Guidage de drone pour relevé photogrammétrique sans GNSS

Méric Chevrier

Juillet 2022

Domaine Ingénierie et Architecture

Master conjoint UNIGE-HES-SO en développement territorial

Orientation Ingénierie Géomatique parcours brevet

Directeur : Dr. Adrien Gressin

Expert : Dr. Mehdi Daakir

Mémoire n° : 1016

UNIVERSITÉ DE GENÈVE

Hes·so
Haute Ecole Spécialisée de Suisse occidentale

Résumé

Chaque 5 ans, les exploitants de barrages sont tenus d'effectuer une inspection détaillée des parements amonts et avals de leurs barrages afin de détecter d'éventuelles anomalies. Actuellement ces inspections se font manuellement par les barragistes à l'aide de nacelle. L'utilisation de drone permet d'obtenir des clichés à haute résolution et de produire des orthophotos. Les analyses pourraient donc être effectuées sur un SIG.

L'environnement créé par un barrage n'est pas optimal pour le relevé par drone étant donné que d'importants masques GNSS sont présents. Ce travail a donc pour but de développer une méthode permettant l'automatisation de relevé par drone dans des zones dépourvues de signal GNSS.

Des mesures terrestres issues d'une station totale sont utilisées comme moyen externe pour positionner le drone par l'intermédiaire d'un filtre de Kalman. Les corrections à appliquer sur le drone pour qu'il suive le plan de vol défini sont calculées par un contrôleur PID. L'ensemble du processus est intégré dans une application Andoid en Java.

Les résultats des tests montrent qu'il est possible d'automatiser ce processus de guidage de drone via des mesures terrestres. Cependant, l'utilisation de drones commerciaux de type DJI présente des limitations au niveau de la liberté de leur guidage.

Mots-clés : Filtre de Kalman, Guidage de drone, Positionnement sans GNSS, Contrôleur PID, MSDK DJI, Application mobile

Abstract

Every 5 years, dam operators are required to carry out a detailed inspection of the upstream and downstream faces of their dams to detect any anomalies. Currently, these inspections are carried out manually by the dam operators with the help of a gondola. The use of a drone makes it possible to obtain high-resolution images and produce orthophotos. The analyses could therefore be carried out on a GIS.

The environment created by a dam is not optimal for drone surveys as important GNSS masks are present. The aim of this work is to develop a method to automate drone surveys in areas without GNSS signals.

Ground measurements from a total station are used as an external means to position the UAV via a Kalman filter. The corrections to be applied to the UAV to follow the defined flight plan are calculated by a PID controller. The whole process is integrated in an Andoid application in Java.

The test results show that it is possible to automate this process of drone guidance via ground measurements. However, the use of commercial UAVs such as DJI has limitations in terms of the freedom of their guidance.

Keywords : Kalman filter, UAV guidance, GNSS-free positioning, PID controller, MSDK DJI, Mobile application

Table des matières

Table des figures

Liste des tableaux

Remerciements

Je tiens à remercier toutes les personnes qui m'ont aidé à la réalisation de ce projet :

Monsieur Adrien Gressin, Dr. ing. ENSG, professeur à la HEIG-VD et directeur de ce travail, pour sa disponibilité permanente et le temps consacré tout au long de ce travail de Master ;

Monsieur Mehdi Daakir, Dr. ing. ENSG, expert de ce travail pour les conseils et remarques lors de la défense intermédiaire ;

Monsieur, Sébastien Guillaume, Dr. ETHZ, professeur à la HEIG-VD pour sa disponibilité et son aide pour le filtrage de Kalman ;

L'entreprise IG group SA, pour la mise à disposition de stations totales et de drone pour effectuer différents tests ;

L'électricité de le Lienne SA, pour la mise à disposition du site du barrage de Zeuzier pour effectuer différents tests.

Quentin Bobillier, camarade du Master pour la relecture technique et critique ;

Ma famille, pour la relecture du présent rapport.

Abréviations

IMU : Intertial Measurement Unit
GNSS : Global Navigation Satellite System
RTK : Real Time Kinematic (Cinématique temps réel)
MN95 : Mensuration Nationale 1995 (système de coordonnées)
WGS84 : World Geodetic System 1984 (système de coordonnées)
SDK : Software Development Kit, Kit de Développement Logiciel
MSDK : Mobile SDK
OSDK : Onboard SDk
GSD : Ground sampling distance (Résolution au sol)

Introduction

Contexte de l'étude

Chaque 5 ans, les exploitants de barrages sont tenus d'effectuer une inspection détaillée des parements amonts et avals de leurs barrages afin de détecter d'éventuelles anomalies. D'année en année, les différentes fissures et marques d'érosion sont répertoriées sur un plan. Actuellement, cette inspection se fait manuellement sur le terrain. Des barragistes passent plusieurs jours sur des nacelles suspendues au parement à répertorier sur un plan tous les nouveaux éléments.
Avec les techniques actuelles, il est possible d'effectuer ces inspections directement sur des images hautes résolutions. Plusieurs solutions sont imaginables :

— **Relevé avec un appareil photo depuis un point fixe** : En utilisant un téléobjectif et un appareil photo haute résolution, il est possible d'obtenir des images à haute résolution d'un parement de barrage. Le processus de prise de vue peut être automatisé par l'utilisation d'un système panoramique (Figure 1). Il est ensuite possible d'obtenir une image panoramique à haute résolution en assemblant les différentes prises de vues. Selon la configuration de l'objet à inspecter, il sera nécessaire d'acquérir des images depuis différents points de vue. Le produit final obtenu est une simple image panoramique où il n'est pas possible de prendre des mesures. De plus, la résolution de cette image varie de part et d'autre, étant donné que la distance entre l'appareil photo et le mur varie d'image en image ;

FIGURE 1 – Système de prise de vue panoramique automatique Rodeon [1]

— **Relevé par drone** : Le relevé par drone permet d'effectuer des prises de vues à une distance constante du parement et ainsi obtenir des images avec une résolution similaire sur l'ensemble du barrage. En ajoutant des points de calage, il est possible de calculer une orthophoto du parement. Les barragistes ont alors la possibilité de prendre des mesures directement sur l'image pour effectuer leur inspection. L'orthophoto étant référencée, il est possible de superposer plusieurs états de mesures pour analyser l'évolution des dégradations du barrage. Le

1. Source : `https://dr-clauss.net/rodeon-pixplorer-2/`

relevé par drone donne en tout point un résultat de meilleure qualité que le relevé avec un appareil photo depuis un point fixe ;

Problématique

Ce travail a pour but d'étudier le relevé de parement de barrage par drone et plus particulièrement l'automatisation de ce processus. Afin d'obtenir une orthophoto à haute résolution (1-3 mm/px), il est nécessaire d'effectuer un nombre important de prises de vues à une distance assez faible du parement (10-15 m).

Pour automatiser la prise de vues des images et pour assurer la résolution et le recouvrement, il est indispensable de pouvoir fournir au drone un plan de vol à suivre. Actuellement, des applications permettent de gérer des plans de vol verticaux, mais le guidage du drone se fait à partir de son récepteur GNSS interne. Pour garantir le bon suivi du plan de vol, le drone doit pouvoir déterminer correctement sa position à l'aide d'un signal GNSS fiable.

Lors de l'oscultation de barrage, le drone étant très proche du mur ou en fond de vallée, il n'y a en général pas de signal GNSS permettant un positionnement précis. Ce sujet consiste donc à développer une méthode pour guider un drone sans l'utilisation de son système GNSS embarqué.

Le guidage de drone sans GNSS n'est pas uniquement applicable pour le cas d'un relevé de barrage, cette méthodologie peut être utilisée dans n'importe quel contexte qui ne permet pas l'utilisation de GNSS.

Chapitre 1

État de l'art

Ce chapitre à pour but de présenter les différentes solutions disponible en ce qui concerne :

1. **Les systèmes de positionnement d'un drone** (Chapitre 1.2) ;
2. **Les drones disponibles sur le marché** (Chapitre 1.3) ;
3. **Le calcul de trajectoire en temps réel** (Chapitre 1.4) ;
4. **Le calcul de correction de trajectoire** (Chapitre 1.5) ;

1.1 Aspects théoriques

1.1.1 Principaux axes d'un avion/drone

Généralement, les rotations d'un aéronef en trois dimensions sont définies par les paramètres : yaw, pitch et roll.

— yaw ou lacet : rotation à gauche ou à droite autour d'un axe pointant vers le bas ;
— pitch ou tangage : rotation du nez vers le haut ou le bas autour d'un axe reliant les ailes ;
— roll ou roulis : rotation autour d'un axe reliant le nez à la queue.

Ces axes sont solidaires au véhicule et se déplacent avec celui-ci.

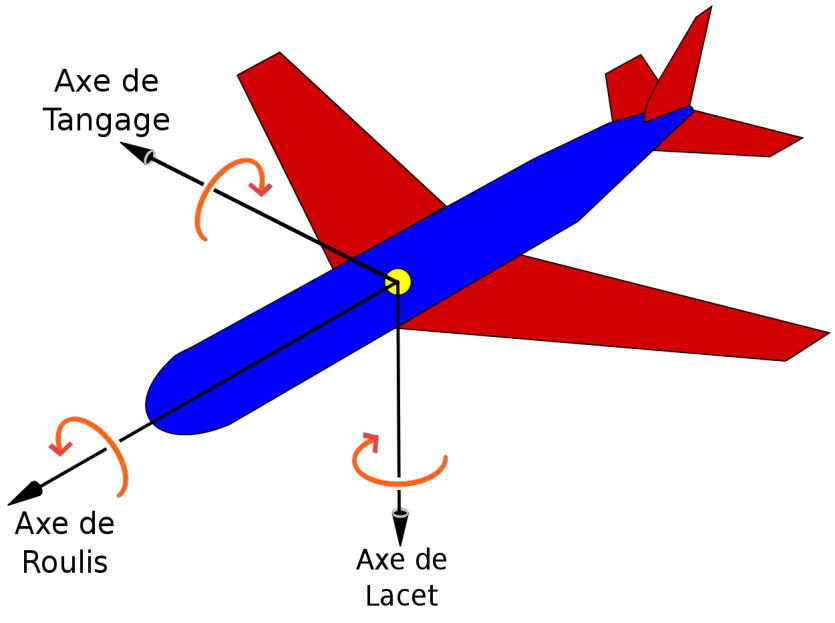


FIGURE 1.1 – Dénomination conventionnelle des axes principaux d'un aéronef [1]

1.2 Système de positionnement d'un drone

Ce chapitre a pour but d'exposer les différentes méthodes qui pourraient être utilisées pour positionner un drone en vol en temps réel et en trois dimensions. Une de ces méthodes sera retenue pour la suite du travail.

1.2.1 GNSS/IMU

Le positionnement par GNSS est sûrement la méthode la plus simple et la plus utilisée actuellement pour le positionnement en temps réel d'un aéronef. Couplée à une IMU, cette méthode permet d'obtenir des données à haute fréquence. Avec l'utilisation d'un service de correction, il est possible d'obtenir en temps réel des coordonnées MN95 (ou autre système) à une précision de quelques centimètres. Comme mentionné dans l'introduction de cette étude, le but est justement de développer une méthode de guidage indépendante du système GNSS. Cette méthode ne sera donc pas utilisée pour la suite du projet.

1.2.2 Positionnement relatif - SLAM

En équipant un drone d'un lidar [Xin et al., 2020], d'un distancemètre ou d'une caméra stéréoscopique [Jin et al., 2018], celui-ci est capable de se positionner par rapport à son environnement. Il s'agit alors de SLAM [Motlagh et al., 2019], [Vanhie-Van Gerwen et al., 2021]. En ajoutant plusieurs capteurs mesurant dans différentes directions, il est possible d'obtenir un positionnement en 3D. Cette technique requiert d'avoir une modélisation de l'environnement dans lequel le drone doit se déplacer. Pour obtenir un positionnement en 3D, des mesures doivent être effectuées dans trois directions bien distinctes, ce qui est facile dans le cas d'un vol, dans un hangar par exemple. Cependant, en extérieur il est difficile d'effectuer des mesures dans trois directions sur des objets qui seront certainement assez éloignés.

1.2.3 Positionnement par caméra

Il est possible de calculer la position en 3D d'un drone par triangulation [Pan, 2021]. Pour cela plusieurs contraintes sont à prendre en compte :

- Plusieurs caméras sont nécessaires ;
- La position et l'orientation des différentes caméras doivent être déterminées ou connues ;
- Les caméras doivent être synchronisées avec précision ;
- Le drone doit pouvoir être détecté dans les images ;
- Le champ de vision d'une caméra est limité ;
- La géométrie du réseau de caméra est importante afin d'avoir un bon recoupement.

L'ensemble de ces contraintes nécessite une mise en place assez lourde. Cependant, ce système permet d'effectuer des mesures à haute fréquence, à plus de 100 Hz.

1.2.4 Positionnement par station totale

Une drone peut être positionné en 3D en utilisant de manière conventionnelle une station totale. Un mini-prisme 360° (Figure 1.2) placé sur le drone permet à l'instrument d'effectuer ses mesures. Avec ce système, un seul instrument est nécessaire, mais la fréquence de mesure d'environ 10 Hz est plus faible que les systèmes précédents. Contrairement au système utilisant des caméras, il est impossible de trigger les mesures d'une station totale. En effet, il existe, sur certaines caméras, des ports pour trigger avec précision le déclenchement des images, ce qui n'existe par sur les stations totales. Il est

1. Source : `https://en.wikipedia.org/wiki/Flight_dynamics`

donc compliqué de synchroniser avec précision les mesures effectuées avec ce type d'instrument. Des incertitudes d'environ 15 ms sont possibles [Pan, 2021].

FIGURE 1.2 – Mini prisme 360° Leica ; Dimension : 28 X 30 mm ; Pois : 30 g

1.2.5 Positionnement par vidéo-théodolite

Le principe de base est similaire à celui du Chapitre 1.2.3, la position du drone est calculée par une triangulation. Deux instruments sont donc nécessaires au minimum. Ce système consiste à coupler une caméra et une station totale en remplaçant l'oculaire de l'instrument par une caméra CCD, comme le système QDaedalus (Figure 1.3). L'avantage de ce système réside dans le fait que la position et l'orientation des caméras sont directement déterminées par la station totale. De plus, en utilisant un instrument robotisé, il est possible de suivre un objet en déplacement. Il n'y a donc plus de contrainte liée au champ de vision de la caméra.

FIGURE 1.3 – Caméra CCD ; Système QDaedalus installé sur une station totale [2].

2. Source : Manuel QDaedalus

1.2.6 Solution retenue

La méthode retenue pour positionner le drone en temps réel et en 3D est la station totale 1.2.4. Cette méthode est retenue pour les raisons suivantes :

- Méthode simple de fonctionnement et facile à mettre en place ;
- Le matériel nécessaire est du matériel commun pour un bureau de géomètre ;
- Plus robuste aux conditions de visibilité difficile que des caméras ;
- Plus intéressante pour le cas d'utilisation sur un barrage. Pour avoir un visuel sur l'ensemble du parement, l'opérateur doit dans certains cas se placer à plusieurs centaines de mètres du mur et donc du drone. Une station totale n'a pas de difficulté à effectuer des mesures précises à ces distances, contrairement à une méthode utilisant uniquement des caméras ;
- Un seul instrument à un seul emplacement est nécessaire. Il n'y a donc pas de contrainte sur la géométrie du réseau de caméras par exemple. Dans le cas d'un barrage, l'ensemble des caméras seraient positionnées dans la même direction pour viser le parement, ce qui n'est pas idéal pour déterminer la position du drone par triangulation ;
- Il n'y a pas de limite de champ de vision, le drone peut être mesuré à 360° ;

1.3 Analyse des drones disponibles sur le marché

Les deux constructeurs principaux de drone grand public sont Parrot (Chapitre 1.3.2) et DJI (Chapitre 1.3.1). Ces deux constructeurs proposent des solutions économiquement intéressantes et prêtes à l'emploi, sans développement ou compétences spécifiques. Il est également possible de construire des drones sur-mesures selon des besoins spécifiques avec des solutions open-source (Chapitre 1.3.3). L'analyse se porte en grande partie sur les possibilités de pilotage du drone automatisé par un script. En ce qui concerne la partie caméra, les différentes solutions offrent des capteurs de grandes qualités et largement utilisés pour des travaux de photogrammétrie.

1.3.1 Drone DJI

Le pilotage du drone est possible au travers de deux *Software Development Kit, Kit de Développement Logiciel* (SDK) principaux : le *Mobile SDK* (MSDK) [3] et le *Onboard SDK* (OSDK) [4].

De plus, DJI met à disposition un simulateur [5] qui permet de tester les développements effectués à l'aide des SDK avant d'effectuer un vol réel.

Mobile SDK

La majorité des fonctionnalités des produits DJI sont accessibles via le MSDK. Les développeurs peuvent contrôler le drone, la caméra et son gimbal, récupérer le flux vidéo en live et accéder aux données des différents senseurs.

Différents modes de contrôle de vol sont possibles :

- **Manuel** : Le pilotage du drone se fait à l'aide de la télécommande et le MSDK sert uniquement à récupérer le flux vidéo et les données des capteurs ;
- **Joystick virtuel** : Le MSDK permet de créer des mouvements de joystick virtuels pour simuler un pilote ;
- **Mission** : Un plan de vol peut être donné et suivi par le drone ;

3. https://developer.dji.com/mobile-sdk-v4/
4. https://developer.dji.com/onboard-sdk/
5. https://www.dji.com/ch/downloads/softwares/assistant-dji-2-for-phantom

Les différents paramètres liés à la caméra peuvent être gérés :
- **Mode de la caméra** : Vidéo ou capture d'images simples ;
- **Exposition** : ISO, ouverture ;
- **Paramètres d'image** : Dimension, contraste, filtre ;
- **Paramètres de vidéo** : Dimension, FPS ;
- **Gestion du gimbal** : Orientation de la caméra sur 3 axes ;

Les caractéristiques suivantes sont également disponibles :
- Accès aux senseurs internes à 10 Hz (vitesse, attitude, compas, GNSS) ;
- Accès à la détection des obstacles ;
- Accès aux informations du drone (Niveau des batteries, État de la connexion entre le drone et la télécommande, État des capteurs internes, Mode de contrôle) ;

Le MDSK est disponible sur IOS et sur Android. Son utilisation passe par le développement d'une application et les communications avec le drone se font au travers de la télécommande DJI couplée au drone (Figure 1.4).

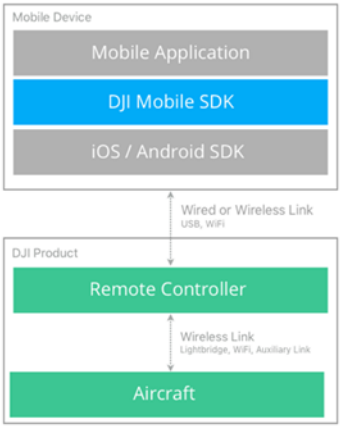


FIGURE 1.4 – MSDK, Plateforme de développement et communication [6].

Finalement, l'ensemble des modèles DJI sont compatibles avec le MSDK.

Onboard SDK

Le OSDK permet d'exécuter des applications directement sur un ordinateur embarqué sur le drone. Cet ordinateur doit être rajouté sur le drone et est compatible uniquement avec les modèles de la gamme Matrice de DJI. Des capteurs supplémentaires peuvent être rajoutés sur le drone et communiqués avec l'ordinateur de bord via des ports séries.

Les mêmes caractéristiques que le MSDK sont disponibles avec le OSDK. Le OSDK permet cependant d'accéder aux données des capteurs internes jusqu'à 200 Hz.

Le OSDK est disponible sur plusieurs plateformes de développement :
- Linux ;
- ROS ;
- Tous systèmes embarqués qui supportent la communication série ;

6. Source : `https://developer.dji.com/mobile-sdk-v4`

1.3.2 Drone Parrot

Les possibilités offertes par Parrot sont assez similaires à celles décrites au Chapitre précédent 1.3.1. Le pilotage du drone est là aussi possible au travers de différents SDK : le Ground SDK[7] et le Air SDK[8].

Ground SDK

Ce SDK permet au développeur de créer une application mobile pouvant contrôler le drone. Les plateformes de développement IOS et Android sont disponibles. Le Ground SDK de Parrot correspond au Mobile SDK de chez DJI. Les fonctionnalités suivantes sont accessibles via ce SDK :

- Contrôle sur les déplacements du drone (Mode Waypoint, Mode Roll/Pitch/Yaw en donnant des angles au drone) ;
- Contrôle des paramètres de caméra et de gimbal ;
- Accès aux données des senseurs internes (Altimètre, Compas, GNSS, Vitesse) ;
- Accès à la détection des obstacles ;
- Flux vidéo en live ;

Air SDK

Le Air SDK de Parrot est l'équivalent du Onboard SDK de chez DJI. Il s'agit donc d'une application qui peut être exécutée directement sur le drone. Aucun ordinateur supplémentaire ne doit être rajouté sur le drone, contrairement à la solution de DJI. L'accès aux données des senseurs est possible à une fréquence plus élevée allant jusqu'à 200 Hz. Des modes de déplacements supplémentaires sont disponibles par rapport au Ground SDK :

- **Move to** : navigation en donnant une coordonnée GNSS ;
- **Relative mode** : navigation en donnant un déplacement relatif à la position actuelle en fournissant au drone une distance, un cap et une altitude ;

1.3.3 Drones modulaires

L'avantage de ce type de drone par rapport aux drones commerciaux prêts à l'emploi est qu'il n'y a pas de limite au niveau software et hardware. Les capteurs embarqués du drone peuvent être sélectionnés et optimisés pour un but bien précis. L'utilisateur a ainsi la possibilité de gérer l'ensemble du système sans limite. L'inconvénient est que ce type de drone nécessite une construction sur-mesure et des développements logiciels.

Plusieurs solutions open source de contrôleur de vol existent sur le marché, les deux principaux sont **PX4**[9] et **Ardupilot**[10]. Ces deux contrôleurs sont la partie "logiciel" et se basent sur la même partie hardware : **Pixhawk**[11]. Ce contrôleur (Figure 1.5) permet d'intégrer différents capteurs :

- Capteurs intégrés dans le contrôleur :
  - 3 IMU ;
  - 2 Baromètres ;
  - 1 Magnétomètre
- Capteurs optionnels :
  - 2 Ports GNSS sont disponibles ;

7. https://developer.parrot.com/docs/groundsdk-android
8. https://developer.parrot.com/docs/airsdk
9. https://px4.io/
10. https://ardupilot.org/
11. https://pixhawk.org/

— Capteur de distance ;
— Capteur de flux optique ;
— Servo moteur ;
— Port pour le contrôle d'un gimbal et d'une caméra ;

FIGURE 1.5 – Contrôleur de vol Pixhawk 5X [12] ;

Concernant le drone en lui-même, des kits comprenant la structure de base ainsi que les moteurs et contrôleurs des moteurs sont disponibles (Figure 1.6)

FIGURE 1.6 – Kit de base [13] ;

12. Source : `https://px4.io`
13. Source : `https://holybro.com`

Le coût approximatif pour la construction d'un drone serait le suivant :

Coût du matériel	
Kit de base [14]	250.-
Contrôleur de vol Pixhawk [15]	270.-
Antenne GNSS [16]	50.-
Radio/Télémétrie [17]	280.-
Capteur de flux optique [18]	130.-
Capteur de distance [19]	100.-
Caméra industrielle	1500.-
Gimbal 3 axes [20]	1500.-
Total	4080.-

TABLE 1.1 – Coût approximatif pour un drone modulaire

Ce coût prend uniquement en compte la partie hardware, il faut encore rajouter à cela un coût en temps pour le développement du système.

1.3.4 Solution retenue

Pour ce travail, la solution passant par la construction et le développement d'un drone modulaire n'est pas retenue étant donné que celle-ci demande un temps important de mise en place. Le but de ce travail étant de développer une méthodologie de guidage et non le développement d'un drone, il a été choisi de s'orienter vers une solution prête à l'emploi.

Les solutions offertes par les deux constructeurs DJI et Parrot sont très proches. Cependant, le SDK de DJI semble beaucoup plus utilisé par la communauté que celui de Parrot. Un grand nombre de forum et de documentations sont donc disponibles sur ce thème. Les mises à jour des SDK de DJI semblent également plus régulières et plus récentes. Les drones DJI sont également plus populaires que les drones Parrot et quasiment chaque géomètre en est équipé.

La solution retenue est le développement d'une application Android en utilisant le Mobile SDK de DJI. Le mode de contrôle de vol employé sera "Joystick virtuel" étant donné qu'il permet de gérer totalement les déplacements du drone en appliquant des corrections sur sa trajectoire.

15. https://shop.holybro.com/x500-v2-kit_p1288.html
16. https://shop.holybro.com/pixhawk-5x_p1279.html
17. https://shop.holybro.com/holybro-m8n-gps_p1094.html
18. https://shop.holybro.com/sik-telemetry-radio-v3_p1103.html
19. https://shop.holybro.com/px4flow-kit_p1035.html
20. https://gremsy.com/pixy-u-store

1.4 Calcul de trajectoire en temps réel

Avant de pouvoir effectuer le guidage d'un drone, il faut pouvoir le positionner en temps réel en 3D. Un filtre de Kalman est utilisé pour calculer la trajectoire en live du drone.

1.4.1 Filtre de Kalman

Le filtre de Kalman [21] est un filtre à réponse impulsionnelle infinie qui permet d'estimer l'état d'un système à partir de mesures incomplètes ou bruitées [Guillaume, 2022]. Dans le cas de ce travail, ce filtre aura pour but d'estimer la trajectoire du drone. Ce filtre est un estimateur récursif, cela veut dire que seul les estimations de l'état précédent et les mesures de l'état actuel sont nécessaires pour estimer l'état courant. Le filtre se décompose en deux phases :

— **Prédiction** : cette phase utilise la prédiction de l'état précédent pour estimer l'état courant ;
— **Mise à jour** : cette phase utilise les mesures de l'état courant pour corriger l'état prédit afin d'obtenir un résultat plus précis ;

Modèle mathématique :

$$\begin{cases} \mathbf{x}_t = \mathbf{F}_{t-1} \cdot \mathbf{x}_{t-1} + \mathbf{G}_{t-1} \cdot \mathbf{w}_{t-1} \\ \mathbf{l}_t = \mathbf{A}_t \cdot \hat{\mathbf{x}}_t - \hat{\mathbf{v}}_t \end{cases} \tag{1.1}$$

avec :

$$\begin{aligned} &\mathbf{x}_t \text{ : Vecteur d'état à l'instant } t \\ &\mathbf{x}_{t-1} \text{ : Vecteur d'état à l'instant } t-1 \\ &\mathbf{w}_{t-1} \text{ : Vecteur de bruit blanc du système à l'instant } t-1 \\ &\mathbf{F}_{t-1} \text{ : Matrice de propagation du système à l'instant } t-1 \\ &\mathbf{G}_{t-1} \text{ : Matrice de diffusion du bruit d'état à l'instant } t-1 \\ &\\ &\mathbf{l}_t \text{ : Vecteur des observations à l'instant } t \\ &\hat{\mathbf{x}}_t \text{ : Vecteur d'état estimé à l'instant } t \\ &\hat{\mathbf{v}}_t \text{ : Vecteur des résidus estimés à l'instant } t \\ &\mathbf{A}_t \text{ : Matrice du modèle fonctionnel à l'instant } t \end{aligned} \tag{1.2}$$

L'écart type de l'unité de poids est, par convention, fixé à $\sigma_0 = 1$. Ainsi, les matrices des cofacteurs et de variance-covariance sont toujours égales :

$$\begin{aligned} \mathbf{Q}_{\mathbf{ww}} &= \mathbf{K}_{\mathbf{ww}} \\ \mathbf{Q}_{\mathbf{ll}} &= \mathbf{K}_{\mathbf{ll}} \end{aligned} \tag{1.3}$$

Estimation séquentielle :

A chaque époque t l'estimation du vecteur d'état $\mathbf{x}_t$ se décompose en deux étapes :

1. La prédiction du vecteur d'état $\hat{\mathbf{x}}_t^-$ et de sa matrice de variance-covariance $\mathbf{Q}_{\hat{\mathbf{x}}_t^-\hat{\mathbf{x}}_t^-}$. Cette prédiction se fait sur la base de l'état mis à jour à l'époque précédente.

$$\begin{aligned} \hat{\mathbf{x}}_t^- &= \mathbf{F}_{t-1} \cdot \hat{\mathbf{x}}_{t-1}^+ \\ \mathbf{Q}_{\hat{\mathbf{x}}_t^-\hat{\mathbf{x}}_t^-} &= \mathbf{F}_{t-1} \cdot \mathbf{Q}_{\hat{\mathbf{x}}_{t-1}^+\hat{\mathbf{x}}_{t-1}^+} \cdot \mathbf{F}_{t-1}^T + \mathbf{G}_{t-1} \cdot \mathbf{Q}_{\mathbf{ww}} \cdot \mathbf{G}_{t-1}^T \end{aligned} \tag{1.4}$$

21. https://fr.wikipedia.org/wiki/Filtre_de_Kalman

2. La mise à jour, à l'aide des observations $\mathbf{l}_t$, du vecteur d'état $\hat{\mathbf{x}}_t^+$ et de sa matrice de variance-covariance $\mathbf{Q}_{\hat{\mathbf{x}}_t^+\hat{\mathbf{x}}_t^+}$.

$$\begin{aligned} \mathbf{K} &= \mathbf{Q}_{\hat{\mathbf{x}}_t^-\hat{\mathbf{x}}_t^-} \cdot \mathbf{A}_t^T \cdot (\mathbf{A}_t \cdot \mathbf{Q}_{\hat{\mathbf{x}}_t^-\hat{\mathbf{x}}_t^-} \cdot \mathbf{A}_t^T + \mathbf{Q}_{\mathbf{ll}})^{-1} \\ \hat{\mathbf{x}}_t^+ &= \hat{\mathbf{x}}_t^- + \mathbf{K} \cdot (\mathbf{l}_t - \mathbf{A}_t \cdot \hat{\mathbf{x}}_t^-) \\ \mathbf{Q}_{\hat{\mathbf{x}}_t^+\hat{\mathbf{x}}_t^+} &= \mathbf{Q}_{\hat{\mathbf{x}}_t^-\hat{\mathbf{x}}_t^-} - \mathbf{K} \cdot \mathbf{A}_t \cdot \mathbf{Q}_{\hat{\mathbf{x}}_t^-\hat{\mathbf{x}}_t^-} \end{aligned} \tag{1.5}$$

1.5 Calcul de correction de trajectoire

Comme expliqué au chapitre 1.3, la solution retenue pour le drone passe par l'utilisation du MSDK de chez DJI. Les déplacements du drone seront effectués en fournissant des commandes de déplacement via le mode "Joystick virtuel". Pour appliquer ces commandes, il faut mettre en place un contrôleur capable de calculer les corrections à appliquer sur le drone.

Dans une situation "parfaite", sans erreur et imprécision sur les différents paramètres qui composent le modèle de déplacement du drone, il suffirait de donner un vecteur de déplacement V au point A, puis de stopper le déplacement lorsque le point B est atteint (Figure 1.7).

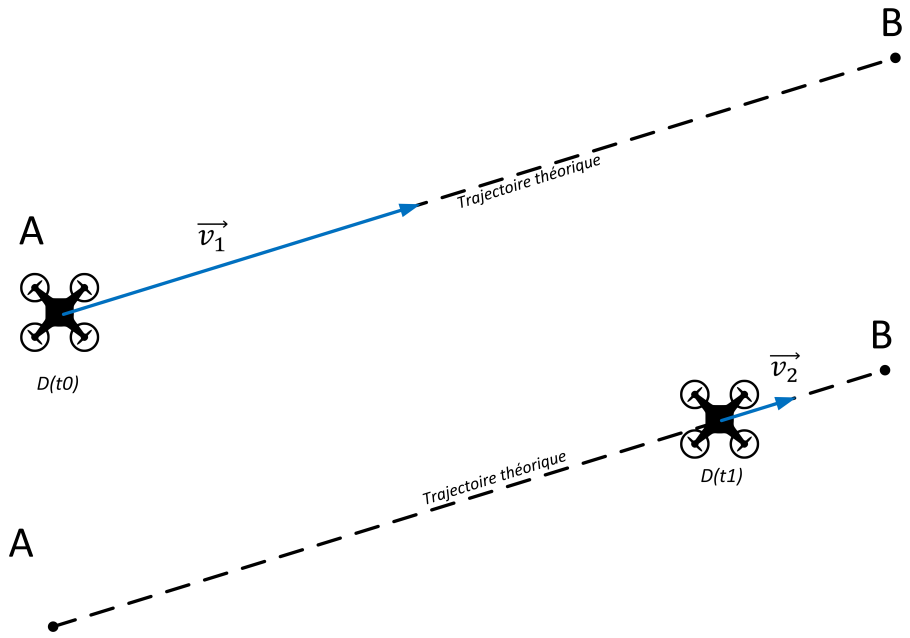


FIGURE 1.7 – Déplacement du drone selon une trajectoire théorique - Situation parfaite.

Or, en réalité, les déplacements du drone sont impactés par des forces supplémentaires comme le vent ou la résistance de l'air. Les mouvements sont également influencés par la précision du positionnement et du guidage ainsi que par le mode du propulsion du drone. En réalité la situation suivante est observée (Figure 1.8).

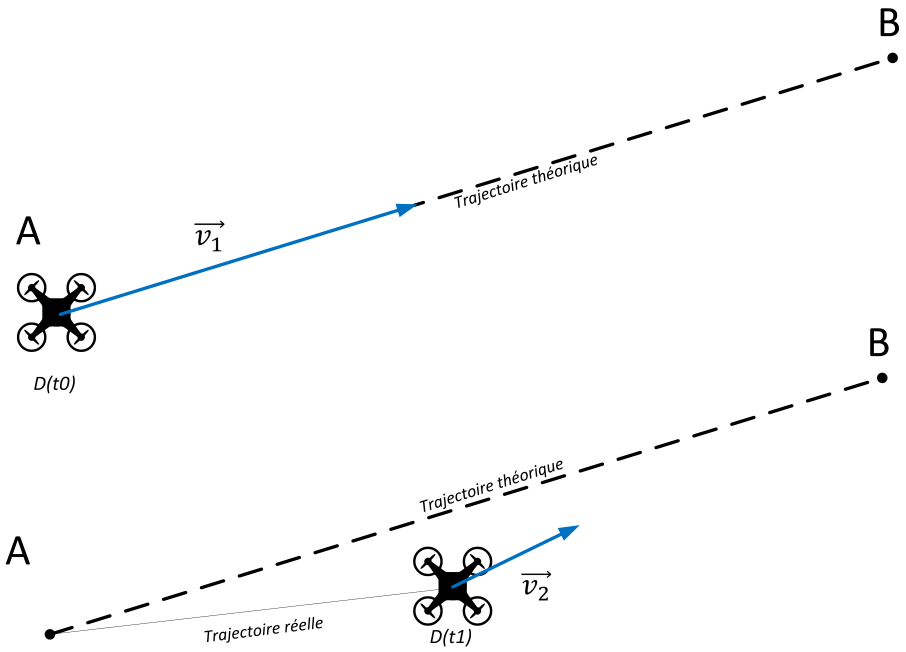


FIGURE 1.8 – Déplacement du drone selon une trajectoire théorique - Situation réelle.

Le drone ne se déplaçant plus sur le vecteur reliant les points A et B, il faut maintenant appliquer une correction sur le vecteur du déplacement du drone afin d'atteindre le point B. Dans un premier temps, il serait possible de simplement recalculer le vecteur entre le drone et le point B (V_2 sur la Figure 1.8).

En appliquant cette simple correction, la situation de la Figure 1.9 peut apparaître. La trajectoire du drone pourrait commencer à osciller de part et d'autre de la trajectoire à suivre pour plusieurs raisons :

— **Décalage temporel** : Entre le moment où la correction est calculée et le moment où elle est appliquée, le drone aura poursuivi son déplacement. Le vecteur de correction sera donc légèrement biaisé étant donné qu'il était calculé pour une position précédente du drone ;
— **Force extérieure appliquée sur le drone** : Le vent peut assez facilement impacter sur la trajectoire du drone, il se pourrait donc que la correction appliquée ne soit plus optimale.

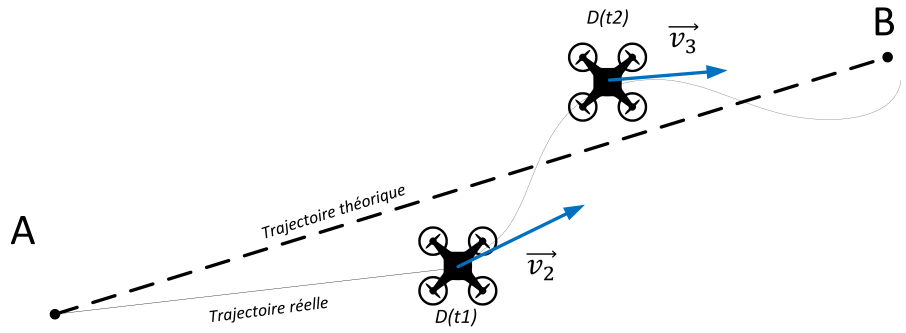


FIGURE 1.9 – Déplacement du drone selon une trajectoire théorique - Risque d'oscillation.

Pour éviter ces différents problèmes, un contrôleur plus élaboré qu'une simple différence doit être mis en place. Deux contrôleurs sont largement utilisés pour calculer des corrections : le **contrôleur PID** (Chapitre 1.5.1) et le **contrôleur MPC** (Chapitre 1.5.2).

1.5.1 Contrôleur PID

Cette section a pour but de présenter le régulateur PID [Babu et al., 2017]. Ce contrôleur est un mécanisme de boucle utilisant la rétroaction qui est largement utilisé dans les systèmes de contrôle industriels [22]. Le régulateur PID calcule en permanence la valeur d'erreur $e(t)$ étant la différence entre la consigne $r(t)$ et la mesure $y(t)$ et applique une correction basée sur les termes proportionnels, intégraux et dérivés selon la Figure 1.10 (désigné par P, I et D). La commande tente de minimiser l'erreur $e(t)$ à travers le temps en appliquant une correction $u(t)$ qui est une somme pondérée des termes P, I et D.

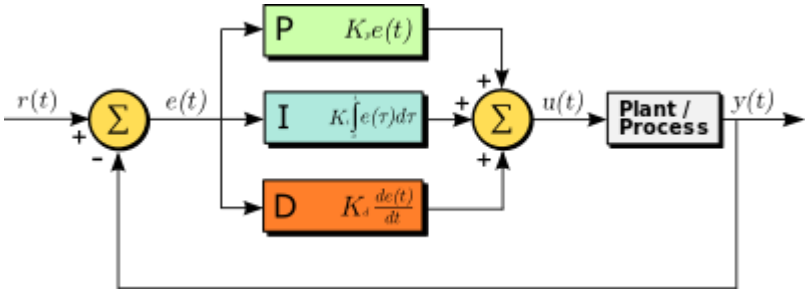


FIGURE 1.10 – Digramme contrôleur PID [23] ;

Le terme **P** est proportionnel à la valeur actuelle de l'erreur $e(t)$.

Le terme **I** tient compte des valeurs passées de l'erreur $e(t)$ et les intègre dans le temps. Par exemple, si il reste une erreur après l'application du terme proportionnel, le terme intégral va chercher à éliminer cette erreur en ajoutant un effet de contrôle dû à la valeur cumulative historique de l'erreur. Une fois l'erreur éliminée, le terme intégral cessera de croître. Ce terme permet de compenser le terme proportionnel lorsque l'erreur $e(t)$ diminue. En effet, plus $e(t)$ diminue, plus **P** diminue, mais **I** compense cela par l'effet intégral croissant. Le terme **I** permet d'éliminer l'erreur statique qui ne peut pas être corrigée par le terme **P** lorsque le système s'approche de sa consigne.

Le terme **D** permet d'anticiper les variations de la sortie. Lorsque le système s'approche de la consigne, ce terme freine le système en appliquant une action dans le sens opposé et permet ainsi une stabilisation plus rapide. Il permet ainsi d'éviter de trop fortes oscillations autour de la consigne.

L'équilibre entre ces différents termes se fait à l'aide de trois coefficients Kp, Ki, Ks.

Forme mathématique :

$$\begin{aligned} u(t) &= P + I + D \\ u(t) &= K_p e(t) + K_i \int_0^t e(\tau)\mathrm{d}\tau + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \end{aligned} \tag{1.6}$$

avec :

$$\begin{aligned} K_p &\geq 0 \\ K_i &\geq 0 \\ K_d &\geq 0 \end{aligned} \tag{1.7}$$

22. Source : `https://wikipedia.org/wiki/PID_controller`
23. Source : `https://wikipedia.org/wiki/PID_controller`

1.5.2 Contrôleur MPC

La commande prédictive ou Model Predictive Control (MPC) est une méthode avancée de l'automation qui permet de contrôler un processus tout en respectant un ensemble de contraintes [24]. Elle est en général utilisée pour commander des systèmes complexes où le simple régulateur PID est insuffisant. Le MPC a également la capacité d'anticiper les événements futurs et peut prendre des mesures de contrôle en conséquence. Les régulateurs PID n'ont pas cette capacité de prédiction.

L'idée de base d'un MPC est de prédire le comportement futur du système sur un horizon de temps fini et de calculer une commande optimale en respectant les contraintes données. A chaque temps une série de commandes optimales sont calculées pour un horizon temporel relativement court dans le futur. Seul la première commande est mise en œuvre, puis l'ensemble des calculs sont répétés à partir du nouvel état actuel ce qui donnera une nouvelle série de commandes optimales.

Pour appliquer un MPC il faut donc :
- Un modèle interne du processus ;
- Une fonction de coût ;
- Un algorithme d'optimisation de la fonction de coût ;

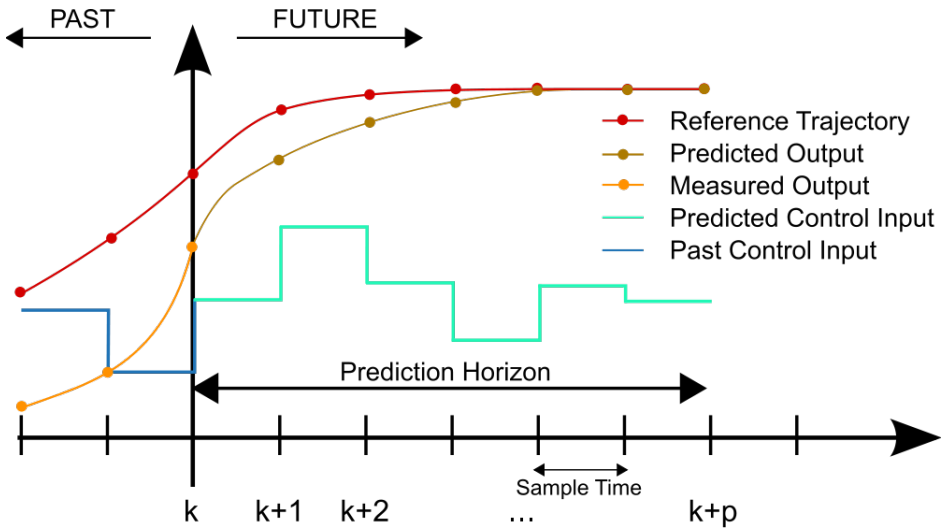


FIGURE 1.11 – MPC Schéma [25]

1.5.3 Solutions retenues

Comme expliqué dans les Chapitres précédents (1.5.1 et 1.5.2), le contrôleur MPC est plus compliqué que le contrôleur PID. C'est pour cela que le contrôleur PID sera favorisé dans un premier temps. Si ce contrôleur ne se montre pas suffisamment performant en effectuant des tests, alors le contrôleur MPC pourra être utilisé.

La position en temps réel du drone sera obtenue à l'aide du filtre de Kalman.

24. https://fr.wikipedia.org/wiki/Commande_predictive
25. Source : https://en.wikipedia.org/wiki/Model_predictive_control

Chapitre 2

Développement

La phase de développement est séparée en plusieurs blocs de validation selon les chapitres suivants. Chacun de ces chapitres a pour but de valider un processus "simple". Une fois l'ensemble de ces processus validés, une dernière phase consiste à l'intégration de l'ensemble des blocs en une seule application fonctionnelle. Les détails concernant l'implémentation sont disponibles en Annexe B.2.
Comme expliqué au Chapitre 1.3, l'ensemble du développement se fait pour le système Android en Java. Afin de communiquer et d'interéagir avec le drone, il est nécessaire d'utiliser le SDK fourni par DJI. La documentation relative à ce SDK est disponible sur le site de *DJI developer*.
Le développement est composé des étapes suivantes :

1. **Valider les déplacements du drone** : Valider et choisir les différents modes de déplacement possibles pour le drone (Chapitre 2.1) ;
2. **Récupérer les données du drone** : Valider la récupération des mesures des capteurs internes du drone (Chapitre 2.2) ;
3. **Récupérer les données de la station totale** : Valider la communication avec l'instrument (Chapitre 2.3) ;
4. **Déterminer la position du drone** : Calculer la position du drone à chaque instant à partir des mesures effectuées par la station totale et des données de vitesse renvoyées par le drone (Chapitre 2.4) ;
5. **Effectuer le guidage du drone** : A partir de la position actuelle du drone, calculer et appliquer les différentes corrections sur le drone afin de suivre un plan de vol donné (Chapitre 2.5) ;
6. **Capture des images** : Effectuer des prises de vues selon les paramètres de vol donnés (Chapitre 2.6) ;
7. **Interface graphique** : L'interface doit permettre à l'utilisateur d'effectuer les différents réglages pour l'automatistion du vol (Chapitre 2.7).

2.1 Déplacements basiques du drone

Ce premier bloc a pour but de valider les déplacements basiques du drone ainsi que de définir les différents systèmes de coordonnées gérés par les drone DJI.

2.1.1 Système de coordonnées DJI

Les déplacements du drone se réfèrent à un système de coordonnées défini par l'emplacement et l'orientation de ses axes. Deux systèmes de coordonnées différents sont disponibles : Body coordinate system et Ground coordinate system.

Body coordinate system

Ce système est lié au drone, les tois axes perpendiculaires sont définis à partir du centre des masses de l'appareil. L'axe **X** pointe vers l'avant du drone, l'axe **Y** vers la droite et l'axe **Z** vers le bas (Figure 2.1a).

Les rotations du drone sont également exprimées dans ce système de coordonnées. Pour les rotations, les axes X, Y et Z sont renommés **Roll**, **Pitch** et **Yaw** (Figure 2.1b).

(a) Axes (b) Rotation

FIGURE 2.1 – Body Coorinate System [1].

Ground coordinate system

Ce système est basé sur la convention **NED** qui définit les axes X, Y et Z positifs selon les directions **Nord**, **Est** et **Bas** (Figure 2.2). L'origine de ce système est fixé au point de décollage du drone et son orientation est faite grâce à la boussole interne du drone.

1. Source : https://developer.dji.com/mobile-sdk/documentation/introduction/flightController_concepts.html

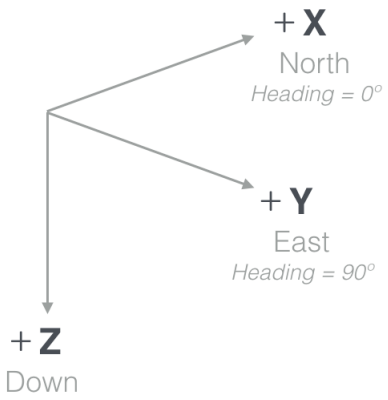


FIGURE 2.2 – Ground Coordinate System [2]

2.1.2 Implémentation

Le système utilisé pour la suite du travail est le "Ground Coordinate System". Ce système permet d'appliquer des corrections selon un système orienté au nord, comme le système MN95 (Système dans lequel les observations terrestre sont effectuées). L'utilisation du système "Body" oblige de re-calculer l'ensemble des corrections par rapport à la position actuel du drone.

En ce qui concerne le mode de déplacement ("Joystick virtuel" selon le chapitre 1.3.1), quatre paramètres sont à envoyer au drone pour effectuer un mouvement :

— Vitesse X ;
— Vitesse Y ;
— Vitesse Z ;
— Yaw (cap) ;

DJI fourni des applications de base sur son Github [3]. L'une d'elle (FPVDemo) est utilisée comme base pour les développements futurs. Celle-ci intègre déjà le SDK Mobile de DJI, un retour video en live sur l'écran ainsi que deux fonctions permettant d'effectuer une prise de vue ou une vidéo.

Trois fonctions supplémentaires sont implémentées sur la gauche de l'écran 2.3, à savoir :

— **Take off** : le drone effectue un décollage puis reste en stationnaire ;
— **Land** : le drone se pose à l'endroit où il se trouve ;
— **Move front** : le drone fait un déplacement en avant pendant X secondes ;

2. Source : Idem figure 2.1.
3. `https://github.com/DJI-Mobile-SDK-Tutorials`

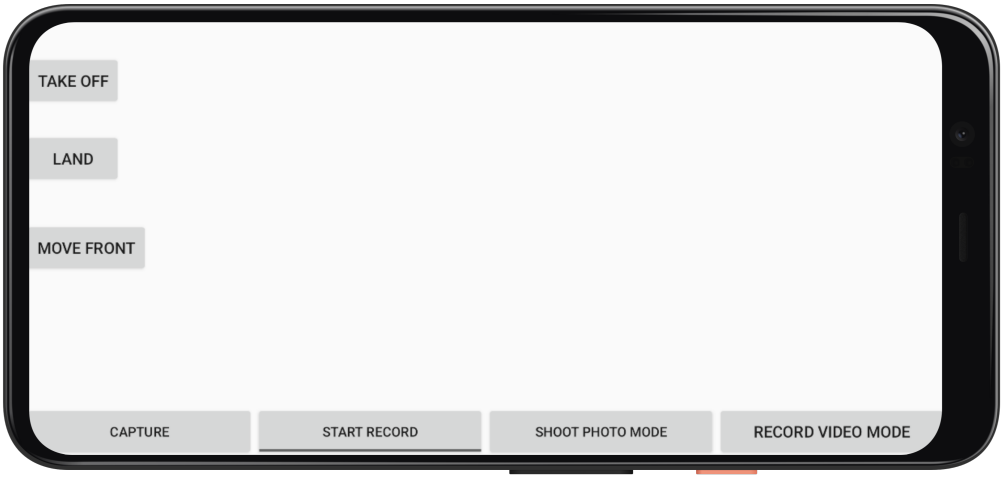


FIGURE 2.3 – Application - Déplacements basiques.

L'implémentation de ces trois fonctions permet de valider la partie concernant les déplacements applicables au drone. En maîtrisant un seul déplacement (Move front) l'ensemble des déplacements en trois dimensions sont validés étant donné que les fonctions appliquées sont identiques.

2.2 Récupération des données du drone

Ce deuxième bloc à pour but valider la partie concernant la récupération des données mesurées par les différents senseurs internes du drone. Les données suivantes peuvent être récupérées :

- **Vitesse X, Y et Z** : récupération de la vitesse selon les 3 axes en m/s dans le système de coordonnées NED (North-East-Down)
- **Attitude** : récupération des valeurs de Yaw, Pitch et Roll ;
- **Compas** : récupération du cap en degré ;
- **Position** : récupération de la longitude/latitude à partir du GNSS ainsi que l'altitude relative du drone par rapport au lieu de décollage à partir du baromètre ;
- **Altitude de décollage** : récupération de l'altitude du point de décollage ;
- **Ultrasons** : récupération de la hauteur par rapport au sol jusqu'à 8m ;

Afin de tester la récupération de ces données, une nouvelle couche a été ajoutée à l'application.

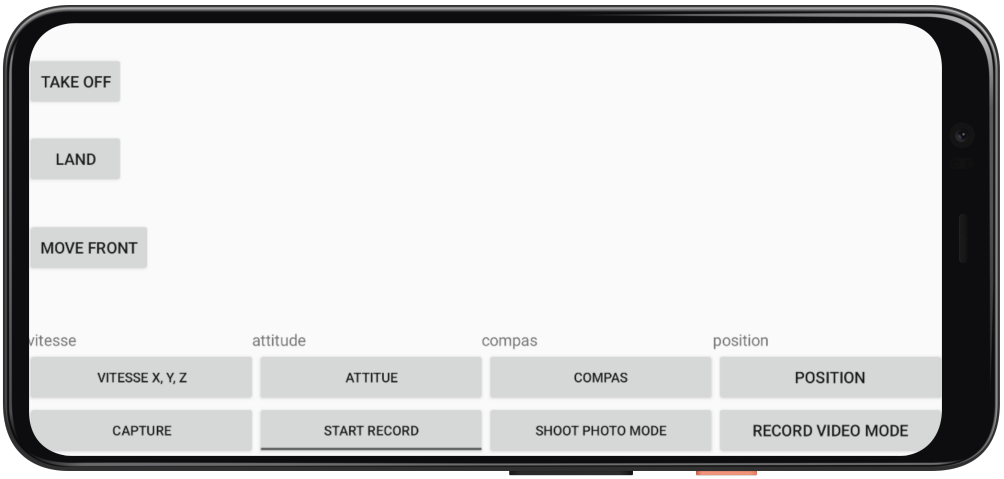


FIGURE 2.4 – Application - Récupération senseurs.

Une fois l'application lancée et connectée au drone, les différentes mesures sont affichées dans les cases correspondantes en live (Figure 2.4). Avec cette première étape la réception des données jusqu'au smartphone est validée.

Il est maintenant intéressant d'étudier la fréquence de réception ainsi que la synchronisation des différents senseurs. Pour cela un timestamp (heure Unix [4]) est associé à chaque mesure reçue, puis cette mesure est enregistrée dans un fichier .txt. Le timestamp correspond au moment où la mesure est reçue selon l'horloge du smartphone.

En ce qui concerne la fréquence de réception des mesures, le constructeur annonce une réception à 10 Hz. La Figure 2.5 représente la réception des vitesses X, Y et Z ainsi que la fréquence en Hz tout au long de la série de mesures. Au lancement de l'application la fréquence est de 1 Hz. Cela correspond à un état où le drone est allumé mais n'a reçu encore aucune commande. Dès que la première commande de déplacement est envoyée au drone (take off), la fréquence augmente jusqu'à 10 Hz. Ensuite la fréquence oscille en 9 Hz et 10 Hz jusqu'à la fin du vol.

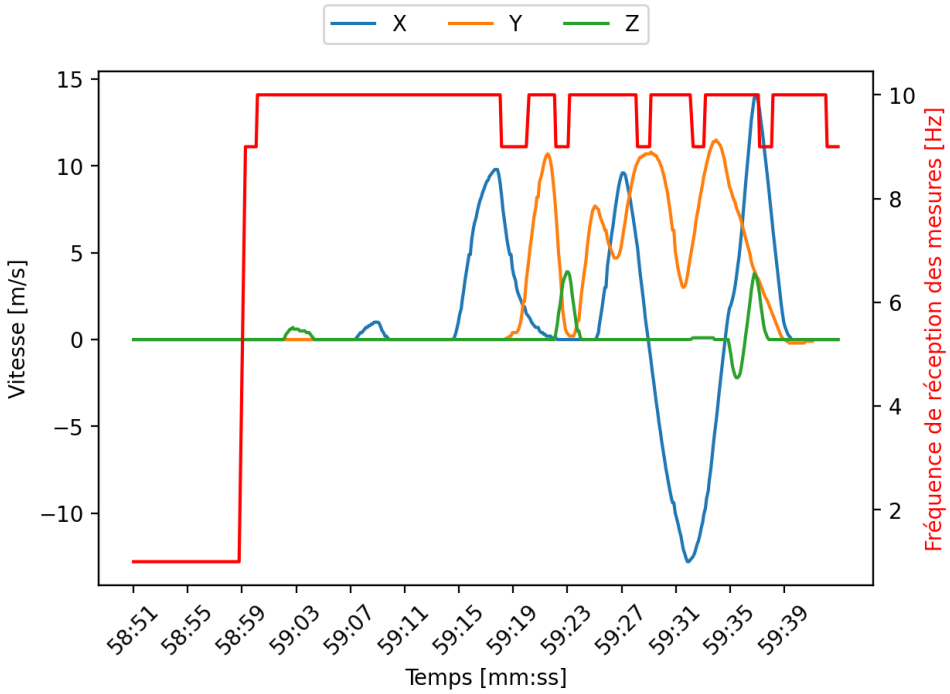


FIGURE 2.5 – Fréquence de récupération des senseurs, en rouge avec la barre d'échelle de droite.

La synchronisation des données reçues peut également être analysée. Pour ce faire une commande de déplacement X, Y égale est appliquée sur le drone. Si la synchronisation est parfaite, les courbes de déplacement X et Y doivent se superposer. Selon le graphique ci-dessous (Figure 2.6) les courbes de vitesse des axes X et Y se superposent.

4. Mesure du temps basée sur le nombre de secondes écoulées depuis le 1er janvier 1970 00 :00 :00 UTC

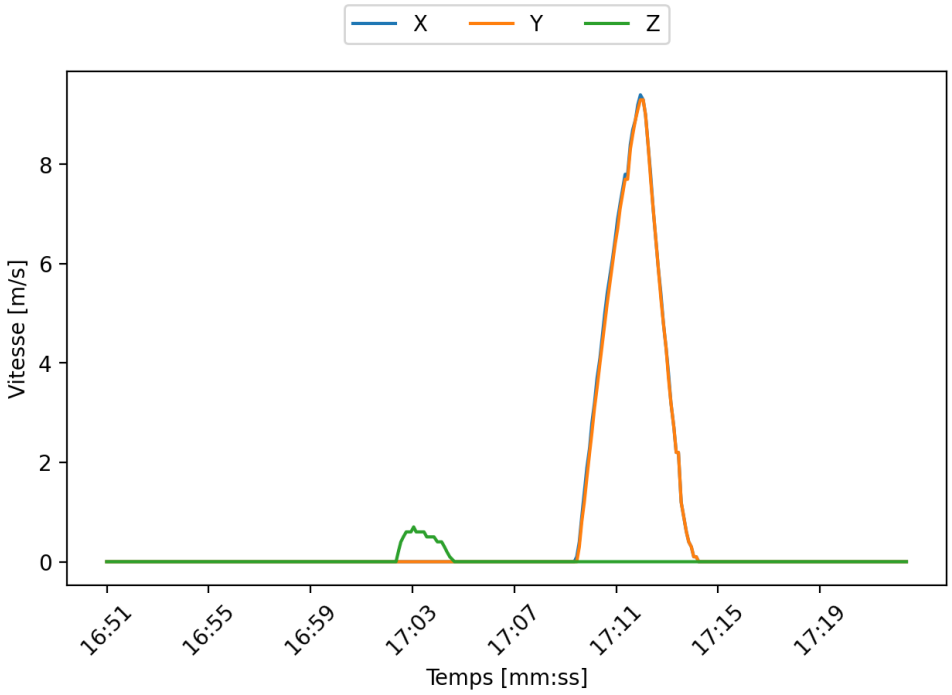


FIGURE 2.6 – Synchronisation déplacement X, Y.

En effectuant un zoom dans la série de mesures (Figure 2.7), un décalage d'un dixième de seconde entre le départ des mesures X et Y est présent. Ce décalage est acceptable pour l'application prévue.

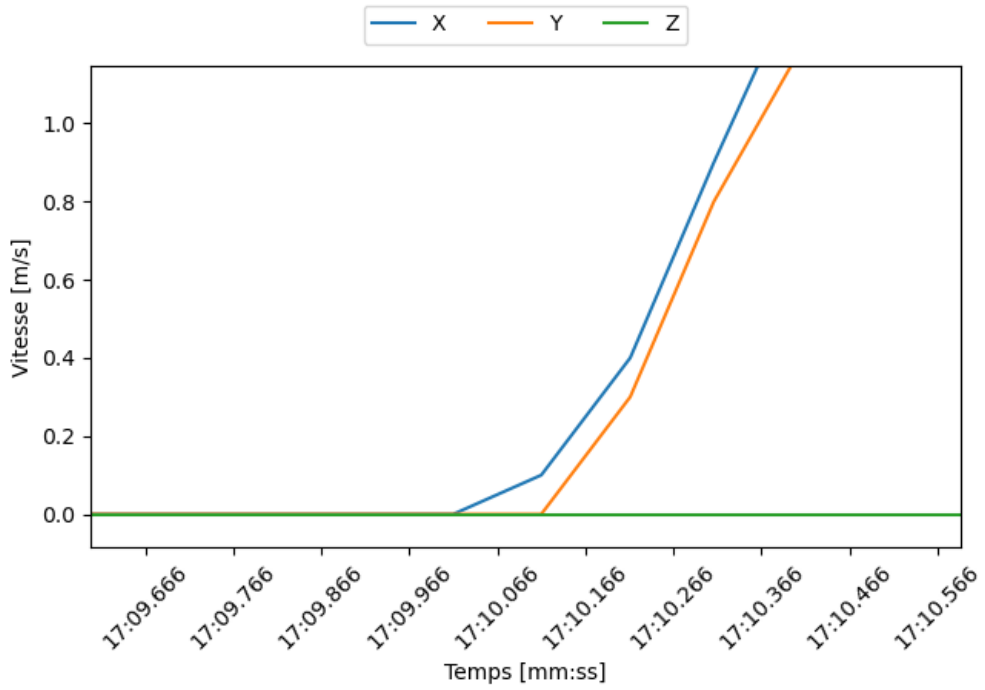


FIGURE 2.7 – Synchronisation déplacement X, Y zoom.

Même constat en analysant les données provenant de deux senseurs différents, à savoir le cap donné

par le compas et la vitesse en Z. Sur la Figure 2.8, la ligne verticale en traitillé représente le premier changement de cap. Simultanément à ce changement de cap, la commande de déplacement vertical à été envoyée au drone.

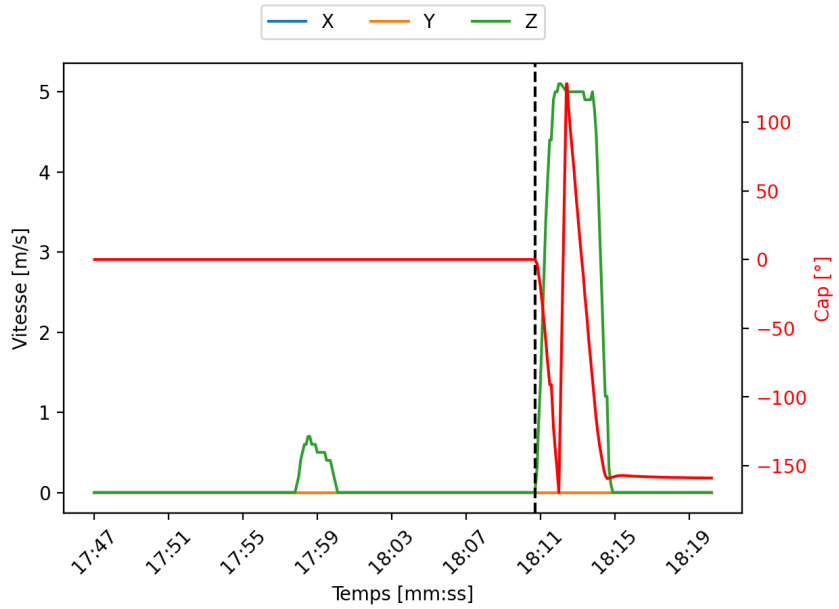


FIGURE 2.8 – Synchronisation déplacement Compas (en rouge) VS Z (en vert).

En zoomant dans la Figure 2.9, le déplacement en Z est bien synchronisé avec la référence du compas (trait-tillé). La réception des données des différents senseurs est donc bien synchronisé.

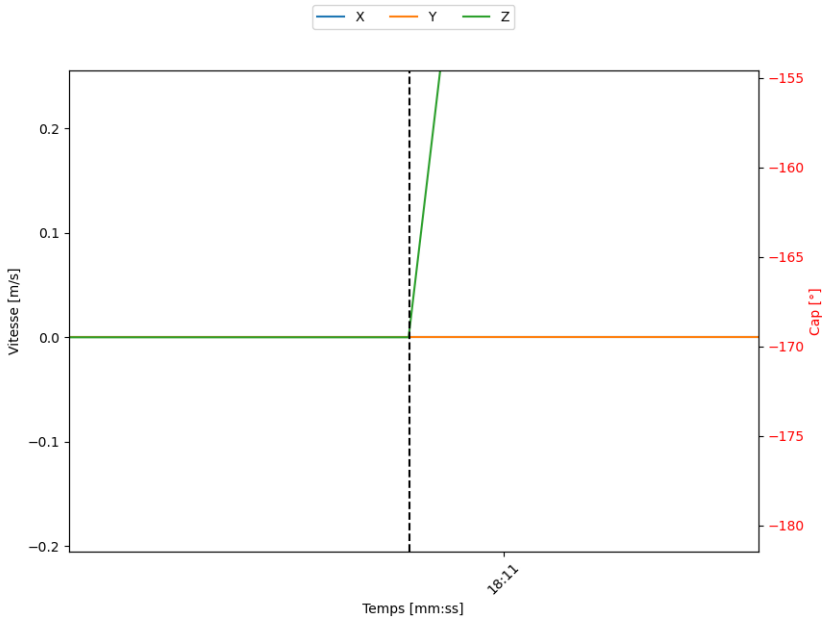


FIGURE 2.9 – Synchronisation déplacement Compas, Z zoom.

Suite à ces différents tests, la réception des données provenant du drone à environ 10 Hz ainsi que la synchronisation des différents senseurs est validée.

2.3 Communication avec la station totale

Ce chapitre à pour but d'étudier les communications entre une station totale et une application Android. Il est en effet nécessaire de pouvoir recevoir les coordonnées mesurées pas l'instrument directement sur l'application de guidage du drone. Les instruments utilisés sont ceux de la marque Leica, équipés d'une poignée radio ainsi que de la licence GeoCOM. Deux méthodes sont possibles pour recevoir les coordonnées depuis l'instrument :

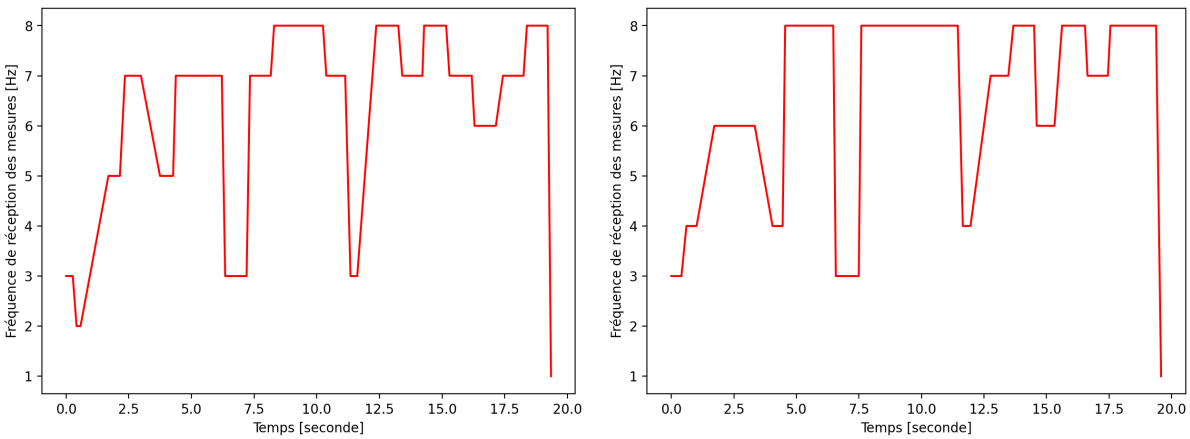
— **Sortie GSI** : Chaque fois que l'instrument effectue une mesure, les coordonnées sont diffusées par le port bluetooth de la poignée radio. Il est ensuite possible de récupérer ces données en "écoutant" le port bluetooth de l'instrument. Pour recevoir des coordonnées en continu, il faut paramétrer la station totale sur le mode "mesure en continu". Ainsi, une fois la mesure démarrée, les coordonnées peuvent être réceptionnées sur le smartphone en live ;
— **Port GeoCOM** : Le port GeoCOM permet de communiquer avec l'instrument, d'envoyer et de recevoir des données. L'avantage par rapport à la sortie GSI est que l'instrument peut totalement être contrôlé en lui envoyant des commandes. Il n'est donc plus nécessaire de démarrer les mesures physiquement sur l'instrument. Un autre avantage est le fait que les mesures effectuées par port GeoCOM ne sont pas stockées dans l'instrument. En effet, en mesurant en continu à 10 Hz cela surcharge vite le job sur l'instrument ;

La solution retenue est l'utilisation du port GeoCOM pour les différents avantages présentés ci-dessus. Cette solution nécessite néanmoins une implémentation un peu plus poussée étant donné qu'il ne suffit plus d'uniquement "écouter", mais qu'il faut également envoyer des données.

Les détails d'implémentation des communications avec la station totales sont disponible en Annexe B.2.4.

Leica annonce une possibilité de mesurer jusqu'à 10 Hz. Deux tests sont effectués pour valider cette fréquence, un en utilisant la sortie GSI et un autre utilisant le port GeoCOM. Pour les deux tests le prisme reste fixe durant la phase de mesures.

Une série de mesures en continu d'environ 20 secondes a été effectuée puis la fréquence en est calculée (Figure 2.10). Pour les deux modes de mesures la fréquence n'est pas stable et des baisses sont constatées à environ 7.5 sec et 12.5 sec. Ces pertes de fréquence sont courtes (moins d'une seconde) et sont certainement dues à un "recalibrage" du distancemètre de l'instrument. En moyenne il est quand même possible de récupérer des coordonnées à environ 6-7 Hz, ce qui est suffisant.



(a) Sortie GSI

(b) Port GeoCOM

FIGURE 2.10 – Fréquences de réceptions des mesures de la station totale.

2.4 Calcul de la trajectoire - filtre de Kalman

Le filtre de Kalman à pour but de déterminer la position du drone à chaque instant à partir des mesures terrestres et des mesures de vitesses renvoyées par le drone. Pour chaque époque le filtre fournira les coordonnées X,Y et Z estimées. L'élaboration du calcul de la trajectoire du drone en temps réel se décompose selon les étapes suivantes :

1. **Modélisation d'un filtre linéaire** : Filtre de Kalman linéaire pour calculer la position et la vitesse du drone (Chapitre 2.4.1) ;
2. **Modélisation d'un filtre non-linéaire** : Filtre de Kalman non-linéaire si la modélisation linéaire se montre insuffisante (Chapitre 2.4.2) ;
3. **Validation des modélisations à l'aide de données simulées** : Les deux modélisations sont validées à l'aide de données simulées en Python (Chapitre 2.4.3) ;
4. **Validation avec des données du simulateur DJI** : Les modélisations sont validées à l'aide de données provenant du simulateur de DJI (Chapitre 2.4.4) ;
5. **Prise en compte de l'orientation du drone et des bras de levier** : La position du prisme est recalculée pour coïncider avec le centre des masses du drone qui est la référence pour les mesures de vitesses (Chapitre 2.4.5) ;
6. **Comparaison des modélisations linéaires et non-linéaires avec des données réelles** : Le choix d'une des modélisation est effectué à partir de tests à l'aide de données réelles (Chapitre 2.4.6) ;
7. **Implémentation en Java et calcul en live** : Le processus développé est implémenté en Java pour pouvoir être exécuté en live (Chapitre 2.4.8).

2.4.1 Modélisation d'un filtre linéaire

Voici la modélisation du filtre de Kalman pour l'estimation en temps réel de la position 3D et des vitesses 3D d'un drone mesurées par une station totale selon les formules de base du Chapitre 1.4.1.

Vecteur d'état

Le vecteur d'état est défini par les composantes de la position et de la vitesse du drone :

$$\mathbf{x}_t = \begin{pmatrix} x_t \\ y_t \\ z_t \\ \dot{x}_t \\ \dot{y}_t \\ \dot{z}_t \end{pmatrix} \tag{2.1}$$

avec :

$$\begin{aligned} x_t &= x_{t-1} + \dot{x}_{t-1} * \Delta t \\ y_t &= y_{t-1} + \dot{y}_{t-1} * \Delta t \\ z_t &= z_{t-1} + \dot{z}_{t-1} * \Delta t \\ \dot{x}_t &= \dot{x}_{t-1} \\ \dot{y}_t &= \dot{y}_{t-1} \\ \dot{z}_t &= \dot{z}_{t-1} \end{aligned} \tag{2.2}$$

Modèle dynamique

Le modèle dynamique établi la relation qui permet de prédire le vecteur $\mathbf{x}_t$ à partir de l'état $\mathbf{x}_{t-1}$. Sous une forme linéaire avec la matrice :

$$\mathbf{F}_{t-1} = \begin{pmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.3}$$

Modèle stochastique de la dynamique

Il est également nécessaire de décrire mathématiquement la précision du modèle dynamique. Dans ce cas, l'incertitude du modèle est considérée comme un bruit blanc sur les accélérations du drone :

$$\mathbf{w}_t = \begin{pmatrix} \ddot{x}_t \\ \ddot{y}_t \\ \ddot{z}_t \end{pmatrix} \tag{2.4}$$

avec :

$$\mathbf{K}_{ww} = \begin{pmatrix} \sigma^2_{\ddot{x}_t} & 0 & 0 \\ 0 & \sigma^2_{\ddot{y}_t} & 0 \\ 0 & 0 & \sigma^2_{\ddot{z}_t} \end{pmatrix} \tag{2.5}$$

Et la matrice de diffusion de ce bruit blanc **w** dans le vecteur d'état :

$$\mathbf{G}_{t-1} = \begin{pmatrix} \frac{1}{2}\Delta t^2 & 0 & 0 \\ 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{pmatrix} \tag{2.6}$$

diffusion du bruit dans le vecteur d'état :

$$\begin{aligned} w_{x_t} &= \tfrac{1}{2}\Delta t^2 * \ddot{x}_t \\ w_{y_t} &= \tfrac{1}{2}\Delta t^2 * \ddot{y}_t \\ w_{z_t} &= \tfrac{1}{2}\Delta t^2 * \ddot{z}_t \\ w_{\dot{x}_t} &= \Delta t * \ddot{x}_t \\ w_{\dot{y}_t} &= \Delta t * \ddot{y}_t \\ w_{\dot{z}_t} &= \Delta t * \ddot{z}_t \end{aligned} \tag{2.7}$$

Modèle des observations

Le vecteur des observations :

$$\mathbf{l}_t = \begin{pmatrix} x_{THEO_t} \\ y_{THEO_t} \\ z_{THEO_t} \\ \dot{x}_{DRONE_t} \\ \dot{y}_{DRONE_t} \\ \dot{z}_{DRONE_t} \end{pmatrix} \tag{2.8}$$

qui est relié au vecteur d'état par la matrice :

$$\mathbf{A}_t = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.9}$$

Le modèle stochastique est défini par la matrice de variance-covariance des observations :

$$\mathbf{K}_{ll} = \begin{pmatrix} \sigma^2_{x_{THEO}} & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{y_{THEO}} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{z_{THEO}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{\dot{x}_{DRONE}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{\dot{y}_{DRONE}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^2_{\dot{z}_{DRONE}} \end{pmatrix} \tag{2.10}$$

2.4.2 Modélisation d'un filtre non linéaire

Cette deuxième modélisation est un dérivé de la modélisation linéaire présentée au Chapitre 2.4.1. Cette fois-ci les vitesses ne sont pas traitées comme des vecteurs $\dot{x}_t, \dot{y}_t, \dot{z}_t$ mais comme la norme de la vitesse en 2D $\|v_{xy}\|$ et un gisement φ. Cette modélisation a pour but de pouvoir intégrer des bruits de modèle sur l'orientation du vecteur de vitesse. Ainsi des biais dans les vecteurs de vitesses données par le drone pourraient directement être compensés par le modèle.

Vecteur d'état

Le vecteur d'état est défini par les composantes de la position et de la vitesse du drone ainsi qu'un angle β qui correspond à un biais sur φ :

$$\mathbf{x}_t = \begin{pmatrix} x_t \\ y_t \\ z_t \\ v_{xy} \\ \varphi \\ \beta \\ \dot{z}_t \end{pmatrix} \tag{2.11}$$

avec :

$$\begin{aligned} x_t &= x_{t-1} + v_{xy_{t-1}} * sin(\varphi_{t-1} + \beta_{t-1}) * \Delta t \\ y_t &= y_{t-1} + v_{xy_{t-1}} * cos(\varphi_{t-1} + \beta_{t-1}) * \Delta t \\ z_t &= z_{t-1} + \dot{z}_{t-1} * \Delta t \\ v_{xy} &= v_{xy_{t-1}} : \text{(vitesse 2D)} \\ \varphi &= \varphi_{t-1} \\ \beta &= \beta_{t-1} \\ \dot{z}_t &= \dot{z}_{t-1} \end{aligned} \tag{2.12}$$

Modèle dynamique

Le modèle dynamique établi la relation qui permet de prédire le vecteur $\mathbf{x}_t$ à partir de l'état $\mathbf{x}_{t-1}$. Sous une forme linéaire avec la matrice jacobienne :

$$\mathbf{F}_{t-1} = \begin{pmatrix} 1 & 0 & 0 & sin(\varphi+\beta)*\Delta t & v*cos(\varphi+\beta)*\Delta t & v*cos(\varphi+\beta)*\Delta t & 0 \\ 0 & 1 & 0 & cos(\varphi+\beta)*\Delta t & -v*sin(\varphi+\beta)*\Delta t & -v*sin(\varphi+\beta)*\Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.13}$$

avec :

$$\begin{aligned} \varphi &= \varphi_{t-1} \\ \beta &= \beta_{t-1} \\ v &= v_{xy_{t-1}} \end{aligned} \tag{2.14}$$

Modèle stochastique de la dynamique

Il est également nécessaire de décrire mathématiquement la précision du modèle dynamique. Dans ce cas, l'incertitude du modèle est considéré comme un bruit blanc sur les accélérations du drone :

$$\mathbf{w}_t = \begin{pmatrix} \dot{v}_{xy} \\ \ddot{\beta}_t \\ \ddot{z}_t \end{pmatrix} \tag{2.15}$$

$$\begin{aligned} \dot{v}_{xy} &: \text{accélération en 2D} \\ \ddot{\beta}_t &: \text{accélération du biais } \beta \\ \ddot{z}_t &: \text{accélération en Z} \end{aligned} \tag{2.16}$$

avec :

$$\mathbf{K}_{ww} = \begin{pmatrix} \sigma^2_{\dot{v}_{xy}} & 0 & 0 \\ 0 & \sigma^2_{\ddot{\beta}_t} & 0 \\ 0 & 0 & \sigma^2_{\ddot{z}_t} \end{pmatrix} \tag{2.17}$$

Et la matrice de diffusion de ce bruit blanc **w** dans le vecteur d'état :

$$\mathbf{G}_{t-1} = \begin{pmatrix} \frac{1}{2} * sin(\varphi_{t-1} + \beta_{t-1}) * \Delta t^2 & 0 & 0 \\ 0 & \frac{1}{2} * cos(\varphi_{t-1} + \beta_{t-1}) * \Delta t^2 & 0 \\ 0 & 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 & 0 \\ 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{pmatrix} \tag{2.18}$$

diffusion du bruit dans le vecteur d'état :

$$\begin{aligned} w_{x_t} &= \tfrac{1}{2} * sin(\varphi_{t-1} + \beta_{t-1}) * \Delta t^2 * \dot{v}_{xy} \\ w_{y_t} &= \tfrac{1}{2} * cos(\varphi_{t-1} + \beta_{t-1}) * \Delta t^2 * \dot{v}_{xy} \\ w_{z_t} &= \tfrac{1}{2}\Delta t^2 * \ddot{z}_t \\ w_{v_{xy}} &= \Delta t * \dot{v}_{xy} \\ w_{\varphi} &= \tfrac{1}{2}\Delta t^2 * \ddot{\beta}_t \\ w_{\beta} &= \Delta t * \ddot{\beta}_t \\ w_{v_z} &= \Delta t * \ddot{z}_t \end{aligned} \tag{2.19}$$

Modèle des observations

Le vecteur des observations :

$$\mathbf{l}_t = \begin{pmatrix} x_{THEO_t} \\ y_{THEO_t} \\ z_{THEO_t} \\ v_{xy_{DRONE_t}} \\ \varphi_{DRONE_t} \\ \dot{z}_{DRONE_t} \end{pmatrix} \tag{2.20}$$

qui est relié au vecteur d'état par la matrice :

$$\mathbf{A}_t = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.21}$$

Le modèle stochastique est défini par la matrice de variance-covariance des observations :

$$\mathbf{K_{ll}} = \begin{pmatrix} \sigma^2_{x_{THEO}} & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{y_{THEO}} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{z_{THEO}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{v_{xy_{DRONE}}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{\varphi_{DRONE}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^2_{\dot{z}_{DRONE}} \end{pmatrix} \tag{2.22}$$

2.4.3 Validation des modélisations à l'aide de données simulées

Le but de ce premier test est de valider la bonne modélisation du filtre de Kalman en utilisant des données 3D simulées par un script Python. L'utilisation de données simulées permet de créer un jeu de données bien connu. Il est alors possible de créer des données "parfaites" ou des données bruitées.

Dans un premier temps la trajectoire suivante est créée, elle servira de trajectoire "vraie" (Figure 2.11).

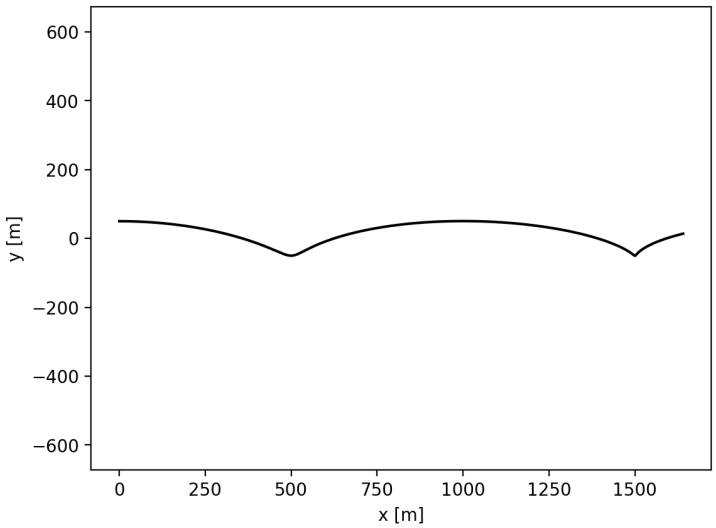


FIGURE 2.11 – Trajectoire simulée avec un script Python.

A partir de cette trajectoire des positions relevées à la station totale ainsi que des vitesses mesurées par le drone sont simulées (Figure 2.12). Un bruit est ajouté à ces mesures.

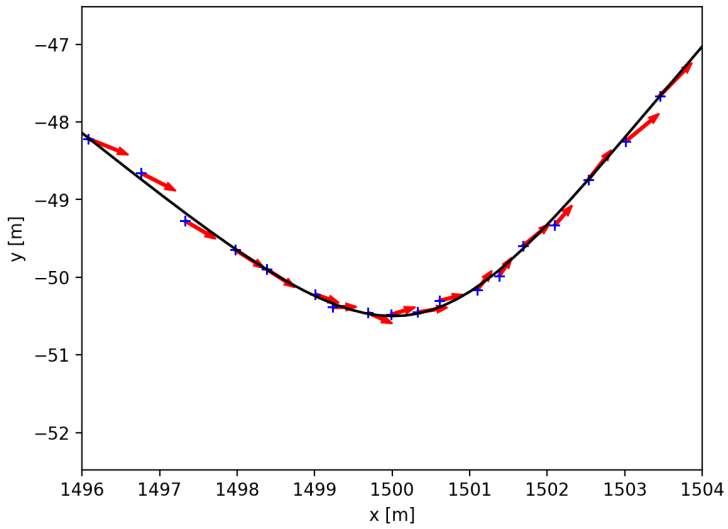
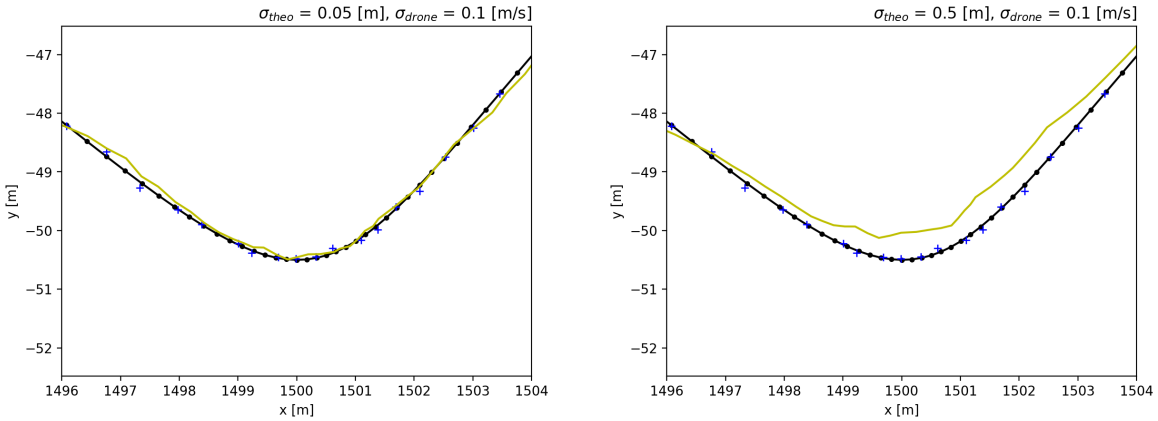


FIGURE 2.12 – Données de position (bleu) et de vitesse (vecteurs rouges) simulées à partir de la trajectoire "vraie" ; Zoom dans la Figure 2.11.

Il est également important d'ajuster les différents paramètres du filtre lors de ces premiers tests pour obtenir le meilleures résultat. La trajectoire peut ensuite être calculée à partir des observations simulées. Les paramètres à ajuster sont les suivants :

— **Précision des observations** (Équation 2.22) : La précision des mesures terrestres ainsi que des mesures de vitesse du drone impactent fortement le résultat final. Si une observation n'a pas assez de poids, elle ne sera quasiment pas prise en compte pour le calcul de la trajectoire finale. Au contraire, si une observation à trop de poids, la trajectoire finale sera trop contrainte par ces observations. La précision des observations est ajustée de manière à donner plus de poids au observation terrestre qui sont plus précises. L'ajustement de ces paramètres se fait essentiellement par des simulations.
Pour la modélisation linéaire, la précision des observations terrestre est fixée à **5 cm** et la précision des observations de vitesse à **0.1 m/s** (Figure 2.13a). Si par exemple il n'y a pas assez de poids sur les mesures terrestres, la trajectoire finale est plus influencée par les mesures de vitesses du drone (Figure 2.13b). Dans ce cas, la trajectoire calculée ne suit plus la trajectoire "vraie" étant donné que les mesures de vitesses du drone sont bruitées.
Pour la modélisation non-linéaire, la précision des observations terrestres est fixée à **5 cm**, la précision des observations de vitesse à **0.1 m/s** et la précision du gisement à **5°**. Des phénomènes similaires à ceux de la modélisation linéaire sont observables si l'ajustement du poids des observations n'est pas optimal ;

— **Précision du modèle dynamique** (Équation 2.4) : Ce paramètre va impacter sur l'inertie du filtre. Si le filtre à trop d'inertie, les observations ne sont pas assez prises en compte. Au contraire, si le filtre n'a pas assez d'inertie, la trajectoire finale est trop corrélée aux observations. Ce paramètre permet donc de plus ou moins "lisser" la trajectoire finale.
Pour la modélisation linéaire, ce paramètre est défini par rapport à un bruit sur les accélérations en X, Y et Z. Ce bruit est fixé à **3 m**/s^2. Par exemple, sur la Figure 2.14 un bruit très faible est donné sur les accélérations (0.1 m/s^2) ce qui implique une inertie importante. La trajectoire à alors de la peine à effectuer le changement de trajectoire rapidement et "dépasse" la trajectoire vraie.
Pour la modélisation non-linéaire, le bruit est ajouté sur l'accélération XY, Z à **3 m**/s^2 et sur l'accélération du biais β à **2 °**/s^2.



(a) Paramètres retenus

(b) Pas assez de poids sur les mesures terrestres

FIGURE 2.13 – Trajectoire calculée par le filtre de Kalman (jaune) à partir des positions et vitesses ; Modélisation linéaire.

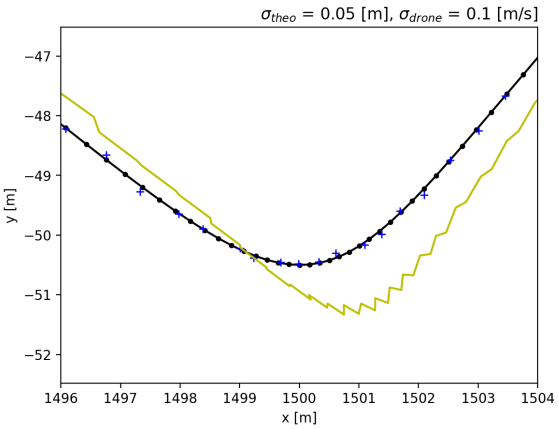


FIGURE 2.14 – Trajectoire calculée par le filtre de Kalman (jaune) à partir des positions et vitesses ; Inertie trop importante du filtre ; Modélisation linéaire.

Suite à ce premier bloc de test, la modélisation du filtre de Kalman (linéaire et non-linéaire) est validée.

2.4.4 Validation des modélisations avec des données du simulateur DJI

Afin de faire un pas de plus vers la réalité, une seconde partie de tests est effectués à partir des données du simulateur DJI. Celui-ci permet d'effectuer tout type de vol (manuel ou guidage à l'aide de SDK) et renvoie les mesures simulées des différents capteurs internes du drone (vitesse, position, orientation, hauteur de vol, attitude). Ces différentes mesures sont récupérées par un smartphone via le SDK de DJI (Chapitre 2.2). Le simulateur qui tourne sur un PC permet également d'enregister une "trace" du drone dans un système local appelé "World" (sous le format : temps/X/Y/H).

Le filtre de Kalman est calculé à partir des vitesses renvoyées par le drone et position enregistrée par le simulateur. Ces positions sont considérées comme des observations terrestres à la station totale. Des incohérences sont visibles étant donné que les données du simulateur et du drone ne sont pas bien synchronisées temporellement (Figure 2.15).

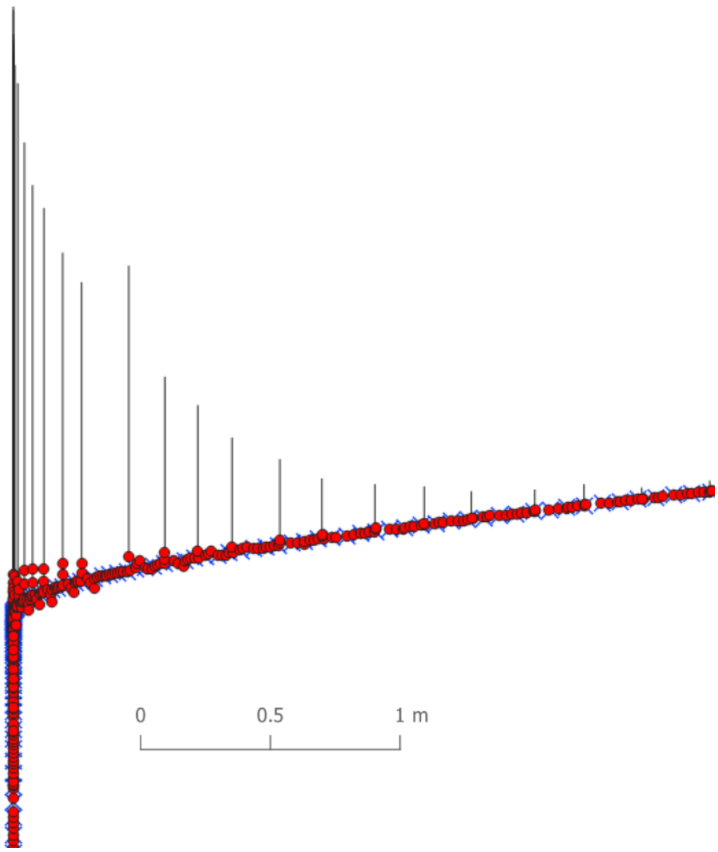
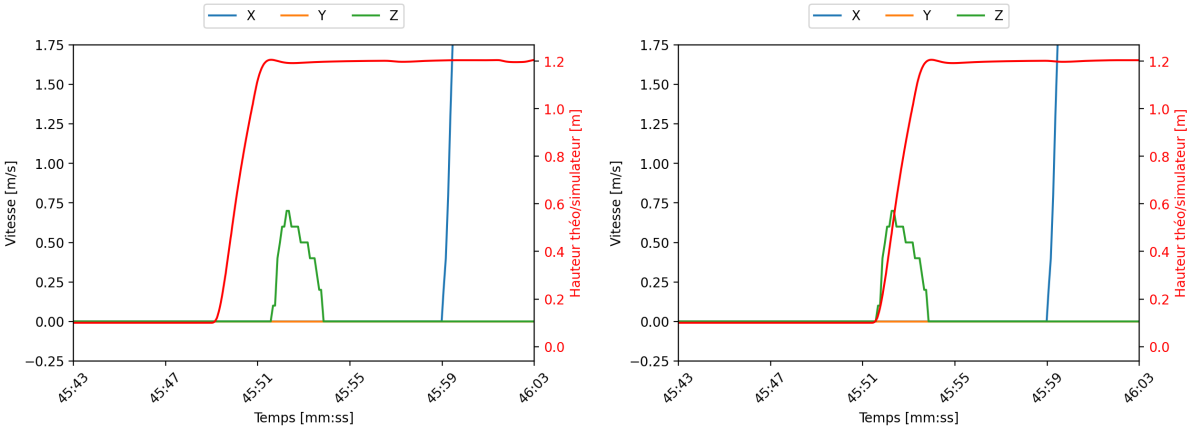


FIGURE 2.15 – Décalage temporel entre les données du simulateur et les données du drone. Les vecteurs de vitesse (ligne noir) sont en "retard".

Pour quantifier ce décalage, les données de vitesse X,Y,Z provenant du drone et l'information de hauteur de vol provenant du simulateur sont représentées sur un même graphique. Si la synchronisation des deux jeux de données est parfaite, lorsque le drone effectue un décollage, la vitesse en Z ainsi que la hauteur de vol devraient accroître simultanément. Cependant, sur le graphique de gauche (Figure 2.16a) un décalage d'environ 2.4 [sec] est bien présent entre la vitesse en Z (vert) et la hauteur de vol (rouge).

Une correction est ensuite appliquée sur les données provenant du simulateur (les données provenant du drone sont considérées comme fixe), ce qui produit un graphique cohérent avec des données bien synchronisées (Figure 2.16b).

Cette valeur de correction n'est pas à considérer comme unique pour tous les utilisateurs. Le décalage de temps va en effet dépendre du PC et du smartphone utilisé. Il y a donc une correction à calculer chaque fois que le matériel utilisé (PC/smartphone) change.



(a) Sans correction - décalage

(b) Avec correction - plus de décalage

FIGURE 2.16 – Synchronisation des données provenant du simulateur (hauteur de vol en rouge) et des données du drone.

Une fois cette correction connue, il est possible de recalculer la trajectoire du drone. Le résultat de la Figure 2.17 est obtenu. La nouvelle trajectoire (en orange) suit maintenant la trajectoire théorique (croix en bleu).

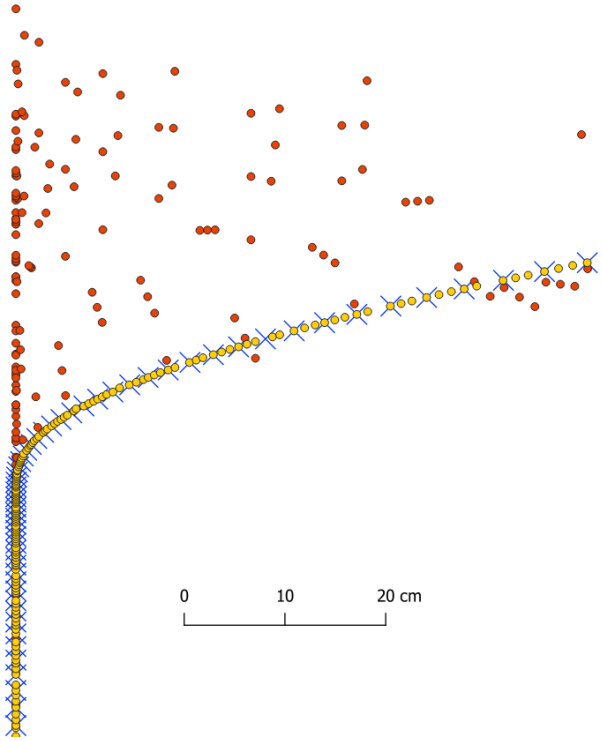


FIGURE 2.17 – Calcul de trajectoire avec données synchronisées. Rouge : données non synchronisées ; Organe : données synchronisées ; Croix bleu : trajectoire "vraie".

Des résultats identiques sont obtenus pour la modélisation linéaire ou non-linéaire. Suite à ce deuxième bloc de test, la modélisation du filtre de Kalman (linéaire et non-linéaire) à partir des données du simulateur de DJI est validée. Il sera par la suite important de pouvoir utiliser le simulateur afin de valider le système de guidage du drone dans un environnement virtuel.

2.4.5 Prise en compte de l'orientation du drone et des bras de levier

Les tests des Chapitres suivant (Chapitre 2.4.6 et 2.4.8), sont effectués avec des données issues de vol réels. Il devient alors important de prendre en compte le bras de levier entre le prisme et le centre du drone. Le prisme positionné sur le drone n'étant pas au même emplacement que le point de référence pour le calcul des vitesses (centre de gravité du drone), une correction peut être appliquée pour faire coïncider ces deux points. Dans les tests précédemment effectués, seul des translations étaient appliquées sur le drone. Dans ce cas, les vecteurs issus des mesures terrestres et les vecteurs issus du drone restent cohérents (Figure 2.18).

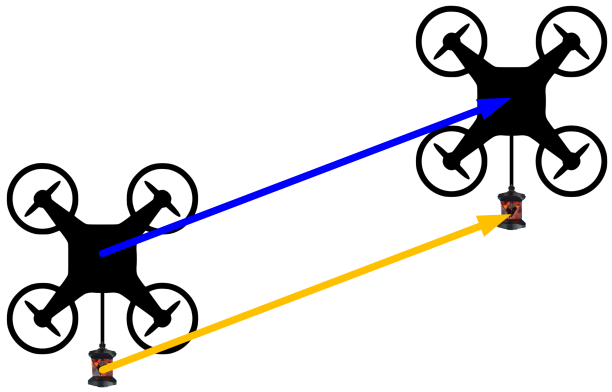


FIGURE 2.18 – Impact d'un bras de levier entre le prisme et le centre du drone pour des translations.

Par contre, si le drone effectue un changement de cap (rotation autour de son axe Z) pendant sont déplacement, les vecteurs issus des mesures terrestres ne sont plus cohérents avec les vecteurs issu dus drone (Figure 2.19).

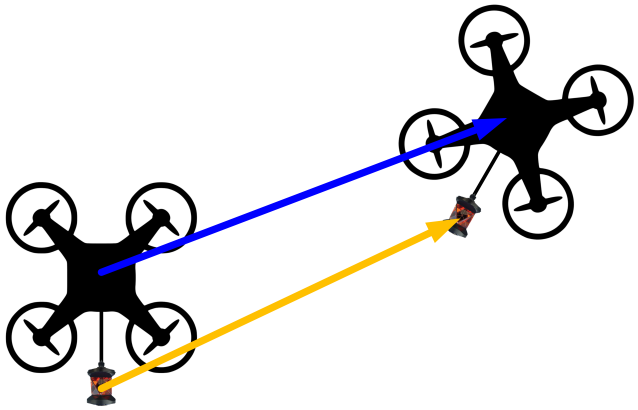


FIGURE 2.19 – Impact d'un bras de levier entre le prisme et le centre du drone lorsqu'il y a un changement de cap.

Le même phénomène est visible si le drone est observé de profil. Avec des vecteurs cohérents lorsque de simples translations sont appliquées, et des vecteurs incohérents lorsque des rotations sont présentes (Figure 2.20, 2.21).

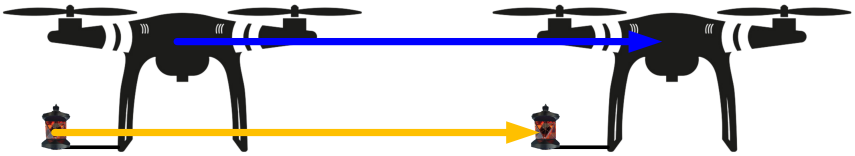


FIGURE 2.20 – Impact d'un bras de levier entre le prisme et le centre du drone lorsqu'il y n'a pas de changement d'orientation Roll ou Pitch - Vue de profil.

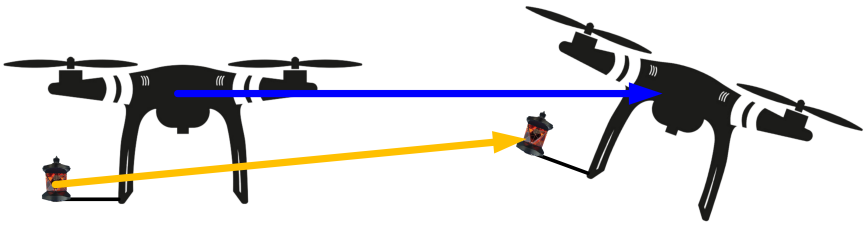


FIGURE 2.21 – Impact d'un bras de levier entre le prisme et le centre du drone lorsqu'il y a un changement d'orientation Roll ou Pitch - Vue de profil.

De telles rotations du drone peuvent notamment apparaître lorsque le cap du drone varie, qu'une forte accélération est demandée ou lorsque le drone doit lutter contre un vent contraire violent.

Pour éviter ces différents problèmes, un bras de levier est calculé entre le prisme et le centre du drone. Les mesures terrestres seront transformées pour coïncider avec le centre du drone avant d'être utilisées dans le filtre de Kalman.

La première étape consiste à mesurer le bras de levier entre le prisme et le centre du drone dans le système de coordonnées du drone (centré sur le centre des masses du drone ; Axe X vers l'avant ; Système "Body" selon le Chapitre 2.1.1). Pour le drone DJI Matrice 210 le vecteur suivant est mesuré en centimètre :

$$Levier_{drone-prisme} = \begin{pmatrix} 1.0 \\ 0.0 \\ -42.5 \end{pmatrix} \tag{2.23}$$

Il est donc possible de ramener une mesure sur le prisme au centre du drone en appliquant l'opposé du vecteur $Levier_{drone-prisme}$. Cette simple transformation est uniquement valable si aucune rotation n'est effectuée sur les trois axes du drone. Pour prendre en compte ces rotations, une rotation 3D doit être appliquée sur le vecteur $Levier_{drone-prisme}$ en tenant compte des mesures de Roll, Pitch et Yaw du drone.

La matrice de rotation suivante doit être calculée pour chaque état selon la convention Cardan (x-y-z) :

$$\begin{aligned}\mathbf{R}_{\mathbf{I}}^{\mathbf{II}}(\alpha,\beta,\gamma) &= \mathbf{R}_z(\gamma)\cdot\mathbf{R}_y(\beta)\cdot\mathbf{R}_x(\alpha)\\ &= \begin{pmatrix}\cos\gamma & \sin\gamma & 0\\ -\sin\gamma & \cos\gamma & 0\\ 0 & 0 & 1\end{pmatrix}\cdot\begin{pmatrix}\cos\beta & 0 & -\sin\beta\\ 0 & 1 & 0\\ \sin\beta & 0 & \cos\beta\end{pmatrix}\cdot\begin{pmatrix}1 & 0 & 0\\ 0 & \cos\alpha & \sin\alpha\\ 0 & -\sin\alpha & \cos\alpha\end{pmatrix}\\ &= \begin{pmatrix}\cos\gamma\cos\beta & \cos\gamma\sin\beta\sin\alpha+\sin\gamma\cos\alpha & -\cos\gamma\sin\beta\cos\alpha+\sin\gamma\sin\alpha\\ -\sin\gamma\cos\beta & -\sin\gamma\sin\beta\sin\alpha+\cos\gamma cos\alpha & -\sin\gamma\sin\beta\cos\alpha+\cos\gamma\sin\alpha\\ \sin\beta & -\cos\beta\sin\alpha & \cos\beta\cos\alpha\end{pmatrix}\end{aligned} \tag{2.24}$$

avec :

— α : le roulis "roll" ;
— β : le tangage "pitch" ;
— γ : le lacet "yaw" ;

Pour chaque époque une nouveau bras de levier est calculé :

$$\mathbf{b}_t = \mathbf{R}_{\mathbf{I}\ t}^{\mathbf{II}}\cdot Levier_{drone-prisme} \tag{2.25}$$

Puis les mesures terrestres sont corrigées :

$$\mathbf{l}_{corr_t} = \mathbf{l}_t - \mathbf{b}_t \tag{2.26}$$

En utilisant les observations corrigées pour calculer le filtre de Kalman, il est maintenant possible d'effectuer des rotations avec le drone tout en gardant de la cohérence entre les mesures de la station totale et du drone.

2.4.6 Comparaison des modélisation linéaire et non-linéaire avec des données réelles

Dans les deux phases de tests précédents, le filtre de Kalman linéaire et non-linéaire donnaient des résultats très proches, étant donné que les observations étaient "parfaites". Les tests avec des données réelles permettent de détecter des phénomènes qui n'étaient pas visibles sur les tests précédents. Pour cette phase de test les deux types de modélisation de filtre sont étudiés séparément pour en ressortir les avantages et inconvénients. Une base de tests en MN95 dans une zone dégagée est créée et permettra d'effectuer l'ensemble des tests réels.

Les jeux de données sont issus d'un vol avec un drone DJI Phantom 4. Le drone est équipé d'un mini prisme 360° ce qui permet le suivi avec un leica TS15 (Figure 2.22).

FIGURE 2.22 – Montage du mini primse 360° sur le DJI Phantom 4.

Plusieurs configurations de plan de vol sont également enregistrées afin de pouvoir tester différents paramètres par la suite.

Filtre de Kalman linéaire

Après avoir calculé des trajectoires sur plusieurs configurations de vol, un premier constat ressort : l'orientation des vecteurs de vitesses mesurés par le drone n'est pas cohérente avec les mesures effectuées par la station totale (Figure 2.23).

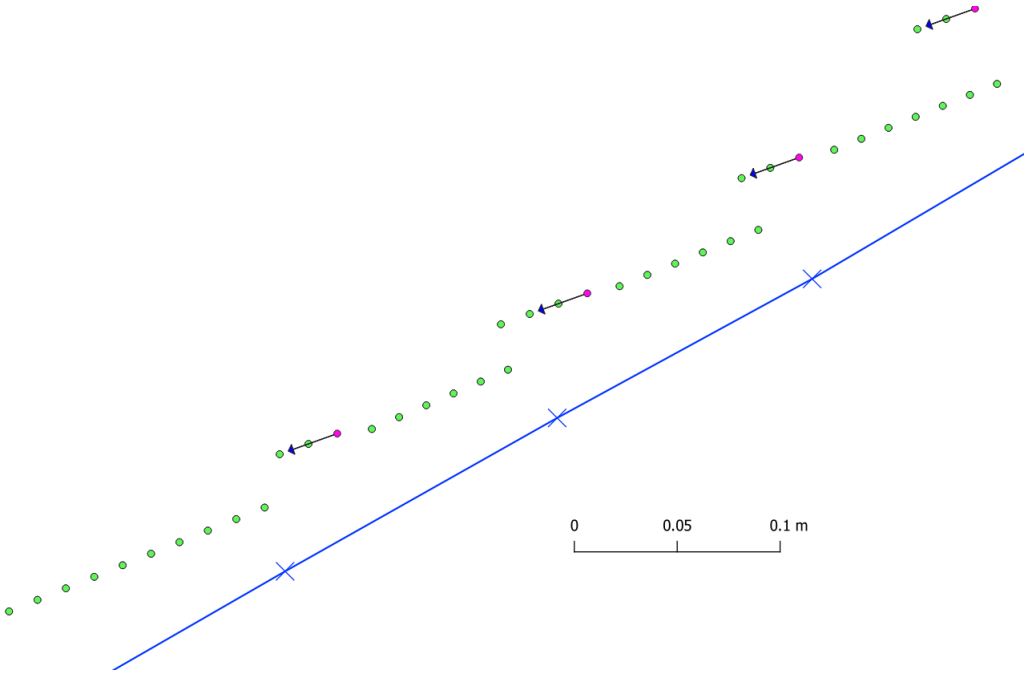


FIGURE 2.23 – Divergence entre la direction des vecteurs de vitesses du drone (en vert) et la trajectoire relevée à la station totale (en bleu).

Ce décalage est l'addition de plusieurs éléments :

— **Problème d'orientation des systèmes** : Des coordonnées MN95 sont assemblées à des vecteurs de vitesses pour calculer la trajectoire optimale. Ces deux observations sont faites dans des repères différents. Pour les mesures à la station totale, dans le repère MN95. Pour les mesures du drone, les vecteurs de vitesses sont calculés dans un repère local fixé au point de décollage et orienté au Nord (Système NED selon le Chapitre 2.1.1). L'orientation du système se faisant à l'aide de la boussole du drone, le repère local du drone est orienté sur le Nord magnétique. Les Nord des deux repères sont donc différents. A l'emplacement des tests, la déclinaison magnétique est d'environ 3°.
— **Calibration du drone** : Étant donné que l'orientation du repère local du drone est fixée par sa boussole, une mauvaise calibration du magnétomètre pourrait entrainer des biais dans l'orientation des vecteurs de vitesses. Le logiciel du drone permet d'effectuer des calibrations de ses instruments internes (boussole, IMU) ;
— **Précision des mesure du drone** : Un biais peut également être lié à la précision de mesure des vecteurs de vitesses. Peu d'informations sur la précision sont données par le constructeur DJI ;

En appliquant une correction de la déclinaison magnétique, la trajectoire calculée par le filtre colle bien avec les observations faites à la station totale a moins de 5 cm (Figure 2.24).

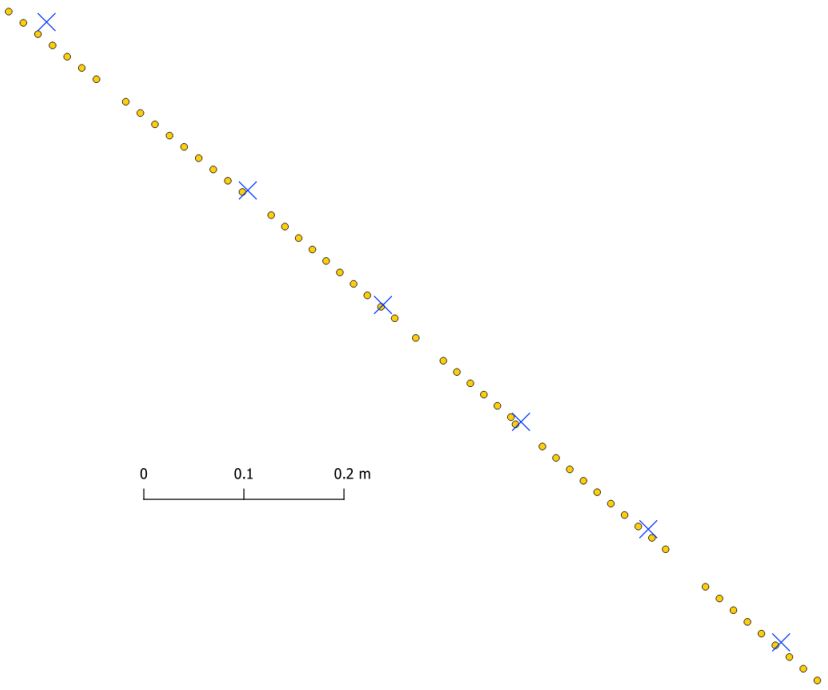


FIGURE 2.24 – Résultat du filtre de Kalman (point jaune) ; Observation de la station totale (Croix bleu).

Comme expliqué au Chapitre 2.3, il y a parfois des baisses de fréquence dans les mesures de la station totale. Cela se traduit par une zone plus longue sans observation terrestre. Tant qu'aucune nouvelle observation terrestre n'est faite, la trajectoire est uniquement calculée à partir des vecteurs de vitesse provenant du drone. Selon la vitesse de déplacement du drone, cette zone peut être plus ou moins grande. Dans l'exemple de la Figure 2.25, la zone sans observation terrestre est d'environ 6 mètres. Lors du raccord avec l'observation terrestre suivante, une dérive de la trajectoire d'environ 10 cm est constatée. Néanmoins cette précision est suffisante pour permettre le guidage du drone. De plus, les vitesses de déplacement pourraient être réduites, ce qui réduirait la longueur des zones sans observation terrestre et donc la dérive de la trajectoire calculée.

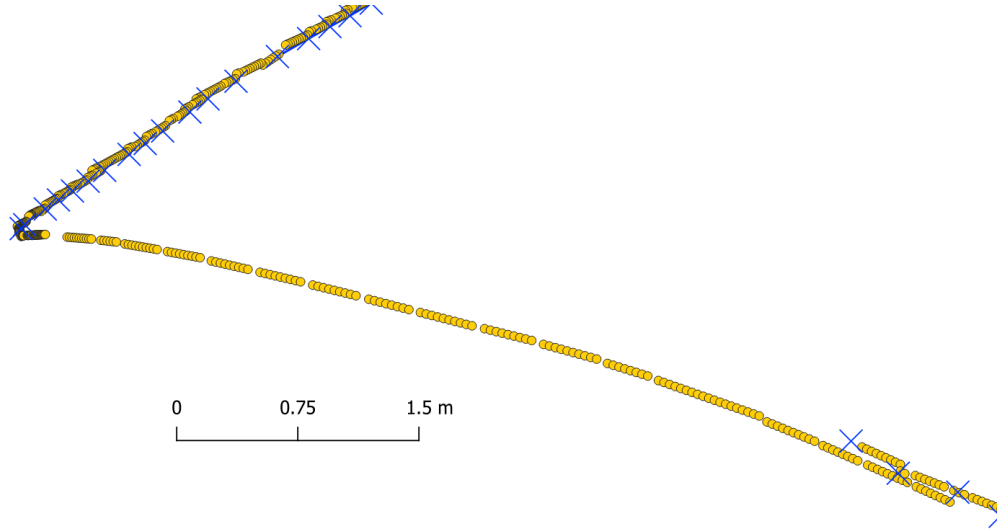


FIGURE 2.25 – Zone sans observation terrestre : résultat du filtre de Kalman (point jaune) ; Observation de la station totale (croix bleu).

En ce qui concerne la réactivité du filtre, la modélisation linéaire permet d'être suffisamment réactif aux changements de direction brusques. Ce qui est possible si le drone effectue un virage serré.

Filtre de Kalman non-linéaire

Comme expliqué dans le Chapitre 2.4.2, les observations faites par le drone sont exploitées sous forme d'une norme de vitesse 2D et d'un gisement. Cette modélisation a pour but d'introduire un biais dans le gisement du vecteur vitesse. Ce biais est calculé à chaque époque du filtre. Ainsi les problèmes liés à la cohérence d'orientation des systèmes de la station totale et des systèmes drone peuvent être corrigés. Cependant cette modélisation impacte directement sur la réactivité du filtre aux changements de direction brusques. En effet, une précision est associée à ce bruit dans le gisement ce qui contraint la direction de la position prédite. Cela pose particulièrement des problèmes lorsqu'il y a des changements de direction violents. Dans les phases rectilignes le filtre fonctionne correctement, et les erreurs d'orientation des vecteurs de vitesses sont bien compensés (Figure 2.26).

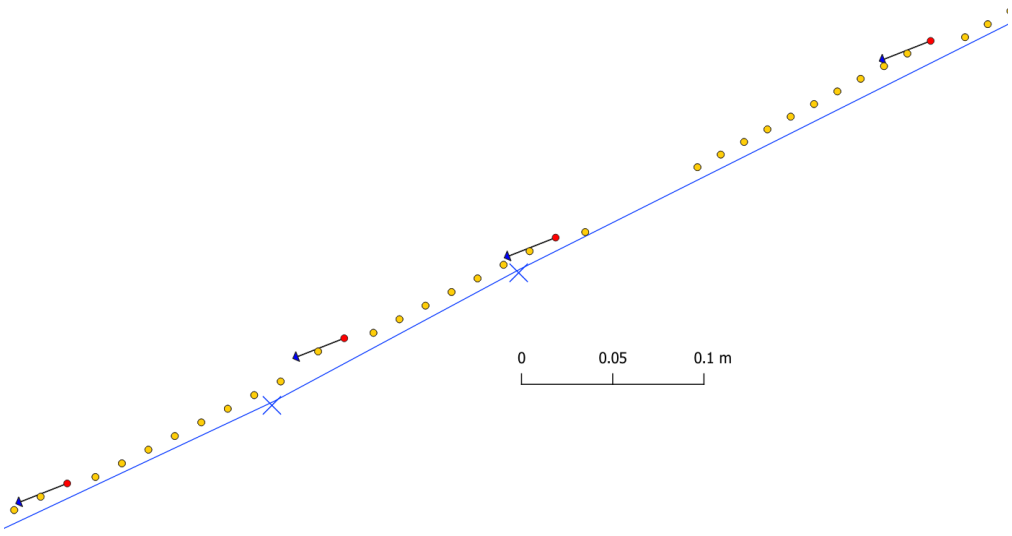


FIGURE 2.26 – Résultat du filtre de Kalman non-linéaire (point jaune) ; Observation de la station totale (Croix bleu).

Analyse selon le modèle de drone utilisé

Les différents tests exposés jusqu'à maintenant ont été réalisés à l'aide d'un drone DJI phantom 4 qui est un drone grand public. Un nouvel échantillon d'observation est mesuré avec cette fois ci un drone professionnel DJI Matrice 210 V2.

La trajectoire de la Figure 2.27 est calculée avec un filtre linéaire et avec les données brutes du drone Matrice 210 (sans correction de la déclinaison magnétique). Contrairement au drone phantom 4, il n'y a ici aucun problème dans l'orientation des vecteurs de vitesses. La trajectoire calculée par le filtre de Kalman est très proche des observations terrestres (2-3 cm).

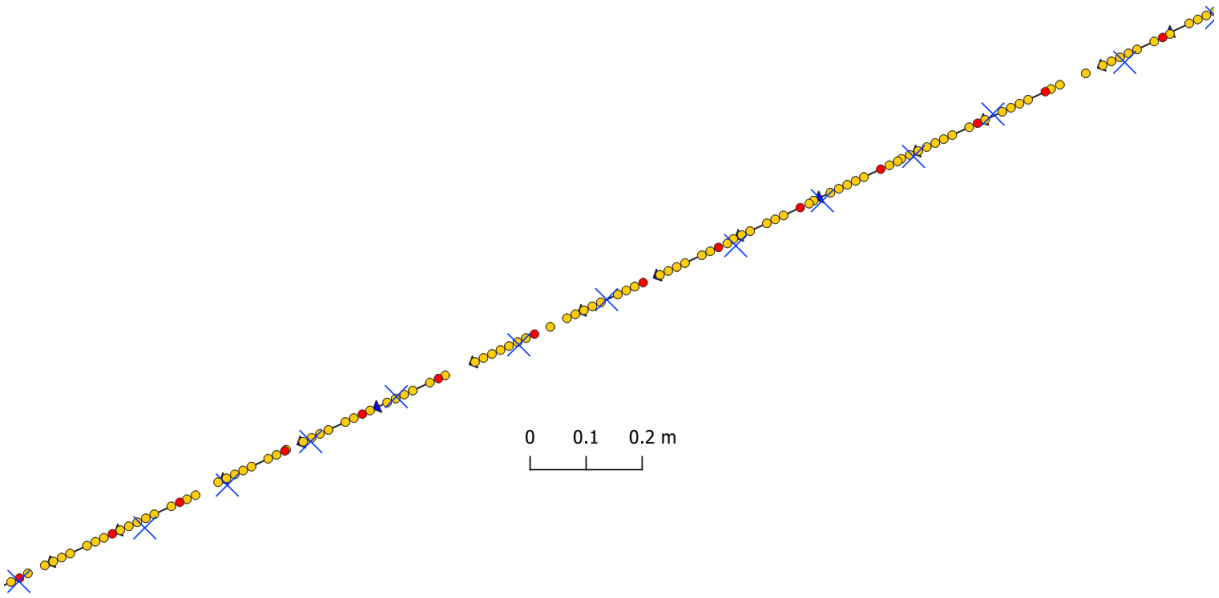


FIGURE 2.27 – Résultat du filtre de Kalman (point jaune) ; Observation de la station totale (Croix bleu).

Le modèle de drone utilisé à donc un impact important sur le calcul de trajectoire. Cette meilleure détermination des vitesses est due à :

— **Robustesse du drone** : Le drone Matrice étant beaucoup plus grand et beaucoup plus lourd, il est plus stable en vol et résiste mieux au vent. Ce paramètre n'impacte pas directement sur la précision des mesures, mais plutôt sur le maintient de la trajectoire ;
— **Système de mesure** : Le drone Matrice embarque plus de capteur qu'un drone grand public (plusieurs compas, plusieurs IMU, plus de caméra pour le calcul du flux optique). Les vecteurs de vitesses étant le résultat de l'assemblage de ces différents capteurs, sa détermination est meilleure.

L'utilisation du Matrice 210 est donc privilégiée pour la suite des tests.

Filtre retenu

Suite à cette série de test, le filtre linéaire est retenu pour la suite du travail pour les raisons suivantes :

— La modélisation du filtre est plus simple et donc son implémentation en Java aussi. Sa consommation en ressources de calculs est également plus faible que le filtre non-linéaire ;
— Le filtre linéaire semble plus réactif et donc s'adapte mieux à la situation d'un drone qui effectue des changements de direction brusques ;
— Les résultats obtenus par la modélisation linéaire sont similaires à ceux obtenus avec la modélisation non-linéaire, il est donc préférable d'utiliser le processus le plus simple.

Les paramètres retenus pour ce filtre sont les suivants :

Paramètres Kalman		
$\sigma_{x_{THEO}}$	0.05 [m]	Équation 2.10
$\sigma_{y_{THEO}}$	0.05 [m]	Équation 2.10
$\sigma_{z_{THEO}}$	0.05 [m]	Équation 2.10
$\sigma_{\dot{x}_{DRONE}}$	0.1 [m/s]	Équation 2.10
$\sigma_{\dot{y}_{DRONE}}$	0.1 [m/s]	Équation 2.10
$\sigma_{\dot{z}_{DRONE}}$	0.1 [m/s]	Équation 2.10
$\sigma_{\ddot{x}_{MODEL}}$	3 [m/s^2]	Équation 2.17
$\sigma_{\ddot{y}_{MODEL}}$	3 [m/s^2]	Équation 2.17
$\sigma_{\ddot{z}_{MODEL}}$	3 [m/s^2]	Équation 2.17

TABLE 2.1 – Valeurs retenues pour les paramètres du filtre de Kalman linéaire

2.4.7 Test de la robustesse des vitesses mesurées par le drone

Les vitesses renvoyées par le drone étant l'assemblage de plusieurs capteurs (GNSS, IMU, Baromètre, Magnétomètre), il est intéressant d'étudier l'impact de la perte d'un de ces capteurs. Le capteur qui risque le plus de poser problème est le GNSS, étant donné qu'il est prévu d'utiliser le guidage du drone près d'un mur. Il ne faudrait pas que les vitesses renvoyées par le drone ainsi que la trajectoire calculée par le filtre de Kalman soient impactées par la perte de signal GNSS. Pour étudier cette problématique une trajectoire test est effectuée en démarrant depuis une position dégagée puis en passant sous un couvert (en trait-tillé vert sur la Figure 2.28). Ce couvert fermé sur trois cotés a pour but de créer une zone sans réception GNSS.
En analysant la trajectoire calculée en live par le filtre de Kalman, aucune incohérence n'est détectée lors du passage du drone dans la zone sans réception GNSS (Figure 2.28). Cette analyse est essentiellement visuelle et consiste à contrôler qu'il n'y a pas de "cassure" dans la trajectoire lors du passage dans la zone de masque GNSS. Les vitesses renvoyées par le drone semblent donc être assez robustes par rapport à une perte de signal GNSS et restent cohérentes.

Cependant, ce test est effectué à une hauteur de vol relativement basse (1.5 mètres). Le système de positionnement optique est donc très efficace et permet de fournir des information de déplacement au drone pour compenser la perte de signal GNSS.

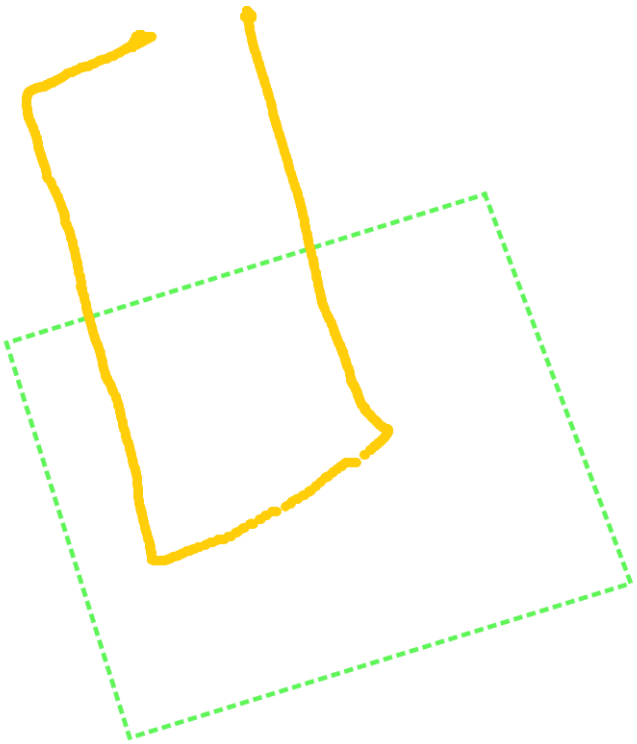


FIGURE 2.28 – Résultat du filtre de Kalman (jaune) ; Zone de masque GNSS (vert).

2.4.8 Implémentation en Java et calcul en live

Les différents tests précédents ont été effectués sur différents jeux de données statiques stockés dans des fichiers texte. Ils ont ainsi permis de valider le bon fonctionnement des différents calculs et d'ajuster correctement les différents paramètres du filtre. Il faut maintenant pouvoir calculer la trajectoire du drone en live lors du vol. Il faut donc dans un premier temps implémenter les différentes fonctions en Java dans l'application qui communique avec le drone et la station totale. Et ensuite gérer la réception des observations en live. Les détails de l'implémentation sont disponibles en Annexe B.2.

La gestion des observations en live se fait selon le schéma de la Figure 2.29. Chaque fois qu'une nouvelle observation (station totale ou drone) est obtenue, une époque du filtre est calculée en tenant compte de cette observation. Si aucune nouvelle observation n'est disponible, une époque du filtre est calculée en mettant des sigmas très haut sur les observations (Équation 2.22). Cela a pour conséquence de ne pas prendre en compte l'observation donnée, le filtre est donc uniquement calculé sur sa phase de prédiction.

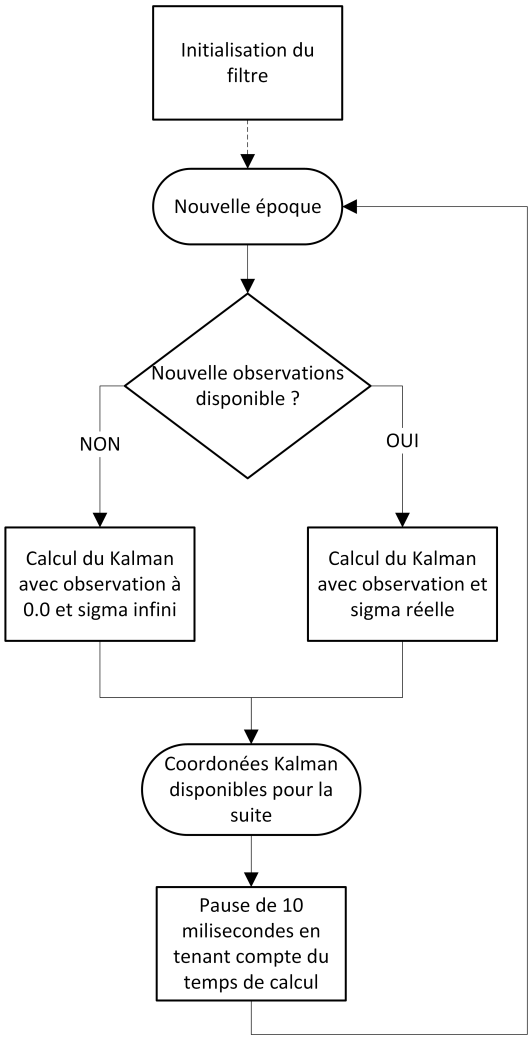


FIGURE 2.29 – Gestion des flux de données en live pour le calcul du filtre de Kalman.

Chaque époque du filtre est calculée dans une boucle "infinie". A la fin de chaque époque un temps de pause est marqué. Le temps de pause est calculé de manière à ce que : $temps_{calcul} + temps_{pause} = 10\ millisecondes$. Cela veut dire que le filtre renvoie des époques à 100 Hz. Les observations étant mesurées à environ 10 Hz, il y a donc environ 10 états prédits entre chaque observation.

Ci-dessous l'algorithme plus détaillé pour la partie des observations terrestres. Le principe est le même pour les observations du drone. La seule différence est sur la fonction "if" qui teste si une nouvelle observation est disponible. Pour les mesures du drone, ce test ne peut pas directement être fait sur les vitesses étant donné qu'il peut y avoir des vitesses constantes entre différentes observations successives. Une vitesse constante est une information importante à prendre en compte dans le calcul du filtre. Ce test se fait donc sur l'information de positions X,Y,Z renvoyée par le drone dont l'envoi est synchronisé avec les vitesses.

Algorithm 1 Détection des nouvelles observations terrestres

$x_{old} \leftarrow 9999$
$y_{old} \leftarrow 9999$
$z_{old} \leftarrow 9999$
$x_{kalman} \leftarrow 9999$
$y_{kalman} \leftarrow 9999$
$z_{kalman} \leftarrow 9999$
while kalman run **do**
  $x_{obs} \leftarrow$ dernière observation
  $y_{obs} \leftarrow$ dernière observation
  $z_{obs} \leftarrow$ dernière observation
  if $x_{obs} = x_{old}$ & $y_{obs} = y_{old}$ & $z_{obs} = z_{old}$ **then**
    $x_{kalman} \leftarrow 0.0$
    $y_{kalman} \leftarrow 0.0$
    $z_{kalman} \leftarrow 0.0$
    $\sigma_{x_{kalman}} \leftarrow 9999^2$
    $\sigma_{y_{kalman}} \leftarrow 9999^2$
    $\sigma_{z_{kalman}} \leftarrow 9999^2$
  else
    $x_{kalman} \leftarrow x_{obs}$
    $y_{kalman} \leftarrow y_{obs}$
    $z_{kalman} \leftarrow z_{obs}$
    $\sigma_{x_{kalman}} \leftarrow 0.05$
    $\sigma_{y_{kalman}} \leftarrow 0.05$
    $\sigma_{z_{kalman}} \leftarrow 0.05$
  end if
  $x_{old} \leftarrow x_{obs}$
  $y_{old} \leftarrow y_{obs}$
  $z_{old} \leftarrow z_{obs}$
  Calcul du filtre en utilisant les variables $_{kalman}$ pour remplir les matrices d'obervation décrites dans les Équations 2.8 et 2.10
  Whait(10 $millisecondes - temps_{calcul}$)
end while

Les éléments calculés par le filtre de kalman en temps réel sont enregistrés dans un fichier .txt sur le smartphone, ce qui permet un contrôle de la trajectoire en post-traitement. Ci-dessous le résultat de la trajectoire calculée en live avec un filtre linéaire et le drone DJI Matrice 210 (Figure 2.30).

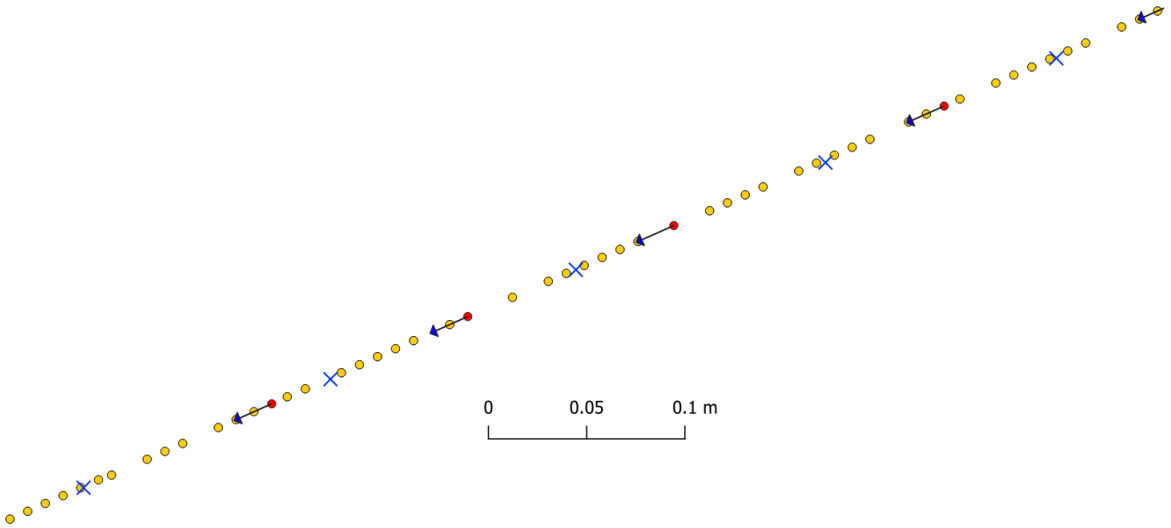


FIGURE 2.30 – Résultat du filtre de Kalman calculé en live pendant le vol (point jaune) ; Vecteur de vitesse mesuré par le drone (point rouge et vecteur) ; Observation de la station totale (Croix bleu).

La trajectoire calculée est très proche des observations terrestres de la station totale (1-2 cm) et les vecteurs de vitesse mesurés par le drone sont cohérents avec les observations terrestres.

Suite à cette implémentation en Java, la trajectoire du drone peut donc être calculée en live et la position du drone peut être obtenue à 100 Hz avec une dérive de quelques centimètres. Ces positions seront utilisées par la suite pour effectuer le guidage du drone par rapport à une trajectoire donnée.

Simulateur de station totale

Le calcul de Kalman implémenté en Java fonctionne avec un flux de données continu provenant du drone et de la station totale. Or, pour effectuer des tests avec le simulateur DJI, aucune observation terrestre à la station totale n'est effectuée, étant donné que le drone reste physiquement fixe. Le but de ce simulateur est de créer un flux de données en continu, idem à celui qu'une station totale pourrait envoyer.

Comme expliqué au Chapitre 2.4.4, les coordonnées issues du simulateur DJI peuvent être utilisées pour remplacer les observations terrestres. Ces coordonnées sont enregistrées en continu dans un fichier .txt par le simulateur DJI. Ces coordonnées sont donc enregistrées sur un PC et doivent être envoyées sur le smartphone pour pouvoir être prise en compte dans le filtre de Kalman.

La communication entre le PC et le smartphone se fait au moyen de sockets [5]. Du côté PC, un script Python (Annexe B.2.7) lit en continu le fichier écrit par le simulateur DJI et diffuse la dernière ligne à l'aide d'un socket (le PC est le socket serveur). Du coté client, le smartphone lit le socket en question et récupère ainsi la dernière coordonnée fournie par le simulateur DJI.

Il est ainsi possible de recevoir un flux de données en continu sur un smartphone, simulant la réception d'observation terrestre. Le calcul du Kalman en live peut donc être effectué à partir des données du simulateur DJI.

5. "connecteur réseau" qui permet d'établir des connexions puis de recevoir et d'envoyer des données grâce à elle.

2.5 Guidage du drone

Cette partie traite tout ce qui concerne le guidage du drone selon un plan de vol donné. Le développement du guidage est composé des étapes suivantes :

1. **Principe de correction de la trajectoire via PID** : Élaboration du contrôleur de vol pour calculer les corrections (Chapitre 2.5.1) ;
2. **Intégration simple du PID dans un boucle de rétroaction** (Chapitre 2.5.2) ;
3. **Intégration avancée du PID dans un boucle de rétroaction** (Chapitre 2.5.3) ;
4. **Mise en place de contrainte sur le PID** : Des contraintes doivent être ajoutées sur les corrections calculées pour respecter des critères de vitesse (Chapitre 2.5.4) ;
5. **Ajustement des coefficients K du PID** : Les coefficients K du PID sont ajustés par modélisation (Chapitre 2.5.5) ;
6. **Gestion de l'orientation du drone** : Un processus de correction est également développé pour gérer l'orientation du drone (Chapitre 2.5.6) ;
7. **Intégration en Java** : Finalement l'ensemble du processus est intégré en Java (Chapitre 2.5.8).

2.5.1 Correction de la trajectoire - PID

Comme expliqué dans le Chapitre 1.5, le contrôleur PID est utilisé. Celui-ci a pour but de calculer les corrections à appliquer au drone, en fonction de la position calculée par le filtre de Kalman. Les corrections appliquées sur le drone se font sous forme de vitesse Vx, Vy, Vz.
La correction est calculée indépendamment pour les trois axes de déplacement et un PID est calculé pour chacune de ces trois composantes de vitesse selon la formule de base de l'Équation 1.6. Ce qui donne :

$$\begin{aligned} u_x(t) &= K_p e_x(t) + K_i \int_0^t e_x(\tau)\mathrm{d}\tau + K_d \frac{\mathrm{d}e_x(t)}{\mathrm{d}t} \\ u_y(t) &= K_p e_y(t) + K_i \int_0^t e_y(\tau)\mathrm{d}\tau + K_d \frac{\mathrm{d}e_y(t)}{\mathrm{d}t} \\ u_z(t) &= K_p e_z(t) + K_i \int_0^t e_z(\tau)\mathrm{d}\tau + K_d \frac{\mathrm{d}e_z(t)}{\mathrm{d}t} \end{aligned} \tag{2.27}$$

Les erreurs e_x, e_y, e_z correspondent à la décomposition du vecteur DB, qui est le vecteur entre le drone et le point de la trajectoire à atteindre (Figure 2.31).

Les corrections u_x, u_y, u_z correspondent aux vitesses à appliquer au drone pour atteindre le point B de la trajectoire.

Le premier terme $K_p e(t)$ représente simplement une correction proportionnelle à l'erreur observée amplifiée par un certain gain. L'ajout de ce gain permet d'obtenir une réponse plus ou moins rapide en effectuant des corrections plus ou moins fortes. Plus ce gain est grand, plus les corrections sont fortes, mais la stabilité du système est impactée.

Le deuxième terme $K_i \int_0^t e(\tau)\mathrm{d}\tau$ correspond à l'intégration de l'erreur par rapport au temps. Il s'agit donc de la somme des erreurs passées qui est multipliée par une constante K_i. Ce terme permet

d'éviter d'éventuelles erreurs statiques qui pourraient survenir avec un simple contrôle proportionnel. Lorsque le système s'approche de sa consigne (drone qui arrive sur le point visé) l'erreur observée peut ne plus être assez importante pour faire avancer le drone. Le terme intégral permet alors de compenser ce manque grâce à la somme des différentes erreurs passées.

Dans le cas d'un drone, il se pourrait aussi que la vitesse de correction retournée par le terme proportionnel soit égale à un vent contraire sur le drone. Cela aurait pour conséquence que le drone ne se déplace pas. Le drone restant en stationnaire, l'erreur dans le terme intégral s'accumulerait et permettrait ainsi de compenser ce phénomène. Cette problématique particulière dépend cependant du mode de déplacement du drone. Si celui-ci commande son déplacement uniquement par des puissances moteur, cette problématique peut apparaitre. Au contraire, si le déplacement du drone se fait à l'aide de capteurs comme le GNSS ou l'IMU, il ne devrait pas y avoir de problème. Les drones DJI utilisent l'ensemble de leurs capteurs internes pour effectuer leurs déplacements.

Le troisième terme $K_d \frac{de(t)}{dt}$ consiste à dériver l'erreur par rapport au temps et à la multiplier par une constante K_d. Cela consiste à multiplier la différence entre l'erreur actuelle et l'erreur passée par K_d. Sans ce terme, il se peut que la consigne soit dépassée ce qui provoquerait des oscillations autour du point visé. Le terme dérivé permet de limiter cela en appliquant une action dans le sens opposé lorsque le système se rapproche de la consigne.

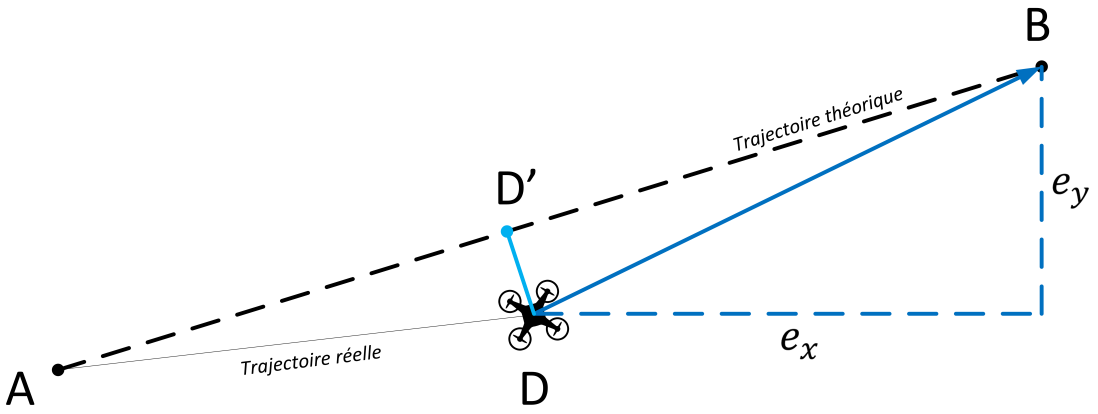


FIGURE 2.31 – Déplacement du drone selon une trajectoire théorique - Décomposition des erreurs selon X, Y et Z (différence d'altitude entre B et D).

Deux possibilités sont imaginées pour l'intégration de ce processus de correction dans une boucle de rétroaction :

— Processus simple (Chapitre 2.5.2) ;
— Processus avancé utilisant un point virtuel (Chapitre 2.5.3).

L'analyse entre ces deux méthodes est traitée au Chapitre 3.1.5.

2.5.2 Processus simple

Ce processus reprend le principe de calcul des erreurs e_x, e_y, e_z selon les explication du Chapitre 2.5.1 et l'intégration se fait selon le schéma ci-dessous (Figure 2.32) :

1. Calcul de la distance 3D entre le drone et le point de la trajectoire à atteindre (Vecteur DB) ;

2. Si cette distance est en-dessous du seuil fixé par l'utilisateur, le point de passage est considéré comme atteint. Le drone ajuste alors son cap selon le prochain segment de trajectoire et démarre son déplacement selon ce nouveau segment ;
3. Si la distance (Norme DB) est supérieure au seuil défini, le processus de correction entre en action ;
4. Le régulateur PID calcul les vitesses de correction à partir des erreurs données en tenant compte des contraintes selon le Chapitre 2.5.4 *Contrainte sur le PID* ci-dessous ;
5. Les corrections sont envoyées au drone ;
6. Puis le processus recommence.

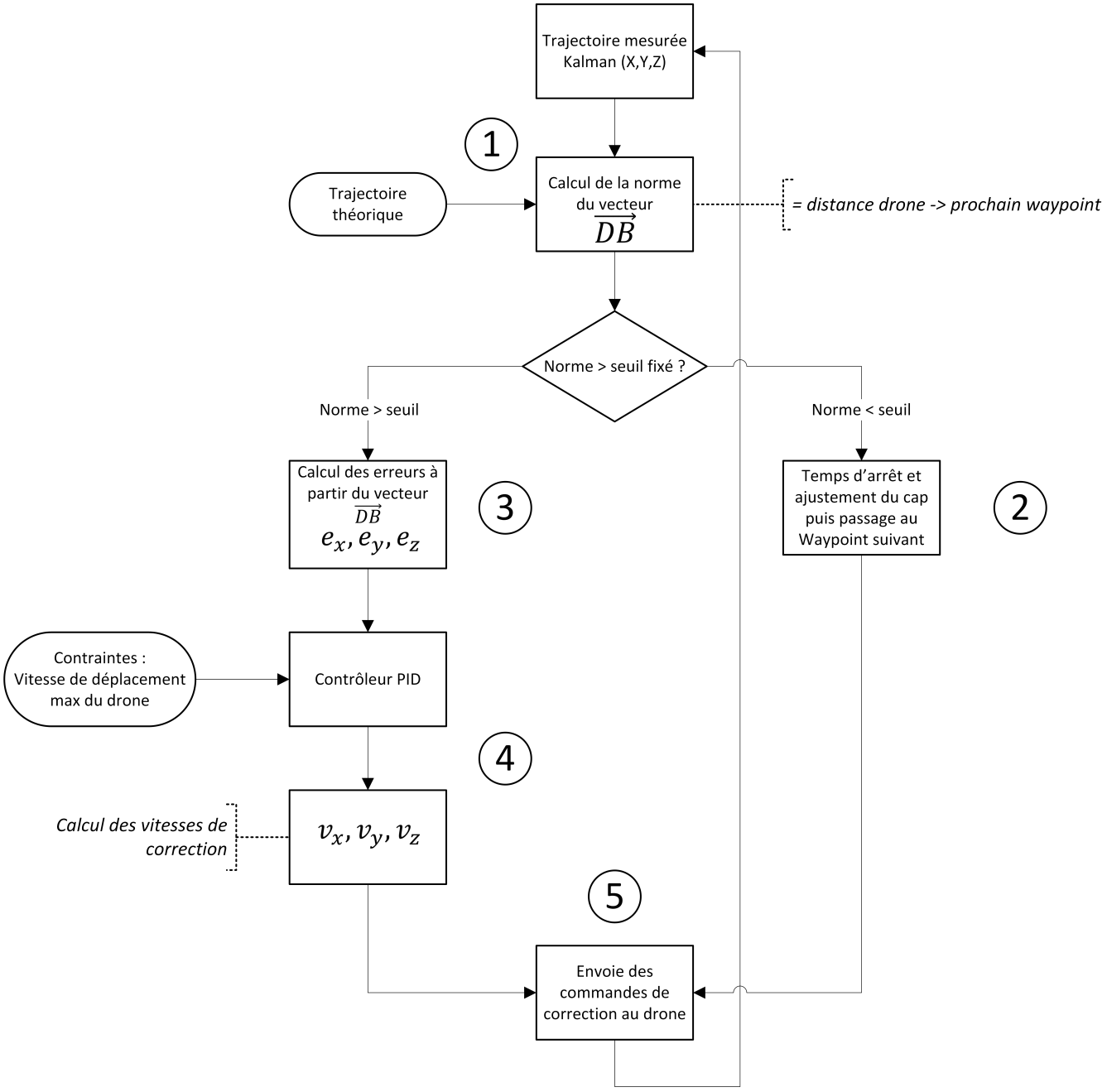


FIGURE 2.32 – Boucle de rétroaction

2.5.3 Processus avancé - point virtuel

Un processus de correction plus complexe peut être mis en place pour effectuer cette correction de trajectoire. Des phénomènes externes, comme une rafale de vent par exemple, peuvent amener le drone à dériver de sa trajectoire de base. En dérivant au départ de la trajectoire, les corrections appliquées sur les déplacements du drone ne permettent pas de le rabattre sur la trajectoire du plan de vol. En effet, la correction à pour but d'atteindre le prochain waypoint, et non la trajectoire proche du drone. Donc si le drone s'écarte fortement du plan de vol, aucune correction n'est appliquée pour se rabattre rapidement sur la trajectoire à suivre.

Il faut alors réduire la distance entre les waypoints pour diminuer ce problème. Une solution est de créé un point virtuel sur la trajectoire (V sur la Figure 2.34). Ce point se déplace avec le drone et est positionné 5 mètres devant le drone sur la trajectoire. Ce point virtuel est calculé selon le principe de la Figure 2.33.

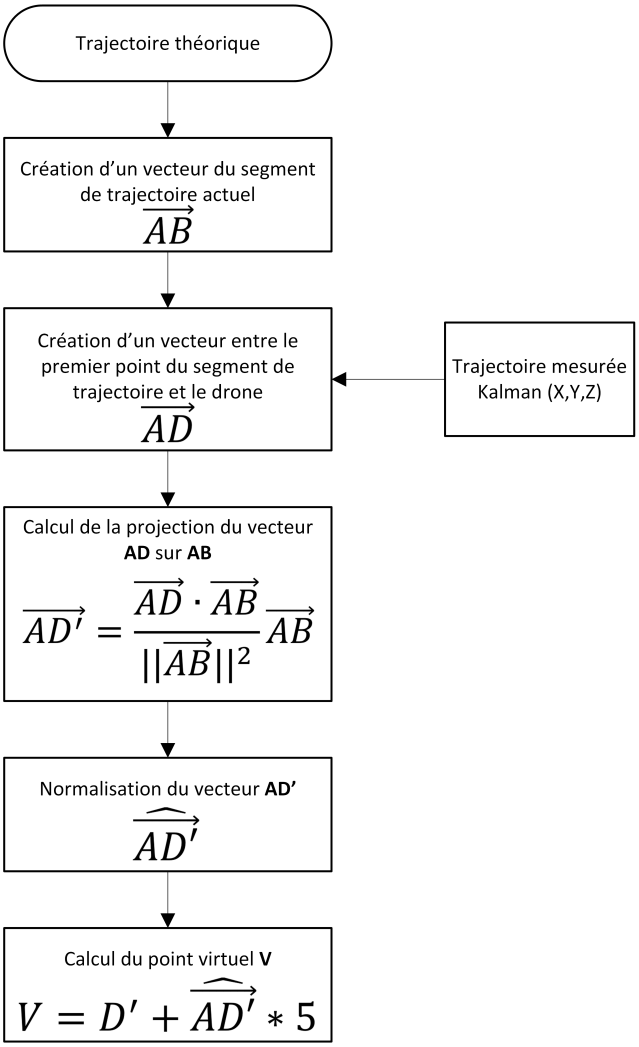


FIGURE 2.33 – Processus pour la création d'un point virtuel.

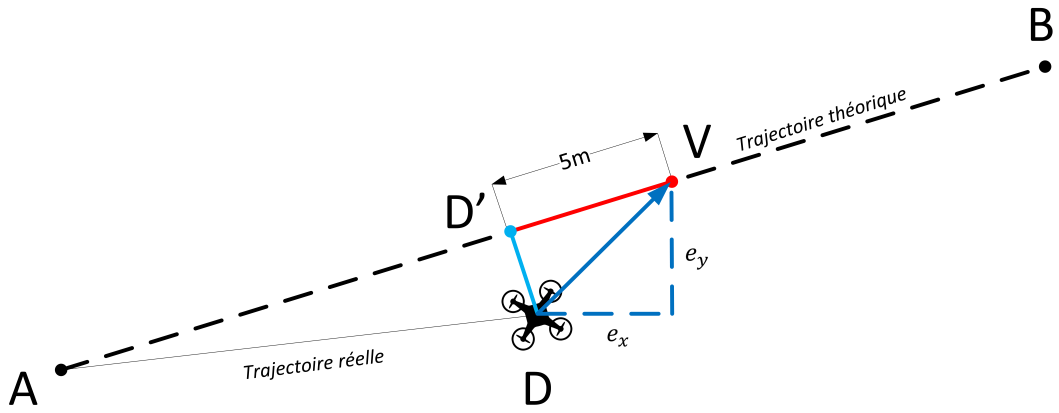


FIGURE 2.34 – Création d'un point virtuel 5 m devant le drone et calcul des erreurs.

Comme exposé au début du Chapitre 2.5.1, un vecteur de delta (e_x, e_y, e_z) est donné au PID pour calculer les corrections. Ce vecteur peut alors être calculé entre le drone et le point virtuel (à la place du waypoint), il s'agit du vecteur DV sur la Figure 2.34. Le drone vise donc un point à seulement 5 mètres, ce qui lui permet de se rabattre plus vite sur la trajectoire et de moins dériver. A chaque état, un nouveau point virtuel est calculé.

Cependant, la norme du vecteur d'erreur est importante pour gérer les vitesses de déplacement (plus l'erreur est grande, plus la vitesse de déplacement est élevée). En intégrant ce point virtuel, la norme de l'erreur reste constante à environ 5 mètres. L'intelligence de gestion des vitesses du PID est alors perdue. Pour palier à ce problème, la direction du vecteur d'erreur est donnée par le point virtuel, mais sa norme est donnée par la distance entre le drone et les waypoints. Cette opération se fait en normalisant le nouveau vecteur de correction puis en le multipliant par la norme de l'ancien vecteur de correction. Au final, le drone est capable de corriger plus rapidement sa trajectoire tout en conservant la gestion des vitesses selon les distances aux waypoints.

En reprenant le schéma de fonctionnement général du PID de la Figure 2.32, le seul changement à appliquer pour prendre en compte l'ajout de ce point virtuel se situe au niveau du point N°3. Le calcul des erreurs ne se fait alors plus à partir du vecteur BD, mais à partir du nouveau vecteur d'erreur DV.

2.5.4 Contrainte sur le PID

Dans le cas ou la distance entre le drone et le waypoint à atteindre est importante, les vitesses de correction renvoyées par le drone sont très élevées étant donné qu'elles sont proportionnelles à l'erreur. Cependant, il y a des limites physiques et mécaniques à prendre compte pour ces vitesses. Comme par exemple la vitesse maximale de déplacement du drone qui peut être définie soit par la vitesse maximale mécanique soit par une vitesse fixée arbitrairement. Dans le cas de prise de vue aérienne, ce qui est le cas ici, la vitesse maximale de déplacement est donnée par rapport aux paramètres du plan de vol. La vitesse de déplacement doit permettre d'obtenir des clichés nets et également permettre un recouvrement suffisant entre les images successives.

L'algorithme ci-dessous est utilisé pour contraindre les vitesses issues du PID (Figure 2.35) :

1. Calcul de la norme du vecteur vitesse calculé par le PID. La vitesse de déplacement 3D est ainsi obtenue ;
2. Si cette vitesse 3D est inférieure à la vitesse maximale, elle peut directement être utilisée pour effectuer les corrections sur le drone ;
3. Si la vitesse 3D est supérieure à la vitesse maximale, elle doit être réduite avant de pouvoir être utilisée comme correction sur le drone. Pour ce faire, le vecteur de vitesse est normé puis multiplié par la vitesse maximale. Ainsi la direction de déplacement est conservée, mais à une vitesse maximale fixée ;

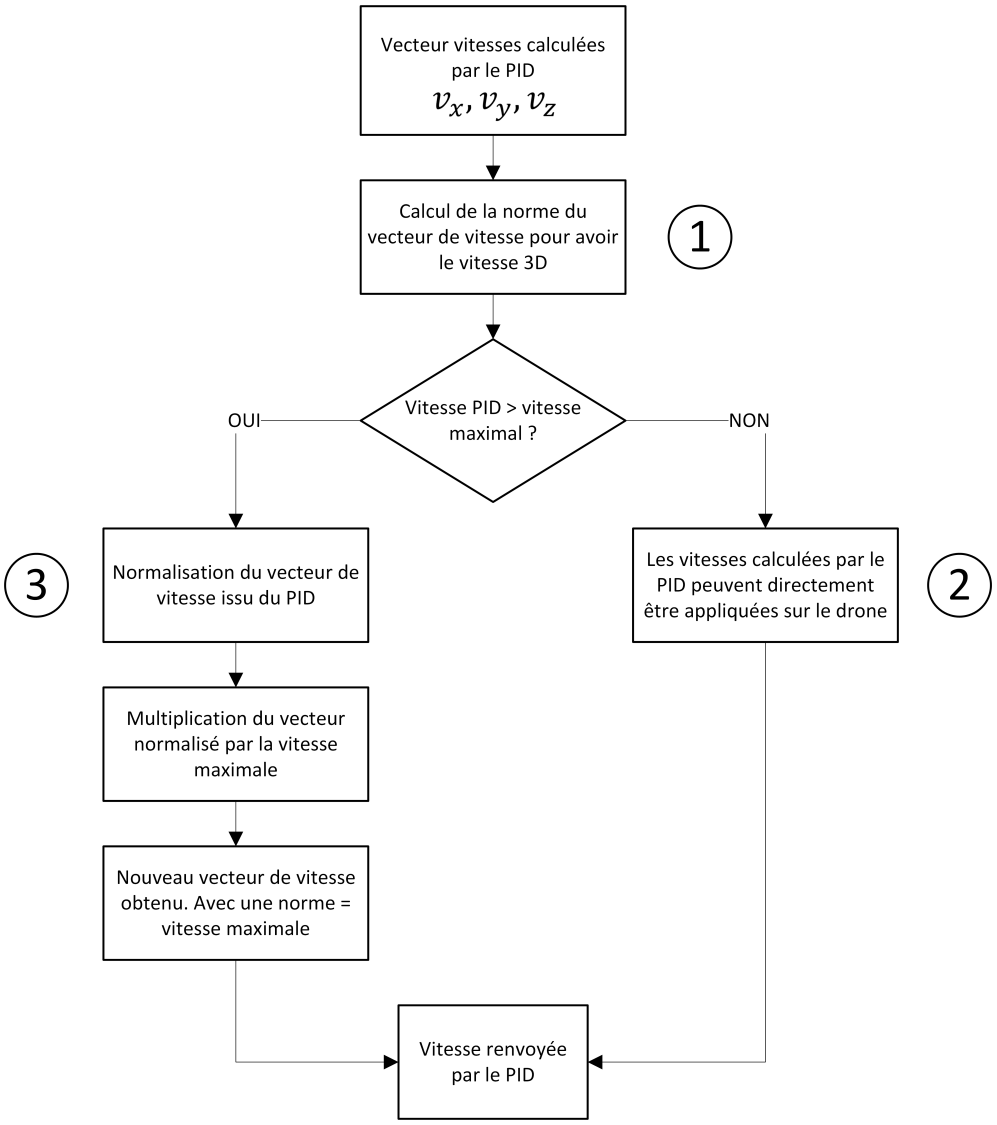


FIGURE 2.35 – Gestion des contraintes sur les vitesses renvoyées par le PID.

2.5.5 Ajustement des coefficients par modélisation

Maintenant que l'algorithme de guidage est défini, il reste une dernière étape qui consiste à déterminer les coefficients K_p, K_i et K_d de l'Équation 2.27. Si le système n'est pas trop compliqué, il est possible de définir ces coefficients en modélisant le système et en effectuant des simulations. Celles-ci sont effectuées en Python avant d'effectuer l'intégration en Java. Pour simuler le guidage du drone, une trajectoire de test qui servira de plan de vol est créée (Figure 2.36).

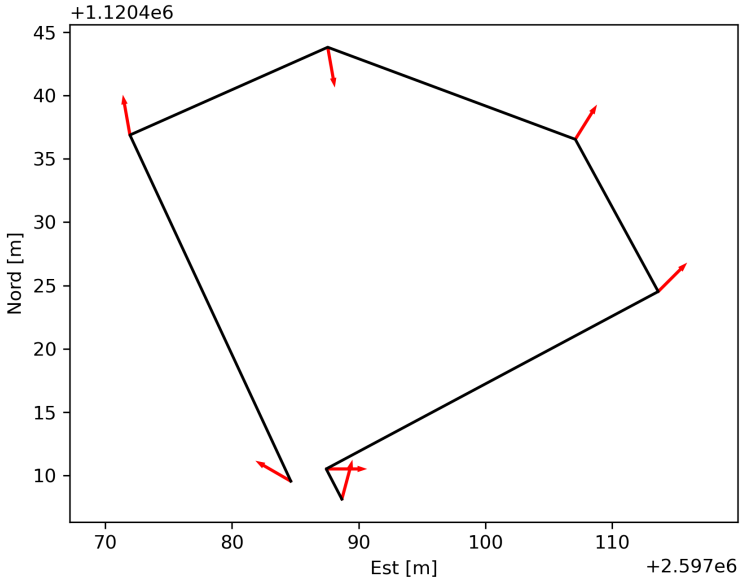


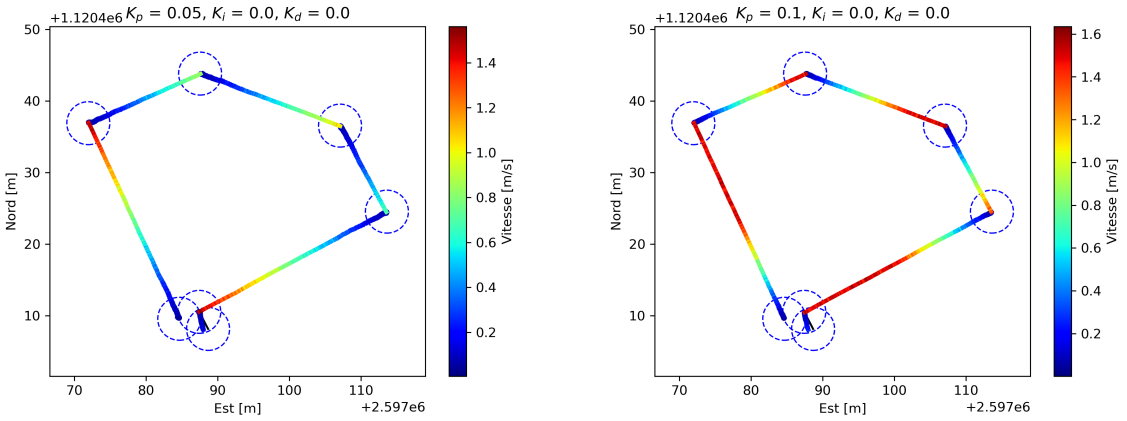
FIGURE 2.36 – Trajectoire test pour simuler le guidage du drone ; Vecteur rouge : Orientation des prises de vues.

La modélisation des déplacements du drone est la suivante :

$$\begin{aligned} x(t) &= x(t-1) + \dot{x}_{PID} * \Delta t \\ y(t) &= y(t-1) + \dot{y}_{PID} * \Delta t \\ z(t) &= z(t-1) + \dot{z}_{PID} * \Delta t \end{aligned} \tag{2.28}$$

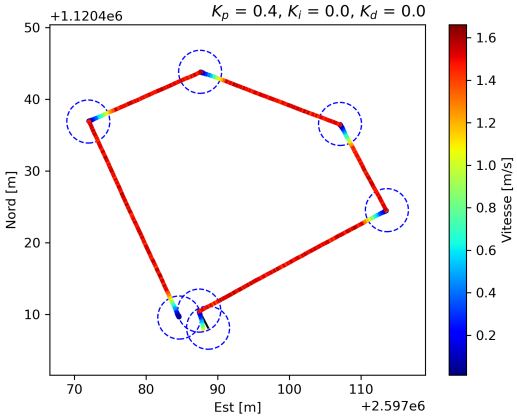
En donnant un point de départ il est ainsi possible de modéliser les déplacements du drone à partir des vitesses renvoyées par le PID. Un biais est également rajouté sur les vitesses renvoyées par le PID pour éviter d'avoir une correction parfaite.

Dans un premier temps il faut déterminer la valeur idéale pour le coefficient K_p qui donne l'aspect général du guidage. En appliquant le processus défini précédemment (Figure 2.32), les Figures 2.37 qui correspondent à la variation de la vitesse du drone le long de la trajectoire en fonction de K_p sont obtenues.

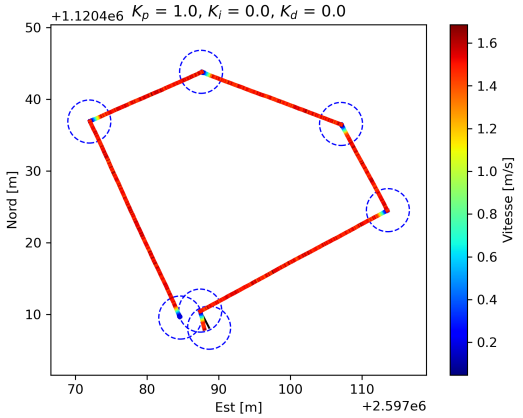


(a) $K_p = 0.05, K_i = 0.0, K_d = 0.0$

(b) $K_p = 0.1, K_i = 0.0, K_d = 0.0$



(c) $K_p = 0.4, K_i = 0.0, K_d = 0.0$

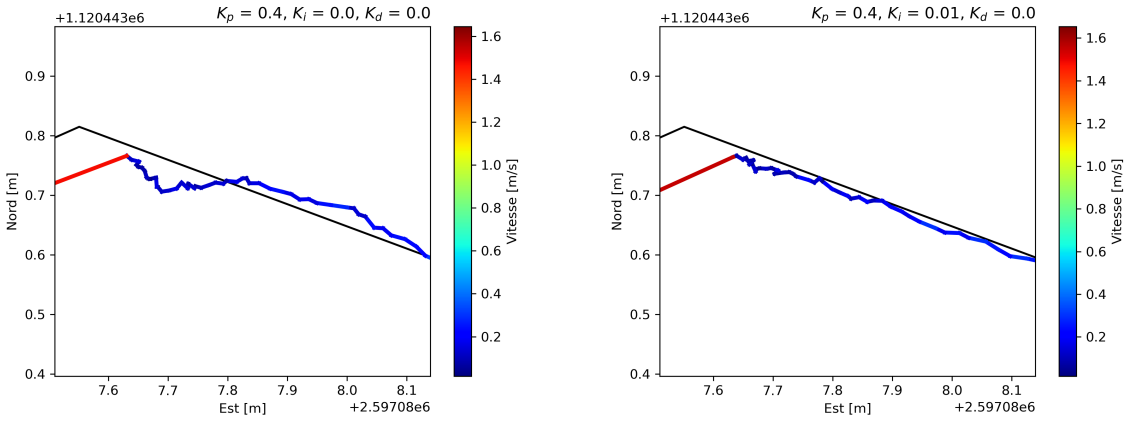


(d) $K_p = 1.0, K_i = 0.0, K_d = 0.0$

FIGURE 2.37 – Impact du coefficient K_p sur les vitesses de correction appliquées au drone ; Cercle trait-tillé bleu = zone à 3 m d'un waypoint ; Sens du plan de vol = contraire aux aiguilles d'une montre.

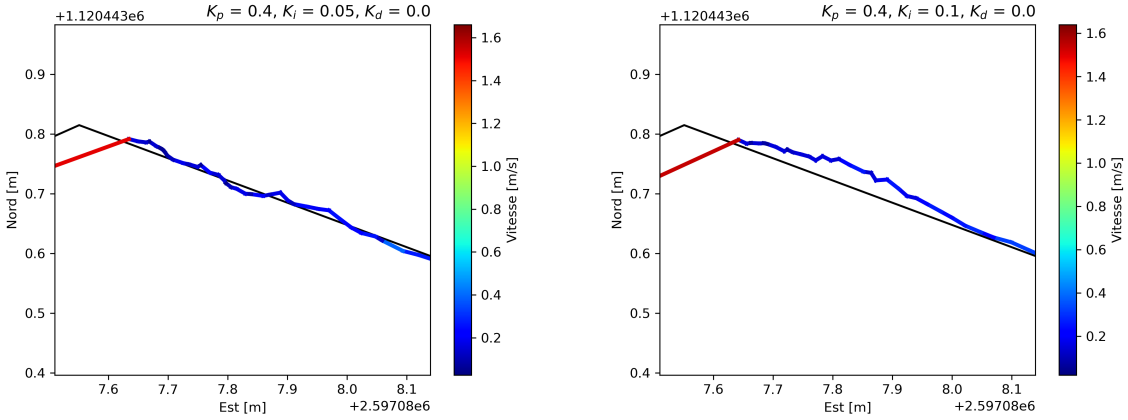
Le but étant que le drone se déplace à la vitesse maximale fixée (ici 1.5 m/s), puis commence à ralentir lorsqu'il arrive à environ 3 m d'un waypoint. Après différents tests, la valeur retenue pour K_p est de 0.40 (Figure 2.37c). Cette valeur permet au drone de se déplacer à 1.5 m/s le long de la trajectoire, puis de ralentir progressivement jusqu'à 0 m/s à 3 mètre d'un waypoint.

Maintenant que le premier coefficient est fixé le second coefficient K_i peut être analysé. L'anayse se fait aux abords d'un point de passage, car en général c'est à ce moment qu'il peut y avoir des oscillations ou des corrections non-cohérentes (Figure 2.38). Comme précédemment, plusieurs tests sont effectués en variant la valeur de K_i :



(a) $K_p = 0.4, K_i = 0.0, K_d = 0.0$

(b) $K_p = 0.4, K_i = 0.01, K_d = 0.0$



(c) $K_p = 0.4, K_i = 0.05, K_d = 0.0$

(d) $K_p = 0.4, K_i = 0.1, K_d = 0.0$

FIGURE 2.38 – Impact du coefficient K_i sur les vitesses de correction appliquées au drone ; Zoom aux niveau d'un point de passage.

La valeur de 0.05 est conservée pour le coefficient K_i (Figure 2.38c). Cette valeur permet de lisser la trajectoire à l'approche des points de passage.

Le dernier coefficient K_d est quant à lui laissé à 0.0 étant donné qu'il n'y pas de dépassement ni d'oscillation autour de la consigne.

2.5.6 Gestion de l'orientation du drone

Maintenant que le drone est capable de se déplacer selon un plan de vol, il reste à gérer l'orientation de celui-ci. L'objet principal de ce développement est d'automatiser la prise de vue pour l'auscultation des barrages à l'aide de plans de vol verticaux. Lors de la capture des images la caméra du drone doit pouvoir être orientée perpendiculairement à l'objet à relever, comme illustré sur la Figure 2.39. Le plan de vol devra définir une orientation de prise de vue pour chaque point de la trajectoire.

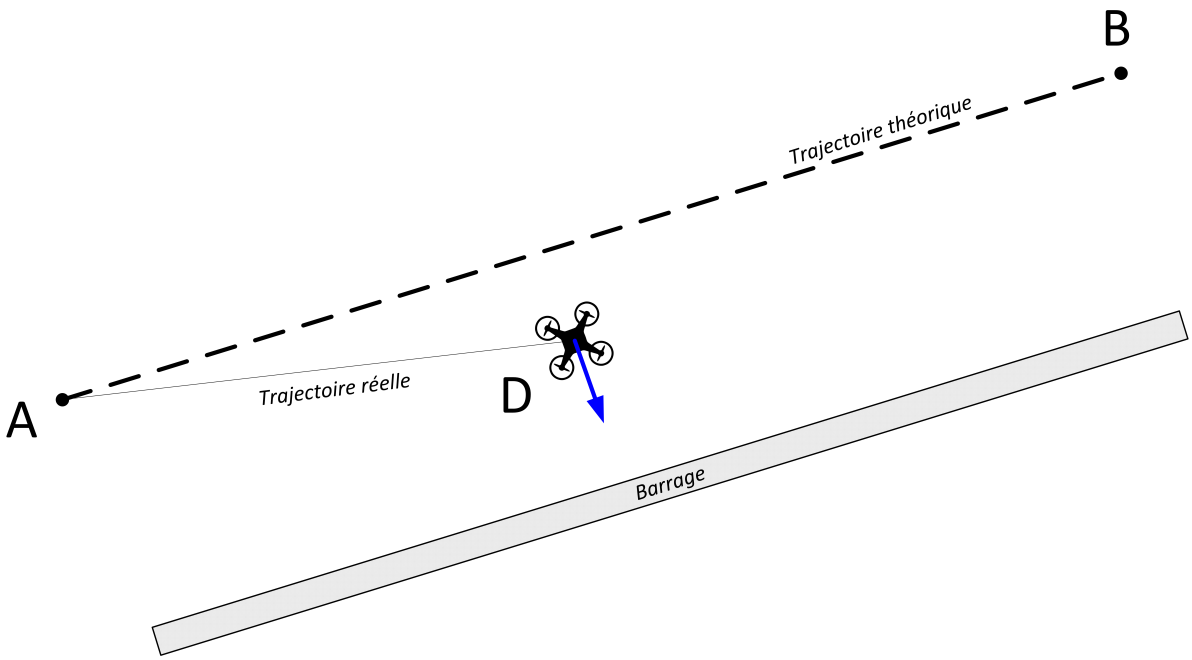


FIGURE 2.39 – Orientation du drone lors des prises de vues ; Vecteur bleu = axe principal du drone, vision de la caméra.

La gestion du cap se fait également à l'aide d'un PID. Entre chaque point de prise de vue, le drone doit ajuster son cap afin d'avoir la bonne orientation une fois arrivé au point suivant.

$$u_\psi(t) = K_{p\psi} e_\psi(t) + K_{i\psi} \int_0^t e_\psi(\tau)\mathrm{d}\tau + K_{d\psi} \frac{\mathrm{d}e_\psi(t)}{\mathrm{d}t} \tag{2.29}$$

L'erreur e_ψ correspond à la différence entre le cap du drone et le cap à atteindre.

La correction u_ψ correspond à la correction angulaire à appliquer sur le cap du drone.

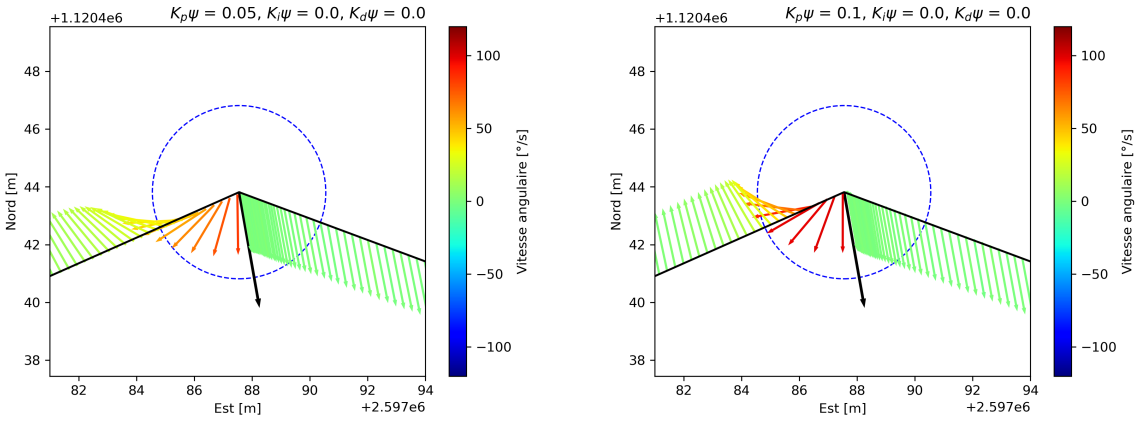
Les coefficients $K_{p\psi}, K_{i\psi}, K_{d\psi}$ ne sont pas les mêmes que ceux utilisés précédemment pour corriger la trajectoire du drone.

La détermination de ces coefficients se fait également par une simulation, la modélisation du cap du drone est la suivante :

$$\psi(t) = \psi(t-1) + u_\psi(t) \tag{2.30}$$

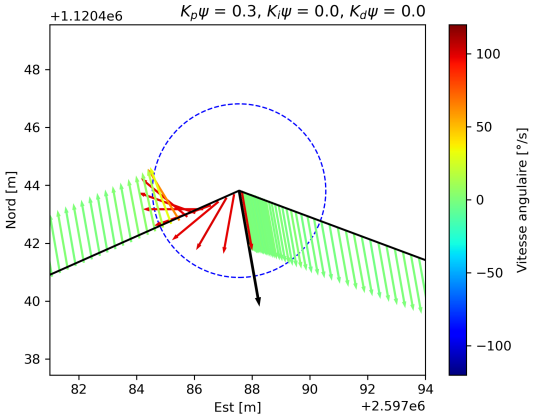
Comme pour le PID gérant le déplacent du drone, il y a des contraintes physiques et mécaniques à respecter pour les corrections angulaires appliquées sur le cap. Pour le drone DJI Matrice 210 V2, la vitesse angulaire maximale est de 120 °/s. Un simple test est effectué pour savoir si la correction demandée au drone est supérieure à la vitesse angulaire maximale. Si oui, la correction est diminuée pour respecter la vitesse maximale.

Pour déterminer la valeur de $K_{p\psi}$, plusieurs simulations sont effectuées et les vitesses angulaires de corrections le long de la trajectoire sont représentées sur la Figure 2.40 :

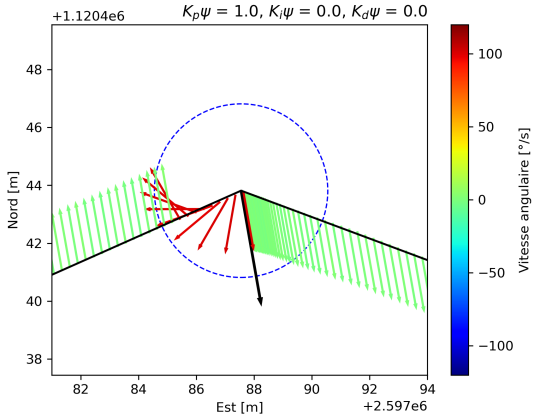


(a) $K_{p\psi} = 0.05, K_{i\psi} = 0.0, K_{d\psi} = 0.0$

(b) $K_{p\psi} = 0.1, K_{i\psi} = 0.0, K_{d\psi} = 0.0$



(c) $K_{p\psi} = 0.3, K_{i\psi} = 0.0, K_{d\psi} = 0.0$



(d) $K_{p\psi} = 1.0, K_{i\psi} = 0.0, K_{d\psi} = 0.0$

FIGURE 2.40 – Impact du coefficient $K_{p\psi}$ sur les vitesses angulaires des corrections de cap appliquées au drone ; Cercle traitillé bleu = zone à 3 m d'un waypoint

Le coefficient $K_{p\psi}$ retenu est 0.3 (Figure 2.40c). Il permet de corriger fortement le cap sur peu de distance tout en effectuant un ralentissement lorsque la consigne est approchée. Dans la simulation de la Figure 2.40 un changement de cap de près de 180° est effectué en moins de 3 mètre. En pratique il sera rare de devoir appliquer des changements de cap aussi importants si le plan de vol est optimisé.

Les coefficients $K_{i\psi}$ et $K_{d\psi}$ sont laissés à 0.0 étant donné que le résultat obtenu est suffisant.

Sur l'ensemble du plan de vol, le résultat suivant est obtenu :

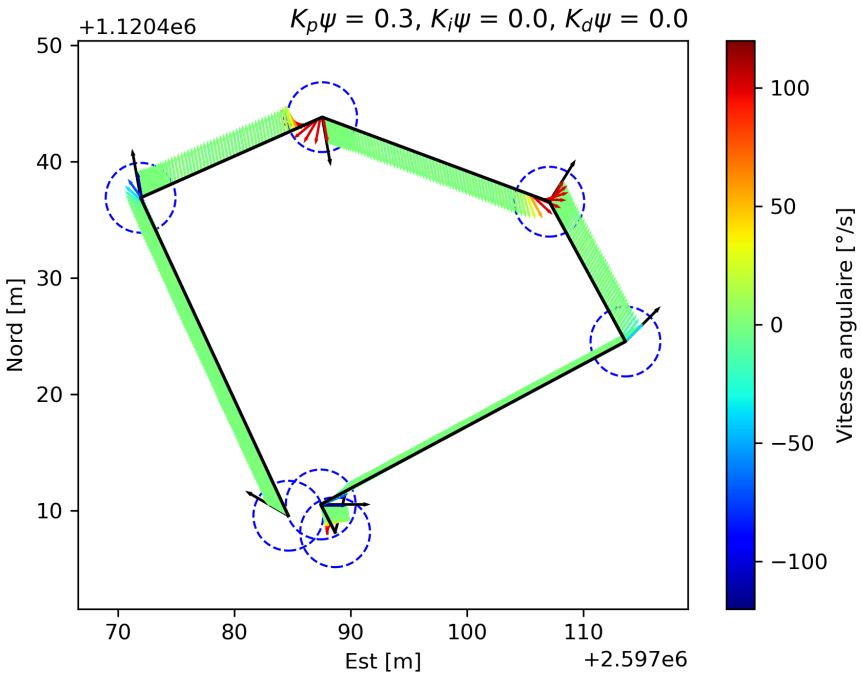


FIGURE 2.41 – Affichage du cap et des valeurs angulaires de correction le long de le trajectoire à partir des données simulées.

A plus de 3 mètres d'un point de passage le cap est déjà totalement corrigé, ce qui permet au drone d'arriver au point de passage suivant avec son orientation qui respecte déjà le plan de vol pour effectuer sa prise de vue sans devoir marquer un temps de pause. Les rotations du drone sont également optimisées afin d'appliquer la correction la plus petite sur le drone. Le drone peut effectuer une rotation à gauche ou à droite autour de son axe Z selon le cap à atteindre.

2.5.7 Valeurs retenues

Voici un récapitulatif des valeurs retenues pour les différents coefficients suite aux différents tests des chapitres précédents :

Valeur des coefficients	
K_p	0.40
K_i	0.05
K_d	0.00
$K_{p\psi}$	0.30
$K_{i\psi}$	0.00
$K_{d\psi}$	0.00

TABLE 2.2 – Valeurs retenues pour les coefficients K

2.5.8 Implémentation en Java et calcul en live

Une fois le processus de calcul des corrections validé par les différentes simulations précédentes, l'ensemble est implanté en Java pour pouvoir effectuer des tests en live.

Calcul de la position de la caméra

Actuellement les coordonnées déterminées par le filtre de Kalman correspondent au centre de gravité du drone. Si ces coordonnées sont directement utilisées pour le calcul du PID, cela veut dire que ce sera le centre du drone qui suivra la trajectoire donnée. Cependant, il est plus intéressant que ce soit la caméra qui suive le plan de vol.
Le principe similaire que celui expliqué au Chapitre 2.4.5 peut être appliqué pour déterminer les coordonnées de la caméra à chaque époque du Kalman.

Le bras de levier mesuré entre la caméra et le centre du drone dans le système de coordonnées du drone est le suivant (en centimètres) :

$$Levier_{drone-camera} = \begin{pmatrix} 14.0 \\ -7.5 \\ 14.0 \end{pmatrix} \tag{2.31}$$

En appliquant la matrice de rotation de l'Équation 2.24 pour chaque époque, un nouveau bras de levier est calculé avec la formule suivante :

$$\mathbf{b}_t = \mathbf{R}^{\mathbf{II}}_{\mathbf{I}\ t} \cdot Levier_{drone-camera} \tag{2.32}$$

Puis les coordonnées de la caméra sont calculées de la manière suivante :

$$\mathbf{P}_{camera_t} = \mathbf{P}_{kalman_t} + \mathbf{b}_t \tag{2.33}$$

Avec $\mathbf{P}_{kalman_t}$: Position calculée par le filtre de kalman au centre du drone à chaque époque.

Il est ainsi possible d'obtenir les coordonnées de la caméra à chaque état calculé par le filtre de Kalman.

Intégration d'un plan de vol

L'intégration du plan de vol dans l'application se fait à l'aide d'un fichier .txt contenant les informations suivantes :

- **Colonne 1** : Coordonnée Est du waypoint ;
- **Colonne 2** : Coordonnée Nord du waypoint ;
- **Colonne 3** : Altitude du waypoint ;
- **Colonne 4** : Gisement de la caméra pour la prise de vue au waypoint ;
- **Colonne 5** : Pitch de la caméra pour la prise de vue au waypoint ;

Ci-dessous un exemple de fichier pour le plan de vol :

```
1 2599841.15 1132924.00 1761.60 339.2176 0
2 2599836.76 1132922.34 1761.60 328.4177 45
3 2599820.58 1132912.39 1761.60 317.0859 90
4 2599807.19 1132899.94 1761.60 308.1767 0
```

Une boîte de dialogue permet d'aller sélectionner un fichier stocké sur le smartphone. Une fois importé, le plan de vol est représenté sur la carte.

Une option est ajoutée pour permettre de commencer le guidage du drone sur un waypoint défini par l'utilisateur. Cette option est essentielle pour les acquisitions qui nécessitent d'effecuter des changements de batterie au milieu du vol. Le dernier point de passage atteint est enregistré et proposé comme valeur par défaut de point de départ pour le vol suivant.

Application des corrections sur le drone

Le guidage du drone en live est effectué selon le schéma ci-dessous (Figure 2.42) :

1. Le bras de levier drone/caméra est calculé pour obtenir la coordonnée de la caméra (Chapitre 2.5.8) ;
2. Si le drone est en-dessous du seuil de distance fixé par rapport au waypoint : le drone reste en stationnaire pendant X secondes (fixé par l'utilisateur dans l'application) puis passe au point suivant ;
3. Sinon, les corrections de vitesses et de cap sont envoyées et appliquées sur le drone ;
4. Les commandes sont envoyées à une fréquence de 20 Hz au drone (Fréquence limitée par DJI).

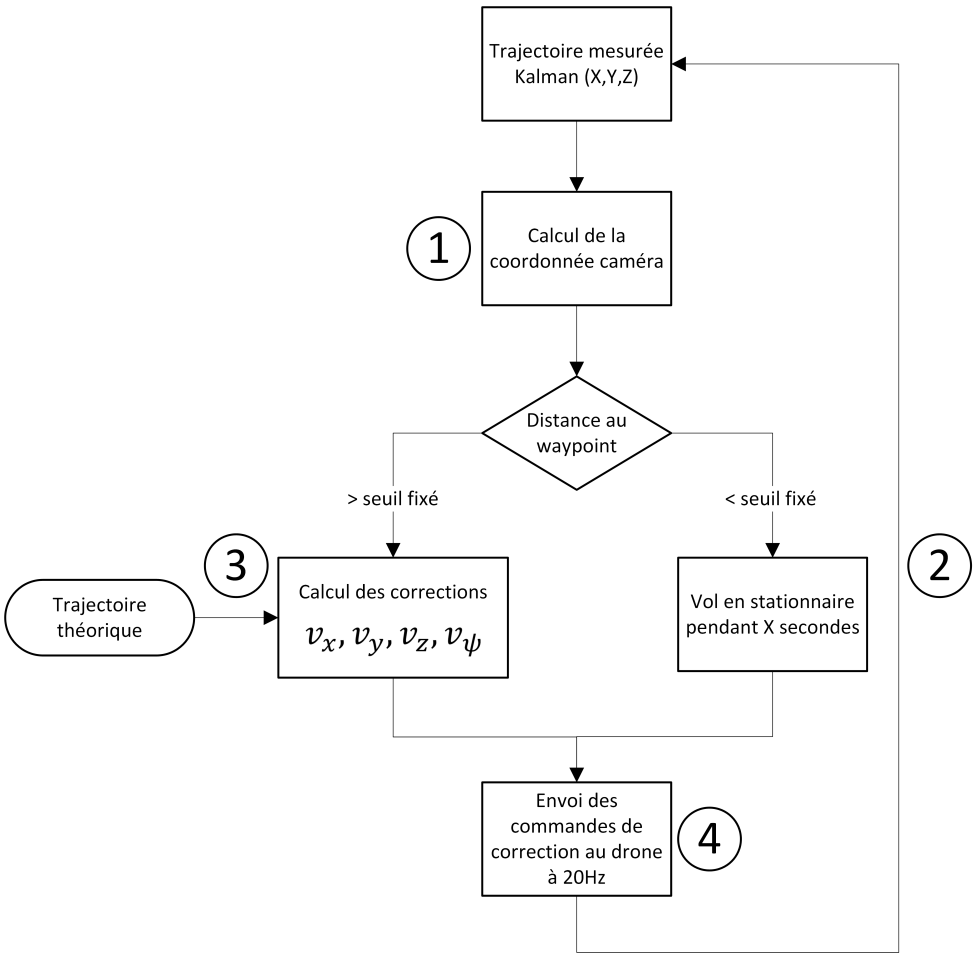


FIGURE 2.42 – Affichage du cap avec les données simulées.

2.6 Déclenchement des prises de vue

Maintenant que le drone est capable de suivre une trajectoire donnée, il reste à traiter la capture des images. Deux modes de capture sont implantés dans l'application :

1. **Capture aux waypoints** : Une image est capturée sur chaque waypoint défini dans le plan de vol (Figure 2.43). Il peut être intéressant d'utiliser cette méthode si les waypoints ne sont pas alignés ou si des orientations de caméra différentes sont nécessaires le long d'une trajectoire ;
2. **Capture selon un intervalle** : Une image est capturée selon un intervalle de distance définie par l'utilisateur ainsi que sur chaque waypoint. Cette option peut-être intéressante pour gagner du temps lors de l'auscultation d'objets linéaires. Au lieu de placer des waypoints chaque 5 mètres par exemple, seul le début et la fin de la trajectoire peuvent être définis. Ensuite des images sont capturées à un intervalle régulier entre ces deux waypoints. Comme vu précédemment un arrêt est marqué sur chaque waypoint, en diminuant le nombre de waypoints il est alors possible de diminuer considérablement le temps d'arrêt du drone (Figure 2.44).

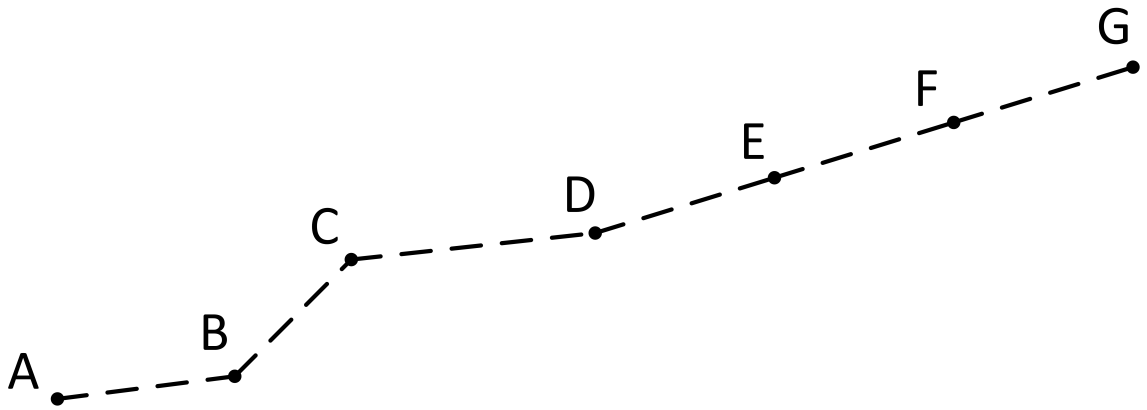


FIGURE 2.43 – Capture aux waypoints, capture des images sur chaque waypoint.

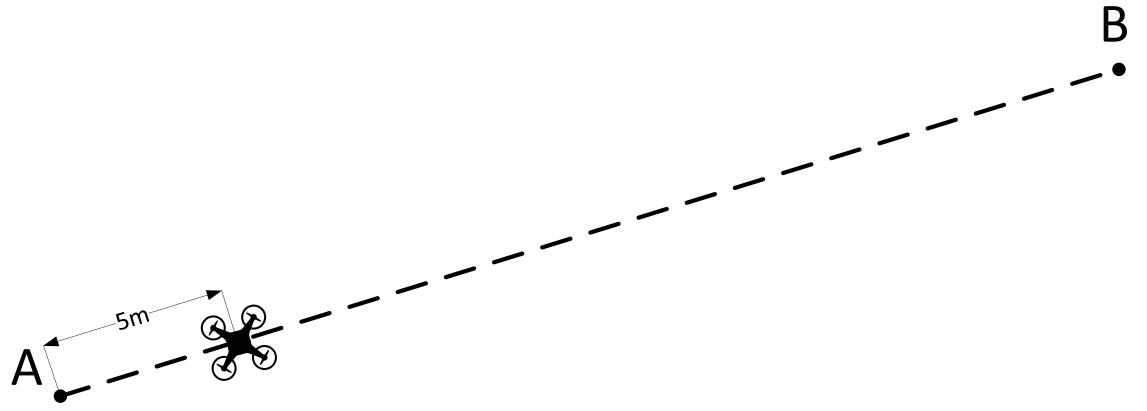


FIGURE 2.44 – Capture selon un intervalle défini, capture des images selon un intervalle de distance.

Pour les deux modes de capture d'images, le Pitch de la caméra est géré avec le gimbal. Quand un waypoint est atteint, le gimbal est orienté selon les valeurs définies dans le plan de vol. Une fois le gimbal orienté, l'image est capturée.

Un marker est également ajouté sur la carte à chaque emplacement de prise de vue.

Le processus détaillé pour la capture des images selon un intervalle donné est le suivant (Figure 2.45) :

1. L'ensemble du processus est effectué dans la boucle du PID, donc à 20Hz ;
2. Calcul de la distance par rapport à la dernière image capturée ;
3. Test pour savoir si l'intervalle est atteint ;
4. Test pour savoir s'il y a plus de 2 secondes par rapport à la dernière image capturée ;
5. Si le temps est de moins de 2 secondes, le drone marque une pause dans son déplacement et reste en stationnaire. L'intervalle de temps entre deux captures d'images ne peut pas être de moins de 2 secondes (Contrainte technique). Une fois la pause effectuée pour assurer un intervalle de plus de 2 secondes la capture de l'image est effectuée ;

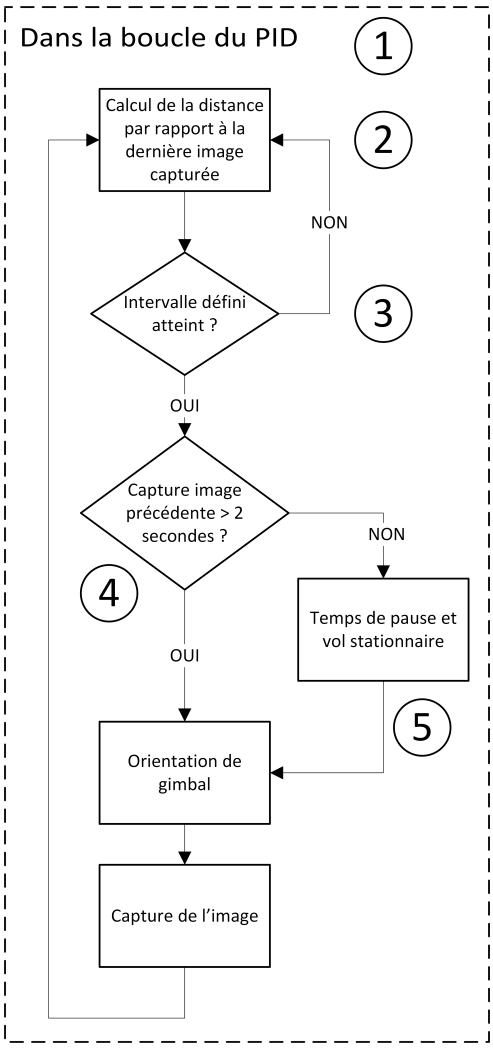


FIGURE 2.45 – Processus de capture des images selon un intervalle défini.

Suite à différents tests, un décalage temporel a été constaté entre le moment où la fonction de capture d'images est appelée et le moment où l'image est réellement capturée. Pour déterminer ce décalage, une série d'images sont capturées à différentes vitesses. Pour chacune de ces images, la position ainsi que le temps sont enregistrés au moment d'appeler la fonction et au moment où l'image est capturée. Il est ainsi possible de calculer un décalage en centimètre ainsi qu'un décalage temporel en tenant compte de la vitesse instantanée du drone. le résultat des différents tests est visible sur le graphique ci-dessous (Figure 2.46) :

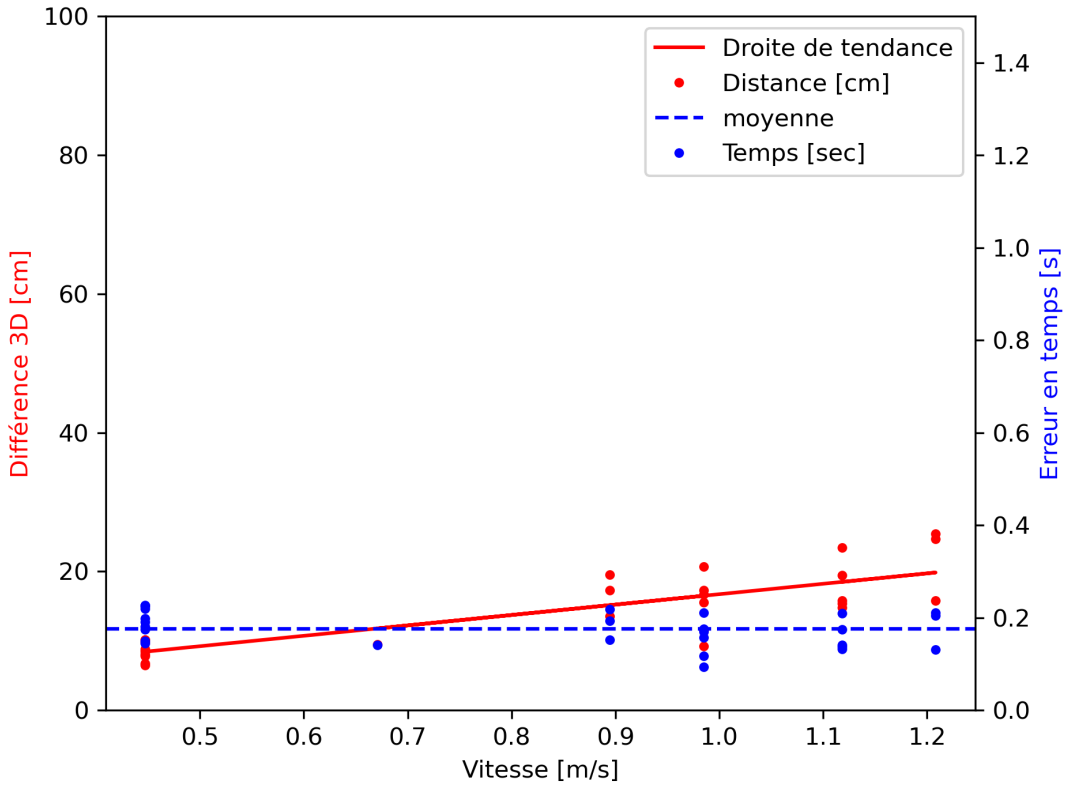


FIGURE 2.46 – Décalage entre l'appel de la fonction pour effectuer une capture d'image et le moment où l'image est réellement capturée.

Le décalage temporel est relativement stable entre les différentes vitesses testées. Le décalage temporel moyen déterminé est de 0.19 secondes avec un écart type de 0.03 secondes. Si un autre drone est utilisé, il faudrait déterminer ce décalage temporel.

Lorsque des images sont uniquement capturées sur les waypoints, ce décalage n'est pas visible, étant donné que le drone effectue un vol stationnaire de plusieurs secondes. Par contre, il est important de prendre en compte ce décalage lorsque des images sont capturées à intervalles réguliers, étant donné que le drone est en mouvement. Sans appliquer de correction, les images sont capturées "trop tard" et l'intervalle défini par l'utilisateur n'est plus respecté.

En fonction de la vitesse du drone et de la constante de décalage de 0.19 secondes, il est possible de déterminer une correction en centimètre à appliquer sur les intervalles à respecter. Par exemple, à 1 m/s une différence de 19 cm se produit. Si l'intervalle défini est de 3 mètres, le drone effectuera la capture de l'image à 2.81 (3 - 0.19) mètres pour compenser le retard dans l'acquisition de l'image.

2.6.1 Position des images

Chaque fois qu'une image est capturée, sa position déterminée par le filtre de Kalman et les différents bras de levier est enregistrée dans un fichier .txt.

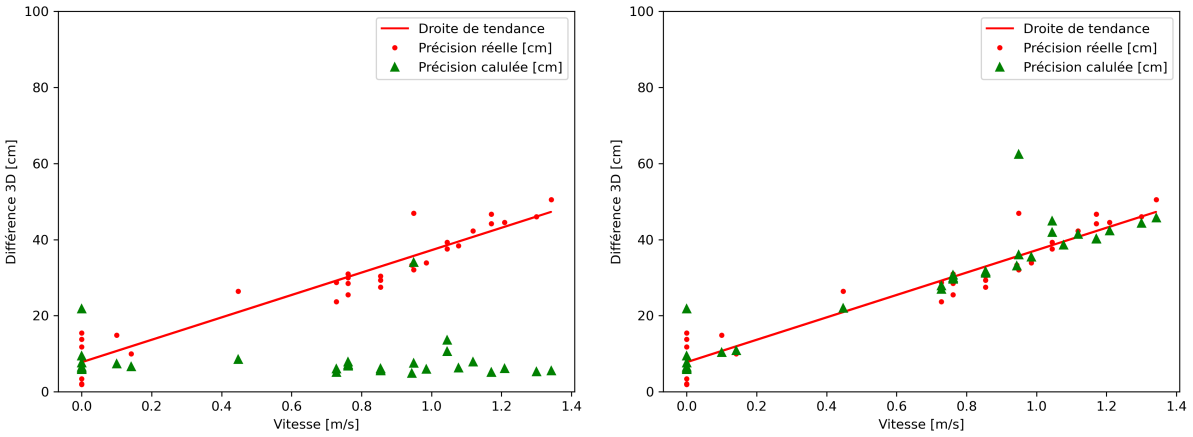
Une précision est également associée à chaque clichés. Cette précision est calculée à partir des précisions retournées par le filtre de Kalman à chaque époques. Afin de déterminer la fiabilité de cette précision calculée lors de l'enregistrement des images, une comparaison est effectuée entre la précision calculée et la précision réelle. La précision réelle correspond à l'écart entre la position des caméras déterminée par aérotriangulation et la position enregistrée par le script lors du vol drone.

Le résultat de la Figure 2.47a est obtenu sur une série de caméras à différentes vitesses en utilisant uniquement les précisions renvoyées par le filtre de Kalman.

Les précisions retournées par le filtre de Kalman sont trop optimistes. L'écart entre la précision réelle et la précision du filtre de Kalman correspond à une constante de temps qui peut être appliquée selon la vitesse de déplacement du drone pour obtenir une correction supplémentaire. Il est alors possible de calculer une nouvelle précision en tenant compte de la précision du filtre de Kalman ainsi que d'une composante liée à la vitesse de déplacement du drone. Cette constante de temps est déterminée à 0.3 seconde. En appliquant cette constante supplémentaire, le résultat de la Figure 2.47b est obtenu. Les précisions calculées correspondent, en moyenne, maintenant à ± 5 cm des précisions réelles. Il reste cependant quelques clichés présentant des divergences plus importantes, il est donc difficile de définir finement une précision pour chaque image. Des analyses avec un nombre plus important d'images sont disponibles au Chapitre 3.2.4.

La formule finale de la précision calculée en live :

$$precision_{live} = precision_{kalman} + vitesse_{drone} * 0.3 \tag{2.34}$$



(a) Précision kalman

(b) Précision kalman + facteur en fonction de la vitesse

FIGURE 2.47 – Différence entre la précision calculée et la précision réelle.

2.7 Interface graphique

Le développent d'une interface graphique est nécessaire afin de présenter les différentes fonctions de manière ergonomique. L'application Android développée en Java est composée des pages suivantes :

- Page d'accueil (Chapitre 2.7.1) ;
- Page principale (Chapitre 2.7.2) ;
- Page de gestion des paramètres (Chapitre 2.7.3) ;
- Page de connexion à l'instrument (Chapitre 2.7.4).

Le fonctionnement de l'application est expliqué en Annexe B.1.

2.7.1 Page d'accueil

Cette page est affichée à l'ouverture de l'application (Figure 2.48). Elle permet de valider la bonne communication avec le drone. Ainsi, si la communication avec la télécommande et le drone est valide, le modèle du drone est affiché à la place de "Information sur le drone", le "status" change et le bouton "OUVRIR" devient cliquable. Si l'application veut être ouverte sans un drone connecté, le bouton "Continuer sans drone" est disponible.

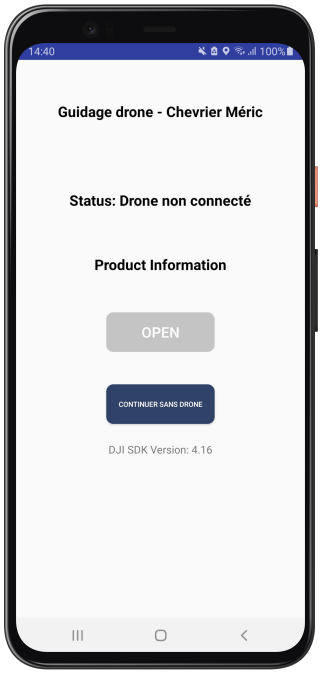


FIGURE 2.48 – Page d'accueil de l'application.

2.7.2 Page principale

C'est à partir de cette page que l'ensemble des fonctions de guidage sont effectuées (Figure 2.49). C'est également à partir de cette page que les différentes pages supplémentaires sont ouvertes. Une représentation cartographique permet de représenter la trajectoire calculée en live par le filtre de Kalman. La représentation cartographique se fait grâce à l'API de Google maps. Plusieurs représentions sont faites sur cette carte :

- **Position de l'opérateur** : Les coordonnées GPS du smartphone sont récupérées et un point (cercle bleu) est ajouté sur la carte. Au démarrage, la carte est automatiquement zoomée sur l'emplacement du smartphone ;
- **Position du drone** : La position issue du filtre de Kalman ainsi que l'orientation du drone sont représentées par un symbole d'avion rouge ;
- **Historique de la trajectoire** : Pour un soucis de lisibilité de la carte, seul les 500 dernières époques du filtre de Kalman sont représentées par une polyline bleue ;
- **Trajectoire à suivre** : Il est également possible d'afficher le plan de vol sur la carte ;
- **Emplacement des prises de vue** : Lorsqu'une image est capturée, un point rouge est rajouté sur la carte.

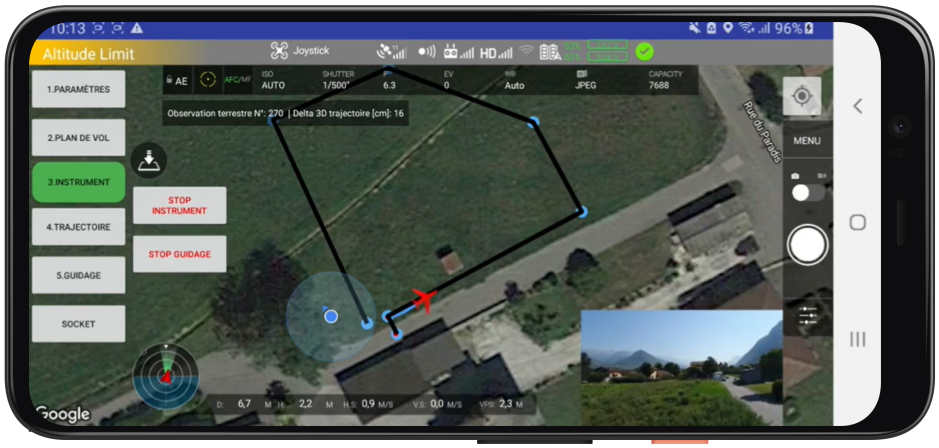


FIGURE 2.49 – Page principale de l' application - Intégration Google Maps

L'API de Google fonctionnant avec des coordonnées WGS84, les coordonnées calculées en MN95 sont transformées à la volée pour pouvoir être représentées sur la carte à l'aide de la formule approchée de transformation [6].

Un retour live de la caméra du drone est également visible directement sur cette page.

Le UX SDK [7] de DJI permet d'intégrer des bloc de fonctionnalité prêt construit pour la gestion des paramètres du drone et de la caméra.

Les détails de fonctionnement de l'applications sont expliqués en Annexe B.1.

2.7.3 Page de gestion des paramètres

Ne nombreux paramètres sont modifiables par l'utilisateur (Voir l'Annexe B.1 pour plus de détails). Le nombre important de paramètres nécessite la création d'une page dédiée à la modification de ceux-ci. Il est possible de faire défiler cette page vers le bas pour rajouter autant de paramètres que nécessaire (Figure 2.50). Les derniers paramètres entrés par l'utilisateur sont sauvegardés sur le smartphone et sont affichés comme valeur par défaut.

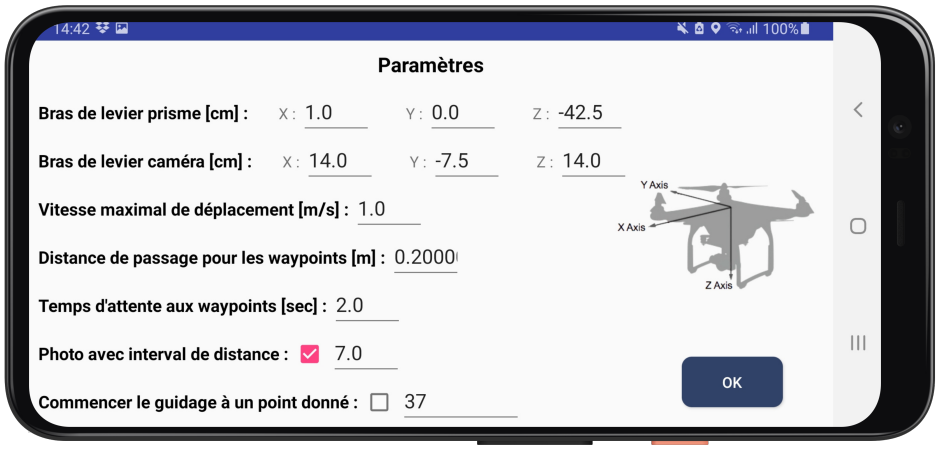


FIGURE 2.50 – Page de gestion des paramètres.

6. https://www.swisstopo.admin.ch/content/swisstopo-internet/fr/topics/survey/reference-systems/switzerland/_jcr_content/contentPar/tabs/items/dokumente_publikatio/tabPar/downloadlist/downloadItems/516_1459343097192.download/ch1903wgs84_f.pdf
7. https://developer.dji.com/ux-sdk/

2.7.4 Page de connexion à l'instrument

Cette page permet de sélectionner le périphérique bluetooth qui correspond à la station totale qui doit être utilisé pour effectuer les mesures terrestres (Figure 2.51).

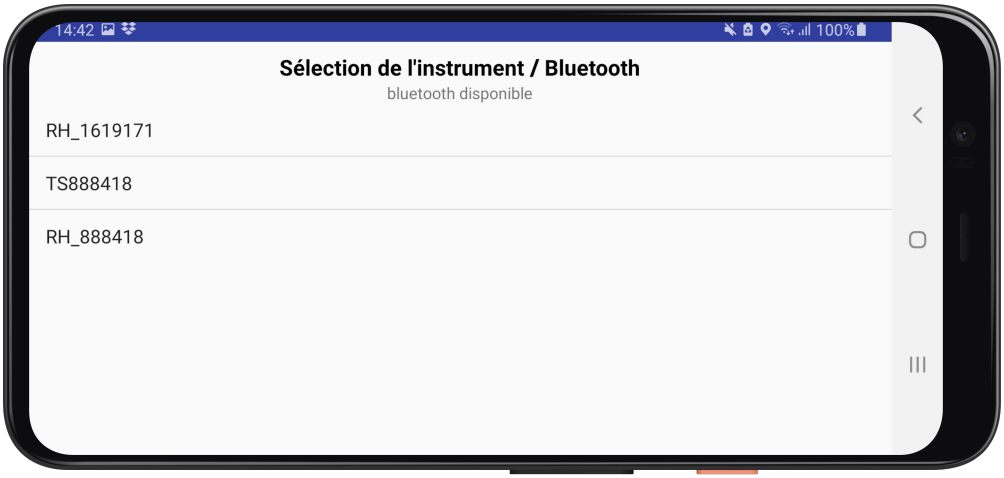


FIGURE 2.51 – Page de connexion à l'instrument.

Chapitre 3

Résultats et expérimentations

Le but de ce chapitre est de tester les différents développements effectués en effectuant des vols réels. Dans un premier temps, des tests sont effectués dans un environnement contrôlé (Chapitre 3.1), puis des tests seront effectués dans un environnement de type "barrage" (Chapitre 3.2). Les différents résultats de ces tests ont pour objectif de valider le bon fonctionnement de l'ensemble du système et d'en définir ses limites.

3.1 Qualification du guidage

Cette première phase de test à pour but de valider le fonctionnement du guidage du drone en live et si nécessaire d'appliquer des corrections sur le code essentiellement développé par simulation pour le moment. Dans un premier temps, les vitesses de déplacement du drone sont analysées (Chapitre 3.1.1). Ensuite la précision du positionnement ainsi que les paramètres de prises de vue seront étudiés (Chapitre 3.1.2 à 3.1.4). Pour finir, le suivi du plan de vol sera analysé (Chapitre 3.1.5).

3.1.1 Vitesse de déplacement

Afin de valider le bon fonctionnement du PID en ce qui concerne les vitesses appliquées sur le drone, le graphique ci-dessous est calculé à partir des vitesses réelles mesurées par le drone (Figure 3.1).

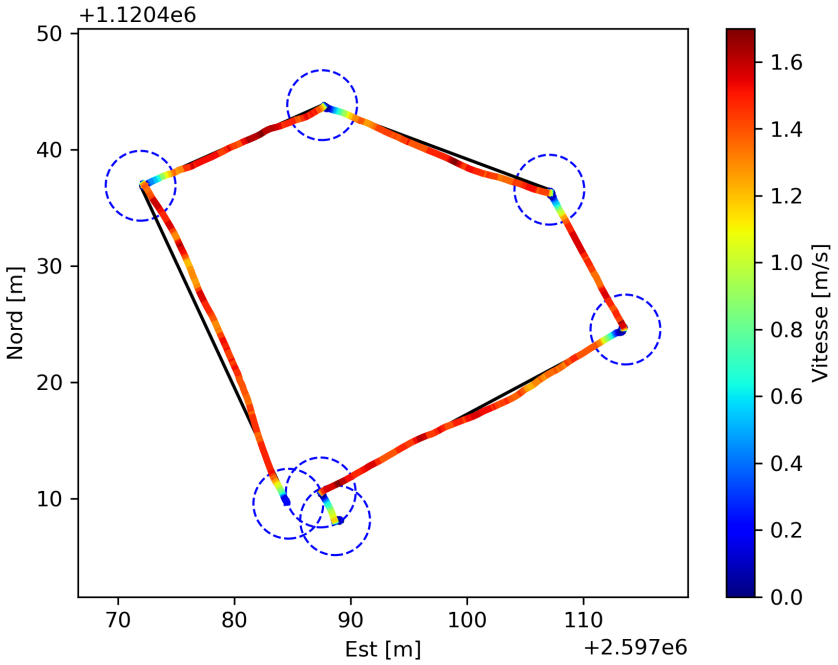


FIGURE 3.1 – Vitesse de déplacement mesurée le long de la trajectoire ; Sens du plan de vol = contraire aux aiguilles d'une montre.

La vitesse maximale qui était fixée à 1.5 m/s est bien respectée et le drone ralenti à environ 3 mètres des points de passage. La gestion des vitesses respecte donc bien ce qui avait était prévu lors de l'élaboration du PID dans le Chapitre 2.5.1.

Les vitesses de corrections appliquées sur le drone sont donc validées.

3.1.2 Précision du positionnement des prises de vue

Afin de déterminer la précision du positionnement du drone, des prises de vue sont effectuées à un intervalle régulier de 3 mètres sans effectuer d'arrêt du drone. Sur l'ensemble de la zone acquise des points de calage sont mesurés au sol selon la représentation ci-dessous (Figure 3.2).

FIGURE 3.2 – Zone de test pour la qualification du guidage ; Points blancs : points de calage déterminés à la station totale ; Ligne rouge : plan de vol.

La qualification de la précision du guidage se fait à partir des coordonnées des caméras. Celle-ci sont déterminées de deux manières indépendantes :

— **A partir de l'aérotriangulation** : ces coordonnées sont considérées comme la valeur "doit" de la position des caméras ;
— **A partir du filtre de Kalman** : A chaque instant il est possible de déterminer la position de la caméra à l'aide du bras de levier ;

Des acquisitions sont effectuées à différentes vitesses, à savoir : 0.2 m/s, 0.5 m/s, 1.0 m/s, 1.2 m/s et 1.5 m/s.

Pour chaque caméra il est ainsi possible de calculer :

— **Un delta distance** : Ce delta correspond à la distance 3D entre la position de caméra calculée par le filtre de Kalman et la position calculée par l'aérotriangulation ;
— **Un delta temps** : En fonction des vitesses de vol instantanées, il est possible de convertir le delta distance en delta temps ;

Pour chaque vitesse testée, le delta distance moyen et le delta temps moyen sont calculés. Le résultat obtenu est visible sur le graphique de la Figure 3.3.

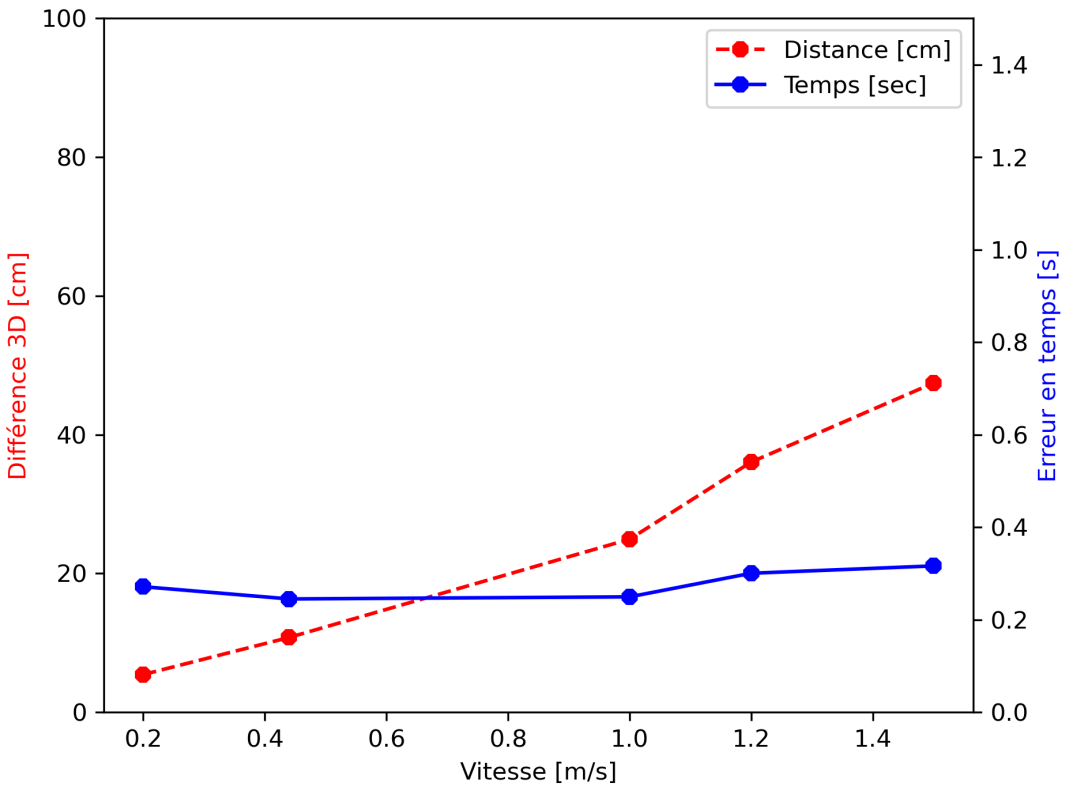
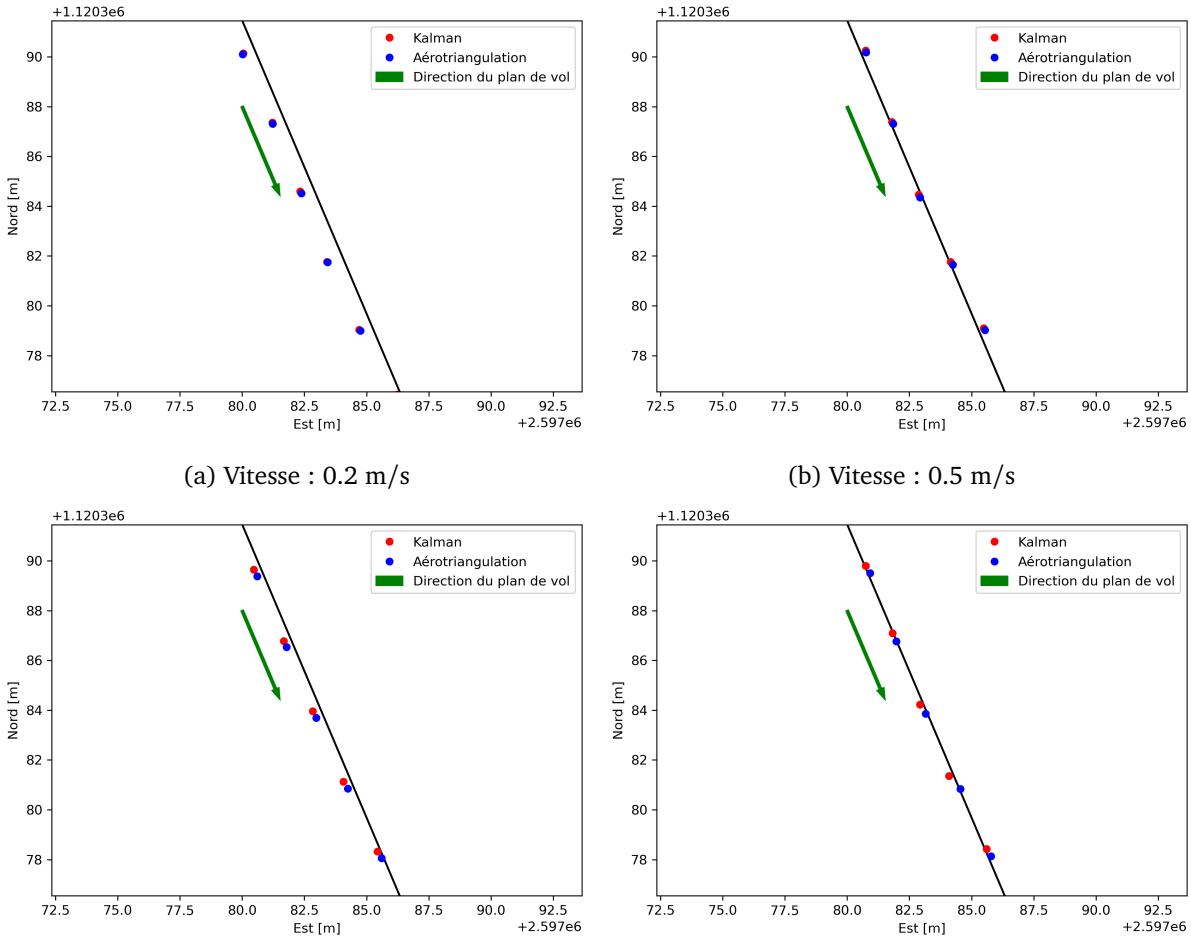


FIGURE 3.3 – Analyse des deltas distance moyen (rouge) et des deltas temps moyen (bleu) par vitesse.

Plus la vitesse de déplacement augmente, plus la différence en distance augmente. Par contre, le delta temps est relativement constant sur les différentes vitesses testées. L'erreur pourrait donc être modélisée par une constante de temps.

Il est également intéressant d'observer dans quelle direction cette différence est visible. Les résultats suivants sont obtenus (Figure 3.4) :



(a) Vitesse : 0.2 m/s

(b) Vitesse : 0.5 m/s

(c) Vitesse : 1 m/s

(d) Vitesse : 1.2 m/s

FIGURE 3.4 – Différence de position entre les caméras issues du filtre de Kalman (rouge) et les caméras calculées par aérotriangulation (bleu).

Pour toutes les vitesses testées, la différence entre les deux caméras se produit toujours le long de la trajectoire et dans le sens de déplacement du drone (flèche verte). Les caméras issues du filtre de Kalman sont toujours "en retard" par rapport aux caméras de l'aérotriangulation. Cette analyse confirme que l'erreur de positionnement peut être modélisée par une constante de temps le long de la trajectoire.

De nombreux tests ont été effectués pour déterminer le moment le plus pertinent pour enregistrer les positions des caméras lors du vol (Enregistrement du temps avant, après le déclenchement). Mais aucun résultat meilleur n'a pu être obtenu. La différence de temps d'environ 0.3 seconde provient en grande partie de l'incertitude de temps lors du déclenchement des images. En effet, il n'est pas possible de trigger précisément le déclenchement des prises de vue avec le SDK de DJI. Une fois la fonction de prise de vue lancée, il n'est pas possible de connaître exactement le moment de la capture de l'image.

Afin de déterminer le delta temps moyen qui pourrait être utilisé comme une constante, l'ensemble des caméras à toutes les vitesses sont utilisées en bloc. Le résultat ci-dessous est obtenu (Figure 3.5) :

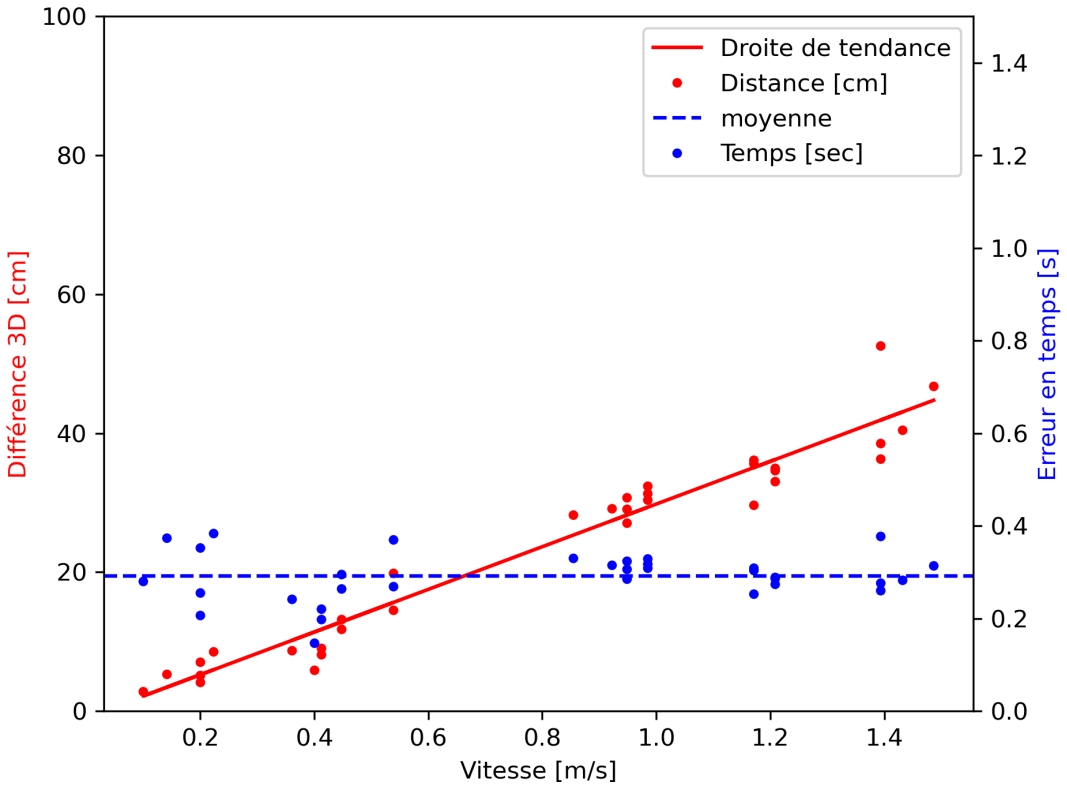


FIGURE 3.5 – Analyse des deltas distance (rouge) et des deltas temps (bleu) selon la vitesse.

Le premier constat est que les données issues des faibles vitesses sont assez bruitées. Un delta temps moyen de 0.29 seconde est obtenu avec un écart type de 0.05 seconde.

En effectuant le même calcul mais en gardant uniquement les vitesses de 1 m/s, 1.2 m/s et 1.5 m/s qui sont moins bruitées, le delta temps moyen est de 0.30 seconde avec un écart type de 0.02 seconde.

Cette constante de temps pourrait être utilisée en post-traitement pour recalculer les positions des caméras 0.3 secondes dans le "futur" le long de la trajectoire. Ainsi des positions plus proche de la réalité pourraient être introduites dans un logiciel de photogrammétrie pour les positions approchées des caméras.

En observant uniquement les delta Z sur les positions des caméras, le résultat suivant est obtenu (Figure 3.6) :

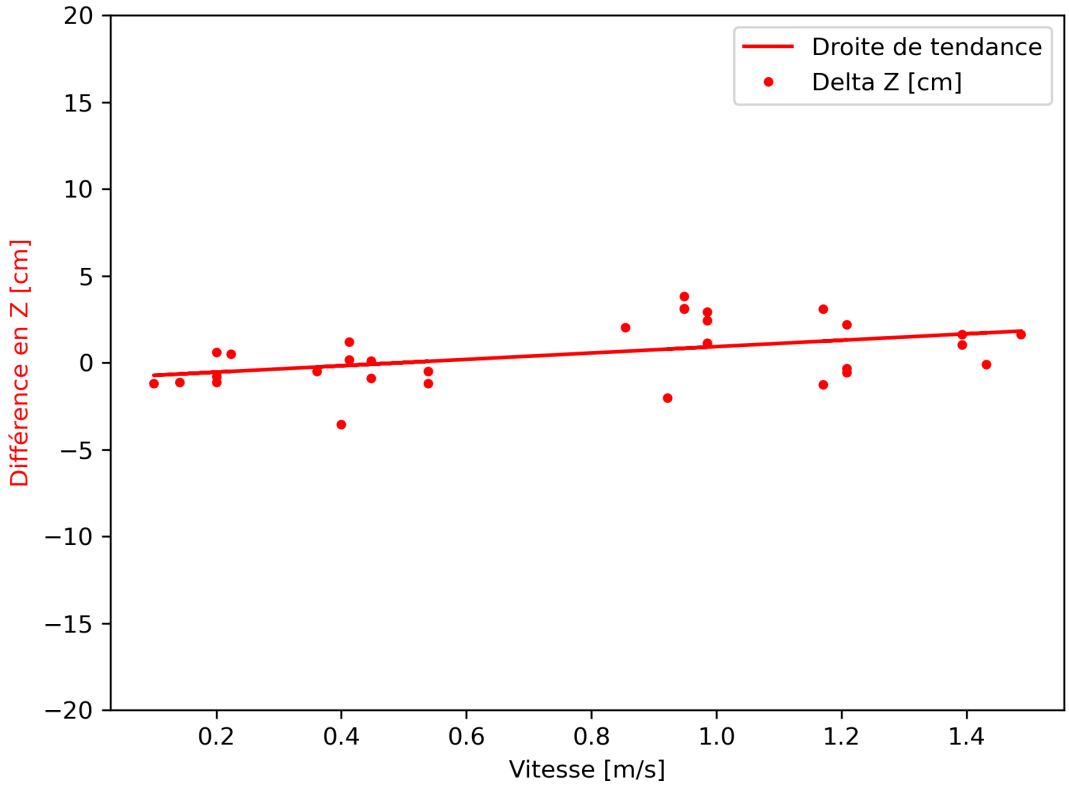


FIGURE 3.6 – Analyse des delta Z sur les caméras.

La trajectoire donnée par le plan de vol étant relativement plate, il est normal que l'erreur de temps n'impacte que légèrement les positions en Z des caméras. Des écarts de maximum 5 cm sont constatés.

Les différents résultats obtenus précédemment sont issus de prises de vue en mouvement (le drone effectue des prises de vue le long de la trajectoire donnée sans marquer d'arrêt). Il est possible d'obtenir un positionnement plus précis lorsque le drone est en stationnaire, ce qui est le cas lors des passages sur les waypoints. Sur ces derniers, le delta distance entre les caméras est d'environ 6 cm.

Suite à ces tests, la précision du positionnement des prises de vue peut être donnée à $0.3 * v_{drone}$. La précision est donc liée à une constante de temps qui peut être corrigée en post-traitement pour obtenir un positionnement plus précis des images. L'analyse des résultats obtenus en appliquant un post-traiement est effectuée au Chapitre 3.1.3.

3.1.3 Précision du positionnement des prises de vue en post-traitement

Comme mentionné au Chapitre 3.1.2, le décalage entre la position enregistrée par le drone et la position réelle des images peut être compensé en post-traitement. En effet, ce décalage est une constante de 0.30 seconde le long de la trajectoire mesurée.

Afin de valider ce principe, un post-traitement est effectué sur une série de mesures dans le but de recalculer la position des images 0.30 seconde dans le "futur" en fonction de la vitesse de déplacement du drone (les détails techniques sont disponibles en Annexe B.1.6). Cette position recalculée est ensuite comparée à la position réelle des images données par l'aérotriangulation. Les différences constatées sont les suivantes (Figure 3.7) :

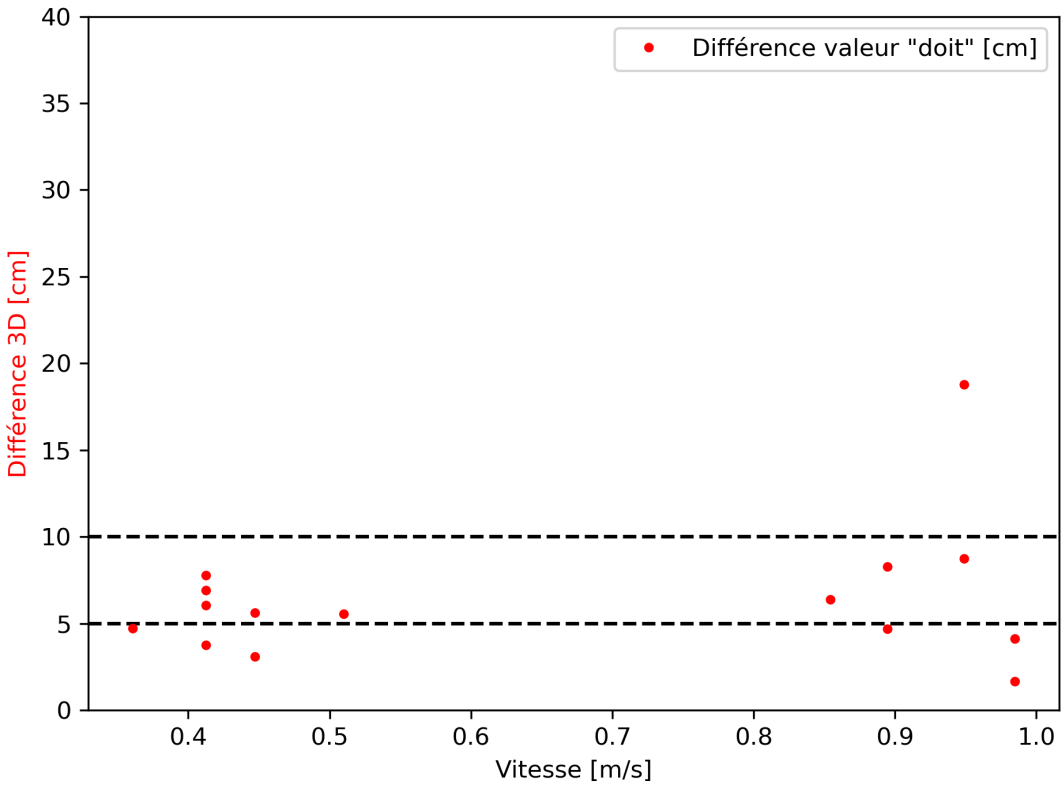


FIGURE 3.7 – Analyse des deltas distance entre les caméras issues de l'aérotriangulation et les caméras issues du drone suite au post-traitement ; traitillé en noir = seuil de 5cm et 10cm

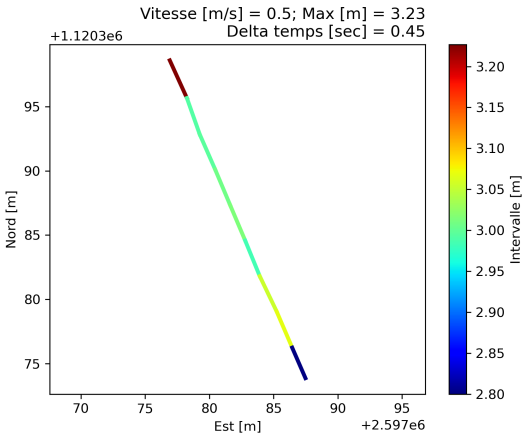
Les différences oscillent autour de 5 cm (premier traitillé noir) et quelques points présentent des différences plus importantes de l'ordre de 10 cm. Un point se démarque avec une différence de presque 20 cm, cela est dû à une dérive de la trajectoire calculée suite à une perte de mesure terrestre à la fin de la trajectoire.

En appliquant un post-traitement, il est donc possible d'obtenir des positions approchées d'une précision d'environ 10 cm pour les différentes images capturées. Ces positions pourront être introduites dans un logiciel de photogrammétrie.

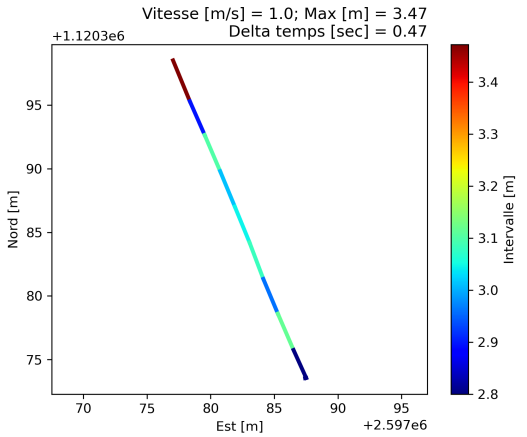
3.1.4 Précision des intervalles de prises de vue

Afin d'analyser le bon fonctionnement du mode de prises de vue par intervalle, la distance entre la position calculée par aérotriangulation de chaque images est analysée à différentes vitesses. Les résultats obtenus sont représentés sur la Figure 3.8.

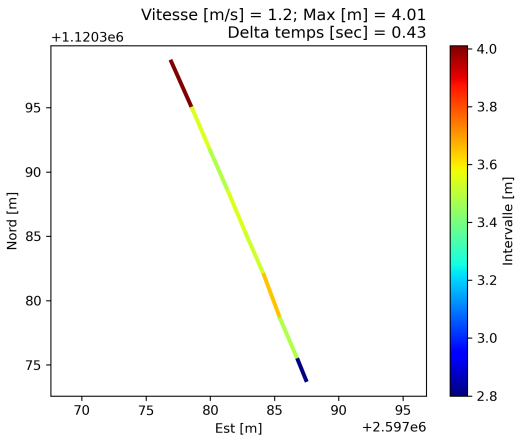
Pour les trois vitesses testées, il y a un écart sur le premier segment (segment en rouge foncé sur les Figures 3.8). Ce segment correspond au segment entre le waypoint et la première image capturée par intervalle. La longueur mesurée de ce premier segment est indiqué comme "Max" sur chaque graphique. Selon la vitesse de déplacement, l'erreur par rapport à l'intervalle défini est calculée en secondes (Indiqué sous "Delta temps [sec]" sur chaque graphique). Pour l'ensemble des vitesses testées, ce delta temps est d'environ 0.45 seconde. Le reste des segments respecte bien l'intervalle défini par l'utilisateur.



(a) Vitesse : 0.5 m/s ; Intervalle : 3 m



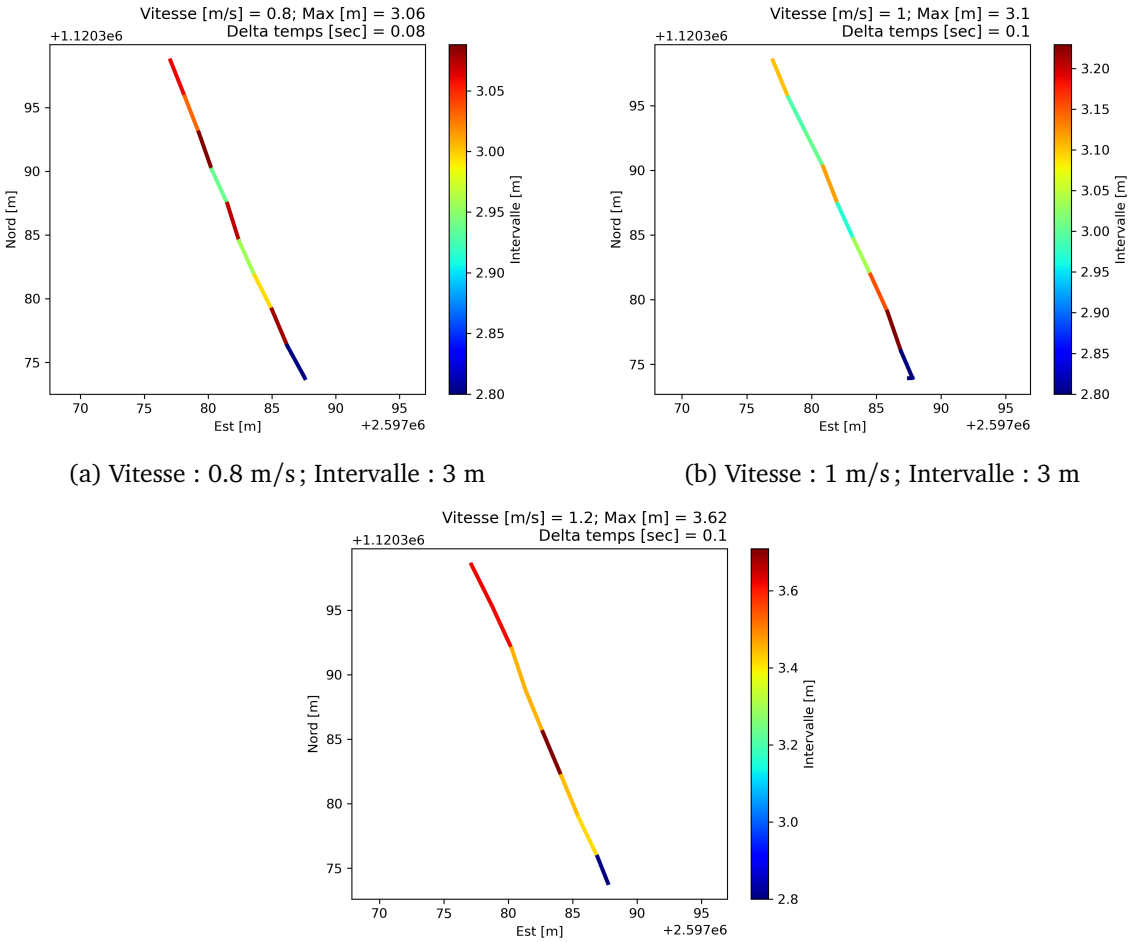
(b) Vitesse : 1 m/s ; Intervalle : 3 m



(c) Vitesse : 1.2 m/s ; Intervalle : 3.5 m

FIGURE 3.8 – Analyse des intervalles à partir des caméras, calculée par aérotriangulation

Ce delta de 0.45 seconde peut donc être intégré dans le script de déclenchement des images pour compenser le décalage. Le delta de 0.45 seconde est transformé en une distance selon la vitesse de déplacement du drone. Puis l'intervalle donné au drone pour déclencher la première image est réduit de cette distance afin de compenser le "retard" dans la capture de l'image. Cette compensation est uniquement appliquée pour la première image capturée après un waypoint. Les résultats suivants sont obtenus après avoir introduit la correction (Figure 3.9) :



(a) Vitesse : 0.8 m/s ; Intervalle : 3 m

(b) Vitesse : 1 m/s ; Intervalle : 3 m

(c) Vitesse : 1.2 m/s ; Intervalle : 3.5 m

FIGURE 3.9 – Analyse des intervalles à partir des caméras calculée par aérotriangulation en intégrant la correction de 0.45 secondes

Les résultats obtenus sont significativement meilleurs, les écarts constatés sont de 0.1 secondes ou 10 cm au maximum.

3.1.5 Analyse du suivi du plan de vol

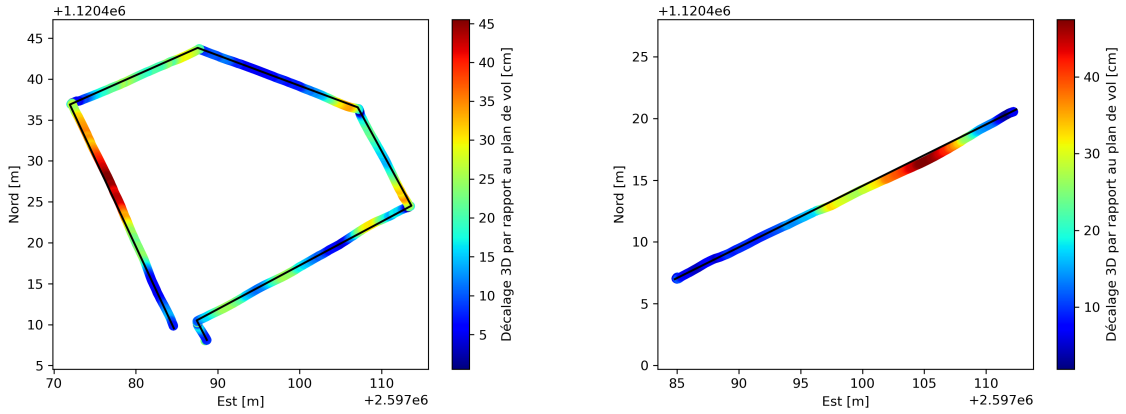
Afin de déterminer si le plan de vol donné est bien respecté, la distance orthogonale entre la trajectoire de la caméra et le plan de vol est calculée.

Au Chapitre 2.5.1, deux intégrations du PID ont été présentées :

— Le processus simple : les corrections sont calculées pour atteindre le waypoint suivant (Chapitre 2.5.2) ;
— Le processus avancé : Les corrections sont calculées pour atteindre un point virtuel qui se déplace avec le drone (Chapitre 2.5.3).

Ici, ces deux intégrations sont comparées pour déterminer si l'ajout d'un point virtuel est nécessaire et apporte une réelle plus-value.

Les résultats suivants sont obtenus pour deux plans de vols testés avec l'inégration "simple" sans prendre en compte un point virtuel (Figure 3.10) :

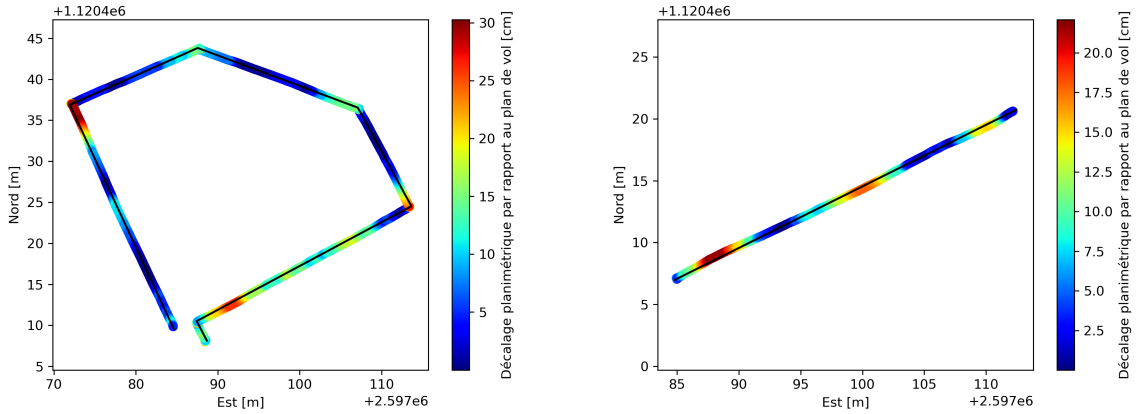


(a) Plan de vol 1 ; Vitesse : 1 m/s (b) Plan de vol 2 ; Vitesse : 1 m/s

FIGURE 3.10 – Analyse du décalage 3D entre la trajectoire de la caméra et le plan de vol selon le processus simple ; Distance entre les waypoints : ≈ 30 m.

Le décalage maximal constaté est d'environ 45 centimètres. Il se produit environ à mi-chemin entre les waypoints espacés d'environ 30 mètres. La trajectoire du drone dérive légèrement et forme une courbe entre les waypoints. Si le drone dérive au départ de la trajectoire, les corrections appliquées sur ses déplacements ne permettent pas de le rabattre sur la trajectoire.

En intégrant le processus avancé dans le PID, les résultats suivants sont obtenus (Figure 3.11) :



(a) Plan de vol 1 ; Vitesse : 1 m/s (b) Plan de vol 2 ; Vitesse : 1 m/s

FIGURE 3.11 – Analyse du décalage 3D entre la trajectoire de la caméra et le plan de vol après l'intégration d'un point virtuel ; Distance entre les waypoints : ≈ 30 m.

Le décalage par rapport au plan de vol donné est significativement réduit et l'écart maximal est de 30 centimètres. De plus, les décalages maximaux se produisent sur les waypoints (voir 3.11a). Ce phénomène est normal étant donné que le waypoint est considéré comme atteint si le drone est à moins de 20 centimètres de celui-ci. Au départ du segment de trajectoire suivant, le drone se trouve donc déjà à 20 centimètres du plan de vol. En analysant les décalages au milieu des segments de trajectoire, les écarts constatés varient entre 5 centimètres et 15 centimètres.

L'utilisation d'un point virtuel proche du drone aide donc significativement le guidage et permet de moins dériver de la trajectoire donnée par le plan de vol.

3.2 Test sur un ouvrage réel

Les différents tests effectués jusqu'à présent se sont déroulés dans un environnement maîtrisé, ce qui a permis de valider les caractéristiques techniques du système développé. Cependant, cet environnement n'est pas représentatif d'un parement de barrage avec d'importants masques GNSS et des contraintes spécifiques.
Il est donc essentiel d'effectuer des tests sur un ouvrage réel afin de déterminer les limites du système développé.

3.2.1 Site de test choisi

Le site choisi est le barrage de Zeuzier, situé en Valais central (Figure 3.12). Ce barrage-voûte d'environ 140 m de haut est particulièrement intéressant pour tester le système développé pour les raisons suivantes :

— Barrage-voûte : La navigation est plus compliquée que sur un élément linéaire. La voûte crée également des surplombs avec des masques GNSS importants ;
— Hauteur : Avec un mur d'environ 140 m, des vols avec une hauteur importante de vol peuvent être effectués pour analyser la robustesse du système de flux optique ;
— Vallée encaissée : Le barrage se situe dans une vallée relativement encaissée. Le sommet du parement est large d'environ 200 m et se réduit progressivement pour ne faire plus que 30 m de large en fond de parement. Cette configuration crée des masques GNSS importants.

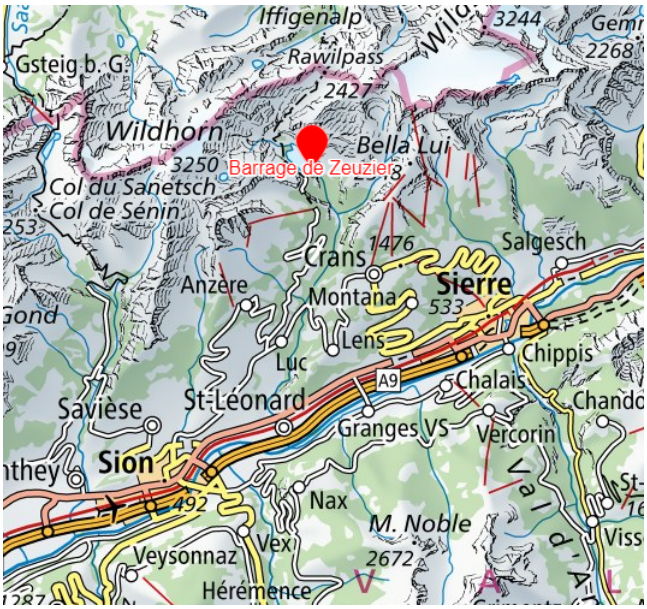


FIGURE 3.12 – Situation barrage de Zeuzier.

3.2.2 Relevé initial

Un relevé initial de l'ouvrage et de ses environs est nécessaire afin de pouvoir calculer différents plans de vol précis pour le relevé du parement aval. Pour ce faire, l'application DJI est utilisée pour effectuer un relevé automatique avec des prises de vue nadiral. Ce vol est complété par des prises de vue manuelles avec différentes orientations sur le parement et sur les environs. Environ 400 images sont capturées et une résolution moyenne de 1.7 cm/px est obtenue. Des points de calage sont mesurés au GNSS sur le sommet du mur et à la station totale sur le parement. L'erreur moyenne obtenue sur les points de calage est de 1.4 cm. Après traitement, le nuage de points du parement et des obstacles aux alentours (Figure 3.13) est obtenu.

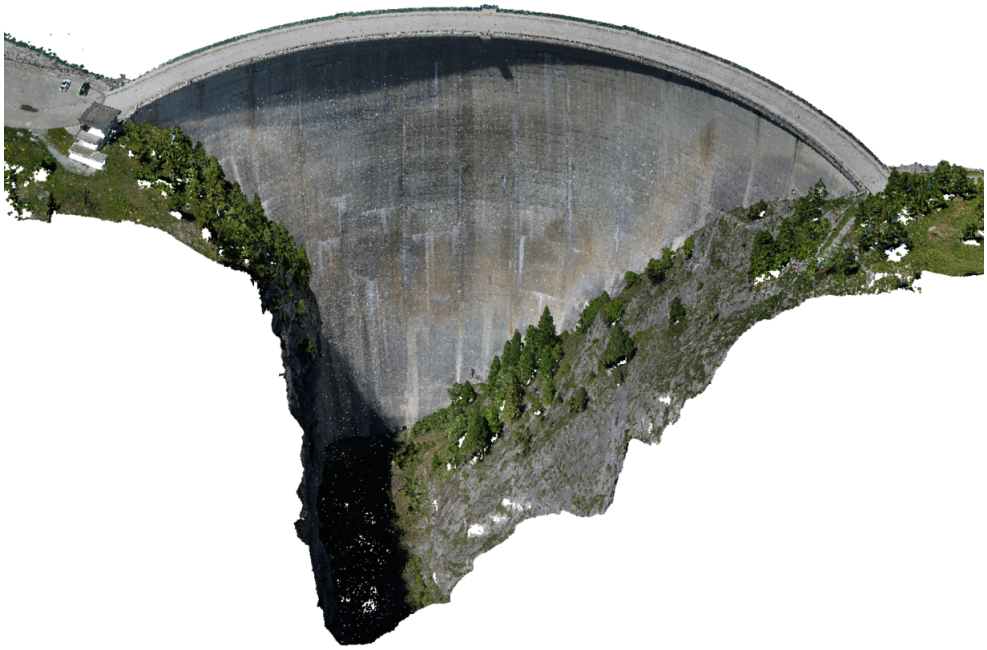


FIGURE 3.13 – Relevé initial, nuage de points.

Au final, un nuage de points et un maillage sont exportés du logiciel Agisoft Metashape.

3.2.3 Calcul d'un plan de vol

Dans un premier temps les contraintes techniques du plan de vol sont définies :

- **Caméra** : Zenmus X4S (5472x3078px avec focale de 8.8 mm) ;
- **Recouvrement** : 70% en X et en Y ;
- **GSD** : 2.5 mm ;
- **Vitesse de vol** : 1 m/s ;

Les paramètres du plan de vol peuvent ensuite être calculés :

- **Hauteur de vol (distance du mur)** : 9 m ;
- **Intervalle entre les prises de vue X** : 4.1 m ;
- **Intervalle entre les prises de vue Y** : 2.3 m ;

Le plan de vol prévoit d'effectuer des lignes de vol horizontales afin d'optimiser la consommation énergétique du drone. Les valeurs "X" citées dans les caractéristiques techniques ci-dessus correspondent donc au déplacement horizontal le long du mur. Les valeurs "Y" correspondent, elles, au déplacement vertical.

La construction des lignes de vol est effectuée avec le logiciel Cyclone 3DR de Leica. Les étapes suivantes sont nécessaires :

1. Nettoyage du maillage pour conserver uniquement le parement aval du barrage ;
2. Effectuer des coupes planaires dans le maillage chaque 2.3 mètres selon l'intervalle Y défini par le plan de vol (Figure 3.14) ;
3. Effectuer un décalage de ces coupes à 9 mètres du mur selon la hauteur de vol définie dans le plan de vol ;
4. Les lignes ainsi obtenues possèdent un nombre de sommets élevés étant donné qu'elles correspondent à l'intersection entre un plan et le maillage du parement qui est très dense. Il y a

donc un sommet par intersection avec un triangle du maillage. Ces lignes sont simplifiées en acceptant une déviation maximale de 1 mètre (Figure 3.15). Dans le cas le plus défavorable une GSD de 3 mm serait obtenu ;

5. Une dernière étape consiste à contrôler si les extrémités de ces lignes se situent à plus de 10 mètres d'un obstacle. Dans le cas contraire, elles sont raccourcies pour respecter cette distance de sécurité de 10 mètres. Cette analyse est effectuée par une inspection des lignes VS nuage de points.
6. Les sommets de lignes correspondent aux points de passage de drone et peuvent être exportés dans un fichier texte.

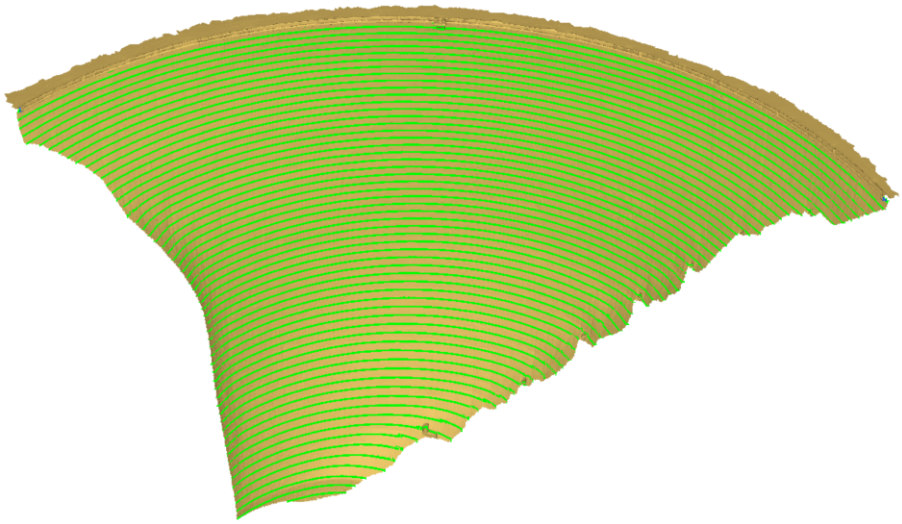


FIGURE 3.14 – Construction du plan de vol, section planaire sur le maillage avec un intervalle de 2.3 mètres.

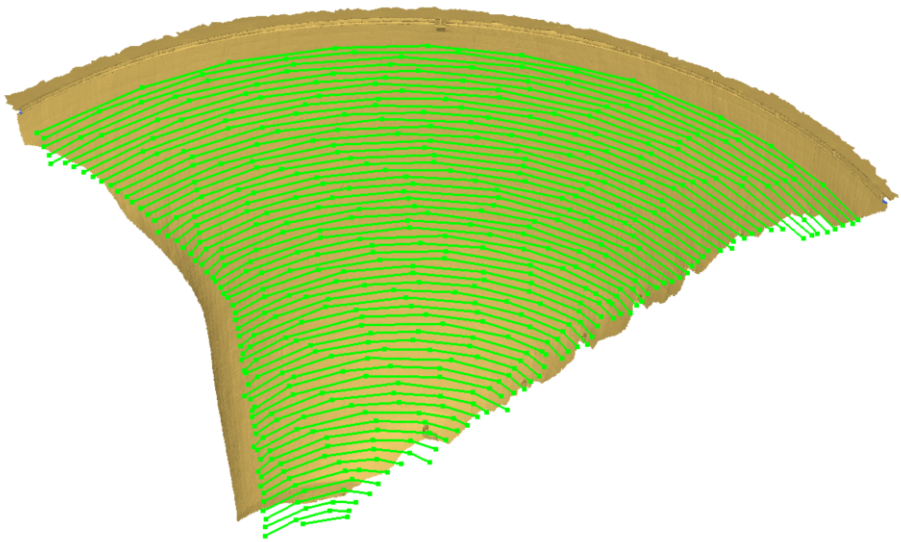


FIGURE 3.15 – Construction du plan de vol, section planaire décalée à 9 mètres du parement et simplifiée ; Représentation des sommets de ligne par les points.

Les points de passage du plan de vol sont ainsi obtenus, il reste à déterminer l'orientation du drone et du gimbal pour chacun de ces points conformément au format défini au Chapitre 2.5.8.

L'orientation du drone est définie par le gisement du segment reliant deux points ± 90° afin que le drone soit orienté face au parement.

L'orientation du gimbal est elle fixée à 0.0° afin que la caméra vise à l'horizontale.

Finalement plusieurs lignes de vol horizontales peuvent être assemblées dans un même fichier afin que le drone effectue des allers-retours le long du parement. Pour ces tests, les lignes sont assemblées quatre par quatre. En pratique le drone devra donc effectuer deux allers-retours le long du mur.

3.2.4 1ère série de tests à 9 m du parement

Maintenant que différents plans de vol sont calculés, ceux-ci peuvent être testés en live sur l'ouvrage réel. Pour ce faire une station totale Leica TS60 est installée sur un pilier du réseau géodésique du barrage. Le pilier choisi permet d'avoir une vue relativement dégagée sur l'ensemble du parement (Figure 3.16). Cette emplacement permet d'effectuer le guidage du drone avec des visées jusqu'à 200 mètres.

FIGURE 3.16 – Installation de la station totale sur un pilier géodésique

Les analyses qui suivent sont faites sur la base des deux premiers plans de vol testés. Cela correspond à huit lignes de vol horizontales sur 18 mètres de hauteur en partant du sommet du barrage. L'ensemble des traitements sont effectués avec le logiciel Agisoft Metashape.

Analyse du respect du plan de vol

Dans un premier temps une analyse peut être effectuée pour valider que les différentes caractéristiques définies par le plan de vol soient bien respectées. Pour ce faire, 380 images sont calées avec 11 points d'appuis répartis sur le parement. 4 points de contrôle sont également à disposition. L'erreur moyenne 3D sur l'ensemble des points d'appuis est de 1.1 cm et de 1.4 cm sur les points de contrôle (Le résultat détaillé est disponible en annexe A.2.1). Cette erreur est satisfaisante puisque la plupart des points de calage sont des points caractéristiques du parement levés au laser. Une calibration de caméra précédemment calculée est également appliquée (Annexe A.1).

Le logiciel Agisoft Metashape permet d'analyser le recouvrement entre l'ensemble des images. Le recouvrement important défini de 70% est respecté car les zones de recouvrement minimales sont couvertes par huit images (Figure 3.17). Ce recouvrement important est nécessaire pour obtenir un bloc d'images solides.

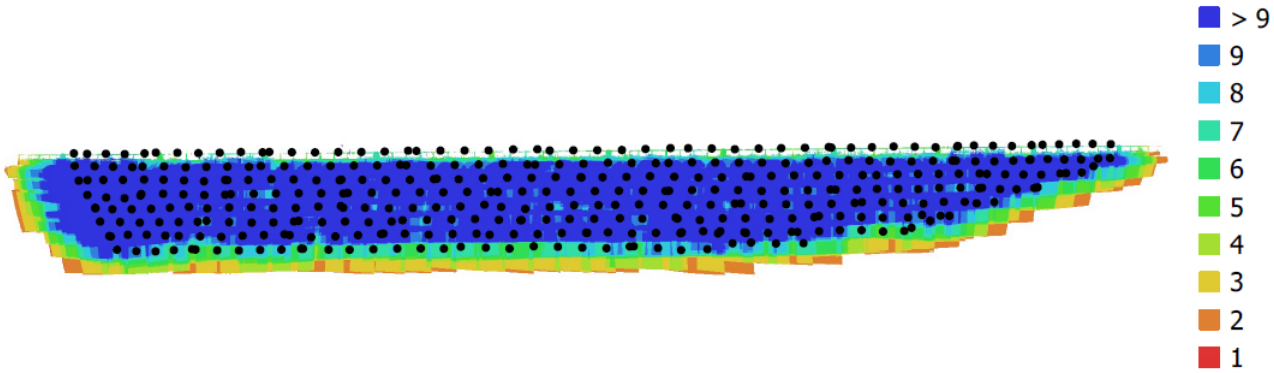


FIGURE 3.17 – Analyse du recouvrement depuis le logiciel Agisoft Metashape.

L'intervalle de distance entre les images est défini à 4.1 m dans le plan de vol (Chapitre 3.2.3). En analysant cet l'intervalle sur la position des images calculée par aérotriangulation, l'histogramme suivant est obtenu (Figure 3.18). 70% des images respectent l'intervalle de 4.1 m ±10 cm. Il y a moins de 10 images avec un intervalle de plus de 4.2 m. Concernant les images à moins de 4.1 m d'intervalle, il est normale qu'il y en a un nombre plus important. En effet, la distance entre les images de fin et début de ligne est défini à 2.3 m par le plan de vol. De plus, avec le mode de capture selon un intervalle, la dernière image capturée d'un segment se trouve à moins 4.1 m du waypoint suivant.

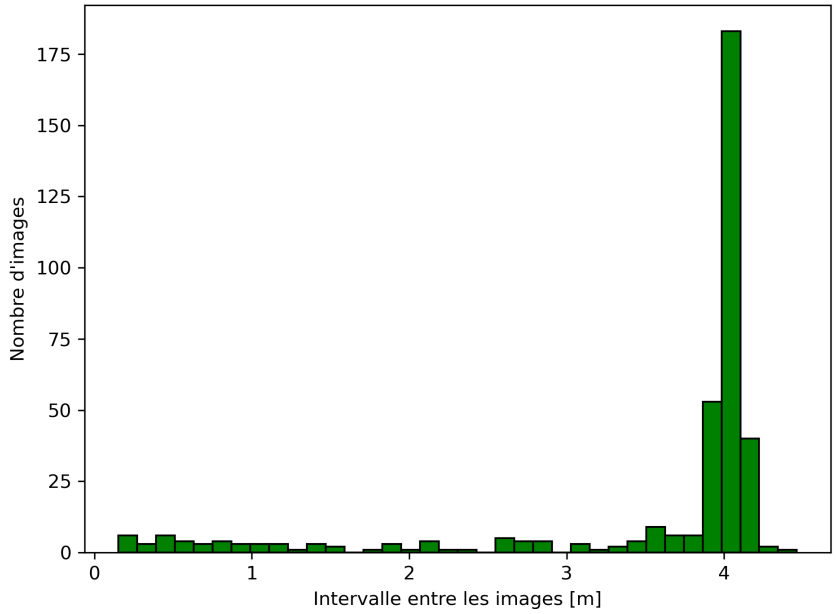


FIGURE 3.18 – Analyse des intervalles entre les images.

La résolution définie par le plan de vol était de 2.5 mm. Celle obtenue réellement est de 2.47 mm (résolution moyenne donnée par le logiciel Metashape).

En analysant le décalage 3D entre le plan de vol et la trajectoire réellement effectuée par le drone, le résultat suivant est obtenu (Figure 3.19). Cette analyse est effectuée sur le premier plan de vol (quatre premières lignes de vol), mais le résultat est similaire pour le deuxième plan de vol. Le décalage moyen est de 11 centimètres. Un décalage supérieur de maximum 50 centimètres est constaté sur un seul segment (points rouges en bas à gauche de la Figure 3.19).

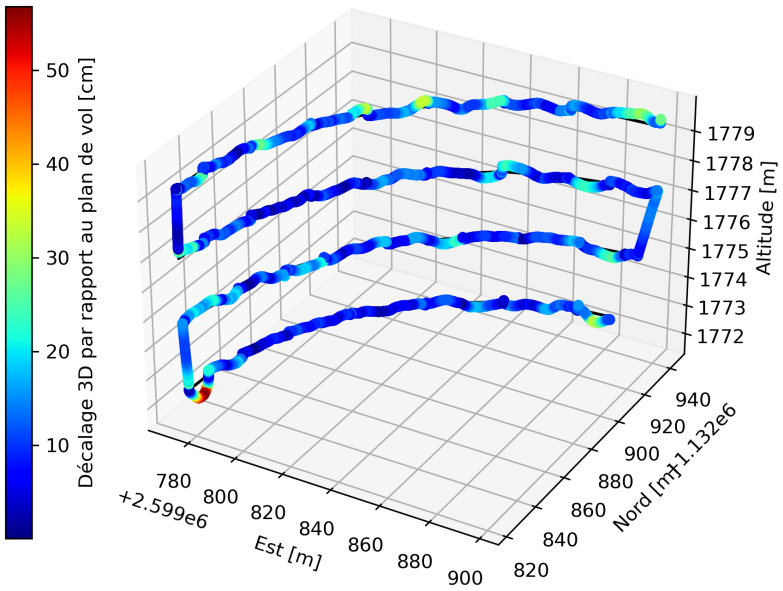


FIGURE 3.19 – Analyse du décalage 3D entre le plan de vol et la trajectoire réellement effectuée.

Suite à ces analyses, il peut être conclu que le plan de vol défini est bien respecté.

Précision du positionnement des images en post-traitement

Comme exposé au Chapitre 3.1.3, il est possible d'effectuer un post-traitement afin de déterminer des positions plus précises pour chaque images capturées. Un fichier contenant les coordonnées 3D de chaque cliché est créé. Ce fichier peut ensuite être importé dans Agisoft pour mettre à jour la position des images.

Le calcul peut alors être exécuté en utilisant uniquement les positions des images pour le calage. La précision sur les images est laissée à 10 mètres sur l'ensemble des clichés afin de ne pas contraindre le système s'il devait y avoir des images avec une position renseignée de mauvaise qualité. Suite à la phase d'alignement des images, des erreurs sont calculées entre la positions de base et la position estimée pour chaque image. Ces erreurs peuvent être analysées pour détecter si certaines images présentent des écarts plus importants. Les images avec une erreur de plus 20 cm ne sont pas utilisées pour effectuer le calage du bloc (Il avait été démontré au chapitre 3.1.3 que la précision atteignable en post-traitement était de l'ordre de 10 cm avec quelques points pouvant atteindre 20 cm), cela concerne 15 images sur les 391 au total. Ces images sont simplement désactivées dans Agisoft Metashape. L'erreur moyenne finale 3D sur les 376 autres images et de 5.1 cm. La précision qui est atteinte avec le système de positionnement par station totale est donc comparable à un système RTK disponible sur certains drones.

L'ensemble des points de calages sont mis en points de contrôle, cela veut dire qu'il ne sont pas pris en compte pour le calage du bloc d'images mais qu'un écart est calculé pour chacun de ces points. Sur les 15 points de contrôle, l'erreur moyenne 3D est de 2.1 cm. l'erreur maximale constatée est de 3.9 cm (Le résultat détaillé est disponible en annexe A.2.2). La méthode de géoréférencement sans point d'appui avec un post-traitement des images permet donc d'obtenir un calage à ± 2 cm.

Il est également intéressant d'analyser combien de points de calage seraient nécessaires pour atteindre la précision de 1.4 cm sur les points de contrôle (précision atteinte dans le chapitre précédent lorsque des points d'appui sont utilisés pour effectuer le calage du bloc d'images). En effectuant un calcul avec 3 points d'appui bien répartis sur le parement, tout en gardant les positions des images actives, l'erreur moyenne obtenue est de 8 mm sur les points d'appui et 2.2 cm sur les points de contrôle. D'autres tests sont effectués avec 1 seul de point d'appui et 5 points d'appui, les précisions atteintes restent les mêmes. Pour atteindre la précision de 1.4 cm, il faudrait utiliser un nombre important de points d'appui. L'avantage du géoréférencement sans points d'appui serait donc perdu.

L'ajout de quelques points d'appui avec la méthode de géoréfrencement sans points d'appui ne permet donc pas d'améliorer significativement le résultat final. Cependant il est toujours obligatoire d'avoir des points de contrôle bien répartis pour confirmer le géoréférencement correct du bloc d'images.

Analyse des précisions calculées

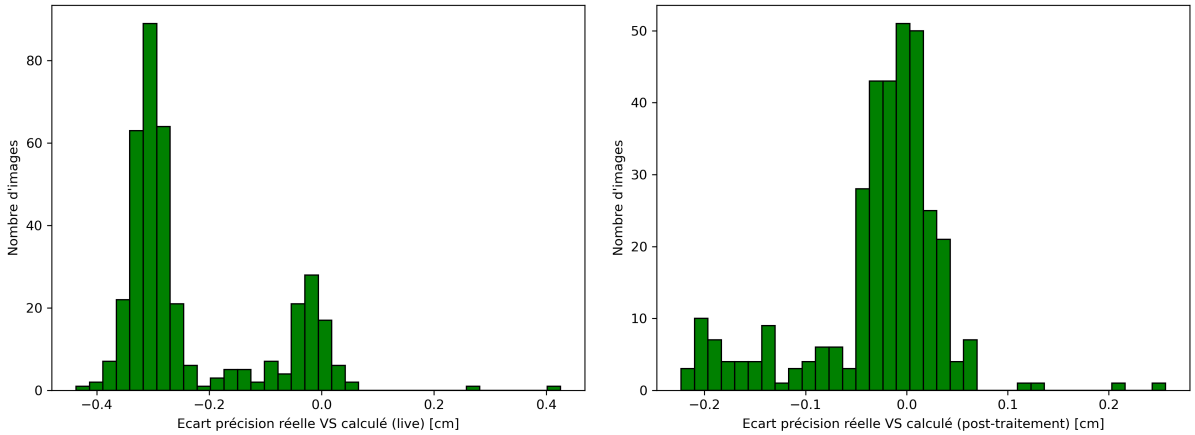
Comme expliqué au Chapitre 2.6, une précision est associée à chaque image lors de sa capture. Lors du calcul d'aérotriangulation, une erreur est obtenue sur chaque image. Celle-ci correspond à la différence entre la position donnée et la position ajustée. La précision calculée en live devrait correspondre à cette erreur (Pour cette analyse, l'erreur calculée par Agisoft est considérée comme la valeur de précision "vraie"). Une analyse est donc effectuée afin de déterminer si la précision associée à chaque image lors du vol est représentative ou non. Cette analyse est simplement effectuée en faisant la différence entre l'erreur calculée par Agisoft et la précision calculée lors du vol.

Une première analyse est faite à partir des données brutes de précision (Figure 3.20a). La majorité des images présentent une erreur dans la détermination de la précision d'environ -30 cm. Cette erreur est due au fait que la position des images à subi un post-traitement, mais que la précision est restée

elle, inchangée. Dans ce cas, la précision enregistrée en live est donc majoritairement trop pessimiste.

Comme mentionné au Chapitre 3.1.2, les images sont recalculées 0.3 seconde dans le "futur", ce qui permet de corriger l'erreur liée à un décalage de temps. La nouvelle précision peut donc être déterminée en ne tenant plus compte du facteur lié à la vitesse de déplacement ainsi qu'au temps. Il s'agit du deuxième facteur de correction expliqué au Chapitre 2.6.1. Il faut donc réduire la précision de $vitesse_{drone} * 0.3$. Ainsi la précision obtenue est uniquement liée à la précision renvoyée par le filtre de Kalman. Le résultat obtenu après cette correction est visible sur la figure 3.20b. Suite à cette correction, 85% des images présentent un écart de calcul de la précision de moins de 5 cm.

Il reste des images avec des écarts de précision allant jusqu'à 20 cm. Cependant la majorité des différences d'erreur sont négatives (Figure 3.20b) ce qui veut dire que les précisions calculées sont trop pessimistes. Il est donc difficile de déterminer finement une précision lors du vol. Une simple perte de mesures terrestres peut provoquer une dérive de la trajectoire et ainsi impacter la précision déterminée. Au final, il est plus intéressant d'effectuer le calcul de l'aérotriangulation en laissant des précisions relativement élevées sur les images (10 m dans les tests des chapitres précédents).



(a) Précision calculée en live

(b) Précision redéterminée suite au post-traitement

FIGURE 3.20 – Analyse des précisions calculées et de la précision réelle donnée par Agisoft.

Limitation du guidage

Un problème est survenu dans le guidage du drone lors d'un test avec un troisième plan de vol. Le début de ce plan de vol se situe à environ 20 mètres du sommet du parement. Le drone DJI passe en mode "ATTI", cela veut dire qu'il n'y a plus de GNSS pour positionner le drone. Lorsque la hauteur de vol est faible (moins de 10 m), le drone utilise ses caméras nadiral pour se stabiliser. Mais dans le cas du barrage, la hauteur de vol est d'environ 100 m. Lorsqu'il y a une perte totale des GNSS, le drone n'a donc aucun autre moyen de se positionner par rapport à son environnent. Celui-ci commence alors à dériver à cause du vent qu'il n'arrive plus à compenser. De plus, le guidage via le SDK n'est pas possible lorsque le drone passe en mode "ATTI". Cette sécurité est mise en place par DJI et il est impossible de la contourner. Lors du passage en mode "ATTI", il faut repasser en guidage manuel.

Lorsque le drone se situe à la limite entre le passage en mode "ATTI" et le mode "GPS", des pertes momentanées de guidage se produisent. Le drone effectue alors des "boucles" et ne suit pas la trajectoire donnée par le plan de vol (Figure 3.21).

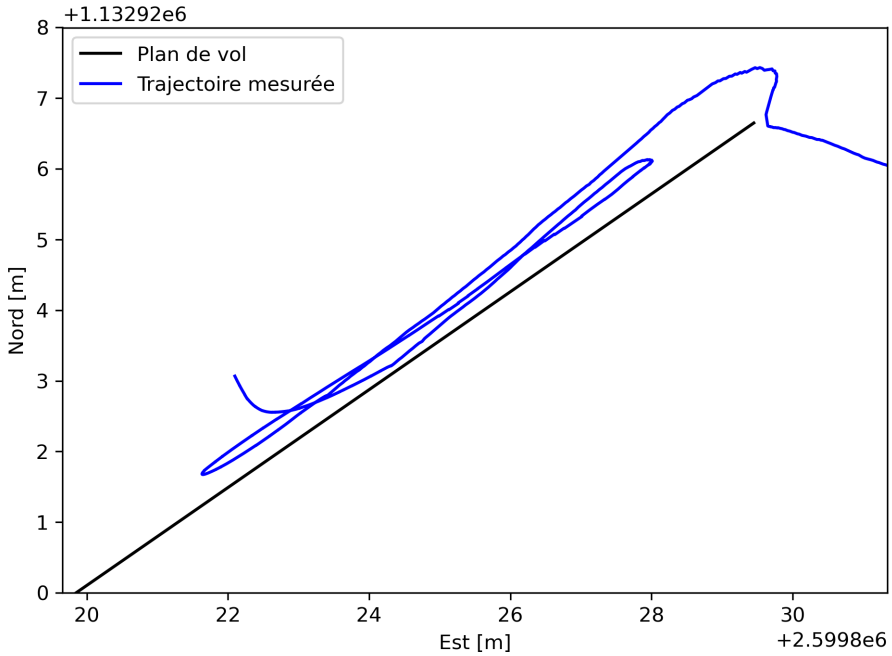


FIGURE 3.21 – Problème de guidage du drone dans la zone limite du passage en mode "ATTI".

En analysant de plus près les différentes données enregistrées lors de ce problème de guidage, il est constaté que les vitesses du drone ne sont pas calculées correctement. En effet, en observant la Figure 3.22, sur certaines portions de trajectoire mesurée, le drone ne renvoie aucune vitesse. Dans le cas d'un fonctionnement normal, il devrait y avoir des vitesses renvoyées (vecteur vert) tout le long de la trajectoire (ligne bleu).

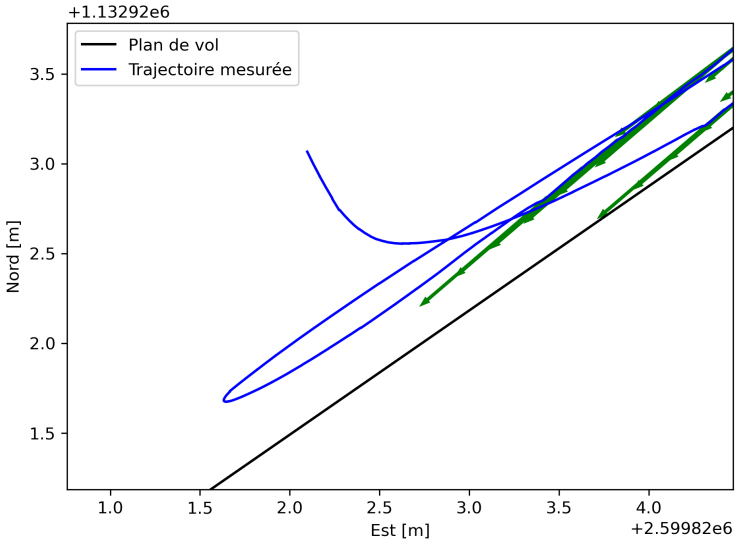


FIGURE 3.22 – Problème de calcul des vitesses du drone dans la zone limite du passage en mode "ATTI" ; Vecteurs verts : vecteurs de vitesse renvoyés par le drone.

Bien que le drone ne renvoie pas de vitesses sur certaines portions de trajectoire, le filtre de Kalman continue de calculer correctement la position du drone avec les mesures terrestres uniquement. Le drone devrait donc effectuer son guidage correctement avec les corrections renvoyées par le PID. Une dernière analyse est effectuée pour s'assurer que les corrections calculées par le PID soient correctes et donc pas à l'origine de la trajectoire en forme de "boucle". En représentant les vecteurs de corrections résultant du PID, il est constaté que ceux-ci sont calculés correctement et sur l'ensemble de la trajectoire (vecteurs rouges sur la Figure 3.23). Le problème n'est donc pas lié à une correction fausse.

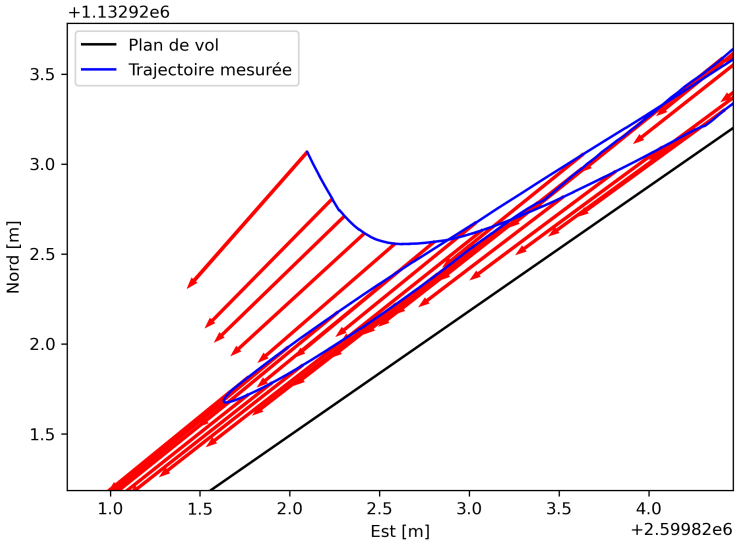


FIGURE 3.23 – Corrections calculées par le PID dans la zone limite du passage en mode "ATTI" ; Vecteur rouge : vecteur de correction calculé par le PID.

Finalement l'origine de ce problème n'est pas clairement déterminé. Une supposition peut être faite, étant donné que le drone n'arrive plus à calculer sa vitesse, il n'arrive pas non plus à appliquer une correction, elle aussi sous forme de vitesse.

3.2.5 2ème série de tests à 16 m du parement

Suite aux tests effectués dans le Chapitre précédent 3.2.4, un deuxième plan de vol est élaboré. Ce plan de vol est défini de telle manière à obtenir une résolution de 4.5 mm sur le parement. Il est alors possible de voler à 16 m du mur pour atteindre cette résolution. Le but de ce test est de savoir si, en étant plus éloigné du mur, les problèmes liés au passage en mode "ATTI" du drone étaient éliminés. Effectivement, en étant plus éloigné du mur, le masque GNSS est moins important.

Contrairement aux tests précédents, il a été possible d'effectuer des lignes de vol sur les 35 premiers mètres du parement (seulement 20 m avec le vol à 9 m du mur). Ensuite les mêmes problèmes liés au mode "ATTI" sont apparus. Ce phénomène peut être expliqué par la vallée qui devient de plus en plus étroite et donc le masque GNSS qui devient de plus en plus important.

Concernant le géoréférencement sans points d'appui, l'erreur moyenne atteinte sur les images est de 6.8 cm et de 5.7 cm sur les points de contrôle (Le résultat détaillé est disponible en Annexe A.3.2). Ces erreurs sont plus importantes que lors du premier vol. Cela s'explique d'une part par la résolution qui est plus faible, mais également par le nombre moins important d'images (190 images contre 390 dans le test précédent).

En utilisant des points d'appui, la précision atteinte est similaire à celle constatée lors des premiers tests, à savoir 1 cm sur les points d'appui et 1.2 cm sur les points de contrôle (Le résultat détaillé est disponible en Annexe A.3.1).

Voler à une distance plus éloignée du mur permet de ralentir l'impact du passage en mode "ATTI", mais la précision et la résolution finale en sont péjorée.

3.2.6 Analyse des produits possibles

Finalement, il est possible d'exporter les éléments suivants :

- **Nuage de points** ;
- **Maillage** ;
- **Orthophoto** : C'est le produit le plus intéressant pour pourvoir ensuite effectuer les analyses du parement sur un SIG, par exemple (Figure 3.24). L'orthphoto est obtenue en effectuant une projection cylindrique. L'axe du cyclindre est défini dans le logiciel 3DR avec une fonction qui permet de déterminer le meilleur cylindre à partir d'un nuage de points. Une contrainte est ajoutée afin que l'axe du cylindre soit confondu avec l'axe Z ;

FIGURE 3.24 – Portion d'orthophoto avec une résolution de 2.5 mm/px.

Conclusion

Le but de ce travail était d'étudier la possibilité d'effectuer des relevés automatisés au drone selon un plan de vol donné dans un environnement dépourvu de signal GNSS.

Pour atteindre ce but, différentes analyses ont été effectuées afin de :

1. Déterminer la plateforme la plus pertinente pour l'acquisition des images ;
2. Déterminer le meilleur moyen de positionner un drone indépendamment du système GNSS ;
3. Calculer la trajectoire du drone en temps réel ;
4. Calculer les corrections à appliquer sur le drone afin de suivre un plan de vol ;
5. Gérer la prise de vue des images.

La plateforme d'acquisition retenue pour ce travail est un drone Matrice 210 V2 de chez DJI. Ce drone de gamme professionnelle permet d'avoir un grande liberté dans les charges utiles emportées. Il est ainsi possible, contrairement à des drones basiques de type Phantom ou Mavic, d'adapter les optiques selon les besoins d'un mandat.

L'utilisation d'un théodolite et d'un prisme 360° est la solution retenue pour positionner le drone indépendamment du système GNSS. Ce système est facilement mis en place et est utilisé par tous les bureaux de géomètres. Ce sont les instruments du constructeur Leica qui ont été utilisés lors de ce travail via une communication bluetooth GeoCom. Des instruments d'autres constructeurs pourraient également être utilisés en modifiant légèrement les fonctions de communication entre le smartphone et le théodolite. La seul contrainte est qu'une communication sans fil soit possible.

Les coordonnées envoyées par le théodolite sont couplées aux mesures de vitesse du drone à l'aide d'un filtre de Kalman. L'utilisation de ce filtre permet d'utiliser ces deux flux de données non synchronisés pour déterminer la position du drone en temps réel selon une fréquence définie.

L'utilisation du contrôleur PID pour le calcul des corrections à appliquer sur drone s'est avéré suffisant et d'autres contrôleurs plus complexe comme le MPC n'ont pas eu besoin d'être employés. A partir de la trajectoire calculée par le filtre de Kalman et d'un plan de vol donné, le contrôleur PID est capable de calculer les commandes à appliquer sur le drone afin d'automatiser le vol.

L'application développée sur Android permet à un utilisateur d'automatiser le processus de capture des images selon un plan de vol défini. Le smartphone communique avec le théodolite et le drone simultanément pour effectuer ce guidage.

Les tests effectués sur le barrage de Zeuzier ont permis de tester l'ensemble du processus en condition réelle. La contrainte principale qui ressort de cette série de tests est liée au drone. En effet, le guidage de ce dernier est impossible lorsqu'aucune position GNSS ne peut être déterminée correctement. Le constructeur DJI a mis en place une sécurité qui empêche le contrôle via le MSDK lors de conditions difficiles. La limite de l'utilisation de drones commerciaux est donc atteinte dans cette situation.

La précision d'un géoréférencement sans points d'appui dans un logiciel de photogrammétrie a pu être déterminée à ± 2 cm pour un vol à 9 m du parement avec une résolution de 2.5 mm/px.

Ce travail à permis de valider le concept de guidage d'un drone à l'aide d'observations terrestres au théodolite et d'automatiser l'ensemble du processus sur une application Android.

3.3 Perspectives

Suite à la limitation principale qui est liée au drone utilisé pour ce travail, il serait intéressant d'étudier d'autre solution pour la plateforme d'acquisition. L'utilisation de drone modulaire open-source pourrait être intéressante étant donné qu'aucune limitation liée à un constructeur n'est mis en place. La prise en compte des mesures terrestres pourrait se faire de manière plus optimale en remplaçant directement les mesures GNSS à l'entrée du contrôleur de vol. Le contrôleur de vol ne verrait alors pas la différence entre des mesures qu'il reçoit du GNSS ou des mesures terrestres transmises par radio. En utilisant cette technique, le contrôleur de vol du drone et ses différentes fonctions déjà intégrées pourraient être utilisés même en l'absence de signal GNSS. Ce procédé est impossible à mettre en place sur des drones commerciaux de type DJI ou Parrot qui bloquent l'accès total à leur contrôleur de vol. L'utilisation de solutions open-source de type Ardupilot ou PX4 s'avère donc essentielle.

L'utilisation de ce type de drone pourrait même permettre d'obtenir de meilleurs résultats car il est possible d'optimiser les capteurs embarqués en fonction d'un besoin bien spécifique. Des capteurs de flux optiques visant vers l'avant du drone pourraient par exemple être ajoutés pour pouvoir se positionner par rapport au parement qui est en face du drone.

Actuellement, l'application développée peut-être utilisée pour effectuer des relevés photogrammétries plus conventionnel tout en apportant une possibilité de géoréférencement sans point d'apui pour des drones ne possédant pas de système RTK.

La poursuite du développement du processus élaboré dans ce travail passe donc pas la conception d'un drone sur-mesure permettant d'avoir une liberté totale sur les capteurs embarqués ainsi que sur la partie logiciel.

Annexe A

Annexes techniques

A.1 Calibration de la caméra Zenmus X4s

Il est essentiel d'effectuer une calibration de la caméra afin d'éviter d'avoir des erreurs de détermination de la focale lors du calcul d'aérotriangulation.

Sans l'utilisation d'une calibration, des erreurs seront surtout visibles sur la composante Z. La position des caméras ainsi que l'altimétrie du nuage de points seront impactées.

Cette erreur de détermination de la focale est surtout visible avec la configuration suivante :

— Plan de vol à une seule hauteur ;
— Point de calage réparti sur un seul plan ;

Une calibration doit donc être calculée puis utilisée pour les futurs calculs. Pour cela, une même zone est acquise avec deux hauteurs de vol bien distinctes. Des points de calage au sol et en hauteur sont également déterminés. La calibration est ensuite calculée avec le logiciel Agisoft Metashape.

Les résultats suivants sont obtenus pour les paramètres de caméra :

Paramètre de caméra X4S calibré	
F	3671.5961616322
Cx	-2.1737146692166
Cy	23.5437504724532
B1	0.710446854840043
B2	-0.0649605228490053
K1	0.0191152470331286
K2	-0.0966343129541084
K3	0.189823504644596
K4	-0.125916825729469
P1	1.93993277388553e-005
P2	0.00125992202957589
P3	-0.500039319434122
P4	0.686483425938737

TABLE A.1 – Paramètres Caméra Zenmuse X4S calibrés

A.2 Barrage de Zeuzier - 1ère série de tests à 9 m du parement

A.2.1 Calcul avec points d'appui

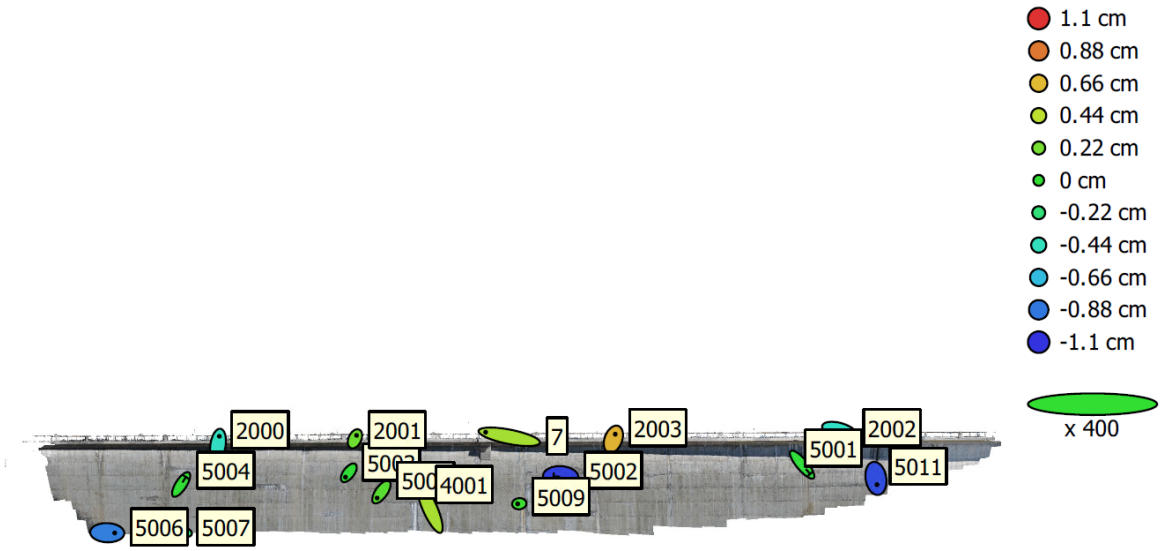


FIGURE A.1 – Emplacement des points d'appui et de contrôle ; L'erreur Z est représentée par la couleur de l'ellipse. Les erreurs X et Y sont représentées par la forme de l'ellipse.

Etiquette	Erreur X (cm)	Erreur Y (cm)	Erreur Z (cm)	Total (cm)	Image (px)
5003	-0.334227	-0.435237	-0.019298	0.549101	1.320 (10)
5006	0.754324	-0.0281214	-0.854096	1.13986	0.676 (3)
5007	0.974036	-0.209725	-0.183739	1.01316	0.511 (2)
5008	-0.432368	-0.603494	0.119396	0.751932	1.489 (11)
5009	-0.162767	-0.0159299	0.0376504	0.167823	1.493 (9)
5011	0.106962	-0.628306	-1.01334	1.1971	0.927 (3)
7	-2.3801	0.461375	0.364055	2.45158	1.694 (7)
2000	0.132789	0.884676	-0.454588	1.00346	0.939 (7)
2001	0.160251	0.342445	0.194244	0.425064	1.216 (7)
2002	0.995024	-0.279407	-0.465431	1.13348	1.046 (7)
2003	0.189559	0.505648	0.691598	0.877451	1.382 (5)
Total	**0.883614**	**0.470973**	**0.511503**	**1.12438**	**1.301**

FIGURE A.2 – Tableau des erreurs sur les points d'appui.

Etiquette	Erreur X (cm)	Erreur Y (cm)	Erreur Z (cm)	Total (cm)	Image (px)
5001	0.796685	-0.936613	-0.0294768	1.22997	1.288 (11)
5002	-0.72514	0.0433547	-1.05097	1.27759	1.261 (9)
5004	0.485218	0.743856	-0.0159245	0.888263	1.528 (11)
4001	-0.804036	1.80361	0.376729	2.01033	1.330 (10)
Total	**0.714571**	**1.0823**	**0.558476**	**1.41205**	**1.361**

FIGURE A.3 – Tableau des erreurs sur les points de contrôle.

A.2.2 Calcul géoréférencement sans points d'appui

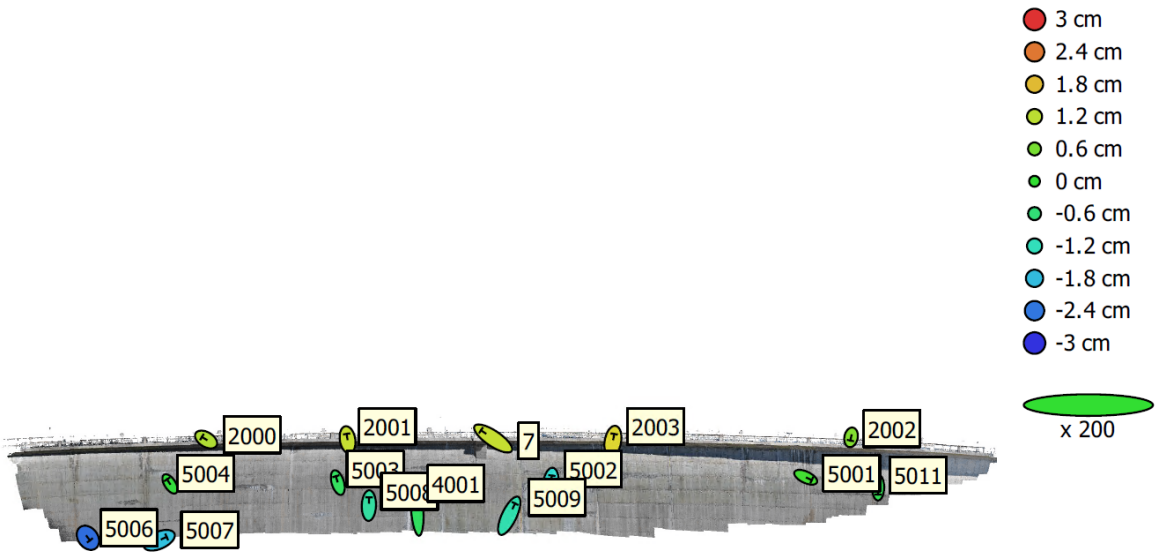


FIGURE A.4 – Emplacement des points de contrôle ; L'erreur Z est représentée par la couleur de l'ellipse. Les erreurs X et Y sont représentées par la forme de l'ellipse.

Etiquette	Erreur X (cm)	Erreur Y (cm)	Erreur Z (cm)	Total (cm)	Image (px)
5003	-0.334227	-0.435237	-0.019298	0.549101	1.320 (10)
5006	0.754324	-0.0281214	-0.854096	1.13986	0.676 (3)
5007	0.974036	-0.209725	-0.183739	1.01316	0.511 (2)
5008	-0.432368	-0.603494	0.119396	0.751932	1.489 (11)
5009	-0.162767	-0.0159299	0.0376504	0.167823	1.493 (9)
5011	0.106962	-0.628306	-1.01334	1.1971	0.927 (3)
7	-2.3801	0.461375	0.364055	2.45158	1.694 (7)
2000	0.132789	0.884676	-0.454588	1.00346	0.939 (7)
2001	0.160251	0.342445	0.194244	0.425064	1.216 (7)
2002	0.995024	-0.279407	-0.465431	1.13348	1.046 (7)
2003	0.189559	0.505648	0.691598	0.877451	1.382 (5)
Total	**0.883614**	**0.470973**	**0.511503**	**1.12438**	**1.301**

FIGURE A.5 – Tableau des erreurs sur les points de contrôle.

A.3 Barrage de Zeuzier - 2ème série de tests à 16 m du parement

A.3.1 Calcul avec points d'appui

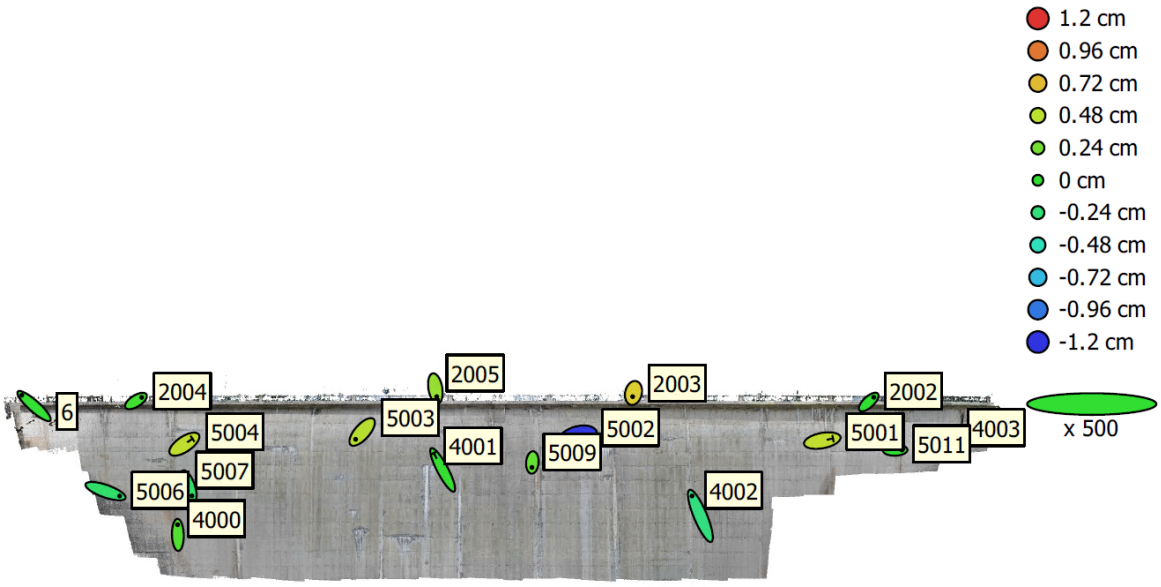


FIGURE A.6 – Emplacement des points d'appui et de contrôle ; L'erreur Z est représentée par la couleur de l'ellipse. Les erreurs X et Y sont représentées par la forme de l'ellipse.

Etiquette	Erreur X (cm)	Erreur Y (cm)	Erreur Z (cm)	Total (cm)	Image (px)
6	-0.953528	0.848858	0.0201058	1.27678	0.627 (3)
2002	0.402044	0.372579	0.00329822	0.548147	0.829 (7)
2003	-0.065391	-0.28961	0.654473	0.718669	0.817 (7)
2004	0.446932	0.281153	0.0658593	0.532102	1.023 (4)
2005	0.109503	-0.643287	0.350155	0.740552	0.858 (8)
4000	-0.0212744	0.797413	0.111019	0.805385	1.141 (10)
4002	-0.652631	1.56132	-0.26669	1.71312	1.188 (14)
4003	0.0547974	-0.644424	-0.038942	0.647921	0.351 (4)
5002	-0.686144	-0.185278	-1.1697	1.36869	1.132 (15)
5003	-0.511794	-0.576531	0.455068	0.895214	1.278 (13)
5006	1.10188	-0.382621	-0.200699	1.18356	1.027 (5)
5007	0.270558	-0.76398	-0.193688	0.833296	1.172 (14)
5009	-0.0307735	-0.390695	0.18881	0.435016	1.102 (10)
5011	0.535826	0.015106	0.0209278	0.536447	0.787 (8)
Total	**0.536148**	**0.664112**	**0.407942**	**0.946**	**1.056**

FIGURE A.7 – Tableau des erreurs sur les points d'appui.

Etiquette	Erreur X (cm)	Erreur Y (cm)	Erreur Z (cm)	Total (cm)	Image (px)
4001	-0.698985	1.31834	0.0171378	1.49228	1.070 (13)
5001	0.84011	0.159864	0.504478	0.992894	0.933 (9)
5004	0.633327	0.423188	0.494735	0.90827	1.226 (15)
Total	**0.729262**	**0.80471**	**0.408067**	**1.16013**	**1.106**

FIGURE A.8 – Tableau des erreurs sur les points de contrôle.

A.3.2 Calcul géoréférencement sans points d'appui

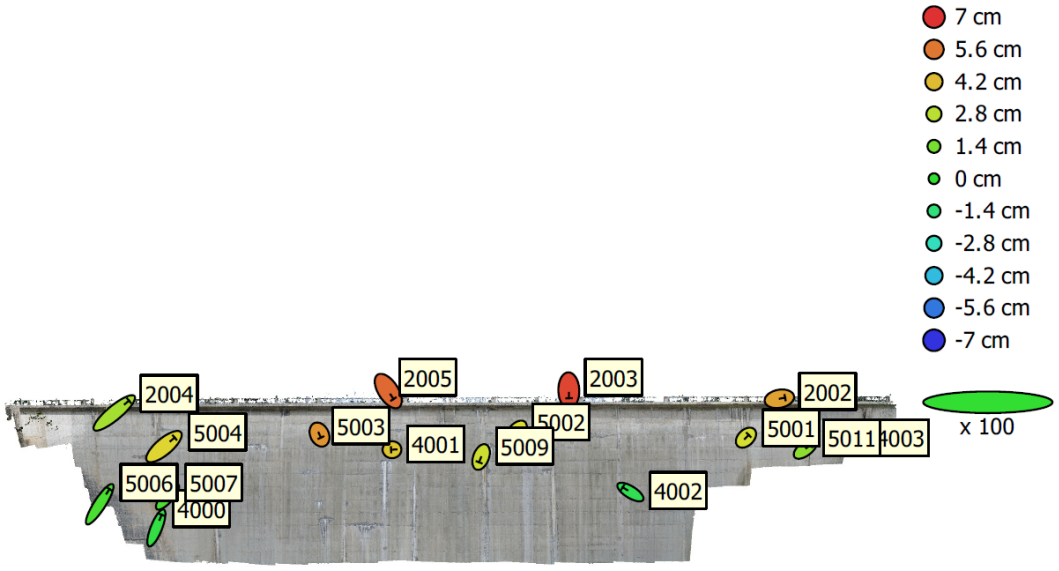


FIGURE A.9 – Emplacement des points de contrôle ; L'erreur Z est représentée par la couleur de l'ellipse. Les erreurs X et Y sont représentées par la forme de l'ellipse.

Etiquette	Erreur X (cm)	Erreur Y (cm)	Erreur Z (cm)	Total (cm)	Image (px)
2002	2.51474	0.479579	4.72884	5.37735	1.950 (7)
2003	0.0992461	-3.37713	6.60743	7.42112	2.183 (7)
2004	6.44194	5.36187	2.31805	8.69607	1.791 (4)
2005	2.28491	-3.51384	5.88625	7.22605	2.050 (8)
4000	2.30743	5.74894	-0.3362	6.20384	3.105 (10)
4001	-0.404609	-0.0145782	4.0113	4.03168	3.733 (13)
4002	-3.37186	2.07431	-0.645198	4.01104	3.724 (14)
4003	3.67849	1.06307	1.43269	4.08827	0.723 (4)
5001	1.20195	1.17221	3.32802	3.72753	2.993 (9)
5002	-1.64047	-1.9602	3.43587	4.28237	3.553 (15)
5003	0.622689	-1.32919	4.99924	5.21027	3.337 (13)
5004	4.67358	3.93052	3.66919	7.12421	3.683 (15)
5006	4.27987	6.62109	0.502159	7.89989	2.387 (5)
5007	3.01524	3.47027	0.496525	4.62396	3.678 (14)
5009	-0.872828	-2.30329	3.14556	3.99519	3.293 (10)
5011	2.32411	1.76325	1.88057	3.47089	1.736 (8)
Total	**2.99353**	**3.33488**	**3.53536**	**5.70802**	**3.151**

FIGURE A.10 – Tableau des erreurs sur les points de contrôle.

Annexe B

Mode d'emploi

B.1 Mode d'emploi utilisateur

L'application peut être installée sur un smartphone Android à partir du fichier .apk disponible sur le lien : `https://www.dropbox.com/sh/mfuv7zg57ax8kyo/AAAI19ahnfTzy0TGwihyDfmMa?dl=0`

Le développement de l'application a été effectuée sur un Samsung Galaxy Note 9 sous Android 10 avec le SDK 4.16 de DJI. La compatibilité avec des versions plus récentes d'Android n'a pas été testée. Le drone utilisé pour les différents tests était un DJI Matrice 210 V2 avec une caméra Zenmus X4s.

Il est nécessaire de créer un compte DJI [1] pour pouvoir accéder au contrôle du drone dans certaines situations. Dans les zones de restriction de vol, des limites d'hauteur de vol sont mises en place et un compte DJI est requis afin d'accepter certaines conditions et responsabilités lors du vol. Lors du premier lancement de l'application, une page popup s'ouvre afin de se connecter à un compte DJI.

B.1.1 Configuration de la station totale

La station totale utilisée doit être capable de communiquer par bluetooth et doit avoir une licence GeoCOM. Le port GeoCOM de l'instrument doit être défini sur la poignée radio (Paramètres/Connexions/Autres connexions/GeoCOM–>éditer).
Une fois ces configurations effectuées, il faut se connecter au bluetooth de l'instrument via le smartphone. En général le nom du port bluetooth de l'instrument commence par "RH".

B.1.2 Démarrage de l'application

1. Démarrer le drone et la télécommande ;

2. Brancher le smartphone à la télécommande via un câble USB. Une interface s'ouvre et permet de sélectionner une application à exécuter (Figure B.1), sélectionner "DroneControl Méric".

1. `https://account.dji.com/`

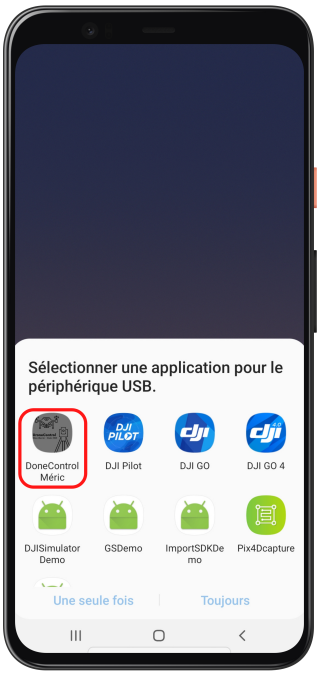
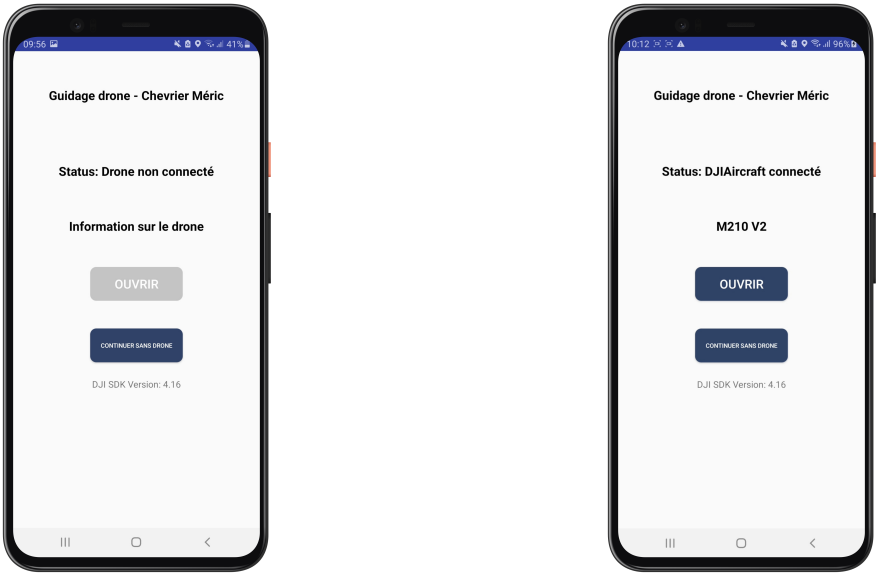


FIGURE B.1 – Démarrage de l'application.

3. Une fois la page d'accueil affichée, la connexion est établie avec le drone et la clé d'activation pour le SDK est contrôlée. Pendant cette vérification, le bouton "OUVRIR" est désactivé (Figure B.2a) ;
4. Une fois le drone détecté, le modèle du drone est affiché et le bouton "OUVRIR" permettant d'accéder à la page principale de l'application est activé (Figure B.2b) ;
5. Il est possible d'accéder à l'application sans connecter de drone en utilisant le bouton "CONTINUER SANS DRONE".



(a) Drone non-connecté (b) Drone connecté

FIGURE B.2 – Page d'accueil de l'application.

B.1.3 Gestion des paramètres du drone et de la caméra

Contrôle des paramètres du drone

Il est possible de gérer l'ensemble des paramètres du drone directement depuis l'application développée :

1. En haut à gauche de l'écran le mode de vol du drone est affiché (**N°1** Figure B.3). En cliquant sur cette partie, une page supplémentaire s'ouvre (Figure B.4). Cette page permet de voir l'état des différents composants du drone et d'effectuer des calibrations si nécessaire ;
2. Au centre, les paramètres les plus importants pour le contrôle du drone sont affichés (**N°2** Figure B.3). De gauche à droite : Mode de vol, Nombre de satellites, État de la connexion avec la télécommande, État de la connexion pour le flux vidéo, État des batteries du drone ;
3. En bas à gauche des informations par rapport à la position du drone sont visibles (**N°3** Figure B.3), De gauche à droite : Radar avec orientation du drone, Distance du drone par rapport à la télécommande, Hauteur de vol du drone par rapport au point de décollage, Vitesse horizontale, Vitesse verticale.

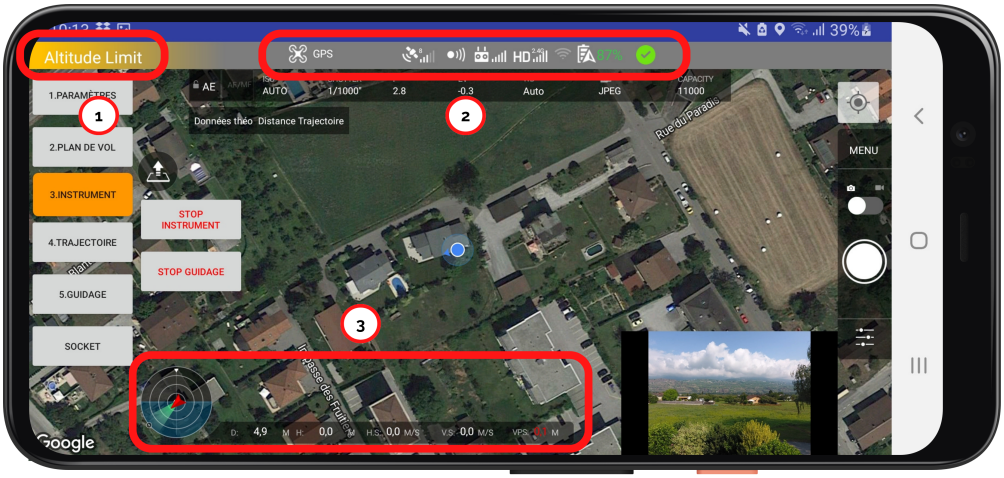


FIGURE B.3 – Gestion des paramètres liés au drone.

FIGURE B.4 – Gestion des paramètres liés au drone ; récapitulatif de l'état des différents composants.

Contrôle des paramètres de caméra

Les différents paramètres de la caméra sont également accessible depuis l'application :

1. Une barre principale au sommet de l'écran affiche les paramètres actuels de la caméra ainsi que l'espace disponible sur la carte SD (**N°1** Figure B.5) ;
2. Sur la droite de l'écran il est possible d'accéder aux paramètres avancés de la caméra en cliquant sur "MENU" (**N°2** Figure B.5) ;
3. Sur la droite de l'écran il est également possible d'accéder aux paramètres d'exposition de la caméra en cliquant sur le bouton du bas (**N°3** Figure B.6) ;

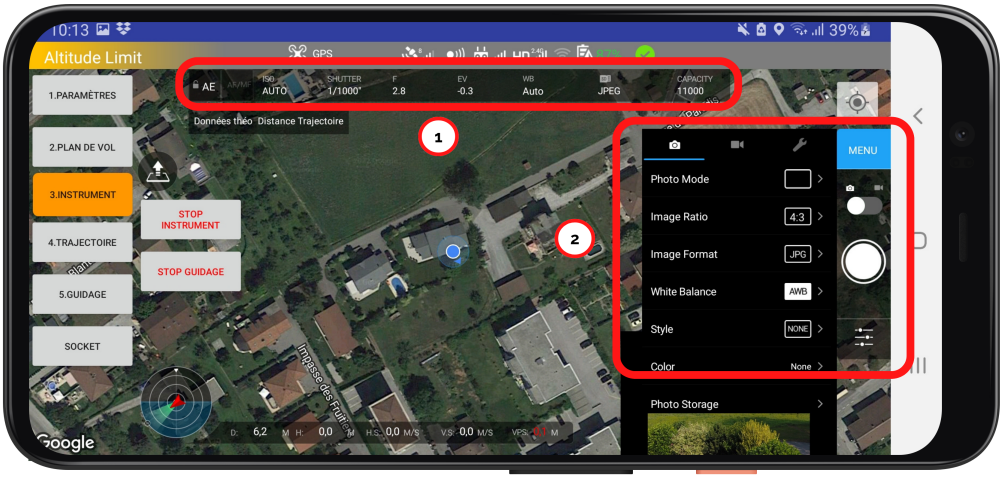


FIGURE B.5 – Gestion des paramètres liés à la caméra.

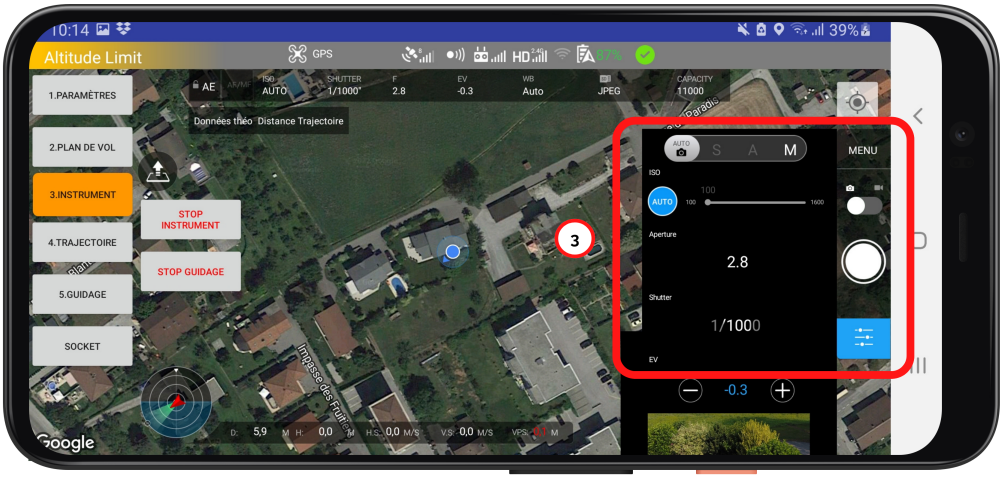


FIGURE B.6 – Gestion des paramètres liés à la caméra ; paramètres d'exposition.

B.1.4 Guidage automatique du drone

Les différentes fonctions nécessaires au guidage du drone sont regroupées sur la gauche de l'écran et se déroulent selon les étapes suivantes :

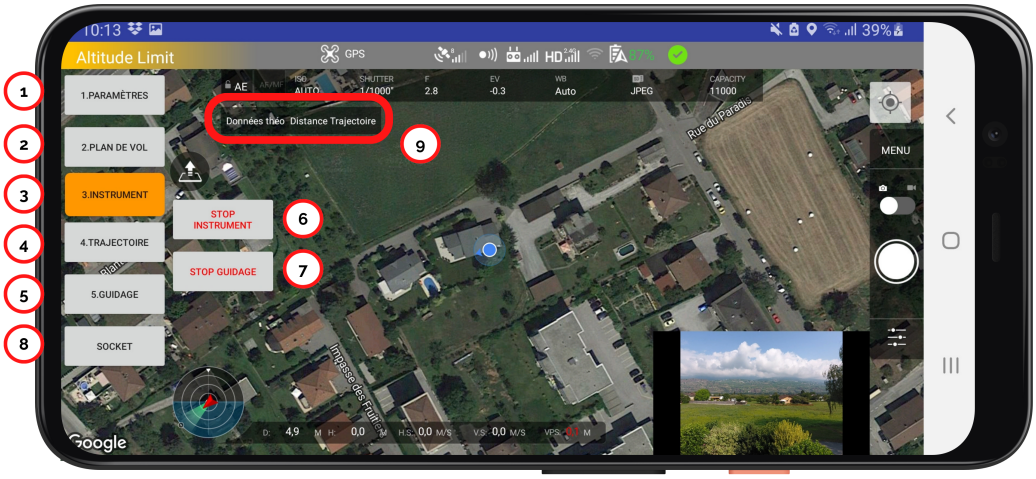


FIGURE B.7 – Gestion du guidage automatique du drone.

1. **Gestion des paramètres**. Dans un premier temps les paramètres des bases doivent être renseignés. Ces paramètres sont accessibles via une page dédiée (Figure B.8) en sélectionnant le bouton dédié sur l'application (**N°1** Figure B.7) ;

   (a) Le bras de levier entre le prisme et le centre de masse du drone en centimètres dans le système du drone. Un rappel des axes de ce système est rappelé avec un schéma sur la droite de l'écran (**N°1** Figure B.8) ;

   (b) Le bras de levier entre la caméra et le centre de masse du drone en centimètres dans le système du drone (**N°2** Figure B.8) ;

   (c) La vitesse maximale de déplacement du drone le long de la trajectoire en m/s (**N°3** Figure B.8) ;

   (d) La distance de passage pour les waypoints en mètres. C'est à dire la distance à partir de laquelle les waypoints sont considérés comme atteint (**N°4** Figure B.8) ;

   (e) Le temps d'attente aux waypoints en seconde. Temps de stationnaire effectué par le drone avant de passer au waypoint suivant (**N°5** Figure B.8) ;

   (f) Photo avec intervalle de distance. La case à cocher permet de sélectionner le mode de prise de vue. Si la case n'est pas cochée, des images sont uniquement capturées sur les waypoints. Si la case est cochée, des images sont capturées à l'intervalle de distance défini en mètres (**N°6** Figure B.8) ;

   (g) Commencer le guidage à un point donné. Si la case est cochée le guidage commence au point entré par l'utilisateur (**N°7** Figure B.8). La numérotation est liée l'ordre des waypoints dans le fichier de plan de vol. La valeur par défaut indiquée dans cette case est mise à jour automatiquement par le numéro du dernier point atteint lors du dernier vol ;

   (h) Nom des fichiers log. Cette dernière option permet d'ajouter un nom aux différents fichiers qui sont enregistrés en interne lors du vol ;

   (i) Une fois les paramètres renseignés, il reste à les valider en cliquant sur le bouton "OK" (**N°8** Figure B.8).

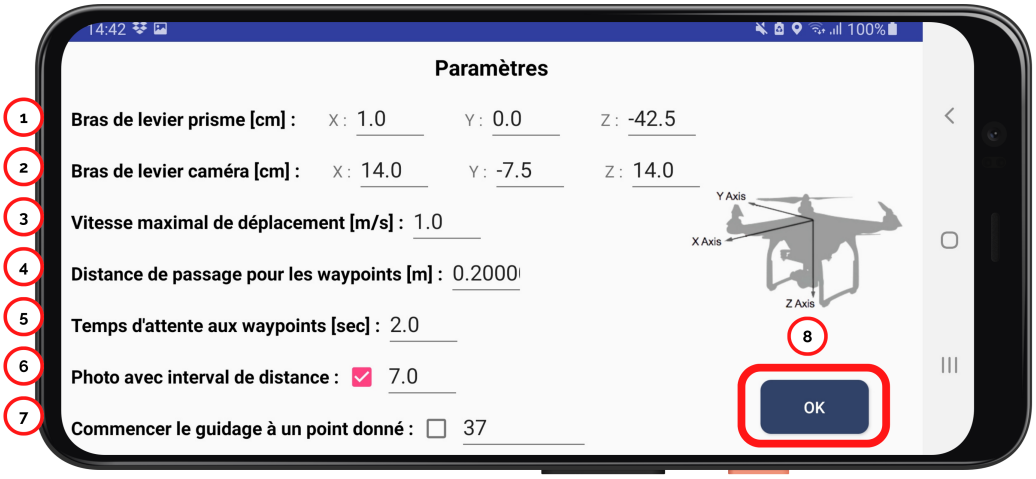


FIGURE B.8 – Gestion du guidage automatique du drone ; Paramètres.

2. **Importation du plan de vol**. En sélectionnant le bouton **N°2** sur la Figure B.7, une nouvelle page qui permet de sélectionner un fichier .txt stocké sur le smartphone s'ouvre. Si le fichier sélectionné est conforme, le plan de vol est importé et est affiché sur la carte ;

3. **Connexion à la station totale**. En sélectionnant le bouton **N°3** sur la Figure B.7, une nouvelle page qui permet de sélectionner le périphérique bluetooth correspondant à l'instrument s'ouvre. Si le périphérique sélectionné correspond bien à l'instrument et qu'il est accessible, les mesures démarrent. Lorsque les mesures démarrent, le nombre de mesures effectuées s'affichent au niveau du **N°9** sur la Figure B.7. Cette élément permet à l'utilisateur de contrôler qu'il n'y a pas de problème avec les mesures de la station totale. Si le smartphone ne reçoit plus de mesure de la part de l'instrument, le nombre de mesures effectués arrête de s'accroître ;

4. **Calcul de la trajectoire du drone**. En sélectionnant le bouton **N°4** sur la Figure B.7, le calcul de la position et de l'orientation du drone en temps réel démarre (filtre de Kalman). Un symbole apparaît sur la carte pour représenter la position et l'orientation du drone. Un historique de la trajectoire parcourue est également affiché ;

5. **Décollage**. A partir de ce moment, il faut faire décoller le drone avant d'activer la phase de guidage. Le décollage peut soit se faire manuellement soit en sélectionnant le bouton **N°1** sur la Figure B.9 ;

6. **Guidage du drone et suivi de plan de vol**. Une fois le drone en vol, en sélectionnant le bouton **N°5** sur la Figure B.7, le suivi du plan de vol donné commence. Le guidage automatique est donc actif et le drone se déplace vers le premier point du plan de vol. **A partir de cet instant les joystick de la télécommande ne permettent plus de contrôler le drone**. Lors du suivi du plan de vol, la distance orthogonale entre le drone et la trajectoire suivie est affichée au niveau de l'emplacement **N°9** sur la Figure B.7 ;

7. **Désactivation du guidage**. A tout moment le guidage automatique peut être désactivé, en sélectionnant le bouton **N°7** sur la Figure B.7. Le drone reste alors en stationnaire et il est à nouveau possible de piloter le drone avec la télécommande. Il faut également utiliser ce bouton lorsque le drone à terminé sont plan de vol ;

8. **Désactivation des mesures de la station totale**. Les mesures automatiques de l'instrument peuvent être interrompues en sélectionnant le bouton **N°6** sur la Figure B.7. La connexion bluetooth est alors fermée proprement. Cette action est a effectuer une fois le guidage automatique terminé.

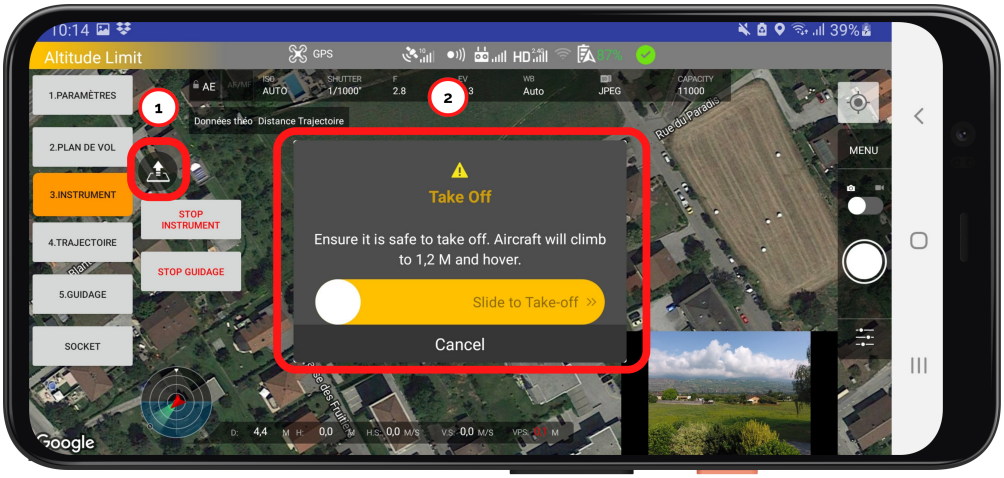


FIGURE B.9 – Gestion du guidage automatique du drone ; Décollage du drone.

B.1.5 Récupération des fichiers créés lors du vol

Lors du vol les différentes fonctions créent les fichiers .txt suivant :

- **Export_sensor_*.txt** : créé par la fonction *droneSensor*, il contient les vitesses, attitudes, positions mesurées par le drone en continu associées à un temps ;
- **Export_theo_*.txt** : créé par la fonction *theodoliteData*, il contient les coordonnées X/Y/Z mesurées par la station totale associée à un temps ;
- **kalman_*.txt** : créé par la fonction *KalmanLive*, il contient toutes les informations utilisées/-calculées par le filtre de Kalman à chaque époque :
  - Mesure théo X, Mesure théo Y, Mesure théo Z : L'observation terrestre utilisée pour calculer l'époque en question (si une nouvelle observation terrestre était disponible) ;
  - Position Kalman X, Position Kalman Y, Position Kalman Z : Les coordonnées calculées par le filtre de Kalman ;
  - Vitesse X, Vitesse Y, Vitesse Z : Les observations de vitesses renvoyées par le drone utilisées pour calculer l'époque en question (si de nouvelles observations étaient disponible) ;
  - Correction levier prisme X, Correction levier prisme Y, Correction levier prisme Z : Le bras de levier calculé pour corriger les observations terrestres et passer du prisme au centre du drone ;
  - Temps unix : Temps UNIX associé à l'époque ;
  - Epoque : Numéro d'époque du filtre de Kalman ;
  - Précision Position X, Précision Position Y, Précision Position Z : Paramètres de précision calculés par le filtre de Kalman ;
  - Précision Vitesse X, Précision Vitesse Y, Précision Vitesse Z : Paramètres de précision calculés par le filtre de Kalman.
- **Paramètres_*.txt** : Récapitulatif des paramètres entrés par l'utilisateur ;
- **Photo_*.txt** : créé par la fonction *takePhoto*, il contient les coordonnées X/Y/Z de la caméra et la précision calculée du positionnent associé à un temps pour chaque prise de vue ;
- **PID_*.txt** : créé par la fonction *Navigation* il contient toutes les informations utilisées/calculées par le contrôleur PID :
  - Position X caméra, Position Y caméra, Position Z caméra : Les coordonnées de la caméra utilisées pour calculer les corrections ;
  - Correction levier X caméra, Correction levier Y caméra, Correction levier Z caméra : Le bras de levier calculé pour obtenir les coordonnées de la caméra à partir des coordonnées renvoyées par le filtre de Kalman ;
  - Correction X PID, Correction Y PID, Correction Z PID : Les corrections calculées par le

contrôleur PID (vitesse m/s) à appliquer sur le drone afin de suivre le plan de vol ;
— Cap, Correction cap PID : Le cap actuel du drone et la correction calculée par le PID à appliquer sur le drone ;
— Temps UNIX : Temps UNIX associé à l'époque du PID.

Ces fichiers sont stockés sur le smartphone et sont accessibles via l'explorateur de fichier d'Android Studio selon la démarche suivante :

1. Brancher le smartphone par USB au PC ;
2. Ouvrir l'explorateur de fichier : Onglet View/Tool Windows/Device File Explorer ;
3. Une nouvelle fenêtre qui permet de naviguer dans les dossiers sur smartphone s'ouvre ;
4. les fichiers sont enregistrés sous : data/data/com.dji.FPVDemo/files. En effectuant un clic droit sur le fichier, il est possible de l'enregistrer dans un dossier sur le PC.

B.1.6 Post-traitement des positions des images

Le script Python *Post_traitement.py* ainsi que des fichiers d'exemples sont disponibles sur le lien suivant : `https://www.dropbox.com/sh/mfuv7zg57ax8kyo/AAAI19ahnfTzy0TGwihyDfmMa?dl=0`.

Pour recalculer la position des images 0.3 seconde (Chapitre 3.1.3) dans le "futur" via ce script, trois fichiers sont nécessaires :

— **Photo_*.txt** : créé par l'application lors du vol ;
— **PID_*.txt** : créé par l'application lors du vol ;
— **Export_Agisoft.txt** : Il s'agit d'un fichier .txt contenant le nom des caméras, qui peut être exporté depuis Agisoft. Ce fichier est uniquement nécessaire pour obtenir une liste de toutes les caméras. Il suffit d'ajouter toutes les images dans Agisoft, puis d'exporter les caméras dans un fichier .txt ;

L'ensemble du fonctionnement de ce script est directement mis en commentaire dans le code *Post_traitement.py*.

Pour l'utilisation du script, seul les noms des fichiers .txt de bases doivent être modifiés au niveau des variables *cameraAgisoft*, *cameraLive* et *pid* (ligne 97 à 99) :

```
# Import des données de base dans des dictionnaires
cameraAgisoft = importAgisoft("Export_Agisoft_16m.txt")
cameraLive = importCameraLive("Photo_16m-1.txt")
pid = importPID("PID_16m-1.txt")
```

Une fois le script exécuté, un nouveau fichier .txt nomé *Photo_new.txt* est créé. Il contient le nom et les nouvelles coordonnées X/Y/Z pour chaque image. Celui-ci peut être importé dans Agisoft pour mettre à jour la position à priori des images.

B.2 Mode d'emploi programmeur

Les caractéristiques de base du développement de cette application sont les suivantes :

— Utilisation du DJI Mobile SDK 4.16.1 pour Android disponible sur leur Github [2] ;
— Utilisation d'Android Studio Bumblebee 2021.1.1 Patch 2 [3] ;
— Test de développement sur un Samsung Galaxy Note 9 sous Android 10.

Le projet de base *Android-FPVDemo* [4] mis à disposition par DJI est utilisé comme base pour le développement du reste de l'application.

Les librairies spécifiques supplémentaires de ce projet sont :

— **com.dji :dji-sdk :4.16** : Mobile SDK de DJI pour le contrôle du drone ;
— **com.dji :dji-uxsdk :4.16** : UX SDK de DJI pour l'intégration des blocs d'interface utilisateur ;
— **com.google.android.gms :play-services-location :19.0.1** : Pour l'utilisation du fond de carte Google Map ;
— **org.ejml :ejml-all :0.41** : Pour le calcul matriciel ;

B.2.1 Environnement de développement

Le développement d'une application nécessite d'effectuer de nombreux tests et débogages. Pour ce faire, DJI fournit un simulateur de vol qui permet d'effectuer l'ensemble des tests dans un environnement virtuel. Les différentes commandes qui seront envoyées au drone à travers le SDK seront appliquées dans le simulateur et celui-ci renverra une réponse comme si le drone volait réellement.

Pour utiliser ce simulateur, le drone DJI doit être relié à un PC par un câble USB. Les commandes sont ensuite envoyées au drone depuis la télécommande comme pour un vol réel. Il est possible d'effectuer des vols manuels ou alors de relier un smartphone à la télécommande et d'envoyer des commandes avec sa propre application.

L'environnement de développement Android Studio est utilisé pour le développement de l'application. Celui-ci permet un débogage efficace de l'application directement sur un smartphone en le reliant à un PC avec un câble USB.

Le problème qui se pose maintenant est qu'il est nécessaire que le smartphone soit relié en USB à la fois à la télécommande et au PC pour pouvoir effectuer un débogage efficace, ceci pour les raisons suivantes :

— Connexion au PC : afin de déboger l'application pendant son exécution ;
— Connexion à la télécommande : afin de pouvoir communiquer avec le drone ;

Il n'est pas possible pour le smartphone d'avoir deux connexions USB simultanément. DJI fourni une application de "Bridge" [5] qui permet d'utiliser deux smartphones pour chacun des ports USB. Les deux smartphones qui doivent être sur le même réseau communiquent ensuite ensemble. Cela permet au smartphone 1 de communiquer par Wifi avec le drone à travers le smartphone 2 (Figure B.10).

2. https://github.com/dji-sdk/Mobile-SDK-Android
3. https://developer.android.com/studio
4. https://github.com/DJI-Mobile-SDK-Tutorials/Android-FPVDemo
5. https://github.com/dji-sdk/Android-Bridge-App

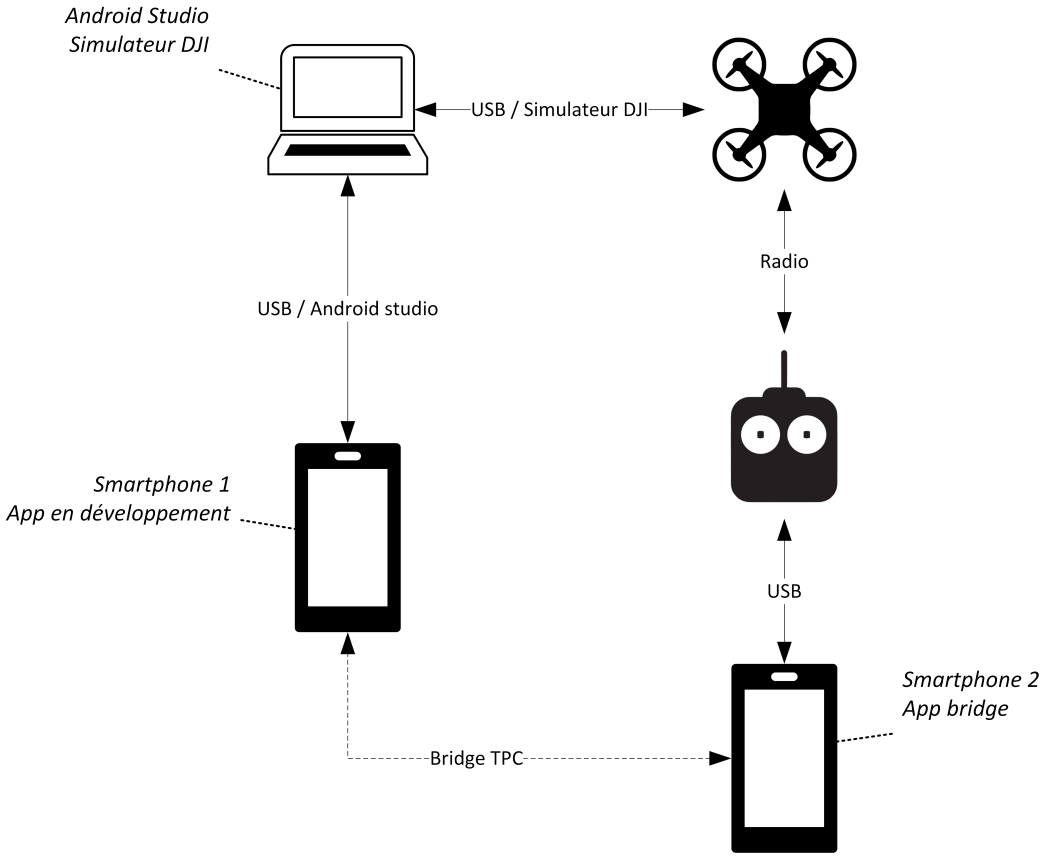


FIGURE B.10 – Environnement de développement

B.2.2 Ouverture du projet sur Android Studio

l'ensemble du projet et disponible dans le dossier "AppTotal" sur le lien suivant : `https://www.dropbox.com/sh/mfuv7zg57ax8kyo/AAAI19ahnfTzy0TGwihyDfmMa?dl=0`

La marche à suivre pour pouvoir ouvrir et compiler le projet sur Android Studio est la suivante :

1. **Modification des paramètres** : Lors de la première ouverture d'Android Studio, la fenêtre de la Figure B.11 s'ouvre. Dans l'onglet "Customize" puis "All settings" la fenêtre de la Figure B.12 s'ouvre. Il faut alors accéder à l'onglet "Android SDK" pour modifier les versions de SDK utilisées. La version Android 11.0 (R) qui correspond à l'API Level 30 doit être cochée. Ensuite appliquer ce changement et cliquer sur le bouton "OK" pour revenir sur la page de base ;

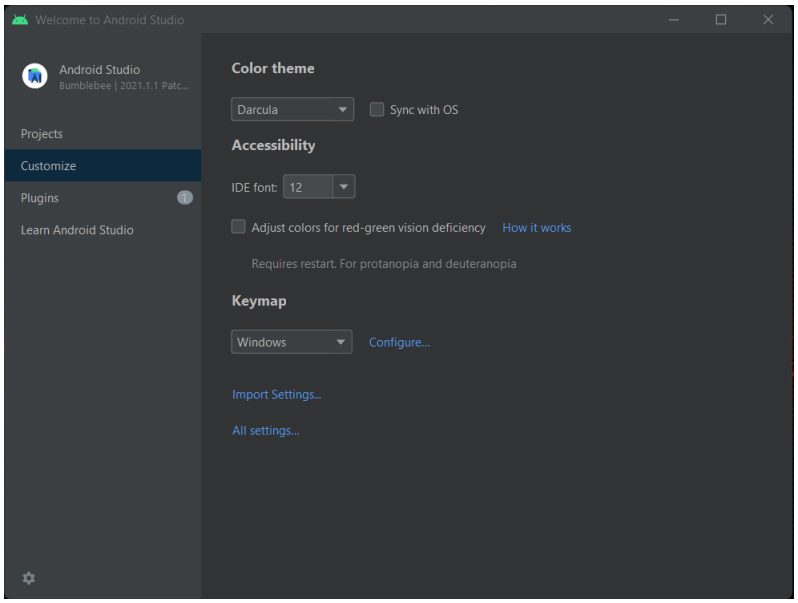


FIGURE B.11 – Android Studio, première ouverture.

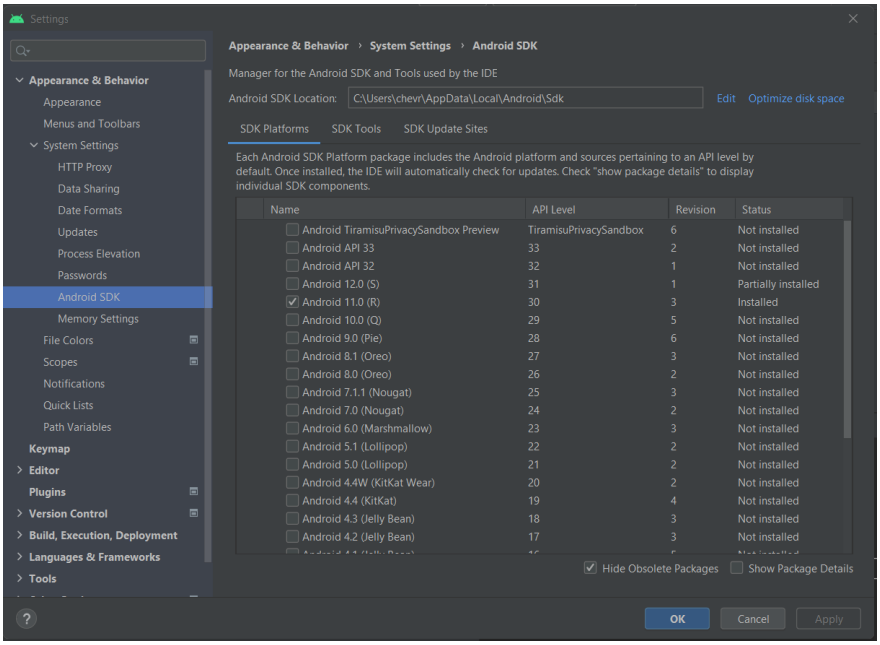


FIGURE B.12 – Android Studio, modification des versions de SDK Android.

2. **Ouverture du projet** : A partir de l'onglet "Projet" de la fenêtre de la Figure B.11, cliquer sur le bouton "Open". Dans la fenêtre qui s'ouvre, sélectionner "DroneControlMeric" en naviguant dans le dossier "AppTotal" puis valider avec le bouton "OK" (Figure B.13) ;
3. **Installation des librairies** : Les différentes librairies sont installées automatiquement lors de l'ouverture du projet grâce au fichier "Gradle" ;
4. **Navigation dans les fichiers** : Le menu de navigation de gauche permet de naviguer dans les différents fichiers .Java qui sont contenus dans le dossier "app/java/com.dji.FPVDemo".

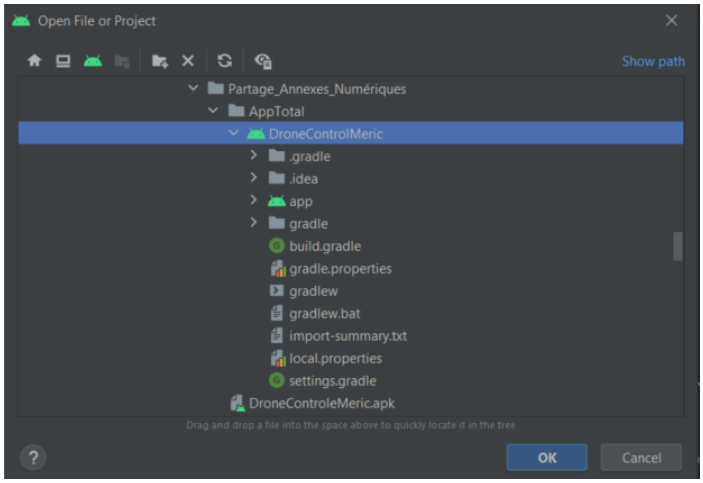


FIGURE B.13 – Android Studio, ouverture du projet.

B.2.3 Architecture générale de l'application

L'architecture générale de l'application est composée de quatre pages selon la Figure B.14. Chacune de ces pages est liée à un fichier .java contenant des fonctions et un fichier .xml contenant l'interface graphique de la page.

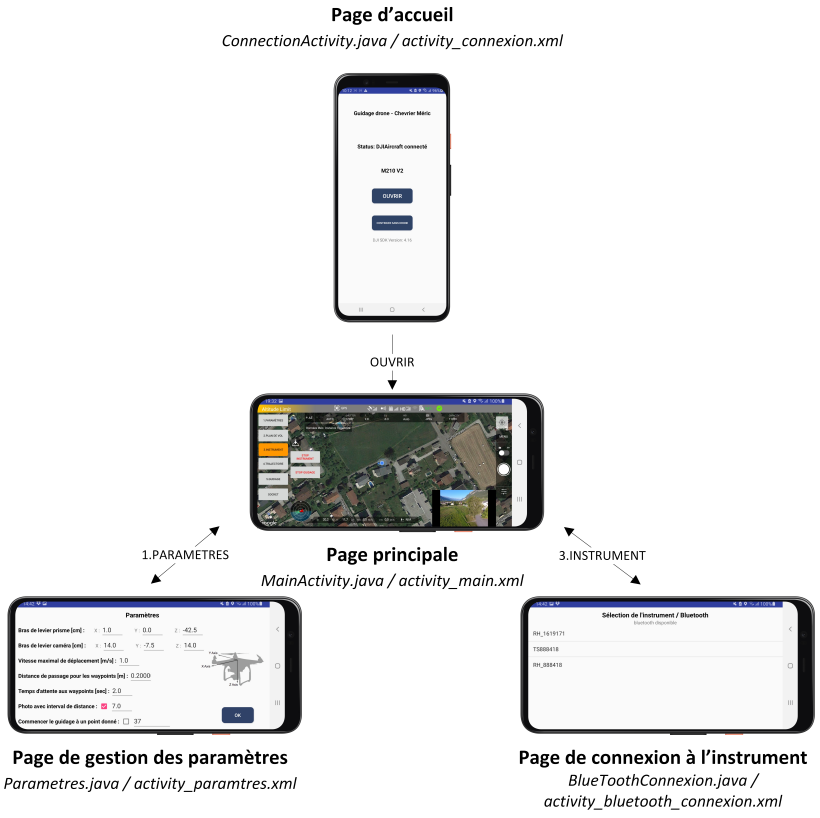


FIGURE B.14 – Architecture générale de l'application.

Fichiers

Les fichiers contenant les fonctions sont stockés dans le dossier "app/java/com.dji.FPVDemo" :

- **BlueToothConnexion.java** : Gestion des fonctions liées à la page de connexion avec l'instrument, création de la liste des périphériques bluetooth ;
- **ConnectionActivity.java** : Gestion des fonctions de la page d'accueil qui permettent de valider l'état de connexion du drone ;
- **FPVDemoApplication.java** : Gestion des fonctions liées à l'enregistrement du SDK ;
- **MainActivity.java** : Gestion des fonctions principales développées lors de ce travail (communication instrument, Calcul filtre de Kalman, Guidage) ;
- **MApplication.java** : Gestion des fonctions qui permettent de précharger des fonction du SDK (fichier de base de l'application FPVDemo de DJI) ;
- **Parametres.java** : Gestion des fonctions liées à la page de paramètres.

Les fichiers contenant les interfaces graphiques sont stockés dans le dossier "app/res/layout" :

- **activity_connexion.xml** : Interface graphique pour la page d'accueil ;
- **activity_main.xml** : Interface graphique pour la page principale ;
- **activity_parametres.xml** : Interface graphique pour la page de gestion des paramètres ;
- **activity_bluetooth_connexion.xml** : Interface graphique pour la page de connexion à l'instrument ;

B.2.4 Principe de communication avec la station totale

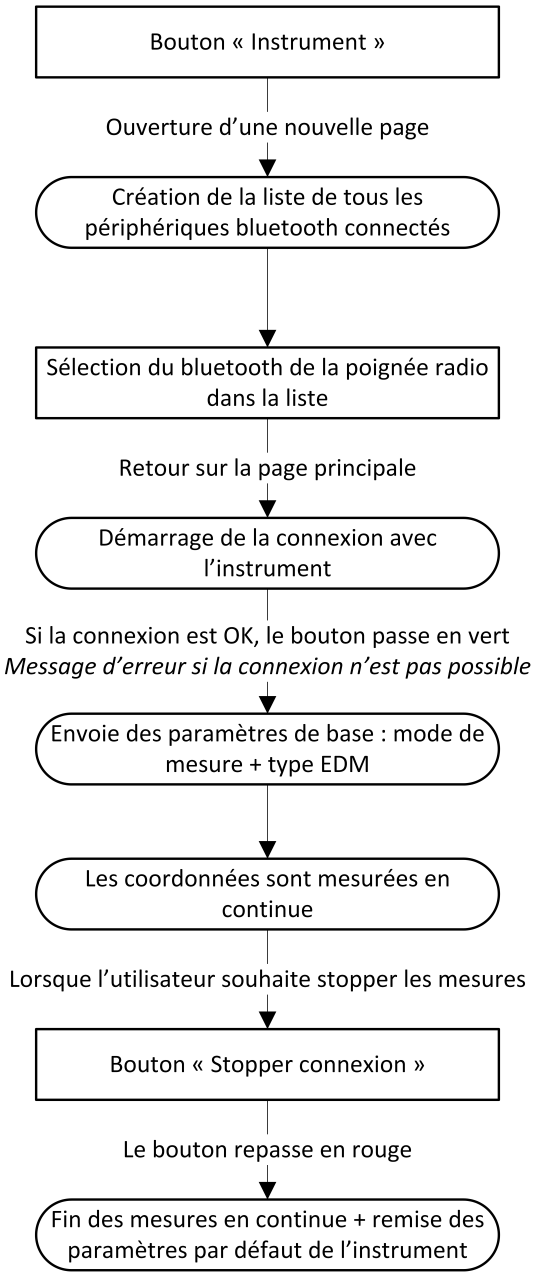


FIGURE B.15 – Schéma Connexion bluetooth.

B.2.5 Détails spécifique sur certaines fonctions

L'ensemble du code est commenté directement dans les fichiers .Java. Ce chapitre à pour but de détailler certaines fonctions spécifiques et l'interaction entre celles-ci.

Les fonctions principales de l'application sont celles visibles sur la Figure B.16. L'ensemble de ces fonctions sont contenues dans le fichier *MainActivity.java*.

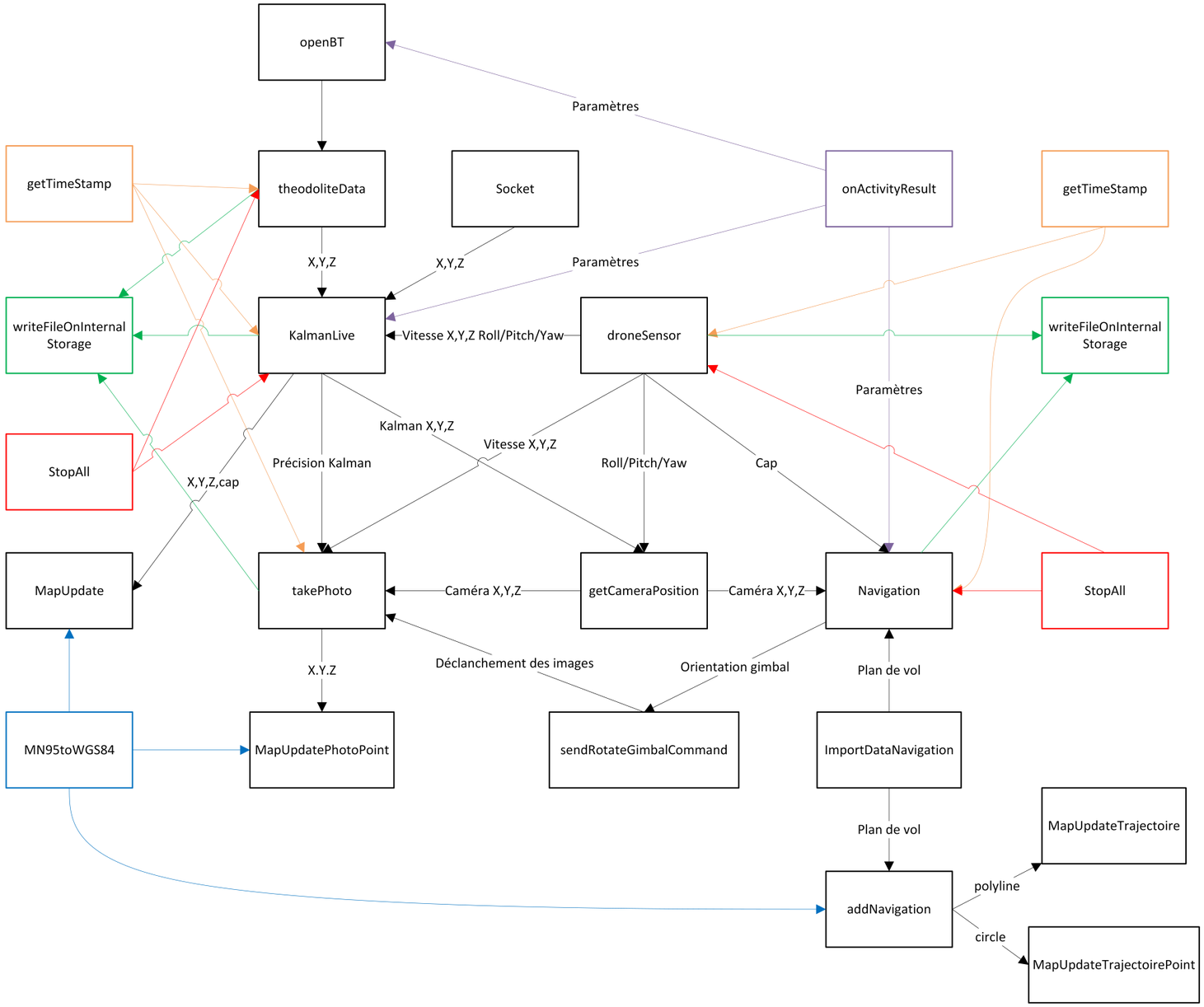


FIGURE B.16 – Interaction entre les fonctions de l'application.

Fonctions principales du guidage :

theodoliteData

- **Exécution** : Bouton "3.INSTRUMENT" sur la page principale ;
- **Entrée** : - ;
- **Sortie** : coordonnées X,Y,Z mesurées par la station totale dans les variables : *obs_x_THEO*, *obs_y_THEO*, *obs_z_THEO* ;
- **Détails** : Dans un premier temps la connexion bluetooth entre la station totale et le smartphone est créée avec la fonction *openBT*. Ensuite, la fonction *theodoliteData* va envoyer les commandes à l'instrument afin de démarrer les mesures et d'obtenir les coordonnées X,Y,Z du prisme visé. Une fois lancée la fonction tourne en boucle. Les variables en sortie sont disponibles pour l'ensemble des fonctions du fichier *MainActivity.java* et sont mises à jour à chaque fois qu'une nouvelle coordonnée est mesurée.

droneSensor

- **Exécution** : Bouton "4.TRAJECTOIRE" sur la page principale ;
- **Entrée** : - ;
- **Sortie** : vitesse X/Y/Z, attitude Roll/Pitch/Yaw, cap mesuré par le drone dans les variables : *vitesseX*, *vitesseY*, *vitesseZ*, *roll*, *pitch*, *yaw*, *cap* ;
- **Détails** : Une fois lancée, la fonction tourne en boucle et récupère les mesures des capteurs interne du drone. Les variables en sortie sont disponibles pour l'ensemble des fonctions du fichier *MainActivity.java* et sont mises à jour à chaque fois qu'une nouvelle valeur est mesurée.

KalmanLive

- **Exécution** : Bouton "4.TRAJECTOIRE" sur la page principale ;
- **Entrée** : *obs_x_THEO*, *obs_y_THEO*, *obs_z_THEO*, *vitesseX*, *vitesseY*, *vitesseZ*, *roll*, *pitch*, *yaw* ;
- **Sortie** : coordonnées X/Y/Z calculées par le filtre de Kalman ainsi que leurs précisions dans les variables : *X_kalman*, *Y_kalman*, *Z_kalman*, *X_kalman_precision*, *Y_kalman_precision*, *Z_kalman_precision* ;
- **Détails** : Une fois lancée, la fonction tourne en boucle et calcule la trajectoire du drone à l'aide d'un filtre de Kalman à partir des observations terrestres et des vitesses mesurées par le drone. Les variables en sortie sont disponibles pour l'ensemble des fonctions du fichier *MainActivity.java* et sont mises à jour à chaque fois qu'une nouvelle époque du filtre de Kalman est calculée.

getCameraPosition

- **Exécution** : Chaque fois qu'une autre fonction à besoin des coordonnées de la caméra ;
- **Entrée** : *X_kalman*, *Y_kalman*, *Z_kalman*, *roll*, *pitch*, *yaw* ;
- **Sortie** : liste contenant les coordonnées X/Y/Z de la caméra ainsi que les corrections du bras de levier pour passer des coordonnées Kalman(centre du drone) aux coordonnées caméra ;
- **Détails** : Chaque fois que la fonction est appelée, la position de la caméra est calculée à partir des dernières coordonnées calculées par le filtre de Kalman ainsi que des attitudes Roll/Pitch/Yaw qui permettent de calculer le bras de levier.

ImportDataNavigation

- **Exécution** : Bouton "2.PLAN DE VOL" sur la page principale ;
- **Entrée** : lien URI du fichier de plan de vol stocké sur le smartphone ;

— **Sortie** : Dictionnaire contenant les coordonnées des waypoints, le cap, l'orientation du gimbal ;
— **Détails** : Le dictionnaire créé est disponible pour l'ensemble des fonctions du fichier *MainActivity.java*.

Navigation

— **Exécution** : Bouton "5.GUIDAGE" sur la page principale ;
— **Entrée** : coordonnées X/Y/Z de la caméra via *getCameraPosition*, le plan de vol à suivre via *ImportDataNavigation*, *cap* ;
— **Sortie** : - ;
— **Détails** : Une fois lancée, la fonction tourne en boucle et calcule les corrections à appliquer sur le drone pour suivre le plan de vol donné. Les commandes de correction sont directement envoyées au drone dans cette fonction.

sendRotateGimbalCommand

— **Exécution** : Chaque fois que la fonction *Navigation* à besoin d'effectuer une prise de vue ;
— **Entrée** : orientation *pitch* du gimbal ;
— **Sortie** : Exécution de la fonction *takePhoto* une fois le gimbal orienté ;
— **Détails** : Chaque fois que la fonction est appelée, une orientation *pitch* du gimbal est donné par rapport au plan de vol.

takePhoto

— **Exécution** : Chaque fois que la fonction *Navigation* à besoin d'effectuer une prise de vue, via la fonction *sendRotateGimbalCommand* ;
— **Entrée** : coordonnées X/Y/Z de la caméra via *getCameraPosition*, *vitesseX*, *vitesseY*, *vitesseZ*, *X_kalman_precision*, *Y_kalman_precision*, *Z_kalman_precision* ;
— **Sortie** : - ;
— **Détails** : Chaque fois que la fonction est appelée, une image est capturée. Au moment de la capture, la position de la caméra est enregistrée et une précision de positionnement est calculée à partir des précisions de Kalman et des vitesses instantanées du drone.

StopAll

— **Exécution** : Bouton "STOP GUIDAGE" sur la page principale ;
— **Entrée** : - ;
— **Sortie** : - ;
— **Détails** : Quand cette fonction est exécutée, l'ensemble des autres fonctions de guidage tournant dans des boucles sont stoppées.

Fonction annexes utilisées par les fonctions présentées précédemment :

onActivityResult

— **Exécution** : Lorsque les paramètres sont entrés par l'utilisateur dans la page dédiée ;
— **Entrée** : récupération des champs de la page de gestion des paramètres ;
— **Sortie** : l'ensemble des paramètres entrés par l'utilisateur sous forme de variables ;
— **Détails** : Les différents paramètres entrés par l'utilisateur sont disponibles pour l'ensemble des autres fonctions sous forme de variables.

getTimeStamp

- **Exécution** : Chaque fois qu'une autre fonction à besoin d'avoir un timestamp ;
- **Entrée** : - ;
- **Sortie** : Une liste contenant le timestamp sous forme de temps UNIX brute et de temps au format hh :mm :ss.ssss ;
- **Détails** : - .

writeFileOnInternalStorage

- **Exécution** : Chaque fois qu'une autre fonction à besoin d'enregistrer une information dans un fichier .txt stocké sur le smartphone ;
- **Entrée** : le nom du fichier .txt dans lequel l'information doit être enregistrée + l'information à enregistrer sous forme de *string* ;
- **Sortie** : - ;
- **Détails** : - .

MN95toWGS84

- **Exécution** : Chaque fois qu'une autre fonction à besoin de transformer des coordonnées MN95 en WGS84 pour effectuer des représentations sur la carte Google Map ;
- **Entrée** : Coordonnées *Est*, *Nord* MN95 ;
- **Sortie** : Une liste contenant les coordonnées WGS84 latitude/longitude ;
- **Détails** : Cette transformation est uniquement nécessaire pour effectuer des représentations cartographiques. L'ensemble des calculs sont effectués en coordonnées MN95. La formule approchée de transformation est donc utilisée pour simplifier le processus [6].

MapUpdatePhotoPoint

- **Exécution** : Chaque fois que la fonction *takePhoto* effectue une prise de vue ;
- **Entrée** : - ;
- **Sortie** : - ;
- **Détails** : Quand une prise de vue est effectuée, un symbole est ajouté sur la carte à cet emplacement.

addNavigation

- **Exécution** : Lorsque la fonction *ImportDataNavigation* est exécutée ;
- **Entrée** : lien URI du fichier de plan de vol stocké sur le smartphone ;
- **Sortie** : objet *polyline*, *cercle* ;
- **Détails** : Lorsque cette fonction est exécutée, les objets nécessaires à la représentation cartographique du plan de vol sont créés. Il s'agit de polylines et de cercles sur les waypoints.

MapUpdateTrajectoire / MapUpdateTrajectoirePoint

- **Exécution** : Lorsque la fonction *addNavigation* est exécutée ;
- **Entrée** : objet *polyline*, *cercle* créés par la fonction *addNavigation* ;
- **Sortie** : - ;
- **Détails** : Lorsque cette fonction est appelée, les objets sont représentés sur la carte.

6. https://www.swisstopo.admin.ch/content/swisstopo-internet/fr/topics/survey/reference-systems/switzerland/_jcr_content/contentPar/tabs/items/dokumente_publikatio/tabPar/downloadlist/downloadItems/516_1459343097192.download/ch1903wgs84_f.pdf

B.2.6 Paramètres avancés

Lors des Chapitres 2.6.1 et 3.1.4 des constantes de temps sont introduites. Ces constantes qui ont pour but de compenser un retard de temps sont introduites dans deux variables :

— **ConstantePrecisionImage** : cette constante permet de calculer la précision du positionnement lorsqu'une prise de vue est effectuée. Elle est déterminée dans le Chapitre 2.6.1 ;
— **ConstanteIntervalleImage** : cette constante à pour but de compenser des décalages dans les intervalles de prises de vue. Elle est l'addition de la compensation de 0.19 seconde du Chapitre 2.6 et de la compensation de 0.45 seconde du Chapitre 3.1.4.

Ces constantes sont déterminées pour le drone DJI Matrice 210 V2 avec la caméra Zenmus X4S. Il n'est pas certains que ces constantes restent les mêmes avec l'utilisation d'un autre drone ou d'une autre caméra. Il faudrait alors refaire les tests des Chapitres 2.6, 2.6.1 et 3.1.4 pour redéterminer ces constantes et les introduire dans le fichier *MainActivity.java*. Si un autre drone est utilisé sans utiliser le mode de capture d'image par intervalle, ces constantes n'entrent pas en jeu et n'ont donc pas besoin d'être redéterminées.

B.2.7 Utilisation du simulateur de station totale via la fonction "Socket"

Le script Python *PCSimDJI.py* est disponible sur le lien suivant : `https://www.dropbox.com/sh/mfuv7zg57ax8kyo/AAAI19ahnfTzy0TGwihyDfmMa?dl=0`

La fonction Socket implémentée permet de communiquer avec un script Python afin de recevoir des coordonnées depuis le simulateur DJI (Chapitre 2.4.8). Les étapes pour utiliser le simulateur sont les suivantes :

1. **Démarrer DJI Assistant** : Il faut tout d'abord relier le drone par USB au PC puis lancer le logiciel *DJI Assistant*. Le modèle du drone apparaît alors à l'écran et est sélectionnable (Figure B.17) ;
2. **Démarrer le simulateur** : Aller dans l'onglet "Simulator", puis sélectionner le bouton "Open" (Figure B.18). Une fois l'interface de simulation ouverte (Figure B.19), il est possible de démarrer la simulation avec le bouton "Start Simulating". Différentes informations en temps réel sont affichées dans la fenêtre. Il est également possible de modifier certains réglages en faisant un clic droit dans l'interface 3D –> Setup. Dans cette fenêtre l'emplacement de l'enregistrement du fichier "trace" est affiché ;

FIGURE B.17 – Démarrage DJI Assistant.

FIGURE B.18 – Démarrage du simulateur.

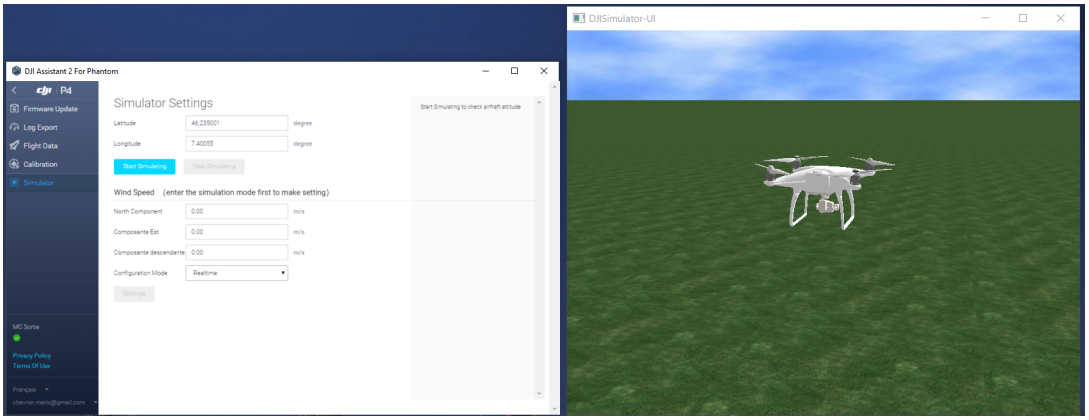


FIGURE B.19 – Interface de simulation.

3. **Démarrer la diffusion des coordonnées** : Une fois le simulateur démarré, les coordonnées calculées sont enregistrées dans le fichier "trace". Un script Python *PCSimDJI.py*, permet de récupérer ces coordonnées et de les diffuser à un smartphone. Les paramétrages à effectuer pour la diffusion des coordonnées sont les suivants :
   (a) Activer un point d'accès mobile sur le smartphone ;
   (b) Se connecter sur le Wifi créé par le smartphone avec le PC ;
   (c) Dans le script Python, modifier la variable *SERVER* qui correspond à l'adresse IP du PC (il est possible d'afficher l'adresse IP du PC dans les paramètres du point d'accès mobile du smartphone). Il faut également vérifier le chemin d'accès au fichier "trace" au niveau de la variable *newest* ;
   (d) Si besoin, modifier les variables *x_ref*, *y_ref*, *z_ref* du fichier *PCSimDJI.py*. Les coordonnées du point de départ du drone sont à 0,0,0. Ces trois variables permettent d'appliquer une translation sur le référentiel afin d'obtenir des coordonnées proches de la réalité (proche du plan de vol à suivre) ;
   (e) Modifier les variables *SocketIP* et *SocketPort* dans le fichier *MainActivity.java* pour les faire correspondre aux informations d'IP et de port du serveur entrée dans le script *PCSimDJI.py*, puis compiler l'application ;

(f) Exécuter le script *PCSimDJI.py*. Une fois le script lancé, celui-ci attend que le smartphone demande une connexion ;

(g) Sur le smartphone, lorsque l'application est lancée, sur la page principale, le bouton "SOCKET" permet de démarrer la connexion avec le PC. Une fois la connexion démarrée, les coordonnées du simulateur sont reçues en continue à 10 HZ. Les coordonnées X/Y/Z reçues sont enregistrées dans les mêmes variables que les observations terrestres. Il n'y a donc pas de différence pour les autres fonctions de l'application entre les coordonnées X/Y/Z simulées et réelles.

B.2.8 Développement potentiel

De nombreuses fonctions seraient intéressantes à implémenter. Voici une liste non-exhaustive des fonctions qui n'ont pas pu être implémentée par manque de temps :

- **Contrôle de la réception des observations**. Dans la fonction de calcul de la trajectoire du drone, *KalmanLive* : mettre en place une fonction de sécurité qui contrôle la bonne réception des observations de base (station totale + drone). Dans le cas où il y aurait un problème avec la station totale, la trajectoire continue d'être calculée par le filtre de Kalman uniquement sur la base des vitesses renvoyées par le drone. Cette situation implique un dérive de la trajectoire. La sécurité mise en place devrait se déclencher si après X secondes aucune nouvelle observation n'est reçue. Une fois la sécurité déclenchée, le guidage automatique devrait être stoppé ;
- **Contrôle de la station totale pour le suivi de prisme**. Dans la situation ou le prisme serait perdu par la station totale, les coordonnées du filtre de Kalman pourraient être utilisées pour orienter l'instrument en direction du prisme ;
- **Contrôle des paramètres**. Dans la page de gestion des paramètres, au moment ou l'utilisateur rempli les différents champs, un contrôle pourrait être effectué. Ce contrôle se porte surtout sur le paramètre d'intervalle de prise de vue en fonction de la vitesse maximale. Il faut que le temps entre deux prises de vue soit suffisant (contrainte technique de 2 secondes). Si le temps est trop court, un message pourrait avertir l'utilisateur ;
- **Page de paramètres avancés**. Une page supplémentaire contenant des paramètres avancés pourrait être rajoutée afin de gérer plus facilement les variables liées à des constantes de temps et à l'utilisation du simulateur de station totale (Chapitre B.2.6, B.2.7) ;
- **Représentation graphique de l'état du système**. L'état de chaque fonction pourrait être représenté par des couleurs sur les différents boutons de la page principale. Ainsi l'utilisateur a une vue d'ensemble sur le fonctionnement de l'ensemble des processus (observation terrestre, observation drone, Kalman, guidage) ;
- **Gestion des fichiers enregistrés lors du vol**. Actuellement les fichiers .txt enregistrés lors du vol sont uniquement accessibles à partir d'Android Studio. Il serait intéressant de mettre en place un processus permettant d'envoyer ces fichiers du smartphone sur un autre périphérique directement depuis l'application.

Annexe C

Annexes Numériques

L'ensemble des annexes numériques sont disponibles sur le lien suivant : `https://www.dropbox.com/sh/mfuv7zg57ax8kyo/AAAI19ahnfTzy0TGwihyDfmMa?dl=0`.

- *AppTotal* : dossier contenant l'ensemble des fichiers de base pour compiler l'application sur Android Studio ;
- *Post_traitement* :
  - *Export_Agisoft_16m.txt* : fichier d'exemple ;
  - *Photo_16m-1.txt* : fichier d'exemple ;
  - *PID_16m-1.txt* : fichier d'exemple ;
  - *Post_traitement.py* : Le script Python pour effectuer le post-traiement.
- *DroneControleMeric.apk* : L'application compilée qui peut directement être installée sur un smartphone Android ;
- *PCSimDJI.py* : Le script Python pour le simulateur de station totale.

Bibliographie

[Babu et al., 2017] Babu, V. M., Das, K., and Kumar, S. (2017). Designing of self tuning pid controller for ar drone quadrotor. In *2017 18th international conference on advanced robotics (ICAR)*, pages 167–172. IEEE.

[Guillaume, 2022] Guillaume, S. (2022). Méthodes d'estimation et réseaux géodésiques.

[Jin et al., 2018] Jin, Y.-H., Ko, K.-W., and Lee, W.-H. (2018). An indoor location-based positioning system using stereo vision with the drone camera. *Mobile Information Systems*, 2018.

[Motlagh et al., 2019] Motlagh, H. D. K., Lotfi, F., Taghirad, H. D., and Germi, S. B. (2019). Position estimation for drones based on visual slam and imu in gps-denied environment. In *2019 7th International Conference on Robotics and Mechatronics (ICRoM)*, pages 120–124. IEEE.

[Pan, 2021] Pan, Y. (2021). Measuring drone trajectory using total station with visual tracking. *Institute of Geodesy and Photogrammetry, ETH Zürich*.

[Vanhie-Van Gerwen et al., 2021] Vanhie-Van Gerwen, J., Geebelen, K., Wan, J., Joseph, W., Hoebeke, J., and De Poorter, E. (2021). Indoor drone positioning : Accuracy and cost trade-off for sensor fusion. *IEEE Transactions on Vehicular Technology*, 71(1) :961–974.

[Xin et al., 2020] Xin, C., Wu, G., Zhang, C., Chen, K., Wang, J., and Wang, X. (2020). Research on indoor navigation system of uav based on lidar. In *2020 12th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, pages 763–766. IEEE.