code pour que les calculs puissent s'effectuer en temps réel.

Afin de tester les algorithmes de vision, une représentation 3D du drone et de son environnement a été faite en *OpenGL*. Ainsi, des caméras virtuelles peuvent être placées dans le drone aux endroits désirés. Les images des caméras sont alors affichées sur l'ordinateur (voir figure 5.38) et leur contenu est placé dans un *buffer* qui sera utilisé par le programme de vision. Dans le cas réel, les calculs de visions peuvent être embarqués ou déportés suivant la complexité de l'algorithme (voir

section 3.9). S'ils sont embarqués, les informations du traitement de l'image sont envoyées au microcontrôleur via un port *SPI* ; sinon les informations sont envoyées par la station sol. Les deux cas ont donc été implémentés sur le simulateur afin d'être le plus proche possible de la réalité. De plus, les programmes de vision travaillent uniquement avec des *buffers* contenant les images, alors qu'un autre programme se charge de remplir le *buffer* depuis la source appropriée (*OpenGL* pour les simulations, *driver Video4Linux* pour le système embarqué, *OpenCV* pour la solution déportée). Cela permet d'utiliser exactement les mêmes programmes en simulation et pour les expériences.

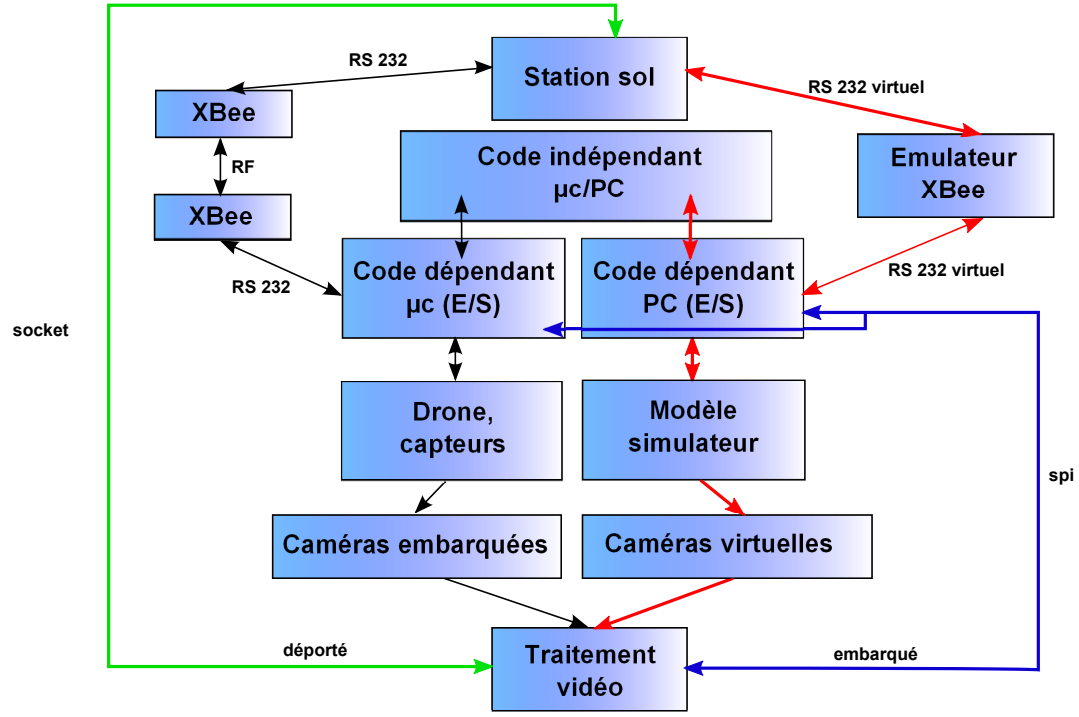


Figure 3.51 – Architecture informatique. Les deux microcontrôleurs sont représentés par un seul bloc par simplicité.

3.11 Conclusion

La plateforme finale est montrée sur la figure 3.52. L'architecture électronique et informatique ont été présentées sur une structure mécanique différente (voir figure 3.6) lors du concours minidrone organisé par l'*Onera* et la *DGA* en 2008-2009. La stabilité en vol a pu être démontrée, notamment lors d'essais dans la soufflerie du banc *B20* de l'*Onera* à Lille, où le drone a volé dans des rafales à 4 m/s. La nouvelle structure offre des performances semblables, en étant plus rigide.

Cependant, le quadrirotor a parfois des difficultés à rester sur une position fixe et a tendance à se déplacer latéralement. Cela est du aux dérives de la centrale, qu'il est pour l'instant difficile de compenser. L'électronique embarquée ayant été pensée de manière évolutive, il sera envisageable de remplacer la centrale inertielle pour effectuer l'estimation de l'orientation à partir des capteurs inertiels.

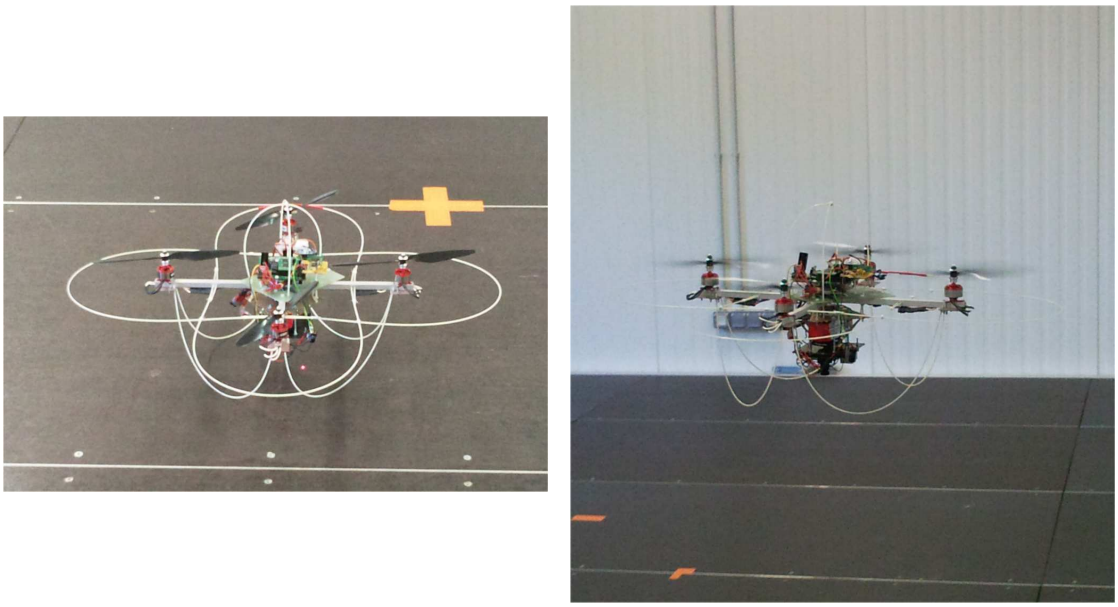


Figure 3.52 – Photos du quadrirotor développé.

Chapitre 4

Lois de commandes

Le but de ce chapitre est de comparer plusieurs lois de commandes, qu'elles soient linéaires ou non linéaires, en les appliquant à un système non linéaire simple, l'avion *PVTOL*[1] (*Planar Vertical Take Off and Landing*). Toutefois ces lois peuvent facilement être transposées à un quadrirotor ; celui-ci étant en fait un modèle augmenté du *PVTOL*. Il sera ensuite fait une généralisation en proposant une loi de commande permettant de stabiliser un système à n intégrateurs en cascade.

4.1 Comparaison de différentes lois de commandes sur une plateforme de type *PVTOL*

Cette section présente dans un premier temps l'état de l'art sur la commande du *PVTOL*. Le modèle dynamique du véhicule est ensuite obtenu par l'approche d'Euler-Lagrange. Puis des lois de commandes linéaires, par fonctions de saturations emboitées et séparées ainsi que par *backstepping* seront présentées. Enfin ces différentes lois seront comparées en simulations et sur des expériences en temps réel que nous avons fait dans [SCGL07].

1. Par abus de langage, l'avion *PVTOL* sera appelé simplement *PVTOL* dans le reste de ce document.

4.1.1 État de l'art

Le *PVTOL* est basé sur un modèle mathématique simple d'un drone évoluant dans un plan vertical, et ayant un nombre minimal d'états et d'entrées de commandes. De plus, c'est un système sous-actionné car il ne possède que deux entrées de commandes (ses deux moteurs) pour trois degrés de liberté. Il conserve cependant les principales caractéristiques devant être considérées lors de l'élaboration de lois de commandes pour un drone. Il suscite donc un grand intérêt dans la communauté de l'automatique pour ses applications et son comportement non linéaire. En particulier, le système est à déphasage non minimum du fait du couplage ε entre le moment de roulis et l'accélération latérale (voir section 4.1.2). De nombreuses méthodes de contrôle ont ainsi été proposées au fil des années. L'étude du *PVTOL* est intéressante dans le cadre de cette thèse car il représente un modèle simplifié du quadrirotor. En effet, ce dernier peut être vu comme deux *PVTOL* orthogonaux (dans le cas du modèle simplifié, voir section 3.1.5) ; les travaux sur le *PVTOL* sont ainsi transposables au quadrirotor.

Les premières approches de contrôle sont basées sur les résultats de [HSM92]. Ces travaux proposent une linéarisation approximée de la réponse entrée-sortie, étant donné que le contrôle par retour d'état n'est pas possible sur la représentation linéarisée d'un système à déphasage non minimal. Une extension de ces résultats a ensuite été proposée par [MDP96]. Leur idée est de trouver une sortie plate au système et de scinder le problème de suivi de trajectoire en deux étapes. Les auteurs construisent donc d'abord un suivi de trajectoire basé sur la linéarisation exacte en utilisant la sortie plate, puis ils utilisent un générateur de trajectoire pour alimenter le contrôleur. Le suivi est ainsi contrôlé en utilisant la sortie plate. Contrairement aux travaux précédents de [HSM92], ce procédé permet de faire du suivi sur un système à déphasage non minimal. Le terme de couplage ε est par ailleurs pris en compte dans ces travaux.

Peu après, un théorème non linéaire dit du petit gain a été proposé par [Tee92a] et appliqué au *PVTOL*, avec une perturbation sur son entrée. La stabilité d'un contrôleur basé sur des saturations emboitées est alors démontrée. Une extension de cet algorithme de contrôle a d'ailleurs ensuite été proposée par [CLD05]. L'efficacité du contrôleur y est montrée avec des expériences en temps réel.

En utilisant une analyse de Lyapunov et en ajoutant un intégrateur dans la boucle de contrôleur, [MP96] a stabilisé le *PVTOL* en utilisant seulement la mesure de la position.

Une technique de *forwarding* a été utilisée par [FL02], afin de proposer une loi de commande pour le *PVTOL*. Une fonction de Lyapunov assure alors la stabilité asymptotique. D'autres techniques basées sur la linéarisation sont aussi proposées

par les mêmes auteurs dans [FL01].

Un régulateur basé sur la rétro-alimentation dynamique de l'erreur a été proposé par [MIS02]. L'approche est basée sur un modèle interne et permet l'atterrissage automatique sur une plateforme oscillante. Le contrôleur est alors robuste aux incertitudes sur les paramètres du système. Le mouvement de la plateforme est en effet modélisé par une somme finie de fonctions sinusoïdales d'amplitudes, phases et fréquences inconnues.

Les travaux de [FLC02], [FZL02], [ZFL02], [ZFL03] et [LCD04] portent sur des stratégies de contrôle prenant en compte un borne arbitraire sur les entrées de commande. La stabilité asymptotique globale est obtenue avec des fonctions de saturations emboitées.

Plus récemment, une loi non linéaire par retour d'état a été proposée par [WC07]. Le schéma de contrôle distribue le système en une structure en cascade et la stabilité globale est alors prouvée. De même, [YWW07] parvient à la stabilité globale grâce à une technique de saturations. Celle-ci est appliquée au système du *PVTOL* représenté par une chaine d'intégrateurs ayant des perturbations non linéaires. Puis, une approche basée sur une prédiction non linéaire a été développée par [CM08]. Le contrôle est basé sur la linéarisation partielle du retour d'état et sur la génération de trajectoires optimales afin d'améliorer la stabilité du système. La robustesse par rapport aux incertitudes du modèle a été montrée seulement sur des simulations. Le suivi de trajectoires et de chemins a aussi été développé. Ainsi, un suivi de trajectoire en boucle ouverte, avec une dynamique interne bornée, utilisant une approche par carte de Poincaré a été présentée par [CT07]. Un algorithme de suivi de chemin a été proposé par [NCMT08], permettant au centre de gravité du *PVTOL* de parcourir un cercle dans une direction donnée.

4.1.2 Préliminaire : Modèle dynamique du *PVTOL*

Le *PVTOL* est représenté de façon schématique sur la figure 4.1. Chacun des moteurs produit une force f_i ; la somme de ces deux forces donne la poussée u et la différence produit un couple τ_ϕ :

$$u = f_1 + f_2 \tag{4.1a}$$
$$\tau_\phi = l(f_1 - f_2) \tag{4.1b}$$

l étant la distance entre le moteur et le centre de gravité du véhicule ; ce dernier étant supposé symétrique par rapport à l'axe vertical. Ces deux entrées de commande permettent ainsi de contrôler la position (x, y) du centre de gravité ainsi que l'angle ϕ du véhicule. Selon (3.8), le lagrangien du *PVTOL* s'écrit :

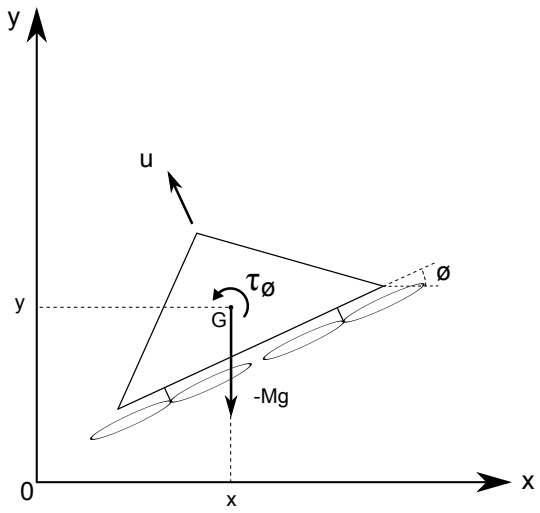


Figure 4.1 – Schéma du *PVTOL*.

$$L = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}M\dot{y}^2 + \frac{1}{2}J\dot{\phi}^2 - Mgy \tag{4.2}$$

En introduisant le paramètre ε représentant le couplage entre le moment de roulis et l'accélération latérale (voir [HSM92]), la force F et le couple τ appliqués au *PVTOL* s'expriment dans le repère fixe par :

$$F = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} \varepsilon\tau_\phi \\ u \end{pmatrix} \tag{4.3a}$$

$$\tau = \tau_\phi \tag{4.3b}$$

où M est la masse de l'engin, J son moment d'inertie et g la norme du vecteur gravité. Notons que ε représente le fait que les forces produites par les moteurs ne sont pas exactement colinéaires. En appliquant le théorème d'Euler-Lagrange (voir (3.13)), on obtient :

$$M\ddot{x} = -u\sin\phi + \varepsilon\tau_\phi\cos\phi \tag{4.4a}$$

$$M\ddot{y} = u\cos\phi + \varepsilon\tau_\phi\sin\phi - Mg \tag{4.4b}$$

$$J\ddot{\phi} = \tau_\phi \tag{4.4c}$$

Un changement de variable a été proposé par [OS99] afin d'obtenir une représentation où le terme ε n'apparait pas explicitement. Ainsi, en posant :

$$\overline{x} = x - \varepsilon\sin\phi \tag{4.5a}$$

$$\overline{y} = y + \varepsilon(\cos\phi - 1) \tag{4.5b}$$

le système (4.4) s'écrit alors :

$$M\ddot{\overline{x}} = -\overline{u}\sin\phi \tag{4.6a}$$
$$M\ddot{\overline{y}} = \overline{u}\cos\phi - Mg \tag{4.6b}$$
$$J\ddot{\phi} = \tau_\phi \tag{4.6c}$$

où $\overline{u} = u\varepsilon\dot{\phi}^2$. Par la suite nous étudierons donc la dynamique simplifiée du $PVTOL$, en considérant soit que le paramètre ε est parfaitement connu ou soit qu'il est négligeable :

$$M\ddot{x} = -u\sin\phi \tag{4.7a}$$
$$M\ddot{y} = u\cos\phi - Mg \tag{4.7b}$$
$$J\ddot{\phi} = \tau_\phi \tag{4.7c}$$

4.1.3 Préliminaire : Stabilisation de l'altitude

Pour contrôler le $PVTOL$, nous proposons de stabiliser dans un premier temps son altitude en la forçant à satisfaire la dynamique d'un système linéaire :

$$u = \frac{-a_1\dot{y} - a_2(y - y_d) + Mg}{\cos\phi} \tag{4.8}$$

où a_1, a_2 sont des constantes positives et y_d l'altitude désirée. Par ailleurs, il est supposé dans tout ce chapitre que $|\phi| < \pi/2$ et donc que $\phi \neq \pi/2$. Cela traduit un fonctionnement normal du $PVTOL$, car si $\phi = \pi/2$ l'engin n'a plus de portance. En introduisant (4.8) dans (4.7) il s'ensuit :

$$M\ddot{x} = -\Big(- a_1\dot{y} - a_2(y - y_d) + Mg\Big)\tan\phi \tag{4.9a}$$
$$M\ddot{y} = -a_1\dot{y} - a_2(y - y_d) \tag{4.9b}$$
$$J\ddot{\phi} = \tau_\phi \tag{4.9c}$$

Ainsi, après un temps T suffisamment grand et d'après (4.9b), y et $\dot{y}$ sont arbitrairement petits, et (4.9a) se réduit à :

$$\ddot{x} = -g\tan\phi \tag{4.10}$$

L'étude comparative des lois de commandes sera ainsi faite pour le sous-système (x, ϕ).

4.1.4 Loi de commande linéaire

En linéarisant le système (x, ϕ) i.e. (4.9c)-(4.10), autour de l'origine, on obtient :

$$\ddot{x} = -g\phi \tag{4.11a}$$

$$J\ddot{\phi} = \tau_{L\phi} \tag{4.11b}$$

où $\tau_{L\phi}$ représente l'entrée de la commande linéaire et sera précisée par la suite. Avec une notation évidente :

$$\dot{x}_1 = x_2 \tag{4.12a}$$

$$\dot{x}_2 = -g\phi_1 \tag{4.12b}$$

$$\dot{\phi}_1 = \phi_2 \tag{4.12c}$$

$$\dot{\phi}_2 = \frac{\tau_{L\phi}}{J} \tag{4.12d}$$

En posant $X = [x_1, x_2, \phi_1, \phi_2]^T$, le système précédent s'écrit alors :

$$\dot{X} = AX + B\tau_{L\phi} \tag{4.13}$$

avec :

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1/J \end{bmatrix} \tag{4.14}$$

En posant la loi de commande suivante :

$$\tau_{L\phi} = -KX \tag{4.15}$$

alors,

$$\dot{X} = (A - BK)X = \bar{A}X \tag{4.16}$$

où $\bar{A} = (A - BK)$. La matrice K est alors obtenue par placement de pôles, afin de la rendre Hurwitz.

4.1.5 Loi de commande par fonctions de saturations emboîtées

Dans cette section, nous rappelons une loi de commande τ_{NS_ϕ} proposée par [CLD05] pour stabiliser le sous système (4.9c)-(4.10). Notons que la loi de commande est basée sur des saturations emboîtées et que l'analyse de la convergence

est faite par la méthode de Lyapunov. Cette méthode utilise le modèle non linéaire du *PVTOL* et permet d'imposer une borne à $|\phi|$ après un certain temps.

$$\tau_{NS_\phi} = -J\sigma_{b_1}\left[\dot{\phi} + \sigma_{b_2}\left(\phi + \dot{\phi} + \sigma_{b_3}\Big(2\phi + \dot{\phi} - \frac{\dot{x}}{g} + \sigma_{b_4}(\dot{\phi} + 3\phi - 3\frac{\dot{x}}{g} - \frac{x}{g})\Big)\right)\right] \tag{4.17}$$

où σ_{b_i} est une fonction de saturation telle que (voir aussi la figure 4.2) :

$$\sigma_{b_i}(s) = b_i, \text{ si } : s > b_i \tag{4.18a}$$
$$\sigma_{b_i}(s) = s, \text{ si } : -b_i \leq s \leq b_i \tag{4.18b}$$
$$\sigma_{b_i}(s) = -b_i, \text{ si } : s < -b_i \tag{4.18c}$$

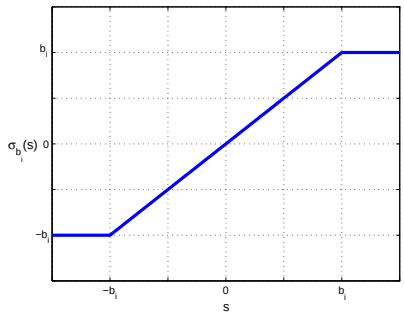


Figure 4.2 – Fonction de saturation.

Par ailleurs les bornes b_i des fonctions de saturations doivent satisfaire les conditions suivantes :

$$b_1 \geq 4b_3 + 3\delta \tag{4.19a}$$
$$b_2 \geq 2b_3 + \delta \tag{4.19b}$$
$$1 \geq b_3 + 2\delta \tag{4.19c}$$
$$b_3 \geq 7(b_3 + 2\delta)^2 + 3\delta \tag{4.19d}$$
$$b_4 \geq 3(b_3 + 2\delta)^2 + \delta \tag{4.19e}$$

où δ est une constante arbitrairement petite. Il faut donc choisir dans un premier temps b_3 et δ afin de satisfaire (4.19c) et (4.19d), puis les autres paramètres sont choisis en fonction de b_3 et δ. Notons que la condition (4.19c) permet d'imposer une borne de 1 à $|\phi|$.

4.1.6 Loi de commande par fonctions de saturations séparées

Récemment, dans [SCE+07] une loi de commande utilisant des fonctions de saturations pour stabiliser le système (4.9c)-(4.10) du *PVTOL* a été proposée. L'algorithme de contrôle est là aussi obtenu grâce à une analyse de Lyapunov. Cependant, la loi de commande ne possède plus de saturations emboîtées. Par contre chaque saturation joue sur une combinaison linéaire d'états du système. Ainsi, d'après [SCE+07] la loi est donnée par :

$$\tau_{S_\phi} = -J\Big(\sigma_{b_4}(k_4 z_4) + \sigma_{b_3}(k_3 z_3) + \sigma_{b_2}(k_2 z_2) + \sigma_{b_1}(k_1 z_1)\Big) \tag{4.20}$$

avec :

$$z_4 = \dot{\phi} \tag{4.21a}$$
$$z_3 = k_4\phi + \dot{\phi} \tag{4.21b}$$
$$z_2 = z_3 + k_3\dot{\phi} - \frac{k_3 k_4}{g}\dot{x} \tag{4.21c}$$
$$z_1 = z_2 + k_2\phi - \frac{k_2 k_3 k_4}{g}x - \frac{k_2(k_3+k_4)}{g}\dot{x} \tag{4.21d}$$

Par ailleurs les bornes b_i des fonctions de saturations doivent satisfaire les conditions suivantes :

$$b_1 > k_5\Big(\frac{b_1+b_2}{k_3 k_4} + \delta\Big)^2 \tag{4.22a}$$
$$1 \geq \frac{b_1+b_2}{k_3 k_4} + \delta \tag{4.22b}$$
$$b_2 > k_3 k_4\Big(\frac{b_1+b_2}{k_3 k_4} + \delta\Big)^2 \tag{4.22c}$$
$$b_3 > b_1 + b_2 \tag{4.22d}$$
$$b_4 > b_1 + b_2 + b_3 \tag{4.22e}$$
$$\frac{k_3 k_4}{2k_5 + k_3 k_4} > \frac{b_1+b_2}{k_3 k_4} + \delta \tag{4.22f}$$

où $k_5 = k_2k_3 + k_2k_4 + k_3k_4$ et δ est une constante arbitrairement petite. Notons que la condition (4.22b) permet d'imposer une borne de 1 à $|\phi|$.

4.1.7 Loi de commande par *backstepping*

Les deux lois de commandes exposées précédemment utilisent une technique récursive dite de *forwarding* (développée par [Tee92a]). La loi présentée dans cette

section est basée au contraire sur une technique récursive dite de *backstepping* (développée par [SKS90]). De manière simplifiée, les techniques de *forwarding* s'appliquent à des systèmes de type *feedforward* (ou triangulaires supérieurs) :

$$\dot{x}_1 = f_1(x_2, x_3, \ldots, x_n, u) \tag{4.23a}$$
$$\dot{x}_2 = f_2(x_3, \ldots, x_n, u) \tag{4.23b}$$
$$\vdots$$
$$\dot{x}_n = f_n(u) \tag{4.23c}$$

alors que les techniques de *backstepping* s'appliquent à des systèmes de type *feedback* (ou triangulaires inférieurs) :

$$\dot{x}_1 = f_1(x_1, x_2) \tag{4.24a}$$
$$\dot{x}_2 = f_2(x_1, x_2, x_3) \tag{4.24b}$$
$$\vdots$$
$$\dot{x}_n = f_n(x_1, x_2, \ldots, x_n, u) \tag{4.24c}$$

Lors de la conception de la loi, les techniques de *forwarding* partent de l'entrée de commande u pour aller jusqu'à x_1. Les techniques de *backstepping* utilisent la procédure inverse, en partant de l'équation la plus éloignée de l'entrée (soit x_1) pour remonter jusqu'à l'entrée. Plus de détails sur ces méthodes récursives peuvent se trouver dans l'ouvrage de [SJK97] par exemple.

La réécriture du système (4.9c)-(4.10) montre qu'il est adapté aux deux techniques :

$$\dot{x}_1 = x_2 \tag{4.25a}$$
$$\dot{x}_2 = -g \tan \phi_1 \tag{4.25b}$$
$$\dot{\phi}_1 = \phi_2 \tag{4.25c}$$
$$\dot{\phi}_2 = \frac{\tau_{B\phi}}{J} \tag{4.25d}$$

où $\tau_{B\phi}$ représente l'entrée de la commande par *backstepping*. Celle-ci est obtenue après les quatre étapes détaillées par la suite.

1^{re} étape

Nous commençons par stabiliser le système suivant :

$$\dot{x}_1 = x_2 \tag{4.26a}$$
$$y_1 = x_1 \tag{4.26b}$$

où x_2 représente l'entrée et y_1 la sortie. Proposons la fonction définie positive suivante :

$$V_1 = \frac{x_1^2}{2} \tag{4.27}$$

alors :

$$\dot{V}_1 = x_1\dot{x}_1 = x_1x_2 \tag{4.28}$$

proposons aussi l'entrée virtuelle $\alpha_1 = (x_2)^v$ telle que :

$$\alpha_1 = -k_1x_1 \tag{4.29}$$

où k_1 est une constante strictement positive. Il s'ensuit :

$$\dot{V}_1 = -k_1x_1^2 \tag{4.30}$$

2^{nde} étape

Soit y_2 la nouvelle sortie :

$$y_2 = x_2 - \alpha_1 \tag{4.31}$$

Le système à stabiliser s'écrit maintenant :

$$\dot{x}_1 = y_2 + \alpha_1 \tag{4.32a}$$
$$\dot{y}_2 = -g\tan\phi_1 - \dot{\alpha}_1 \tag{4.32b}$$

Proposons la fonction définie positive suivante :

$$V_2 = V_1 + \frac{y_2^2}{2} \tag{4.33}$$

alors :

$$\dot{V}_2 = \dot{V}_1 + y_2\dot{y}_2 = \dot{V}_1 - y_2(g\tan\phi_1 + \dot{\alpha}_1) \tag{4.34}$$

proposons aussi l'entrée virtuelle $\alpha_2 = (g\tan\phi_1)^v$ telle que :

$$\alpha_2 = k_2y_2 - \dot{\alpha}_1 \tag{4.35}$$

où k_2 est une constante strictement positive. Il s'ensuit :

$$\dot{V}_2 = \dot{V}_1 - k_2y_2^2 \tag{4.36}$$

3^{ime} **étape**

Soit y_3 la nouvelle sortie :

$$y_3 = g\tan\phi_1 - \alpha_2 \tag{4.37}$$

Le système à stabiliser s'écrit maintenant :

$$\dot{x}_1 = y_2 + \alpha_1 \tag{4.38a}$$
$$\dot{y}_2 = -y_3 - \alpha_2 - \dot{\alpha}_1 \tag{4.38b}$$
$$\dot{y}_3 = g(1+\tan^2\phi_1)\phi_2 - \dot{\alpha}_2 \tag{4.38c}$$

Proposons la fonction définie positive suivante :

$$V_3 = V_2 + \frac{y_3^2}{2} \tag{4.39}$$

alors :

$$\dot{V}_3 = \dot{V}_2 + y_3\dot{y}_3 = \dot{V}_2 + y_3\Big(g(1+\tan^2\phi_1)\phi_2 - \dot{\alpha}_2\Big) \tag{4.40}$$

proposons aussi l'entrée virtuelle $\alpha_3 = \Big(g(1+\tan^2\phi_1)\phi_2\Big)^v$ telle que :

$$\alpha_3 = -k_3 y_3 + \dot{\alpha}_2 \tag{4.41}$$

où k_3 est une constante strictement positive. Il s'ensuit :

$$\dot{V}_3 = \dot{V}_2 - k_3 y_3^2 \tag{4.42}$$

4^{ime} **étape**

Soit y_4 la nouvelle sortie :

$$y_4 = g(1+\tan^2\phi_1)\phi_2 - \alpha_3 \tag{4.43}$$

Le système à stabiliser s'écrit maintenant :

$$\dot{x}_1 = y_2 + \alpha_1 \tag{4.44a}$$
$$\dot{y}_2 = -g\tan\phi_1 - \dot{\alpha}_1 \tag{4.44b}$$
$$\dot{y}_3 = y_4 + \alpha_2 - \dot{\alpha}_2 \tag{4.44c}$$
$$\dot{y}_4 = g(1+\tan^2\phi_1)\frac{\tau_{B\phi}}{J} + 2g(1+\tan^2\phi_1)\phi_2^2\tan\phi_1 - \dot{\alpha}_3 \tag{4.44d}$$

Proposons la fonction de Lyapunov suivante :

$$V_4 = V_3 + \frac{y_4^2}{2} = \frac{1}{2}(y_1^2 + y_2^2 + y_3^2 + y_4^2) \tag{4.45}$$

alors :

$$\dot{V}_4 = \dot{V}_3 + y_4\dot{y}_4 = \dot{V}_3 + y_4\Big(g(1+\tan^2\phi_1)\frac{\tau_{B\phi}}{J} + 2g(1+\tan^2\phi_1)\phi_2^2\tan\phi_1 - \dot{\alpha}_3\Big) \tag{4.46}$$

proposons aussi l'entrée $\tau_{B\phi}$ telle que :

$$\tau_{B\phi} = J\frac{-k_4y_4 - 2g(1+\tan^2\phi_1)\phi_2^2\tan\phi_1 + \dot{\alpha}_3}{g(1+\tan^2\phi_1)} \tag{4.47}$$

où k_4 est une constante strictement positive. Il s'ensuit :

$$\dot{V}_4 = \dot{V}_3 - k_4y_4^2 = -k_1x_1^2 - k_2y_2^2 - k_3y_3^2 - k_4y_4^2 \tag{4.48}$$

L'entrée $\tau_{B\phi}$ proposée permet donc d'avoir $\dot{V}_4 < 0$ et le système (4.25) est globalement asymptotiquement stable. Afin d'exprimer $\tau_{B\phi}$ en fonction de x_1, x_2, ϕ_1 et ϕ_2, il faut d'abord réécrire y_4 et $\dot{\alpha}_3$ en fonction de ces mêmes variables. On obtient alors (voir les détails du calcul en annexe A) :

$$y_4 = g(1+\tan^2\phi_1)\phi_2 + (k_1+k_2+k_3)g\tan\phi_1 - (k_1k_2+k_1k_3+k_2k_3)x_2 - k_1k_2k_3x_1 \tag{4.49a}$$

$$\dot{\alpha}_3 = -(k_1+k_2+k_3)g(1+\tan^2\phi_1)\phi_2 - (k_1k_2+k_1k_3+k_2k_3)g\tan\phi_1 + k_1k_2k_3x_2 \tag{4.49b}$$

L'entrée de contrôle se réécrit alors :

$$\tau_{B\phi} = J\Big(\frac{\overline{k}_1x_1 + \overline{k}_2x_2 - \overline{k}_3g\tan\phi_1}{g(1+\tan^2\phi_1)} - \overline{k}_4\phi_2 - 2\phi_2^2\tan\phi_1\Big) \tag{4.50}$$

avec :

$$\overline{k}_1 = k_1k_2k_3k_4 \tag{4.51a}$$

$$\overline{k}_2 = k_1k_2k_3 + k_1k_2k_4 + k_1k_3k_4 + k_2k_3k_4 \tag{4.51b}$$

$$\overline{k}_3 = k_1k_2 + k_1k_3 + k_1k_4 + k_2k_3 + k_2k_4 + k_3k_4 \tag{4.51c}$$

$$\overline{k}_4 = k_1 + k_2 + k_3 + k_4 \tag{4.51d}$$

4.1.8 Simulations

Cette section présente les résultats en simulations des lois de commandes (4.15), (4.17), (4.20) et (4.50) pour stabiliser l'angle de tangage et le déplacement horizontal du *PVTOL*. Le modèle a été choisi tel que $M = J = 1$. Les paramètres utilisés pour chacun des contrôleurs sont reportés dans la table 4.1. Les valeurs pour les contrôleurs par saturations ont été choisies de telle sorte que les entrées τ_ϕ soient bornées par la même valeur pour les deux contrôleurs (voir table 4.1), c'est à dire :

$$|\tau_{NS_\phi}| \leq b_1 = 0.47 \tag{4.52a}$$

$$|\tau_{S_\phi}| \leq b_1 + b_2 + b_3 + b_4 = 0.47 \tag{4.52b}$$

Il est à noter que l'échelle de temps n'est pas la même sur toutes les figures de cette section, afin de pouvoir montrer plus de détails si cela est nécessaire.

Les figures 4.3 et 4.4 montrent respectivement les états $(x, \dot{x})$ et $(\phi, \dot{\phi})$ du *PVTOL* avec comme conditions initiales : $x(0) = 5$, $\dot{x}(0) = 0$, $\phi(0) = 0$, $\dot{\phi}(0) = 0$. Lors de cet essai, les valeurs des gains ont été ajustées de façon à ce que les temps de réponses à 5% de la position x soit à peu près identiques pour les contrôleurs linéaire, par *backstepping* et par saturations emboitées. Ces gains sont ceux de la colonne "Valeurs 1" de la table 4.1. La figure 4.4 montre par exemple que les amplitudes du mouvement en ϕ sont plus petites avec les lois par saturations ; les mouvements sont donc en général moins brusques. La figure 4.5 montre d'ailleurs les valeurs de l'entrée de commande : dans le cas de la loi linéaire, la sortie vaut plus de 175 fois la sortie de loi par saturations emboitées ; la loi par *backstepping* quant à elle étant 17 fois supérieure. Ces lois sont donc beaucoup moins efficaces car à temps de réponses équivalents, elles demandent plus d'énergie aux actionneurs. De plus, les valeurs de l'entrée de commande peuvent ne pas être réalistes car la vitesse des moteurs du *PVTOL* est en fait limitée sur la plateforme expérimentale. Enfin, le zoom sur la figure 4.5 montre que les entrées de commandes des fonctions par saturations sont loin d'arriver à la saturation maximale (qui est de 0.47 selon (4.52)). Cependant la condition (4.19c) sur la loi par fonctions de saturations emboitées (4.17) est assez restrictive et ne permet pas d'augmenter beaucoup b_3 afin d'améliorer le temps de convergence.

Les figures 4.6 et 4.7 montrent respectivement les états $(x, \dot{x})$ et $(\phi, \dot{\phi})$ du *PVTOL* avec comme conditions initiales : $x(0) = 25$, $\dot{x}(0) = 0$, $\phi(0) = \pi/6$, $\dot{\phi}(0) = 0$. Dans ce cas, le *PVTOL* est plus éloigné que précédemment du point d'équilibre. Les paramètres utilisés sont ceux de la colonne "Valeur 2" de la table 4.1 où le gain $K(1)$ de la loi linéaire a été augmenté. La loi de commande linéaire

Contrôleur	Paramètre	Valeur 1	Valeur 2
Linéaire (τ_{L_ϕ})	$K(1)$	-2.1	-7
	$K(2)$	-10	-10
	$K(3)$	62	62
	$K(4)$	14	14
Backstepping (τ_{B_ϕ})	$\bar{k}_1$	2	2
	$\bar{k}_2$	10	10
	$\bar{k}_3$	9	9
	$\bar{k}_4$	4	4
Saturations emboitées (τ_{NS_ϕ})	b_1	0.4700	0.4700
	b_2	0.2349	0.2349
	b_3	0.1174	0.1174
	b_4	0.0587	0.0587
Saturations séparées (τ_{S_ϕ})	b_1	0.0400	0.0400
	b_2	0.0700	0.0700
	b_3	0.1200	0.1200
	b_4	0.2400	0.2400
	k_1	1	1
	k_2	0.1	0.1
	k_3	1	1
	k_4	1	1

Table 4.1 – Gains utilisés dans les lois de commandes (4.15), (4.50), (4.17) et (4.20).

n'est alors plus capable de stabiliser le *PVTOL* car elle impose un mouvement trop important à l'angle ϕ qui se stabilise à π. Cela montre les limites de cette loi, qui a été faite sur un modèle linéarisé autour du point d'équilibre. Les autres lois de commandes quant à elles arrivent à stabiliser le *PVTOL*. En particulier, la loi par *backstepping* est toujours très rapide, même si l'entrée de commande prend des valeurs importantes (voir figure 4.8).

Enfin, notons que les figures 4.3 et 4.6 ne permettent pas de conclure sur la rapidité de convergence des lois de commandes par saturations. Chacune des deux est en effet plus rapide que l'autre dans un cas particulier.

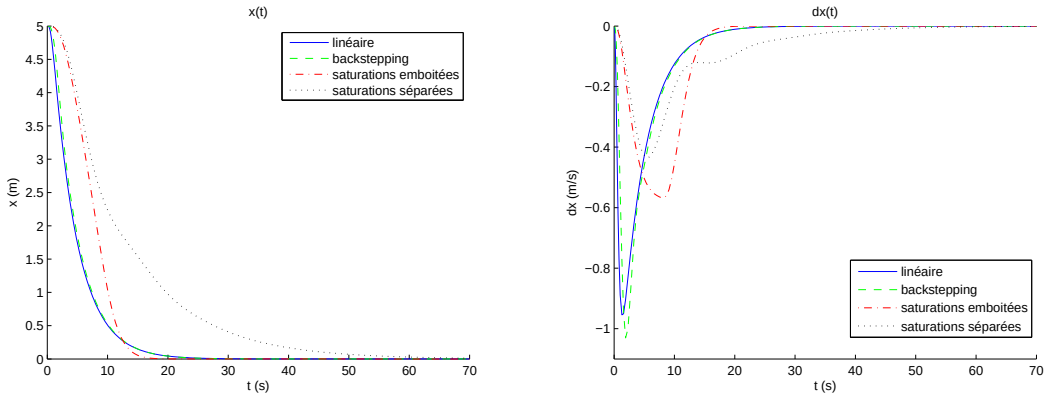


Figure 4.3 – Etats $x(t)$ et $\dot{x}(t)$ du *PVTOL*. Les conditions initiales sont : $x(0) = 5$, $\dot{x}(0) = 0$, $\phi(0) = 0$, $\dot{\phi}(0) = 0$.

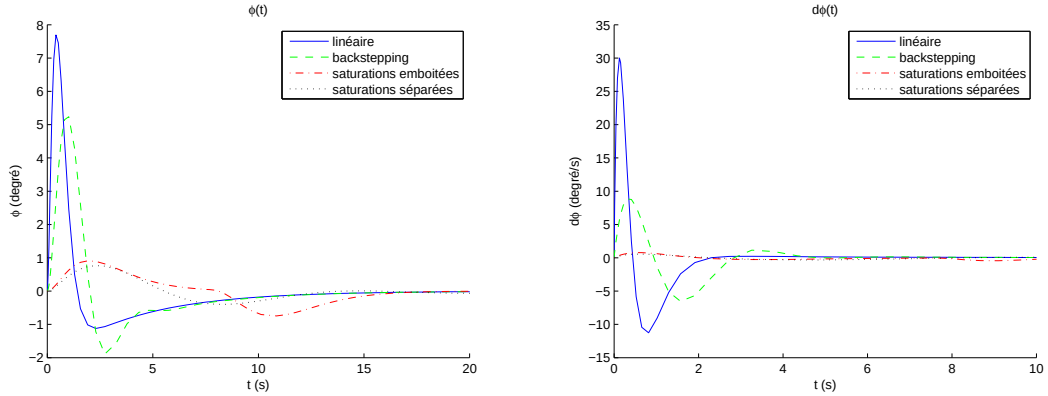


Figure 4.4 – Etats $\phi(t)$ et $\dot{\phi}(t)$ du *PVTOL*. Les conditions initiales sont : $x(0) = 5$, $\dot{x}(0) = 0$, $\phi(0) = 0$, $\dot{\phi}(0) = 0$.

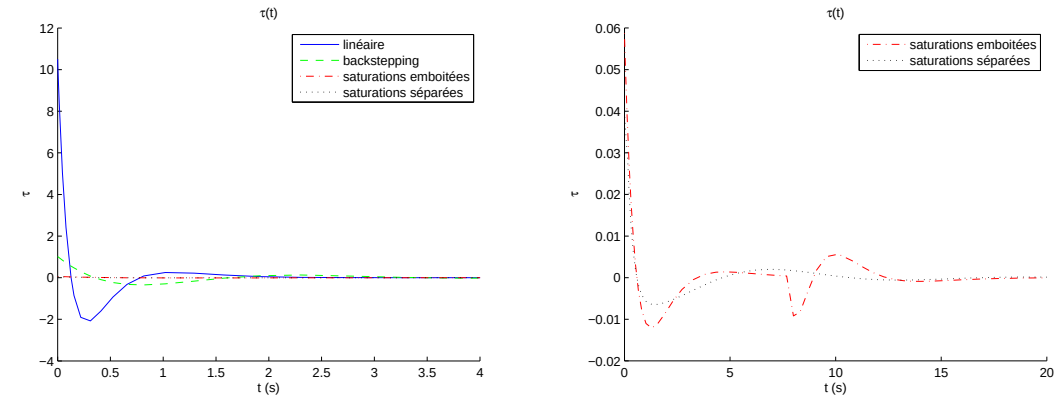


Figure 4.5 – Entrée de contrôle $\tau_\phi(t)$ du *PVTOL*. Zoom des entrées pour les lois par fonctions de saturations. Les conditions initiales sont : $x(0) = 5$, $\dot{x}(0) = 0$, $\phi(0) = 0$, $\dot{\phi}(0) = 0$.

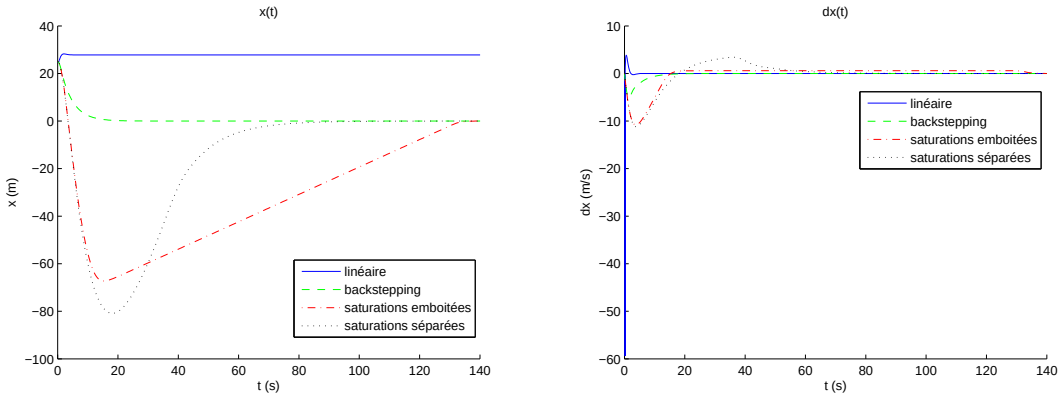


Figure 4.6 – Etats $x(t)$ et $\dot{x}(t)$ du *PVTOL*. Les conditions initiales sont : $x(0) = 25$, $\dot{x}(0) = 0$, $\phi(0) = \pi/6$, $\dot{\phi}(0) = 0$.

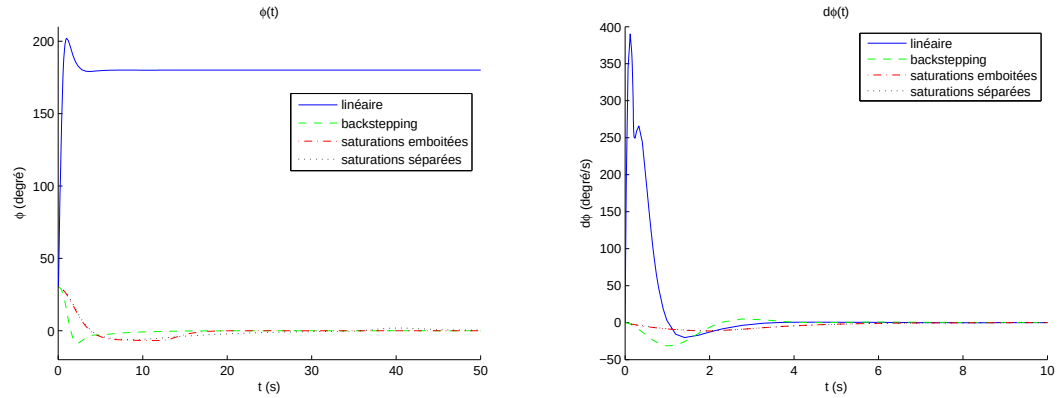


Figure 4.7 – Etats $\phi(t)$ et $\dot{\phi}(t)$ du *PVTOL*. Les conditions initiales sont : $x(0) = 25$, $\dot{x}(0) = 0$, $\phi(0) = \pi/6$, $\dot{\phi}(0) = 0$.

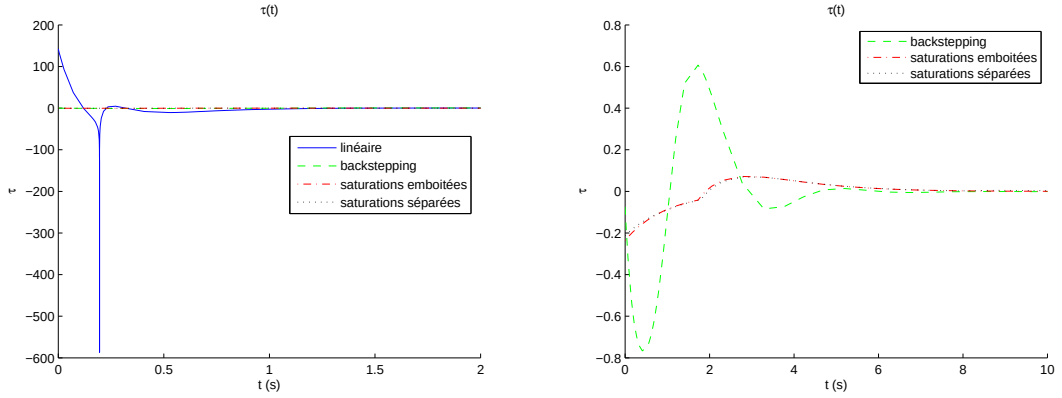


Figure 4.8 – Entrée de contrôle $\tau_\phi(t)$ du *PVTOL*. Zoom des entrées pour les lois par fonctions de saturations et par *backstepping*. Les conditions initiales sont : $x(0) = 25$, $\dot{x}(0) = 0$, $\phi(0) = \pi/6$, $\dot{\phi}(0) = 0$.

4.1.9 Expériences

Cette section présente les résultats des expériences en temps réel, obtenus en appliquant les lois de commandes (4.15), (4.17), (4.20) et (4.50) sur une plateforme de type *PVTOL*. La plateforme expérimentale est composée d'un *X4* de *Dragan-Fly*, d'une radio *Futaba*, de deux ordinateurs et d'un système de positionnement 3D *Fastrak* de *Polhemus* (voir [idtf]) pour mesurer la position et l'orientation du quadrirotor. Les différentes lois de commandes ont été implémentées sur le système temps réel *xPC Target* (voir [itu]). Un ordinateur "maitre" est ainsi utilisé pour dessiner la loi de commande sous *Matlab Simulink*, puis la loi est compilée et envoyée à l'ordinateur "esclave" exécutant *xPC Target*. La loi est calculée grâce aux informations du système de positionnement et envoyée à la radiocommande via une carte d'entrées/sorties analogiques. L'ordinateur remplace donc un pilote humain en agissant directement sur la radiocommande et le quadrirotor n'a alors pas besoin d'être modifié. Notons que l'hélicoptère possède déjà des gyromètres pour aider au pilotage, mais que ceux-ci ne sont pas suffisants pour le vol autonome.

Dans ces expériences, les angles de roulis et de lacet du quadrirotor sont contrôlés par la loi proposée par [CLD05], c'est à dire avec des saturations emboitées. De plus, l'altitude est contrôlée par la loi (4.8), alors que l'angle de tangage et la position latérale sont contrôlés par les lois de commandes présentées dans ce chapitre ; cela permet donc de réaliser une plateforme expérimentale de type *PVTOL*.

Les paramètres utilisés en simulation ne peuvent pas être repris ici car le modèle de simulation est différent du modèle réel. Les gains ont donc été ajustés par tâtonnementsjusqu'à obtenir un bon comportement. Les positions désirées sont $x_d = 0$ cm, $z_d = 30$ cm et $\phi_d = 0°$.

Les figures 4.9, 4.10, 4.11 et 4.12 représentent respectivement les résultats des essais pour les lois linéaire (4.15), par saturations emboitées (4.17), par saturations séparées (4.20) et par *backstepping* (4.50). Pour chaque figure, les positions en x et z, ainsi que l'angle de tangage ϕ et l'entrée de commande τ_ϕ y sont représentés ; les lignes en pointillés représentant la position désirée.

Globalement, les performances des contrôleurs linéaire (figure 4.9), par saturations emboitées (figure 4.10) et par saturations séparées (figure 4.11) sont assez similaires. Dans les trois cas, le drone reste autour de sa position d'équilibre même s'il existe des oscillations. Cependant celles-ci sont d'un peu plus grande amplitude dans le cas de la loi par saturations emboitées (notamment pour l'angle ϕ). La loi par saturations séparées semble donc légèrement mieux se comporter car l'amplitude des oscillations de ϕ est plus petite que dans les deux autres cas. De plus, la fréquence des oscillations sur la position x est aussi plus faible. La loi de commande

linéaire à un comportement satisfaisant car le drone est resté relativement proche de sa position désirée. S'il avait été placé plus loin, la loi aurait sans doute rendu l'appareil plus instable et plus dangereux, en imposant de grands mouvements sur l'angle ϕ (tel que sur la figure 4.7).

La loi de commande par *backstepping* a été implémentée sans succès sur la plateforme expérimentale. En effet, les gains n'ont pas pu être réglés correctement pour obtenir une bonne stabilisation du drone. La figure 4.12 montre ainsi les résultats obtenus ; de fortes oscillations sur l'angle ϕ empêchent l'hélicoptère de réellement décoller. Notons que les paramètres $\overline{k}_1$ et $\overline{k}_2$ dans la loi (4.50) jouent sur plusieurs états, ce qui rend plus difficile le réglage de la loi de commande.

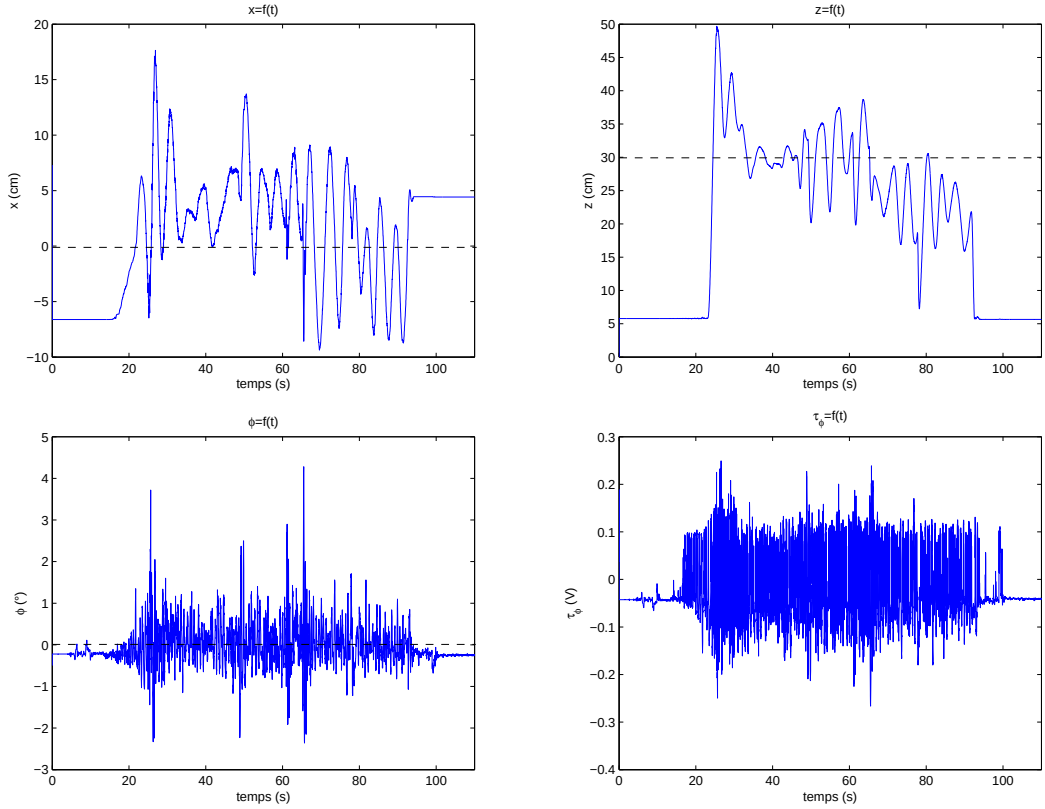


Figure 4.9 – Stabilisation du quadrirotor avec le contrôleur linéaire (4.15).

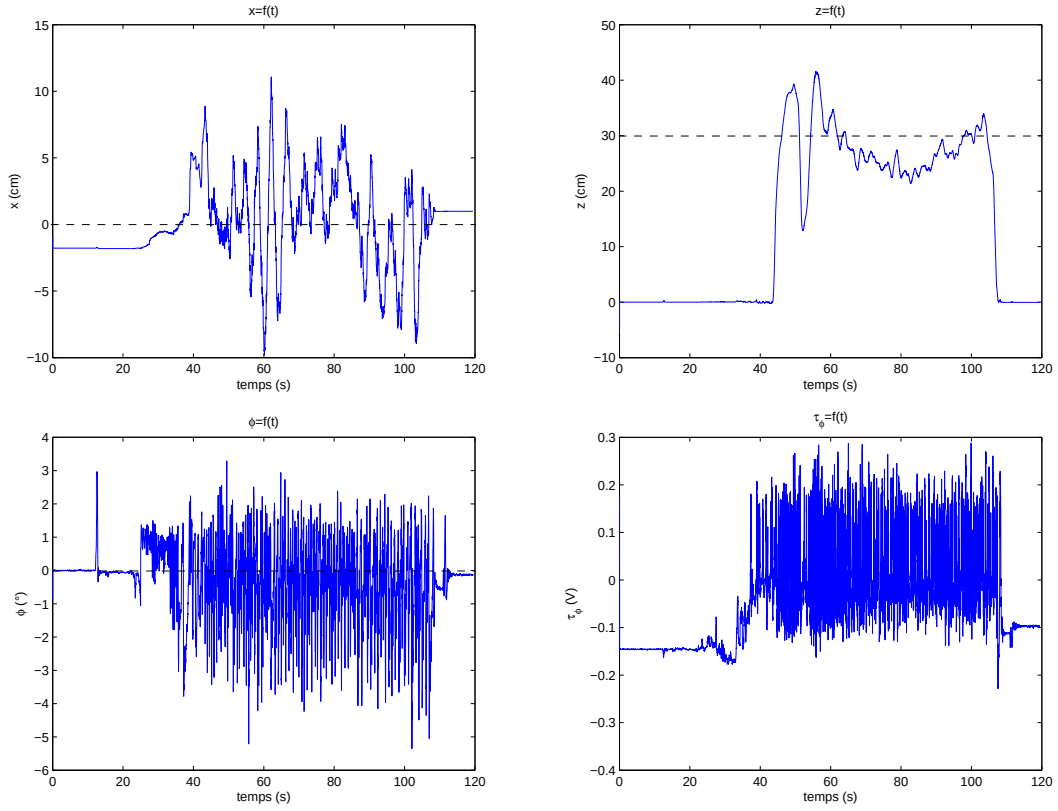


Figure 4.10 – Stabilisation du quadrirotor avec le contrôleur (4.17) par saturations emboitées.

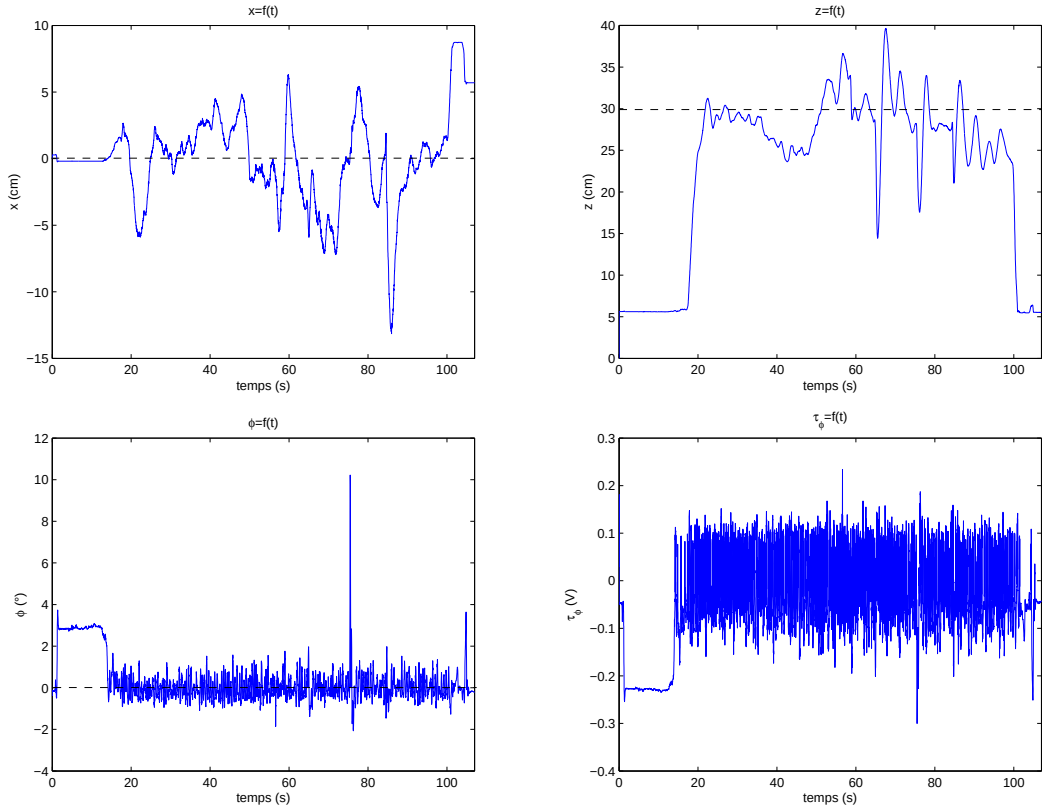


Figure 4.11 – Stabilisation du quadrirotor avec le contrôleur (4.20) par saturations séparées.

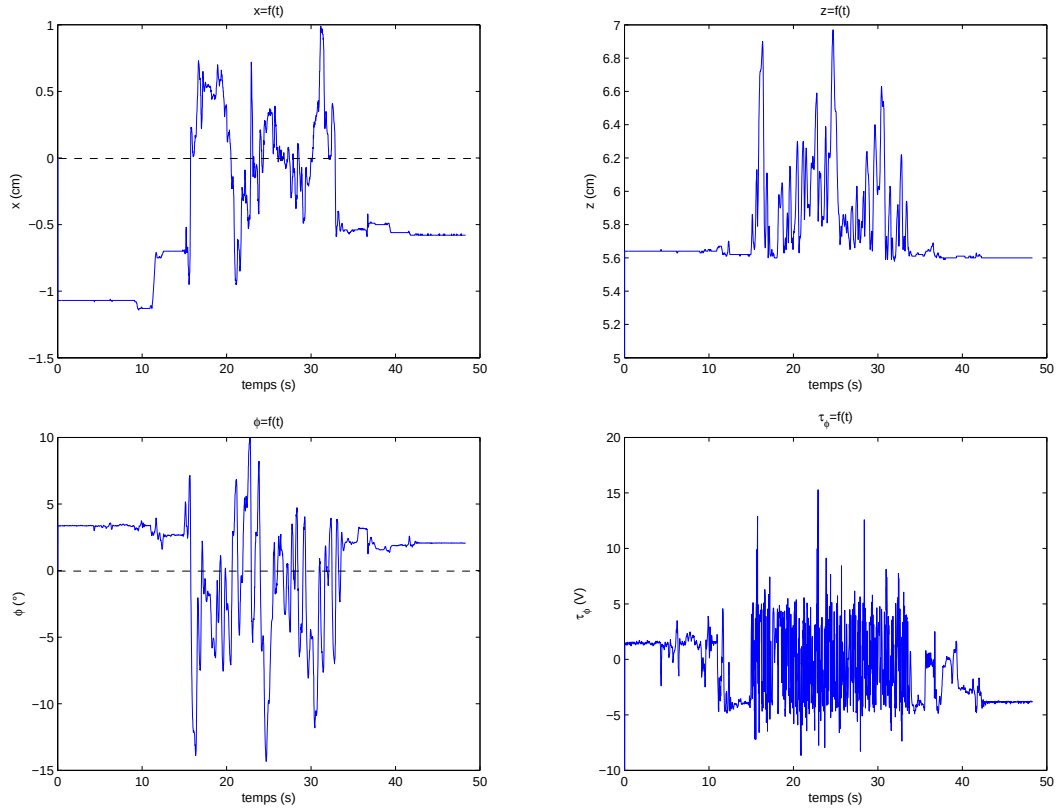


Figure 4.12 – Réponse instable du quadrirotor avec le contrôleur par *backstepping* (4.50). Les oscillations sur l'angle de tangage ϕ empêchent l'hélicoptère de réellement décoller.

4.1.10 Conclusion

Cette section a présenté plusieurs types de lois de commande, linéaires et non linéaires. Les simulations montrent dans l'ensemble de bons résultats, les expériences étant pas contre plus difficiles à comparer. Cependant, pour le système réel il faut prendre en compte que les entrées peuvent être bornées ; or toutes les lois présentées ici ne le permettent pas ((4.15) et (4.50)). De plus, pour adapter une loi à un système réel dont le modèle n'est pas bien connu, il faut en général ajuster les paramètres par tâtonnements. Cette opération sera d'autant plus difficile si les paramètres influent chacun sur plusieurs états du système ((4.50), (4.17) et (4.20)). C'est pourquoi il peut être intéressant d'avoir une loi de commande ayant des saturations et dont les gains ne jouent que sur un seul état. La section suivante présente donc une loi répondant à ces deux critères. Par ailleurs, comme le montrent les équations (4.9b) et (4.12), le système à stabiliser est en fait une chaîne à deux ou quatre intégrateurs. La section suivante s'intéresse donc au cas général d'une chaîne de n intégrateurs, afin d'avoir une loi de commande pouvant s'adapter à plusieurs types de systèmes.

4.2 Généralisation d'une loi de commande par saturation, pour un système à n intégrateurs

4.2.1 État de l'art

La stabilisation de chaines d'intégrateurs a largement été étudiée ces derniers temps. Ainsi, la stabilisation d'une chaine d'intégrateurs en utilisant une entrée bornée avec retard [MMN03], [MR01] ou sans [Tee92a] a été abordée. Ce sont d'ailleurs ces derniers travaux ([Tee92a]) qui ont permis de résoudre le problème (ouvert jusque là) de chaines d'intégrateurs de dimension supérieure (strictement) à deux. Grâce à ces résultats des problèmes théoriques (voir [Tee92b], [SSY94]) mais aussi plus appliqués (voir [Tee93]) ont pu être résolus par la suite. Les travaux de [Tee92a] sont de plus à l'origine de la méthode de conception de lois de contrôle dite de *forwarding* (voir par exemple [MP96], [JSK96]).

De manière générale, que se soit pour des chaines d'intégrateurs linéaires ([LS93]) ou pour des chaines non linéaires ([MI00], [GSB99]), la loi de commande proposée consiste en une généralisation du schéma de saturations emboitées de [Tee92a] ou en une combinaison linéaire de saturations introduites par [SSY94]. Notamment,

une loi de commande basée sur ces derniers travaux a été proposée par [MH05], en introduisant des fonctions de saturations à bornes variables. Cette approche est intéressante car elle permet à un état d'utiliser une "réserve" de saturation, rendue disponible par les états supérieurs qui n'auraient pas encore été saturés, améliorant ainsi le temps de réponse.

Un contrôle borné pour une classe de systèmes non linéaires de type *feedforward* a été proposé par [KA04] et [KA05], en utilisant des fonctions de saturations séparées. Plus récemment, [SCE+07] et [SGCL08] ont alors proposé un algorithme basé sur le même principe pour stabiliser le $PVTOL$, représentant un système à quatre intégrateurs. Ces résultats ont inspiré nos travaux de recherche.

Cette section présente donc une nouvelle méthode (que nous avons proposée dans [SCS]) pour stabiliser une chaîne d'intégrateurs en cascade avec des entrées bornées. Les résultats obtenus en simulations et lors d'expériences en temps réel sont aussi montrés. La différence par rapport aux travaux précédents est que le contrôleur n'utilise pas de saturations emboitées, et que chaque saturation borne un état seulement. De plus, la procédure présentée est plus simple à comprendre que celles vues dans la littérature.

4.2.2 Système

Un système général ayant n intégrateurs en cascade peut s'écrire de la façon suivante :

$$\dot{x}_1 = \alpha_1 x_2 \tag{4.53a}$$
$$\vdots$$
$$\dot{x}_{n-1} = \alpha_{n-1} x_n \tag{4.53b}$$
$$\dot{x}_n = \alpha_n u \tag{4.53c}$$

avec $\alpha_i \neq 0$ constant, $\forall i \in [1, n]$. Introduisons le changement de coordonnées suivant :

$$y_1 = x_1 \tag{4.54a}$$
$$y_2 = \alpha_1 x_2 \tag{4.54b}$$
$$\vdots$$
$$y_n = \alpha_1 ... \alpha_{n-1} x_n \tag{4.54c}$$
$$u' = \alpha_1 ... \alpha_n u \tag{4.54d}$$

Le système (4.53) devient alors

$$\dot{y}_1 = y_2 \tag{4.55a}$$
$$\vdots$$
$$\dot{y}_n = u' \tag{4.55b}$$

4.2.3 Loi de commande

Nous allons montrer dans les sections suivantes que la loi de commande (4.56) stabilise asymptotiquement le système (4.55) et borne chaque état y_i.

$$u' = -\sum_{i=1}^{n} \sigma_{b_i}(k_i y_i) \tag{4.56}$$

où k_i, $\forall i \in [1, n]$ est une constante. Soit ξ_i une fonction bornée, $|\xi_i| \leq b_{\xi_i}$ avec $b_{\xi_i} > 0$, telle que :

$$\xi_i = \sum_{j=1}^{i-1} \sigma_{b_j}(k_j y_j) \tag{4.57}$$

L'équation (4.56) peut alors s'écrire :

$$u' = -\sum_{j=i}^{n} \sigma_{b_j}(k_j y_j) - \xi_i \tag{4.58}$$

quant à ξ_i :

$$\xi_1 = 0 \tag{4.59a}$$
$$\xi_i = \sigma_{b_{i-1}}(k_{i-1} y_{i-1}) + \xi_{i-1},\ \forall i \in [2, n] \tag{4.59b}$$

Introduisons la proposition suivante :

Proposition P_i**.** *$\forall i \in [1, n]$, $\exists$ un temps T_{n-i+1} suffisamment grand tel que $\forall t > T_{n-i+1}$, $u' = -\sum_{j=i}^{n} k_j y_j - \xi_i$ et $|y_i|$ est borné.*

Cette proposition sera démontrée par un raisonnement par récurrence dans les sections suivantes.

Par ailleurs, sur l'intervalle de temps $[0, T_{n-i+1}]$, les autre états ne peuvent pas diverger. Cela est prouvé par le lemme 4.2.1 introduit par [Mar03a].

Lemme 4.2.1. *Toute trajectoire en boucle fermée d'un système linéaire quelconque ne peut diverger en un temps fini si l'entrée de commande est bornée.*

Démonstration. En réécrivant le système (4.55) sous forme matricielle :

$$\dot{y} = Ay + Bu' \tag{4.60}$$

tel que,

$$|u'| \leq \sum_{i=1}^{n} b_i = M \tag{4.61}$$

alors :

$$\frac{d\|y\|^2}{dt} = 2y^T Ay + 2y^T Bu' \leq 2\lambda_{max}^A \|y\|^2 + 2\lambda_{max}^B M\|y\| \tag{4.62}$$

où λ_{max}^P représente la plus grande valeur singulière d'une matrice P. En reconnaissant une équation différentielle de Bernoulli, il s'ensuit :

$$\|y(t)\|^2 \leq \Big(\frac{\lambda_{max}^B}{\lambda_{max}^A} M(e^{\lambda_{max}^A t} - 1) + \|y(0)\| e^{\lambda_{max}^A t}\Big)^2 \tag{4.63}$$

Ce qui conclut la démonstration, le système ne peut pas diverger en un temps fini. □

4.2.4 Première étape, $i = n$

Il sera prouvé ici que P_i est vraie pour $i = n$. Ainsi d'après (4.58) :

$$u' = -\sigma_{b_n}(k_n y_n) - \xi_n \tag{4.64}$$

Proposons la fonction définie positive suivante :

$$V_n = \frac{1}{2} z_n^2 \tag{4.65}$$

avec

$$z_i = k_{i+1} y_i + z_{i+1},\ \forall i \in [1, n-1] \tag{4.66a}$$

$$z_n = y_n \tag{4.66b}$$

alors,

$$\dot{V}_n = u' y_n = -y_n \Big(\sigma_{b_n}(k_n y_n) + \xi_n\Big) \tag{4.67}$$

Supposons que $b_n > b_{\xi_n}$. Alors, $|k_n y_n| > b_{\xi_n}$ implique que $\dot{V}_n < 0$. Il existe donc un temps T_1 tel que $\forall t > T_1$,

$$|y_n| \leq \frac{b_{\xi_n}}{k_n} \tag{4.68}$$

ce qui implique que :

$$u' = -k_n y_n - \xi_n, \, \forall t > T_1 \tag{4.69}$$

Ainsi P_i est vraie pour $i = n$.

4.2.5 Seconde étape

Supposons maintenant que P_i est vraie pour un i donné. Il faut alors montrer que P_{i-1} est aussi vraie. En utilisant P_i et (4.69), il s'ensuit que :

$$\begin{aligned} u' &= -\sum_{j=l}^{n} k_j y_j - \xi_l, \, \forall t > T_{n-l+1}, \, \forall l \in [i, n] \\ &\quad \text{et } |y_l| \text{ est borné} \end{aligned} \tag{4.70}$$

D'après (4.66),

$$\sum_{j=i-1}^{n-1} z_j = \sum_{j=i-1}^{n-1} \Big(k_{j+1} y_j + z_{j+1} \Big) \tag{4.71a}$$

$$z_{i-1} - z_n = \sum_{j=i-1}^{n-1} k_{j+1} y_j \tag{4.71b}$$

$$z_{i-1} = \sum_{j=i-1}^{n-1} k_{j+1} y_j + y_n \tag{4.71c}$$

d'où,

$$\begin{aligned} \dot{z}_{i-1} &= \sum_{j=i-1}^{n-1} k_{j+1} y_{j+1} + u' \\ &= \sum_{j=i}^{n} k_j y_j + u' \\ &= -\xi_i \end{aligned} \tag{4.72}$$

Proposons la fonction définie positive suivante :

$$V_{i-1} = \frac{1}{2} z_{i-1}^2 \tag{4.73}$$

d'où,

$$\begin{aligned}\dot{V}_{i-1} &= -\Big(\sum_{j=i-1}^{n-1} k_{j+1}y_j + y_n\Big)\xi_i \\ &= -\Big(k_i y_{i-1} + \sum_{j=i}^{n-1} k_{j+1}y_j + y_n\Big)\xi_i \\ &= -(k_i y_{i-1} + z_i)\Big(\sigma_{b_{i-1}}(k_{i-1}y_{i-1}) + \xi_{i-1}\Big) \qquad (4.74)\end{aligned}$$

or $\xi_1 = 0$, d'où :

$$\dot{V}_1 = -\sigma_{b_1}(k_1 y_1)(k_2 y_1 + z_2) \qquad (4.75)$$

Supposons que,

$$b_{i-1} > b_{\xi_{i-1}} \qquad (4.76)$$

donc si :

$$|y_{i-1}| > \frac{b_{\xi_{i-1}}}{k_{i-1}} \qquad (4.77)$$

il s'ensuit alors :

$$sgn\Big(\sigma_{b_{i-1}}(k_{i-1}y_{i-1}) + \xi_{i-1}\Big) = sgn(y_{i-1}) \qquad (4.78)$$

où $sgn()$ est la fonction signe telle que :

$$sgn(u) = 1,\ \forall u > 0 \qquad (4.79a)$$
$$sgn(0) = 0 \qquad (4.79b)$$
$$sgn(u) = -1,\ \forall u < 0 \qquad (4.79c)$$

comme y_j est borné $\forall j \in [i, n]$, alors z_j est aussi borné ; sa borne sera nommée b_{z_j}. De plus, si

$$|y_{i-1}| > \frac{b_{z_i}}{k_i} \qquad (4.80)$$

alors,

$$\mathrm{sgn}(k_i y_{i-1} + z_i) = \mathrm{sgn}(y_{i-1}) \qquad (4.81)$$

et

$$|y_{i-1}| > \frac{b_{\xi_{i-1}}}{k_{i-1}} > \frac{b_{z_i}}{k_i},\ \forall i \in [2, n] \qquad (4.82a)$$
$$|y_1| > \frac{b_1}{k_1} > \frac{b_{z_2}}{k_2} \qquad (4.82b)$$

implique que $\dot{V}_{i-1} < 0$ et $\exists\ T_{n-i+2} > T_{n-i+1}$ tel que $\forall t > T_{n-i+2}$

$$|y_{i-1}| < \frac{b_{\xi_{i-1}}}{k_{i-1}},\ \forall i \in [2, n] \tag{4.83a}$$

$$|y_1| < \frac{b_1}{k_1} \tag{4.83b}$$

et

$$u' = -\sum_{j=i-1}^{n} k_j y_j - \xi_{i-1},\ \forall t > T_n \tag{4.84}$$

4.2.6 Conclusion du théorème de récurrence

Il a été démontré que si P_i est vraie, alors P_{i-1} est vraie aussi. Donc en utilisant le théorème de récurrence P_i est vraie $\forall i \in [1, n]$. Ainsi, pour $i = 1$ on obtient :

"Il existe un temps T_n tel que,

$$u' = -\sum_{i=1}^{n} k_i y_i,\ \forall t > T_n\text{"} \tag{4.85}$$

Notons que les conditions suivantes doivent être satisfaites :

$$\frac{b_{\xi_i}}{k_i} > \frac{b_{z_{i+1}}}{k_{i+1}},\ \forall i \in [2, n-1] \tag{4.86a}$$

$$\frac{b_1}{k_1} > \frac{b_{z_2}}{k_2} \tag{4.86b}$$

$$b_i > b_{\xi_i},\ \forall i \in [2, n] \tag{4.86c}$$

avec d'après (4.59b),

$$b_{\xi_i} = \sum_{j=2}^{i} b_{j-1} \tag{4.87}$$

et d'après (4.71c) et (4.83a),

$$b_{z_i} = \sum_{j=i}^{n-1} \frac{k_{j+1} b_{\xi_j}}{k_j} + \frac{b_{\xi_n}}{k_n} \tag{4.88}$$

4.2.7 Convergence vers zéro

D'après (4.85) tous les états sont bornés et $\forall\ t > T_n$ la loi de commande non linéaire u', devient linéaire (sans les saturations) ; le système restant stable.

Afin que chacun des états convergent vers zéro, nous utilisons le critère de Routh Hurtwitz. Pour cela, il faut exprimer le polynôme caractéristique de la matrice suivante :

$$A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & \dots & & & 1 \\ -k_1 & -k_2 & \dots & & -k_n \end{bmatrix} \tag{4.89}$$

La matrice A est une matrice compagne, le déterminent de son polynôme caractéristique est le suivant (voir [Rou08]) :

$$(-\lambda)^n + (-1)^n \sum_{j=0}^{n-1} \lambda^j k_{j+1} \tag{4.90}$$

En utilisant le critère de Routh Hurwitz sur ce déterminant, les paramètres k_i peuvent alors être choisis afin de satisfaire la stabilité du système (4.55) et de garantir la convergence de tous les états vers 0. De plus, les bornes σ_i et les gains k_i doivent satisfaire (4.86).

4.2.8 Application au *PVTOL*

Les résultats obtenus précédemment sont appliqués dans cette partie au *PVTOL*.

Plateforme

La plateforme utilisée précédemment (voir section 4.1.9) n'étant plus fonctionnelle lors des essais de cette nouvelle loi de commande, un autre type de *PVTOL* a été utilisé. Il s'agit ici d'un engin à deux hélices se déplaçant grâce à des billes sur un plan incliné d'un angle α. Cela permet d'avoir une plateforme évoluant seulement dans un plan. Un schéma est présenté sur la figure 4.13.

En gardant les mêmes notations qu'à la section 4.1.2, les équations de Newton appliquées à ce système donnent :

$$M\ddot{x} = -u \sin\phi \tag{4.91a}$$
$$M\ddot{y} = u\cos\phi - Mg\sin\alpha \tag{4.91b}$$
$$J\ddot{\phi} = \tau_\phi \tag{4.91c}$$

Ce système est bien équivalent à celui (4.7) établi précédemment.

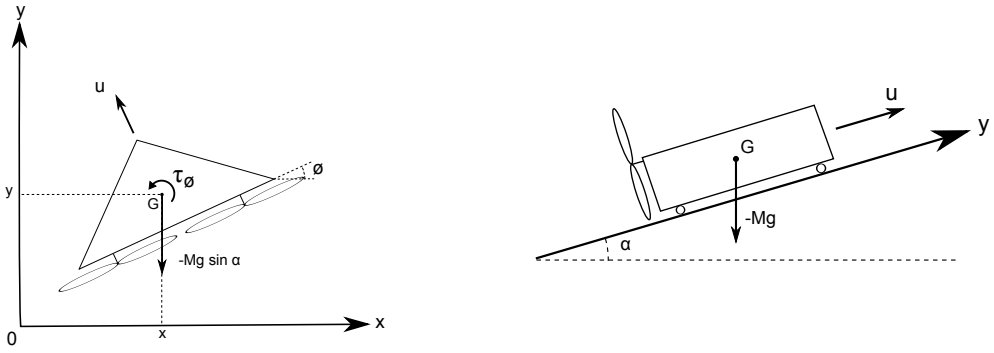


Figure 4.13 – Schéma du *PVTOL*.

Lois de commandes

Afin de stabiliser le *PVTOL*, les lois de commande suivantes sont proposées :

$$u = \frac{u' + Mg\sin\alpha}{\cos\phi} \tag{4.92a}$$

$$\tau_\phi = J\tau'_\phi \tag{4.92b}$$

où u' et τ'_ϕ sont les nouvelles entrées de commandes afin de garantir la convergence vers zéro. En les introduisant dans (4.91), il s'ensuit :

$$\ddot{x} = -(u' + Mg\sin\alpha)\tan\phi \tag{4.93a}$$

$$\ddot{y} = u' \tag{4.93b}$$

$$\ddot{\phi} = \tau'_\phi \tag{4.93c}$$

Pour la position verticale, l'équation précédente donne :

$$\dot{y}_1 = y_2 \tag{4.94a}$$

$$\dot{y}_2 = u' \tag{4.94b}$$

où $[y_1\ y_2]^T = [y\ \dot{y}]^T$. Appliquons alors les résultats précédents au déplacement vertical :

$$u' = -\sigma_a\Big(k_a(y - y_d)\Big) - \sigma_b(k_b\dot{y}) \tag{4.95}$$

où y_d est l'altitude désirée. Soit b_a la borne de σ_a et b_b celle de σ_b. En utilisant (4.90), on obtient les conditions : $k_a,\ k_b > 0$. De plus, afin de satisfaire (4.86) il

s'ensuit que :

$$b_b > b_a \tag{4.96a}$$
$$k_b^2 > k_a \tag{4.96b}$$

D'après (4.91), (4.92a) et (4.95) il existe un temps $T > 0$ suffisamment grand tel que $\ddot{y}$, $\dot{y}$ et $y - y_d$ sont arbitrairement petits. On suppose alors que (4.93c) se réduit à

$$\ddot{x} = -Mg \sin\alpha \tan\phi, \, \forall t > T \tag{4.97a}$$
$$\ddot{\phi} = \tau'_\phi \tag{4.97b}$$

Afin de simplifier l'écriture, posons $K_g = Mg \sin\alpha$. De plus, linéarisons (4.97) autour de l'origine. Ainsi, le sous système (4.97) se réduit à :

$$\dot{x}_1 = x_2 \tag{4.98a}$$
$$\dot{x}_2 = -K_g \phi_1 \tag{4.98b}$$
$$\dot{\phi}_1 = \phi_2 \tag{4.98c}$$
$$\dot{\phi}_2 = \tau'_\phi \tag{4.98d}$$

où $[x \ \dot{x}]^T = [x_1 \ x_2]^T$ et $[\phi \ \dot{\phi}]^T = [\phi_1 \ \phi_2]^T$. Notons que K_g est constant et que (4.98) représente quatre intégrateurs en cascade. Ainsi, d'après ce qu'il précède, on obtient pour $n = 4$:

$$\tau'_\phi = \frac{1}{K_g}\Big(\sigma_1(k_1 x_1) + \sigma_2(k_2 x_2) - \sigma_3(k_3 K_g \phi_1) - \sigma_4(k_4 K_g \phi_2)\Big) \tag{4.99}$$

où $k_1, k_2, k_3, k_4 > 0$ et $|\sigma_i(\cdot)| \leq b_i \ \forall \ i \in [1, 4]$. En introduisant (4.99) dans (4.98) on obtient que $\forall \ t > T$ suffisamment grand, $\dot{x}, x, \dot{\phi}, \phi$ sont bornés. D'après (4.90) il vient,

$$\det(A) = \lambda^4 + \lambda^3 k_4 + \lambda^2 k_3 + \lambda k_2 + k_1 \tag{4.100}$$

puis, le critère de Routh Hurwitz donne,

$$k_3 k_4 > k_2 \tag{4.101a}$$
$$k_2 k_3 k_4 > k_1 k_4^2 + k_2^2 \tag{4.101b}$$

Afin de satisfaire (4.86) on obtient :

$$b_4 > b_1 + b_2 + b_3 \tag{4.102a}$$
$$b_3 > b_1 + b_2 \tag{4.102b}$$
$$b_2 > b_1 \tag{4.102c}$$
$$\frac{b_1}{k_1} > \frac{k_3}{k_2^2} b_1 + \frac{k_4}{k_2 k_3}(b_1 + b_2) + \frac{b_1 + b_2 + b_3}{k_2 k_4} \tag{4.102d}$$
$$\frac{b_1}{k_2} > \frac{k_4}{k_3^2}(b_1 + b_2) + \frac{b_1 + b_2 + b_3}{k_3 k_4} \tag{4.102e}$$
$$\frac{b_1 + b_2}{k_3} > \frac{b_1 + b_2 + b_3}{k_4^2} \tag{4.102f}$$

En utilisant (4.91) à (4.99) et avec les conditions (4.96), (4.101) et (4.102) on obtient que pour un temps suffisamment grand, $\dot{x}, x, \dot{\phi}, \phi \to 0$.

Simulations

Afin de vérifier la loi de commande proposée, des simulations ont été effectuées sur le modèle (4.97) du *PVTOL*. Les paramètres du modèle choisis pour les simulations sont $J = M = 1$ et $\alpha = 90°$; cela permet d'utiliser le même modèle qu'à la section 4.1.8. Ainsi la loi de commande proposée peut être située par rapport aux autres lois par fonctions de saturations. Les valeurs des gains utilisés sont données dans la table 4.2. Notons qu'ici la loi de commande admet pour borne :

$$|\tau'_\phi| \leq \frac{b_1 + b_2 + b_3 + b_4}{g} = 1.2742 \tag{4.103}$$

Les résultats de la première simulation (identique à celle vue à la section 4.1.8) sont reportés sur les figures 4.14, 4.15 et 4.16, où la loi de commande proposée est nommée "saturations états séparés". Les conditions initiales sont : $x(0) = 5$, $\dot{x}(0) = 0$, $\phi(0) = 0$, $\dot{\phi}(0) = 0$. Les performances obtenues sont comparables aux autres lois de commandes ; notamment la figure 4.16 montre que l'entrée de commande reste dans le même ordre de grandeur que les autres lois. La borne totale de la loi (4.99) (voir (4.103)) n'ayant pourtant pas été choisie identique à celle des deux autres lois (voir (4.52)).

Les résultats de la seconde simulation (identique à celle vue à la section 4.1.8) sont reportés sur les figures 4.17, 4.18 et 4.19. Les conditions initiales sont $x(0) = 25$, $\dot{x}(0) = 0$, $\phi(0) = \pi/6$, $\dot{\phi}(0) = 0$. Les performances de la loi proposée sont toujours bonnes, même si le *PVTOL* se trouve plus éloigné du point d'équilibre. L'entrée de commande sur la figure 4.16 montre encore une fois que les valeurs

Table 4.2 – Paramètres utilisés pour les simulations.

Paramètre	Valeur
k_1	0.06
k_2	0.52
k_3	2.22
k_4	2.2
b_1	1.36
b_2	1.61
b_3	2.98
b_4	6.55

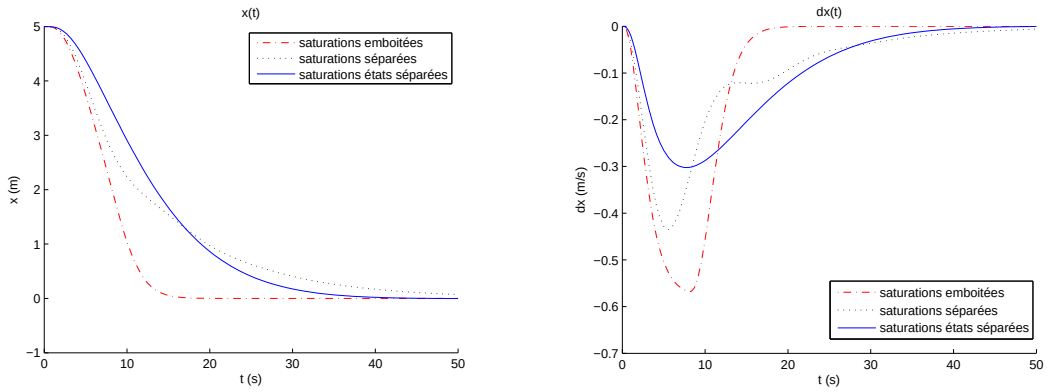


Figure 4.14 – Etats $x(t)$ et $\dot{x}(t)$ du *PVTOL*. Les conditions initiales sont : $x(0) = 5$, $\dot{x}(0) = 0$, $\phi(0) = 0$, $\dot{\phi}(0) = 0$.

des trois lois sont dans le même ordre de grandeur ; par ailleurs l'entrée est bien inférieur à sa borne totale (donnée par (4.52) et (4.103)).

Expériences

Cette section présente les résultats en temps réel obtenus avec la loi de commande (4.99) proposée précédemment, appliqué au *PVTOL* (voir figure 4.20). Une

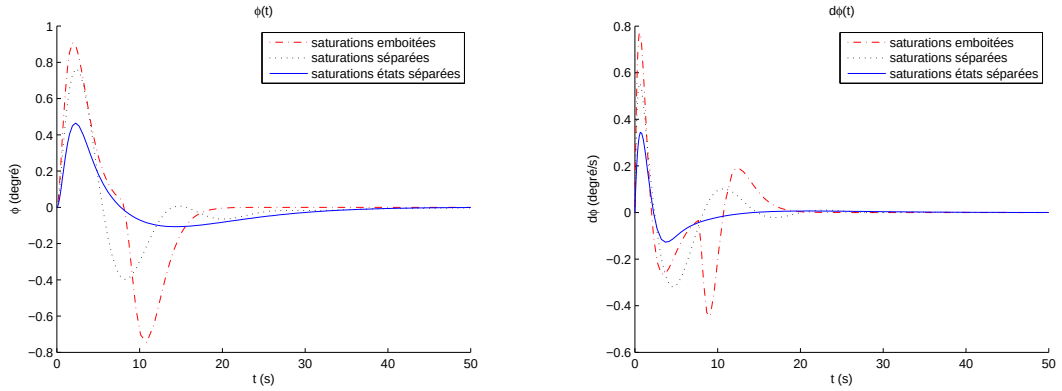


Figure 4.15 – Etats $\phi(t)$ et $\dot{\phi}(t)$ du $PVTOL$. Les conditions initiales sont : $x(0) = 5$, $\dot{x}(0) = 0$, $\phi(0) = 0$, $\dot{\phi}(0) = 0$.

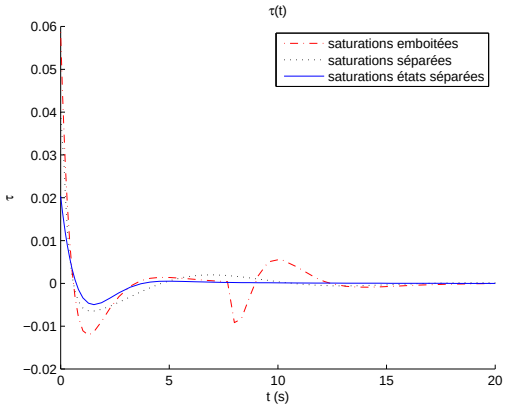


Figure 4.16 – Entrée de contrôle $\tau_\phi(t)$ du $PVTOL$. Les conditions initiales sont : $x(0) = 5$, $\dot{x}(0) = 0$, $\phi(0) = 0$, $\dot{\phi}(0) = 0$.

caméra de type *WebCam* est placée au dessus du plan incliné afin de calculer la position et l'orientation du $PVTOL$; le plan image étant parallèle au plan incliné. L'engin possède deux leds puissantes afin d'aider à sa détection dans l'image. Les données du système de vision sont ensuite envoyées à un PC exécutant *xPC Target* par port série. Celui-ci possède une carte d'entrées/sorties pilotant les moteurs du $PVTOL$. Le schéma du dispositif est montré sur la figure 4.21.

Les paramètres utilisés pour cette expérience ont été réglés manuellement afin d'obtenir de bons résultats. Les figures 4.22 et 4.24 montrent les résultats expérimentaux. Sur les figures 4.22 et 4.23, les déplacements x et y sont donnés en pixels. La consigne du contrôleur est de stabiliser le $PVTOL$ à (x, ϕ)= (200 pixels, 0°) et de suivre une trajectoire en y. Des perturbations manuelles ont été ajoutées afin

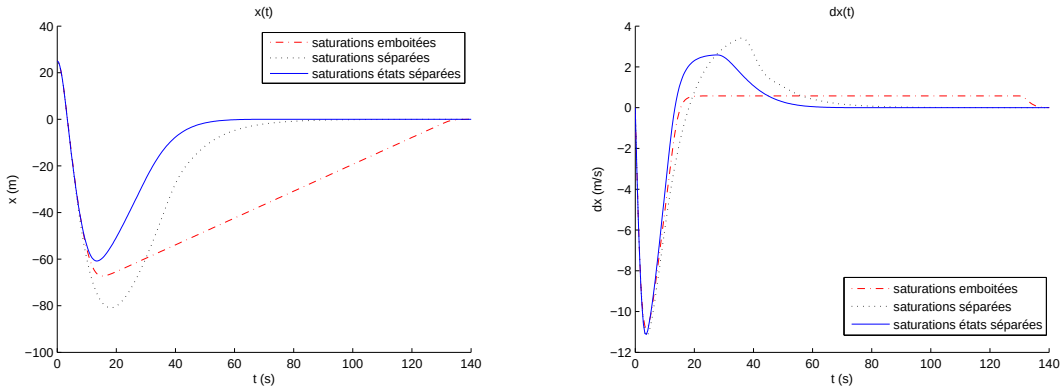


Figure 4.17 – Etats $x(t)$ et $\dot{x}(t)$ du *PVTOL*. Les conditions initiales sont : $x(0) = 25$, $\dot{x}(0) = 0$, $\phi(0) = \pi/6$, $\dot{\phi}(0) = 0$.

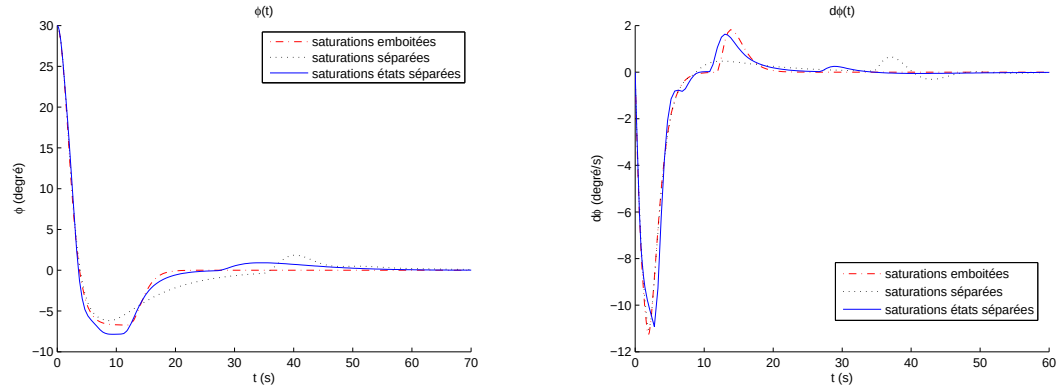


Figure 4.18 – Etats $\phi(t)$ et $\dot{\phi}(t)$ du *PVTOL*. Les conditions initiales sont : $x(0) = 25$, $\dot{x}(0) = 0$, $\phi(0) = \pi/6$, $\dot{\phi}(0) = 0$.

de montrer la robustesse de la loi de commande.

4.2.9 Conclusion

Une loi de commande par fonctions de saturations séparées a donc été présentée. Celle-ci permet de stabiliser un système linéaire ayant n intégrateurs en cascade. Contrairement aux autres lois présentées au début du chapitre, celle proposée utilise séparément une fonction de saturation par état. De manière expérimentale, cette propriété permet de régler plus facilement les gains car chacun ne joue que sur

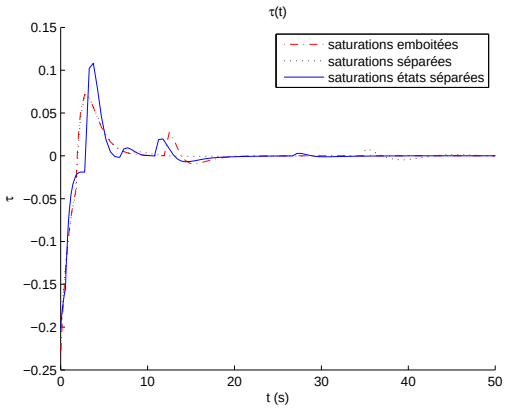


Figure 4.19 – Entrée de contrôle $\tau_\phi(t)$ du $PVTOL$. Les conditions initiales sont : $x(0) = 25$, $\dot{x}(0) = 0$, $\phi(0) = \pi/6$, $\dot{\phi}(0) = 0$.

Figure 4.20 – Plateforme expérimentale.

un seul état. Les coefficients de la loi sont cependant sujets à certaines conditions (voir (4.86)), mais les lois (4.17) et (4.20) apportent elles aussi le même genre de contraintes (voir (4.19) et (4.22)). Les résultats en simulations montrent que la loi proposée à un bon comportement et est comparable aux autres lois en termes de performances (en gardant la borne totale sur l'entrée à un niveau quasi identique). Enfin les résultats expérimentaux sur une plateforme de type $PVTOL$ viennent valider l'utilisation de la loi en temps réel.

C'est donc cette loi de commande qui a été retenue pour stabiliser le quadrirotor lors des expérimentations de cette thèse.

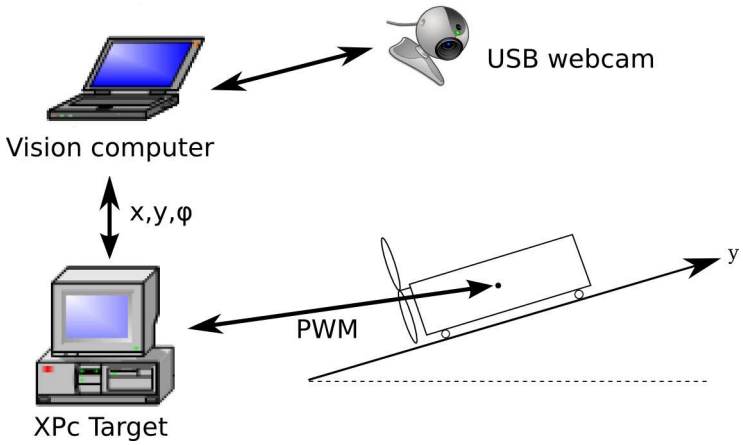


Figure 4.21 – Schéma du dispositif expérimental.

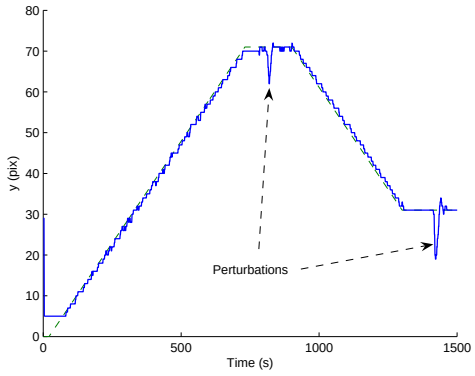


Figure 4.22 – Position y du $PVTOL$.

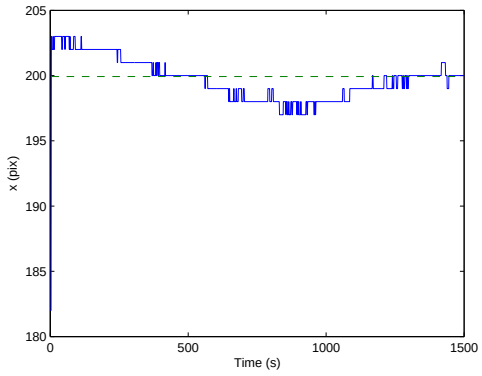


Figure 4.23 – Position x du $PVTOL$.

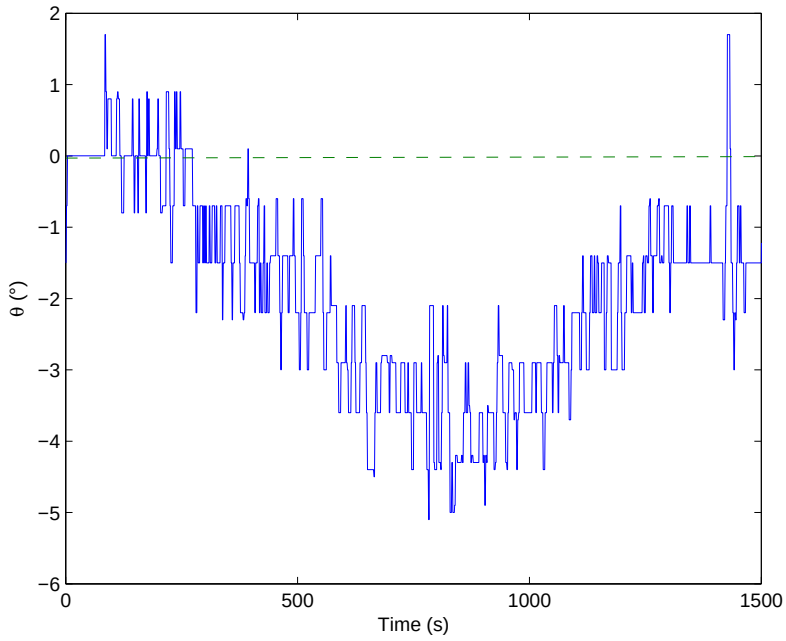


Figure 4.24 – Angle ϕ du $PVTOL$.

Chapitre 5

Estimation et navigation d'un drone en utilisant la vision

Ce chapitre est consacré à la vision. Quelques généralités sont d'abord introduites, notamment le modèle de la caméra perspective, une méthode pour calibrer la caméra et une méthode pour calibrer la caméra par rapport à la centrale inertielle. En effet, les algorithmes de vision devant aider à stabiliser le drone, il est important que l'orientation de la caméra par rapport à la centrale inertielle soit connue afin de pouvoir utiliser les deux informations. Les algorithmes de flux optique et de stéréovision sont ensuite introduits ; enfin quelques applications sur le drone sont présentées. La plupart des notations utilisées dans ce chapitre sont données en annexe B.

5.1 Généralités sur la vision

5.1.1 Modèle de la caméra

Cette section présente le modèle de la caméra perspective ou *pinhole*. Celui-ci est obtenu à partir d'un cas simple de projection, puis est complété au fur et à mesure.

Notations

La caméra (voir schéma 5.1) est caractérisée par un centre de projection C et un plan image parallèle à (C, x, y) situé à une distance f de C ; f est appelée la distance focale. Soit p la projection de C sur le plan image selon l'axe z de la caméra ; p est appelé point principal de la caméra.

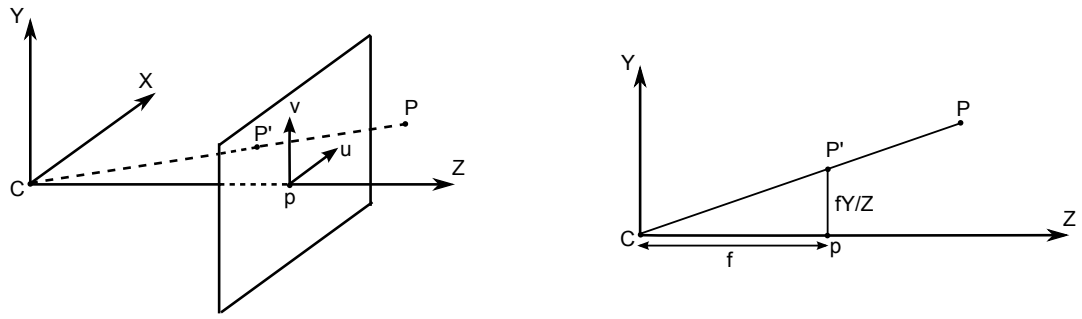


Figure 5.1 – Schéma de la caméra perspective.

Ainsi un point P de coordonnées (X, Y, Z) a pour projeté dans le plan image un point P' de coordonnées $(fX/Z, fY/Z, f)$; cette projection n'est donc pas linéaire. Afin de résoudre ce problème, les points sont exprimés avec leurs coordonnées homogènes. En utilisant cette notation, le projeté du point P dans le repère (p, u, v) du plan image peut s'écrire :

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} \tag{5.1}$$

Cependant, le centre de la caméra ne coïncide pas toujours avec l'origine du repère fixe et le plan image n'est pas toujours parallèle au plan (O, x, y) du repère fixe. Une transformation représentant les paramètres extrinsèques de la caméra doit donc être introduite. De plus, les unités dans le repère de la caméra et dans le repère fixe peuvent être différentes. Une transformation représentant les paramètres intrinsèques de la caméra est alors aussi introduite. La relation (5.1) devient :

$$\begin{bmatrix} u \\ v \\ Z \end{bmatrix} = \begin{bmatrix} \text{transformation} \\ \text{représentant les} \\ \text{paramètres intrinsèques} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \text{transformation} \\ \text{représentant les} \\ \text{paramètres extrinsèques} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5.2}$$

où (u, v, Z) représente les coordonnées homogènes de P' dans l'image (en pixels). En notant K la matrice 3 x 3 représentant les paramètres intrinsèques de la camera et Rt la matrice 4 x 4 représentant les paramètres extrinsèques de la caméra, il s'ensuit :

$$\begin{bmatrix} u \\ v \\ Z \end{bmatrix} = K \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} Rt \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5.3}$$

Les matrices K et Rt sont explicitées dans les deux sections suivantes.

Paramètres intrinsèques

Le paramètre f est défini en unités de longueur or les points de l'image sont en pixels, introduisons donc des facteurs d'échelles f_u (suivant l'axe u) et f_v (suivant l'axe v) en pixels. Ceux-ci peuvent être différents et représentent alors le fait que les pixels d'une image ne sont pas carrés. Soit :

$$u = f_u X/Z \tag{5.4a}$$
$$v = f_v Y/Z \tag{5.4b}$$

L'origine de l'image ne coïncidant pas avec le point principal (voir figure 5.2), les coordonnées du point P' dans le repère (O_c, u, v) de l'image sont en fait :

$$u' = f_u X/Z + p_u \tag{5.5a}$$
$$v' = f_v Y/Z + p_v \tag{5.5b}$$

où p_u et p_v sont les coordonnées du point principal dans l'image. Ce point n'est pas nécessairement au centre de l'image ; cela dépend de la position de l'optique par rapport au capteur. L'origine de l'image quant à elle est généralement en bas à gauche ou en haut à gauche.

Enfin, les axes u et v pouvant ne pas êtres orthogonaux, il est possible d'introduire un terme s représentant cela. La matrice K s'écrit donc :

$$K = \begin{bmatrix} f_u & s & p_u \\ 0 & f_v & p_v \\ 0 & 0 & 1 \end{bmatrix} \tag{5.6}$$

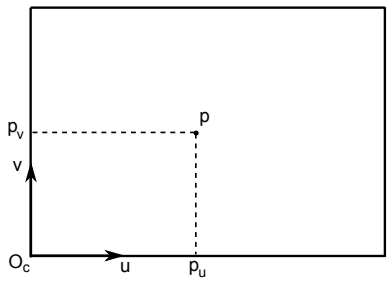


Figure 5.2 – Point principal de la caméra.

Le modèle présenté jusqu'ici ne prend cependant pas en compte les distorsions de l'image dues à la lentille de la caméra. En effet, l'optique placée devant le capteur peut déformer l'image en induisant des distorsions radiales (voir figure 5.3) ou tangentielles (voir figure 5.4). Ainsi, la projection d'une droite n'est plus une droite dans le plan image.

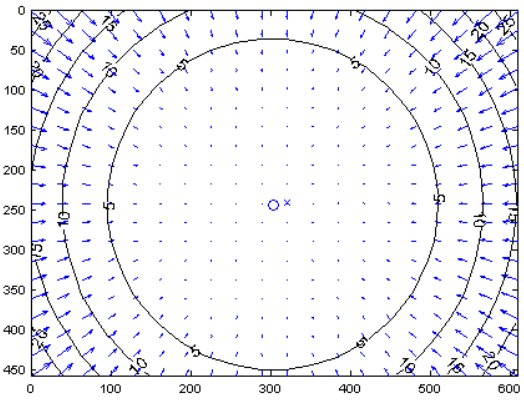


Figure 5.3 – Distorsion radiale de l'image. Source : *Toolbox* de calibration pour *Matlab*, [idtg].

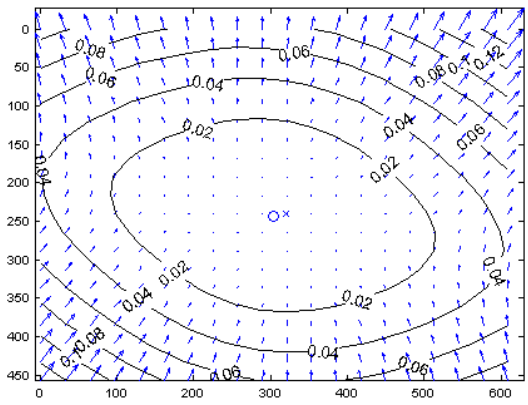


Figure 5.4 – Distorsion tangentielle de l'image. Source : *Toolbox* de calibration pour *Matlab*, [idtg].

Les figures 5.3 et 5.4 montrent le déplacement en pixels induit par la lentille sur une caméra classique. On constate que plus les pixels sont éloignés du point principal, plus ils sont déviés. Le déplacement maximal est de 25 pixels pour la distorsion radiale mais il n'est que de 0.014 pixels dans le cas de la distorsion tangentielle. Cette dernière est donc bien souvent négligeable. Par ailleurs, seuls les termes de degrés 1 et 2 de la distorsion radiale seront ici pris en compte. Des modèles plus complexes existent (voir [WCH92]) mais il a été montré (voir [WM94]) qu'en général le modèle adopté ici est suffisant.

Soient $(x, y, 1)$ et $(x_d, y_d, 1)$ respectivement les coordonnées homogènes idéales (sans distorsion) et réelles (avec distorsion) d'un point de l'image. Alors, un modèle approché des distorsions est donné par [WM94] :

$$x_d = x + x\Big(k_1(x^2+y^2) + k_2(x^2+y^2)^2\Big) \tag{5.7a}$$

$$y_d = y + y\Big(k_1(x^2+y^2) + k_2(x^2+y^2)^2\Big) \tag{5.7b}$$

où k_1 et k_2 sont les coefficients de distorsion radiale ; le centre des distorsions étant le point principal. Notons de même $(u, v, 1)$ et $(u_d, v_d, 1)$ respectivement les coordonnées idéales (sans distorsion) et réelles (avec distorsion) d'un point de l'image en pixels. En utilisant (5.6), il s'ensuit la relation :

$$u_d = p_u + f_u x_d + s y_d \tag{5.8a}$$

$$v_d = p_v + f_v y_d \tag{5.8b}$$

ainsi que :

$$u = p_u + f_u x + s y \tag{5.9a}$$

$$v = p_v + f_v y \tag{5.9b}$$

En utilisant (5.7) et (5.9), les équations (5.8) peuvent se réécrire :

$$u_d = u + (u - p_u)\Big(k_1(x^2+y^2) + k_2(x^2+y^2)^2\Big) \tag{5.10a}$$

$$v_d = v + (v - p_v)\Big(k_1(x^2+y^2) + k_2(x^2+y^2)^2\Big) \tag{5.10b}$$

Les équations précédentes permettent d'obtenir les coordonnées avec distorsion à partir des coordonnées sans distorsions. Cependant, la plupart des applications de vision nécessitent de reprojeter les points d'une image et il est nécessaire d'utiliser la transformation inverse à (5.7). La solution algébrique étant difficile à exprimer, des méthodes d'approximations sont proposées. Ainsi, [Mel94] utilise une technique itérative afin d'estimer les coordonnées sans distorsions à partir des coordonnées réelles. La *toolbox* de calibration de caméra pour *Matlab* retenue pour cette thèse (voir section 5.1.2) est basée sur ces travaux. Elle calcule par itérations les expressions suivantes :

$$r_k = 1 + k_1(x_{k-1}^2 + y_{k-1}^2) + k_2(x_{k-1}^2 + y_{k-1}^2)^2 \tag{5.11a}$$

$$x_k = \frac{x_d}{r_k} \tag{5.11b}$$

$$y_k = \frac{y_d}{r_k} \tag{5.11c}$$

avec comme valeurs initiales :

$$x_0 = x_d \tag{5.12a}$$

$$y_0 = y_d \tag{5.12b}$$

Une vingtaine d'itérations sont alors nécessaires pour arriver au résultat approché. Une fois les coefficients de distorsion identifiés, cet algorithme sera utilisé pour éliminer les effets de la lentille. Les distorsions ne seront donc plus prises en compte dans les applications de vision de cette thèse car elles auront été compensées au préalable.

Paramètres extrinsèques

La figure 5.5 montre la translation et la rotation de la caméra par rapport au repère fixe. Soient (X^c, Y^c, Z^c) les coordonnées d'un point M dans le repère caméra et (X, Y, Z) ses coordonnées dans le repère fixe, on a alors la relation :

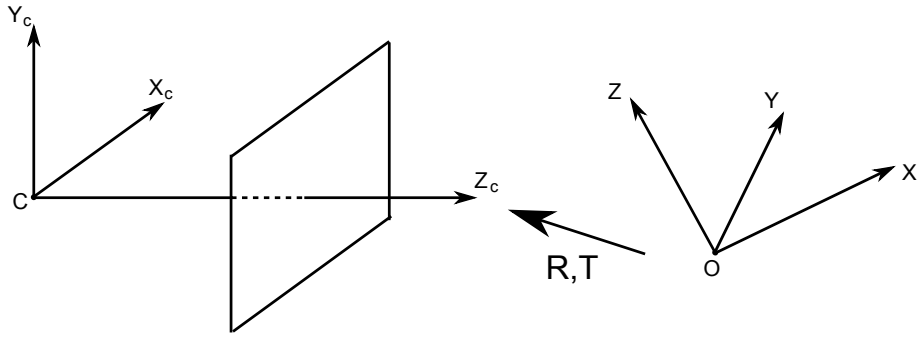


Figure 5.5 – Rotation et translation de la caméra.

$$\begin{bmatrix} X^c \\ Y^c \\ Z^c \end{bmatrix} = R\Big(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - C \Big) \tag{5.13}$$

R étant une matrice de rotation orthonormale telle que $RR^T = I$ et C étant les coordonnées du centre de la caméra dans le repère fixe. En notant $T = -RC$, on obtient en coordonnées homogènes :

$$\begin{bmatrix} X^c \\ Y^c \\ Z^c \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = Rt \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5.14}$$

5.1.2 Calibration de la caméra

Les techniques de calibration de caméra sont nombreuses et se divisent principalement en deux grandes familles. La première famille utilise une mire de calibration dont la géométrie est supposée parfaitement connue. Il s'agit en général de deux ou trois plans orthogonaux (voir [Fau93]). Dans certains cas ([Tsa87]), la mire est composée d'un seul plan auquel est imposée une translation (supposée connue) entre deux images. La seconde famille n'utilise aucun objet particulier ; ce sont des techniques d'auto-calibration (*self-calibration*). La scène observée étant rigide, il est alors possible d'estimer les paramètres de la caméra avec au moins trois images différentes (voir [Har94a] ou [LF97]). Ce type de méthode est donc plus souple, car elle ne nécessite aucun outil supplémentaire. Cependant, selon [Bou98], l'auto-calibration n'est pas toujours fiable à cause du grand nombre de paramètres à estimer.

Cette section décrit une technique de calibration proposée par [Zha99]. Elle se classe dans la première famille ; cependant la mire utilisée est plane et l'orientation de celle-ci entre deux images n'a pas besoin d'être connue. Elle est donc plus souple que les méthodes proposées jusque là, en se rapprochant de la seconde famille sans les problèmes de robustesses. De plus, cette technique est disponible dans une *Toolbox* de calibration de caméra pour *Matlab* (voir [idtg]). Elle a donc été retenue pour sa facilité de mise en œuvre afin de calibrer les différentes caméras utilisées pour cette thèse.

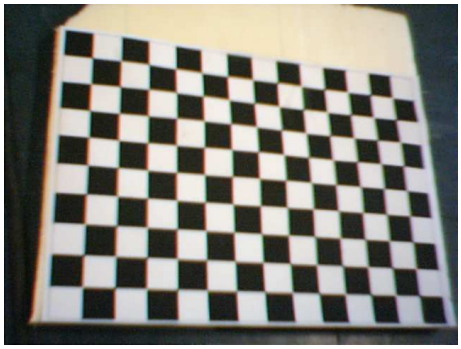


Figure 5.6 – Mire de calibration de la caméra.

La première étape de la calibration consiste à estimer les cinq paramètres intrinsèques en négligeant les distorsions de la caméra. Puis, les paramètres extrinsèques pour chaque image sont calculés. Les paramètres k_1 et k_2 pourront alors être à leur tour estimés. Enfin, l'ensemble des paramètres seront alors affinés par un algorithme d'estimation du maximum de vraisemblance.

Homographie

Pour la calibration, nous utilisons une mire plane à laquelle est attaché le repère fixe. Tous les points M appartenant à cette mire se trouvent donc dans le plan $Z = 0$. En notant $m(u, v)$ le projeté de M sur le plan image, l'équation (5.3) devient :

$$\tilde{m} = K \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \tag{5.15}$$

cette équation peut se réécrire sous la forme :

$$\tilde{m} = K \begin{bmatrix} r_1 & r_2 & T \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{5.16}$$

où r_i est la $i^{\text{ème}}$ colonne de R. En réécrivant $\tilde{M} = [X, Y, 1]^T$, nous pouvons définir l'homographie H telle que :

$$\tilde{m} = H\tilde{M} \tag{5.17}$$

avec,

$$H = \lambda K \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \tag{5.18}$$

H étant une matrice 3 x 3 définie à un facteur d'échelle λ près. En pratique, les points extraits de l'image ne satisfont pas la relation (5.16) à cause du bruit. La matrice H peut alors être déterminée en utilisant un critère du maximum de vraisemblance. En supposant un bruit gaussien de moyenne nulle et de covariance Λ_{m_i} sur les points m_i, H est obtenue en minimisant la fonction suivante (voir [Zha99]) :

$$\sum_i (\tilde{m}_i - \hat{m}_i)^T \Lambda_{m_i}^{-1} (\tilde{m}_i - \hat{m}_i) \tag{5.19}$$

où :

$$\hat{m}_i = \frac{1}{\overline{h}_3{}^T \tilde{M}_i} \begin{bmatrix} \overline{h}_1{}^T & \tilde{M}_i \\ \overline{h}_2{}^T & \tilde{M}_i \end{bmatrix} \tag{5.20}$$

avec $\overline{h}_i$ la $i^{\text{ème}}$ ligne de H. La minimisation peut se faire à l'aide de l'algorithme de Levenberg-Marquardt (voir [Lev44] et [Mar63]) ; une valeur initiale doit cependant être fournie. Pour cela, notons $A = [\overline{h}_1{}^T, \overline{h}_2{}^T, \overline{h}_3{}^T]^T$; l'équation (5.16) devient :

$$\begin{bmatrix} \tilde{M} & 0 & -u\tilde{M} \\ 0 & \tilde{M} & -v\tilde{M} \end{bmatrix} A = 0 \tag{5.21}$$

avec n points, n équations de ce genre peuvent être écrites ; ce qui mène à l'équation $LA = 0$ où L est une matrice $2n$ x 9. La solution A, servant de valeur initiale, est alors le vecteur propre associé à la plus petite valeur propre de $L^T L$.

Paramètres intrinsèques et extrinsèques

Une fois que H est connue pour une image, le fait que r_1 et r_2 soient orthonormaux donne deux relations :

$$h_1^T (K^{-1})^T K^{-1} h_2 = 0 \tag{5.22a}$$

$$h_1^T (K^{-1})^T K^{-1} h_1 = h_2^T (K^{-1})^T K^{-1} h_2 \tag{5.22b}$$

Définissons la matrice A telle que :

$$A = (K^{-1})^T K^{-1} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{12} & A_{22} & A_{23} \\ A_{13} & A_{23} & A_{33} \end{bmatrix} = \begin{bmatrix} \frac{1}{f_u^2} & -\frac{s}{f_u^2 f_v} & \frac{sp_v - p_u f_v}{f_u^2 f_v} \\ -\frac{s}{f_u^2 f_v} & \frac{s^2}{f_u^2 f_v^2} + \frac{1}{f_v^2} & -s\frac{sp_v - p_u f_v}{f_u^2 f_v^2} - \frac{p_v}{f_v^2} \\ \frac{sp_v - p_u f_v}{f_u^2 f_v} & -s\frac{sp_v - p_u f_v}{f_u^2 f_v^2} - \frac{p_v}{f_v^2} & \frac{(sp_v - p_u f_v)^2}{f_u^2 f_v^2} + \frac{p_v^2}{f_v^2} + 1 \end{bmatrix} \tag{5.23}$$

La matrice A étant symétrique, nous pouvons définir un vecteur a la caractérisant :

$$a = [A_{11}, A_{12}, A_{22}, A_{13}, A_{23}, A_{33}]^T \tag{5.24}$$

En décomposant $h_i = [h_{i1}, h_{i2}, h_{i3}]^T$, on peut écrire :

$$h_i^T A h_j = v_{ij}^T a \tag{5.25}$$

avec $v_{ij} = [h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}]^T$. Ainsi, les deux relations (5.22a) et (5.22b) deviennent :

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} a = 0 \tag{5.26}$$

Avec n images différentes, n équations de cette forme peuvent être obtenues ; cela peut s'écrire de la façon suivante :

$$Va = 0 \tag{5.27}$$

où V est une matrice de dimension $2n$ x 6. Étant donné que K a cinq paramètres, il faut au moins trois images pour obtenir une solution de a. Puis, les paramètres intrinsèques de la caméra peuvent être calculés avec a et (5.23). En utilisant K, les paramètres extrinsèques sont déterminés avec (5.18) :

$$r_1 = \frac{1}{\lambda} K^{-1} h_1 \tag{5.28a}$$

$$r_2 = \frac{1}{\lambda} K^{-1} h_2 \tag{5.28b}$$

$$r_3 = [r_1]_\times r_2 \tag{5.28c}$$

$$T = \frac{1}{\lambda} K^{-1} h_3 \tag{5.28d}$$

où $\lambda = \| K^{-1}h_1 \| = \| K^{-1}h_2 \|$. A cause du bruit dans les images, la matrice R n'a pas les propriétés d'une matrice de rotation. Une approximation R^* de R peut donc être obtenue en minimisant le norme de Frobenius :

$$min_{R^*} \|R^* - R\|_F^2 \tag{5.29}$$

avec $R^{*T}R^* = I$.

Coefficients de distorsion

Les paramètres de la caméra étant connus, il est possible de déterminer les coefficients de distorsion k_1 et k_2. D'après les équations (5.10) il s'ensuit pour chaque point, le système :

$$\begin{bmatrix} (u-p_u)(x^2+y^2) & (u-p_u)(x^2+y^2)^2 \\ (v-p_v)(x^2+y^2) & (v-p_v)(x^2+y^2)^2 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} u_d - u \\ v_d - v \end{bmatrix} \tag{5.30}$$

Ou, avec une notation évidente : $Dk = d$. Avec m points et n images, on obtient $2mn$ équations. La solution au sens des moindres carrés est donc :

$$k = (D^T D)^{-1} D^T d \tag{5.31}$$

Estimation du maximum de vraisemblance

L'estimation du maximum de vraisemblance permet d'affiner les valeurs trouvées précédemment. En supposant que les points images sont corrompus par un bruit indépendant et également distribué, le maximum de vraisemblance peut être obtenu en minimisant la fonction suivante (voir [Zha99]) :

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \|m_{ij} - \hat{m}(A, k_1, k_2, R_i, T_i, M_j)\|^2 \tag{5.32}$$

où $\hat{m}(A, k_1, k_2, R_i, T_i, M_j)$ est la projection d'un point M_j dans l'image i selon l'équation (5.16), auquel il a été ajouté les distorsions (5.10). Minimiser (5.32) est un problème de minimisation non linéaire pouvant être résolu avec l'algorithme de Levenberg-Marquardt (voir [Lev44] et [Mar63]). Afin d'initialiser l'algorithme, une première estimation de A et $\{R_i, T_i\}$, $\forall i \in [1, n]$ doit être fournie ; les résultats des sections précédentes peuvent donc être utilisés.

5.1.3 Calibration de la caméra par rapport à la centrale inertielle

Formulation du problème

Afin de pouvoir mettre en relation les données de la caméra avec celles de la centrale inertielle, il est nécessaire d'estimer la transformation entre ces deux repères. Ce type de problème est connu sous le nom de calibration *hand-eye*. Il est issu de la robotique, et plus particulièrement du cas des manipulateurs. En effet lorsqu'une caméra est fixée sur l'actionneur final d'un robot (en général une pince), il est souvent utile de connaitre la transformation entre les deux repères. Ainsi la caméra peut guider les mouvements de la pince ; d'où le nom de *hand-eye*. Le but de la calibration est donc de trouver les matrices de rotation et de translation permettant de passer d'un repère à l'autre. Une approche classique à ce problème est d'utiliser des matrices de transformations homogènes. Notons X la transformation de la caméra à la pince, A_i la transformation de la caméra au repère fixe (celui de la mire par exemple) et B_i la transformation du corps du robot à la pince ; l'indice i indiquant le numéro de la position de l'ensemble. La figure 5.7 illustre cette notation.

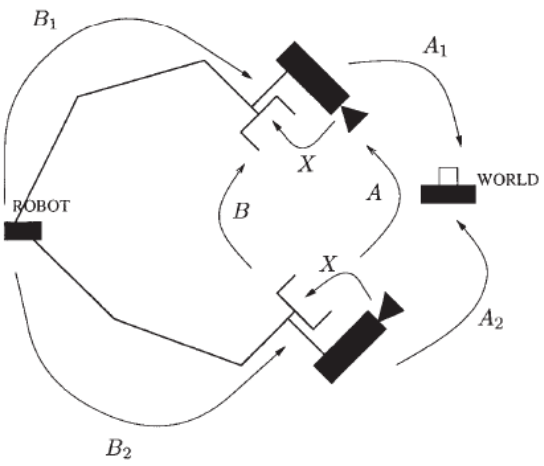


Figure 5.7 – Schéma du problème *hand-eye*. Source : [Dan99].

La matrice A_i est obtenue grâce à la méthode décrite dans la section 5.1.2. La matrice B_i est obtenue directement grâce à la cinématique du robot qui est supposée connue. Pour chaque position, il reste donc deux inconnues : la transformation du corps du robot au repère fixe (de la mire) et la transformation X de la

caméra à la pince. Ce problème est résolu en utilisant le déplacement entre deux positions. Cela mène ([SA89] et [TL89] sont les premiers à avoir établi ce résultat) à l'équation *hand-eye* :

$$AX = XB \tag{5.33}$$

avec $A = A_2 A_1^{-1}$ et $B = B_2^{-1} B_1$. Une matrice de transformation H peut s'écrire de la façon suivante :

$$H = \begin{bmatrix} R_H & T_H \\ 0 & 1 \end{bmatrix} \tag{5.34}$$

où R dénote la rotation et T la translation. L'équation (5.33) peut donc se décomposer de la façon suivante :

$$R_A R_X = R_X R_B \tag{5.35a}$$

$$(R_A - I)T_X = R_X T_B - T_A \tag{5.35b}$$

De nombreuses approches estiment la rotation indépendamment de la translation : en utilisant l'axe de rotation et l'angle ([SA89] et [TL89]), en utilisant les quaternions ([CK91]), ou encore en utilisant une représentation par matrices canoniques ([LB95]). Selon [TL89], le problème peut être résolu avec au moins deux rotations d'axes non colinéaires. Les premiers travaux estimant simultanément la rotation et la translation sont de [Che91], qui introduisit le concept de vis (*screw theory*) au problème *hand-eye*. Une minimisation non linéaire a ensuite été décrite par [HD95], prenant en compte le quaternion de rotation et le vecteur translation. Une méthode utilisant des quaternions duaux unitaires a été introduite par [Dan99] afin de modéliser les déplacements et de réécrire l'équation (5.33). Cela facilite alors la résolution simultanée de la rotation et de la translation.

Cas d'une centrale inertielle

Combiner une caméra avec des capteurs inertiels afin d'obtenir un système de *tracking* temps réel semble une solution intéressante. En effet, cette approche permet de s'affranchir des limitations de chacun des capteurs. Le *tracking* basé sur les informations visuelles est précis et sans dérive pour des mouvements lents, mais il peut échouer à cause d'occlusions ou de la perte des points d'intérêts lors de mouvements rapides. Les capteurs inertiels quant à eux peuvent donner de bons résultats pour des mouvements rapides, mais le bruit et la dérive les rendent inefficaces sur le long terme. De nombreux travaux (voir [KD04] ou [RLG+02] par exemple) montrent ainsi l'aspect complémentaire des deux types de capteurs.

Cependant, afin de pouvoir fusionner les données de ces capteurs et d'obtenir de bons résultats, leur orientation et leur position relative doivent être connues. Le

problème est alors identique à celui du *hand-eye*. La différence étant que dans le cas des robots manipulateurs, les mouvements de translation et de rotation de la pince sont relativement bien connus grâce à la cinématique du robot. Dans le cas d'une caméra et d'une centrale inertielle, la translation de cette dernière est plus difficile à estimer. Ainsi, [KS09] utilise un filtre de Kalman sans parfum (*UKF*) pour estimer à la fois la rotation et la translation entre la caméra et les capteurs inertiels (accéléromètres et gyromètres). L'algorithme nécessite de bouger l'ensemble caméra et capteurs inertiels autour d'une mire de calibration et d'enregistrer images et données inertielles, le filtre étant initialisé avec une mesure manuelle de la translation et de la rotation. L'avantage de cette méthode est qu'elle ne nécessite aucun outil supplémentaire. L'approche de [LP05] est semblable à ce qui a été vu pour la calibration *hand-eye*. Après avoir calibré la caméra et les capteur inertiels, ceux-ci sont traités identiquement comme deux capteurs capables de fournir une orientation par rapport à un repère fixe ; la translation n'étant pas prise en compte. En utilisant la différence de rotation entre deux prises de vues, les auteurs ramènent alors le problème à l'équation (5.35a), qui est résolue en introduisant les quaternions et une fonction à maximiser. Cependant peu d'indices sont donnés pour maximiser cette fonction. Une autre approche pour calibrer la rotation seule est donnée par [ALD03]. Les auteurs ramènent eux aussi le problème à l'équation (5.35a) (sous forme de quaternions unitaires), mais en utilisant le fait que les deux capteurs mesurent un même vecteur : la gravité. La mire de calibration de la caméra doit alors être correctement placée pour qu'un de ses axes coïncide avec le vecteur gravité. L'équation (5.35a) est alors résolue par la méthode de [Hor87], revenant à maximiser une fonction. Il s'agit en fait de trouver le vecteur propre associé à la plus grande valeur propre d'une matrice. Celle-ci étant de dimension 4 x 4, le vecteur propre trouvé représente le quaternion solution. Plus récemment, les mêmes auteurs exposent une technique (voir [LD07]) pour calibrer la translation entre les deux capteurs, une fois la rotation déterminée par la méthode présentée précédemment. Le principe réside dans l'utilisation d'une table tournante sur laquelle est positionné le système à calibrer. Les capteurs inertiels doivent alors être précisément placés pour être au centre de la rotation, afin de n'avoir aucune accélération. Ainsi, la caméra subira la même rotation que les capteurs inertiels et une translation correspondant au bras de levier. Le problème est donc celui de l'équation (5.35b) avec $T_B = 0$. Enfin, les travaux de [HSG08] sont basés sur l'utilisation d'un filtre de Kalman étendu (*EKF*) ; les données inertielles servant à prédire le mouvement de la caméra. Une fonction de coût est introduite, celle-ci étant la somme normalisée des innovations du filtre au carré. La fonction est alors minimisée par rapport à la position et à l'orientation relative des deux capteurs, mais aussi par rapport aux *offsets* des capteurs inertiels. Une estimation de l'orientation relative (par une technique semblable à [ALD03]) est fournie comme valeur

initiale à l'algorithme de minimisation. La valeur initiale de la translation est par contre fournie grâce aux plans du système.

Solution retenue

Certaines des approches sur la calibration caméra/centrale inertielle sont basées sur les données inertielles, ce qui permet par ailleurs d'estimer la translation entre les deux repères. Ces méthodes nécessitent aussi parfois des équipements supplémentaires. Dans notre application, il n'est pas nécessaire de connaitre la translation. Seule la rotation entre le repère de la caméra et le repère de la centrale inertielle est intéressante.

Étant donné que nous utilisons une centrale inertielle, nous connaissons directement son orientation par rapport au repère terrestre ; ainsi que l'orientation de la caméra par rapport au repère fixe de la mire (voir section 5.1.2). Tout comme les travaux de [LP05], le problème se ramène au cas classique de la calibration *hand-eye*. Notons R_{Ii} et R_{Ci} respectivement les rotations de la centrale inertielle et de la caméra, avec i le numéro de la position. Soient $R_I = R_{I2}^{-1} R_{I1}$ et $R_C = R_{C2}^{-1} R_{C1}$ les différences de rotations. L'équation (5.35a) s'écrit alors, pour chaque différence de rotation :

$$R_V R_X = R_X R_I \tag{5.36}$$

En réécrivant cette équation sous forme de quaternions unitaires (et avec une notation évidente), il s'ensuit (voir les formules (B.27) et (B.28)) :

$$\mathring{q}_V \otimes \mathring{q}_X = \mathring{q}_X \otimes \mathring{q}_I \tag{5.37}$$

D'après [Hor87], le quaternion solution $\mathring{q}_X$ est celui maximisant la fonction suivante :

$$S = \sum_{i=1}^{n} (\mathring{q}_V \otimes \mathring{q}_X).(\mathring{q}_X \otimes \mathring{q}_I) \tag{5.38}$$

où n est le nombre de différences de rotations. En utilisant la forme matricielle de la multiplication de quaternions, on obtient :

$$S = \sum_{i=1}^{n} (\mathring{q}_V^g \mathring{q}_X).(\mathring{q}_I^d \mathring{q}_X) = \sum_{i=1}^{n} (\mathring{q}_V^g \mathring{q}_X)^T (\mathring{q}_I^d \mathring{q}_X) = \sum_{i=1}^{n} \mathring{q}_X^T \mathring{q}_V^{gT} \mathring{q}_I^d \mathring{q}_X = \mathring{q}_X^T \sum_{i=1}^{n} (\mathring{q}_V^{gT} \mathring{q}_I^d) \mathring{q}_X \tag{5.39}$$

soit,

$$S = \mathring{q}_X^T N \mathring{q}_X \tag{5.40}$$

avec,

$$N = \sum_{i=1}^{n} \mathring{q}_V^{gT} \mathring{q}_I^d \tag{5.41}$$

D'après [Hor87], le quaternion solution du problème est celui formé par le vecteur propre de N correspondant à la valeur propre la plus élevée. Les valeurs propres étant obtenues en développant et résolvant l'équation :

$$det(N - \lambda I) = 0 \tag{5.42}$$

en notant λ_M la plus grande des quatre valeurs propres (N est de dimension 4 x 4), $\mathring{q}_X$ est alors solution de :

$$[N - \lambda_M]\mathring{q}_X = 0 \tag{5.43}$$

Enfin, la matrice de rotation R_X peut être déduite de $\mathring{q}_X$ avec la formule (B.26).

Conclusion

L'orientation entre la caméra et la centrale inertielle a donc été calibrée dans cette section. Cependant, la centrale inertielle a été placée à la main dans le drone, grâce aux schémas donnés par le constructeur permettant de connaitre la position des différents axes. La position et l'orientation relative n'est donc pas certaine. Nous ne disposons en effet pas d'outils pour effecteur cette calibration avec précision. Il a toutefois été vérifié en tournant manuellement le drone autour d'un axe que les mesures sur les autres axes données par la centrale variaient peu. Par la suite, il sera donc considéré que les mesures de la centrale sont représentatives de l'orientation du drone.

5.2 Asservissement visuel

Deux méthodes d'asservissement visuel sont présentées dans cette section : le flux optique et la stéréovision. Le flux optique (aussi appelé *optical flow*) permet d'estimer le champ de déplacement visuel, c'est aussi un algorithme dit $d2D/dt$ car il calcul le mouvement sur une image. La stéréovision quant à elle permet un asservissement 3D grâce à l'utilisation de deux caméras. L'asservissement 3D est aussi possible avec une seul caméra mais dans ce cas, il faut évoluer dans un environnement parfaitement connu, ce qui est plus contraignant. Les autres types d'asservissements visuels sont le 2D ou le 2D 1/2, mais ne seront pas traités ici.

5.2.1 Flux optique

Généralités

Le flux optique est une estimation du champ dense de déplacement visuel des points entre deux images. Ce dernier est en fait la projection en 2D sur l'image du déplacement des points 3D d'une scène. C'est donc un concept géométrique, qu'il n'est pas toujours possible de mesurer. En effet, le flux optique est en général calculé par une méthode basée sur le gradient supposant que l'intensité d'un point d'une scène est constante (voir section 5.2.1). Le calcul du flux optique ne peut donc pas fonctionner sur des objets uniformes, n'ayant pas de gradient d'intensité. De plus, le problème d'ouverture (voir figure 5.8) ne permet d'estimer le flux optique que dans la direction orthogonale à l'orientation locale de l'objet (voir [Mar82] ou [Mal00]). Cependant, le flux optique est bien souvent confondu avec le champ de déplacement.

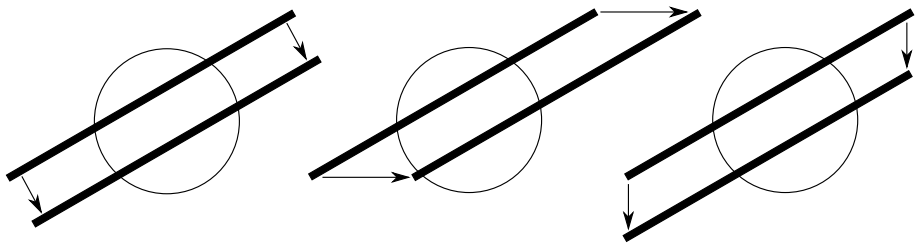


Figure 5.8 – Problème d'ouverture. Le motif homogène n'est vu qu'à l'intérieur du cercle. Dans les trois cas, le déplacement parait alors identique.

Afin de décrire plus simplement le champ de déplacement, considérons un capteur visuel sphérique de rayon unitaire. L'image est alors formée par projection sphérique. Cette caméra est donc différente de celle présentée à la section 5.1.1 ; cependant d'après [NA89], si l'angle de vision de la caméra n'est pas trop large, elle peut être supposée sphérique. Cela permet que tous les points de l'image soient géométriquement équivalents. La figure 5.9 représente ce modèle. Les points sur la sphère sont repérés par leur azimut ψ et leur élévation θ. Un objet est alors caractérisé par sa distance $D(\psi,\theta)$ suivant la direction $d(\psi,\theta)$, ce dernier vecteur étant unitaire. Le champ de déplacement est noté par un vecteur $p(\psi,\theta)$, tangent à la sphère. Les vecteurs T et R représentent respectivement la partie translationnelle et la partie rotationnelle du déplacement de la caméra dans son environnement.

Ainsi, le champ de déplacement est donné par [KD87] :

$$p(\psi,\theta) = -\frac{T - \Big(T^T d(\psi,\theta)\Big) d(\psi,\theta)}{D(\psi,\theta)} - [R]_\times d(\psi,\theta) \tag{5.44}$$

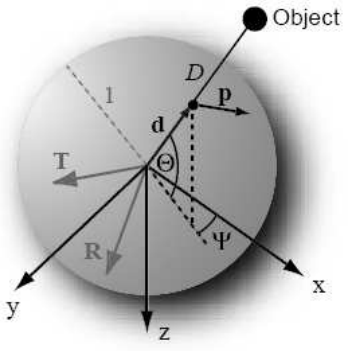


Figure 5.9 – Modèle de la caméra sphérique. Source : [Zuf05].

Cette équation montre que les composantes de translation et de rotation sont découplées dans le calcul du champ de déplacement, ce qui est illustré par la figure 5.10. Par ailleurs la partie rotationnelle est complètement indépendante des distances.

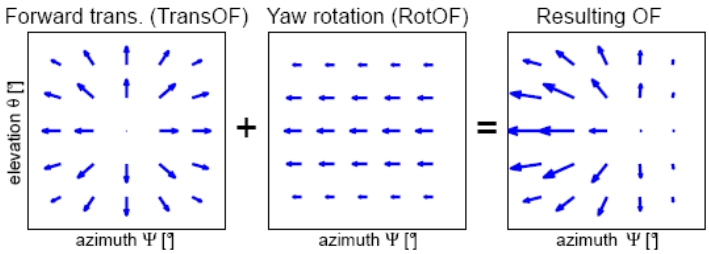


Figure 5.10 – Superposition du flux optique translationnel et rotationnel. La caméra avance en tournant vers un plan. Source : [Zuf05].

Méthodes de calcul du flux optique

Comme vu précédemment, le flux optique peut être calculé par une méthode basée sur le gradient supposant que l'intensité d'un point d'une scène est constante, et que le déplacement est faible entre deux images consécutives [HS81] :

$$I(x, y, t) = I(x + \partial x, y + \partial y, t + \partial t) \tag{5.45}$$

où I est l'intensité et (x, y) la position du point dans l'image. Par simplicité, il sera noté par la suite : $I = I(x, y, t)$, $I_x = \frac{\partial I(x,y,t)}{\partial x}$, $I_y = \frac{\partial I(x,y,t)}{\partial y}$,$I_t = \frac{\partial I(x,y,t)}{\partial t}$. Un

développement de Taylor au premier ordre donne alors :

$$I_x u + I_y v + I_t = 0 \tag{5.46}$$

où u et v sont les composantes suivant x et y de la vitesse de déplacement. En notant $\nabla I = [I_x, I_y]^T$ le gradient de I et $V = [u, v]^T$, il s'ensuit :

$$\nabla I.V + I_t = 0 \tag{5.47}$$

Notons que nous obtenons une équation à deux inconnues. L'équation (5.47) ne permet en fait que de calculer la composante normale au contour (d'intensité constante) du champ de déplacement ; c'est le problème d'ouverture décrit précédemment (voir figure 5.8). Il faut donc une autre condition pour estimer le flux optique. Deux méthodes différentes sont alors brièvement présentées par la suite ; il s'agit des plus utilisées. Celles-ci ont été comparées par [BFB94], ainsi qu'avec d'autres techniques n'étant pas basées sur le gradient mais sur l'énergie, la phase ou la corrélation. Ces dernières ne seront pas abordées dans ce rapport.

Horn et Schunck. La méthode de Horn et Schunck (voir [HS81]) introduit une contrainte de lissage, supposant que le champ de déplacement est régulier. Combinée avec la condition (5.47), le champ de déplacement est obtenu en minimisant la fonction suivante :

$$\int\int (\nabla I.V + I_t)^2 + \lambda^2 \Big(\|\nabla u\|^2 + \|\nabla v\|^2 \Big) dxdy \tag{5.48}$$

où λ est une constante contrôlant l'influence du terme de lissage et $\|\nabla u\|$ et $\|\nabla v\|$ sont respectivement les gradients de u et de v. Des équations itératives sont utilisées pour minmiser (5.48), ce qui mène à :

$$u^{k+1} = \bar{u}^k - I_x \frac{I_x \bar{u}^k + I_y \bar{v}^k + I_t}{\lambda^2 + I_x^2 + I_y^2} \tag{5.49a}$$

$$v^{k+1} = \bar{v}^k - I_y \frac{I_x \bar{u}^k + I_y \bar{v}^k + I_t}{\lambda^2 + I_x^2 + I_y^2} \tag{5.49b}$$

où k dénote le numéro de l'itération, et $\bar{u}^k$ et $\bar{v}^k$ dénotent respectivement les moyennes du voisinage de u^k et v^k. Les vitesses initiales u^0 et v^0 peuvent être prises égales à 0.

Lucas et Kanade. La méthode de Lucas et Kanade (voir [LK81] et [Luc84]) est sans doute la plus populaire des méthodes différentielles. Elle suppose que le

flux (u, v) est constant dans une petite fenêtre de dimension m x m (avec $m > 1$) centrée sur le pixel (x, y). On obtient alors m^2 équations du type (5.47) ; le problème devient alors sur déterminé et peut s'écrire :

$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \\ I_{xn} & I_{yn} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -I_{t1} \\ -I_{t2} \\ \vdots \\ -I_{tn} \end{bmatrix} \tag{5.50}$$

où $n = m^2$. Avec une notation évidente, on obtient :

$$AV = -b \tag{5.51}$$

Une solution au sens des moindres carrés est alors donnée par :

$$V = (A^T A)^{-1} A^T (-b) \tag{5.52}$$

Une fonction de poids est généralement ajoutée afin de donner plus d'influence aux pixels du centre de la fenêtre. De plus, afin de traiter des champs de déplacement élevés une technique dite pyramidale peut être appliquée. Il s'agit d'effectuer plusieurs itérations, en commençant avec une échelle grossière et en affinant petit à petit. Le résultat d'une étape est alors utilisé à l'étape suivante afin de translater l'image de la valeur trouvée.

Applications

De nombreux travaux ont étudié les insectes volants ; les chercheurs étant intrigués par leur étonnante capacité à voler de façon stable, à éviter les obstacles ou à atterrir ; ces insectes ayant d'ailleurs très peu de capteurs à leur disposition. Ainsi, [SZC+00] ont étudié l'atterrissage des abeilles et se sont aperçus que ces insectes utilisent le flux optique pour déterminer le temps avant contact avec le sol. En maintenant ce temps constant, l'atterrissage se fait en douceur. Les travaux de [BS06] portent aussi sur les abeilles, et ont montré que l'insecte régule sa vitesse de vol grâce au flux optique du sol. L'abeille est alors capable de conserver sa vitesse, même en présence de vent.

Ce type de découverte a ouvert une nouvelle voie dans la robotique, le biomimétisme. De nombreux chercheurs ont en effet voulu reproduire dans leurs robots ce qui existait déjà dans la nature et que les insectes savaient faire très bien. L'une des applications les plus populaire est la navigation dans un couloir (*corridor centring*) grâce au flux optique. En contrôlant le cap d'un robot, le flux latéral gauche peut être égalisé avec celui de droite ; le robot se retrouvera alors au centre du

couloir. Ainsi, les travaux de [CR93] ou [DW94] utilisent une caméra pour cela, alors que [SVS95] en utilise deux. De même, [FRS07] et [HHMR09] se servent le flux optique pour maintenir l'altitude d'un drone constante, ou effectuer l'atterrissage automatique.

Mathématiquement, l'équation (5.44) peut être utilisée de différentes façons. Tout d'abord, il est souvent nécessaire de compenser la partie rotationnelle du flux optique, afin d'isoler la partie translationnelle. L'approche la plus courante est d'utiliser des données inertielles, par exemple [ZF05] se sert des données provenant des gyromètres sur un avion, ou [LW05] de la centrale inertielle pour un robot mobile. Toutefois, les drones ayant plus de degrés de libertés, cette opération est généralement plus difficile que pour un robot mobile se déplaçant dans un plan ; ceux-ci ayant d'ailleurs souvent des encodeurs sur leurs roues afin d'aider à estimer le déplacement. Une fois la partie rotationnelle éliminée, la connaissance de la distance à l'obstacle (ou au mur) permet donc d'en déduire la vitesse de translation. Inversement, en connaissant la vitesse de translation il est possible de calculer la distance à l'objet. Ainsi, un drone asservi en altitude (grâce à un télémètre ou à un capteur de pression, voir section 3.5.3) peut estimer ses vitesses de translation latérales grâce à une caméra pointant vers le sol. De plus, s'il possède une caméra frontale, il pourra alors calculer la distance à un obstacle. C'est le schéma que nous avons proposé dans [RFCSS09].

Les travaux de [GSLS03] sont différents, car ils utilisent l'équation (5.44) afin de déterminer les vitesses de rotation angulaires, en supposant connaitre la vitesse de translation. Cette dernière est en fait obtenue à partir des données des accéléromètres. Seuls des résultats en simulations sont montrés, le but étant d'embarquer le système dans un avion. Cependant, il est montré que les vitesses angulaires doivent être faibles pour avoir une bonne précision.

Malgré tout, le calcul du flux optique reste lourd et difficile à faire en embarqué. Les meilleurs résultats ne sont d'ailleurs pas obtenus avec des caméras mais avec des capteurs spécialisés. Par exemple, les travaux de [ZKB+07] utilisent un capteur optique linéaire de 102 pixels associé à un gyromètre, le tout pouvant fonctionner jusqu'à 1000 Hz, et permettant à un petit avion d'éviter les obstacles. Ce système a ensuite été amélioré (voir [BZF09]), en utilisant des capteurs de souris optiques permettant d'avoir le flux optique sur deux dimensions. Une optique doit alors être ajoutée sur le capteur car celui-ci a une focale très courte. Sept capteurs de ce type (voir figure 5.11) sont placés dans l'avion afin d'avoir un angle de vue large.

D'autres travaux portent sur l'utilisation d'*aVLSI* (*Analog Very Large Scale Integration)* afin de déterminer le flux optique. Le *VLSI* est un procédé datant des années 70 permettant de créer des circuits intégrés contenant des milliers de tran-

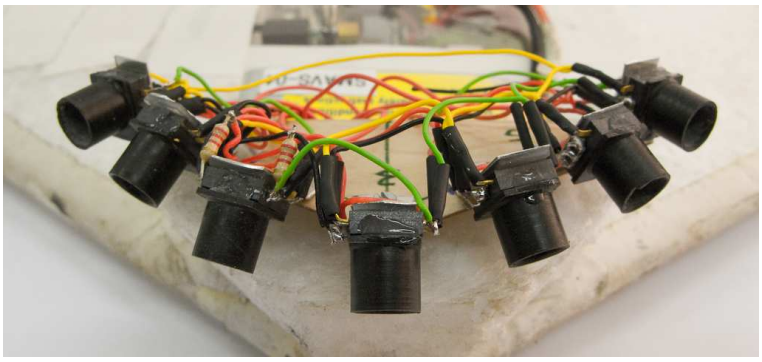


Figure 5.11 – Disposition des capteurs de souris. Source : [BZF09].

sistors sur une seule puce. Le microprocesseur est un *VLSI* mais ce terme est de moins en moins utilisé, les puces intégrant maintenant des milliards de transistors. L'*aVLSI* est une technique similaire mais permettant de combiner des composants analogiques ; voir par exemple [LKI+02] pour plus de détails sur cette technologie. L'avantage de cette technique est donc de pouvoir créer un capteur adapté au besoin. Ainsi, [Sto06] a présenté un prototype de 30 x 30 cellules permettant d'estimer le flux optique (supposé uniquement translationnel). La conception et la fabrication de *aVLSI* reste cependant difficile et nécessite de grandes compétences dans le domaine ; les chercheurs travaillant sur la réalisation d'*aVLSI* pour déterminer le flux optique sont d'ailleurs peu nombreux.

5.2.2 Stéréovision

Généralités

La stéréovision est une technique utilisant deux images prises à des angles différents d'une même scène afin de pouvoir la reconstruire en 3D. On distingue alors deux cas :

- les deux images proviennent de la même caméra ; celle-ci ayant été déplacée entre les deux vues.
- les deux images proviennent de deux caméras différentes ; la position relative de celles-ci étant fixe.

Les deux cas sont en fait géométriquement équivalents. Le premier nécessite cependant de connaitre précisément le déplacement de la caméra entre les deux vues (afin de faire l'appariement de points rapidement) ce qui n'est pas possible dans notre application car le drone ne possède pas cette information. La reconstruction 3D

est possible sans connaitre le déplacement entre les deux vues (photogrammétrie), mais serait beaucoup plus coûteuse en temps de calcul. Le second cas est par contre plus adapté ; il correspond d'ailleurs à ce que fait le cerveau à partir des images de nos deux yeux.

Dans cette section, les deux caméras utilisent le modèle présenté section 5.1.1. Une variable ayant un exposant *prime* représente la seconde caméra. Ainsi, la matrice de calibration intrinsèque de la première caméra est notée K, et celle de la seconde K', chaque caméra pouvant en effet avoir une matrice intrinsèque différente. L'origine du repère fixe sera prise au centre de la première caméra, la seconde caméra étant définie par son déplacement $\begin{bmatrix} R & T \end{bmatrix}$. Les centres des caméras sont supposés distincts, c'est à dire $T \neq 0$.

Il est présenté rapidement par la suite les bases géométriques de la stéréovision. Pour plus de détails, se référer à [HZ03].

Géométrie épipolaire et matrice fondamentale

La géométrie épipolaire est la géométrie projective intrinsèque entre deux vues. Elle est donc indépendante de la scène, et ne dépend que des paramètres internes des caméras et de leur position relative. Son but est d'aider à faire l'appariement de points entre deux vues correspondantes.

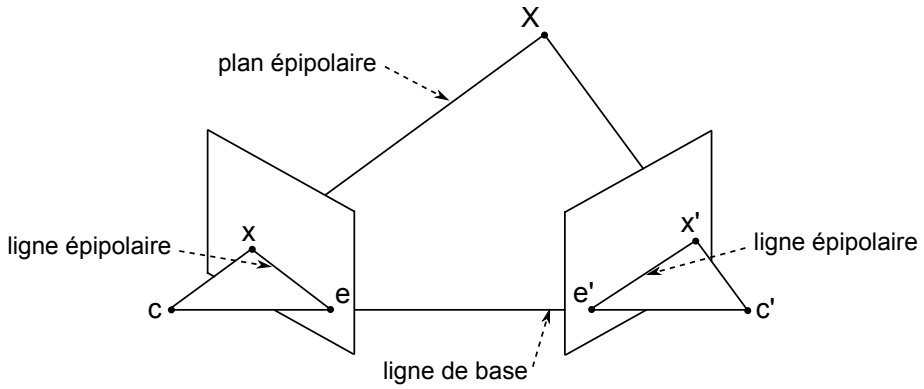


Figure 5.12 – Projection du point 3D sur chacune des caméras.

Soit X un point 3D, ses projections sur chacun des plans images des caméras sont notées x et x', voir figure 5.12. Il apparait ainsi que X, x, x' et les centres des caméras appartiennent au même plan. Nous pouvons alors définir :

- **la ligne de base** est la droite joignant les centres C et C' des caméras.
- **l'épipole** (e) est le point d'intersection entre la ligne de base et le plan image. Si les deux plans images sont parallèles, les épipoles se trouveront à l'infini.

- **un plan épipolaire** (π) est un plan contenant la ligne de base.
- **une ligne épipolaire** (l) est l'intersection d'un plan épipolaire avec le plan image, toutes les lignes épipolaires se croisant à l'épipole.

Ainsi, si l'on connait seulement la projection x d'un point X, la projection correspondante x' sur l'autre plan image devra être cherchée seulement sur une ligne épipolaire. Celle-ci étant formée par le plan épipolaire contenant la ligne de base et x. Si x se trouve sur l'épipole, ce plan n'est pas définissable et x' se trouve sur l'autre épipole. On en déduit que tous les points $3D$ se trouvant sur la ligne de base se projetteront aux mêmes endroits sur les plans images : les épipoles.

La matrice fondamentale F est la représentation algébrique de la géométrie épipolaire. La matrice F permet donc de connaitre la ligne épipolaire l' de la seconde vue correspondant à la projection x de la première vue selon l'équation (voir [XZ96]) :

$$\tilde{l}' = F\tilde{x} \tag{5.53}$$

La matrice F est donc de dimension 3 x 3. Elle peut être déterminée par l'équation suivante :

$$F = [e']_\times P' P^+ \tag{5.54}$$

où $P = K \begin{bmatrix} I & 0 \end{bmatrix}$, $P' = K' \begin{bmatrix} R & T \end{bmatrix}$, et l'exposant "+" représente la pseudo inverse d'une matrice. La projection x' de X appartenant à l', on obtient le résultat suivant pour une paire (x, x') :

$$\tilde{x}'^T F \tilde{x} = 0 \tag{5.55}$$

Les équations (5.54) et (5.55) montrent qu'il est possible de déterminer F de deux manières différentes. En utilisant (5.54), il faut toutefois connaitre les matrices P et P'. L'équation (5.55) par contre ne fait référence qu'à des paires de points. Il est montré par [HZ03] que sept correspondances de points sont au minimum nécessaires.

La matrice F a été définie de telle sorte qu'elle permette de déterminer la ligne épipolaire l' connaissant x. La matrice F^T permet alors de déterminer la ligne épipolaire l connaissant x', soit :

$$\tilde{l} = F^T \tilde{x}' \tag{5.56}$$

Par ailleurs, les épipoles vérifient :

$$F\tilde{e} = 0 \tag{5.57a}$$

$$F^T \tilde{e}' = 0 \tag{5.57b}$$

Détermination de la matrice fondamentale

La matrice fondamentale est définie par l'équation (5.55). En notant $\tilde{x} = (u, v, 1)$, $\tilde{x}' = (u', v', 1)$, et f_{ij} les éléments de F, il s'ensuit :

$$u'uf_{11} + u'vf_{12} + u'f_{13} + v'uf_{21} + v'vf_{22} + v'f_{23} + uf_{31} + vf_{32} + f_{33} = 0 \quad (5.58)$$

en notant $f = (f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33})$ le vecteur formé par les 9 composantes de F, (5.58) s'écrit alors :

$$(u'u, u'v, u', v'u, v'v, v', u, v, 1)f = 0 \quad (5.59)$$

avec n paires de points, on obtient :

$$Af = \begin{bmatrix} u'_1u_1 & u'_1v_1 & u'_1 & v'_1u_1 & v'_1v_1 & v'_1 & u_1 & v_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u'_nu_n & u'_nv_n & u'_n & v'_nu_n & v'_nv_n & v'_n & u_n & v_n & 1 \end{bmatrix} f = 0 \quad (5.60)$$

Cet ensemble d'équations homogènes permet d'obtenir f à un facteur d'échelle prêt. Il faut alors au moins 8 paires de points pour pouvoir résoudre f avec ces équations. Une autre méthode existe (voir [Har94b]), utilisant seulement 7 points et le fait que F soit singulière. Cependant elle peut mener à 3 solutions et reste moins simple ; cette méthode ne sera pas traitée ici.

Avec plus de 8 paires de points, une solution de (5.60) peut être trouvée au sens des moindres carrés. Cependant cela n'assure pas que la matrice F soit singulière. Il faut donc faire une étape supplémentaire, pour forcer cette contrainte. Sinon, les lignes épipolaires déterminées par l'équation (5.56) ne seront pas concourantes au même point. De plus, avant de former les équations linéaires (5.60), il est conseillé d'effectuer une transformation des points de l'image. Cela permet d'améliorer le conditionnement du problème et donc la stabilité du résultat. La transformation proposée par [HZ03] est une translation pour amener le barycentre des points sur l'origine du repère, suivi d'une mise à l'échelle afin que la moyenne quadratique (ou *RMS*, *Root Mean Square)* des points à l'origine soit de $\sqrt{2}$; c'est à dire que le point moyen se trouve à $(1, 1, 1)$ (en coordonnées homogènes). L'algorithme global pour déterminer F est alors donné par [LH81] :

1. **normalisation :** transformer les coordonnées images selon : $\hat{x}_i = T\tilde{x}_i$ et $\hat{x}'_i = T'\tilde{x}'_i$, où T et T' sont des transformations de normalisation (translation et mise à échelle).
2. trouver la matrice $\hat{F}'$ correspondant aux paires $(\hat{x}_i, \hat{x}'_i)$:
   (a) **solution linéaire :** déterminer $\hat{F}$ en résolvant (5.60), avec $\hat{A}$ composée des paires $(\hat{x}_i, \hat{x}'_i)$.

(b) **singularité :** remplacer $\hat{F}'$ par $\hat{F}$ tel que $\det \hat{F}' = 0$, en utilisant une décomposition en valeurs singulières.

3. **dénormalisation :** prendre $F = T'\hat{F}'T$ afin d'obtenir la matrice fondamentale F correspondant aux paires originales (x_i, x_i').

Triangulation

En général, à cause du bruit dans la mesure des points x et x', le point X ne peut pas être déterminé en prenant l'intersection des rayons reprojetés car ceux-ci ne se croisent pas (voir figure 5.13). Il n'y a donc pas de point X satisfaisant $\tilde{x} = P\tilde{X}$ et $\tilde{x}' = P'\tilde{X}$. Cela signifie aussi que la contrainte épipolaire $\tilde{x}F\tilde{x}' = 0$ n'est pas vérifiée.

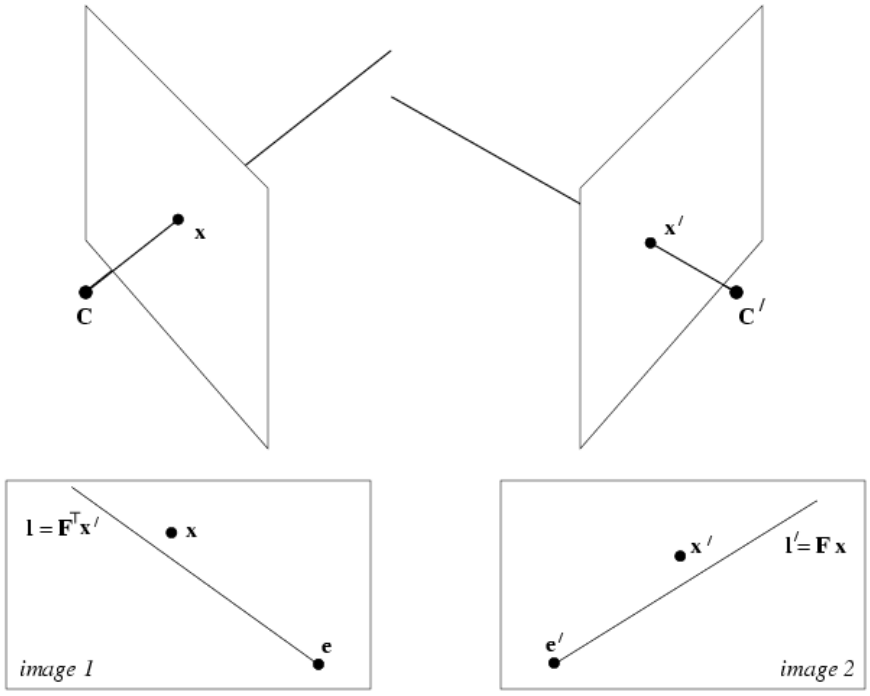


Figure 5.13 – Les rayons reprojetés à partir de mesures bruitées ne se croisent pas. Le point x' n'appartient pas à la ligne épipolaire l' formée grâce à x et inversement. Source : [idtdld].

Une solution optimale au problème de triangulation est donnée par [HZ03]. Il s'agit de trouver une paire de points $(\hat{x}, \hat{x}')$ vérifiant la condition $\hat{x}F\hat{x}' = 0$ et

minimisant la fonction suivante :

$$d(x, \hat{x})^2 + d(x', \hat{x}')^2 \tag{5.61}$$

où d est la distance euclidienne entre deux points. Le point $\hat{X}$ sera alors facilement calculable car les deux rayons se croiseront bien. La minimisation de cette fonction se fait en introduisant un paramètre t définissant l'ensemble des lignes épipolaires $l(t)$ et $l'(t)$. La fonction à minimiser peut alors s'écrire :

$$s(t) = d\Big(x, l(t)\Big)^2 + d\Big(x', l'(t)\Big)^2 \tag{5.62}$$

Deux transformations rigides sont ensuite appliquées sur chaque image afin de placer x et x' à l'origine du nouveau repère (translations T et T') ainsi que les épipoles sur l'axe x (rotations R et R'). Soit, en coordonnées homogènes :

$$\tilde{x} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \tag{5.63a}$$

$$\tilde{x}' = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \tag{5.63b}$$

$$\tilde{e} = \begin{bmatrix} 1 & 0 & f \end{bmatrix}^T \tag{5.63c}$$

$$\tilde{e}' = \begin{bmatrix} 1 & 0 & f \end{bmatrix}^T \tag{5.63d}$$

où f et f' sont deux facteurs d'échelle. D'après (5.57), la matrice F est alors de la forme :

$$\begin{bmatrix} ff'd & -f'c & -f'd \\ -fb & a & b \\ -fd & c & d \end{bmatrix} \tag{5.64}$$

En définissant une ligne épipolaire $l(t)$ comme la droite passant par e et le point de coordonnées $(0, t, 1)$, la fonction à minimiser s'écrit alors (voir [HZ03]) :

$$s(t) = \frac{t^2}{(1 + f^2t^2)^2} - 2\frac{(ad - bc)(at + b)(ct + d)}{\Big((at + b)^2 + f'^2(ct + d)^2\Big)^2} \tag{5.65}$$

Les minimums et maximums de $s(t)$ ont lieu lorsque $s'(t) = 0$, c'est à dire si :

$$t\Big((at + b)^2 + f'^2(ct + d)^2\Big)^2 - (ad - bc)(1 + f^2t^2)^2(at + b)(ct + d) = 0 \tag{5.66}$$

Ce qui correspond à trouver les racines de ce polynôme de degré six. En évaluant la fonction $s(t)$ avec la partie réelle de chacune des racines, il est alors possible de trouver le minimum global $min\Big(s(t)\Big) = s(t_{min})$. Les lignes épipolaires $l(t_{min})$ et $l'(t_{min})$ peuvent ensuite être évaluées ainsi que la paire $(\hat{x}, \hat{x}')$. Enfin, il faut appliquer les transformations inverses ($-T$, $-T'$, R^T et R'^T) afin de retrouver le repère original.

Conclusion

Il a été présenté quelques résultats géométriques de stéréovision. Cependant les méthodes d'appariements des points x et x' n'ont pas été présentées. Celles-ci peuvent être complexes, même si les correspondances sont à chercher au voisinage des lignes épipolaires. Il faut en effet dans un premier temps choisir des points d'intérêts dans la première image qui seront facilement retrouvables dans la seconde.

Les calculs relatifs à la stéréovision peuvent donc être lourds, sachant que l'objectif est d'avoir un système embarqué. L'autre inconvénient est le fait d'avoir deux caméras, dont les images doivent être traitées de façon synchrone. Les solutions de traitement vidéo embarqué les plus performantes présentées dans la section 3.9 ne possèdent en effet qu'un seul bus de capture vidéo dédié. Il faut donc utiliser deux cartes de traitement, ou utiliser des caméras USB au risque de perdre des ressources lors de l'acquisition des images. La synchronisation des caméras est aussi difficile, sauf si la caméra est prévue pour. C'est le cas par exemple de la *uEye* de *IDS-Imaging* qui possède un port *trigger* permettant de relier deux caméras entres elles. La synchronisation des images est alors déclenchée par logiciel.

Afin d'éviter le problème de la faible puissance de calcul à bord, les calculs peuvent être déportés ; mais le problème de la synchronisation reste le même. Dans ce cas, et comme vu précédemment, il faut faire attention à utiliser des transmetteurs sur deux fréquences différentes pour éviter les perturbations. Sinon, des caméras filaires doivent être utilisées ; au risque de déstabiliser le drone.

Malgré tout, la stéréovision reste intéressante car elle permet de reconstruire un environnement 3D et de s'y localiser. Contrairement à des travaux précédents utilisant des marqueurs artificiels pour se repérer (voir [RBL06]), la stéréovision peut utiliser l'environnement naturel pour connaitre la distance à un objet, un mur, etc. Ainsi, deux solutions utilisant des pointeurs lasers sont présentées par la suite (voir sections 5.3.2 et 5.3.3). Celles-ci s'inspirent de la stéréovison dans le sens où les lasers remplacent la seconde caméra en projetant un rayon fixe. L'appariement est alors rapide à faire car le pointeur est facilement reconnaissable dans l'image. Ces solutions sont donc beaucoup plus adaptées aux contraintes de notre système embarqué. Cependant, elles ne permettent de reconstituer qu'un nombre limité de points.

5.3 Applications

Cette section présente quelques applications du flux optique et de la stéréovision. L'objectif étant d'avoir un système embarqué, des simplifications doivent être faites car les algorithme présentés précédemment sont trop lourds.

5.3.1 Évitement d'obstacles par flux optique [1]

Le flux optique peut fournir de nombreuses informations pour la commande d'un drone. Cependant, l'algorithme proposé par [LK81] et [Luc84] a été testé sur la carte de traitement embarquée et n'a pu fonctionner qu'à 1 Hz sur des images de 320x240 pixels. Cette partie présente donc une application simple d'évitement d'obstacles grâce à un algorithme léger de flux optique. Le but de cet exemple est d'étudier la faisabilité de l'application sur un robot mobile avant de pouvoir l'embarquer dans un drone. Le robot se déplaçant dans un plan, le problème est alors simplifié, et il suffit de calculer le flux optique dans une seule direction.

Plateforme

Figure 5.14 – Robot *e-puck*. Source : [MBR+09].

La plateforme utilisée est un robot *e-puck* conçu par l'*EPFL*, voir figure 5.14. Ce robot a l'avantage de posséder un nombre important de capteurs (accéléromètres,

1. Ces travaux ont été effectués dans le cadre d'un stage à l'*Instituto Universitario de Automatica e Informatica Industrial* de l'Université Polytechnique de Valencia, Espagne ; grâce à une bourse octroyée par le gouvernement espagnol.

microphones, caméra, capteurs de proximité, télécommande infra rouge) et d'actionneurs (moteurs pas à pas, haut parleur, LEDs). Il est équipé d'une liaison *bluetooth* afin de communiquer avec un ordinateur ou de charger le programme (via un *bootloader*), le tout étant architecturé autour d'un *dsPIC* de *Microchip*. Celui-ci est un microcontrôleur 16 bits, fonctionnant à 64 MHz. Il a la particularité de posséder une unité *DSP*, lui permettant par exemple d'effectuer efficacement des calculs types produits scalaires ou transformations de Fourier (*FFT*, *Fast Fourier Transform*). Enfin, des extensions sont vendues pour l'*e-puck* ; citons par exemple une caméra omnidirectionnelle (figure 5.15) et une caméra linéaire à large champ de vision (figure 5.16). Cette dernière est intéressante pour le calcul de flux optique. En effet cette caméra est en fait constituée de trois capteurs linéaires *TAOS TSL3301* de 102 pixels, pouvant fonctionner jusqu'à 1 MHz par pixel. Le large champ du capteur permet alors au robot de repérer les obstacles sur le coté. Des travaux (voir [ZKB+07]), ont ainsi montré des applications de flux optique avec ce type de capteurs.

Figure 5.15 – Extension pour *e-puck* : caméra omnidirectionnelle. Source : [MBR+09].

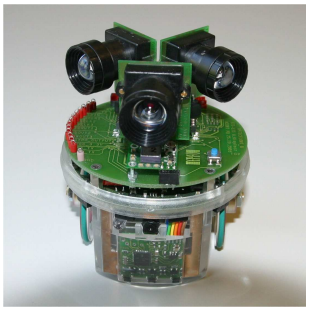


Figure 5.16 – Extension pour *e-puck* : caméra linéaire à large champ de vision. Source : [MBR+09].

Calcul du flux optique

Notre application est donc inspirée de ce qui a été fait sur cette extension de l'*e-puck* ; néanmoins nous n'utilisons ici que la caméra et non les capteurs linéaires. Le microcontrôleur étant limité (en puissance de calcul et en taille mémoire), il ne permet pas de faire du traitement sur des images entières. Il faut donc définir des fenêtres sur lesquelles sera effectué le traitement. Cependant, il est possible

d'utiliser des fonctions de *down sampling* permettant de réduire la résolution d'une fenêtre. Étant donné que nous nous intéressons au flux optique dans une seule direction, la fenêtre choisie fait toute la largeur de l'image (640 pixels), et une hauteur de 16 pixels. La fenêtre est ensuite *down samplée* à une fenêtre de 320x1 pixels.

Afin de déterminer le flux optique, nous utilisons un algorithme proposé par [Zuf05], dérivé de la méthode de [Sri94]. Ce dernier a en effet proposé un algorithme d'interpolation d'image (*I2A*, *Image Interpolation Algorithm*) consistant à faire une interpolation entre une nouvelle image et les images de référence précédentes en une seule étape, sans itérations. Contrairement aux méthodes basées sur le gradient, l'*I2A* joue sur la minimisation d'erreur. Cet algorithme a donc été simplifié par [Zuf05], pour effectuer le traitement sur une dimension seulement. Il suppose que la transformation entre deux images consécutives est une translation pure ; l'expansion de l'image ou d'autres déformations n'étant pas prises en comptes par l'algorithme. Ainsi, en notant $I(n,t)$ le niveau de gris du n^{ime} pixel d'une image à l'instant t et s la translation entre deux images, alors $I(n,t+\Delta t)$ peut être approximé par $\hat{I}(n,t+\Delta t)$:

$$\hat{I}(n,t+\Delta t) = I(n,t) + s\frac{I(n-k,t) - I(n+k,t)}{2k} \tag{5.67}$$

l'estimation est alors une combinaison linéaire de l'image de référence et de deux versions translatées de $\pm k$. Le déplacement s est ensuite calculé en minimisant l'erreur E entre l'image estimée et l'image réelle par rapport à s :

$$E = \sum_n \Big(I(n,t+\Delta t) - \hat{I}(n,t+\Delta t)\Big)^2 \tag{5.68}$$

le minimum est atteint pour $\frac{dE}{ds} = 0$ soit,

$$s = 2k\frac{\sum_n \Big(I(n,t+\Delta t) - I(n,t)\Big)\Big(I(n-k,t) - I(n+k,t)\Big)}{\sum_n \Big(I(n-k,t) - I(n+k,t)\Big)^2} \tag{5.69}$$

L'amplitude de déplacement k doit donc être choisie telle que sur l'intervalle de temps Δt, le déplacement ne dépasse pas $\pm k$. De plus, l'algorithme suppose que le flux optique est constant sur l'image considérée ; sinon il faut découper l'image en plusieurs régions et appliquer l'*I2A* à chacune d'entres elles.

Résultats

Le but de l'application est que le robot *e-puck* puisse éviter des obstacles. Ainsi, il doit être capable de détecter un mur et de tourner pour l'éviter. Pour cela

nous avons découpé la fenêtre d'acquisition de l'image en deux régions (gauche et droite). Ainsi, la zone ayant un flux plus élevé correspondra à la zone la plus proche du robot ; le robot pourra alors prendre la décision optimale (virage à gauche ou à droite) pour éviter le mur. Enfin, l'algorithme fonctionnant sur des images de basse résolution, il est préférable de fournir un environnement texturé composé de bandes noires et blanches. L'algorithme s'avère aussi fonctionner dans des endroits plus naturels, mais les résultats restent parfois aléatoires. Un essai pour un virage à droite est montré sur la figure 5.17 où le déplacement est décomposé en plusieurs images.

Figure 5.17 – Évitement d'un mur.

Le robot n'étant pas équipé de gyromètres, il est difficile de compenser le flux rotationel lorsque le robot tourne sur lui même. Ainsi, l'amplitude de la rotation est calculée avant d'amorcer le virage, proportionnellement à la valeur du flux optique. Le robot ne prend alors plus en compte les informations de la caméra

jusqu'à la fin de la rotation.

Conclusion

Les résultats obtenus montrent la faisabilité de l'évitement d'obstacles par flux optique. Notre cas est cependant simplifié au maximum, afin de pouvoir faire tourner l'algorithme sur le microcontrôleur du robot. Cette configuration pourrait cependant être adaptée au drone, en utilisant deux caméras (vers le sol et vers un mur par exemple), chacune calculant le flux optique sur une dimension grâce à l'*I2A* (voir figure 5.18). Les vitesses de déplacement pourraient alors être estimées, en mesurant aussi la distance au sol et au mur. Cependant, ce dispositif nécessite de compenser la partie rotationnelle du flux, à partir des mesures des gyromètres par exemple.

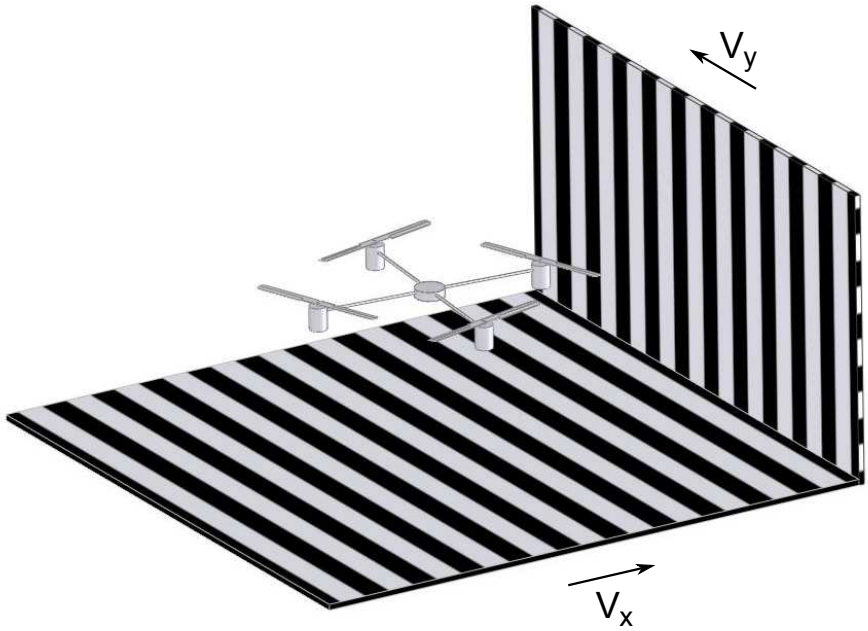


Figure 5.18 – Application au drone. Une caméra pointant vers le sol permet d'estimer la vitesse V_x ; une autre pointant vers le mur permet d'estimer la vitesse V_y.

5.3.2 Estimation de l'orientation avec un système de pointeurs lasers

En se basant sur le principe de la stéréovision, nous proposons dans cette partie un système de vision composé de trois pointeurs lasers. Ceux-ci forment trois points sur un sol supposé plan et horizontal, voir figure 5.19. L'algorithme de vision peut alors calculer les angles de roulis et tangage ainsi que l'altitude de la caméra (et donc du drone) par rapport au sol.

Un point ou un vecteur sera noté ici avec un exposant "c" s'il est exprimé dans le repère de la caméra, et avec un exposant "p" s'il est exprimé dans le repère de la mire.

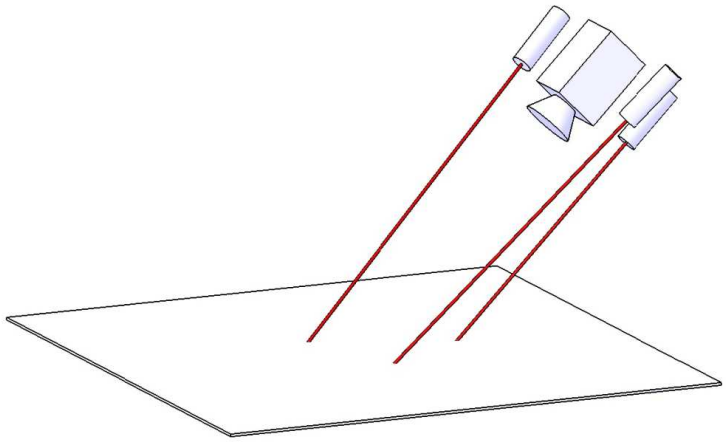


Figure 5.19 – Schéma du système de vision et des trois pointeurs lasers.

Calibration du laser dans le repère de la caméra

L'objectif est de trouver un point P_l^c appartenant au rayon laser et un vecteur V_l^c définissant sa direction. Une technique similaire à celle décrite pour calibrer la caméra (voir section 5.1.2) a été utilisée.

Une fois le laser fixé par rapport à la caméra, des images de la mire de calibration sont prises. Pour chaque image i, les paramètres extrinsèques (R_i, T_i) peuvent être calculés et un point laser (voir figure 5.20) $P_{li}^p = [x_{li}^p, y_{li}^p, 0]^T$ peut être défini. Les coordonnées de ce point dans le repère de la caméra , $P_{li}^c = [x_{li}^c, y_{li}^c, z_{li}^c]^T$, sont données pour chaque image i par la formule suivante :

$$P_{li}^c = R_i P_{li}^p + T_i \tag{5.70}$$

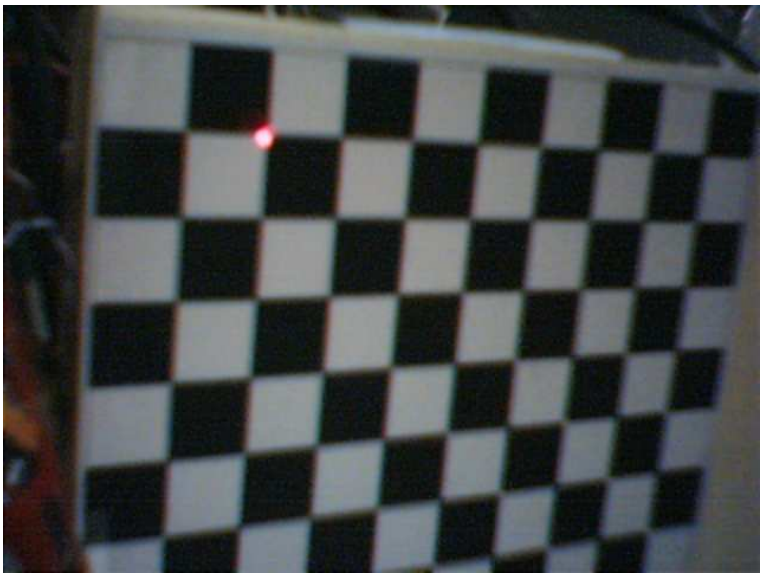


Figure 5.20 – Mire de calibration du laser.

Avec une image, il est possible de trouver un point P_l^c appartenant au rayon laser ; avec deux images sa direction V_l^c peut être définie. En utilisant n images (et donc n points), une régression linéaire en trois dimensions permet de trouver la ligne approchant au mieux l'ensemble de points. Cette régression peut se faire à l'aide de *Matlab* et de l'analyse de composantes principales (*Principal Components Analysis*, voir [Pea01]). Cette opération est effectuée afin d'améliorer l'imprécision due au bruit, notamment lors de la mesure des points P_{li}^p.

Ligne épipolaire

De même que pour la stéréovision, il est possible de définir une ligne épipolaire sur le plan image de la caméra. La position et l'orientation du laser étant fixes, cette ligne est donc fixe aussi. La détermination de son équation permettra par la suite de faire la recherche du point laser dans l'image plus rapidement.

Soit p_{li} un point dans l'image, correspondant à la projection d'un point P_{li}^c. L'ensemble des points p_{li} appartiennent donc à la ligne épipolaire l ; ils vérifient donc l'équation :

$$\tilde{p}_{li}^T \tilde{l} = 0 \tag{5.71}$$

Deux points sont nécessaires au minimum pour déterminer $\tilde{l}$; avec n points, il s'ensuit :

$$\begin{bmatrix} \tilde{p}_{l1}^T \\ \tilde{p}_{l2}^T \\ \vdots \\ \tilde{p}_{ln}^T \end{bmatrix} \tilde{l} = 0 \tag{5.72}$$

ou, avec une notation évidente :

$$A\tilde{l} = 0 \tag{5.73}$$

une solution de $\tilde{l}$ est alors le vecteur propre associé à la plus petite valeur propre de $A^T A$.

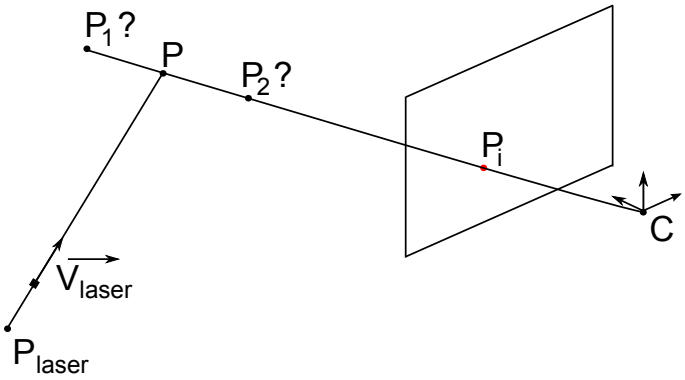
Triangulation

Une fois la caméra et le laser calibrés, calculer la distance de la caméra au point laser est un problème de triangulation, de même que pour la stéréovision (voir section 5.2.2). Le but est donc ici de trouver l'intersection de deux rayons : celui provenant du laser (P_l^c, V_l^c) et celui provenant du centre optique de la caméra C passant par le point P' vu dans l'image (voir figure 5.21).

En pratique, à cause du bruit, les deux rayons ne se croisent pas. La solution optimale à ce problème a été décrite précédemment (voir section 5.2.2), cependant elle a l'inconvénient d'être assez lourde. Le but de cette application étant de fonctionner en temps réel dans la carte embarquée, une solution plus simple a été retenue. Elle consiste à choisir le *midpoint*, c'est à dire le milieu de la perpendiculaire commune aux deux rayons (voir figure 5.21). Une comparaison de différentes méthodes de triangulations a été faite par [HS97], et montre que la solution *midpoint* n'est pas la meilleure mais c'est celle qui est la plus légère en calculs.

Le *midpoint* des deux rayons est le point minimisant la somme des distances de ce point à chacun des rayons. Les coordonnées du *midpoint* seront d'abord exprimées dans le cas général. Ainsi, si chaque rayon i est caractérisé par un vecteur unitaire $V_i = [a_i, b_i, c_i]^T$ et un point $P_i = [x_i, y_i, z_i]^T \ \forall i \in [1, 2]$; alors la distance d_i au carré entre un point $P_0 = [x_0, y_0, z_0]^T$ et le rayon i est donnée par :

$$\begin{aligned} d_i^2 \ = \ & x_0^2(b_i^2 + c_i^2) + y_0^2(a_i^2 + c_i^2) + z_0^2(a_i^2 + b_i^2) - 2x_0y_0a_ib_i - 2x_0z_0a_ic_i - 2y_0z_0b_ic_i \\ & +2x_0(b_ik_{i3} - c_ik_{i2}) + 2y_0(c_ik_{i1} - a_ik_{i3}) + 2z_0(a_ik_{i2} - b_ik_{i1}) \\ & +k_{i1}^2 + k_{i2}^2 + k_{i3}^2 \end{aligned} \tag{5.74}$$



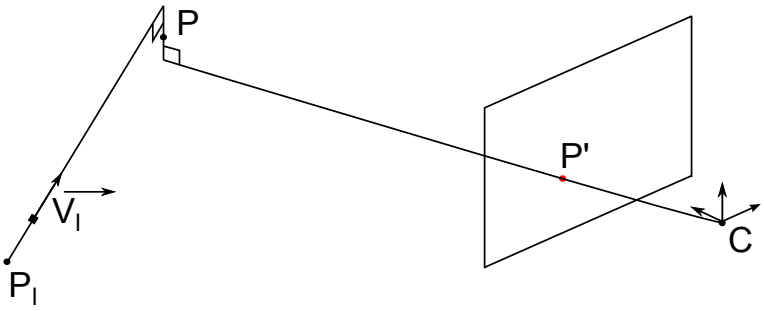


Figure 5.21 – Laser et caméra. P' est la projection du point laser P dans le plan image.

avec

$$k_{i1} = z_i b_i - y_i c_i \tag{5.75a}$$
$$k_{i2} = x_i c_i - z_i a_i \tag{5.75b}$$
$$k_{i3} = y_i a_i - x_i b_i \tag{5.75c}$$

Le *midpoint* est le point minimisant la somme $D = {d_1}^2 + {d_2}^2$. Ainsi, en dérivant

D par rapport à x_0, y_0 et z_0, il s'ensuit :

$$\frac{\partial D}{\partial x_0} = \sum_{i=1}^{2} 2x_0(b_i^2 + c_i^2) - \sum_{i=1}^{2} 2y_0 a_i b_i - \sum_{i=1}^{2} 2z_0 a_i c_i + \sum_{i=1}^{2} 2(b_i k_{i3} - c_i k_{i2}) \quad (5.76a)$$

$$\frac{\partial D}{\partial y_0} = \sum_{i=1}^{2} 2y_0(a_i^2 + c_i^2) - \sum_{i=1}^{2} 2x_0 a_i b_i - \sum_{i=1}^{2} 2z_0 b_i c_i + \sum_{i=1}^{2} 2(c_i k_{i1} - a_i k_{i3}) \quad (5.76b)$$

$$\frac{\partial D}{\partial z_0} = \sum_{i=1}^{2} 2z_0(a_i^2 + b_i^2) - \sum_{i=1}^{2} 2x_0 a_i c_i - \sum_{i=1}^{2} 2y_0 b_i c_i + \sum_{i=1}^{2} 2(a_i k_{i2} - b_i k_{i1}) \quad (5.76c)$$

Pour trouver le minimum, les équations précédentes sont rendues égales à zéro. En réécrivant (5.76) sous forme matricielle :

$$v = BP_0 \quad (5.77)$$

avec,

$$v = \begin{bmatrix} \sum_{i=1}^{2}(c_i k_{i2} - b_i k_{i3}) \\ \sum_{i=1}^{2}(a_i k_{i3} - c_i k_{i1}) \\ \sum_{i=1}^{2}(b_i k_{i1} - a_i k_{i2}) \end{bmatrix} \text{ et } B = \begin{bmatrix} \sum_{i=1}^{2}(b_i^2 + c_i^2) & -\sum_{i=1}^{2} a_i b_i & -\sum_{i=1}^{2} a_i c_i \\ -\sum_{i=1}^{2} a_i b_i & \sum_{i=1}^{2}(a_i^2 + c_i^2) & \sum_{i=1}^{2} b_i c_i \\ -\sum_{i=1}^{2} a_i c_i & -\sum_{i=1}^{2} b_i c_i & \sum_{i=1}^{2}(a_i^2 + b_i^2) \end{bmatrix} \quad (5.78)$$

Le *midpoint* P_0 est alors donné par :

$$P_0 = B^{-1}v \quad (5.79)$$

Dans l'application du laser et de la caméra, le centre de la caméra est $C^c = [0, 0, 0]^T$ et la projection P' du point laser P dans le plan image est :

$$P'^c = K^{-1} \begin{bmatrix} l_x \\ l_y \\ 1 \end{bmatrix} \quad (5.80)$$

où l_x et l_y représentent la projection du point laser dans l'image (en pixels). Les

expressions de P_1^c, V_1^c, P_2^c et V_2^c deviennent alors :

$$P_1^c = [x_1, y_1, z_1]^T = [0, 0, 0]^T \tag{5.81a}$$
$$V_1^c = [a_1, b_1, c_1]^T = \frac{P'^c}{\sqrt{(P'^c)^T P'^c}} \tag{5.81b}$$
$$P_2^c = [x_2, y_2, z_2]^T = P_l^c \tag{5.81c}$$
$$V_2^c = [a_2, b_2, c_2]^T = V_l^c \tag{5.81d}$$

Le midpoint P_0^c est donné par :

$$P_0^c = B^{-1} v \tag{5.82}$$

avec,

$$v = \begin{bmatrix} c_2 k_{22} - b_2 k_{23} \\ a_2 k_{23} - c_2 k_{21} \\ b_2 k_{21} - a_2 k_{22} \end{bmatrix} \text{ et } B = \begin{bmatrix} \sum_{i=1}^{2}(b_i^2 + c_i^2) & -\sum_{i=1}^{2} a_i b_i & -\sum_{i=1}^{2} a_i c_i \\ -\sum_{i=1}^{2} a_i b_i & \sum_{i=1}^{2}(a_i^2 + c_i^2) & \sum_{i=1}^{2} b_i c_i \\ -\sum_{i=1}^{2} a_i c_i & -\sum_{i=1}^{2} b_i c_i & \sum_{i=1}^{2}(a_i^2 + b_i^2) \end{bmatrix} \tag{5.83}$$

Calcul de l'orientation et de l'altitude

Grâce à la triangulation vue précédemment, les coordonnées des trois points lasers dans le repère de la caméra sont désormais connues. Ces points seront notés $P_i^c = [P_{xi}^c, P_{yi}^c, P_{zi}^c]^T$ avec $i \in [1, 3]$. Nous proposons maintenant une technique permettant d'obtenir l'orientation et l'altitude du drone.

Dans cette partie, les angles seront supposés appartenir à l'intervalle $]-\pi/2, \pi/2[$. En effet, le quadrirotor retenu pour les expériences ne peut pas voler si les angles sont en dehors de ces bornes ; par ailleurs le système de vision ne pourrait pas fonctionner car les points formés par les lasers ne seraient pas tous sur le sol.

Afin de calculer les angles, introduisons les matrices de rotations des angles de roulis et tangage. L'angle de roulis ϕ représente la rotation autour de l'axe X, et

l'angle de tangage θ autour de l'axe Y. Les matrices correspondantes sont donc :

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{5.84a}$$

$$R_\theta = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{5.84b}$$

Introduisons la matrice des cosinus directeurs (voir section 3.1.1), permettant de passer d'un système de coordonnées à un autre. Celle-ci est donc donnée par l'expression suivante :

$$R_{fc} = R_\theta R_\phi = \begin{bmatrix} \cos\theta & \sin\theta\sin\phi & \cos\phi\sin\theta \\ 0 & \cos\phi & -\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{5.85}$$

La relation entre un vecteur exprimé dans le repère fixe (V^f) et le même vecteur exprimé dans le repère de la caméra (V^c) est donnée par :

$$V^f = R_{fc} R_X V^c \tag{5.86}$$

où R_X est la matrice traduisant l'orientation entre la caméra et la centrale inertielle (voir section 5.1.3).

En introduisant $V_1^c = P_1^c - P_2^c$ et $V_2^c = P_1^c - P_3^c$, un vecteur normal et unitaire au sol peut s'exprimer :

$$N^c = \begin{bmatrix} N_x^c \\ N_y^c \\ N_z^c \end{bmatrix} = \frac{[V_1^c]_\times V_2^c}{\sqrt{\left([V_1^c]_\times V_2^c\right)^T \left([V_1^c]_\times V_2^c\right)}} \tag{5.87}$$

N^c est donc défini si et seulement si V_1^c et V_2^c ne sont pas colinéaires ; c'est à dire si P_1^c, P_2^c et P_3^c ne sont pas alignés. Le placement des lasers est discuté dans la partie suivante pour éviter ce problème. Le système devant mesurer seulement des angles entre $]-\pi/2, \pi/2[$, si la composante N_z^c de N^c est négative alors le vecteur opposé sera choisi ($N^c = -N^c$). Ainsi,

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = R_{fc} R_X N^c \tag{5.88}$$

En notant $N^b = R_X N^c = [N_x^b, N_y^b, N_z^b]^T$ le vecteur normal au sol dans le repère de la centrale inertielle, il s'ensuit,

$$N_x^b \cos\theta + N_y^b \sin\theta \sin\phi + N_z^b \cos\phi \sin\theta = 0 \tag{5.89a}$$
$$N_y^b \cos\phi - N_z^b \sin\phi = 0 \tag{5.89b}$$
$$-N_x^b \sin\theta + N_y^b \cos\theta \sin\phi + N_z^b \cos\theta \cos\phi = 1 \tag{5.89c}$$

En utilisant (5.89b) :

$$\phi = \arctan\left(\frac{N_y^b}{N_z^b}\right) \tag{5.90}$$

En utilisant (5.89a) et (5.89b) :

$$\theta = \arctan\left(\frac{-N_x^b}{N_y^b \sin\phi + N_z^b \cos\phi}\right) \tag{5.91}$$

Si $N_z^b = 0$ alors $\phi = \pm\pi/2$ ou $\theta = \pm\pi/2$, cela n'est donc pas possible avec les hypothèses faites auparavant. Les angles θ et ϕ sont donc toujours définis.

En général, la distance D entre un point (x_A, y_A, z_A) et un plan $ax+by+cz+d = 0$ est donnée par :

$$D = \frac{|ax_A + by_A + cz_A + d|}{\sqrt{a^2 + b^2 + c^2}} \tag{5.92}$$

En définissant l'altitude z comme la distance entre le centre de la caméra $(0, 0, 0)$ et le plan du sol, il s'ensuit (le vecteur N^c étant normal au sol) :

$$z = \frac{|d|}{\sqrt{N_x^{c2} + N_y^{c2} + N_z^{c2}}} \tag{5.93}$$

le point P_1 appartenant au plan du sol, le coefficient d peut être déterminé. Par ailleurs N^c est un vecteur unitaire, d'où :

$$z = |N_x^c P_{x1}^c + N_y^c P_{y1}^c + N_z^c P_{z1}^c| \tag{5.94}$$

Placement des lasers

Cette partie présente la méthode que nous avons développé pour placer efficacement les lasers. En effet, il a été vu précédemment que l'ensemble des mesures

possibles est sur une ligne épipolaire l. Le système peut donc calculer n distances différentes, où n représente la longueur en pixels de l. Cette longueur est d'ailleurs maximale si l est sur une diagonale de l'image.

Il est important de bien définir une plage de mesure afin de maximiser la précision. La figure (5.22) montre en effet la précision en fonction de l'altitude. La précision étant définie comme la valeur absolue de l'écart entre deux mesures consécutives. Ces mesures sont obtenues à partir des pixels appartenant à l et il est supposé que le point laser peut être détecté dans l'image avec une précision d'un demi pixel. Il apparait alors clairement que plus les points sont éloignés de la caméra, plus la précision est mauvaise.

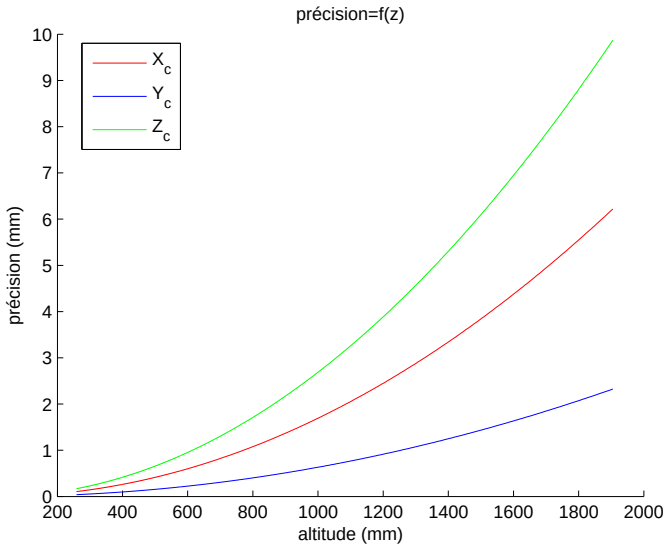


Figure 5.22 – Précision en fonction de l'altitude.

Si l'on veut que le point laser soit visible dans l'image entre les altitudes z_{min} et z_{max} (dans le repère caméra), il faut d'abord définir où se projetteront ces points dans l'image. Si les points $p_{min}(u_{min}, v_{min})$ et $p_{max}(u_{max}, v_{max})$ sont respectivement les projections des points P_{min} et P_{max} (d'altitudes z_{min} et z_{max}), alors :

$$P^c_{min} = z_{min} K^{-1} \begin{bmatrix} u_{min} \\ v_{min} \\ 1 \end{bmatrix} = \begin{bmatrix} z_{min}(u_{min} - p_u)/f_u \\ z_{min}(v_{min} - p_v)/f_v \\ z_{min} \end{bmatrix} \tag{5.95a}$$

$$P^c_{max} = z_{max} K^{-1} \begin{bmatrix} u_{max} \\ v_{max} \\ 1 \end{bmatrix} = \begin{bmatrix} z_{max}(u_{max} - p_u)/f_u \\ z_{max}(v_{max} - p_v)/f_v \\ z_{max} \end{bmatrix} \tag{5.95b}$$

Le laser doit donc être placé de telle sorte qu'il passe par les deux points P_{min} et P_{max} ; ceux-ci permettant de déterminer P_l^c et V_l^c (avec la notation de (5.81d)) :

$$P_l^c = P_{min}^c \tag{5.96a}$$

$$V_l^c = \frac{P_{max}^c - P_{min}^c}{(P_{max}^c - P_{min}^c)^T (P_{max}^c - P_{min}^c)} = \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix} \tag{5.96b}$$

Cependant, le choix du point P_l^c tel que défini ci-dessus n'est pas judicieux pour placer dans la pratique le pointeur laser. Il est alors plus commode de choisir un point appartenant au plan (X_c, Y_c) de la caméra. Définissons le point P_l^c tel que (avec la notation de (5.81c)) :

$$P_l^c = P_{min}^c + kV_l^c = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} \tag{5.97}$$

où k est un réel permettant de vérifier $z_2 = 0$; soit,

$$k = -\frac{z_{min}}{c_2} \tag{5.98}$$

d'où :

$$P_l^c = P_{min}^c - \frac{z_{min}}{c_2} V_l^c \tag{5.99}$$

En pratique, ce placement n'est pas toujours évident à effectuer. La figure 5.23 montre l'altitude calculée à partir des points appartenant à l dans le cas où l'orientation du laser n'est pas optimale. La figure montre que sur les 100 premiers pixels l'altitude est négative, c'est à dire que les rayons se croisent sur la partie négative de l'axe Z_c. Cette situation ne peut pas arriver sur le système réel car le pointeur ne projette le faisceau que dans un sens. La précision sera donc moins bonne car la longueur en pixels de l est réduite. Après calibration, il faut donc vérifier sur quelle plage de pixels le système pourra effectivement faire une mesure. Cela permet alors de réduire la zone de recherche du point laser sur l.

Enfin, en plaçant l'ensemble des trois pointeurs lasers, il faut veiller à ce que les trois points projetés au sol ne soient jamais alignés ; sinon le calcul de l'orientation n'est pas possible. Si les trois points formés par les lasers sont alignés sur le sol, alors leur projections dans le plan image sont aussi alignées. Par contraposée, si les points ne sont pas alignés dans le plan image, alors ils ne le sont pas non plus au sol. On obtient ainsi une condition suffisante sur les lignes épipolaires l_i de chaque laser i. Cette condition est donc restrictive car elle n'est pas nécessaire et suffisante. En effet, des points alignés dans le plan image peuvent ne pas être alignés au sol.

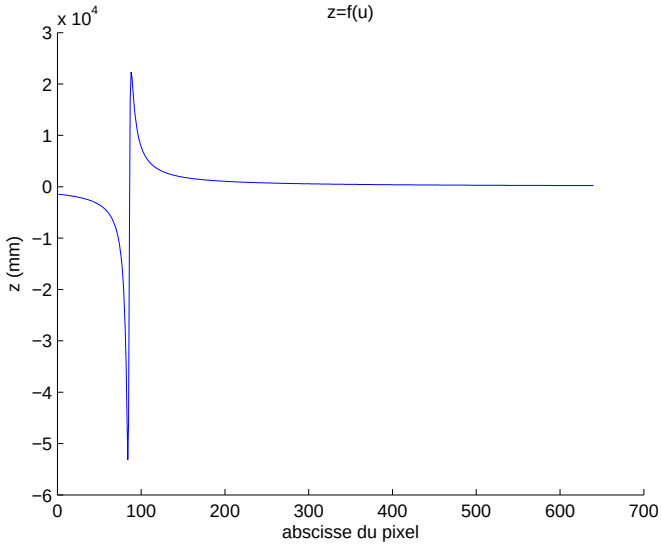


Figure 5.23 – Altitude en fonction de l'abscisse u du pixel. L'ordonnée est calculée grâce à l.

Cependant l'étude de ce cas est beaucoup plus complexe et ne sera pas abordée ici. La figure 5.24 montre une solution possible au problème. Les lignes épipolaires sont telles que les points ne puissent pas être alignés, tout en maximisant leur longueur pour ne pas perdre trop de précision.

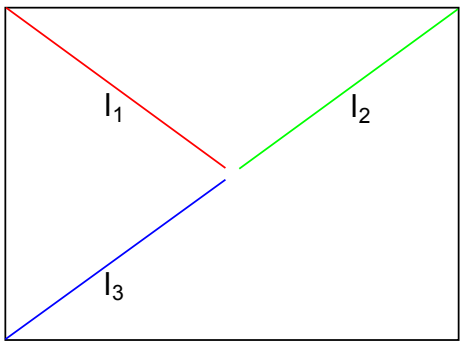


Figure 5.24 – Solution possible de placement des lignes épipolaires.

Résultats

Le système construit est montré sur la figure 5.25. Les figures 5.26 et 5.27 montrent les résultats provenant de la centrale inertielle et du système de vision. Le

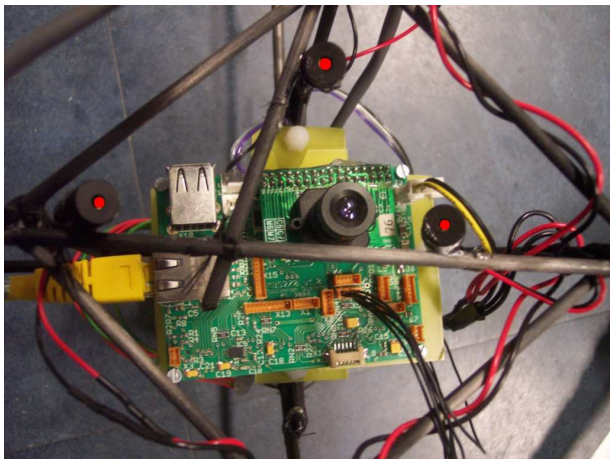


Figure 5.25 – Vue de dessous du drone montrant le système de vision avec ses trois lasers. Un point rouge a été ajouté sur ces derniers pour bien les repérer.

quadrirotor ne vole pas lors de cet essai mais est bougé à la main afin de comparer les deux mesures. Les résultats obtenus par la caméra sont bons, même s'il y a toujours une erreur entre les deux capteurs. La centrale est supposée représenter la vérité terrain, mais aucune mesure n'a pu être faite pour le vérifier. Le principal inconvénient du système par caméra est sa fréquence d'échantillonnage de 30 Hz qui reste trop faible pour stabiliser le quadrirotor en vol stationnaire. Cependant, ce système pourrait être utilisé conjointement avec des capteurs inertiels afin d'améliorer l'estimation de l'attitude. En effet, le système de vision à l'avantage de ne pas avoir de dérive contrairement aux systèmes inertiels (à cause des accélérations latérales, des vibrations ou des perturbations magnétiques par exemple). L'altitude calculée est donnée sur la figure 5.28 à titre d'information car elle n'a pas été comparée avec un autre capteur.

5.3.3 Suivi de mur grâce à un laser ligne

Le système présenté précédemment permet de reconstruire un plan à partir de trois lasers. Cependant il a le désavantage de ne pouvoir fournir que l'information de ces trois points. La suite naturelle est donc de ne pas utiliser un pointeur laser mais plutôt un laser ligne, permettant d'obtenir plus de données.

Ce style de technique est généralement utilisé pour faire la reconstruction 3D d'objets, en utilisant une lumière structurée. La figure 5.29 montre par exemple les travaux de [RCM+01]. Des bandes de couleurs sont projetées sur l'objet à reconstruire. A chaque pas, les bandes sont divisées en deux, afin d'aider à faire

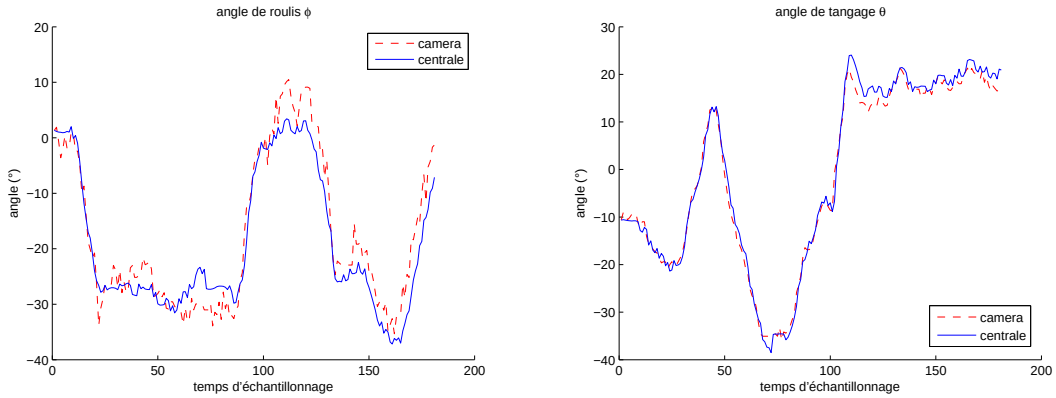


Figure 5.26 – Angles de roulis et de tangage donnés par la centrale inertielle et la caméra.

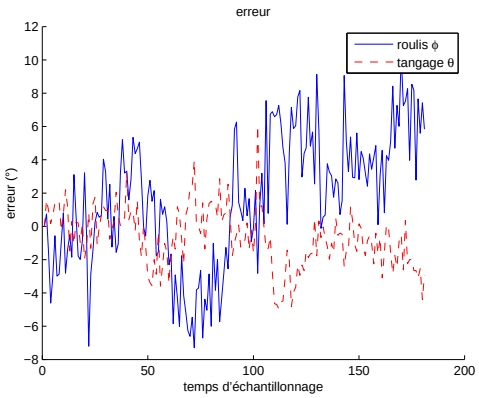


Figure 5.27 – Erreur entre la caméra et la centrale inertielle.

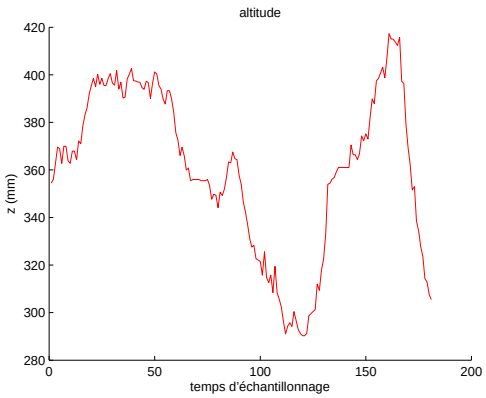


Figure 5.28 – Altitude donnée par la caméra.

l'appariement entre la bande projetée et la bande vue par la caméra. De plus, la couleur des bandes est choisie de façon à pouvoir distinguer l'ombre sur l'objet. Notons cependant que cette méthode n'est pas adaptée au temps réel, car le temps de calcul est d'autant plus long qu'il y a d'itérations. D'ailleurs le système de projection n'est pas "embarquable". L'utilisation d'un laser ligne à la place des bandes projetées représente donc une alternative simplifiée à cette technique, car elle ne permet pas un tel niveau de reconstruction 3D.

Ainsi, en robotique, [KMM+96] utilise par exemple un capteur de ce style. Deux lasers lignes sont placés sur un bras manipulateur où est aussi fixée une caméra. Ce système de vision permet alors de positionner le bras par rapport à une sphère

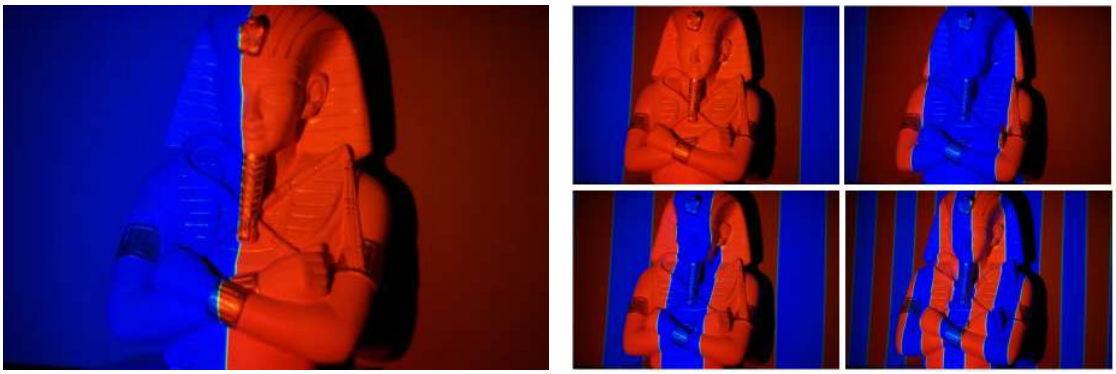


Figure 5.29 – Reconstruction 3D par lumière structurée. Source : [RCM+01].

fixe.

Dans cette partie, nous proposons alors un système de vision composé d'un laser ligne. Celui-ci permet de projeter une droite sur un mur (supposé vertical) afin de connaitre l'angle de lacet et la distance entre le drone et le mur. Le système est par ailleurs capable de détecter plusieurs pans de mur et donc de faire du suivi de mur.

Calibration du laser dans le repère de la caméra

L'objectif est de trouver un point O_p appartenant au plan laser π_l et un vecteur Y_p normal à ce plan. Une technique similaire à celle décrite pour calibrer les pointeurs lasers (voir section 5.3.2) a été utilisée. Le schéma du système laser/caméra est représenté sur la figure 5.30.

Plusieurs images de la mire (voir figure 5.31) sont donc prises puis les points appartenant au laser sont exprimés suivant l'équation (5.70) dans le repère de la caméra. Une régression linéaire en trois dimensions (*Principal Components Analysis*, voir section 5.3.2) est ensuite effectuée pour trouver un point appartenant au plan laser ainsi qu'un vecteur normal. Le plan π peut être représenté dans le repère caméra par :

$$\tilde{\pi}^c = \begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \end{bmatrix} \tag{5.100}$$

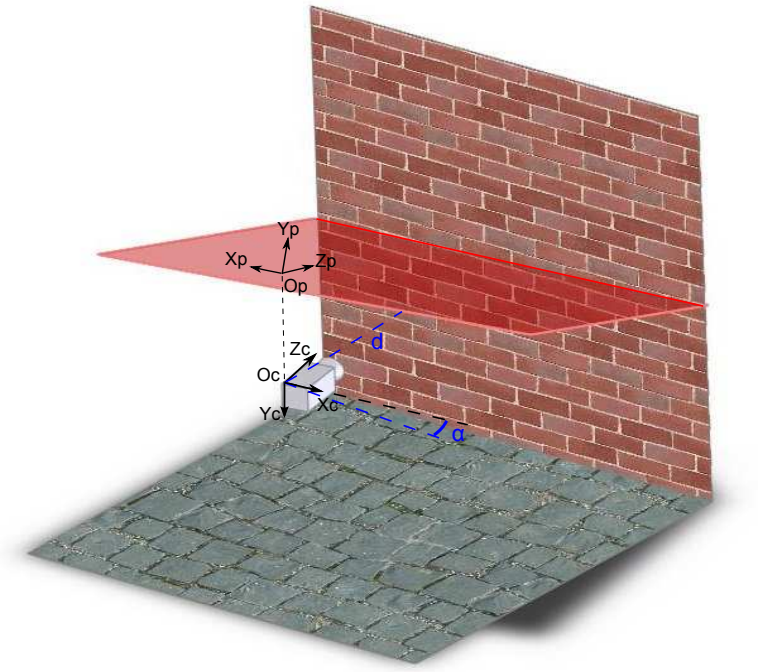


Figure 5.30 – Laser ligne et caméra.

avec,

$$(\tilde{\pi}^c)^T \tilde{O}_p^c = 0 \tag{5.101a}$$

$$Y_p^c = \begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \end{bmatrix} \tag{5.101b}$$

Afin de simplifier les calculs qui suivent, un repère tel que défini sur la figure 5.30 va être construit sur le plan laser. Soit O_p la projection de O_c sur le plan du laser suivant le vecteur Y_c. Le point O_p sera l'origine de ce nouveau repère. Soit Z_p un vecteur unitaire coplanaire à Z_c et appartenant au plan laser. Soit Y_p le vecteur normal au plan laser trouvé précédemment. Enfin X_p est le vecteur tel que X_p, Y_p et Z_p forment une base orthonormale directe.

Calcul de l'angle et de la distance

Cette partie présente comment calculer l'angle α, ainsi que la distance d entre le mur et le drone (voir figure 5.30).

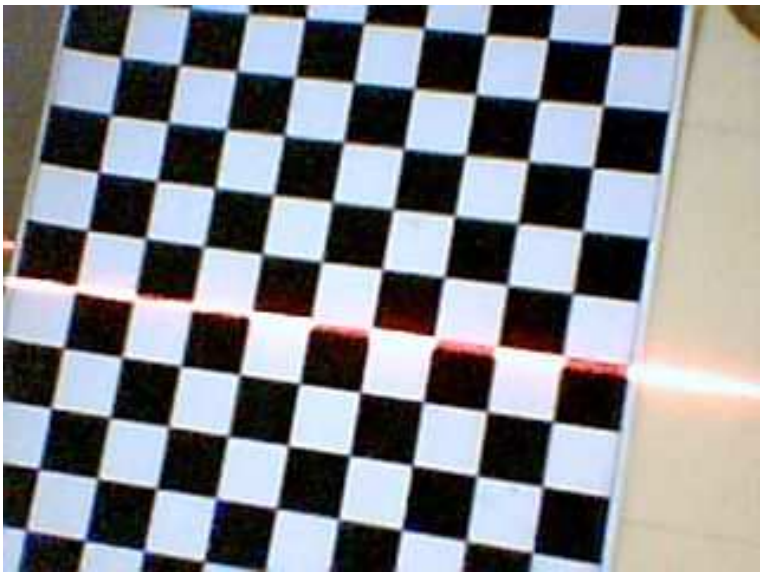


Figure 5.31 – Mire de calibration du laser.

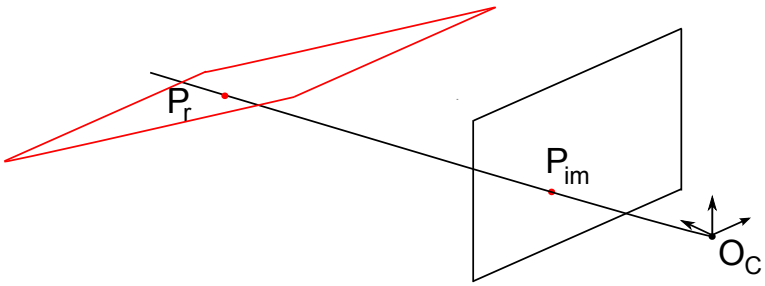


Figure 5.32 – Point appartenant au plan du laser.

Soit un point P_r appartenant au plan laser, sa projection dans le plan image de la caméra est noté P_{im}, voir figure 5.32. La relation entre les coordonnées $(x^c_{im}, y^c_{im}, z^c_{im})$ de P_{im} et ses coordonnées (u, v) en pixels est :

$$\begin{bmatrix} x^c_{im} \\ y^c_{im} \\ z^c_{im} \end{bmatrix} = K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{5.102}$$

Les coordonnées (x^c_r, y^c_r, z^c_r) du point réel vérifient deux conditions. D'une part P_r appartient à la droite passant par O_c et P_{im}, donc il existe un réel k tel que :

$$P_r^c = \begin{bmatrix} x_r^c \\ y_r^c \\ z_r^c \end{bmatrix} = k \begin{bmatrix} x_{im}^c \\ y_{im}^c \\ z_{im}^c \end{bmatrix} \tag{5.103}$$

et d'autre part, il appartient au plan π, donc :

$$(\tilde{\pi}^c)^T \tilde{P}_r^c = 0 \tag{5.104}$$

soit :

$$k = -\frac{\pi_4}{\pi_1 x_{im}^c + \pi_2 y_{im}^c + \pi_3 z_{im}^c} \tag{5.105}$$

Les points laser trouvés peuvent alors s'exprimer dans le plan laser par $P_r^p = [x_r^p, 0, z_r^p]^T$, avec :

$$x_r^p = (P_r^c - O_p^c)^T X_p^c \tag{5.106a}$$
$$z_r^p = (P_r^c - O_p^c)^T Z_p^c \tag{5.106b}$$

Afin de détecter s'il y a plusieurs pans de mur, il est nécessaire de calculer la distance de chacun des points trouvés par rapport à la droite passant par les deux points extrêmes. Si le point le plus éloigné se trouve à une distance supérieure à un seuil à fixer, alors il y a plusieurs pans de mur (voir figure 5.33). L'opération est réitérée sur les deux nouveaux pans de mur jusqu'à ce que tous les points soient à une distance inférieure au seuil fixé.

Le reste de l'étude se fait donc sur un pan de mur. Une régression linéaire en 2D de la droite laser (voir figure 5.34) dans le plan π permet de déterminer son équation :

$$z = ax + b \tag{5.107}$$

Dans le repère caméra, cette droite est caractérisée par un point P_{ligne}^c et un vecteur V_{ligne}^c :

$$P_{ligne}^c = O_p^c + bZ_P^c \tag{5.108a}$$
$$V_{ligne}^c = X_P^c + aZ_P^c \tag{5.108b}$$

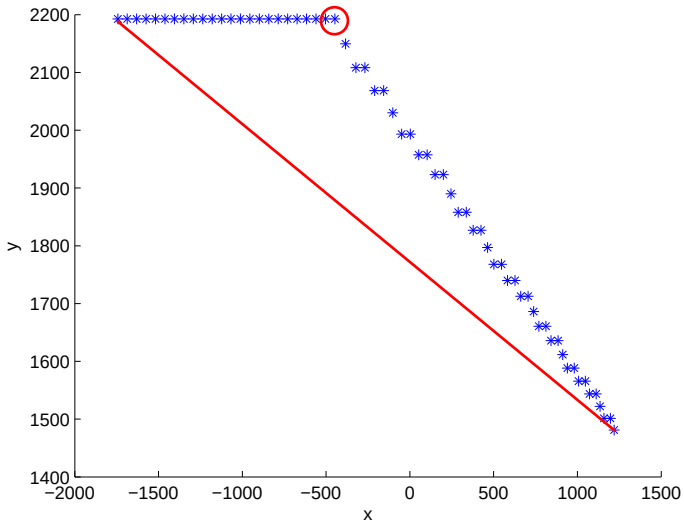


Figure 5.33 – Détection de plusieurs pans de mur.

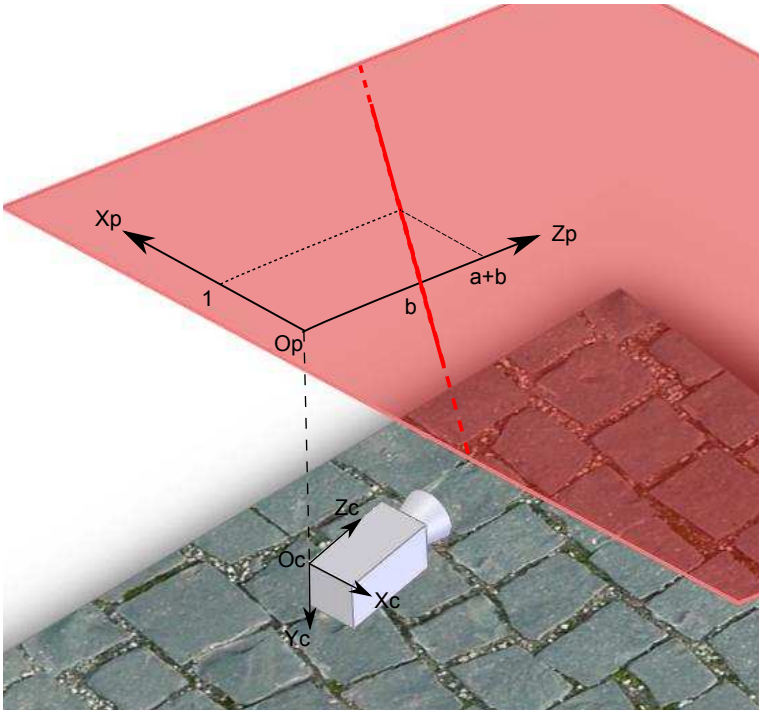


Figure 5.34 – Régression linéaire en 2D de la droite laser.

Soit, dans le repère fixe (noté avec un exposant f) :

$$P^f_{ligne} = R_{fb} R_X P^c_{ligne} = \begin{bmatrix} x_{P^f_{ligne}} \\ y_{P^f_{ligne}} \\ z_{P^f_{ligne}} \end{bmatrix} \tag{5.109a}$$

$$V^f_{ligne} = R_{fb} R_X V^c_{ligne} = \begin{bmatrix} x_{V^f_{ligne}} \\ y_{V^f_{ligne}} \\ z_{V^f_{ligne}} \end{bmatrix}$$

où R_{fb} est la matrice des cosinus directeurs définie dans la section 3.1.1 et R_X est la matrice traduisant l'orientation entre la caméra et la centrale inertielle (voir section 5.1.3).

Introduisons un plan "virtuel" du sol, celui-ci étant orthogonal au plan du mur ; son altitude n'ayant peu d'importance. Soit O le projeté orthogonal de O_c sur ce plan et soient x et y les vecteurs unitaires issus des projections de X_c et Z_c sur ce plan. Dans (O, x, y), la projection de la ligne laser a pour équation (voir figure 5.35) :

$$y = a'x + b' \tag{5.110}$$

avec,

$$a' = y_{V^f_{ligne}} / x_{V^f_{ligne}} \tag{5.111a}$$

$$b' = y_{P^f_{ligne}} - a' x_{P^f_{ligne}} \tag{5.111b}$$

Soit $M(x_M, y_M)$, le projeté orthogonal de O sur la droite ; ses coordonnées satisfont les deux relations suivantes :

$$a'x_M + b' = y_M \tag{5.112a}$$

$$\begin{bmatrix} x_M & y_M \end{bmatrix} \begin{bmatrix} 1 \\ a' \end{bmatrix} = 0 \tag{5.112b}$$

D'où :

$$x_M = -\frac{a'b'}{1 + a'^2} \tag{5.113a}$$

$$y_M = -\frac{a'^2 b'}{1 + a'^2} + b \tag{5.113b}$$

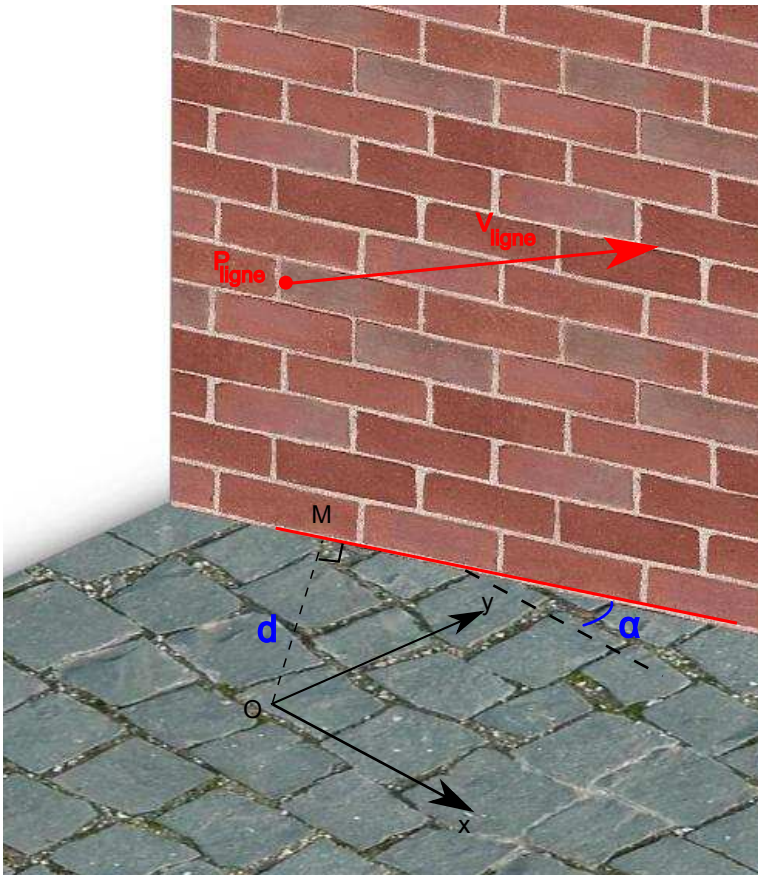


Figure 5.35 – Régression linéaire en 2D de la droite laser.

Finalement :

$$d = \sqrt{x_M^2 + y_M^2} \tag{5.114a}$$

$$\alpha = atan2(y_{V_{ligne}^f}, x_{V_{ligne}^f}) \tag{5.114b}$$

Principe de commande

Nous proposons ici un algorithme de commande pour effectuer le suivi de mur. Il est supposé que seuls deux murs au maximum sont simultanément visibles pour simplifier le problème. Les distances et angles seront respectivement notés d_1 et α_1 pour le premier mur et d_2 et α_2 pour le second. Notons de plus que les angles α_1 et α_2 sont orientés du repère du mur vers le repère du drone.

Le système présenté est seulement capable de mesurer la distance et l'angle par

rapport à un mur (auquel est attaché le repère fixe $R_{f1}(O_1, x_{f1}, y_{f1}, z_{f1})$) ; c'est à dire qu'il est possible d'asservir le drone à une distance et à un angle fixes du mur. Le déplacement latéral n'est par contre pas mesuré ce qui rend la tâche de suivi de mur difficile. Pour résoudre ce problème, nous proposons d'imposer au drone une avance diagonale en zigzag (voir figure 5.36), afin que les vitesses frontale et latérale soient (en principe) égales. Dans la réalité, ces vitesses ne sont pas égales car les dynamiques de roulis et de tangage ne sont pas strictement identiques, même si elles doivent être proches car le drone a été conçu de manière symétrique.

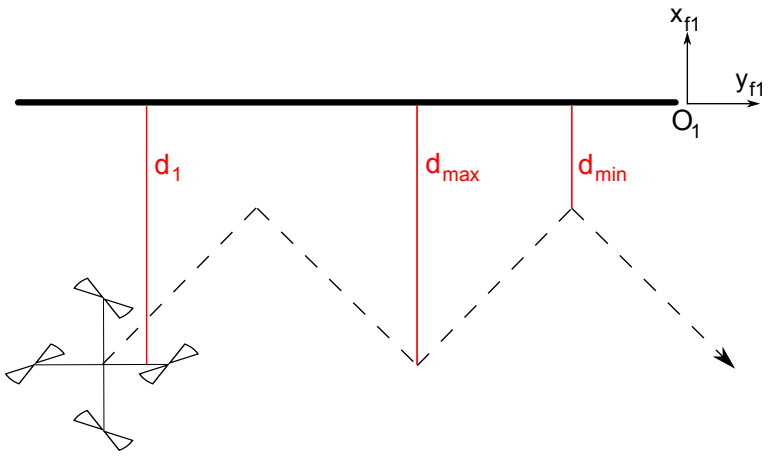


Figure 5.36 – Déplacement du quadrirotor. Un seul mur est visible.

Dans le cas où deux murs sont visibles, il est alors possible de mesurer les distances dans deux directions différentes, voir figure 5.37. Le mouvement d'avance peut alors être rectiligne. Le repère fixe $R_{f2}(O_2, x_{f2}, y_{f2}, z_{f2})$ est alors attaché au second mur.

L'algorithme de suivi de mur peut alors se résumer de la façon suivante :

1. **Un seul mur visible** (figure 5.36) : le drone asservi son angle par rapport au mur ($\psi = \alpha_1$), et se déplace en zigzag grâce à la distance ($x = d_1$) calculée. La consigne en x alterne successivement entre d_{min} et d_{max}.
2. **Deux murs visibles** (figure 5.36) : lorsqu'un deuxième mur apparait, deux angles et deux distances sont calculables. Le drone se déplace maintenant de manière rectiligne et s'oriente progressivement vers le second mur ($\psi = \alpha_2$).
   (a) le drone se rapproche des deux murs pour atteindre $d_1 = d_2 = d_{min}$ (point A),
   (b) d_2 est ensuite maintenue à d_{min} et d_1 augmente progressivement,
   (c) dès que le premier mur disparait, retour au cas 1.

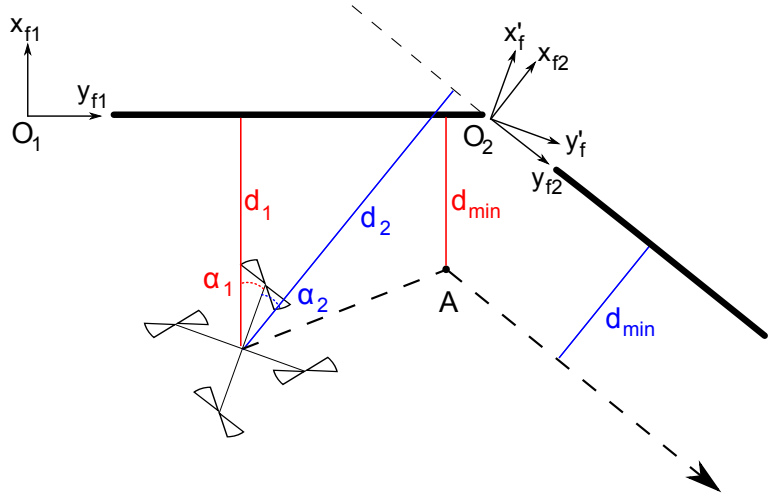


Figure 5.37 – Déplacement du quadrirotor. Deux murs sont visibles.

Lois de commande

Afin de réaliser cela, le modèle simplifié (3.26) du quadrirotor est utilisé. Les résultats de la section 4.2 servent alors à proposer des lois de commandes par fonctions de saturations.

L'altitude est ainsi stabilisée (à une consigne z_d) pendant toute la tâche de suivi de mur par la loi :

$$u_z = -m\Big[\sigma_{pz}\Big(k_{pz}(z - z_d)\Big) + \sigma_{dz}(k_{dz}\dot{z})\Big] \tag{5.115}$$

Soit b_{pz} la borne de σ_{pz} et b_{dz} celle de σ_{dz}. En utilisant (4.90), on obtient les conditions : $k_{pz},\ k_{dz} > 0$. De plus, afin de satisfaire (4.86) il s'ensuit que :

$$b_{dz} > b_{pz} \tag{5.116a}$$
$$k_{dz}^2 > k_{pz} \tag{5.116b}$$

De même, l'angle de lacet est contrôlé par la loi de commande suivante :

$$\tau_{b_z} = -I_{b_{zz}}\Big(\sigma_{p\psi}(k_{p\psi}\psi) + \sigma_{d\psi}(k_{d\psi}\dot{\psi})\Big) \tag{5.117}$$

Soit $b_{p\psi}$ la borne de $\sigma_{p\psi}$ et $b_{d\psi}$ celle de $\sigma_{d\psi}$. En utilisant (4.90), on obtient les conditions : $k_{p\psi},\ k_{d\psi} > 0$. De plus, afin de satisfaire (4.86) il s'ensuit que :

$$b_{d\psi} > b_{p\psi} \tag{5.118a}$$
$$k_{d\psi}^2 > k_{p\psi} \tag{5.118b}$$

L'altitude et l'angle de lacet étant stabilisés, les états suivants tendent vers zéro :

$$\psi,\dot{\psi} \to 0 \tag{5.119a}$$
$$z,\dot{z} \to 0 \tag{5.119b}$$

d'où,

$$u_z \to 0 \tag{5.120a}$$
$$u \to mg \tag{5.120b}$$

Le modèle du quadrirotor se réduit à (3.27), soit pour le déplacement selon l'axe x :

$$\ddot{x} = g\theta \tag{5.121a}$$
$$I_{b_{yy}}\ddot{\theta} = \tau_{y_b} \tag{5.121b}$$

La loi de commande par fonctions de saturations suivante est alors utilisée :

$$\tau_{y_b} = -\frac{I_{b_{yy}}}{g}\Big[\sigma_{px}\Big(k_{px}(x-x_d)\Big) + \sigma_{dx}(k_{dx}\dot{x}) + \sigma_{p\theta}(k_{p\theta}g\theta) + \sigma_{d\theta}(k_{d\theta}g\dot{\theta})\Big] \tag{5.122}$$

où $k_{px}, k_{dx}, k_{p\theta}, k_{d\theta} > 0$ et σ_{px}, σ_{dx}, $\sigma_{p\theta}$ et $\sigma_{d\theta}$ sont respectivement bornées par b_{px}, b_{dx}, $b_{p\theta}$ et $b_{d\theta}$. La position désirée est notée x_d. De plus, d'après (4.90) il vient,

$$\det(A) = \lambda^4 + \lambda^3 k_{d\theta} + \lambda^2 k_{p\theta} + \lambda k_{dx} + k_{px} \tag{5.123}$$

puis, le critère de Routh Hurwitz donne,

$$k_{p\theta}k_{d\theta} > k_{dx} \tag{5.124a}$$
$$k_{dx}k_{p\theta}k_{d\theta} > k_{px}k_{d\theta}^2 + k_{dx}^2 \tag{5.124b}$$

Afin de satisfaire (4.86) on obtient :

$$b_{d\theta} > b_{px} + b_{dx} + b_{p\theta} \tag{5.125a}$$
$$b_{p\theta} > b_{px} + b_{dx} \tag{5.125b}$$
$$b_{dx} > b_{px} \tag{5.125c}$$
$$\frac{b_{px}}{k_{px}} > \frac{k_{p\theta}}{k_{dx}^2}b_{px} + \frac{k_{d\theta}}{k_{dx}k_{p\theta}}(b_{px}+b_{dx}) + \frac{b_{px}+b_{dx}+b_{p\theta}}{k_{dx}k_{d\theta}} \tag{5.125d}$$
$$\frac{b_{px}}{k_{dx}} > \frac{k_{d\theta}}{k_{p\theta}^2}(b_{px}+b_{dx}) + \frac{b_{px}+b_{dx}+b_{p\theta}}{k_{p\theta}k_{d\theta}} \tag{5.125e}$$
$$\frac{b_{px}+b_{dx}}{k_{p\theta}} > \frac{b_{px}+b_{dx}+b_{p\theta}}{k_{d\theta}^2} \tag{5.125f}$$

Suivant l'axe y, le modèle est le suivant :

$$\ddot{y} = -g\phi \tag{5.126a}$$

$$I_{b_{xx}}\ddot{\phi} = \tau_{x_b} \tag{5.126b}$$

La loi suivante est utilisée :

$$\tau_{x_b} = \frac{I_{b_{xx}}}{g}\Big[\sigma_{py}\Big(k_{py}(y - y_d)\Big) + \sigma_{dy}(k_{dy}\dot{y}) - \sigma_{p\phi}(k_{p\phi}\phi) - \sigma_{d\phi}(k_{d\phi}\dot{\phi})\Big] \tag{5.127}$$

où $k_{py}, k_{dy}, k_{p\phi}, k_{d\phi} > 0$ et σ_{py}, σ_{dy}, $\sigma_{p\phi}$ et $\sigma_{d\phi}$ sont respectivement bornées par b_{py}, b_{dy}, $b_{p\phi}$ et $b_{d\phi}$. De plus k_{py}, k_{dy}, $k_{p\phi}$, $k_{d\phi}$, b_{py}, b_{dy}, $b_{p\phi}$ et $b_{d\phi}$ doivent satisfaire les mêmes conditions que k_{px}, k_{dx}, $k_{p\theta}$, $k_{d\theta}$, b_{px}, b_{dx}, $b_{p\theta}$ et $b_{d\theta}$ dans (5.124) et (5.125). La position désirée étant notée y_d.

Notons que les vitesses $\dot{x}$, $\dot{y}$ et $\dot{z}$ nécessaires dans les lois (5.122), (5.127) et (5.115) peuvent être obtenues grâce à un filtre de Kalman (voir section 3.5.3) utilisant une mesure de distance et la valeur d'un accéléromètre.

Cas d'un seul mur. Dans ce cas, l'angle de lacet est contrôlé par (5.117), avec :

$$\psi = \alpha_1 \tag{5.128}$$

De plus, le déplacement en zigzag s'effectue grâce à (5.122) et (5.127), avec :

$$x = d_1 \tag{5.129a}$$

$$y = d_1 \tag{5.129b}$$

$$\dot{y} = \dot{x} \tag{5.129c}$$

les positions de consignes x_d et y_d passant alternativement de d_{min} à d_{max}.

Cas de deux murs. Lorsqu'un deuxième mur apparait, l'angle de lacet est contrôlé par (5.117), avec :

$$\psi = \alpha_2 \tag{5.130}$$

Afin d'assurer le déplacement rectiligne tout en s'orientant sur le deuxième mur, introduisons un repère $R'(O_2, x'_f, y'_f, z'_f)$ tournant avec le drone. Ce repère est obtenu en effectuant une rotation d'axe z_f et d'angle α_2 au repère fixe formé sur le second mur. La rotation autour de l'axe z_f par la loi (5.117) peut être relativement lente (en choisissant de faibles bornes aux fonctions de saturations), et le repère R' peut donc être considéré comme inertiel. Dans ce cas, le modèle (3.27) reste valable. La première étape consiste à rejoindre le point A (voir figure 5.37), c'est à dire asservir

d_1 à d'_{min} (suivant l'axe x_{f1}) et d_2 à d_{min} (suivant l'axe x_{f2}). Le déplacement est alors commandé par (5.122) et (5.127) dans R', avec :

$$x - x_d = x_A \cos\alpha_2 + y_A \sin\alpha_2 \tag{5.131a}$$
$$y - y_d = -x_A \sin\alpha_2 + y_A \cos\alpha_2 \tag{5.131b}$$

avec (voir l'annexe C) :

$$x_A = d_2 - d_{min} \tag{5.132a}$$
$$y_A = \frac{\cos(\alpha_1 - \alpha_2)}{\sin(\alpha_1 - \alpha_2)}(d_2 - d_{min}) - \frac{d_1 - d_{min}}{\sin(\alpha_1 - \alpha_2)} \tag{5.132b}$$

Une fois le point A atteint, la distance d_1 suivant l'axe x_{f1} est augmentée progressivement et la distance d_2 suivant l'axe x_{f2} est maintenue à d_{min} ; c'est à dire :

$$x - x_d = x_A \cos\alpha_2 + y'_A \sin\alpha_2 \tag{5.133a}$$
$$y - y_d = -x_A \sin\alpha_2 + y'_A \cos\alpha_2 \tag{5.133b}$$

avec :

$$x_A = d_2 - d_{min} \tag{5.134a}$$
$$y'_A = \frac{\cos(\alpha_1 - \alpha_2)}{\sin(\alpha_1 - \alpha_2)}(d_2 - d_{min}) - \frac{d_1 - (d_{min} + kt)}{\sin(\alpha_1 - \alpha_2)} \tag{5.134b}$$

où le temps t est pris à partir du point A et k est une constante.

Notons qu'une fois que $\psi = 0$, le drone est alors orienté vers le second mur et R' et R_{f2} sont confondus. Le mouvement continue dans R_{f2} où les équations précédentes restent valables.

Simulations

Les simulations sont effectuées avec le simulateur présenté dans la section 3.10.2 ; une image de la simulation est donnée sur la figure 5.38. La caméra pointant vers le bas n'est pas utilisée pour cette simulation. Lors de cet essai, seuls deux murs au maximum peuvent être vus simultanément. Les angles et la distance de chacun des murs sont montrés sur la figure 5.39. La figure 5.40 montre la position du drone par rapport aux murs durant le vol. Cette position est donnée par le modèle dynamique du simulateur mais n'est pas utilisée par la loi de commande. Le mur n°1 est toujours le premier vu dans l'image, c'est à dire celui à gauche.

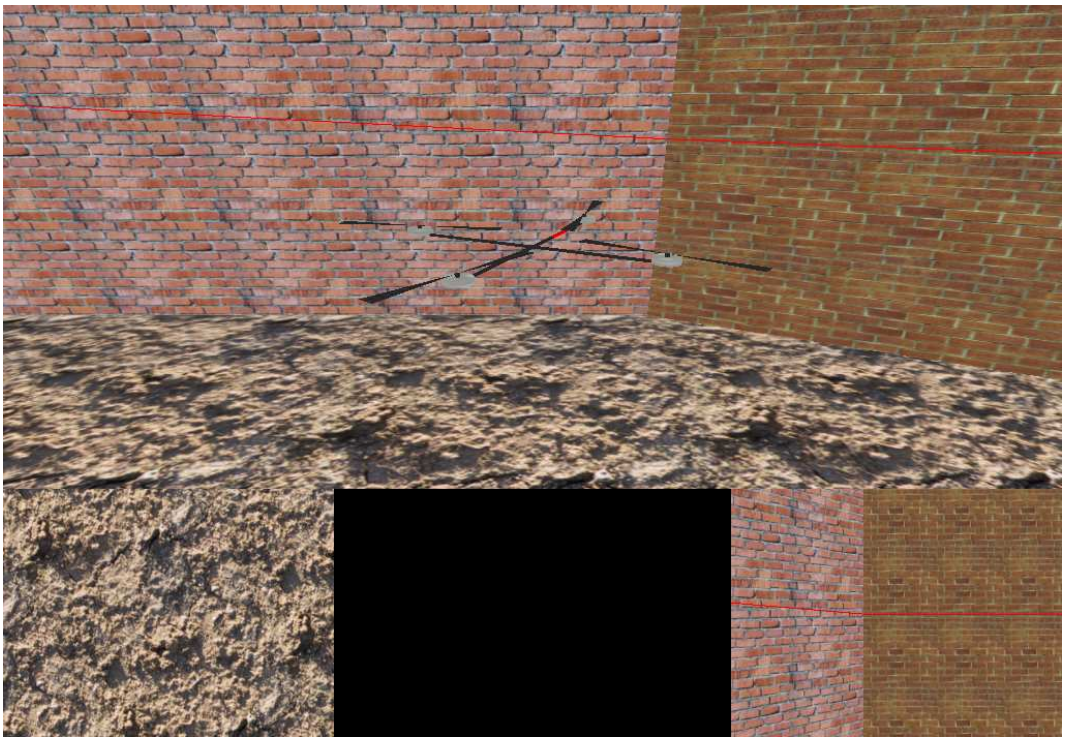


Figure 5.38 – Image du simulateur. En haut : vue globale du drone et de son environnement. En bas à gauche : caméra embarquée fixant le sol. En bas à droite : caméra embarquée regardant droit devant le drone.

Ainsi, s'il n'y a qu'un seul mur visible, la distance et l'angle du second mur sont fixés à 0 sur les graphes. Lorsque un second mur apparait sur l'image, la distance et l'angle sont alors disponibles. Puis, lorsque le premier mur disparait de l'image, le second mur devient le mur n ° 1 ; d'où certaines discontinuités sur les graphes.

La figure 5.40 montre que l'algorithme fonctionne bien et que le drone est capable de suivre le mur. Les différentes phases du suivi sont représentées par des couleurs différentes. Les traits rouges signifient qu'un seul mur est visible et les traits bleus que deux murs sont visibles. Selon l'algorithme présenté, lorsque le drone voit deux murs, il va se placer au point A puis s'éloigner du premier mur jusqu'a ce que celui-ci disparaisse du champ de vision (trajets bleus). A ce moment, un seul mur est visible et le drone reprend sa trajectoire en zig-zag (courts trajets rouges juste après les trajets bleus). Cependant en s'éloignant du second mur, le premier mur redevient visible (trajets verts). Il a été choisi dans ce cas d'ignorer le premier mur et de continuer la trajectoire en zig-zag. Sinon, le drone aurait tenté de revenir vers le point A et se serait bloqué entre les deux murs.

La figure 5.41 montre les angles et les distances sur une partie plus réduite du

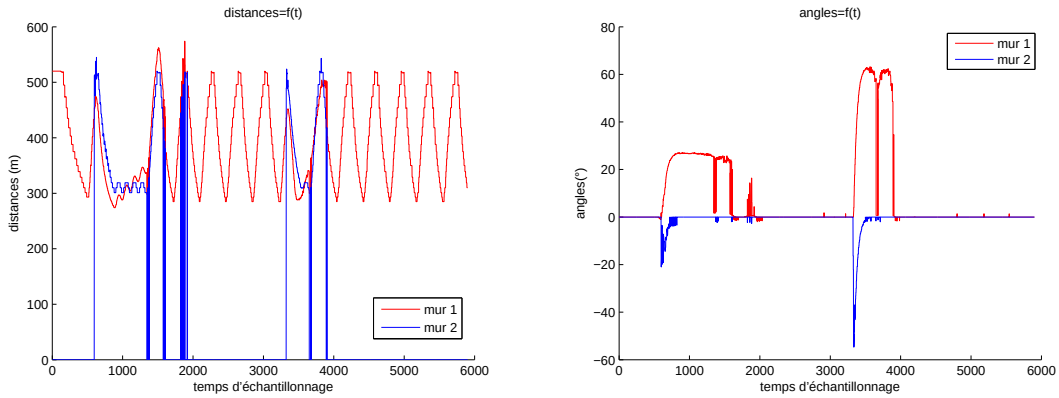


Figure 5.39 – Distances et angles.

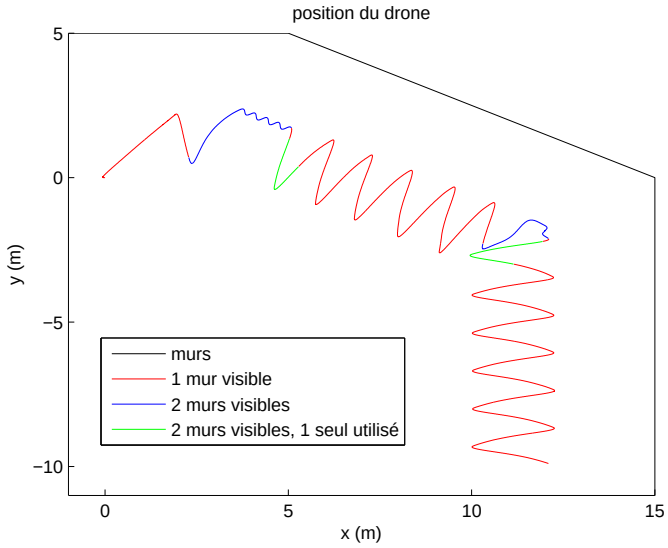


Figure 5.40 – Position du drone et des murs. Position initiale : $(0, 0)$.

trajet (correspondant à la figure 5.42). Sur le graphe de la distance par exemple, on voit clairement apparaitre et disparaitre le second mur. Ainsi, aux alentours du temps d'échantillonnage 800, le second mur est perdu (fin du trajet bleu) mais est de nouveau visible peu de temps après (trajet vert). Notons que par moment le système voit deux murs par intermittences (aux alentours du temps d'échantillonnage 1400), mais cela ne perturbe pas le drone qui est maintenant sur le second trajet rouge. En effet, dans ce cas l'algorithme sélectionne toujours le mur le plus à droite pour asservir le drone.

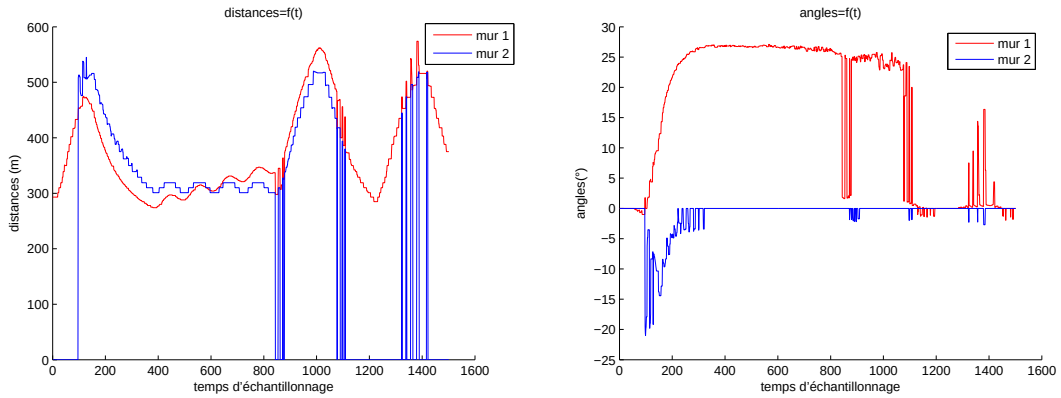


Figure 5.41 – Distances et angles.

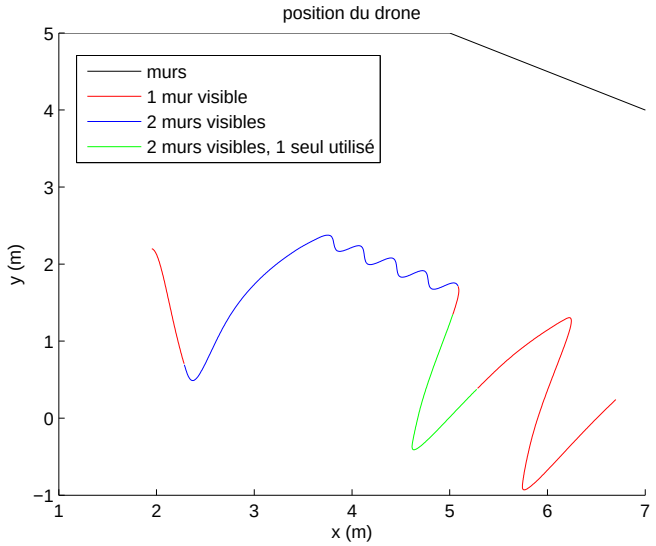


Figure 5.42 – Position du drone et des murs. Position initiale : $(0, 0)$.

Enfin, notons la forme en escalier des graphes, cela est du au fait que la résolution de l'image est assez faible (320x240) et que le drone est relativement éloigné du mur, ce qui réduit la précision.

Résultats expérimentaux

La figure 5.43 montre le drone utilisé lors de cette expérience et plus particulièrement la carte de vision embarquée (voir section 3.9.2) ainsi que le laser.

L'algorithme présenté précédemment fonctionne ainsi à 30 Hz ; cependant dans un premier temps seuls des essais avec un seul mur ont été menés.

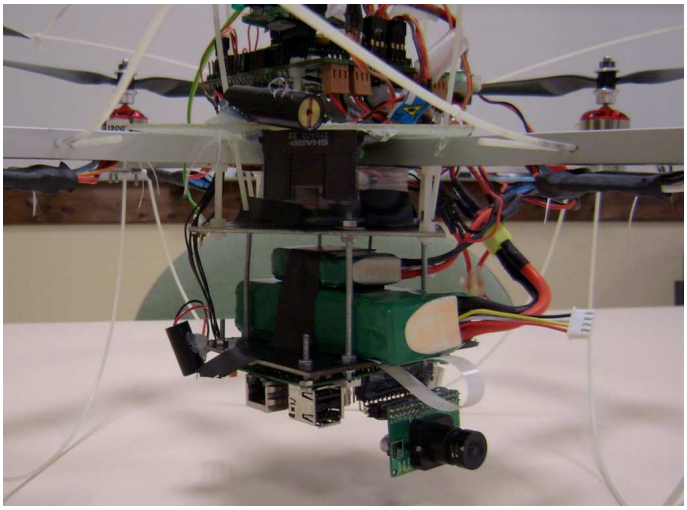


Figure 5.43 – Drone équipé du laser et de la caméra.

La figure 5.44 donne les résultats obtenus pour la mesure de l'angle de lacet. L'estimation de l'angle par le système de vision est comparée aux données issues de la centrale inertielle. Ces graphes montrent que l'estimation est bonne, les données de la caméra ayant cependant un peu plus de bruit par moments (dû au bruit dans l'image). Cependant, la mesure de la caméra à l'avantage de ne pas comporter de dérive, tel que le montre la figure 5.45 où l'erreur finale est d'environ 3°.

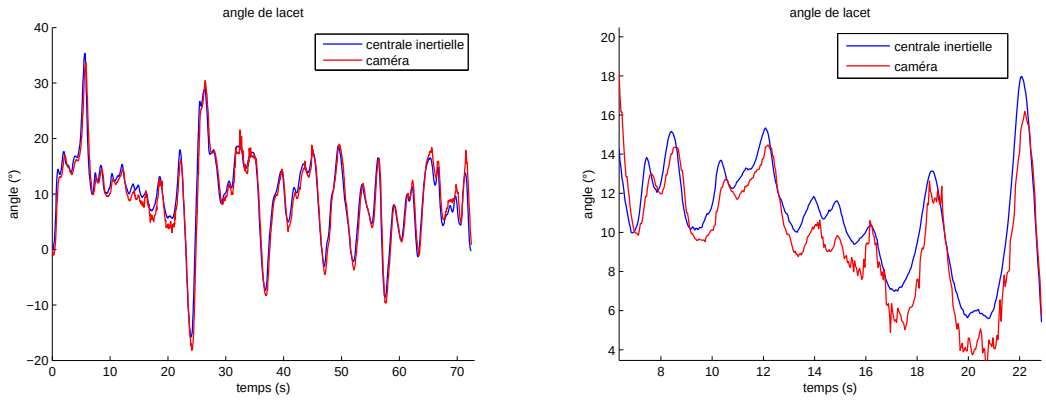


Figure 5.44 – Angle de lacet mesuré par la caméra et la centrale inertielle.

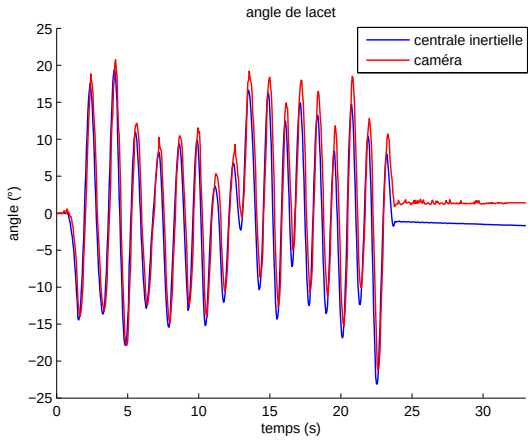


Figure 5.45 – Angle de lacet mesuré par la caméra et la centrale inertielle. Dérive de la centrale inertielle.

La figure 5.46 donne les résultats en vol pour l'estimation de la position et de la vitesse. La consigne de stabilisation est à 700 mm du mur. Ces graphes montrent que le drone oscille autour de sa position d'équilibre, le réglage des gains doit être encore ajusté pour améliorer cette réponse. Cependant, l'estimation de la vitesse n'est pas encore satisfaisante et nuit sans doutes à la bonne stabilité du drone. En effet, le bruit présent sur la mesure des accéléromètres (dû aux vibrations) empêche une bonne estimation de la vitesse de déplacement.

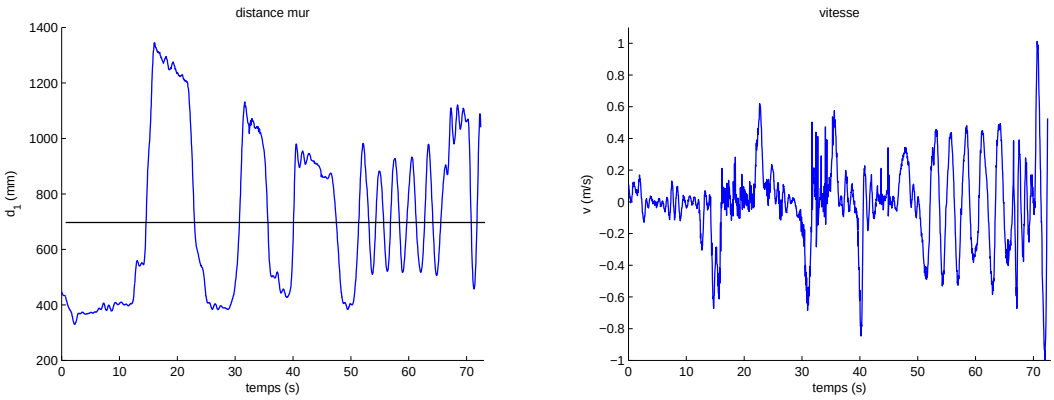


Figure 5.46 – Distance et vitesse par rapport au mur.

5.4 Conclusion

Après avoir introduit quelques généralités sur le modèle de la caméra et sa méthode de calibration, nous avons présenté deux techniques pouvant servir à l'asservissement visuel (flux optique et stéréovision). La contrainte de traitement vidéo embarqué nous a cependant obligé à étudier des versions simplifiées de ces algorithmes. Ainsi, le flux optique peut être calculé avec peu de ressources grâce à la méthode *I2A* mais seulement sur une dimension. Les résultats sur un robot mobile sont convaincants, et une méthode est suggérée pour l'adapter au drone. Cependant, l'environnement doit être contrôlé afin d'obtenir de bons résultats. La stéréovision est un outil très puissant permettant de reconstruire des scènes en 3D, mais elle nécessite le traitement de deux caméras, ce qui n'est pas possible avec notre solution embarquée. Ainsi, une solution avec des pointeurs lasers s'inspirant de la stéréovision est proposée, afin de n'utiliser qu'une seule caméra. Cette solution résout aussi le problème d'appariements de points car le laser est facile à repérer dans l'image. Une première application a été proposée avec trois pointeurs lasers classiques afin d'estimer l'attitude et l'altitude. Les résultats sont prometteurs mais la cadence de la caméra est trop lente pour pouvoir stabiliser le drone ; une solution est alors d'utiliser ce système avec des capteurs inertiels afin d'améliorer l'estimation de l'orientation. Une seconde application a alors été proposée afin d'effectuer le suivi de mur grâce à un laser ligne. Dans ce cas, le système de vision fournit des informations de distances et d'angles par rapport aux murs. La faible cadence de la caméra est alors moins pénalisante que dans le cas de la stabilisation des angles de roulis et tangage avec les trois lasers. Les simulations montrent que l'algorithme proposé est efficace et ont permis de valider le code écrit. Les premières expériences en vol donnent des résultats encourageants, mais nécessitent d'être poursuivies pour obtenir un bon comportement du drone.

Chapitre 6

Algorithme de commande pour un système retardé

6.1 Introduction

Il a été vu au chapitre précédent que la caméra est un capteur très intéressant pour les drones (et en général pour tous les systèmes robotisés), car les informations qu'elle peut fournir sont très riches et variées. Cependant, cela implique une grande quantité de données à traiter et des algorithmes pouvant être lents à calculer. Ainsi, les systèmes de vision peuvent délivrer une mesure retardée de la mesure réelle à cause du temps de traitement. En utilisant des systèmes de vision embarqué, dont la puissance de calcul est limitée, ce phénomène est encore plus important et il est à prendre en compte ; soit en utilisant des algorithmes plus légers, soit en tenant compte d'un éventuel retard lors de la conception de la loi de commande. De plus, les caméras classiques fonctionnent à environ 25 Hz, ce qui implique que le temps mis pour délivrer une image complète au système de calcul est relativement long (40 ms). Cette fréquence est donc en général beaucoup plus faible que celle utilisée par la loi de commande. Par ailleurs, le temps de traitement d'une image (s'ajoutant au temps d'acquisition) peut être variable et donc le retard induit aussi. En effet, suivant la complexité de la scène par exemple, le calcul peut être plus au moins rapide. De plus, nos systèmes de vision ont été construits autour

d'un système d'exploitation qui n'est pas temps réel, mais à temps partagé. Il a été utilisé *OpenEmbbedded* pour le système embarqué *MBS270* (voir section 3.9), et *Windows XP* pour les applications présentées dans ce chapitre ; or ces systèmes d'exploitations ne permettent pas d'avoir une période d'échantillonnage fixe pour un processus donné. Une solution à ce problème est d'utiliser des systèmes temps réel. Il en existe de nombreux, y compris dans l'embarqué, cependant rendre fonctionnel un tel système sur le *MBS270* nous a semblé assez complexe, notamment à cause de son interface de capture vidéo dédiée (*Quick Capture*) pour laquelle il aurait sans doute fallu réécrire un driver. Sur un *PC* classique (pour les calculs déportés), l'utilisation d'un système temps réel n'aurait pas posé trop de problèmes mais étant donné que le but final est d'avoir un système embarqué, nous n'avons pas non plus tenté cette voie. Néanmoins, il existe des solutions pour prendre en compte un retard variable. Ceci nous a paru plus intéressant que d'essayer de faire fonctionner un système temps réel (qui d'ailleurs aurait seulement rendu le retard fixe). Cette méthode est présentée dans ce chapitre, ainsi qu'une application sur un chariot mobile que nous avons proposée dans [SGCA08].

6.2 Problème du retard

Indépendamment des systèmes de vision, le problème du retard a largement été étudié par la communauté d'automatique. Il est en effet commun de trouver des systèmes possédant un retard interne au processus à contrôler ; c'est le cas par exemple des procédés chimiques ou biologiques, des procédés possédant des échanges thermiques, etc. Le retard peut aussi venir du contrôleur en lui même : temps de calculs de la loi de commande, système contrôlé à distance... Enfin les différents capteurs et actionneurs induisent eux aussi un certain délai ; tout comme les liens réseaux pouvant les relier avec le calculateur. En général, les performances du système contrôlé sont très sensibles au retard, plus que tout autre paramètre du modèle. Un système bouclé peut en effet facilement devenir instable à cause du retard. Par ailleurs, les systèmes à retard sont de dimension infinie, leur fonction de transfert ayant un nombre infini de pôles. Ajuster ces pôles grâce aux paramètres d'un contrôleur classique peut alors s'avérer difficile.

La principale méthode de contrôle d'un système linéaire à retard est d'utiliser un predicteur de Smith (voir [Smi59]), ou une méthode *FSA* (*Finite Spectrum Assignment*, voir [MO79]). Les premiers résultats ont été reportés pour des systèmes continus, puis appliqués à des systèmes régulièrement échantillonnés. Cependant le désavantage de ces solutions, lié à l'instabilité interne de la prédiction, est qu'elles ne peuvent stabiliser un système instable.

Afin de régler le problème du manque de données, un prédicteur basé sur le modèle du système est en général proposé pour calculer les mesures manquantes ou pour estimer l'état, en utilisant un filtre de Kalman. Les travaux de [SA05] montrent ainsi de bons résultats, cependant cette approche n'est pas souple et la loi de commande ne peut pas être remplacée par une autre (non linéaire par exemple) à cause de la structure *multi-rate*. Au contraire, les travaux de [SPA07], n'imposent pas de restriction sur la loi de commande et la méthode fonctionne sur des systèmes stables ou non. Par contre cette technique est assez lourde en calculs.

Plus récemment, une nouvelle méthode pour contrôler un système instable avec un retard variable a été présentée par [GCLA06] et [GCLA07]. L'algorithme proposé est basé sur un retour d'état en temps discret utilisant la prédiction de l'état. La convergence est montrée, même avec certaines incertitudes sur les paramètres du système, sur le retard ou même sur des variations de la période d'échantillonnage.

6.3 Contrôle basé sur un prédicteur-observateur

Un algorithme de contrôle basé sur un prédicteur et un observateur proposé par [LCGD04] et [GCLA06] pour un système linéaire à retard est présenté ici. Pour une analyse plus profonde, se référer à ces travaux où le problème est traité de façon plus générale.

6.3.1 Prédicteur

Considérons la représentation d'état suivante d'un système continu avec une entrée retardée :

$$\dot{X}(t) = A_c X(t) + B_c u(t - \tau) \tag{6.1}$$

où les matrices $A_c \in \Re^{n \times n}$ et $B_c \in \Re^{n \times m}$ sont les paramètres du système, et $\tau > 0$ est le retard.

Étant donné que l'algorithme de commande va être implémenté sur un ordinateur, définissons la période d'échantillonnage telle que $T = t_{k+1} - t_k$. De plus, supposons que le retard est proportionnel à T, soit $\tau = dT$ où $d \in \mathcal{Z}^+$. En réécrivant (6.1) en temps discret, il s'ensuit :

$$X_{k+1} = AX_k + Bu_{k-d} \tag{6.2}$$

avec :

$$A = e^{A_c T} \tag{6.3a}$$

$$B = \int_0^T e^{A_c \lambda} B_c d\lambda \tag{6.3b}$$

Réécrivons τ de la façon suivante :

$$\tau = dT = (h + \Delta h)T \tag{6.4}$$

où $\Delta h \in \mathcal{Z}$ et $h \in \mathcal{Z}^+$. Le terme dT représente alors le véritable retard (inconnu), hT est le retard utilisé pour la prédiction et ΔhT est l'erreur entre les deux.

Définissons l'entrée de commande suivante :

$$u_k = -K\hat{X}_{k+h} \tag{6.5}$$

où $K \in \Re^{m \times n}$ et :

$$\hat{X}_{k+h} = A^h X_k + A^{h-1} B u_{k-h} + \cdots + B u_{k-1} \tag{6.6}$$

Il alors été montré par [GCLA06] que la prédiction $\hat{X}_{k+h}$ précédente peut s'exprimer :

$$\hat{X}_{k+h} = X_{k+d} + A^h X_k - A^h X_{k+d-h} \tag{6.7}$$

En introduisant (6.7) dans (6.5), il s'ensuit :

$$u_k = -K(X_{k+d} + A^h X_k - A^h X_{k+d-h}) \tag{6.8}$$

ou,

$$u_{k-d} = -K(X_k + A^h X_{k-d} - A^h X_{k-h}) \tag{6.9}$$

Puis, en introduisant (6.9) dans (6.2) :

$$X_{k+1} = AX_k - BKX_k - BKA^h X_{k-d} + BKA^h X_{k-h} \tag{6.10}$$

En notant $M = (A - BK)$ et $A_1 = BKA^h$, l'équation (6.10) devient :

$$X_{k+1} = MX_k + A_1 X_{k-h} - A_1 X_{k-d} \tag{6.11}$$

Il a alors été montré par [GCLA06] que le système (6.11) est asymptotiquement stable s'il existe les matrices définies positives : P, Q_1, Q_2, R_1 et R_2 ; ainsi que les

matrices S_1, S_2, T_1 et T_2, telles que la contrainte *LMI* (*Linear Matrix Inequality* suivante soit vraie :

$$\begin{pmatrix} (1,1) & -T_1 & -T_2 & M^TP & h\Gamma^TR_1 & d\Gamma^TR_2 \\ -T_1^T & -Q_1 & 0 & A_1^TP & hA_1^TR_1 & dA_1^TR_2 \\ -T_2^T & 0 & -Q_2 & -A_1^TP & -hA_1^TR_1 & -dA_1^TR_2 \\ PM & PA_1 & -PA_1 & -P & 0 & 0 \\ hR_1^T\Gamma & hR_1^TA_1 & -hR_1^TA_1 & 0 & -hR_1 & 0 \\ dR_2^T\Gamma & dR_2^TA_1 & -dR_2^TA_1 & 0 & 0 & -dR_2 \end{pmatrix} < 0 \qquad (6.12a)$$

$$\begin{pmatrix} S_1 & T_1 \\ T_1^T & R_1 \end{pmatrix} \geq 0 \qquad (6.12b)$$

$$\begin{pmatrix} S_2 & T_2 \\ T_2^T & R_2 \end{pmatrix} \geq 0 \qquad (6.12c)$$

avec

$$\Gamma = (M - I) \qquad (6.13a)$$

$$(1,1) = -P + hS_1 + dS_2 + T_1 + T_1^T + T_2 + T_2^T + Q_1 + Q_2 \qquad (6.13b)$$

Notons que si $d = h$, alors d'après (6.7), $\hat{X}_k = X_k$. Puis, (6.10) donne :

$$X_{k+1} = (A - BK)X_k \qquad (6.14)$$

6.3.2 Observateur *multi-rate*

En général, l'état d'un système n'est pas complètement accessible et il faut employer un observateur pour l'estimer. La stabilité du système en utilisant un observateur d'état est montrée dans [GCLA07], grâce au principe de séparation entre l'observateur et le prédicteur.

Considérons alors la disponibilité de la sortie. Supposons que son équation est :

$$Y_k = CX_k \qquad (6.15)$$

mais que la mesure est seulement accessible à chaque instant N (NT représente ainsi le temps de traitement de l'image). Afin d'implémenter (6.5), l'état devrait être disponible à chaque temps d'échantillonnage. L'observateur du système (6.2) est donc construit selon l'idée de [APS07] :

$$\hat{X}_{k+1} = A\hat{X}_k + Bu_{k-d} + Lm_k(Y_k - \hat{Y}_k) \tag{6.16a}$$
$$\hat{Y}_k = C\hat{X}_k \tag{6.16b}$$

où L est la matrice d'observation. Le terme m_k vaut 1 si la mesure est disponible, 0 sinon. Ainsi, la mise à jour de l'estimation est seulement activée lorsqu'une nouvelle mesure est accessible. C'est pour cela que l'observateur est appelé *multi-rate*, il fournit un état estimé à une période différente de celle de la mesure.

Soit $\tilde{X}_k$ l'erreur d'estimation de l'état :

$$\tilde{X}_k = X_k - \hat{X}_k \tag{6.17}$$

Le terme d'innovation, $Lm_k(Y_k - \hat{Y}_k)$, doit assurer la convergence de $\tilde{X}_k$:

$$\lim_{k\to\infty} \tilde{X}_k = 0 \tag{6.18}$$

En considérant l'évolution de l'estimation de l'état sur N périodes d'échantillonnage, il s'ensuit :

$$\tilde{X}_{k+N} = \mathcal{A}\tilde{X}_k = \left(I - LCA^{-d}\right) A^N \tilde{X}_k \tag{6.19}$$

Ainsi, selon [APS07], la condition nécessaire et suffisante pour stabiliser le prédicteur est de prendre une matrice L telle que les valeurs propres de $\mathcal{A}$ soient à l'intérieur du cercle unité.

6.4 Application à un chariot mobile

Avant d'implémenter le schéma proposé précédemment sur le quadrirotor, il est préférable de d'abord le tester sur une plateforme moins complexe et moins dangereuse (en cas d'instabilité). Il a donc été décidé d'utiliser un chariot mobile à un degré de liberté et de l'asservir en position par rapport à un mur. La mesure de la position s'effectue grâce au système d'une caméra et d'un pointeur laser présenté précédemment (voir section 5.3.2). Un prédicteur et un observateur sont ajoutés dans la boucle de commande afin d'aider à la stabilisation en cas de retard et de période d'échantillonnage variable, dûs notamment au temps de calcul vidéo. La plateforme expérimentale est une voiture téléguidée telle qu'on en trouve dans le commerce, dont la direction a été bloquée pour la forcer à n'avoir qu'un seul degré de liberté.

6.4.1 Formulation du problème

Modèle dynamique

Cette partie présente la dynamique longitudinale du véhicule. Celui-ci est supposé avoir un degré de liberté suivant l'axe x comme montré sur la figure 6.1. La voiture est donc schématisée en deux dimensions, avec deux roues identiques et un châssis (voir figures 6.1 et 6.2).

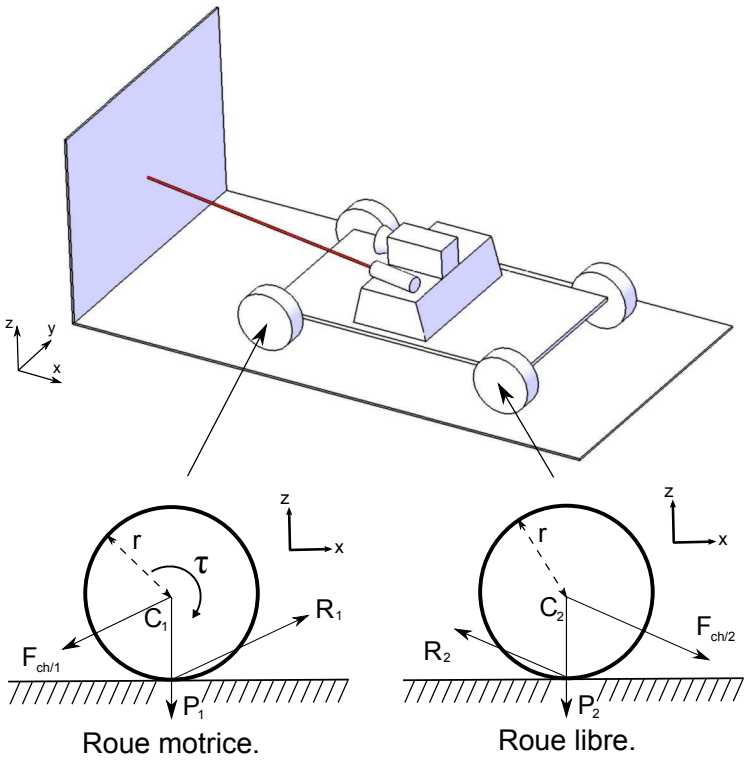


Figure 6.1 – Schéma du chariot.

Par la suite, il est noté : m la masse de chacune des roues, C_i le centre de gravité de la roue i (qui est supposé être aussi son centre géométrique), r le rayon de la roue et J son moment d'inertie. Pour le châssis il est noté : M sa masse et L la distance entre les deux roues. Le centre de gravité du châssis est supposé être en son milieu. Le contact roue/sol est supposé ponctuel et l'étude est faite à la limite du glissement pour chacune des roues. Les liaisons dans les essieux sont supposées sans frottement.

Chaque roue i, $\forall i \in [1, 2]$, est soumises aux forces suivantes :

- son poids $P_i = mg$,

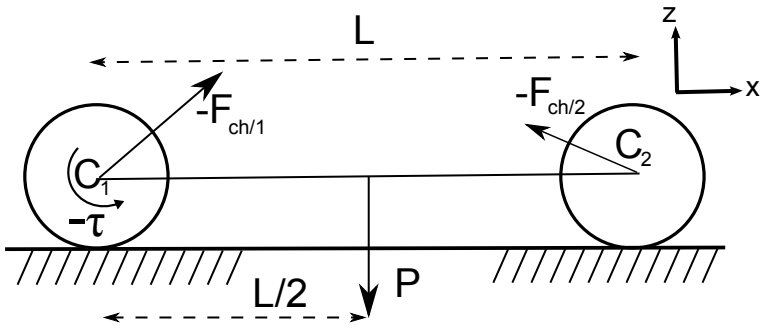


Figure 6.2 – Châssis du chariot.

- la force de réaction, R_i, du sol,
- la force, $F_{ch/i}$, exercée par le châssis et appliquée sur l'axe de la roue.

Sous l'hypothèse de la limite de glissement, il s'ensuit : $\overrightarrow{R_i} = F_{f,i}.\overrightarrow{x} + N_i.\overrightarrow{z}$ et $F_{f,i} = \mu N_i$, avec μ le coefficient de frottement entre la roue et le sol. Le couple moteur appliqué sur l'axe de la roue 1 est noté τ. De plus, le châssis est soumis à son poids, $P = Mg$, appliqué en son centre. D'après la figure 6.1 et en utilisant les équations de Newton, on obtient pour la roue motrice :

$$\mu N_1 - \overrightarrow{F_{ch/1}}.\overrightarrow{x} = m\ddot{x} \tag{6.20a}$$

$$-mg - \overrightarrow{F_{ch/1}}.\overrightarrow{z} + N_1 = 0 \tag{6.20b}$$

$$\tau - r\mu N_1 = \frac{J}{r}\ddot{x} \tag{6.20c}$$

La dynamique de la roue libre est décrite par :

$$-\mu N_2 + \overrightarrow{F_{ch/2}}.\overrightarrow{x} = m\ddot{x} \tag{6.21a}$$

$$-mg - \overrightarrow{F_{ch/2}}.\overrightarrow{z} + N_2 = 0 \tag{6.21b}$$

$$r\mu N_2 = \frac{J}{r}\ddot{x} \tag{6.21c}$$

d'après la figure 6.2, il s'ensuit pour le châssis :

$$\overrightarrow{F_{ch/1}}.\overrightarrow{x} - \overrightarrow{F_{ch/2}}.\overrightarrow{x} = M\ddot{x} \tag{6.22a}$$

$$-Mg + \overrightarrow{F_{ch/1}}.\overrightarrow{z} + \overrightarrow{F_{ch/2}}.\overrightarrow{z} = 0 \tag{6.22b}$$

$$-\tau - L\overrightarrow{F_{ch/2}}.\overrightarrow{z} + Mg\frac{L}{2} = 0 \tag{6.22c}$$

Après quelques manipulations algébriques (voir l'annexe D), on obtient :

$$\ddot{x} = k_\tau \tau \tag{6.23}$$

avec

$$k_\tau = \frac{1}{2mr + Mr + 2\frac{J}{r}} \tag{6.24}$$

L'équation (6.24) représente la dynamique d'un chariot se déplaçant suivant son axe x. Les termes de frottements ont été négligés, ils seront considérés comme une perturbation dans le reste de l'étude.

Loi de commande

Étant donné que l'objectif final est d'implémenter le schéma prédicteur/observateur dans un drone, nous utilisons ici une loi de commande non linéaire similaire à celle du quadrirotor. A partir des résultats de la section 4.2, la loi est :

$$\tau = -\frac{1}{k_\tau}\Big(\sigma_p(k_p x) + \sigma_d(k_d \dot{x})\Big) \tag{6.25}$$

où σ_p et σ_d sont des fonctions de saturations (voir (4.18)) bornées respectivement par s_p et s_d. De plus les conditions suivantes doivent être respectées :

$$s_d > s_p \tag{6.26a}$$
$$k_d^2 > k_p > 0 \tag{6.26b}$$

La stabilité de la loi a été prouvée en temps continu, il sera supposé que la loi est aussi stable en temps discret :

$$\tau_k = -\frac{1}{k_\tau}\Big(\sigma_p(k_p x_k) + \sigma_d(k_d \dot{x}_k)\Big) \tag{6.27}$$

Par ailleurs, afin d'obtenir le système (6.24) en temps discret, posons :

$$X = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \tag{6.28}$$

alors le système (6.24) s'écrit :

$$\dot{X} = A_c X + B_c \tau \tag{6.29}$$

avec :

$$A_c = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{6.30a}$$

$$B_c = \begin{bmatrix} 0 \\ k_\tau \end{bmatrix} \tag{6.30b}$$

En temps discret, on obtient alors :

$$X_{k+1} = AX_k + B\tau_k \tag{6.31}$$

avec :

$$A = e^{A_c T} = I + A_c T = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \tag{6.32}$$

et :

$$\begin{aligned} B &= \int_0^T e^{A_c \lambda} B_c d\lambda = \int_0^T \begin{bmatrix} 1 & \lambda \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ k_\tau \end{bmatrix} d\lambda \\ &= k_\tau \int_0^T \begin{bmatrix} \lambda \\ 1 \end{bmatrix} = k_\tau \begin{bmatrix} \frac{T^2}{2} \\ T \end{bmatrix} \end{aligned} \tag{6.33}$$

où T est la période d'échantillonnage.

6.4.2 Simulations

Le schéma de contrôle présenté précédemment a été implémenté sous *Simulink*. Pour cela, la mesure (supposée venir de la caméra) a été retardée d'un temps variable $d_c \in [d_{cmin}, d_{cmax}]$ et rendue disponible à des intervalles de temps irréguliers de la période de contrôle. Ainsi, lorsque la mesure n'est pas disponible, l'observateur (6.16) fonctionne en boucle ouverte ($m_k = 0$). De plus, un retard dans l'actionneur d_a a été introduit. Le retard h donné au prédicteur (6.6) a alors été choisi comme le retard maximum possible $h = d_a + d_{cmax}$. Enfin, le paramètre k_τ n'étant pas censé être connu avec précision, l'observateur et le prédicteur utilisent une constante k'_τ pouvant être différente. Les paramètres pour chacune des simulations sont donnés dans la table 6.1. Quatre essais ont ainsi été réalisés, afin de comparer la réponse du système avec ou sans prédicteur. L'observateur *multi-rate* quant à lui est utilisé dans tous les cas.

La figure 6.3 montre les résultats de l'essai n°1. Le système sans observateur oscille un peu à cause du retard de la caméra. L'ajout du prédicteur évite ces oscillations.

Catégorie	Paramètre	Essai 1	Essai 2	Essai 3	Essai 4
Loi de commande	k_p	2	2	2	2
	k_d	3	3	3	3
	s_p	0.5	0.5	0.5	0.5
	s_d	0.6	0.6	0.6	0.6
Retard caméra	d_{cmin}	0.2	0.4	0.4	0.4
	d_{cmax}	0.35	0.8	0.8	0.8
Retard actionneur	d_a	0	0	0.8	0
Prédicteur	h	0.35	0.8	1.6	0.8
Observateur	L	$[0.01\ 0.06]^T$	$[0.01\ 0.06]^T$	$[0.01\ 0.06]^T$	$[0.01\ 0.06]^T$
Modèle	T	0.05	0.05	0.05	0.05
	k_τ	1	1	1	1
	k'_τ	1	1	1	0.5

Table 6.1 – Paramètres utilisés pour les différentes simulations.

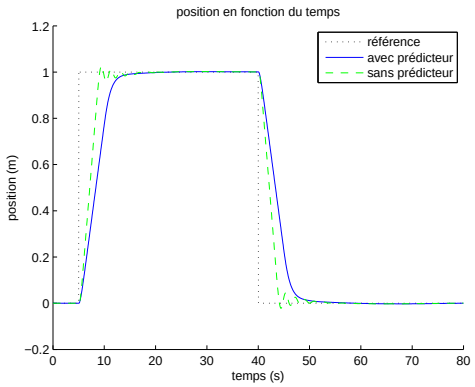


Figure 6.3 – Expérience n°1.

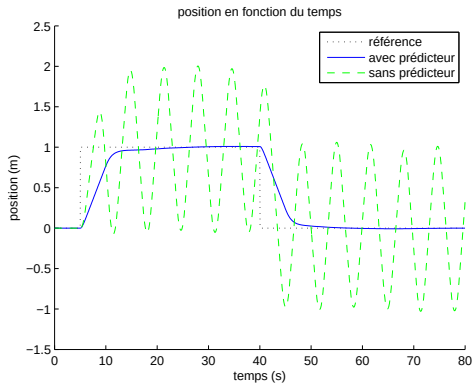


Figure 6.4 – Expérience n°2.

La figure 6.4 montre les résultats de l'essai n°2. Ici, le retard de la mesure de la caméra est plus important. Sans prédicteur, le système ne peut plus être stabilisé.

La figure 6.5 montre les résultats de l'essai n°3. Un retard supplémentaire dans l'actionneur a été ajouté. Le système sans observateur reste instable, alors que l'observateur parvient toujours à stabiliser la réponse.

La figure 6.6 montre les résultats de l'essai n°4. Cet essai est identique au

n°2, sauf que $k'_\tau \neq k_\tau$. Dans ce cas, le système avec prédicteur oscille avant de se stabiliser ; mais la réponse reste toujours meilleure que sans prédicteur.

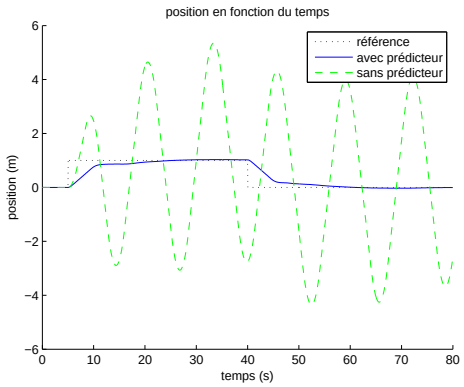


Figure 6.5 – Expérience n°3.

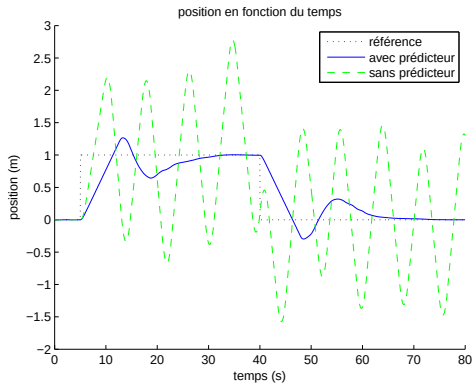


Figure 6.6 – Expérience n°4.

La figure 6.7 montre la mesure de la distance fournie à l'observateur *mutli-rate* (sur les données de l'essai n°4). Cette mesure est retardée par rapport à la position calculée par le modèle ; de plus la période d'échantillonnage est rendue variable pour simuler le vrai système de vision. Lorsque aucune information n'est disponible (*i.e.* l'image est en cours de traitement), la mesure est virtuellement mise à 0 afin que l'algorithme fasse fonctionner l'observateur en boucle ouverte.

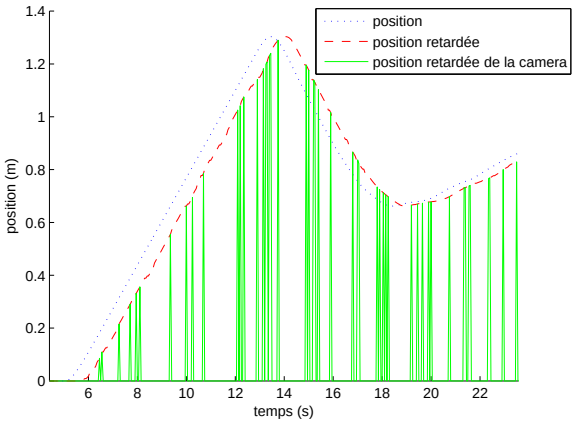


Figure 6.7 – Mesure de la caméra.

Ces simulations valident donc le bon fonctionnement du prédicteur et montrent son intérêt lorsque le retard devient important. De plus, le système est robuste aux erreurs de modélisations.

6.4.3 Expériences

La plateforme expérimentale est montrée sur la figure 6.8 ; elle est composée d'une voiture électrique se déplaçant sur un seul axe, d'une *WebCam* et d'un pointeur laser. La caméra est reliée à un ordinateur sous *Windows XP* réalisant la mesure de la distance, selon la méthode présentée à la section 5.3.2. La loi de commande est implémentée sous *xPC Target* sur un autre ordinateur, celui-ci recevant l'information de la distance par port série et envoyant des consignes de vitesses au moteur par *PWM*. Notons que les deux ordinateurs sont désynchronisés. En effet *xPC Target* va lire périodiquement la valeur de la distance dans le *buffer* de réception du port série. Or l'ordinateur de vision envoie cette donnée à chaque fois qu'un calcul est fini. La mesure ne sera donc pas disponible à chaque temps d'échantillonnage (fixé à 0.02 s), le scénario est bien celui décrit précédemment. Étant donné que l'algorithme de vision (destiné à de l'embarqué) est traité sur un ordinateur, le retard induit par le calcul (en plus du temps d'acquisition) est faible. Un retard supplémentaire sera donc volontairement ajouté sur la mesure, afin de simuler un algorithme plus long à traiter (ou de simuler un matériel plus lent à effectuer le traitement d'image).

Figure 6.8 – Plateforme expérimentale.

La première expérience a été faite sans ajouter de retard supplémentaire dans la mesure de la distance, cependant il existe le retard "naturel" du traitement d'image. La figure 6.9 montre que le système reste stable sans utiliser de prédicteur.

La seconde expérience (voir figure 6.10) est faite en ajoutant un retard équivalent à $10T$ dans la mesure. Le système est toujours stable sans avoir recours au prédicteur.

Le retard est ensuite augmenté à $15T$ lors de la troisième expérience. La figure 6.11 montre que dans ce cas, le système devient oscillant. La figure 6.12 montre ensuite le résultat d'un essai dans les mêmes conditions mais en utilisant le

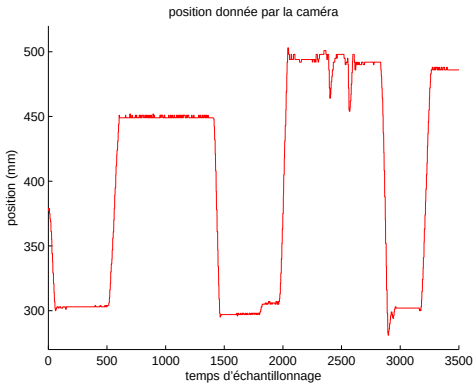


Figure 6.9 – Expérience sans prédicteur, sans retard.

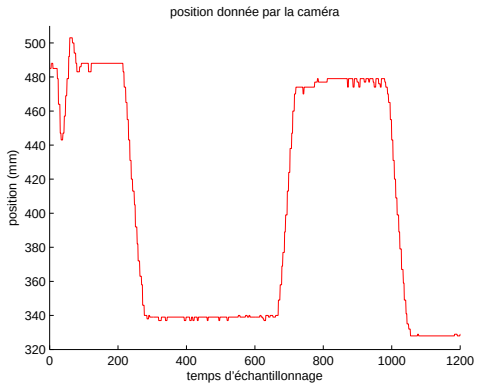


Figure 6.10 – Expérience sans prédicteur, avec retard de $10T$.

prédicteur. La réponse est alors meilleure et les oscillations moins nombreuses.

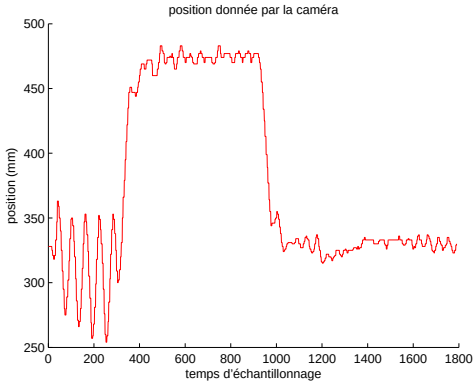


Figure 6.11 – Expérience sans prédicteur, avec retard de $15T$.

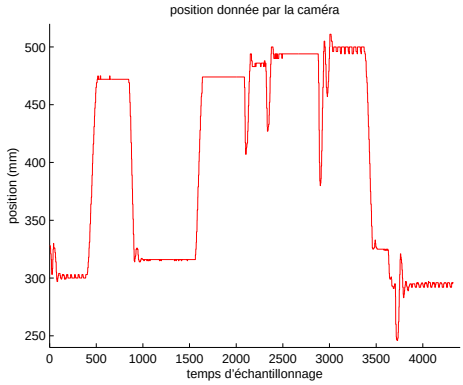


Figure 6.12 – Expérience avec prédicteur, avec retard de $15T$.

6.5 Conclusion

L'efficacité du schéma proposé a été montrée en simulations et grâce à des expériences sur un chariot mobile. Le prédicteur permet de prendre en compte les effets du retard et de garder le système stable. De plus, l'irrégularité des mesures de la caméra a été traitée avec un observateur *multi-rate*. Même si le schéma de l'observateur/prédicteur a été étudié avec une entrée de commande linéaire

(voir 6.5), les résultats montrent que le principe fonctionne aussi avec une loi de commande non linéaire basée sur des fonctions de saturations. Enfin le schéma est plutôt simple et peut facilement être embarqué dans un microcontrôleur. L'objectif est maintenant d'adapter ce schéma au quadrirotor et de l'utiliser par exemple pour l'algorithme de suivi de mur présenté précédemment.

Conclusions et perspectives

Conclusions

Nous nous sommes intéressés au début de cette thèse aux différentes configurations existantes de drones, afin de sélectionner celle qui est la plus adaptée à nos expériences. Le quadrirotor s'est ainsi rapidement imposé car il a été étudié par de nombreuses équipes (y compris la notre qui a une forte expérience sur cette plateforme), des lois de commandes pour le stabiliser ayant déjà été proposées avec succès. Ce drone est capable de vol stationnaire, ce qui est un atout pour utiliser des algorithmes de vision, et ce qui facilite aussi les expériences car elles peuvent être menées dans un espace relativement réduit.

Une plateforme expérimentale a alors été construite en intégralité. Notre équipe ayant l'habitude jusque là de travailler sur des structures venant du *DraganFly*, celle que nous avons réalisée a été faite directement à partir de profilés en aluminium. La rigidité a ainsi été améliorée (mais légèrement au détriment du poids), et les coûts ont été grandement réduits. L'électronique embarquée a aussi été repensée, les solutions employées habituellement par l'équipe étant devenues trop limitées. Le microcontrôleur utilisé est maintenant plus puissant et un système de mesure de la vitesse de rotation des moteurs a été mis en place. Une station sol a aussi été développée afin de pouvoir envoyer des ordres de haut niveau au drone sans passer par la radiocommande classique, trop limitée. Les gains de la loi de commande sont alors réglables en temps réel ce qui apporte plus de souplesse. Enfin les problèmes de pertes de données de la communication sans fil ont été abordés, en réalisant un protocole fiable entre la station sol et le drone grâce à la priorétisation de certaines trames importantes. La stabilité de la plateforme finale a été démontrée notamment lors du challenge minidrone organisé par l'*Onera* et la *DGA*. Des essais de vol ont alors pu être faits dans une soufflerie, avec un vent allant jusqu'à 4 m/s.

Afin d'améliorer la sécurité, il a été développé un simulateur permettant de tester au sol et sans danger tous les programmes et algorithmes (y compris ceux de vision) sur un drone virtuel. Ce simulateur, relié à la station sol, permet de valider toute la chaine et d'éliminer les (nombreux) *bugs* de programmation avant d'effectuer le vol réel. Les programmes peuvent alors être plus complexes puisqu'ils peuvent être testés facilement sans avoir recours au drone. De plus, les temps de développement sont réduits car tout se fait sur un seul ordinateur, sans avoir à reprogrammer un microcontrôleur (opération relativement longue) ni avoir à attendre que les batteries se rechargent. Enfin, une seule personne est nécessaire pour tout mettre en œuvre, alors que lors des vols réels il y en a toujours au moins deux : une à côté du drone pour le rattraper en cas de problème, et une autre sur la station sol.

Plusieurs lois de commandes ont ensuite été étudiées sur une plateforme de type *PVTOL* qui peut notamment représenter une bonne approximation du quadrirotor si les dynamiques de roulis et de tangage sont supposées découplées. Notre choix s'est orienté vers des lois de commandes par fonctions de saturations, et une loi permettant de stabiliser un système à n intégrateurs a alors été proposée. Celle-ci se différencie des travaux précédents car chaque saturation ne joue que sur un seul état du système ce qui la rend plus pratique à régler sur le système réel.

Des algorithmes de vision ont aussi été étudiés, afin d'effectuer de l'asservissement visuel. Un des objectifs de la thèse étant d'avoir le traitement d'images embarqué, ces algorithmes ont dû être revus pour tenir dans la carte embarquée. Celle-ci a en effet une puissance limitée, même si elle semblait parmi les plus performantes au début de cette thèse. Un algorithme de flux optique à une dimension a donc été testé sur un robot mobile de type *e-puck*, afin de valider l'évitement d'obstacle. Ce système pourrait être adapté au drone mais n'a pas été fait. Des systèmes de vision par laser ont ensuite été étudiés, en s'inspirant de la stéréovision. Ainsi, un système permettant d'estimer les angles de roulis et tangage a été réalisé. Les résultats sont corrects mais la cadence obtenue (celle de la caméra) est trop faible pour pouvoir stabiliser le drone. Cette expérience montre néanmoins que la centrale inertielle est indispensable pour la stabilisation du drone, et que la vision peut venir en complément afin d'asservir la position par exemple, ou de recaler les mesures d'angles dérivantes fournies par la centrale. Un autre système a été construit autour d'un laser ligne afin d'effectuer du suivi de mur. Des simulations ont été réalisées et ont validé le concept ainsi que les programmes écrits. Les expériences sont alors en cours de réalisations.

Enfin, un schéma utilisant un prédicteur et un observateur a été proposé. Il permet de résoudre les problèmes de retards introduits par la caméra, et autorise un fonctionnement en *multi-rate*. En effet, la période d'échantillonnage des infor-

mations visuelles n'est pas fixe et est bien inférieure à celle de la loi de commande (qui elle par contre est fixe). Le concept a été validé sur un chariot mobile à un axe, muni d'un système de mesure de distance par vision laser. Cette solution est donc intéressante pour des systèmes de vision embarquée, dont les temps de calculs peuvent être longs, c'est à dire avec des retards importants et une faible cadence. Cela peut représenter une alternative (ou une complémentarité) aux algorithmes de vision simples présentés précédemment.

Perspectives

Les perspectives et travaux futurs sont nombreux. En effet il a été soulevé plusieurs points lors de cette thèse qui pourraient être améliorés.

Tout d'abord, la plateforme en elle-même peut être améliorée. Il a été vu par exemple que l'asservissement en vitesse des moteurs améliore leur réponse, cependant la solution proposée n'est que temporaire et doit mener à terme à la conception de *drivers* pour moteurs *brushless*. Ceux-ci permettraient de faire directement l'asservissement de vitesse, en soulageant cette tâche du microcontrôleur principal. De plus, les *drivers* commerciaux actuels utilisent des microcontrôleurs classiques bon marchés, alors que les constructeurs de microcontrôleurs proposent maintenant des solutions dédiées beaucoup plus efficaces. Celles-ci permettraient un meilleur contrôle du moteur, et sans doute une économie d'énergie. De grands efforts doivent être faits dans ce sens afin d'augmenter l'autonomie du drone. Ainsi, la carte électronique réalisée utilise de nombreux régulateurs de tension or ceux-ci sont peu efficaces car la majeur partie de la puissance est perdue par dégagement de chaleur ; des convertisseurs de tension seraient alors beaucoup plus efficaces.

En ce qui concerne la vision, effectuer le traitement embarqué reste encore un défi aujourd'hui ; la solution utilisée pour cette thèse étant encore trop limitée. Les nouvelles technologies et notamment l'*OMAP 3530* semblent très prometteuses. Son processeur *ARM* est en effet plus puissant que celui que nous avons utilisé mais il est surtout épaulé par un *DSP* capable d'effectuer du traitement d'images plus évolué. Cependant, une très bonne compréhension de l'architecture *OMAP 3530* et de nombreuses optimisations sont nécessaires afin d'obtenir un système fonctionnel. Le travail est donc conséquent, mais nécessaire car il n'existe actuellement aucune plateforme "clé en main" permettant d'implémenter facilement des algorithmes de vision comme il est possible de le faire sur un ordinateur avec *OpenCV* par exemple. Une fois le système en place, des traitements beaucoup plus évolués seront possibles. Notamment, les algorithmes de flux optique pour contrôler les vitesses de déplacement et/ou faire de l'évitement d'obstacle pourraient être testés, ou bien des algorithmes d'odométrie visuelle sur des repères naturels.

La centrale inertielle utilisée a montré des problèmes de dérives. Un travail intéressant mais ambitieux serait d'effectuer soi-même la fusion des capteurs inertiels (et des magnétomètres), afin d'arriver à une meilleure estimation. Notamment, les informations visuelles pourraient aider à améliorer l'estimation en corrigeant les dérives ou en filtrant les accélérations latérales. De plus, la mesure du champ magnétique pourrait être améliorée en filtrant le champ créé par les moteurs grâce à la connaissance de leur vitesse de rotation. La dynamique de l'engin pourrait aussi être prise en compte, cependant cela nécessiterait un modèle relativement précis du drone que nous ne possédons pas actuellement.

Enfin, l'intérêt d'un prédicteur/observateur a été démontré en pratique sur un système à retard. Cette technique doit être étendue au quadrirotor afin d'améliorer le système de vision. Ainsi, la faible fréquence de la caméra pourrait être virtuellement augmentée. Le cas du drone reste cependant plus compliqué à traiter car le schéma présenté est construit pour un système linéaire or le modèle complet du quadrirotor ne l'est pas. Il serait donc intéressant de travailler sur des observateurs non linéaires par exemple.

Annexe A

Compléments de calculs sur la loi par *backstepping*

Réécrivons y_4 en fonction de x_1, x_2, ϕ_1 et ϕ_2 :

$$\begin{aligned}
y_4 &= g(1+\tan^2\phi_1)\phi_2 - \alpha_3 \\
&= g(1+\tan^2\phi_1)\phi_2 - (-k_3 y_3 + \dot{\alpha}_2) \\
&= g(1+\tan^2\phi_1)\phi_2 + k_3(g\tan\phi_1 - \alpha_2) - (k_2\dot{y}_2 - \ddot{\alpha}_1) \\
&= g(1+\tan^2\phi_1)\phi_2 + k_3(g\tan\phi_1 - k_2 y_2 + \dot{\alpha}_1) - k_2(-g\tan\phi_1 - \dot{\alpha}_1) - k_1\ddot{x}_1 \\
&= g(1+\tan^2\phi_1)\phi_2 + k_3\Big(g\tan\phi_1 - k_2(x_2-\alpha_1) - k_1\dot{x}_1\Big) - k_2(-g\tan\phi_1 + k_1\dot{x}_1) - k_1\dot{x}_2 \\
&= g(1+\tan^2\phi_1)\phi_2 + k_3\Big(g\tan\phi_1 - k_2(x_2+k_1x_1) - k_1x_2\Big) - k_2(-g\tan\phi_1 + k_1x_2) + k_1 g\tan\phi_1 \\
&= g(1+\tan^2\phi_1)\phi_2 + (k_1+k_2+k_3)g\tan\phi_1 - (k_1k_2+k_1k_3+k_2k_3)x_2 - k_1k_2k_3x_1 \qquad \text{(A.1)}
\end{aligned}$$

Réécrivons $\dot{\alpha}_3$ en fonction de x_1, x_2, ϕ_1 et ϕ_2 :

$$
\begin{aligned}
\dot{\alpha}_3 &= -k_3\dot{y}_3 + \ddot{\alpha}_2 \\
&= -k_3\Big(g(1+\tan^2\phi_1)\phi_2 - \dot{\alpha}_2\Big) + k_2\ddot{y}_2 - \dddot{\alpha}_1 \\
&= -k_3\Big(g(1+\tan^2\phi_1)\phi_2 - k_2\dot{y}_2 + \ddot{\alpha}_1\Big) + k_2\Big(-g(1+\tan^2\phi_1)\phi_2 - \ddot{\alpha}_1\Big) + k_1\dddot{x}_1 \\
&= -k_3\Big(g(1+\tan^2\phi_1)\phi_2 - k_2(-g\tan\phi_1 - \dot{\alpha}_1) - k_1\ddot{x}_1\Big) \\
&\quad +k_2\Big(-g(1+\tan^2\phi_1)\phi_2 + k_1\ddot{x}_1\Big) + k_1\ddot{x}_2 \\
&= -k_3\Big(g(1+\tan^2\phi_1)\phi_2 - k_2(-g\tan\phi_1 + k_1x_2) - k_1\dot{x}_2\Big) \\
&\quad +k_2\Big(-g(1+\tan^2\phi_1)\phi_2 + k_1\dot{x}_2\Big) - k_1g(1+\tan^2\phi_1)\phi_2 \\
&= -k_3\Big(g(1+\tan^2\phi_1)\phi_2 - k_2(-g\tan\phi_1 + k_1x_2) + k_1g\tan\phi_1\Big) \\
&\quad +k_2\Big(-g(1+\tan^2\phi_1)\phi_2 - k_1g\tan\phi_1\Big) - k_1g(1+\tan^2\phi_1)\phi_2 \\
&= -(k_1+k_2+k_3)g(1+\tan^2\phi_1)\phi_2 - (k_1k_2+k_1k_3+k_2k_3)g\tan\phi_1 + k_1k_2k_3x_2
\end{aligned}
\tag{A.2}
$$

Annexe B

Notations du chapitre 5

Cette section présente l'ensemble des notations utilisées dans le chapitre 5.

B.1 Points

Un point P de coordonnées (x, y, z) est représenté par une matrice colonne formée de ses coordonnées :

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{B.1}$$

Les coordonnées homogènes (x, y, z) représentent le point $2D$ $(x/z, y/z)$. De même, les coordonnées homogènes (x, y, z, t) représentent le point $3D$ $(x/t, y/t, z/t)$. Les coordonnées homogènes d'un point P seront notées $\tilde{P}$.

B.2 Vecteurs

De même que pour les points, un vecteur $\overrightarrow{V}(x, y, z)$ est représenté par une matrice colonne formée de ses composantes :

$$V = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{B.2}$$

On associe à ce vecteur $\overrightarrow{V}$ la matrice antisymétrique suivante :

$$[V]_\times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \tag{B.3}$$

Ainsi, le produit scalaire de deux vecteurs $\overrightarrow{U}$ et $\overrightarrow{V}$ peut être défini par le produit matriciel :

$$\overrightarrow{U}.\overrightarrow{V} = U^T V \tag{B.4}$$

et le produit vectoriel de $\overrightarrow{U}$ et $\overrightarrow{V}$ est défini par le produit matriciel :

$$\overrightarrow{U} \wedge \overrightarrow{V} = [U]_\times V = \left(U[V]_\times\right)^T \tag{B.5}$$

La norme d'un vecteur $\overrightarrow{V}$ peut s'écrire :

$$\|\overrightarrow{V}\| = \sqrt{x^2 + y^2 + z^2} = \sqrt{V^T V} \tag{B.6}$$

Enfn, un vecteur $\overrightarrow{V}$ peut être défini par deux points $P_1 = [x_1, y_1, z_1]^T$ et $P_2 = [x_2, y_2, z_2]^T$:

$$V = P_1 - P_2 = \begin{bmatrix} x_1 - x_2 \\ y_1 - y_2 \\ z_1 - z_2 \end{bmatrix} \tag{B.7}$$

La notation $\overrightarrow{V}$ pour un vecteur sera donc remplacée par une matrice V par la suite, les calculs vectoriels se ramenant à des calculs matriciels.

B.3 Plans, lignes

Un plan π dans l'espace est représenté par son équation :

$$\pi_1 x + \pi_2 y + \pi_3 z + \pi_4 = 0 \tag{B.8}$$

La représentation homogène d'un plan peut donc s'écrire :

$$\tilde{\pi} = [\pi_1, \pi_2, \pi_3, \pi_4]^T \tag{B.9}$$

Un point M appartient au plan π si et seulement si :

$$\tilde{\pi}^T \tilde{M} = 0 \tag{B.10}$$

Une ligne l dans un plan est représentée par son équation :

$$ax + by + c = 0 \tag{B.11}$$

La représentation homogène d'un ligne peut donc s'écrire :

$$\tilde{l} = [a, b, c]^T \tag{B.12}$$

Un point M du plan appartient alors à la ligne l si et seulement si :

$$\tilde{M}^T \tilde{l} = 0 \tag{B.13}$$

L'intersection de deux lignes l et l' est le point M du plan tel que :

$$\tilde{M} = [\tilde{l}]_\times \tilde{l'} \tag{B.14}$$

La ligne l passant par les points M et M' du plan vérifie :

$$\tilde{l} = [\tilde{M}]_\times \tilde{M'} \tag{B.15}$$

B.4 Quaternions

Cette partie présente quelques résultats sur les quaternions qui seront utilisés par la suite. Pour plus de détails sur les quaternions, se référer par exemple à [Kui98] ou [KT82].

Un quaternion est un quadruplet de réels formé d'un scalaire et d'un vecteur. Un quaternion sera noté avec l'exposant " ° " :

$$\mathring{q} = q_0 + q_x i + q_y j + q_z k \tag{B.16}$$

où q_0 représente la partie scalaire, (q_x, q_y, q_z) représente la partie vectorielle et (i, j, k) satisfont les relations suivantes :

$$i^2 = -1 \tag{B.17a}$$
$$j^2 = -1 \tag{B.17b}$$
$$k^2 = -1 \tag{B.17c}$$
$$ij = k \tag{B.17d}$$
$$jk = i \tag{B.17e}$$
$$ki = j \tag{B.17f}$$
$$ji = -k \tag{B.17g}$$
$$kj = -i \tag{B.17h}$$
$$ik = -j \tag{B.17i}$$

Le quaternion peut alors s'écrire sous forme matricielle :

$$\mathring{q} = \begin{bmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{bmatrix} \tag{B.18}$$

Soit $\mathring{r} = r_0 + r_x i + r_y j + r_z k$ un quaternion. L'addition de deux quaternions est définie composant par composant :

$$\mathring{q} + \mathring{r} = q_0 + r_0 + (q_x + r_x)i + (q_y + r_y)j + (q_z + r_z)k \tag{B.19}$$

Le produit de deux quaternions, noté "$\otimes$", s'obtient en développant le produit formellement, soit :

$$\begin{aligned} \mathring{r} \otimes \mathring{q} \;=\; & (r_0 q_0 - r_x q_x - r_y q_y - r_z q_z) \\ & + (r_0 q_x + r_x q_0 + r_y q_z - r_z q_y) i \\ & + (r_0 q_y - r_x q_z + r_y q_0 + r_z q_x) j \\ & + (r_0 q_z + r_x q_y - r_y q_x + r_z q_0) k \end{aligned} \tag{B.20}$$

Le produit n'est donc en général pas commutatif. En définissant deux matrices $\mathring{r}^g$ et $\mathring{r}^d$ correspondant respectivement au produit à gauche et à droite, le produit de deux quaternions peut se mettre sous forme matricielle :

$$\mathring{r} \otimes \mathring{q} = \begin{bmatrix} r_0 & -r_x & -r_y & -r_z \\ r_x & r_0 & -r_z & -r_y \\ r_y & r_z & r_0 & -r_x \\ r_z & -r_y & r_x & r_0 \end{bmatrix} \mathring{q} = \mathring{r}^g \mathring{q} \tag{B.21}$$

et :

$$\mathring{q} \otimes \mathring{r} = \begin{bmatrix} r_0 & -r_x & -r_y & -r_z \\ r_x & r_0 & r_z & -r_y \\ r_y & -r_z & r_0 & r_x \\ r_z & r_y & -r_x & r_0 \end{bmatrix} \mathring{q} = \mathring{r}^d \mathring{q} \tag{B.22}$$

Le produit scalaire de deux quaternions est noté ".", c'est la somme des produits de chaque composants :

$$\mathring{q}.\mathring{r} = q_0 r_0 + q_x r_x + q_y r_y + q_z r_z \tag{B.23}$$

soit, sous forme matricielle :

$$\mathring{q}.\mathring{r} = \mathring{q}^T \mathring{r} \tag{B.24}$$

Un quaternion unitaire est un quaternion de norme 1 ; la norme d'un quaternion étant définie par la relation suivante :

$$\|\mathring{q}\|^2 = \mathring{q}.\mathring{q} = \mathring{q}^T \mathring{q} \tag{B.25}$$

Un quaternion unitaire $\mathring{q}$ est associé à la matrice de rotation suivante :

$$R = \begin{bmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_0 q_z) & 2(q_x q_z + q_0 q_y) \\ 2(q_x q_y + q_0 q_z) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_0 q_x) \\ 2(q_x q_z - q_0 q_y) & 2(q_y q_z + q_0 q_x) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{B.26}$$

Inversement, une matrice de rotation R (d'éléments R_{ij}) est associée au quaternion suivant :

$$q_0 = \frac{1}{2}\sqrt{1 + R_{11} + R_{22} + R_{33}} \tag{B.27a}$$

$$q_x = \frac{1}{4q_0}(R_{32} - R_{23}) \tag{B.27b}$$

$$q_y = \frac{1}{4q_0}(R_{13} - R_{31}) \tag{B.27c}$$

$$q_z = \frac{1}{4q_0}(R_{21} - R_{12}) \tag{B.27d}$$

Le quaternion formé ci-dessus n'est pas nécessairement unitaire ; pour cela il suffit de le diviser par sa norme. Le quaternion unitaire $\mathring{r}$ associé à $\mathring{q}$ est donc :

$$\mathring{r} = \frac{\mathring{q}}{\|\mathring{q}\|} \tag{B.28}$$

Annexe C

Compléments de calculs sur l'algorithme de suivi de mur

Nous exprimons ici l'expression des coordonnées du point A (voir figures 5.37 et C.1) dans le repère du drone (O, x'_f, y'_f).

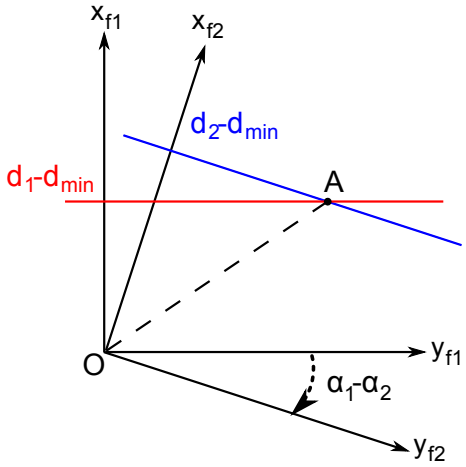


Figure C.1 – Position du point A.

Le point A est le point d'intersection de la droite $x = d_2 - d_{min}$ dans le repère (O, x_{f2}, y_{f2}) avec la droite $x = d_1 - d_{min}$ dans le repère (O, x_{f1}, y_{f1}). Réécrivons

cette dernière dans le repère (O, x_2, y_2), l'équation est de la forme :

$$y = mx + p \tag{C.1}$$

or le vecteur

$$\begin{bmatrix} \sin(\alpha_1 - \alpha_2) \\ \cos(\alpha_1 - \alpha_2) \end{bmatrix} \tag{C.2}$$

est colinéaire à la droite, d'où :

$$m = \frac{\cos(\alpha_1 - \alpha_2)}{\sin(\alpha_1 - \alpha_2)} \tag{C.3}$$

puis le point $(d_1 - d_{min}, 0)$ dans le repère (O, x_{f1}, y_{f1}) appartenant à cette droite, on en déduit dans (O, x_{f2}, y_{f2}) :

$$-(d_1 - d_{min})\sin(\alpha_1 - \alpha_2) = \frac{\cos(\alpha_1 - \alpha_2)}{\sin(\alpha_1 - \alpha_2)}(d_1 - d_{min})\cos(\alpha_1 - \alpha_2) + p \tag{C.4}$$

puis,

$$p = -(d_1 - d_{min})\Big(\sin(\alpha_1 - \alpha_2) + \frac{\cos(\alpha_1 - \alpha_2)}{\sin(\alpha_1 - \alpha_2)}\cos(\alpha_1 - \alpha_2)\Big) \tag{C.5a}$$

$$= -\frac{d_1 - d_{min}}{\sin(\alpha_1 - \alpha_2)} \tag{C.5b}$$

Le point A a donc pour coordonnées dans (O, x_{f2}, y_{f2}) :

$$x_A = d_2 - d_{min} \tag{C.6a}$$

$$y_A = \frac{\cos(\alpha_1 - \alpha_2)}{\sin(\alpha_1 - \alpha_2)}(d_2 - d_{min}) - \frac{d_1 - d_{min}}{\sin(\alpha_1 - \alpha_2)} \tag{C.6b}$$

puis dans $(O, x_{f'}, y_{f'})$:

$$x = x_A \cos\alpha_2 + y_A \sin\alpha_2 \tag{C.7a}$$

$$y = -x_A \sin\alpha_2 + y_A \cos\alpha_2 \tag{C.7b}$$

Annexe D

Compléments de calculs sur le modèle du chariot

En introduisant μN_1 de (6.20a) dans (6.20c), il s'ensuit :

$$\tau - r(m\ddot{x} + \overrightarrow{F_{ch/1}}.\overrightarrow{x}) = \frac{J}{r}\ddot{x} \tag{D.1}$$

En introduisant $\overrightarrow{F_{ch/1}}.\overrightarrow{x}$ de (6.22a) dans ce qu'il précède :

$$\tau - r(m\ddot{x} + M\ddot{x} + \overrightarrow{F_{ch/2}}.\overrightarrow{x}) = \frac{J}{r}\ddot{x} \tag{D.2}$$

En introduisant $\overrightarrow{F_{ch/2}}.\overrightarrow{x}$ de (6.21a),

$$\tau - r(m\ddot{x} + M\ddot{x} + m\ddot{x} + \mu N_2) = \frac{J}{r}\ddot{x} \tag{D.3}$$

En introduisant μN_2 de (6.21c), il s'ensuit :

$$\tau - r\Big(m\ddot{x} + M\ddot{x} + m\ddot{x} + \frac{J}{r^2}\ddot{x}\Big) = \frac{J}{r}\ddot{x} \tag{D.4}$$

d'où :

$$\tau = \Big(2mr + Mr + 2\frac{J}{r}\Big)\ddot{x} \tag{D.5}$$

soit,

$$\ddot{x} = k_\tau \tau \tag{D.6}$$

avec :

$$k_\tau = \frac{1}{2mr + Mr + 2\frac{J}{r}} \tag{D.7}$$

Annexe E

Publications

Ce travail de thèse a donné lieu aux publications suivantes :

1. E. Rondon, I. Fantoni-Coichot, A. Sanchez et G. Sanahuja, *Optical Flow-Based Controller for Reactive and Relative Navigation dedicated to a Four Rotor Rotorcraft*. Proceedings de IEEE/RSJ International Conference on Intelligent Robots and Systems, Saint Louis, États-Unis, 2009.
2. G. Sanahuja, P. Castillo et A. Sanchez, *Stabilization of n integrators in cascade with bounded input with experimental application to a VTOL laboratory system*, International Journal of Robust and Nonlinear Control. DOI : 10.1002/rnc.1494.
3. G. Sanahuja, P. Garcia, P. Castillo et P. Albertos, *Control of unstable delayed systems with input saturations and measurement constraints : An electrical cart application*, Proceedings de International Federation of Automatic Control World Congress, Séoul, Corée, 2008.
4. G. Sanahuja, P. Castillo, O. Garcia et R. Lozano, *Linear and nonlinear control strategies to stabilize a VTOL aircraft comparative analysis*, Proceedings de IFAC Symposium on Intelligent Autonomous Vehicles, Toulouse, France, 2007.
5. G. Sanahuja, P. Castillo, O. Garcia et R. Lozano, *New Trends on Information Technology Applied to Communications, Control, and Optics Eds*, chapitre “Linear and nonlinear control strategies to stabilize a VTOL aircraft : comparative analysis”, 2007.

Bibliographie

[ALD03] J. Alves, J. Lobo, and J. Dias. Camera-inertial sensor modeling and alignment for visual navigation. In 11^{th} *International Conference on Advanced Robotics*, Coimbra, Portugal, juillet 2003. [p. 135]

[APS07] P. Albertos, I. Peñarrocha, and R. Sanchis. Virtual sensor. An overview. In *The First International Conference on Industrial Informatics*, Mexico, novembre 2007. [p. 191, 192]

[BC05] G. Le Besnerais and F. Champagnat. Dense optical flow estimation by iterative local window registration. In *IEEE International Conference on Image Processing*, Genève, Italie, septembre 2005. [p. 75]

[BFB94] J.L. Barron, D.J. Fleet, and S.S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1) :43–77, 1994. [p. 140]

[BIDM91] I.Y. Bar-Izthack, J. Deutschmann, and F.L. Markley. Quaternion normalization in aditive EKF for spacecraft attitude estimation. In *AIAA Guidance, Navigation, and Control Conference*, La Nouvelle Orléans, États-Unis, août 1991. [p. 49]

[BK97] A.J. Baerveldt and R. Klang. A low-cost and low-weight attitude estimation system for anautonomous helicopter. In *IEEE International Conference on Intelligent Engineering Systems*, Budapest, Hongrie, 1997. [p. 48]

[BMSP09] P.J. Bristeau, P. Martin, E. Salaun, and N. Petit. The role of propeller aerodynamics in the model of a quadrotor UAV. In *IEEE European Control Conference*, Budapest, Hongrie, 2009. [p. 25]

[Bou98] S. Bougnoux. From projective to euclidean space under any practical situation, a criticism of self- calibration. In *International Conference on Computer Vision*, janvier 1998. [p. 129]

[BS06] A. Barron and M.V. Srinivasan. Visual regulation of ground speed and headwind compensation in freely flying honey bees (Apis mellifera L.). *Journal of Experimental Biology*, 209(5) :978–984, 2006. [p. 141]

[BZF09] A. Beyeler, J.C. Zufferey, and D. Floreano. OptiPilot : control of take-off and landing using optic flow. In *European Micro Air Vehicle conference and competition*, 2009. [p. 142, 143]

[Che91] H. Chen. A screw-motion approach to uniqueness analysis of head-eye geometry. In *IEEE Conference on Computer Vision and Pattern Recognition*, 3-6 juin 1991. [p. 134]

[Cho03] D. Choukroun. *Novel methods for attitude determination using vector observations*. PhD thesis, Israel Institute of Technology, Haifa, Israël, 2003. [p. 49]

[CK91] J. Chou and M. Kamel. Finding the position and orientation of a sensor on a robot manipulator using quaternions. *International Journal of Robotics Research*, 10(3) :240–254, 1991. [p. 134]

[CLD05] P. Castillo, R. Lozano, and A. Dzul. *Modelling and Control of Mini-Flying Machines*. Springer-Verlag in Advances in Industrial Control, juillet 2005. [p. 84, 88, 100]

[CM03] J.L. Crassidis and F.L. Markley. Unscented filtering for spacecraft attitude estimation. *Journal of Guidance, Control, and Dynamics*, 26(4) :536–542, 2003. [p. 49]

[CM08] A. Chemori and N. Marchand. A prediction-based nonlinear controller for stabilization of a non-minimum phase PVTOL aircraft. *International Journal of Robust and nonlinear control*, 18 :876–889, 2008. [p. 85]

[CPB+09] F. Champagnat, A. Plyer, G. Le Besnerais, B. Leclaire, and Y. Le Sant. How to calculate dense PIV vector fields at video rate. In 8^{th} *International Symposium On Particle Image Velocimetry*, Melbourne, Australie, 25-28 août 2009. [p. 75]

[CR93] D. Coombs and K. Roberts. Centering behaviour using peripheral vision. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Los Alamitos, États-Unis, 1993. [p. 142]

[CT07] L. Consolini and M. Tosques. On the VTOL exact tracking with bounded internal dynamics via a poincaré map approach. *IEEE Transactions on Automatic Control*, 52(9) :1757–1762, September 2007. [p. 85]

[dal] Forum dédié a l'*OSBDM*. "`http://forums.freescale.com/freescale/board?boa`
[p. 63]

[Dan99] K. Daniilidis. Hand-eye calibration using dual quaternions. *The International Journal of Robotics Research*, 18(3) :286–298, mars 1999. [p. 133, 134]

[dedsdtpfo] Vidéos d'atterrissage et de suivi de terrain par flux optique. "`http://brunoherisse.free.fr/page.php?part=demo&lang=fr`". [p. 13]

[DW94] A.P. Duchon and W.H. Warren. Robot navigation from a gibsonian viewpoint. In *IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, 1994. [p. 142]

[Egb07] J. Egbert. Low-altitude road following, using strap-down cameras on miniature aerial vehicles. Master's thesis, Department of Electrical and Computer Engineering, Brigham Young University, décembre 2007. [p. 7]

[EKMH07] M. Euston, J. Kim, R. Mahony, and T. Hamel. A complementary filter for attitude estimation of a fixed-wing UAV with a low-cost IMU. In *International Conference on Field and Service Robotics*, Chamonix Mont-Blanc, France, 9-12 juillet 2007. [p. 48]

[ESGL08] J. Escareño, A. Sanchez, O. Garcia, and R. Lozano. Triple tilting rotor mini-UAV : Modeling and embedded control of the attitude. In *American Control Conference*, Seattle, États-Unis, 11-13 juin 2008. [p. 11]

[Fau71] P. Faurre. *Navigation inertielle et filtrage stochastique. Méthodes mathématiques de l'informatique.* Dunod, 1971. [p. 50]

[Fau93] O. Faugeras. *Three-Dimensional Computer Vision : a Geometric Viewpoint.* MIT Press, 1993. [p. 129]

[FL01] I. Fantoni and R. Lozano. Control of nonlinear mechanical systems. *European Journal of Control, Special Issue on ECC'01 : Fundamental Issues in Control*, 47(2-3), Novembre 2001. [p. 85]

[FL02] I. Fantoni and R. Lozano. *Nonlinear Control for Underactuated Mechanical Systems.* Springer-Verlag, 2002. [p. 84]

[FLC02] I. Fantoni, R. Lozano, and P. Castillo. A simple stabilization algorithm for the PVTOL aircraft. In *International Federation of Automatic Control World Congress*, Barcelone, Espagne, 2002. [p. 85]

[Fou52] L. Foucault. Sur les phénomènes d'orientation des corps tournants entraînés par un axe fixe à la surface de la terre. *Comptes rendus hebdomadaires des séances de l'Académie des Sciences (Paris)*, 35 :424–427, 1852. [p. 45]

[FRS07] N. Franceschini, F. Ruffier, and J. Serres. A bio-inspired flying robot sheds light on insect piloting abilities. *Current biology*, 17(4) :329–335, février 2007. [p. 142]

[FZL02] I. Fantoni, A. Zavala, and R. Lozano. Global stabilization of a PVTOL aircraft with bounded thrust. In *IEEE Conference on Decision and Control*, Las Vegas, États-Unis, December 2002. [p. 85]

[GC08] J.F. Guerrero-Castellanos. *Estimation de l'attitude et commande bornée en attitude d'un corps rigide : application à un mini hélicoptère à quatre rotors.* PhD thesis, Université Joseph Fourier, 2008. [p. 50]

[GCLA06] P. Garcia, P. Castillo, R. Lozano, and P. Albertos. Robustness with respect to delay uncertainties of a predictor-observer based discrete-time controller. In *IEEE Conference on Decision and Control*, San Diego, Etats-Unis, décembre 2006. [p. 189, 190]

[GCLA07] P. Garcia, P. Castillo, R. Lozano, and P. Albertos. Robustness with respect to delay uncertainties of a predictor-observer based controller for time-delay systems. *Automatica*, 40(3) :603–612, 2007. [p. 189, 191]

[GERLon] O. Garcia, J. Escareño, V. Rosas, and R. Lozano. *Unmanned Aerial Vehicles Embedded Control*, chapter Modeling and control of a convertible plane. John Wiley-ISTE Ltd, En préparation. [p. 48]

[GO05] W.E. Green and P.Y. Oh. A MAV that flies like an airplane and hovers like a helicopter. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2005. [p. 17]

[Goe] D. Goel. CMUcam3 face detector, "`http://www.cmucam.org/wiki/viola-jones`". [p. 67]

[Gol80] H. Goldstein. *Classical Mechanics.* Addison Wesley Series in Physics. Adison Wesley, 1980. [p. 28]

[GSB99] F. Grognard, R. Sepulchre, and G. Bastin. Global stabilization of feedforward systems with exponentially unstable jacobian linearization. *Systems and Control Letters*, 37(2) :107–115, 1999. [p. 105]

[GSEL08] O. Garcia, A. Sanchez, J. Escareño, and R. Lozano. Tail-sitter UAV having one tilting rotor : Modeling, control and real-time experiments. In *International Federation of Automatic Control World Congress*, Séoul, Corée, 2008. [p. 8]

[GSLS03] G. Gebert, D. Snyder, J. Lopez, and N. Siddiqi. Optical flow angular rate determination. In *International Conference on Image Processing*, 2003. [p. 142]

[GWA01] M.S. Grewal, L.R. Weill, and A.P. Andrews. *Global positioning systems, inertial navigation, and integration.* Wiley Inter-science, 2001. [p. 50]

[Har94a] R.I. Hartley. An algorithm for self calibration from several views. In *IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, États-Unis, juin 1994. [p. 129]

[Har94b] R.I. Hartley. Projective reconstruction and invariants from multiple images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16 :1036–1041, octobre 1994. [p. 146]

[HD95] R. Horaud and F. Dornaika. Hand-eye calibration. *International Journal of Robotics Research*, 14(3) :195–210, 1995. [p. 134]

[HHMR09] B. Herisse, T. Hamel, R. Mahony, and F.X. Russotto. A nonlinear terrain-following controller for a VTOL unmanned aerial vehicle using translational optical flow. In *IEEE International Conference on Robotics and Automation*, Kobe, Japon, 12-17 mai 2009. [p. 13, 142]

[Hor87] K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4(4), avril 1987. [p. 135, 136, 137]

[Hor93] I. Horswill. Polly : A vision-based artificial agent. In *The National Conference on Artificial Intelligence*, Washington, États-Unis, 11-15 juillet 1993. [p. 67]

[HRHM08] B. Herisse, F.X. Russotto, T. Hamel, and R. Mahony. Hovering flight and vertical landing control of a VTOL unmanned aerial vehicle using optical flow. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, France, 22-26 septembre 2008. [p. 12]

[HS81] K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial intelligence*, 7 :185–203, 1981. [p. 139, 140]

[HS97] R.I. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2) :146–157, 1997. [p. 157]

[HSG08] J.D. Hol, T. B. Schön, and F. Gustafsson. A new algorithm for calibrating a combined camera and IMU sensor unit. In *International Conference on Control, Automation, Robotics and Vision*, Hanoï, Vietnam, 17-20 décembre 2008. [p. 135]

[HSM92] J. Hauser, S. Sastry, and G. Meyer. Nonlinear control design for slightly nonminimum phase systems : Application to V/STOL aircraft. *Automatica*, 28 :665–679, 1992. [p. 84, 86]

[HZ03] R.I. Hartley and A. Zisserman. *Multiple View Geometry in computer vision.* Cambridge Press, 2003. [p. 144, 145, 146, 147, 148]

[id] Site internet d'*Analog Devices*. "http://www.analog.com". [p. 46]

[idl] Site internet de l'*Onera*. "http://www.onera.fr". [p. 46]

[idldt] Site internet de l'*AR.Drone* de *Parrot*. "http://ardrone.parrot.com". [p. 13, 14]

[idlsrdàt] Site internet de l'agence spatial russe dédié à *GLONASS*. "http://www.glonass-ianc.rsa.ru". [p. 52]

[idlt] Site internet de la *CMUCam*. "http://www.cs.cmu.edu/~cmucam". [p. 68]

[idpt] Site internet du projet *PixHawk*. "http://pixhawk.ethz.ch". [p. 71]

[idta] Site internet de *DelFly*. "http://www.delfly.nl". [p. 20]

[idtb] Site internet de *Freescale*. "http://www.freescale.com". [p. 57, 65]

[idtc] Site internet de *Futaba*. "http://www.futaba-rc.com". [p. 58]

[idtd] Site internet de *Mobisense Systems*. "http://www.mobisensesystems.com". [p. 70]

[idte] Site internet de *POB-Technology*. "http://pob-technology.com". [p. 69]

[idtf] Site internet de *Polhemus*. "http://www.polhemus.com/?page=Motion_Fastrak". [p. 100]

[idtg] Site internet du *California Institute of Technology*. "http://www.vision.caltech.edu". [p. 126, 129]

[idth] Site internet du *LSC*. "http://lsc.univ-evry.fr/Projets/DIRIGEABLE/index.ht [p. 18]

[idtdld] Site internet du *Visual Geometry Group* de l'Université d'Oxford. "http://www.robots.ox.ac.uk/~vgg/hzbook/index.html". [p. 147]

[ita] Site internet *Aéroplanes Détable*. "http://pagesperso-orange.fr/avionsanspilo [p. 2]

[itb] Site internet *Beagle Board*. "http://beagleboard.org". [p. 72]

[itc] Site internet *Draganfly*. "http://www.draganfly.com". [p. 13, 14, 17, 31]

[itd] Site internet *Embest*. "http://www.embedinfo.com". [p. 72]

[ite] Site internet *Gumstix*. "http://www.gumstix.com". [p. 71, 72]

[itf] Site internet *Helicommand*. "http://en.helicommand.com/". [p. 67]

[itg] Site internet *Hokuyo*. "http://www.hokuyo-aut.jp". [p. 54]

[ith] Site internet *Mikrokopter*. "http://www.mikrokopter.de". [p. 32]

[iti] Site internet *Mindsensors*. "http://www.mindsensors.com". [p. 69]

[itj] Site internet *MLB Company*. "http://www.spyplanes.com". [p. 17]

[itk] Site internet *Newpower-Modélisme*. "http://www.newpower-modelisme.fr". [p. 6, 65]

[itl] Site internet *RC and BLDC motor control*. "http://bldc.wikidot.com". [p. 35]

[itm] Site internet *Robotshop*. "http://www.robotshop.ca". [p. 54]

[itn] Site internet *SBG Systems*. "http://www.sbg-systems.com". [p. 51, 65]

[ito] Site internet *Sharp*. "http://sharp-world.com". [p. 55, 65]

[itp] Site internet *SparkFun*. "http://www.sparkfun.com". [p. 59, 65]

[itq] Site internet *Toradex*. "http://www.toradex.com". [p. 71, 74]

[itr] Site internet *Vectron*. "http://www.vectronblackhawk.com". [p. 12]

[its] Site internet *Via*. "http://via.com.tw/en/products/embedded/". [p. 73]

[itt] Site internet *Wikipedia*. "http://fr.wikipedia.org". [p. 44, 46]

[itu] Site internet *xPC Target*. "http://www.mathworks.fr/products/xpctarget". [p. 100]

[itv] Site internet *Xufo*. "http://www.xufo-shop.de". [p. 32]

[itw] Site internet *Yamaha*. "http://www.yamaha-motor.co.jp". [p. 9]

[JSK96] M. Janković, R. Sepulchre, and P.V. Kokotović. Constructive Lyapunov stabilization of nonlinear cascade systems. *IEEE Transactions on Automatic Control*, 41(12) :1723–1735, 1996. [p. 105]

[JU97] S.J. Julier and J.K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *AeroSense : The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, Orlando, États Unis, 1997. [p. 49]

[JUDW95] S.J. Julier, J.K. Uhlmann, and H. F. Durrant-Whyte. A new approach for filtering nonlinear systems. In *American Control Conference*, Seattle, États Unis, 1995. [p. 49]

[KA04] G. Kaliora and A. Astolfi. Nonlinear control of feedforward systems with bounded signals. *IEEE Transactions on Automatic Control*, 49(11) :1975–1990, 2004. [p. 106]

[KA05] G. Kaliora and A. Astolfi. On the stabilization of feedforward systems with bounded control. *Systems and Control Letters*, 54 :263–270, 2005. [p. 106]

[Kal60] R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, 82 :35–45, 1960. [p. 48, 55]

[KD87] J.J. Koenderink and A.J. Van Doorn. Facts on optic flow. *Biological Cybernetics*, 56(4) :247–254, juin 1987. [p. 138]

[KD04] G.S.W. Klein and T.W. Drummond. Tightly integrated sensor fusion for robust visual tracking. *Image and Vision Computing*, 22(10) :769–776, septembre 2004. [p. 134]

[KMM+96] D. Khadraoui, G. Motyl, P. Martinet, J. Gallice, and F. Chaumette. Visual servoing in robotics scheme using a camera/laser-stripe sensor : Special section on vision-based control of robot manipulators. *IEEE Transactions on Robotics and Automation*, 12(5) :743–750, 1996. [p. 167]

[KS09] J. Kelly and G.S. Sukhatme. *Experimental Robotics*, chapter Fast Relative Pose Calibration for Visual and Inertial Sensors. Springer Berlin / Heidelberg, 2009. [p. 135]

[KT82] P. Kelland and P.G. Tait. *Introduction to quaternions : with numerous examples*. London Macmillan, 1882. [p. 211]

[Kui98] J. Kuipers. *Quaternions and Rotation Sequences*. Princeton University Press, 1998. [p. 211]

[LB95] M. Li and D. Betsis. Handeye calibration. In *International Conference on Computer Vision*, Boston, États-Unis, 20-23 juin 1995. [p. 134]

[LCD04] R. Lozano, P. Castillo, and A. Dzul. Global stabilization of the PVTOL : real-time application to a mini-aircraft. *International Journal of Control*, 77(8) :735–740, 2004. [p. 85]

[LCGD04] R. Lozano, P. Castillo, P. Garcia, and A. Dzul. Robust prediction-based control for unstable delay systems : Application to the yaw control of a mini-helicopter. *Automatica*, 40(3) :603–612, 2004. [p. 189]

[LD07] J. Lobo and J. Dias. Relative pose calibration between visual and inertial sensors. *International Journal of Robotics Research*, 26(6) :561–575, juin 2007. [p. 135]

[Lev44] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *The Quarterly of Applied Mathematics*, 2 :164–168, 1944. [p. 130, 132]

[LF97] Q.T. Luong and O. Faugeras. Self-calibration of a moving camera from point correspondences and fundamental matrices. *The International Journal of Computer Vision*, 22(3) :261–289, 1997. [p. 129]

[LH81] H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293 :133–135, septembre 1981. [p. 146]

[LK81] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Imaging understanding workshop*, 1981. [p. 140, 150]

[LKI+02] S.C. Liu, J. Kramer, G. Indiveri, T. Delbrück, and R. Douglas. *Analog VLSI : Circuits and Principles*. The MIT Press, 2002. [p. 143]

[LMS82] E.J. Lefferts, F.L. Markley, and M.D. Shuster. Kalman filtering for spacecraft attitude estimation. *Journal of Guidance, Control, and Dynamics*, 5(5) :417–429, 1982. [p. 49]

[LP05] P. Lang and A. Pinz. Calibration of hybrid vision / inertial tracking systems. In 2^{nd} *Workshop on Integration of Vision and Inertial Sensors (INERVIS'05)*, Barcelone, Espagne, avril 2005. [p. 135, 136]

[LS93] Z. Lin and A. Saberi. Semi-global exponential stabilization of linear systems subject to input saturation via linear feedbacks. *Systems and Control Letters*, 21 :225–239, 1993. [p. 105]

[Luc84] B.D. Lucas. *Generalized Image Matching by the Method of Differences*. PhD thesis, Carnegie Mellon University, 1984. [p. 140, 150]

[LW05] T. Low and G. Wyeth. Obstacle detection using optical flow. In *Australasian Conference on Robotics and Automation*, Sydney, Australie, 2005. [p. 142]

[Mal00] H.A. Mallot. *Computational Vision : Information Processing in Perception and Visual Behavior*. The MIT Press, 2000. [p. 138]

[Mar63] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11 :431–441, 1963. [p. 130, 132]

[Mar82] D. Marr. *Vision : A Computational Investigation into the Human Representation and Processing of Visual Information*. W. H. Freeman, 1982. [p. 138]

[Mar03a] N. Marchand. Further results on global stabilization for multiple integrators with bounded controls. In *IEEE Conference on Decision and Control*, Maui, Hawai, 2003. [p. 107]

[Mar03b] F.L. Markley. Attitude error representations for kalman filtering. *Journal of Guidance, Control, and Dynamics*, 63(2) :311–317, 2003. [p. 49]

[MBR+09] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.C. Zufferey, D. Floreano, and A. Martinoli. The *e-puck*, a robot designed for education in engineering. In *Conference on Autonomous Robot Systems and Competitions*, 2009. [p. 150, 151]

[MDP96] P. Martin, S. Devasia, and B. Paden. A different look at output tracking : control of a VTOL aircraft. *Automatica*, 32 :101–107, 1996. [p. 84]

[Mel94] T. Melen. *Geometrical modelling and calibration of video cameras for underwater navigation*. PhD thesis, Norges tekniske hϕgskole, Institutt for teknisk kybernetikk, 1994. [p. 127]

[MH05] N. Marchand and A. Hably. Global stabilization of multiple integrators with bounded controls. *Automatica*, 41(12) :2147–215, 2005. [p. 106]

[MI00] L. Marconi and A. Isidori. Robust global stabilization of a class of uncertain feedforward nonlinear systems. *Systems and Control Letters*, 41(4) :281–290, 2000. [p. 105]

[MIS02] L. Marconi, A. Isidori, and A. Serrani. Autonomous vertical landing on an oscillating platform : an internal-model based approach. *Automatica*, 38 :21–32, 2002. [p. 85]

[MMN03] F. Mazenc, S. Mondié, and S.I. Niculescu. Global asymptotic stabilization for chains of integrators with a delay in the input. *IEEE Transactions on Automatic Control*, 2003. [p. 105]

[MO79] A.Z. Manitius and A.W. Olbrot. Finite Spectrum Assignment problem for systems with delays. *IEEE Transactions on Automatic Control*, 24(4) :541–553, 1979. [p. 188]

[MP96] F. Mazenc and L. Praly. Adding integrations, satured controls, and stabilization for feedforward systems. *IEEE Transactions on Automatic Control*, 41, 1996. [p. 84, 105]

[MR01] W. Michiels and D. Roose. Global stabilization of multiple integrators with time-delay and input constraints. In 3^{rd} *IFAC Workshop Time Delay Systems*, Santa Fe, États-Unis, décembre 2001. [p. 105]

[NA89] R.C. Nelson and Y. Aloimonos. Obstacle avoidance using flow field divergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(10) :1102–1106, 1989. [p. 138]

[NCMT08] C. Nielsen, L. Consolini, M. Maggiore, and M. Tosques. Path following for the PVTOL : a set stabilization approach. In *IEEE Conference on Decision and Control*, Cancun, Mexique, décembre 2008. [p. 85]

[NIM97] Department of Defense World Geodetic System 1984, its definition and relationships with local geodetic systems. Technical report, NIMA Technical Report TR8350.2, juillet 1997. [p. 52]

[OS99] R. Olfati-Saber. Global configuration stabilization for the VTOL aircraft with strong input coupling. In *IEEE Conference on Decision and Control*, Phoenix, États-Unis, 1999. [p. 86]

[Pea01] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6) :559–572, 1901. [p. 156]

[PH04] S. Park and J. How. Examples of estimation filters from recent aircraft projects at MIT. Technical report, Massachusetts Institute of Technology, novembre 2004. [p. 49]

[PNG97] A.K. Padhi, K.S. Nanjundaswamy, and J.B. Goodenough. Phospho-olivines as positive-electrode materials for rechargeable lithium batteries. *Journal of The Electrochemical Society*, 144(4) :1188–1194, 1997. [p. 42]

[Rad00] J.C. Radix. Accéléromètres inertiels. *Techniques de l'ingénieur. Mesures et contrôle*, RD2(R1930) :R1930.1–R1930.8, 2000. [p. 43]

[Rak06] T. Rakotomamonjy. *Modélisation et contrôle du vol d'un microdrone à ailes battantes*. PhD thesis, Université Paul Cézanne Aix-Marseille, 2006. [p. 19]

[RBL06] H. Romero, R. Benosman, and R. Lozano. Stabilization and location of a four rotors helicopter applying vision. In *American Control Conference*, Minneapolis, Minnesota, États-Unis, juin 2006. [p. 149]

[RCM+01] C. Rocchini, P. Cignoni, C. Montani, P. Pingi, and R. Scopigno. A low cost 3D scanner based on structured light. *ERCIM News Online Edition*, 44, janvier 2001. [p. 166, 168]

[RFCSS09] E. Rondon, I. Fantoni-Coichot, A. Sanchez, and G. Sanahuja. Optical flow-based controller for reactive and relative navigation dedicated to a four rotor rotorcraft. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Saint Louis, États-Unis, 2009. [p. 142]

[RGGN07] A. Rowe, A. Goode, D. Goel, and I. Nourbakhsh. CMUcam3 : An open programmable embedded vision sensor. Technical report, Robotics Institute, Carnegie Mellon University, 2007. [p. 67]

[RKS06] S. Rathinam, Z. Kim, and R. Sengupta. Vision-based following of structures using an unmanned aerial vehicle. Technical report, Institute of Transportation Studies, University of California at Berkeley, 2006. [p. 7]

[RLG+02] M. Ribo, P. Lang, H. Ganster, M. Brandner, C. Stock, and A. Pinz. Hybrid tracking for outdoor augmented reality applications. *IEEE Computer Graphics and Applications*, 22(6) :54–63, 2002. [p. 134]

[RLP05] P. Rongier, E. Lavarec, and F. Pierrot. Kinematic and dynamic modeling and control of a 3-rotor aircraft. In *IEEE International Conference on Robotics and Automation*, Barcelone, Espagne, 18-22 avril 2005. [p. 11]

[RMO04] T. Rakotomamonjy, T. Le Moing, and M. Ouladsine. Simulation of a flapping-wing Micro Air Vehicle. In *European Micro Aerial Vehicle Conference*, Braunschweig, Allemagne, juillet 2004. [p. 20]

[RMP08] H. Rifaï, N. Marchand, and G. Poulin. *Intelligent Aerial Vehicles*, chapter Attitude and Position Control of a Flapping Micro Aerial Vehicle, pages 553–578. I-Tech, 2008. [p. 19]

[ROM06] T. Rakotomamonjy, M. Ouladsine, and T. Le Moing. Modélisation et commande d'un microdrone à aile battante selon l'axe vertical. In *Conférence Internationale Francophone d'Automatique*, mai 2006. [p. 20]

[ROM07] T. Rakotomamonjy, M. Ouladsine, and T. Le Moing. Modelisation and kinematics optimization for a flapping-wing Micro Air Vehicle. *Journal of aircraft*, 44(1) :217–231, 2007. [p. 20]

[Rom08] H. Romero. *Modélisation et asservissement visuel d'un mini hélicoptère*. PhD thesis, Université de Technologie de Compiègne, 2008. [p. 16]

[Rou08] H. Roudier. *Algèbre linéaire, une introduction*. Vuibert, 2008. [p. 112]

[RRN02] A. Rowe, C. Rosenberg, and I. Nourbakhsh. A low cost embedded color vision system. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Suisse, 2002. [p. 67]

[RSS+07] H. Romero, S. Salazar, A. Sánchez, P. Castillo, and R. Lozano. Modeling and real-time control stabilization of a new VTOL aircraft with eight rotors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, États-Unis, 2007. [p. 15]

[RSSL07] H. Romero, S. Salazar, A. Sánchez, and R. Lozano. A new UAV configuration having eight rotors : Dynamical model and real-time control. In *IEEE Conference on Decision and Control*, La Nouvelle Orléans, États-Unis, 2007. [p. 15]

[SA89] Y. Shiu and S. Ahmad. Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form $AX = XB$. *IEEE Transactions on Robotics and Automation*, 5(1) :16–27, 1989. [p. 134]

[SA05] J. Salt and P. Albertos. Model-based multirate controllers desing. *IEEE Transactions on Control Systems Technology*, 13(6) :988–997, 2005. [p. 189]

[SC01] H. Stone and G. Clarke. Optimization of transition maneuvers for a tail-sitter unmanned air vehicle UAV. In *Australian International Aerospace Congress*, 2001. [p. 17]

[SCE+07] A. Sanchez, P. Castillo, J. Escareno, H. Romero, and R. Lozano. Simple real-time control strategy to stabilize the PVTOL aircraft using

bounded inputs. In *IEEE European Control Conference*, Kos, Grèce, 2007. [p. 90, 106]

[SCGL07] G. Sanahuja, P. Castillo, O. Garcia, and R. Lozano. Linear and nonlinear control strategies to stabilize a VTOL aircraft comparative analysis. In *IFAC Symposium on Intelligent Autonomous Vehicles*, Toulouse, France, septembre 2007. [p. 83]

[SCS] G. Sanahuja, P. Castillo, and A. Sanchez. Stabilization of n integrators in cascade with bounded input with experimental application to a VTOL laboratory system. *International Journal of Robust and Nonlinear Control*. [p. 106]

[SGCA08] G. Sanahuja, P. Garcia, P. Castillo, and P. Albertos. Control of unstable delayed systems with input saturations and measurement constraints : An electrical cart application. In *International Federation of Automatic Control World Congress*, Séoul, Corée, 2008. [p. 188]

[SGCL08] A. Sanchez, P. Garcia, P. Castillo, and R. Lozano. Simple real-time stabilization of a VTOL aircraft with bounded signals. *AIAA Journal of Guidance, Control and Dynamics*, 31(4), 2008. [p. 106]

[SJK97] R. Sepulchre, M. Janković, and P. Kokotović. *Constructive Nonlinear Control*. Springer-Verlag London, 1997. [p. 91]

[SKS90] A. Saberi, P.V. Kokotović, and H.J. Sussmann. Global stabilization of partially linear composite systems. *SIAM Journal on Control and Optimization*, 28 :1491–1503, 1990. [p. 91]

[SL05] S. Salazar and R. Lozano. Stabilization and nonlinear control for a novel trirotor mini-aircraft. In *IEEE International Conference on Robotics and Automation*, Barcelone, Espagne, 18-22 avril 2005. [p. 11]

[Smi59] O.J.M. Smith. Closer control of loops with dead time. *Chemical Engineering Progress*, 53 :217–219, 1959. [p. 188]

[SMS03] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme. Visually guided landing of an unmanned aerial vehicle. *IEEE Transcations on Robotics and Automation*, 19(3) :371–381, 2003. [p. 10, 11]

[SPA07] R. Sanchis, I. Peñarrocha, and P. Albertos. Desing of robust output predictors under scarce measurements with time-varying delays. *Automatica*, 43(2) :281–289, 2007. [p. 189]

[Sri94] M.V. Srinivasan. An image-interpolation technique for the computation of optic flow and egomotion. *Biological Cybernetics*, 71 :401–416, 1994. [p. 152]

[SRL08] S. Salazar, H. Romero, and R. Lozano. Real-time stabilization of an eight-rotor UAV using optical flow. In *Workshop on Visual guidance*

systems for small autonomous aerial vehicles, IROS, Nice, France, septembre 2008. [p. 16]

[SSK+06] O. Seungyong, K. Sungchul, L. Kyungjoon, A. Sangchul, and K. Euntai. Flying display : Autonomous blimp with real-time visual tracking and image projection. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, Chine, 9-15 octobre 2006. [p. 18, 19]

[SSY94] H.J. Sussmann, E.D. Sontag, and Y. Yang. A general result on the stabilization of linear systems using bounded controls. *IEEE Transactions on Automatic Control*, 39 :2411–2425, 1994. [p. 105]

[Sto06] A. Stocker. Analog integrated 2-D optical flow sensor. *Analog Integrated Circuits and Signal Processing*, 46(2) :121–138, février 2006. [p. 143]

[SVS95] J. Santos-Victor and G. Sandini. Divergent stereo in autonomous navigation : From bees to robots. *International Journal of Computer Vision*, 14(2) :159–177, 1995. [p. 142]

[SZC+00] M.V. Srinivasan, S.W. Zhang, J.S. Chahl, E. Barth, and S. Venkatesh. How honeybees make grazing landings on flat surfaces. *Biological Cybernetics*, 83(3) :171–183, août 2000. [p. 141]

[Tee92a] A.R. Teel. Global stabilization and restricted tracking for multiple integrators with bounded controls. *Systems and Control Letters*, 18 :165–171, 1992. [p. 84, 90, 105]

[Tee92b] A.R. Teel. Semi-global stabilization of minimum phase nonlinear systems in special normal forms. *Systems and Control Letters*, 19 :187–192, 1992. [p. 105]

[Tee93] A.R. Teel. Semi-global stabilization of the ball and beam using output feedback. In *American Control Conference*, San Francisco, États-Unis, 1993. [p. 105]

[TL89] R. Tsai and R. Lenz. A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. *IEEE Transactions on Robotics and Automation*, 5(2) :345–358, 1989. [p. 134]

[tpltea] *Firmware open source* pour le *Xufo* (en allemand). "`http://forum.xufo.net/bb/viewtopic.php?t=12161`". [p. 12]

[Tsa87] R.Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, 3(4) :323–344, août 1987. [p. 129]

[VBMP08] D. Vissière, P.J. Bristeau, A.P. Martin, and N. Petit. Experimental autonomous flight of a small-scaled helicopter using accurate dynamics model and low-cost sensors. In *International Federation of Automatic Control IFAC World Congress*, Séoul, Corée, 2008. [p. 49]

[VJ04] P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2) :137–154, 2004. [p. 67]

[VMP07a] D. Vissière, A. Martin, , and N. Petit. Using magnetic disturbances to improve IMU-based position estimation. In *IEEE European Control Conference*, Kos, Grèce, 2007. [p. 47, 50]

[VMP07b] D. Vissière, A. Martin, and N. Petit. Using spatially distributed magnetometers to increase IMU-based velocity estimation in perturbed areas. In *IEEE Conference on Decision and Control*, La Nouvelle Orléans, États-Unis, 2007. [p. 50]

[WC07] R. Wood and B. Cazzolato. An alternative nonlinear control law for the global stabilization of the PVTOL vehicle. *IEEE Transactions on Automatic Control*, 52 :1282–1287, 2007. [p. 85]

[WCH92] J. Weng, P. Cohen, and M. Herniou. Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10) :965–980, octobre 1992. [p. 126]

[WM94] G. Wei and S. Ma. Implicit and explicit camera calibration : Theory and experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5) :469–480, 1994. [p. 126, 127]

[XZ96] G. Xu and Z. Zhang. *Epipolar Geometry in Stereo, Motion and Object Recognition*. Kluwer Academic Publishers, 1996. [p. 145]

[YNSC07] Z. Yu, K. Nonami, J. Shin, and D. Celestino. 3D vision based landing control of a small scale autonomous helicopter. *International Journal of Advanced Robotic Systems*, 4(1) :51–56, 2007. [p. 10]

[YWW07] H. Ye, H. Wang, and H. Wang. Stabilization of a PVTOL aircraft and a inertia wheel pendulum using saturation technique. *IEEE Transactions on Control Systems Technology*, 15 :1143–1150, 2007. [p. 85]

[ZF05] J.C. Zufferey and D. Floreano. Toward 30-gram Autonomous Indoor Aircraft : Vision-based Obstacle Avoidance and Altitude Control. In *IEEE International Conference on Robotics and Automation*, Barcelone, Espagne, 2005. [p. 142]

[ZFL02] A. Zavala, I. Fantoni, and R. Lozano. Global stabilization of a PVTOL aircraft with bounded inputs. In *IFAC Latin-American Conference on Automatic Control CLCA*, Guadalajara, Mexique, December 2002. [p. 85]

[ZFL03] A. Zavala, I. Fantoni, and R. Lozano. Global stabilization of a PVTOL aircraft with bounded inputs. *International Journal of Control*, 76(18) :1833–1844, 2003. [p. 85]

[Zha99] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *International Conference on Computer Vision*, Kerkyra, Grèce, septembre 1999. [p. 129, 130, 132]

[ZKB+07] J.C. Zufferey, A. Klaptocz, A. Beyeler, J.D. Nicoud, and D. Floreano. A 10-gram vision-based flying robot. *Advanced Robotics*, 21(14) :1671–1684, 2007. Special issue on IROS 2006. [p. 6, 7, 142, 151]

[Zuf05] J.C. Zufferey. *Bio-inspired vision-based flying robots*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2005. [p. 139, 152]