SARAH DELMAS

RAPPORT DE PROJET DE FIN D'ETUDES

VISION SPHÉRIQUE POUR LA ROBOTIQUE AÉRIENNE

22 août 2019

ENCADRANTS : M. CARON, M. PEGARD

TUTEUR : M. ZERR

LABORATOIRE MIS, UNIVERSITÉ DE PICARDIE JULES VERNE, AMIENS

Table des matières

Résumé

Contrôler un système par asservissement visuel, c'est utiliser des informations données par un, ou des, capteurs de vision pour contrôler les déplacements du système. La principale problématique de cet asservissement est la sélection des données nécessaires et utiles à la commande voulue, parmi toutes les données reçues. De plus, pour obtenir un asservissement fonctionnel, il convient d'avoir cette sélection et le traitement des données en temps réel. Dans ce but, les objectifs du stage étaient de réaliser un algorithme d'acquisition de données de localisation -visuelles et géolocalisées- et de données inertielles en temps réel, puis de réaliser un asservissement utilisant les données acquises.

A un mois de la fin du stage, l'acquisition des données en temps réel est fonctionnelle et a été testée et vérifiée lors de plusieurs missions en extérieur. Un algorithme de redressement des images et de sélection de points d'une mire sur celles-ci a également été réalisé, ainsi qu'un début d'asservissement, qui calcule les angles et vitesses nécessaires pour que le drone se positionne au-dessus de la mire. Ces algorithmes ont été testés et, dans le mois restant, il reste à les améliorer, et à mettre en place le contrôle du drone.

Abstract

Visual servoing is about using information given by one, or more, vision sensors to control the movements of a system. The main issue of this servocontrol is the selection of the necessary and useful data to accomplish the desired command, among all the data received by the sensors. Furthermore, to obtain a functional servocontrol, it is necessary to have this selection and the data processing in real time. For this purpose, the objectives of the internship were to realize an algorithm for acquiring location-based -visual and geolocated- and inertial data in real time, then to produce a servocontrol using the acquired data.

At one month from the end of the internship, real-time data acquisition is functional and has been tested and verified during several outdoor missions. An algorithm for straightening the acquired images and for selecting the points of a test target was also realized, as well as the beginning of a servocontrol, which calculates the angles and speeds necessary, so that the drone will position itself above of the target. These algorithms have been tested and, in the last month, it remains to improve them, and to set up the control of the drone.

Mots clés

Robotique, drones, vision omnidirectionnelle, asservissement visuel, redressement d'images par centrale inertielle, rapport de stage, rapport PFE.

Robotic, drones, omnidirectional vision, visual servoing, image straightening by IMU, internship report, end of studies report.

Remerciements

Je remercie particulièrement :
— Guillaume Caron et Claude Pégard pour leur aide et leur encadrement lors de ce projet, ainsi que leur accueil chaleureux,
— Hervé Midavaine pour tous ses travaux sur la connectique du drone et sa création des supports pour l'ordinateur embarqué et des caméras,
— Eder Alejandro Rodriguez Martinez pour son aide sur l'homographie.

Mais aussi à tous les membres du MIS pour leur camaraderie.

Ce travail est financé dans le cadre du projet INTERREG VA FMA ADAPT "Assistive Devices for empowering disAbled People through robotic Technologies", adapt-project.com. Le programme FMA est un programme de Coopération Territoriale Européenne qui vise à financer des projets de coopération ambitieux dans la région frontalière entre la France et l'Angleterre. Le programme est financé par le Fonds Européen de Développement Régionale (FEDER).

Définitions et acronymes

Asservissement visuel : Commande d'un robot, à partir d'informations extraites d'images acquises par ce robot. *retour au texte*

GPS : Global Positioning System. Système de localisation par satellites. *retour au texte*

SDK : Software Development Kit ou kit de développement logiciel. Un ensemble d'outils utiles pour le développement de logiciels, programmes ou applications pour une plateforme particulière. *retour au texte*

ROS : Robot Operationg System, le principe est présenté en Annexe A. *retour au texte*

Cardan : Système de suspension donnant une position invariable malgré les mouvements. *retour au texte*

IMU : Inertial Measurement Unit ou centrale inertielle. Instrument mesurant l'assiette et les vitesses angulaires.*retour au texte*

Heure/temps UNIX : C'est le temps en secondes écoulé depuis le premier Janvier 1970 à minuit. *retour au texte*

HDMI : High Definition Multimedia Interface. Une norme d'interface audio/vidéo, généralement utilisée pour les écrans. *retour au texte*

SSH : Secure SHell. Un protocole de communication sécurisé entre machines distantes et un programme informatique. *retour au texte*

Gazebo : Un logiciel de simulation, compatible avec ROS, prenant un compte les lois physiques, telle la gravité ou les frottements. *retour au texte*

Intergiciel : ou middleware en anglais. Un logiciel créant un réseau de communication entre différentes applications informatiques. *retour au texte*

Introduction

Les drones sont beaucoup utilisés pour la cartographie, l'évaluation de zones sensibles et les missions de sauvetages. Ils sont également souvent dépendants de l'humain pour la navigation. Or, ceci n'est pas toujours possible ou efficace. Il existe plusieurs méthodes pour se passer de ce contrôle humain, dont celles qui sont basées sur la perception de l'environnement.

Ce rapport présente les travaux effectués lors de mon stage au laboratoire du MIS à Amiens. L'équipe Perception Robotique travaille sur l'asservissement visuel de drones et a fait l'acquisition récente d'un héxacoptère. Le but du stage était de trouver un moyen de réaliser des acquisitions de données géolocalisées, inertielles, et images en interfaçant le drone, une caméra omnidirectionnelle et un ordinateur embarqué, puis de faire un asservissement visuel, afin de créer une base de travail exploitable pour valider des travaux de recherches en navigation autonome basée vision. Sauf s'il est précisé autrement, tous les algorithmes mentionnés ont été réalisé en C++.

Le drone fourni pour le stage est un modèle qui n'avait pas été pris en main par le MIS auparavant, n'étant pas un drone communément utilisé pour la recherche. Mes travaux ont donc nécessité une partie importante de recherche sur la programmation de ce drone.

Le contexte du stage ainsi que le laboratoire sont présentés dans la première partie. La deuxième section décrit les drones et caméras fournis par le laboratoire pour mettre en œuvre ce projet. Ensuite, la section trois présente la connectique et le système d'assemblage de l'ordinateur et des caméras sur le drone. En quatrième, les travaux d'acquisitions sont décrits, avec en cinquième partie les descriptions des tests en vol extérieur. Finalement, la chaîne de traitements vers un asservissement visuel omnidirectionnel de drone est présentée, ainsi que ces résultats préliminaires.

Chapitre 1

Le contexte du stage

J'ai réalisé mon stage au MIS, un laboratoire de l'Université de Picardie Jules Verne à Amiens. Ce stage a commencé le 18 Avril et finira le 27 Septembre. Ci-dessous sont présentés succinctement le laboratoire et ses équipes, ainsi que les objectifs du stage.

1.1 Le laboratoire MIS

Le MIS, pour Modélisation, Information et Systèmes, se spécialise sur la recherche en informatique, automatique, robotique, et vision par ordinateur. En 2019, son effectif est d'environ 80 membres, qui sont pour la moitié d'entre eux des enseignants chercheurs. Le reste de l'effectif est, en grande partie, composé de doctorants avec, lors de mon stage, la présence d'une dizaine de stagiaires, allant de la licence au master.

Les membres sont répartis en quatre équipes :

— L'équipe COVE (COmmande et VEhicules) : élaboration de stratégie de contrôle et de diagnostic de systèmes,
— L'équipe GOC (Graphes, Optimisation et Contraintes) : modélisation de problèmes et d'informations et développement de système d'aide à la décision,
— L'équipe SDMA (Systèmes Distribués, Mots et Applications) : étude de modèles de parallélisme, d'algorithmique et analyse d'algorithmes,
— L'équipe PR (Perception Robotique) : étude de perception visuelle de robots

Plus d'informations sur les travaux passés et courants sont disponibles sur le site du MIS [1]. Ayant fait partie de l'équipe PR, celle-ci va être présentée avec plus de détails ci-dessous.

1.2 L'équipe Perception Robotique

L'équipe PR, dirigée par M. Caron, un de mes encadrants, se spécialise sur l'amélioration de l'autonomie d'engins mobiles par la perception artificielle. Plus précisément, leur recherche passe

par la conception de capteurs originaux, par exemple de vision omnidirectionnelle monoculaire, et le développement de méthodes pour les exploiter.

L'équipe est sous divisée en deux groupes : l'un qui travaille plus sur la vision omnidirectionnelle et son optimisation, l'autre sur la localisation et la navigation de robots, principalement au moyen de capteurs visuels omnidirectionnels. C'est dans ce deuxième groupe que mon stage se situe.

1.3 Les objectifs initiaux du stage

Le MIS possède depuis peu deux drones DJI ; les DJI M200 et M600, présentés dans la partie suivante de ce document. De ces deux drones, seul le M200 avait été testé en vol avant le début du stage.

Comme précédemment explicité, une grande partie des travaux de l'équipe PR est basée sur l'asservissement visuel (définition en page 7) de robots. L'objectif de mon stage était donc de mettre en place en système d'acquisition de données visuelles, inertielles, et de localisation, puis de mettre en place un asservissement visuel.

A plus long terme, le MIS souhaite explorer un asservissement autonome par chemin visuel ; l'un des doctorants du MIS, Eder Rodriguez, travaille en ce moment sur cette possibilité. Ses travaux pourront bénéficier des miens, notamment du point de vue de la validation expérimentale.

Au début du stage, le premier drone à devoir être interfacé était le DJI M200, un quadricoptère. Ce modèle de drone n'était, au final, pas interfaçable et c'est donc uniquement le M600 Pro qui a été utilisé. Ceci est expliqué plus en détail dans la section suivante, qui décrit le matériel utilisé lors du stage.

Chapitre 2

Le matériel fourni

Cette section décrit les drones et les caméras mis à ma disposition lors du stage. L'ordinateur embarqué était un DELL OptiPlex 7040 Micro.

2.1 Les drones fournis par le MIS

Pour réaliser les objectifs du stage, deux modèles de drones DJI étaient disponibles : le M200 et le M600 Pro. Ceux sont respectivement un quadricoptère et un héxacoptère.

2.1.1 L'entreprise DJI

DJI, [2], est une entreprise chinoise qui se présente comme le leader international des drones civils et de la vision aérienne. Leurs drones sont principalement construits pour la surveillance de l'état des champs agricoles, l'inspection de structures produisant ou transportant de l'énergie -telles les centrales nucléaires ou les lignes électriques-, le suivi de site de construction et la surveillance d'accidents sur des infrastructures, l'aide visuelle lors de catastrophes, et, enfin, la surveillance, et la sécurité, des fonctionnaires d'État.

2.1.2 Les modèles de drones M200 et M600 Pro

Le M200 est un quadricoptère de 88 cm de côté, fait pour pouvoir être utilisé dans des conditions difficiles. Il est en effet capable de rester stable même face à des vents allant jusqu'à 35km/h, avec au maximum 50cm de variation verticale et 1m50 de variation horizontale. De plus, il a été conçu pour résister à la poussière et à la pluie. Son altitude de vol légale maximale est de 110m.

FIGURE 2.1 – DJI M200 (à gauche), DJI M600 Pro (à droite)

Le M600 Pro, lui, est un héxacoptère faisant 1m60 par 1m50, avec une altitude de vol maximale de 120m. Sa capacité à se stabiliser en conditions difficiles semble légèrement plus faible que celle du M200 : sa résistance maximale au vent étant de 8m/s au lieu des 12m/s du M200, mais il est également étanche. Différents modules visuels peuvent également lui être ajoutés.

Le M200 est capable de porter jusqu'à 2,374Kg de charge utile, et le M600 Pro, jusqu'à 6Kg. C'est pour cette raison que le MIS a acheté ces drones : ils peuvent transporter sans problème un ordinateur et une caméra.

Les deux drones ont une autonomie de vol d'une trentaine de minutes. Ils disposent de trains d'atterrissage, les « pieds » sur lesquels reposent les drones dans la figure 2.1. Ceux du M600 Pro se replient automatiquement quand le drone s'envole. De plus, ils ont tous deux au moins un GPS (définition en page 7) ; le M600 Pro en a trois pour la redondance.

Le M200 a rapidement été remplacé par le modèle M210, et il y a donc peu de documentation disponible sur ce modèle. Le M210, comme le M600, possède un port permettant la connexion d'un ordinateur embarqué au drone. Initialement, le laboratoire pensait que c'était aussi le cas du M200, mais cette hypothèse s'est révélée fausse. Ceci a conduit à l'abandon de l'utilisation du M200 lors du stage.

2.1.3 Les logiciels disponibles

DJI a en ligne 4 SDKs (définition en page 7) téléchargeables sur leur site [3] :
— Mobile : pour développer des applications mobiles,
— UX : apportant des interfaces utilisateur pour le SDK Mobile,
— Onboard : pour l'interfaçage avec un ordinateur embarqué,
— Payload : pour intégrer directement sur le drone des outils particuliers.

De ces SDKs, seul le Payload est payant. Il est réservé aux entrepreneurs souhaitant développer à des fins de production et de vente. Pour nous, le SDK Onboard était le plus intéressant. Sa plateforme de développement est Linux, accompagné de ROS (définition en page 7 et principe en Annexe A). Plus exactement, le SDK Onboard ne fonctionne qu'avec la version Kinetic Kame de ROS, et cette version est disponible sous Ubuntu 16,04, mais pas 18.04, voir [4].

En plus des SDKs, DJI met à disposition un assistant, le DJI Assistant 2, sous Windows et MAC. Celui-ci permet non seulement d'obtenir l'accès initial au drone, mais également d'avoir un retour, ou une commande, sur certain de ses capteurs. Pour nous, le plus utile a été le suivi de charge des batteries. Il existe également un logiciel de simulation de vol de drone sous Windows 10.

2.2 Les caméras

Pour ce qui est de l'acquisition d'images, il y avait deux caméras disponibles : la Ricoh Theta S et la uEye de IDS, aussi appelée UI, modèle 1245_LE.

FIGURE 2.2 – Ricoh Theta S, uEye, et uEye + objectif Fisheye

La Ricoh Theta S et la uEye, avec son objectif, sont toutes les deux des caméras Fisheye. C'est à dire que leurs objectifs ont un champ de vision de 180 degrés. La Ricoh Theta S ayant deux objectifs, visibles sur la figure 2.2, elle a donc un champ de vision de 360 degrés.

La caméra Ricoh Theta S étant reconnue comme webcam USB par l'ordinateur, son flux vidéo peut être obtenu par la librairie open-CV et elle ne nécessite donc pas de nœud d'acquisition particulier. Par contre, elle possède une latence de transmission qui a été mesurée par M. Caron lors de tests de synchronisations des données.

La caméra uEye, elle, nécessite l'utilisation d'un nœud "ueye_cam", disponible librement sous le site de ROS. Par contre, elle n'a pas de latence. C'est pour cette raison qu'elle est préférée pour l'asservissement en temps réel. Toutefois, le modèle fourni était monochromatique.

Une grande partie des missions ont été faites avec la Ricoh Theta, dû à son champ de vision de 360 degrés et son rendu couleur.

Chapitre 3

La connectique et les supports

Les installations électroniques et les supports pour l'ordinateur et les caméras ont été faites par M. Hervé Midavaine, ingénieur d'étude et responsable de l'électronique du laboratoire.

3.1 La connectique

Comme mentionné précédemment, le M600 Pro a un port série qui permet à un ordinateur embarqué de recevoir les informations des nœuds ROS du drone. Ce port série se trouve sous le capot du drone et les détails des ports sont disponibles sous le site Developper DJI, [5].

L'ordinateur embarqué doit être branché sur les batteries du drone afin de pouvoir fonctionner. Il existe deux ports de connexion disponibles sur le drone, tous les deux en 18 Volts 3 Ampères. Les deux ports se trouvent en dessous du drone, mais l'un est câblé au cardan (définition en page 7), au cas où une caméra y serait attachée. C'est ce câble que M. Midavaine a sectionné pour créer un système d'alimentation alternatif dans l'ordinateur embarqué. L'ordinateur peut alors être branché directement sur le port d'alimentation.

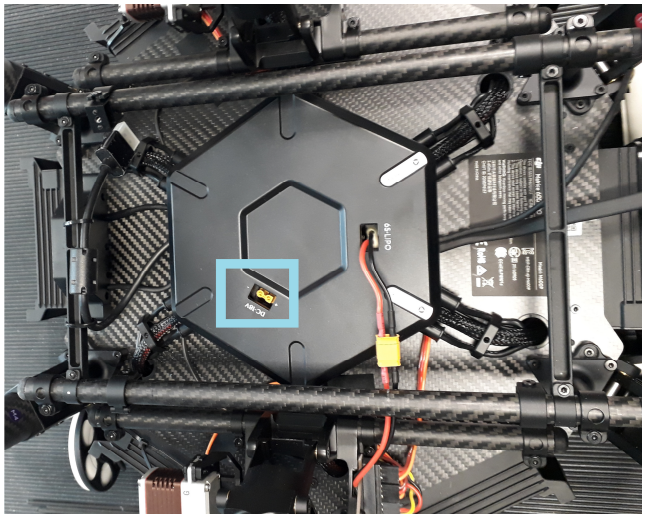

FIGURE 3.1 – Dessous du drone et port d'alimentation de l'ordinateur (encadré en bleu)

3.2 Les supports

Par soucis de facilité et de montage, M. Midavaine a choisi de fixer l'ordinateur au drone par une plaque pouvant être vissée sur le support du cardan.

Pour ce qui est du support des caméras, celui-ci est également vissable, mais cette fois ci sur le support de l'ordinateur.

FIGURE 3.2 – Photos des supports caméras, (Ricoh (gauche), uEye (droite)

FIGURE 3.3 – Photo du support monté sur drone, avec Ricoh Theta S

Chapitre 4

L'acquisition des données

Une fois le système mis en place, il fallait récupérer la position GPS du drone, son assiette et son cap, ainsi que le flux vidéo de la caméra embarquée pour obtenir les données nécessaires à l'asservissement. Excepté le flux vidéo, ces données sont envoyées par les nœuds ROS du drone.

4.1 Les nœuds ROS du drone

Le système ROS en place sur le M600 émet sur plusieurs topics, dont un pour le GPS et un pour la centrale inertielle, appelée IMU (définition en page 7) par la suite. Récupérer ces données ne nécessite donc que la création d'un autre nœud écoutant sur ces topics.

La composition des messages sur les topics est la suivante :

IMU	**GPS**
header : l'horodatage et le nom du topic	header
orientation : les angles de l'IMU en quaternions	statut
covariance de l'orientation	latitude
vitesse angulaire	longitude
covariance de la vitesse angulaire	altitude
accélération linéaire	covariance de la position
covariance de l'accélération linéaire	type de covariance de position

TABLE 4.1 – Composition des messages IMU et GPS

Pour le projet, les informations nécessaires sont la latitude, la longitude, et l'altitude du GPS ainsi que les angles de l'IMU. De plus, l'horodatage des messages permet de vérifier la synchronisation des données en post-traitement.

4.2 L'acquisition

4.2.1 La fréquence d'acquisition

En premier lieu, il a été nécessaire de vérifier les délais d'acquisition et de stockage des données. Pour cela, les temps UNIX (définition en page 7), avant l'acquisition des données et ceux après le stockage, ont été comparés sur plusieurs minutes.

Type de donnée	**Délais**
IMU	11 à 47 us
GPS	4 à 15 us
image Ricoh Theta S	33 à 75ms
image uEye	200 à 750 us

TABLE 4.2 – Tableau des délais de stockages

La plus haute fréquence possible pour nos acquisitions était donc de $1/0,075s = 13Hz$. Il a paru prudent de descendre à une fréquence de 10Hz.

4.2.2 Schéma du programme d'acquisition

Algorithm 1: Schéma du programme d'acquisition

```
Result: Acquisition des données
écoute en continu les nœuds GPS et IMU;
crée un dossier de sauvegarde "mois_jours_heure :minutes";
crée les fichiers textes pour enregistrer les données GPS, IMU, et horodatage des images;
écrit dans les fichiers textes une ligne descriptive;
mission lancée = false;
période d'acquisition = false;
ancienne IMU = 0;
ancienne altitude = 0;
ancien horodatage = 0;
while ROS do
    if décollage du drone then
        période d'acquisition = true;
        mission lancée = true;
    end
    if mission lancée then
        if période d'acquisition then
            if IMU ou altitude suffisamment différentes d'avant ou délai entre acquisitions
              suffisamment grand then
                stocke GPS, IMU, image, horodatage d'image dans leur liste;
                ancienne IMU = IMU;
                ancienne altitude = altitude;
                ancien horodatage = horodatage;
            end
            if taille de liste images > 4500 then
                période d'acquisition = false;
            end
        end
        if période d'acquisition = false then
            enregistre les données des listes dans fichiers textes;
            enregistre les images;
            vide les listes;
            période d'acquisition = true;
        end
    end
end
enregistre les donnée restantes des listes;
vide les listes;
```

4.3 Explications du code et sauvegarde des données

Le choix de stocker les données dans des listes, au lieu de les enregistrer immédiatement est dû aux temps d'enregistrements nécessaires. En effet, les délais d'enregistrement, en particulier ceux des images, causent une pause de plusieurs dizaines de millisecondes. A notre fréquence d'acquisition, une toutes les 100 millisecondes, ceci peut provoquer un décalage temporel important. Pour obtenir des données synchronisées temporellement, il est donc plus efficace de les stocker sur une période et de ne les enregistrer qu'une fois celle-ci finie.

Ici, nous avons choisi d'effectuer des missions d'au maximum 7 minutes 30 secondes. Avec la fréquence à 10Hz, cela donne 4500 jeux d'acquisitions pour une mission. L'acquisition prend fin une fois les 4500 jeux de données enregistrés, avec un temps d'environ 45 secondes pour l'enregistrement en lui-même. Au cas où le programme serait interrompu manuellement avant d'atteindre les 4500 jeux, les données restantes dans les listes sont enregistrées avant la fermeture du nœud.

Pour éviter l'amassement des données inutiles, l'acquisition ne se lance que quand le drone s'envole pour la première fois ; plus précisément, quand il subit un changement de vitesse verticale de plus d'un 1.5 m/s. Ce chiffre est basé sur les données à l'arrêt de l'IMU : si les vitesses sur les axes x et y du drone sont toujours proches de 0, la vitesse en z, elle, oscille entre 9 et 10,2 m/s. Il est fort possible qu'elle prenne en compte la gravité, mais la raison exacte des variations n'est mentionnée nulle part. De même, il y a peu d'intérêt à enregistrer des données quand le drone est stabilisé. Les données ne sont mises dans leur liste que si l'assiette du drone a été modifié d'au moins 2 degrés, si l'altitude du drone a changé de 10cm ou s'il s'est écoulé plus de 0.5 secondes depuis la dernière acquisition.

En prenant en compte un asservissement en temps réel, l'envoi d'un message de stabilisation du drone lors de l'enregistrement a été considéré, mais n'est encore pour l'instant que théorique. Les 7 minutes 30 suffisant amplement aux missions prévues, garder le drone immobile lors de l'enregistrement pour que, quand l'acquisition reprenne, il n'y ai pas de sauts entre les données est pour l'instant inutile.

Chapitre 5

Les tests d'acquisition

Les programmes d'acquisitions des données ont été testés en premier lieu au MIS, soit en intérieur pour les images et l'IMU, soit en extérieur, pour le GPS. Cependant, il était également nécessaire de tester les algorithmes en conditions de vol. Pour cela, nous avons fait plusieurs tests sur le terrain.

5.1 Communication sur le terrain

Avant de lancer le programme d'acquisition, il faut en premier lieu démarrer le nœud ROS du drone. Celui-ci étant difficile à connecter, il est laissé actif en arrière-plan.

Si le début de l'acquisition est déclenché par l'envol du drone, la fin, elle, est déclenchée par une interruption clavier. Pour pouvoir lancer et arrêter l'acquisition, il faut donc avoir accès au terminal sur lequel sont lancés les nœuds. Nous avons pour cela utilisé une tablette Flysight comme écran, connecté en HDMI (définition en page 7) à l'ordinateur.

La tablette fonctionnait correctement si elle était connectée avant que l'ordinateur ne soit démarré. Dans le cas contraire, si l'on reconnectait la tablette alors que l'ordinateur fonctionnait, il pouvait y avoir plusieurs minutes de latence pour obtenir un retour écran. Pour contourner ce problème, une connexion SSH (définition en page 7) a été mise en place. Cela nous permettait de pouvoir arrêter et relancer le programme d'acquisition beaucoup plus rapidement.

FIGURE 5.1 – Photo de mission

5.2 Les missions

Afin de tester les programmes, nous avons effectué plusieurs missions de vol avec le M600. Pour des raisons légales et de sécurité, ces vols se sont fait dans les champs près de Vaux-en-Amiénois, un village au nord-ouest d'Amiens.

Les missions ont généralement duré entre 4 et 6 minutes, avec quelques milliers de mètres de vol. Afin de tester les programmes au maximum, les missions ont toutes comprises des changements d'angles radicaux et des variations de hauteurs. Les points de départ et d'arrivée étaient à l'intersection de trois routes, afin de simplifier l'atterrissage du drone et de ne pas dégrader les champs. Le tracé des données GPS d'une des missions est illustré sur la figure 5.2.

FIGURE 5.2 – Tracé GPS d'un vol

Un problème récurrent rencontré lors des missions est que le câble connectant le drone vers l'ordinateur se déconnectait parfois de son port USB du fait du vent. Dans ce cas, les données GPS et IMU n'étaient plus transmises à l'ordinateur, qui enregistrait donc en boucle les dernières données reçues. C'est un problème qui c'est également produit avec le câble de la caméra, mais le câble étant plus court, ce n'est arrivé qu'une seule fois sur les 9 missions. C'est la raison pour laquelle, dans le tracé de la figure 5.2, le drone ne retourne pas à son point de départ. L'acquisition des données GPS s'est arrêtée avant la fin de la mission.

5.3 Tests de synchronisation des données

Une fois les premières missions effectuées, M. Caron a synchronisé l'affichage des données IMU, GPS et images sous Matlab, pour vérifier la concordance temporelle. Il est apparu que les données images avaient un retard de 2 images par rapport aux données IMU ; une variation angulaire du drone en t n'apparaissait qu'à la $t + 2$ image. Ceci est vraisemblablement dû à la latence de transmission de la Ricoh Theta S, utilisée pour la grande majorité des missions d'acquisition.

Les données GPS, bien que sensées pourvoir être reçues jusqu'à 100Hz d'après l'Assistant DJI 2, n'ont en réalité pas cette fréquence d'émission. De plus, comme illustré dans la figure 5.3, la fréquence du GPS n'est pas fixe. Dans cette figure, les données identiques ont été séparées par des cadres de couleurs pour rendre plus visible la répétition des données. Le premier chiffre est l'heure UNIX de l'émission de la donnée, les suivants sont, respectivement, la latitude, la longitude et l'altitude. Des tests ont montré qu'au minimum la fréquence du GPS était de 1Hz, c'est à dire d'une 1 donnée par seconde.

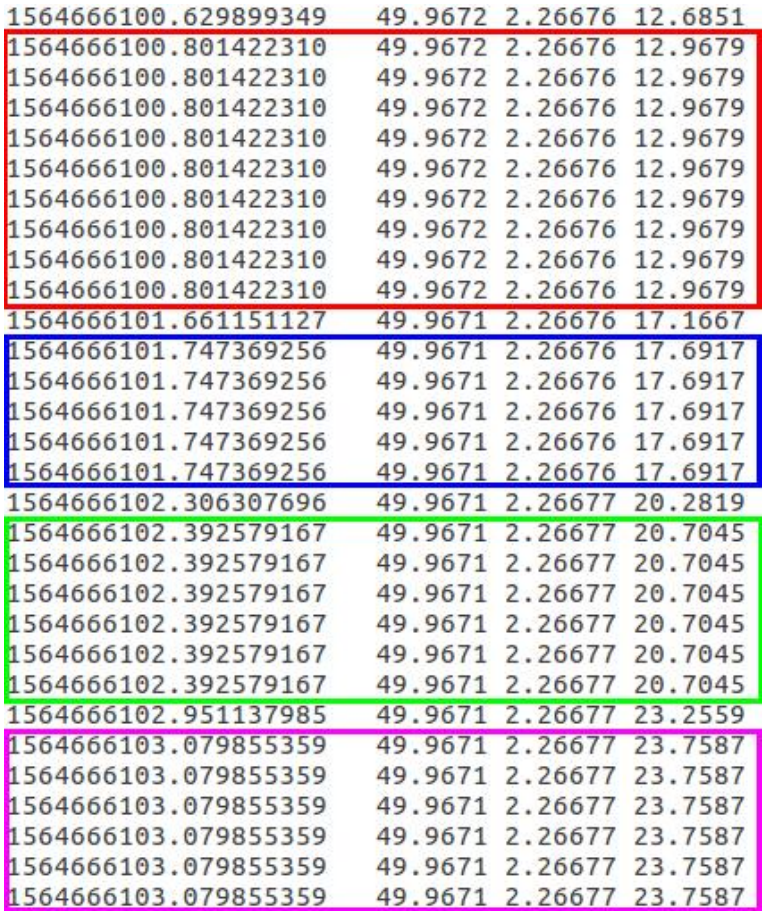

FIGURE 5.3 – Données GPS identiques

Lors du traçage du chemin du drone, voir figure 5.2, nous avons donc des sauts entre deux positions, parfois de plusieurs mètres, dus à ces problèmes de fréquence. Étrangement, l'application liée à la manette du drone, qui trace elle aussi le parcours du drone, n'a pas ce problème de fréquence. Il est possible que la fréquence du GPS soit bridée sous ROS, ou qu'il faille modifier les codes de DJI pour obtenir une fréquence correcte.

Chapitre 6

L'asservissement visuel

Le programme d'asservissement visuel a été développé en plusieurs étapes. En premier, pour des raisons de simulation, c'est un asservissement avec des photos perspectives. Puis, l'asservissement Fisheye a été mis en place.

L'asservissement est fonctionnellement divisé en deux parties. La première, le redressement, corrige l'image en fonction des angles de l'IMU. La deuxième est l'asservissement en tant que tel.

6.1 Redressement d'images perspectives

6.1.1 Le principe d'homographie

Sur le drone à l'arrêt, la caméra perspective pointe vers le sol. Au contraire, lors d'un vol, l'orientation de la caméra dépendra de l'assiette du drone. La mire que l'on veut repérer étant plane sur le sol, il y a peu d'intérêt à ce que l'orientation de la caméra suive celle du drone. Pour compenser cela, il faut redresser l'image en fonction des angles donnés par l'IMU.

Pour cela, on utilise l'homographie. C'est une application mathématique entre deux espaces, qui préserve la structure d'un objet lors du passage d'un plan à un autre. Les informations suivantes viennent d'un cours de Gerhard Roth de l'Université de Carleton au Canada, [6].

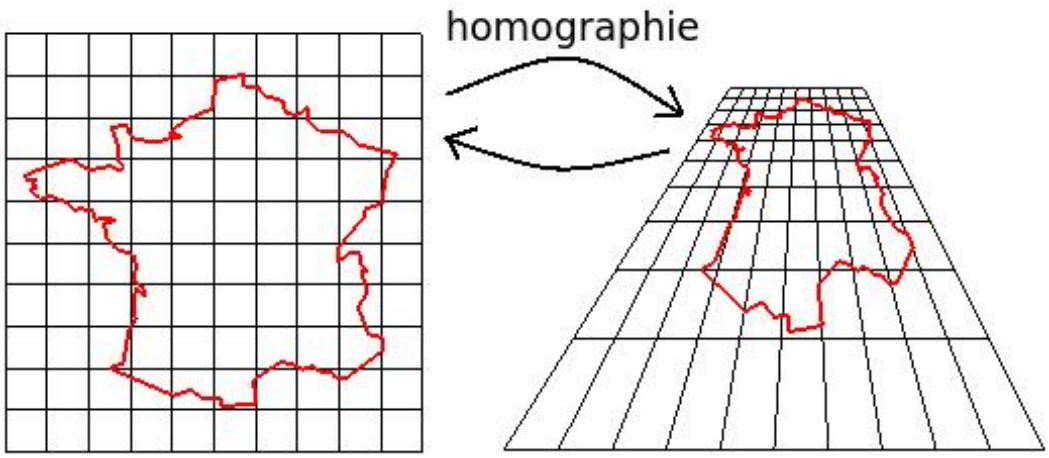

FIGURE 6.1 – Application d'homographie entre 2 plans

Il y a deux manières d'utiliser l'homographie :
— à partir de points de repère présents dans deux images, on retrouve les angles de rotation entre celles-ci,
— à partir des angles de rotation, on trouve les nouvelles positions des pixels d'une image.

C'est ce deuxième point qui nous intéresse ici. Le passage d'un plan à un autre se fait en multipliant chaque pixel de l'image initiale par la matrice d'homographie M. Cette matrice est la multiplication de la matrice des paramètres intrinsèques de la caméra par la matrice de rotation entre les plans, puis par l'inverse de la matrice intrinsèque.

La matrice d'homographie M est donc égale à $KRK^{-}1$, avec K la matrice de paramètres intrinsèques de la caméra et R la matrice de rotation entre les plans.

$$\text{K} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\text{R} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{pmatrix} . \begin{pmatrix} cos(\phi) & 0 & sin(\phi) \\ 0 & 1 & 0 \\ -sin(\phi) & 0 & cos(\phi) \end{pmatrix} . \begin{pmatrix} cos(\xi) & -sin(\xi) & 0 \\ sin(\xi) & cos(\xi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Ici, θ est l'angle de roulis, ϕ l'angle du tangage, et ξ celui du lacet.

6.1.2 Le redressement inertiel des images perspectives

Comme dit plus tôt, le premier redressement des images a été réalisé sur des images perspectives. Afin de tester le programme, une image simple était d'abord déformée, puis redressée.

Pour déformer l'image, il faut calculer la matrice de rotation à partir des angles de l'IMU. On calcule ensuite la matrice d'homographie qui est alors appliquée à l'image : , on calcule la nouvelle

position dans l'image de chaque pixel.

Algorithm 2: Algorithme du déformation perspective

Result: image perspective déformée
calcule la matrice de rotation R avec les angles de l'IMU;
calcule la matrice intrinsèque K;
calcule la matrice M;
for *tout i, j de l'image initiale* **do**
  calcule i_{new}, j_{new};
  $image_{new}(i, j) = image(i_{new}, j_{new})$;
end

Le redressement de l'image suit le même procédé à une différence près : la matrice de rotation n'est plus $R(\theta)R(\psi)R(\xi)$ mais $R(\xi)^T R(\psi)^T R(\theta)^T$ pour obtenir la transformation inverse.

6.1.3 Simulation sous logiciel Gazebo

Afin de tester l'asservissement visuel avec images perspectives, j'ai créé une simulation Gazebo (définition en page 7) simple.

Le but de cette simulation était de percevoir une mire, identique à celle utilisée pour l'asservissement réel, avec une caméra et d'envoyer l'image perçue à un nœud ROS faisant l'asservissement. La simulation de rotors, et surtout leur comportement physique, étant complexe, il a fallu faire un compromis entre le temps passé à mettre en place la simulation et la capacité à tester les réponses du nœud de traitement. Au lieu d'un drone, j'ai donc choisi de simuler un robot type char, c'est à dire un robot à quatre roues.

Pour avoir le plus de ressemblance possible entre le char et le drone, le char en lui-même est très simplifié. Il s'agit d'une plateforme suspendue à 1m du sol à laquelle est attachée verticalement la caméra. Les supports de la plateforme sont invisibles pour obtenir le plus grand champ de vision possible du sol.

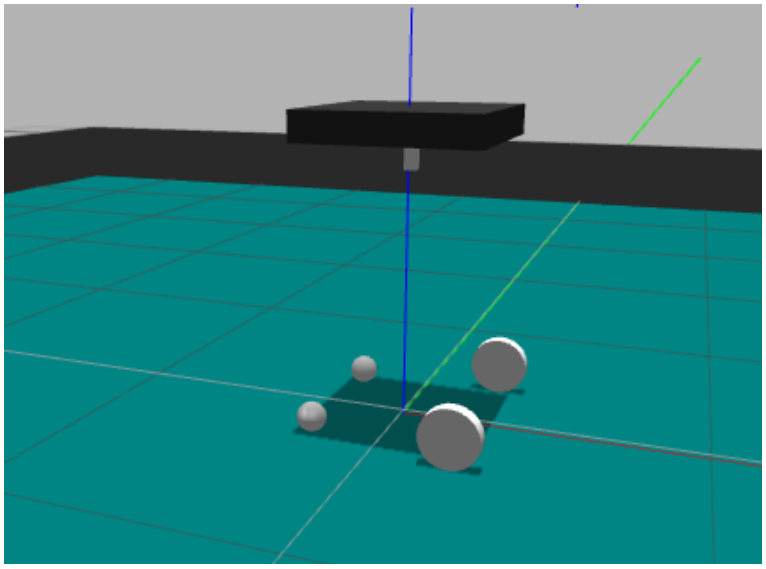

FIGURE 6.2 – Simulation du "char"

6.1.4 Résultats du redressement perspective

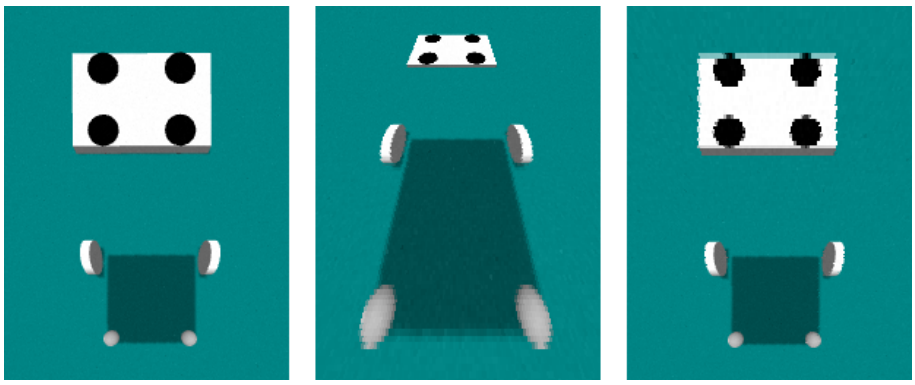

FIGURE 6.3 – Images perçue, déformée en tangage, et redressée

Les figures 6.3 illustrent une torsion, de 45 degrés, d'une image prise par la caméra simulée de Gazebo, ainsi que le redressement de l'image déformée.

D'après de nombreux tests, le redressement est correct quelque soient les angles en roulis et tangage choisis. Par contre, l'utilité de l'image redressée dépend de la présence de la mire, ou du moins de ses points, dans l'image déformée. Dans le cas contraire, la mire redressée peut être également déformée ou ne pas contenir suffisamment d'informations, c'est à dire au moins trois cercles distincts, pour l'asservissement.

Une fois le redressement perspective mis en place, j'ai mis en place sur le redressement Fisheye.

6.2 Redressement inertiel des images Fisheye

6.2.1 Le redressement

Contrairement aux images perspectives, l'homographie ne peut pas être appliquée sur les images Fisheye.

FIGURE 6.4 – Image Fisheye (gauche) et image Fisheye déformée en perspective (droite)

L'image de droite est l'image initiale avec un tangage de 23 degrés, mais ce n'est pas une rotation du contenu de l'image initiale. Pour redresser une image Fisheye, il faut tout d'abord projeter l'image plane sur une sphère virtuelle, ensuite faire tourner cette sphère, et enfin, re-projeter la sphère sur une image plane. Ce procédé a été repris de la thèse de M. Caron [7] : .

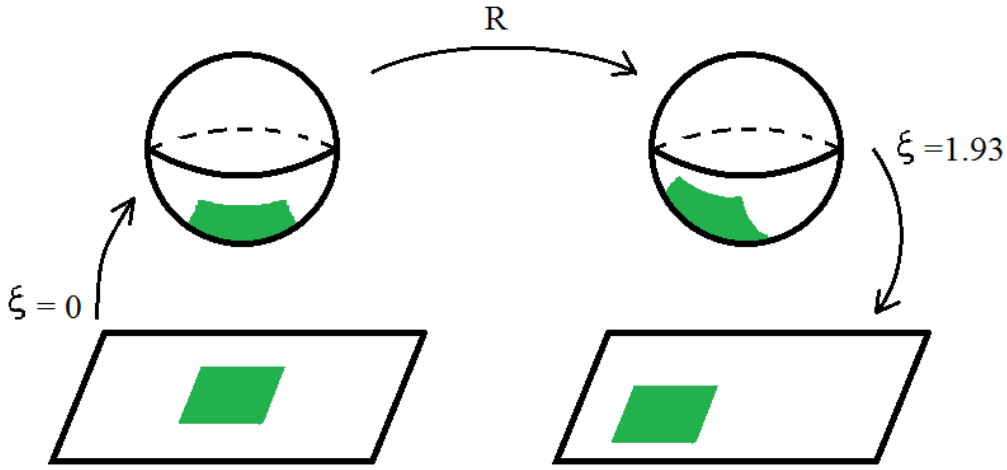

FIGURE 6.5 – Transformations pour redressement Fisheye

Les ξ utilisés dans cette partie sont ceux des caméras calibrées.

Algorithm 3: Algorithme du redressement Fisheye

Result: image Fisheye redressée
calcule la matrice de rotation R inverse avec les angles de l'IMU;
for *tout i, j de l'image initiale* **do**
    trouve les coordonnées de (i,j) sur la sphère;
    applique la rotation;
    retrouve sur le plan les coordonnées (i_{new}, j_{new});
end

Pour trouver les coordonnées sur la sphère, l'équation, également tirée de la thèse [7] est la suivante :

$$\begin{pmatrix} x_{sphere} \\ y_{sphere} \\ z_{sphere} \end{pmatrix} = \begin{pmatrix} \frac{\xi_{sphere}+\sqrt{1+(1-\xi_{sphere_e}^2)(x^2+y^2)}}{1+x^2+y^2} x \\ \frac{\xi_{sphere}+\sqrt{1+(1-\xi_{sphere_e}^2)(x^2+y^2)}}{1+x^2+y^2} y \\ \frac{\xi_{sphere}+\sqrt{1+(1-\xi_{sphere_e}^2)(x^2+y^2)}}{1+x^2+y^2} - \xi_{sphere} \end{pmatrix}$$

Les variables x et y sont les cordonnées en pixel de l'image initiale.

6.2.2 Résultats de redressement Fisheye

Les tests du redressement Fisheye ont été effectués directement sur des données de missions effectuées avec la Ricoh Theta S ainsi qu'avec la uEye.

FIGURE 6.6 – Image initiale Fisheye (gauche) et image redressée (droite)

Comme dans la figure 6.4, dans l'image initiale de la figure 6.6 le drone à un tangage de 23 degrés : l'axe $\vec{z}$ du drone n'est donc pas vertical. Au contraire, dans l'image de droite, la direction de l'axe $\vec{z}$ de l'image est bien verticale. Ceci nous permet d'ignorer le ciel, qui n'apporte pas d'informations nécessaires à l'asservissement, et d'avoir de meilleurs informations visuelles sur le sol, et donc sur l'emplacement de la mire.

Une fois l'image redressée, et donc l'impact des informations visuelles inutiles, tel le ciel, ayant diminué, on passe alors à l'asservissement visuel.

6.3 Asservissement visuel

6.3.1 Perception de la mire et création de son repère

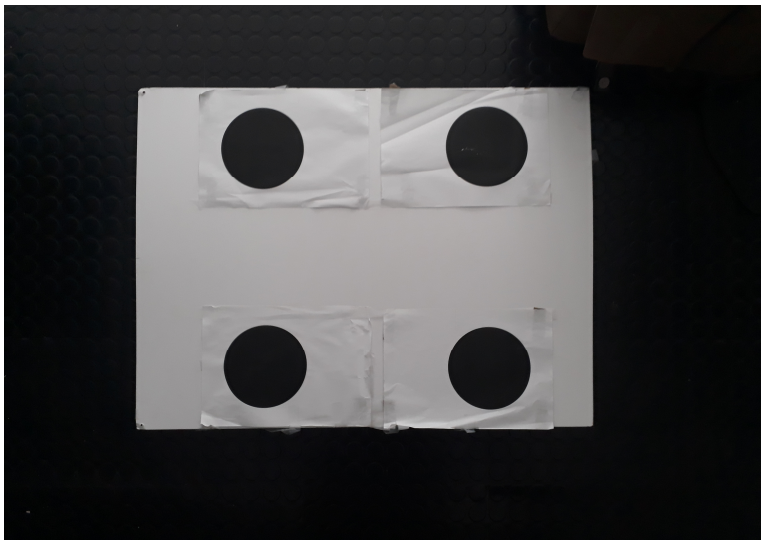

FIGURE 6.7 – Photo de la mire

La mire, en photo dans la figure 6.7, étant noire sur fond blanc, le plus simple pour la repérer a été de détecter des cercles noirs dans l'image. Le but du stage n'étant pas de fournir un travail fonctionnant en toutes situations, mais la mise en place d'un asservissement, les environnements des missions sont choisis pour ne pas apporter de confusion sur le repérage de la mire.

Si seulement trois cercles sont perçus, on estime alors la position du quatrième en fonction de celle des trois premiers. Une fois les cercles repérés, il faut trouver le repère de la mire. Cela permettra de trouver le lacet à faire pour que le drone se retrouve bien face à elle.

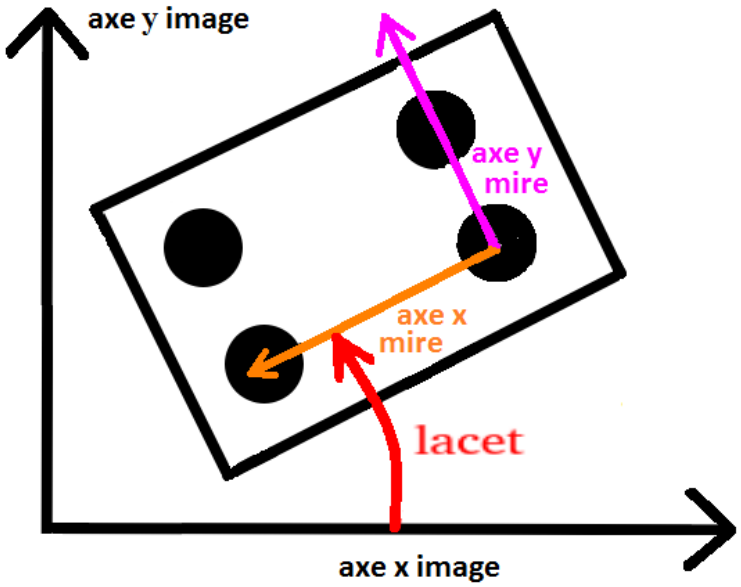

FIGURE 6.8 – Lacet à faire

Pour cela, on prend le centre de ces cercles comme points d'intérêts. Pour créer le repère de la mire, ces centres sont en premier triés en sens horaire : le barycentre des centres est calculé, et le tri se fait à partir de l'angle entre le barycentre et chaque centre, selon l'axe x du repère image. Ceci permet d'éviter que ce soit une des diagonales de la mire qui est choisie comme axe du repère.

On calcule alors les côtés du rectangle créé par les centres. La plus petite distance est assimilée à l'axe y de la mire et l'axe x est trouvé en prenant l'un des centres adjacent à l'axe y.

Le monochromatisme de la uEye complique la détection de la mire. Alors qu'avec des images couleurs, il est possible de ne garder que les parties noires de l'image, avec les images monochromes, c'est toutes les couleurs sombres qui sont reconnues comme « noires », comme le montre la figure 6.9. De ce fait, les images utilisées pour tester l'asservissement par la uEye ont été modifiées pour supprimer les parties gênantes.

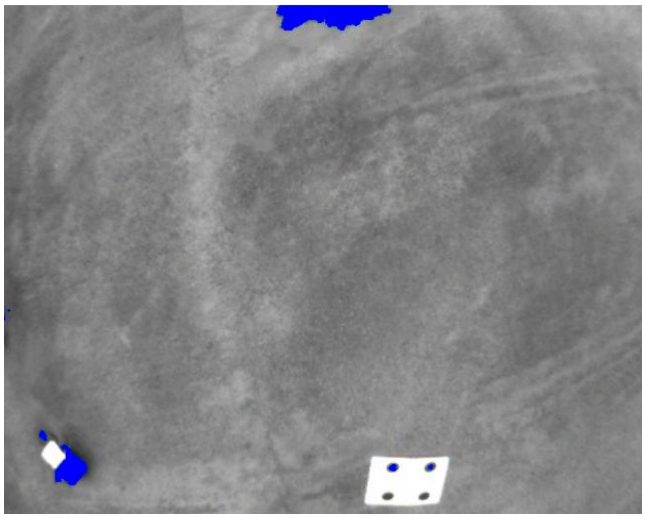

FIGURE 6.9 – Parties noires repérées (en bleu ici) sur image monochrome

6.3.2 Création du vecteur vitesse

Le but de l'asservissement visuel est ici que le drone se localise et se positionne par rapport à la mire. Pour les tests, il doit se repositionner au-dessus de l'origine du repère de la mire.

Une fois le repère de la mire trouvé, on applique des changements de repères :
— du repère mire vers le repère image,
— du repère image vers le repère caméra,
— du repère caméra vers le repère drone.

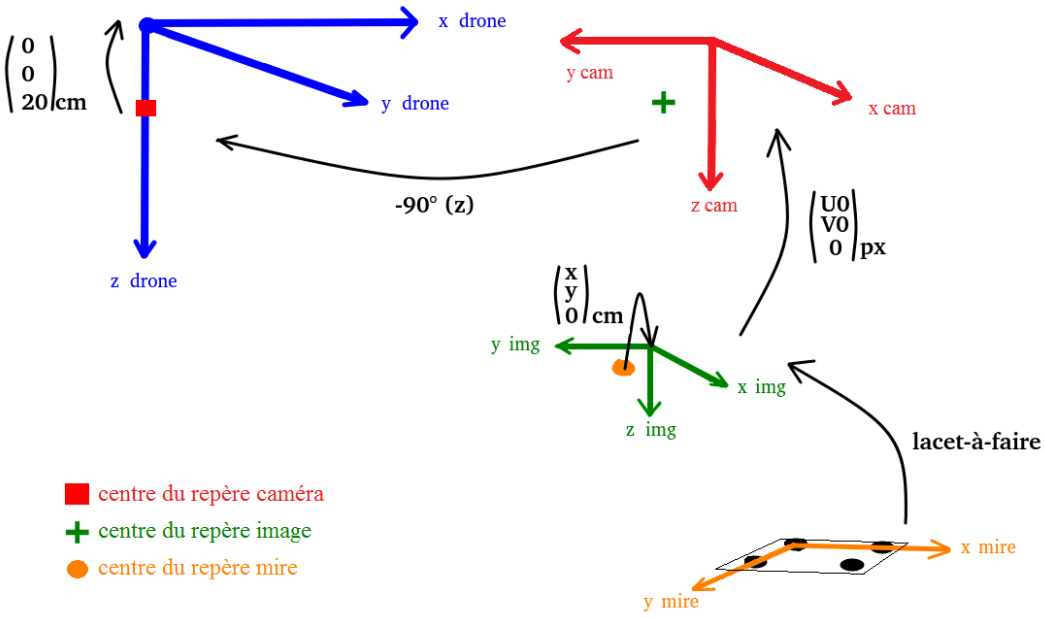

FIGURE 6.10 – Schéma des changements de repères

Cela nous donne les coordonnées de la mire dans le repère du drone.

Pour obtenir un vecteur vitesse, nous avons simplement pris la distance entre le drone et la mire perçue, en la plafonnant la vitesse maximum à 1m/s. Pour cela, on utilise la distance maximale à laquelle peut être perçue la mire en fonction de la taille de l'image vue.

$$\vec{v} = \overrightarrow{dist}/\overrightarrow{dist\_max}$$

Pour éviter que le drone ne s'arrête jamais d'osciller autour de l'origine de la mire, les vitesses sont mises à zéro si elles sont trop faibles. Ce seuil a été choisi pour que, même si le char perçoit alors un autre point de la mire comme origine, le vecteur vitesse qui en découle reste inférieur au seuil.

6.3.3 Tests du vecteur vitesse

Le vecteur vitesse calculé en fonction de la distance entre l'origine du repère de la mire et le drone a en premier été testé sous Gazebo. En effet, il est possible d'envoyer des vitesses linéaires et angulaires à Gazebo pour faire bouger les robots simulés.

Vu que l'acquisition des images vues par Gazebo et l'asservissement se trouve dans la même boucle ROS, le programme recalcule les vitesses nécessaires au robot pour se placer au-dessus de la cible toutes les 0,1 secondes. Petit à petit, le robot va ainsi se positionner au-dessus de l'origine de la cible, comme l'illustre la trajectoire sur la figure 6.11.

Lors de ce test, le char a commencé par se diriger vers le point de la mire étant perçu comme l'origine du repère de celle-ci. Il s'en est approché suffisamment pour que son vecteur vitesse soit mis à zéro, mais son inertie l'a fait continuer sur sa trajectoire. Ceci est visible sur la figure 6.12 qui montre la distance à la mire en fonction du temps, et la norme de la vitesse du char en fonction du temps. A $t = 4s$, la distance entre le char et la mire est presque nulle et la norme de sa vitesse est nulle, puis elles ré-augmentent ; c'est le moment où le char a dépassé l'origine de la mire. Le char a alors corrigé son déplacement pour revenir sur la mire.

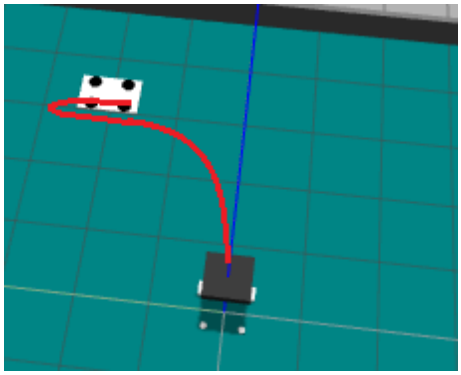

FIGURE 6.11 – Trajectoire faite par le char

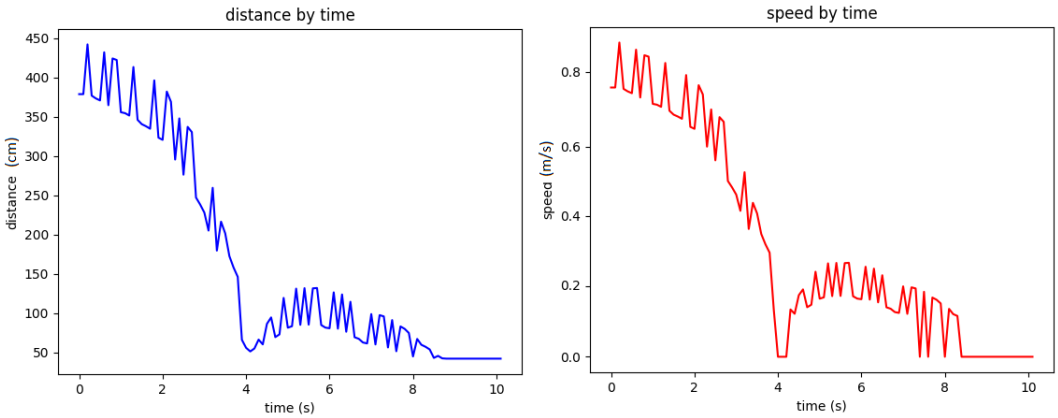

FIGURE 6.12 – Courbes de distance et de la norme du vecteur vitesse par le temps

Les variations de distance et de vitesse sur les courbes de la figure 6.12 sont dues à la perception de la mire par le char. En effet, l'origine du repère mire étant choisie en fonction des

distances entre les points de la mire perçus, il arrive que l'origine change lors de la mission.

Ensuite, l'asservissement a été testé avec les données d'une mission avec la caméra uEye. Dans ce cas-ci, la direction du vecteur vitesse est affichée par une flèche.

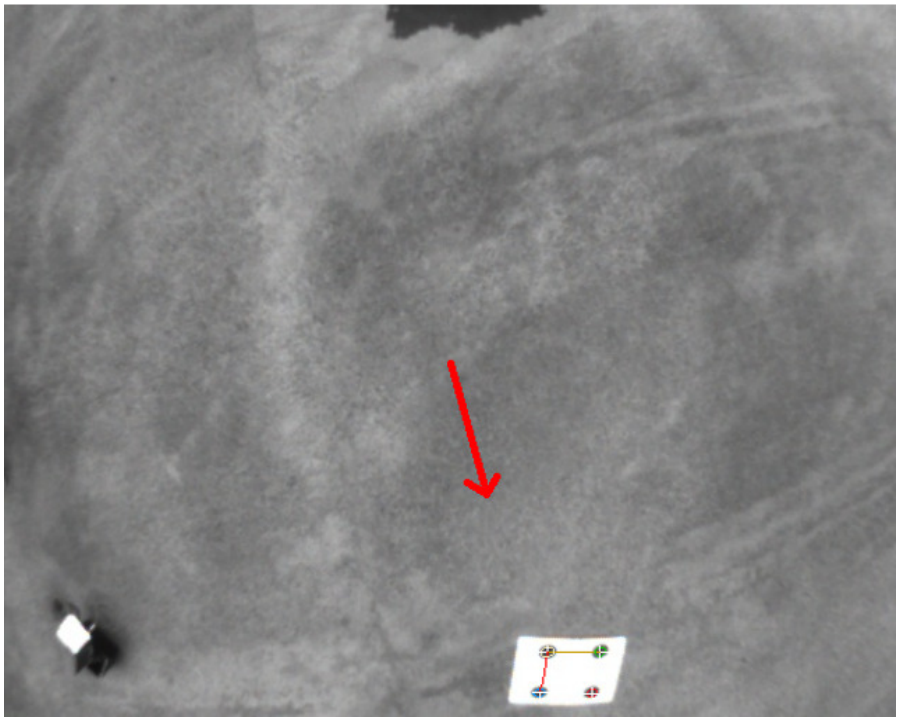

FIGURE 6.13 – Image avec mire perçue et vecteur vitesse calculé

Le vecteur vitesse Fisheye semble correct, mais il reste à le tester de manière exhaustive. Il serait possible de faire cela soit au moyen d'une simulation, soit en envoyant ce vecteur au drone.

En effet, le M600 peut être contrôlé de plusieurs manières différentes. Les principales sont par la manette ou par un smartphone, mais il existe également deux types de messages ROS sensés contrôler le drone lui-même. L'un de ces messages est basé sur l'envoi de points GPS, l'autre sur l'envoi d'un vecteur vitesse. Malgré de nombreux tests et recherches, il semble qu'il manque une instruction pour pouvoir contrôler les rotors, potentiellement un message débloquant ceux-ci.

De plus, bien que DJI ait un logiciel de simulation gratuit, celui-ci n'existe que sous Windows 10. Ceci rend son utilisation en parallèle d'un nœud ROS sous Ubuntu complexe. Il existe d'autres simulateurs d'héxacoptères, particulièrement RotorS, disponible sous Github [8], et ayant un modèle Gazebo du drone Firefly de AscTec.

FIGURE 6.14 – Drone Firefly

Afin, de pouvoir tester le programme d'asservissement, il me reste donc soit à adapter une simulation pour recevoir une commande en vecteur vitesse, soit à tenter de trouver ce qui bloque le mouvement des rotors du M600.

Conclusion

Pour l'instant, une grande partie des travaux demandés par le stage ont été effectués. L'acquisition des données visuelles, GPS et IMU fonctionne avec les deux caméras et les problèmes de connectique créés par le vent doivent pouvoir être réglés. Pour ce qui est de la latence de la Ricoh Theta S, les horodatages des données sont suffisants pour retrouver la synchronisation, mais il serait possible de décaler les lignes dans les fichiers textes pour que l'utilisation de ses données soit plus instinctive.

Pour ce qui est de l'asservissement visuel, il fonctionne correctement avec des images perspectives, mais doit encore être testé plus précisément avec des images Fisheyes. De plus, le monochromatisme de la caméra uEye crée des problèmes de perception de la mire, pour l'instant ignorés vu le but du projet, mais potentiellement complexes à résoudre.

Enfin, la dernière partie de l'asservissement, le contrôle du drone, reste à faire. C'est cette partie qui risque d'être la plus difficile à mettre en place, vu le manque d'informations disponibles en ligne et sur les programmes DJI. Le dernier mois du stage sera donc consacré à l'amélioration de l'algorithme d'asservissement ainsi qu'au contrôle du drone, aux tests correspondants, et à la rédaction de documents explicatifs pour des projets ultérieurs.

Table des figures

Liste des tableaux

Bibliographie

[1] MIS. Modélisation, Information Systèmes. https://www.mis.u-picardie.fr/. Accessed : 2019-08-18.

[2] DJI. Le leader mondial des drones. https://www.dji.com/fr. Accessed : 2019-08-15.

[3] DJI. DJI Developer. https://developer.dji.com/. Accessed : 2019-06-25.

[4] DJI. Software Environment Setup Guide - DJI Onboard SDK Documentation. https://developer.dji.com/onboard-sdk/documentation/development-workflow/environment-setup.html#ubuntu-linux. Accessed : 2019-06-25.

[5] DJI. Hardware Setup Guide - DJI Onboard SDK Documentation. https://developer.dji.com/onboard-sdk/documentation/development-workflow/hardware-setup.html. Accessed : 2019-07-02.

[6] Chang Shu Gerhard Roth. COMP 4900C : Introduction to Computer Vision. http://people.scs.carleton.ca/~c_shu/Courses/comp4900d/. Accessed : 2019-08-10.

[7] Guillaume Caron Damien Eynard. *Multiple camera types simultaneous stereo calibration*. PhD dissertation, 2011.

[8] ETHZ-ASL. RotorS is a UAV gazebo simulator. https://github.com/ethz-asl/rotors_simulator. Accessed : 2019-08-18.

Principe de ROS

ROS, Robot Operating System, est une plateforme intergicielle (définition en page 7), open source, qui permet de développer en robotique. Elle permet d'assurer la communication entre différents périphériques et des programmes de différents langages.

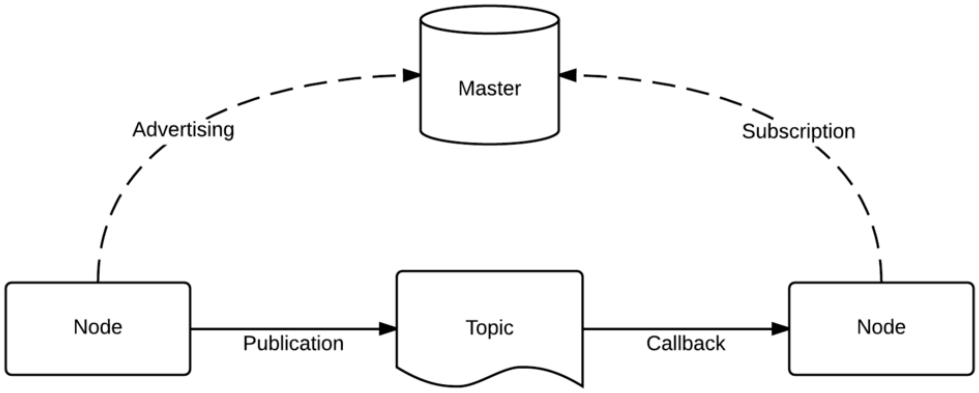

FIGURE A1 – Schéma du fonctionnement de ROS

Il existe un « master » qui se charge d'établir les connexions ente différents nœuds. Chaque nœud, « node » dans la figure A1 est capable de demander ou de transmettre des informations à d'autres nœuds au moyen de canaux, appelés « topics ».

L'avantage de ROS, et d'autres intergiciels du même type, est l'accès à des fonctionnalités standardisées, qui sont abstraites du matériel particulier du robot. Cela évite de devoir re-coder les systèmes de communication interne de chaque robot.

retour au texte