Hes·so VALAIS WALLIS
Haute Ecole de Gestion
Hochschule für Wirtschaft

Filière Informatique de gestion
Studiengang Wirtschaftsinformatik

Travail de Bachelor 2020

Détection d'obstacles à basse altitude pour drone en utilisant du machine learning

Étudiant : Jean-Marie Alder

Professeur : Dominique Genoud

Travail rendu le 31 juillet 2020

Résumé

Étant donné la croissance de l'utilisation des drones dans le milieu agricole et l'utilité de ceux-ci, il semble convenable d'étudier les possibilités d'amélioration de l'autonomie de ces machines. Cet travail explore la possibilité d'utiliser des capteurs et de l'apprentissage automatique afin de détecter en temps réel les obstacles autour de l'appareil et de maintenir une distance de sécurité par rapport aux plantations. Nous allons utiliser la méthode itérative « CRISP-DM » spécialisée pour l'apprentissage automatique afin de développer et tester un modèle qui permet de prédire si le drone est trop proche du sol ou d'un obstacle et nécessite une manœuvre d'évitement. Ce document comprend un état de l'art sur les types de senseurs efficaces en situation de traitement agricole, sur les programmes de systèmes d'informations géographiques qui permettent une visualisation des données géolocalisées et sur la détection d'obstacles embarquée pour drones agricoles. Nous allons ensuite présenter le processus effectué en passant par la compréhension des données, la visualisation de la trajectoire du drone et des mesures récupérées par les senseurs, la transformation et la classification des données ainsi que l'entraînement et l'évaluation de plusieurs modèles de prédiction. Pour finir, nous allons démontrer que l'apprentissage automatique à bord d'un drone permet d'améliorer la reconnaissance et l'évitement d'obstacles en temps réel.

Mots-clés : drone, senseur, détection d'obstacle, apprentissage automatique, système d'information géographique

Avant-propos

Ce document à été réalisé dans le cadre d'un travail de Bachelor à la HES-SO Valais en informatique de gestion en partenariat avec « Aero 41 », une entreprise spécialisée dans le traitement des vignes avec des drones. L'objectif principal de ce travail est d'étudier les possibilités d'améliorations du système de détection d'obstacles déjà présent sur les machines en utilisant des senseurs et de l'apprentissage automatique. Au début de cette étude, le drone proposé par « Aero 41 » dispose d'une version en production ainsi qu'en développement. Le drone de développement dispose d'une architecture fonctionnelle avec notamment un ordinateur d'accompagnement relié à une caméra pour effectuer de la reconnaissance d'image en temps réel. Ce système est capable de reconnaîtres les arbres sur sa trajectoire, mais celui-ci ne permet pas d'évaluer avec précision le terrain et l'altitude du drone. La difficulté principale se trouve dans le relief du terrain qui varie fortement entre le haut des ceps de vigne et le sol. Les autres défis de cette étude sont notamment d'utiliser des senseurs qui peuvent être intégrés à un drone, qui favorisent le développement d'une couche logicielle personnalisée et qui respectent le budget défini à l'avance.

Remerciements

Je voudrais dans un premier temps remercier M. Dominique Genoud, professeur à la HES-SO Valais, pour sa patience, sa confiance, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter ce travail et mes connaissances. M. Jérome Treboux, doctorant à la HES-SO, pour sa disponibilité, sa flexibilité et son aide technique précieuse lors du développement de ce projet. L'équipe de « Aero 41 » qui m'a accueilli durant les vols d'essais et qui m'a présenté en détail le système sur lequel nous avons travaillé. Mme Émilie Fournier qui a accepté avec générosité de relire la syntaxe et l'orthographe de ce document durant ses vacances. Et, finalement, toute ma famille et mes amis qui m'ont soutenu moralement tout au long de cette étude.

Table des matières

Liste des figures

Liste des abréviations

Unmanned aerial vehicle	UAV
Machine learning	ML
Deep learning	DL
Cross-industry standard process for data mining	CRISP-DM
Product owner	PO
Product backlog	PB
Companion computer	CC
Système d'information géographique	SIG / GIS
Inertial measurement mnit (gyroscope)	IMU
Light detection and ranging	Lidar
Global Positioning System	GPS
Software development kit	SDK
Keyhold Markup Language	KML
Area under curve	AUC

Glossaire

Machine learning	Apprentissage automatique, système qui apprend automatiquement à prendre des décisions en fonction de son expérience
Deep learning	Forme d'apprentissage automatique qui utilise des réseaux de neurones
Dual boot	Multi-booting, permet de choisir entre deux systèmes d'opération au démarrage de l'ordinateur
Open source	En informatique : qui autorise l'utilisation et la modification du code source et du contenu d'un programme
Workflow	Séquences de processus pour traiter des données
CRISP-DM	Processus standard pour effectuer des tâches d'apprentissage automatique
Détection d'obstacles	Processus qui utilise des senseurs et des algorithmes pour détecter les objets ou le terrain sur la trajectoire d'un drone
Évitement d'obstacles	Actions mises en oeuvre pour éviter la collision, par exemple en effectuant une manœuvre d'évitement
Software development kit	Collection d'outils pour le développement logiciel
Keyhold Markup Language	Format standard de fichier pour des données géolocalisées

1. Introduction

Le domaine de l'agriculture ne cesse d'évoluer dans une période où les avancements technologiques sont quotidiens. Lors de ces dernières années, les drones sont devenus des outils performants et accessibles au grand public. Ils se sont alors fait une place dans le milieu agricole pour aider à diverses tâches. Certains ont notamment été conçus dans le but de transporter et de répandre des produits de traitement sur les plantations.

De manière générale, le traitement et l'épandage des plantations se font de manière terrestre à l'aide de pulvérisateurs manuels ou de machines spécialisées. Néanmoins, lorsque les cultures sont difficiles d'accès, notamment dans les régions montagneuses, une aide aérienne est nécessaire. Il est commun d'utiliser des hélicoptères pour traiter ces zones reculées. Cette technique a le désavantage d'être peu précise et de gaspiller plus de la moitié du produit de traitement à cause du fort vent créé par les pales de l'appareil en fonction. Un drone agricole permet de palier à ce problème étant donné qu'il est plus facile de le manipuler précisément et que les hélices produisent moins de turbulences dans l'air.

Cette étude a été réalisée en partenariat avec « Aero 41 », une start-up suisse spécialisée dans les drones agricoles de pulvérisation. Cette entreprise propose d'ores et déjà des services de traitement pour la vigne. Leur drone peut être piloté de manière manuelle et à l'aide d'un mode automatique à paramétrer sur-place. Récemment, le système a été complété par une reconnaissance d'obstacles à l'aide d'une caméra et de techniques informatiques d'apprentissage automatique. Cette technique permet d'éviter les obstacles qui se trouvent sur la trajectoire en face du drone. Il reste néanmoins une zone sous le drone qui n'est pas couverte par la reconnaissance actuelle.

Pour garantir un traitement optimisé, il est important que la pulvérisation se fasse proche des plantations et toujours à la même altitude. Le problème avec les vignes est que la distance entre le sol et le sommet des ceps varie fortement. Un capteur placé de manière verticale vers le sol aura tendance à retourner des hauteurs variables et ne permet pas à lui seul de garantir une altitude constante par rapport au point le plus haut des cultures.

D'après ces différents éléments, nous avons émis l'hypothèse suivante : « L'apprentissage automatique permet d'améliorer la détection d'obstacle d'un drone agricole à basse altitude et de garder une distance constante par rapport aux cultures ». Cette thèse propose une méthode afin de tester et vérifier cette proposition.

1.1. Méthodologie

Étant donné la liberté dans le choix de méthodologie, nous avons estimé au début de l'étude qu'un mélange entre un développement agile et « waterfall » (en cascade) serait la technique la plus efficace pour atteindre notre but. Nous avons sélectionné la méthode agile « Scrum » car l'auteur du document est déjà habitué à utiliser celle-ci. Puisque cette technique demande par principe une équipe de trois personnes au minimum, nous avons dû l'adapter à notre situation en combinant les procédés et artefacts « Scrum » avec une approche « Waterfall » plus linéaire.

Nous avons donc mis en place un « product backlog » (carnet du produit, ou PB) contenant les objectifs principaux à atteindre et nous avons fixé une itération (sprint) à une semaine. À la fin de chaque sprint, nous avons fixé une réunion avec le « product owner » (PO) afin de valider le travail accompli et de fixer les nouveaux objectifs pour la semaine suivante. Un journal de bord disponible en annexe de ce document a été mis en place afin de garder un historique complet des tâches effectuées ainsi qu'un bref résumé des séances avec le PO contenant les problèmes rencontrés ainsi que les solutions envisagées.

Afin de compléter cette méthodologie avec une technique spécifique à l'apprentissage automatique (ML), nous avons choisi le modèle de processus « Cross Industry Standard Process for Data Mining » (CRISP-DM) qui est un standard pour les professionnels en apprentissage automatique et qui permet un développement hiérarchique et itératif tout en gardant une flexibilité dans les itérations en fonction des résultats (Bošnjak, Grljević, & Bošnjak, 2014).

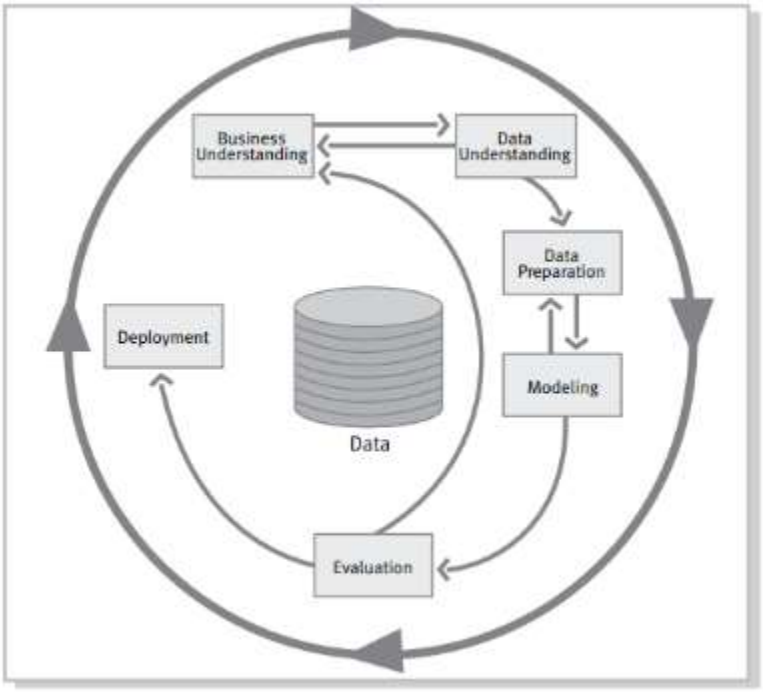

Figure 1: étapes de la méthode CRISP-DM, récupérée sur https://www.the-modeling-agency.com/crisp-dm.pdf

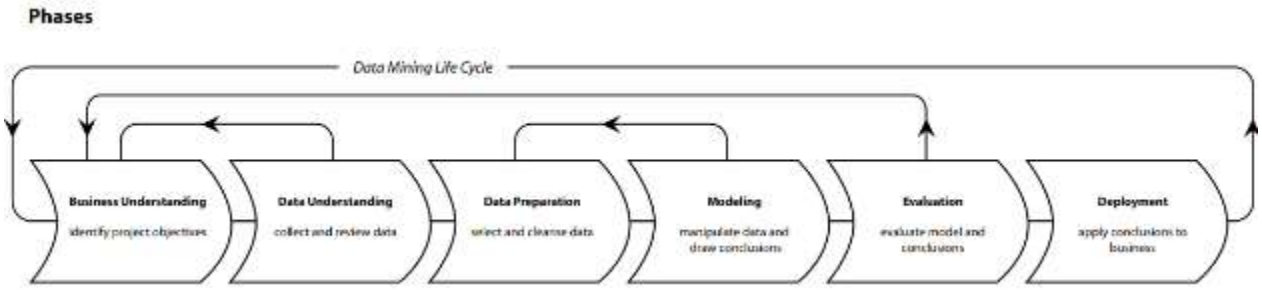

Figure 2: étapes de la méthode CRISP-DM détaillées, récupéréé sur https://exde.wordpress.com/2009/03/13/a-visual-guide-to-crisp-dm-methodology/

À noter que le cycle de vie et les liaisons (flèches) sur les schémas ci-dessus proposent uniquement un aperçu des relations entre les différentes tâches car, en fonction du but de l'étude, toutes les phases peuvent être reliées dans un ordre différent suivant la situation (Champman, et al., 2000).

1.1.1. Étude

Cette étude nécessite des connaissances et des compétences techniques variées, notamment les drones et de la représentation de ceux-ci dans l'espace, le développement de scripts en Python et de l'apprentissage automatique embarqué. L'auteur, n'ayant pas toutes ces compétences lors du début de ce travail, s'engage à se former et se documenter sur les thèmes nécessaires dans l'accomplissement de cette étude.

1.1.2. Définitions

Nous allons définir les termes et concept utilisés tout au long de cette étude afin de mieux comprendre le problème et de mieux cerner les concepts mis en œuvre. Nous avons sélectionné les définitions qui sont primordiales pour la bonne compréhension de ce document.

1.1.3. État de l'art

Avant d'implémenter notre système, nous souhaitons avoir une vision globale des techniques et programmes déjà présents sur le marché afin de s'orienter vers les solutions les plus optimales. Nous avons défini trois parties différentes pour mieux séparer nos besoins. La première partie porte sur le type de senseur à utiliser. La seconde va déterminer quel outil logiciel est le plus efficace dans la visualisation de données géographiques. La dernière va examiner les solutions existantes qui permettent une détection d'obstacle en temps réel avec des senseurs radar et lidar.

Nous allons effectuer des tests en situation dans la mesure du possible sur les senseurs que nous possédons et nous allons procéder à des tests de logiciels qui permettent une visualisation du parcours du drone ainsi que les obstacles détectés par les senseurs en 3D.

1.1.4. Choix technologique

D'après les résultats obtenus durant l'état de l'art, nous allons dresser un tableau comparatif en se basant sur des critères de sélction et une pondération de ceux-ci en fonction des besoins de notre étude. Nous évaluerons aussi les différentes restrictions matérielles, logicielles et budgétaires qui résultent de notre cas de figure.

1.1.5. Implémentation

Nous allons ensuite proposer une proposition d'implémentation qui permet d'améliorer la reconnaissance et l'évitement d'obstacles lors de vols de drones agricoles à basse altitude à l'aide de l'apprentissage automatique. Nous allons détailler les différentes étapes effectuées pour obtenir le résultat voulu.

Vu la Figure 1 sur la méthodologie CRISP-DM, nous allons procéder en commençant par la récolte de données sur le terrain, la compréhension ainsi que la visualisation des données. Nous allons ensuite transformer ces données afin de créer un modèle d'apprentissage automatique. Les données transformées seront utilisées dans le but d'entraîner et d'améliorer notre modèle. Nous procéderons ensuite à une évaluation des résultats obtenus via des simulations et, si possible, sur le terrain. Si nécessaire, nous allons répéter ce processus jusqu'à obtenir un modèle fiable. Lorsque nous aurons atteint des résultats positifs, nous allons effectuer un déploiement de ce système à bord du drone.

2. Définitions

Cette section permet aux lecteurs de se familiariser avec les termes utilisés tout au long de cette étude.

2.1. Drone

Un drone, ou « Unmanned Aerial Vehicle » (UAV) en anglais, est un appareil volant sans pilote qui peut être contrôlé à distance via une télécommande radio ou de manière autonome en suivant une route définie à l'avance (Kardasz, Doskocz, Hejduk, Wiejkut, & Zarzycki, 2016). Cette étude se concentre uniquement sur la définition anglaise du drone qui ne prend pas en compte les véhicules terrestres ou sous-marins. Nous allons donc utiliser le terme "drone" pour signifier un appareil volant sans pilote.

Figure 3: Drone agricole de la startup Aero41 en situation de traitement sur des vignes, image récupérée sur https://www.aero41.ch/

Les drones sont utilisés dans de nombreuses situations, notamment de la reconnaissance de terrain, de l'aide humanitaire et de l'agriculture (Food and Agriculture Organization of the United Nations, 2018). Cette étude porte sur les drones agricoles qui permettent un traitement des cultures difficile d'accès par voie terrestre. Ceux-ci permettent d'effectuer plusieurs opérations, notamment des missions d'observation et d'inspection des champs ainsi que des travaux d'épandage (ABot, 2020)

2.1.1. Fonctionnement d'un drone agricole

De manière générale, les drones agricoles sont composés de plusieurs composants interconnectés qui effectuent une action particulière et qui communique avec les autres composants. Les parties principales du drone sont les suivantes: le système d'alimentation qui alimente en énergie l'appareil, le système de communication qui est relié à la radiocommande, le système de propulsion qui gère les moteurs des hélices et le controleur de vol et l'ordinateur de bord ("Flight Control Unit") qui s'occupe de stabiliser le drone et de déterminer sa position GPS (Arbellay, 2019). Afin d'effectuer des calculs et des processus supplémentaires en vol, il est possible d'ajouter un ordinateur d'accompagnement ("companion computer") relié à l'ordinateur de bord. Relié à des senseurs, celui-ci nous permet entre autres d'effectuer de la reconnaissance et de l'évitement d'obstacle en temps réél en analysant les mesures récupérées et en communiquant si nécessaire avec l'ordinateur de bord pour effectuer une manœuvre d'évitement.

2.2. Senseur

Par définition, un senseur est un mécanisme qui reçoit des stimulus et qui renvoie une réponse à l'aide d'un signal électrique (Fraden, 2010). En d'autres termes, un senseur est un type de capteur qui permet de convertir une mesure physique en un signal ou une donnée qui peut être lue ou utilisée par un observateur ou un système (Chen, Janz, Zhu, & Brychta, 2012). Il existe un vaste choix de types de senseur avec une multitude d'utilisations possibles. Nous allons nous focaliser sur deux types que nous avons utilisés durant cette étude : les radars et les lidars.

2.2.1. Radar

Un radar est un type de senseur qui permet de détecter et de calculer la distance et la position d'objets sur sa trajectoire en utilisant la bande des ultra haute fréquence (UHF) ou celle des micro-ondes dans le spectre des fréquences radio pour recréer des cartes et des représentations de l'environnement de manière précise (Rouse, 2005). Le radar envoie des ondes électromagnétiques et récupère l'écho retrourné après avoir été reflété par l'obstacle afin de mesurer le temps écoulé entre l'émission et la réception (Skolnik, 1990).

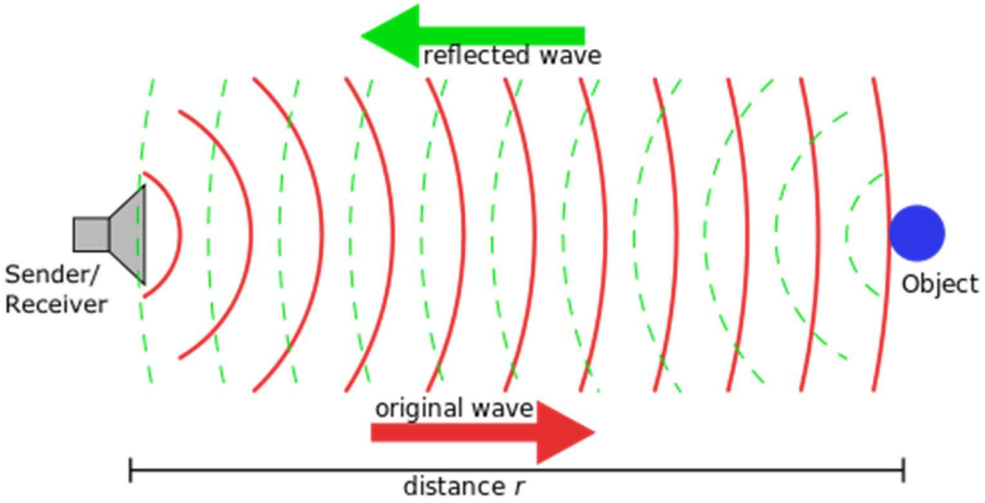

Figure 4: fonctionnement d'un radar, calcul de la distance en fonction du temps de l'écho des ondes, image de Georg Wiora récupérée sur https://commons.wikimedia.org/wiki/File:Sonar_Principle_EN.svg

Le type de radar qui va nous intéresser est celui qui utilise des micro-ondes (entre 300 MHz et 300 GHz) car nous avons effectué cette étude à l'aide de ceux-ci. Cette technologie permet entre autres de fonctionner dans des conditions atmosphériques difficiles et offre une certaine capacité à pénétrer les obstacles afin de détecter des mouvements derrière une surface (Zhengyu & Changzhi, 2019). Le traitement avec un drone en milieu agricole génère un nuage de liquide conséquent aux abords de l'appareil, il est donc crucial que les senseurs ne soient pas altérés par ces éléments pour effectuer une reconnaissance efficace du terrain et des obstacles sur sa trajectoire.

2.2.2. Lidar

Lidar est l'acronyme en anglais du terme « Light detection and ranging » qui signifie télédétection par laser. Un lidar peut être défini comme étant un senseur qui émet des rayons de lumière pour déterminer la distance jusqu'à la cible (Gatziolis & Andersen, 2008). Tout comme le radar à micro-ondes décrit au point précédant, le système est constitué d'un récepteur et d'un émeteur et le calcul de la distance se fait en multipliant la vitesse de la lumière avec le temps pris par l'écho d'une pulsation laser après avoir reflété sur la cible (Wandinger, 2005).

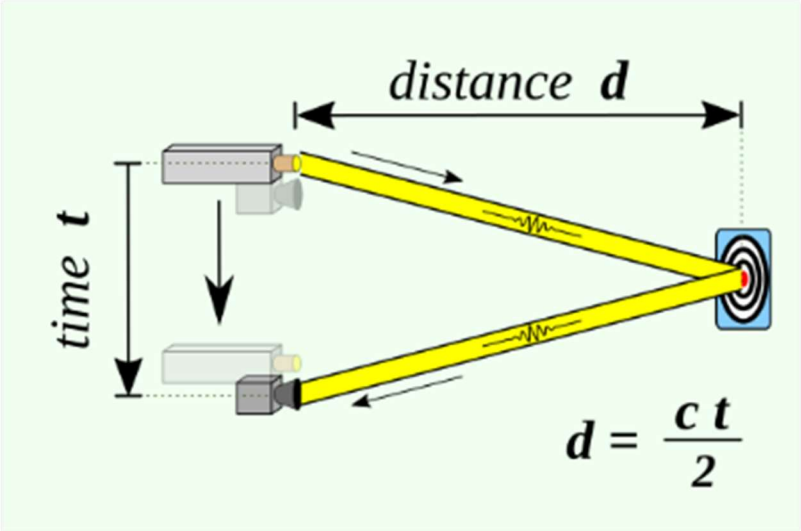

Figure 5: calcul de la distance à l'aide d'un lidar et du principe "Time of Flight", image de [RCraig09] récupérée sur https://commons.wikimedia.org/wiki/File:20200501_Time_of_flight.svg

Il existe plusieurs situations dans lesquelles les senseurs lidar peuvent être utilisés, notamment le "mapping" de terrain à l'aide de représentations sous forme d'un nuage de points, le suivi des cultures ou des forêts et la détection d'obstacles (Gatziolis & Andersen, 2008). Durant cette étude, nous allons tester et déterminer si l'utilisation de la technologie lidar permet d'améliorer la reconnaissance d'obstacle à bord d'un drone. Nous allons aussi utiliser les données récupérées par ces senseurs afin de visualiser l'environnement autour de la machine en vol sur un globe 3D.

2.3. Machine Learning

Le « machine learning » ou apprentissage automatique en français désigne un concept en informatique dont le but est d'apprendre à une application à prendre des décisions de manière autonome en lui donnant des données d'entraînement (Shalev-Shwartz & Ben-David, 2014). Dans le domaine de l'apprentissage automatique supervisé, les données d'entraînement sont labellisées, c'est-à-dire que la décision du classement des valeurs a été effectué manuellement et le but est que le programme apprenne par lui même à classer celles-ci dans le futur en trouvant des relations entre les différentes informations à disposition (Expert System Team, 2017). Il existe plusieurs algorithmes de "machine learning" adaptés à différentes situations et il s'agit de trouver lequel d'entre eux offre les meilleurs résultats avec le moins de marge d'erreur possible (Smola & Vishwanathan, 2008).

Dans notre cas de figure, l'apprentissage automatique peut nous permettre d'apprendre à l'ordinateur de bord du drone à identifier les obstacles pour éviter d'entrer en collision avec celui-ci.

2.4. Système d'information géographique

Il n'existe pas de définition exacte pour les systèmes d'information géographique (SIG). Néanmoins, il est communément accepté qu'il s'agit d'un système informatique désigné à analyser, manipuler et modéliser des données géoréférencées afin de résoudre des problèmes complexes de planning et de gestion du terrain (Goodchild & Kemp, 1990). Les SIG permettent à l'utilisateur de combiner des couches de données (Figure 6) contenant des informations sur le terrain et la carte ainsi que les informations géographiques que nous souhaitons représenter afin de refléter au mieux la réalité (Escobar, Hunter, Bishop, & Zerger, 1998).

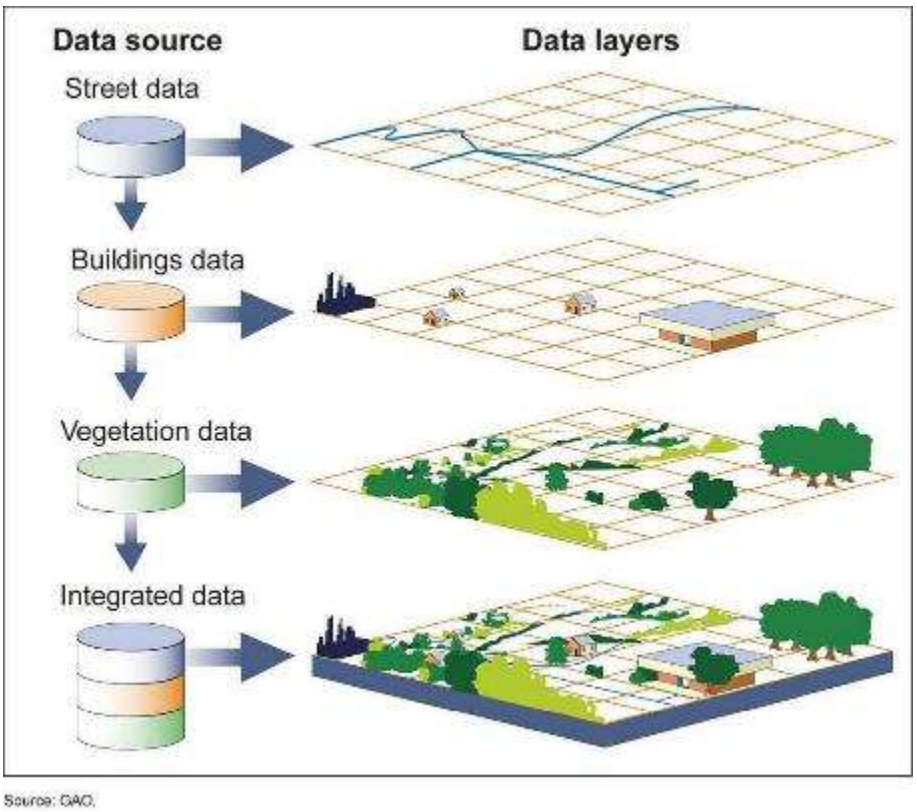

Figure 6: création d'une carte virtuelle à l'aide de plusieurs couches de données sur un SIG, image récupérée sur https://estavela.in.rs/view/routes.php?page=showgis

Il existe un grand choix de SIG disponible sur le marché. Nous y trouvons des logiciels « open source » gratuits tout comme des programmes propriétaires payants. Le plus populaire pour le grand public est « Google Earth Pro », un logiciel gratuit de visualisation de la terre sous forme de globe 3D sur lequel l'utilisateur peut déplacer son angle de vue librement.

Nous souhaitons utiliser un SIG dans le but de représenter les données de vol du drone, notamment la trace GPS et les obstacles détectés par les senseurs. Cela va nous permettre de visualiser précisément les données sur lesquelles nous allons travailler par la suite. Par exemple, nous allons pouvoir identifier les situations qui représentent un danger de collision et qui nécessitent une

manœuvre d'évitement afin de pouvoir classifier manuellement les données. Une grande partie de cette étude va porter sur la manière de visualiser l'espace autour du drone le plus précisément possible.

2.5. Axes principaux des aéronefs – Roll, Pitch, Yaw

Durant cette étude, nous avons travaillé avec des mesures radars et lidar bruts que nous avons transformées en représentation 3D. Les informations dont nous disposons sont les suivantes : les distances et les angles mesurés par les senseurs, la position GPS, l'altitude du drone et les mesures des axes de rotation de l'appareil.

Afin de représenter l'orientation d'un véhicule aérien dans l'espace en 3D, nous allons utiliser les trois axes principaux de rotation appelés également les angles d'Euler. Pour un aéronef, nous pouvons définir un système de coordonnées en trois dimensions dans lequel les axes partent depuis le centre de gravité de l'appareil avec chaque axe perpendiculaire aux autres (NASA, 2015). L'axe « yaw » qui représente une rotation du nez de l'avion à gauche ou à droite, l'axe « pitch » qui représente un mouvement du nez vers le haut ou vers le bas et l'axe « roll » qui représente une rotation des ailes de haut en bas (NASA, 2015).

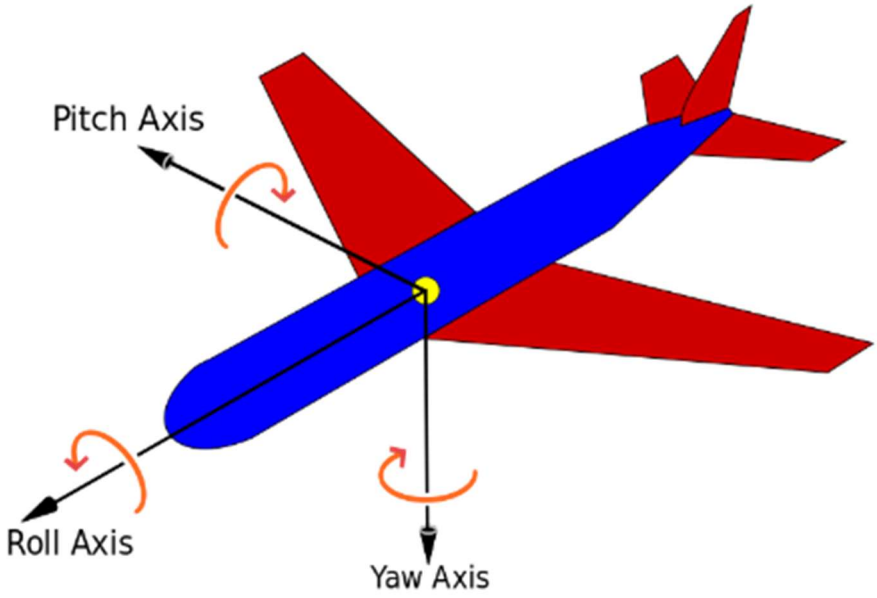

Figure 7: représentation des axes principaux d'un avion, image de [Jrvz] récupérée sur https://commons.wikimedia.org/wiki/File:Yaw_Axis_Corrected.svg

Il existe plusieurs formats de mesures en fonction de la possibilité de mouvement de l'appareil. Par exemple, pour un drone, les mesures sont en degrés et les rotations maximales qu'il peut

effectuer de manière réaliste sont entre -180° et 180° pour le « YAW » et entre -90° et 90° pour le « PITCH » et le « YAW ».

2.5.1. Angles d'Euler

Afin de représenter ces axes de rotation, il est nécessaire de passer par un modèle mathématique qui représente au plus proche la situation réelle. Une étude de Alaimo, Artale, Milazzo, & Ricciardello (2013) propose une comparaison entre la méthode des angles d'Euler et la méthode des quaternions. Un quaternion est une façon de représenter une rotation sous la forme de quatre valeurs et en utilisant la théorie des nombres complexes. Cette représentation est plus courte et lisible qu'une matrice de rotation standard. Il est démontré qu'il est préférable d'utiliser la représentation sous forme de quaternions d'un point de vue informatique, car les calculs sont plus rapides et plus précis (Alaimo et al., 2013). De plus, les quaternions permettent d'éviter l'effet indésirable du "gimball lock" ou blocage de cadran qui peut se produire lorsqu'une rotation d'Euler fait tourner l'un des cadrans de rotation jusqu'à être perpendiculaire à un second cadran (voir Figure 8) (Alaimo et al., 2013).

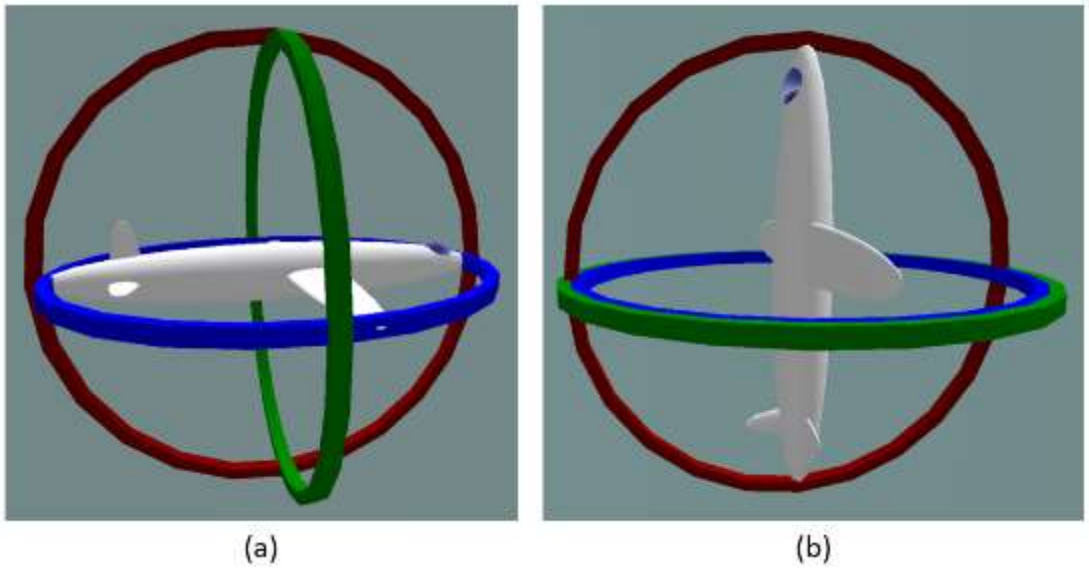

Figure 8: cadrans des axes de rotation principaux. (a) position neutre, (b) "gimbal lock" car deux cadrans sont coplanaires, image de [MathsPoetry] récupérée sur https://commons.wikimedia.org/wiki/File:No_gimbal_lock.png

Nous allons donc utiliser les angles d'Euler pour représenter la direction du drone. Celui-ci peut être tourné autour des trois axes principaux et chacune de ces rotations peut être effectuée mathématiquement à l'aide d'une matrice de rotation (LaValle, 2012). Les matrices présentées sur la Figure 9 correspondent aux matrices de rotation pour chaque axe séparément.

$$R_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad R_y(\beta) = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix}. \quad R_x(\gamma) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{pmatrix}.$$

(a) (b) (c)

Figure 9: matrices de rotation. (a) Yaw, rotation autour de l'axe Z. (b) Pitch, rotation autour de l'axe Y, (c) Roll, rotation autour de l'axe X, image récupérée sur http://planning.cs.uiuc.edu/node102.html

En multipliant les trois matrices, il est possible de former une seule matrice de rotation qui peut être utilisée pour placer un corps en 3D dans n'importe quelle direction à l'aide des angle « roll, pich & yaw » (LaValle, 2012). Celle-ci est représentée à la figure ci-dessous.

$$R(\alpha,\beta,\gamma) = R_z(\alpha)\, R_y(\beta)\, R_x(\gamma) = \begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{pmatrix}.$$

Figure 10: matrice de rotation pour effectuer les trois angles d'Euler en même temps, image récupérée sur http://planning.cs.uiuc.edu/node102.html

La matrice finale de rotation correspond à l'angle de Tait-Bryan « ZYX » qui signifie que ce sont les axes qui tournent et non pas le vecteur (l'objet) et que l'ordre des rotations se fait en premier par rapport à l'axe Z, ensuite par rapport à l'axe Y et finalement par rapport à l'axe X.

Il est impératif d'effectuer les rotations dans l'ordre en commençant par le « yaw », ensuite le « pitch » et finalement le « roll », car, si l'ordre est différent, la matrice de rotation ne sera pas identique, c'est-à-dire que l'orientation finale de l'appareil varie en fonction de l'ordre des rotations appliqué (LaValle, 2012). Cela est dû au fait que chaque rotation sur l'un des axes modifie la position de tous les axes et la rotation suivante se fait sur un plan différent du plan initial.

3. État de l'art – Choix du type des senseurs

Cet état de l'art est divisé en trois parties. La première va traiter des différents types de radar et leurs spécificités afin de déterminer quel est le matériel le plus adapté à un drone agricole. La deuxième s'intéresse aux différentes solutions qui permettent de modéliser et visualiser l'espace autour du drone durant les vols. Dans la dernière partie, nous allons analyser les possibilités d'intégrer du ML sur le « companion computer » en vol afin de pouvoir réagir automatiquement en temps réel au cas où les senseurs détectent un obstacle.

Nous allons tout d'abord établir l'état de l'art des solutions de reconnaissance d'obstacles déjà existantes sur le marché. Nous allons nous concentrer sur les technologies qui permettent de détecter la distance à laquelle se trouve un obstacle par rapport au drone afin d'éviter celui-ci. Les critères de sélection dont nous avons besoin sont les suivants :

- Portée d'au moins dix mètres
- Précision au centimètre près
- Poids et dimensions réduits et facilement intégrable au drone
- La capacité à fonctionner à travers le liquide de traitement pulvérisé
- Possibilité de développement/d'intégration à un système personnalisé
- Gamme de prix accessible

Cette étude va s'orienter sur des techniques de « machine learning » qui utilisent des données récoltées via des senseurs radar ou lidar durant le vol. Les recherches menées par Arbellay (2019) se sont tournées vers des solutions qui utilisent du « machine learning » sur des images récoltées via une caméra embarquée. Ces résultats nous montrent qu'il est possible d'implémenter du ML sur des drones agricoles à l'aide d'une caméra qui envoie les images à un « companion computer » qui, à son tour détermine si un obstacle est sur sa trajectoire (Arbellay, 2019). Néanmoins, cette étude révèle certaines difficultés, notamment un délai élevé entre chaque image reconnue par le « companion computer » et un réseau de neurones « tiny-yolo » moyennement précis et la puissance de calcul du « companion computer » ne permet pas d'obtenir suffisament d'images par seconde (Arbellay, 2019). Il nous semble donc important de se tourner vers d'autres solutions qui peuvent être combinées avec la reconnaissance d'image afin d'améliorer la détection d'obstacles actuelle.

3.1. Radar

3.1.1. Yanwu Tech FMK24-E

Étant donné que les radars à micro-ondes semblent être adaptés pour les drones agricoles, nous allons analyser un modèle de ce type. Nous allons étudier les caractéristiques du FMK24-E qui est un radar à micro-ondes à faible coût idéal pour une utilisation sur les appareils autonomes.

Plusieurs séries différentes du même modèle sont disponibles et varient en fonction des valeurs retournées par le radar. La solution la plus adaptée à notre situation est le modèle E5200 car il est possible de l'utiliser dans des conditions extérieures et il retourne la distance de tous les objets détectés dans son champ de vision en allant du plus près au plus lointain.

Le FMK24-A est conçu pour être intégré à tout type de systèmes et nous pouvons donc l'installer sur le drone et le brancher au « companion computer » afin d'effectuer les opérations dont nous avons besoin.

Le FMK-E5200 est disponible à la livraison pour CHF 70 et mesure 70*60*20 mm pour un poids de 70 grammes. Ces caractéristiques permettent donc une intégration au drone sans trop le déséquilibrer.

Ce senseur utilise une fréquence de 24 GHz pour une distance de détection maximum de 20 mètres. L'angle horizontal de détection est de 78° et l'angle vertical est de 23°. Cela signifie que la détection ne se fait qu'à l'avant du radar et que, pour obtenir une reconnaissance du sol, il faut l'orienter contre le bas en fonction des besoins. Yanwu Tech affirme que la précision est de plus ou moins de 10 cm. Ce radar peut capturer jusqu'à 10 images par secondes (10 Hz).

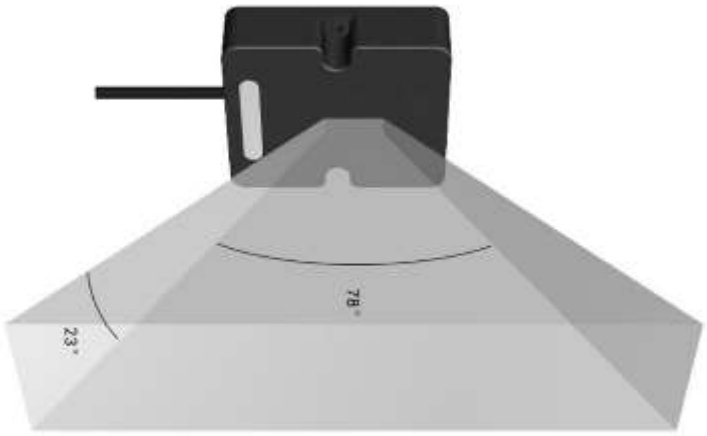

Figure 11: Angle de vision du FMK-25E, image récupérée sur https://rfbros.com/product/fmk24-e5310/

La technologie des radars à micro-ondes est favorable lorsque du liquide est vaporisé entre le senseur et l'obstacle.

Le FMK25-E est donc une solution adaptée à notre situation, car tous les critères testés ont été concluants. Ce radar est économiquement avantageux et offre des mesures de qualité et une portée plus que suffisante pour le but de cette étude.

3.1.2. Acconeer XM112

Acconeer propose des senseurs adaptés à de multiples applications, notamment dans le domaine de l'internet des objets et des drones. Ces senseurs sont spécialement conçus afin que les utilisateurs puissent modifier et ajouter leur propre programme pour satisfaire au mieux leurs besoins. Nous avons choisi de tester le modèle XM112 car celui-ci est adapté à notre cas de figure et nous souhaitons pouvoir modifier le code lié à la détection d'obstacles.

Le XM112 est un radar cohérent pulsé (plused coherent radar) qui combine les avantages d'un radar pulsé et d'un radar cohérent, le premier consomme peu d'énergie et le second offre une meilleure précision (Acconeer AB, 2019). Contrairement à la technologie lidar (laser) qui reconnaît la distance sur un point unique, les radars permettent de détecter la distance sur une zone définie et retourne plusieurs valeurs.

Figure 12: Senseur du radar Acconeer XM112 face avant, image récupérée sur https://www.acconeer.com/products

Acconeer encourage les utilisateurs du senseur à développer eux-mêmes une couche logiciel à intégrer à leur module déjà en place. Ils proposent un kit de développement qui inclut un « software development kit » (SDK) et une interface de programmation (API). Cela facilite la connexion à un « companion computer » et offre une manière rapide d'effectuer un prototypage et une série de tests.

Acconeer met aussi à disposition sur Github un logiciel contenant des outils qui facilitent la lecture d'information des senseurs. Ce programme est sous une licence BSD et peut être donc modifié. Puisqu'il contient des scripts Python utiles à la récolte de données, il est possible de les utiliser et de les adapter en fonction des besoins.

Le senseur XM112 mesure 24x16 mm ce qui représente le senseur le plus compact de notre série de tests. Cette taille est parfaitement adaptée à une situation de traitement agricole et peut être attaché sur un bord de la machine sans prendre trop de place et sans la déséquilibrer.

Le prix par unité du XM112 est de CHF 60 et fait de lui une solution abordable et qui correspond à notre budget.

La portée du XM112 est de 10 mètres et peut détecter plusieurs objets dans sa trajectoire. Le rayon visible par le senseur de 40° permet d'identifier une zone particulièrement large.

Le XM112 est doté du senseur A111 qui utilise une fréquence de 60 GHz sur une bande ISM. Celle-ci permet entre autres de ne pas être perturbé par les conditions météorologiques comme la brume ou la pluie. Cette caractéristique est primordiale dans notre cas de figure lors de traitements.

Nous avons effectué des tests statiques et en vol afin de s'assurer de la qualité des mesures prises par le senseur XM112.

Durant les tests en vol, nous avons observé que les mesures prisent sont cohérentes et que nous pouvons facilement les représenter et les visualiser. Nous avons néanmoins constaté que, dans le cas où l'objet est trop petit, par exemple un fil de fer, le senseur ne permet pas de reconnaître ce type d'obstacle. Plus de détails sur les résultats de ce senseur sont disponibles à la section 7.5.1.

Nous constatons que le radar Acconeer XM112 possède beaucoup d'avantages pour un prix abordable. Le senseur est facilement installé sur le drone et il est compatible avec le « companion computer » Raspberry Pi. Il est possible d'effectuer un développement logiciel afin de garantir une personnalisation en fonction des besoins et de la situation.

La portée du XM112 de maximum 10 mètres est relativement faible. Cela a de l'importance lorsque l'on souhaite pouvoir anticiper les obstacles le plus rapidement possible. Il faut donc être attentif à la vitesse du drone et du temps de réaction entre la détection par le radar et la commande d'évitement donnée par l'ordinateur de bord.

3.2. Lidar

3.2.1. TFmini Plus

Contrairement aux deux senseurs précédents, Le TFmini Plus est un lidar. Cela signifie que les ondes radars sont remplacées par un laser et la distance est mesurée en fonction de l'intensité de lumière retournée après la réflexion sur l'objet visé (Simon, Lisa, & Bryan, 2014).

Nous allons comparer le lidar Tfmini Plus avec les autres senseurs déjà analysés afin de s'assurer que ce senseur peut être utilisé dans une situation de traitement agricole.

Le Tfmini peut être facilement connecté au « companion computer », notamment via le protocole « Universal Asynchronous Receiver/Transmitter » (UART) ou « Inter-Integrated Circuit » (IIC) qui sont des bus standards utilisés pour la communication des données entre machines. Dans tous les cas, bien que le hardware soit propriétaire, il est possible de récupérer les données du senseur afin de procéder à des opérations business tel que la représentation de l'environnement ou du « machine learning ». Les utilisateurs du Tfmini disposent donc d'une grande marche de manœuvre avec les données récupérées.

La taille du Tfmini plus est de 35 x 21 x 18.5 mm pour un poids total de 11 grammes. Il s'intègre facilement sur un drone sans y ajouter trop de charges et sans déséquilibre.

Le prix par unité du Tfmini est de CHF 35. Il est donc financièrement possible d'installer plusieurs senseurs de ce modèle afin d'obtenir un plus grand nombre de mesures simultanément.

Ce lidar est capable de détecter un obstacle sur une distance de 12 mètres et avec une marge d'erreur de plus ou moins 5 cm. Le nombre de captures par secondes est ajustable et peut monter jusqu'à 1000 Hz (1000 rafraîchissements par secondes).

Parmi les senseurs testés, le lidar est le senseur le plus susceptible d'obtenir des résultats incorrects lors d'un traitement à cause du liquide pulvérisé. Les mesures de distance à travers le spray de traitement sont potentiellement altérées dans le cas ou le laser est reflété sur le liquide.

Nous avons testé le Tfmini en effectuant plusieurs vols d'essais sur des vignes. Nous avons constaté que le lien entre les mesures d'altitude du GPS et la distance mesurée par le lidar est cohérent. Le graphique ci-dessous représente les mesures des lidars et l'altitude en centimètres en fonction du temps. Étant donné que les lidars ne pointent pas le sol de manière perpendiculaire mais en diagonale en direction de l'avant, la distance mesurée par les lidars est légèrement supérieure à l'altitude GPS. Le terrain sur lequel les tests ont été effectués n'est pas exactement plat et peut expliquer les quelques irrégularités dans les mesures.

Figure 13: photo du lieu où les mesures de vols ont été prises, image de Dominique Genoud

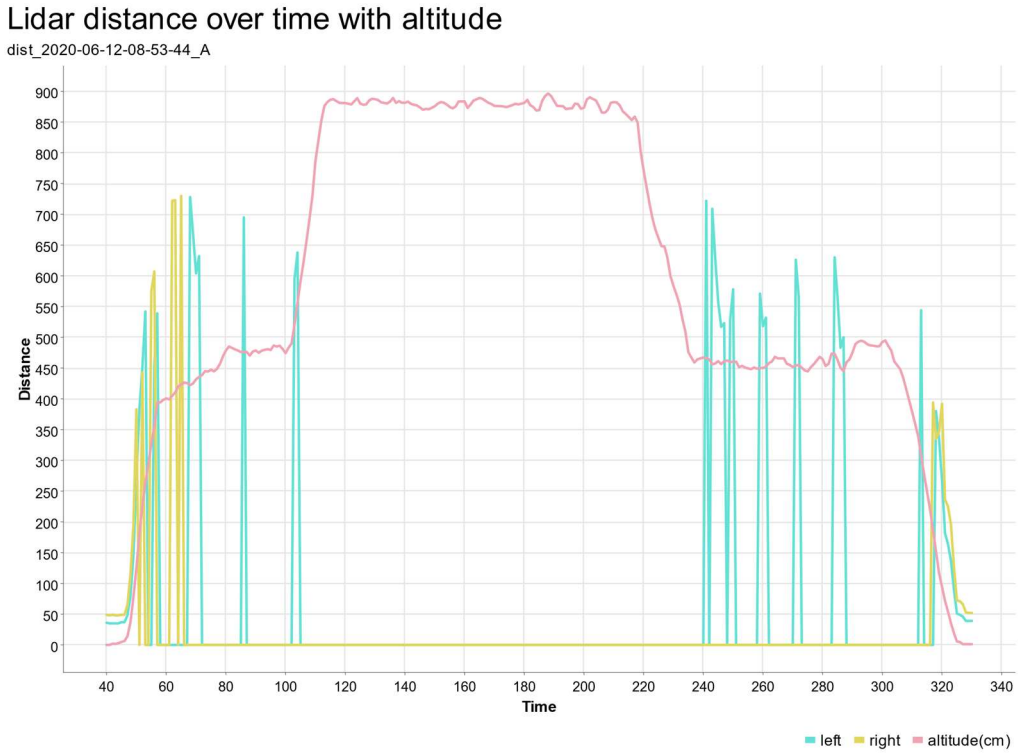

Figure 14: altitude est distance lidar [cm] en fonction du temps (intervalle de 0.5 sec.) lors d'un vol d'essai, image de l'auteur

Nous avons remarqué que le Tfmini est une solution avantageuse financièrement et qui offre une précision adaptée pour des vols de drone à basse altitude. De plus, il est possible d'installer plusieurs Tfmini sur le UAV afin de couvrir une zone plus importante en fonction des besoins du vol. Les données retournées par ce lidar sont simples et cohérentes et peuvent être analysées par l'ordinateur de bord en temps réel. Cela fait du Tfmini un bon candidat pour effectuer du « machine learning » lors d'un vol.

Il est important de préciser que le Tfmini ne retourne qu'une seule mesure de distance et ne permet de viser qu'à un seul endroit par lidar simultanément. Cela signifie qu'une petite différence dans l'inclinaison du Tfmini change drastiquement les résultats obtenus. Il est donc possible qu'un obstacle ne soit pas reconnu dans le cas où la trajectoire du laser passe à côté de celui-ci. Un autre cas de figure problématique est le fait qu'un objet peut se retrouver sur la trajectoire du laser entre deux mesures. Il est donc important de diminuer au mieux l'intervalle entre chaque mesure.

Pour pallier ce problème, il existe des lidars rotatifs qui permettent une détection sur une zone en 2D ou en 3D, mais à un coût plus élevé.

3.3. Drone

3.3.1. DJI AGRAS M1G1S Obstacle Avoidance Radar

DJI propose des drones agricoles qui permettent de traiter des champs de différentes natures. Ils proposent une solution qui permet une automatisation du parcours et une détection d'obstacle en temps réel en utilisant des senseurs radars à micro-ondes placés à l'avant, à l'arrière et sous la machine.

DJI étant une marque propriétaire de son système, il est impossible de trouver les documents techniques sur les technologies qu'ils utilisent. En consultant le guide d'utilisation de leur radar (DJI S. , 2016), nous pouvons néanmoins noter la position des senseurs et la direction de ceux-ci.

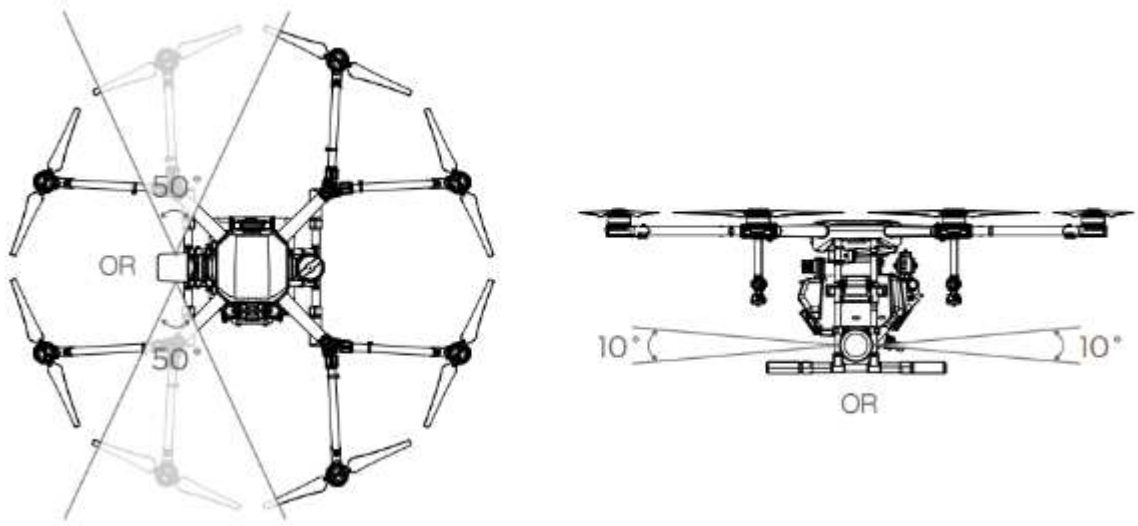

Figure 15: Direction et angle de vue du radar du DJI Agras MG-1S, récupérée sur https://dl.djicdn.com/downloads/mg_1s/20170717/Obstacle_Avoidance_Radar_User_Guide_v1.0_Multi.pdf

Nous constatons que le radar principal peut observer à l'avant ou à l'arrière (en fonction du sens de la marche) sur un angle de 50° à l'horizontale et 10° à la verticale. De plus, les radars à micro-ondes présentent des résultats pertinents lors de vols dans de mauvaises conditions atmosphériques et donc lors d'un traitement agricole. Puisque le hardware est propriétaire, il n'est pas possible de développer son propre système et de le modifier en fonction de nos besoins.

Le senseur proposé par DJI est particulièrement puissant et précis. Cela nécessite un hardware plus conséquent. Son poids est d'environ 406gr pour une consommation de 12 kW Sa taille est de 109x152mm et se trouve sur le côté de la machine comme indiqué sur la Figure 15. Il est le plus grand et le plus volumineux de notre liste.

Malgré tous les avantages du senseur DJI, il est le plus coûteux de notre étude. Son prix trop conséquent n'entre pas dans un budget limité et nous devons donc nous restreindre à des solutions plus accessibles.

Ce radar permet de mesurer la distance des obstacles avec une précision de 10 cm et permet une capture de l'environnement à une fréquence élevée d'au moins 24 GHz. DJI affirme (2018) pouvoir détecter des obstacles aussi petits que des lignes à haute tension jusqu'à 30 mètres de distance.

Le senseur DJI est un radar à micro-ondes. Les études à ce sujet ont montré (Zhengyu & Changzhi, 2019) que cette technologie permet une exploitation de la détection d'obstacle par toutes conditions atmosphériques notamment de nuit.

Cet élément est décisif pour une situation de traitement agricole, car nous devons pouvoir détecter la distance au sol lorsque le liquide diffusé par les pompes se situe sur la trajectoire du senseur. Nous constatons que la technologie radar à micro-ondes est adaptée à notre cas de figure.

Nous avons analysé le radar de détection d'obstacles proposé par DJI et nous avons constaté qu'il s'agit d'une solution performante est adaptée à un traitement agricole avec un drone. La précision offerte par ce senseur est plus que suffisante dans notre situation. La technologie radar à micro-ondes semble être idéale pour résoudre le problème du spray entre le senseur et l'obstacle.

DJI est propriétaire de son matériel et de ses logiciels. Nous ne pouvons pas effectuer de modifications techniques sur cet appareil. Puisque nous recherchons une solution permettant le développement et l'adaptation, nous ne pouvons pas choisir ce senseur. De plus, le prix, le poids et la taille n'entrent pas dans les caractéristiques recherchées.

4. État de l'art – Représentation de l'environnement

Durant cette étude, nous souhaitons représenter l'environnement autour du drone afin de détecter les obstacles au plus vite et de créer une zone de sécurité à proximité de la machine. Le but, dans un premier temps, est de récupérer les données GPS, les mesures des axes principaux de l'appareil (roll, pitch, yaw) ainsi que les mesures prises par les senseurs (radar et lidar). Nous allons ensuite utiliser ces données en les intégrant dans un logiciel de « mapping » afin de les visualiser de manière précise dans une simulation qui représente au mieux le terrain et l'environnement réel.

Nous avons mis en place plusieurs critères précis que nous allons évaluer sur chaque programme testé afin de cerner la meilleure solution à notre disposition. Les logiciels vont être examinés sur les points suivants :

- Facilité d'installation et d'utilisation
- Logiciel open source / propriétaire
- Logiciel Multiplateforme
- Qualité des résultats obtenus
- Intégration des différents formats de données GPS

Les trois logiciels sélectionnés pour cette étude font partie de trois catégories différentes de produits. Nous avons sélectionné Google Earth Pro comme logiciel propriétaire gratuit, QGIS comme logiciel « open-source » gratuit, ArcGIS en tant que logiciel propriétaire payant et Earth Enterprise comme logiciel libre et open source.

4.1. Google Earth Pro

La solution la plus populaire auprès du grand publique dans le milieu des SIG est Google Earth Pro étant donné que le logiciel est gratuit, intuitif et facile d'installation. Ce programme dispose d'une large palette de fonctionnalités pour visualiser le terrain en 3D ainsi que des outils techniques de visualisation de données géographiques telles que des itinéraires, des polylignes et des polygones. Il est aussi possible d'importer et exporter des données via des formats classiques tels que des fichiers csv ou kml.

Google Earth Pro est disponible sur le site internet de Google[1]. Nous avons testé l'installation sur Windows 10 et nous avons constaté que celle-ci est rapide et intuitive. Nous n'avons pas rencontré d'erreurs lors de cette étape et lors du premier lancement.

L'interface de ce programme est facile à comprendre et à prendre en main. Celle-ci est séparée en deux parties principales : la carte en 3D et le volet qui contient les données géographiques de l'utilisateur ainsi que des options pour modifier les couches de la carte.

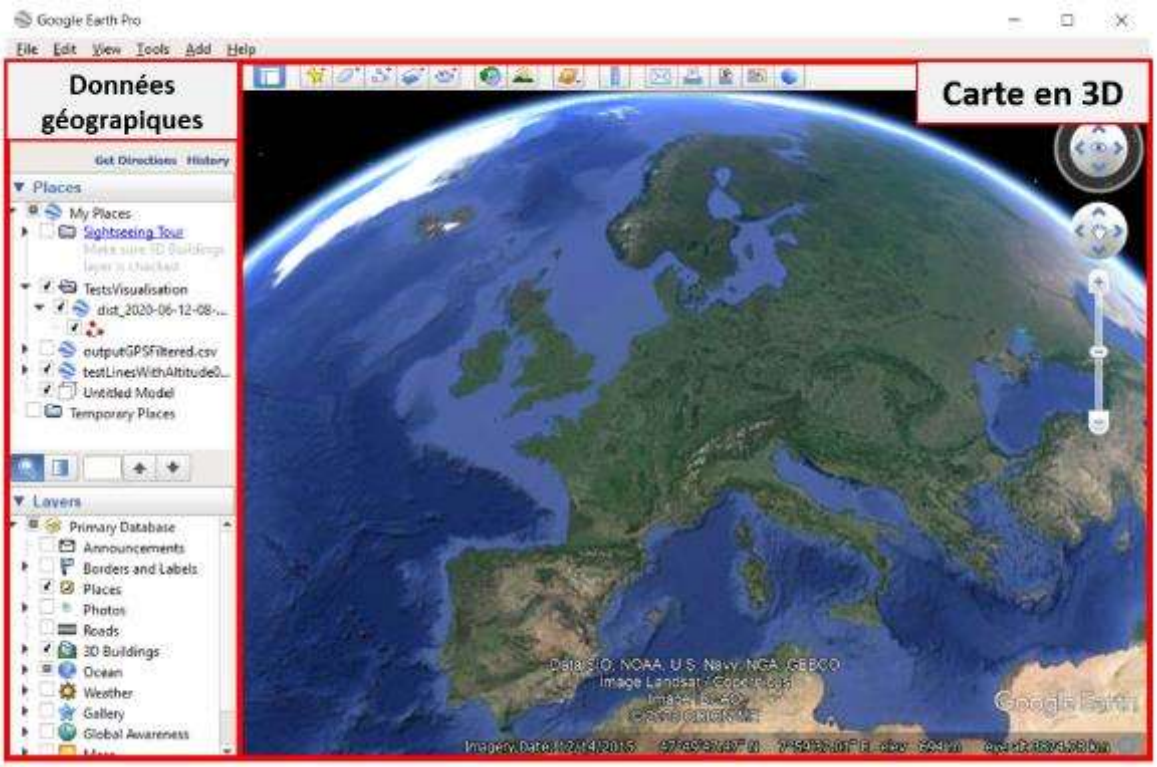

Figure 16: interface utilisateur de Google Earth Pro et ses composants, image de l'auteur

[1] https://www.google.com/earth/versions/#earth-pro

Nous nous sommes particulièrement intéressés à la partie sur les données géographiques, car notre but est de pouvoir importer et visualiser les données récoltées durant les vols du drone. Nous avons constaté qu'il est possible de créer une hiérarchie de fichiers afin de mieux organiser les résultats importés, ce qui facilite la réutilisation et la lecture de ceux-ci dans le temps.

La navigation sur la carte peut se faire facilement à l'aide de la souris et ne nécessite pas de raccourci ou de combinaison de touches complexes. Il est possible de zoomer jusqu'au niveau du sol et d'orienter librement la caméra dans l'espace. La fonction Google Street View permet parfois d'avoir une représentation identique à la réalité grâce à des images sphériques.

Google Earth Pro est un logiciel gratuit depuis 2015. Il reste néanmoins un logiciel propriétaire développé par Google et le code source n'est pas disponible. Il est important de noter que, dans le cas de figure de cette étude, notre objectif avec un SIG est de visualiser rapidement les données et nous ne souhaitons pas développer nous-même une solution, pour autant que nous puissions remplir nos objectifs de visualisation.

Google earth peut être installé et utilisé sur les systèmes Windows, Mac et Linux

Nous avons facilement pu ajouter nos fichiers kml contenant des polylignes et des points GPS. La fonction d'import permet aux utilisateurs d'ajouter des données via plus de trente formats de fichiers différents. Nous avons testé l'importation avec un fichier csv contenant plusieurs coordonnées de points et nous avons obtenu le résultat attendu à l'aide de l'assistant d'import intuitif.

4.1.1. Qualité des résultats

Afin d'analyser la qualité des résultats obtenus sur Google Earth Pro, nous avons transformé les données GPS bruts d'un vol d'essai dans un format kml. Ce fichier contient les coordonnées d'une polyligne qui représente le parcours du drone dans l'espace.

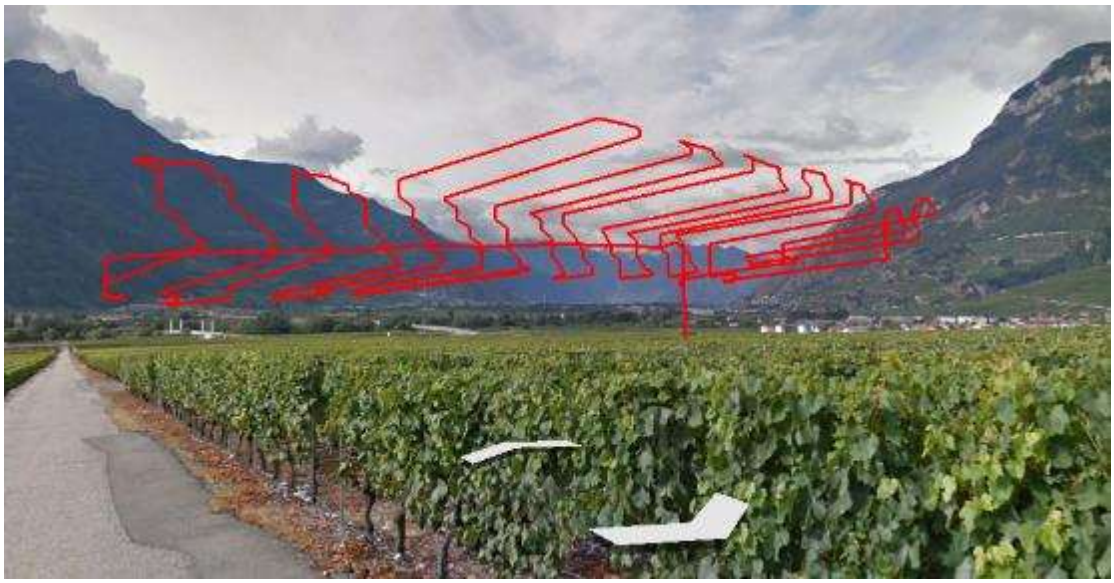

Figure 17: représentation d'un vol de test en terrain agricole sur Google Earth Pro, image de l'auteur

L'image ci-dessus nous montre que les résultats sont précis et que nous sommes capables de visualiser les données voulues. Lorsque la fonction Google Street View est disponible, il est possible de représenter les données géographiques en se rapprochant de la réalité (voir figure ci-dessus). Ces résultats sont concluants et permettent d'atteindre notre objectif de visualisation dans l'espace.

4.1.2. Avantages et inconvénients

Nous avons constaté que Google Earth Pro est une solution gratuite et facile d'utilisation. Les résultats obtenus de visualisation 3D sont plus que suffisants et de très haute qualité surtout lorsque Google Street View est disponible. Il est possible de représenter des polylignes et des points via le format kml qui est intuitif et bien documenté sur Internet. Bien que le programme ne soit pas open source, nous sommes en mesure d'accomplir les tâches de visualisation nécessaires pour atteindre les objectifs de cette étude.

4.2. Earth Enterprise

Earth Enterprise est la version open source de Google Earth Enterprise dont le code source est disponible librement sur Github (Google, 2020). Ce programme permet de créer et d'héberger des serveurs contenant des cartes 2D ou 3D en utilisant une combinaison d'imagerie pour les cartes, du terrain pour les reliefs et des vecteurs pour les données géospatiales (Google, 2020). Ainsi, il est possible de déployer des cartes personnalisées en ligne et de les rendre accessibles pour tous les acteurs concernés.

Ce programme est séparé en plusieurs modules qui remplissent différentes missions :

- Fusion, pour créer des cartes en 2D ou 3D
- Server, pour héberger les cartes créées avec fusion, basé sur Apache
- Earth Enterprise Client, un client similaire à Google Earth Pro qui peut se connecter à un serveur personnalisé

Durant l'installation de Earth Enterprise Fusion, nous nous sommes aperçus que plusieurs points nous ont posés des difficultés. Premièrement, certaines parties de la documentation ne sont pas claires et d'autres se contredisent, en particulier concernant l'installation. Par exemple, afin d'utiliser ce programme, il est nécessaire de créer un « build » pour ensuite obtenir un package d'installation. Aucun « build » en production facile d'utilisation et prêt à l'emploi n'est disponible pour le moment.

Il est crucial de noter que la liste des systèmes d'exploitation compatible avec Earth Enterprise Fusion et Server est limitée et nous avons choisi d'utiliser Ubuntu 16.04 LTS après avoir testé l'installation sur Ubuntu 20.04 LTS sans succès.

La création du fichier d'installation peut prendre un temps considérable (une heure avec 8GB de mémoire vive et deux processeurs) et nous avons rencontré plusieurs erreurs dues à de mauvaises configurations des dépendances et des particularités propres aux systèmes Linux sur lesquelles la documentation n'était pas suffisante.

Le programme Earth Enterprise Client qui permet d'accéder aux cartes créées via Fusion dispose de la même interface utilisateur que Google Earth Pro (voir Figure 16 et Figure 18). Ce client peut être téléchargé sur le Github de Earth Enterprise[2] et peut s'installer de manière classique et rapide.

Nous nous sommes rendu compte rapidement durant le tutoriel d'introduction mis à disposition par les contributeurs du projet que l'utilisation de Earth Enterprise Fusion et Server nécessite une charge de travail importante et des connaissances techniques avancées sur les SIG. L'utilisation de ce programme n'est donc pas adaptée pour une utilisation intuitive et rapide à des fins de visualisation uniquement.

Earth Enterprise est distribué sous la licence Apache 2.0. Il est donc possible d'utiliser et de modifier le code source du programme pour des utilisations commerciales (The Apache Software Foundation, 2004).

[2] https://github.com/google/earthenterprise/wiki/Google-Earth-Enterprise-Client-(EC)

Earth Entreprise Fusion et Server sont disponibles uniquement sur les systèmes d'exploitation suivants : CentOS 6 et 7, Red Hat Enterprise Linux 6 et 7, Ubuntu 16.04 LTS.

Le client Earth Enterprise Client est disponible sur tous les systèmes Windows, Mac et Linux.

Il est possible d'importer des fichiers au format kml via Fusion ou directement via le client et, comme pour Google Earth Pro, la fonction pour convertir un fichier csv contenant des coordonnées GPS est disponible sur le client.

4.2.1. Qualité des résultats obtenus

Nous avons testé Earth Enterprise Client sur le serveur officiel de Google Earth étant donné que nous n'avons pas configuré notre propre serveur. Les résultats sont identiques à Google Earth Pro. Ils sont donc précis et remplissent nos objectifs de visualisation.

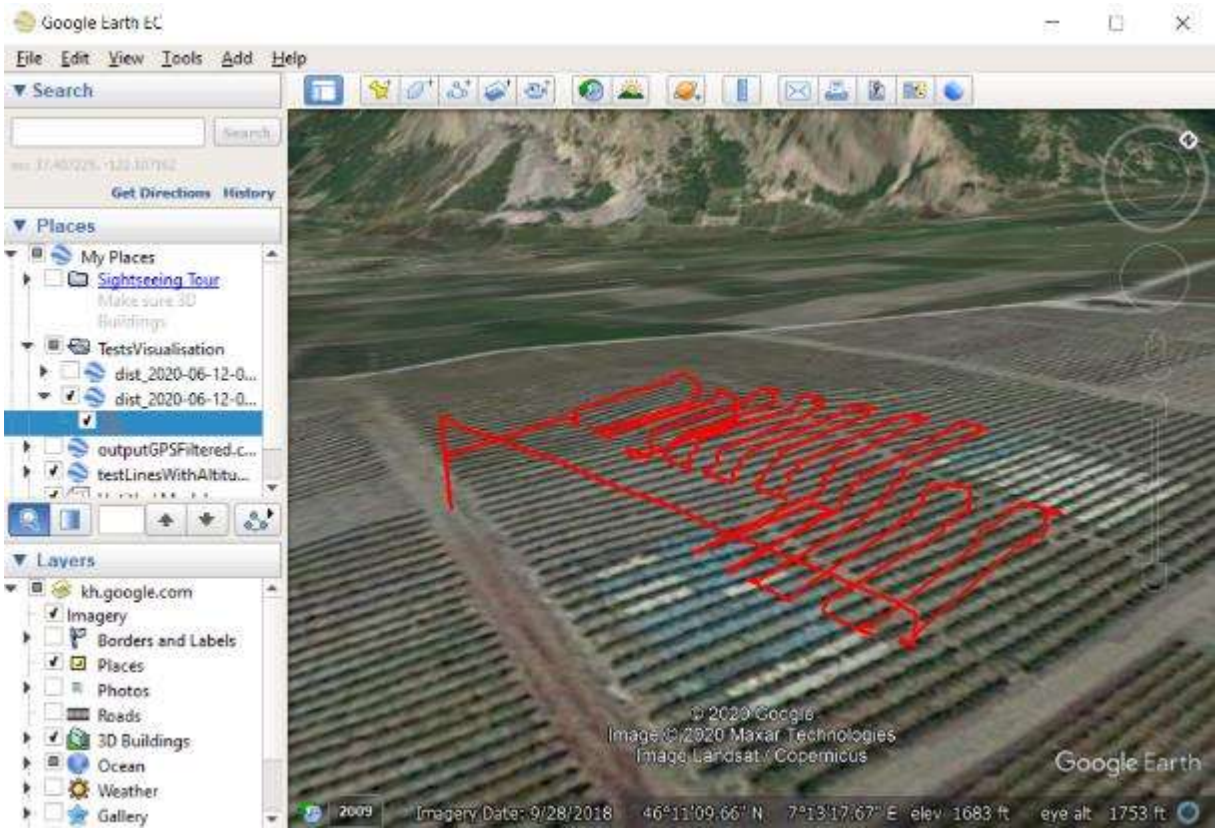

Figure 18: résultat du parcours du vol d'essai sur Earth Enterprise Client, image de l'auteur

4.2.2. Avantages et inconvénients

Nous avons constaté que Earth Enterprise nécessite un investissement en temps et en ressources considérable si nous souhaitons mettre en place toute l'architecture, y compris la partie Fusion et Server. Nous avons obtenu des résultats concluants et similaires à ceux de Google Earth Pro. Nous

avons néanmoins constaté que la version open source de ce produit est potentiellement moins optimisée, car nous avons eu des ralentissements plus importants lors de l'utilisation de celui-ci.

Cette solution est donc potentiellement avantageuse dans la mesure où nous souhaitons créer et mettre à disposition sur Internet un serveur dédié consacré à la visualisation en 3D. Pour cela, il est nécessaire d'obtenir ses propres ressources en ce qui concerne l'imagerie, le terrain (les reliefs) ainsi que les données vectorielles (kml et autres) que nous souhaitons représenter. Durant nos recherches, nous avons constaté que les données qui concernent le terrain sont en majorité payantes et ne sont pas obligatoirement compatibles avec le programme concerné. Cela signifie que l'intégration de ces données peut demander du temps et des compétences substantiels.

4.3. QGIS

QGIS est un programme SIG open source qui permet à ses utilisateurs de visualiser des données géographiques en 2D et 3D à l'aide d'une interface utilisateur intuitive et des outils facilement pris en main (QGIS project, 2020).

Nous avons sélectionné ce programme car nous souhaitons mettre en avant les solutions open source dans la mesure du possible. QGIS s'est démarqué dans notre sélection car celui-ci a été développé par une communauté qui, au moment de la rédaction de ce document, est toujours active et continue de mettre à jour le programme. De plus, le projet est fortement documenté et les utilisateurs ont créé et mis à disposition une grande quantité de ressources (articles et vidéos) qui facilitent l'utilisation d'outils spécifiques.

L'installation de QGIS se fait à l'aide d'un assistant d'installation standard. Elle est simple et ne demande aucune action spécifique de la part de l'utilisateur.

L'interface utilisateur ressemble à celle de Google Earth Pro. La partie principale est vide lors de la création d'un nouveau projet et l'utilisateur peut ajouter ses propres cartes. Dans le cas où l'on ne dispose pas de carte, celle de OpenStreetMap est disponible par défaut via l'onglet Browser -> XYZ Tiles -> OpenStreetMap. L'ajout de composants et d'éléments à la carte peut se faire simplement via le glisser-déposer.

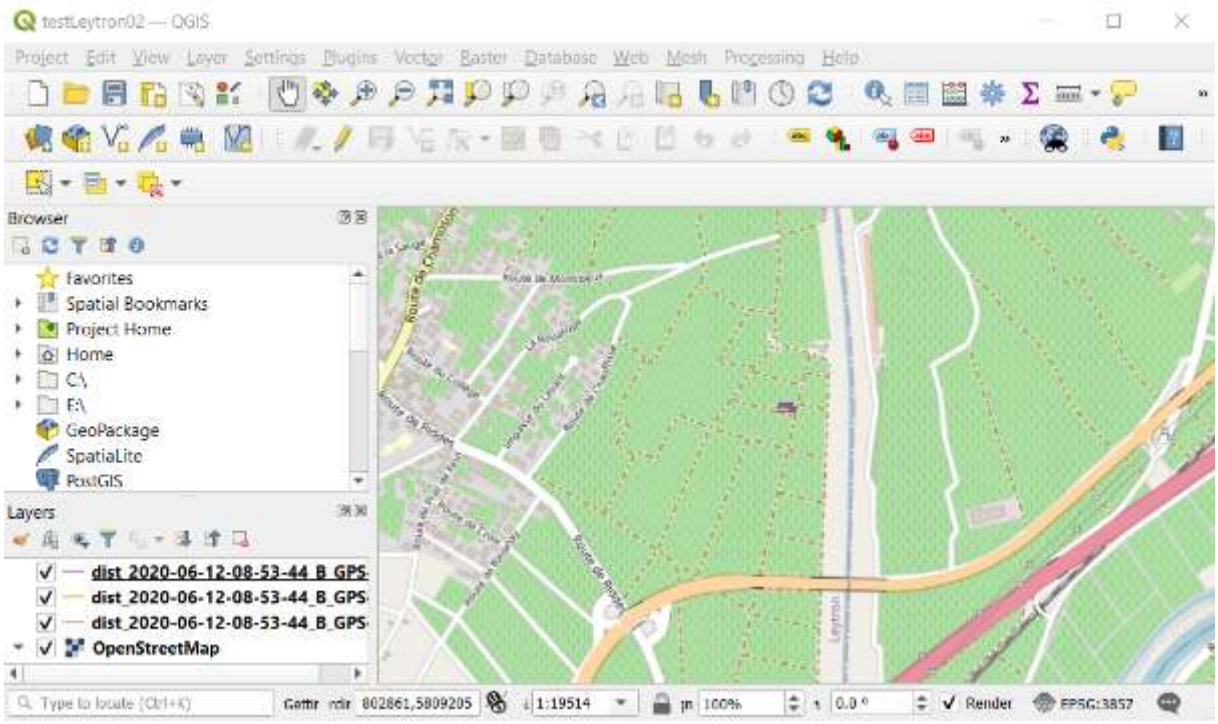

Figure 19: interface utilisateur principale de QGIS avec la couche OpenStreetMap, image de l'auteur

L'écran principal permet une prévisualisation en 2D uniquement. Pour accéder à la vue 3D, il faut créer et ouvrir une vue en 3D via View -> New 3D Map View. Par défaut, la carte est plate et il est nécessaire d'ajouter un fichier DEM qui contient les vecteurs de relief. La procédure pour visualiser des données géographiques dans l'espace sur une carte demande beaucoup de temps.

QGIS est open source avec la licence GNU GPL v2.0. Cela signifie qu'il est possible d'utiliser, modifier et distribuer ce programme à des fins privées et commerciales à condition de garder la même licence (Free Software Foundation, 1991).

QGIS est disponible sur Windows, Mac et Linux. Il est aussi possible de le télécharger sur BSD et trois applications mobiles sont disponibles afin de garder QGIS avec soi sur le terrain.

Durant nos tests, nous avons été capables d'importer des fichiers kml contenant des lignes et des points ainsi que des fichiers csv avec des données GPS.

4.3.1. Qualité des résultats

Les résultats obtenus lors de nos tests n'ont pas été concluants car nous n'avons pas obtenu une visualisation 3D de nos données malgré une série de recherches et de tests. En effet, il est impossible d'obtenir un rendu avec des perspectives si l'on ne possède pas de fichier « digital elevation model » (DEM) qui contient les vecteurs, les points et le relief nécessaires.

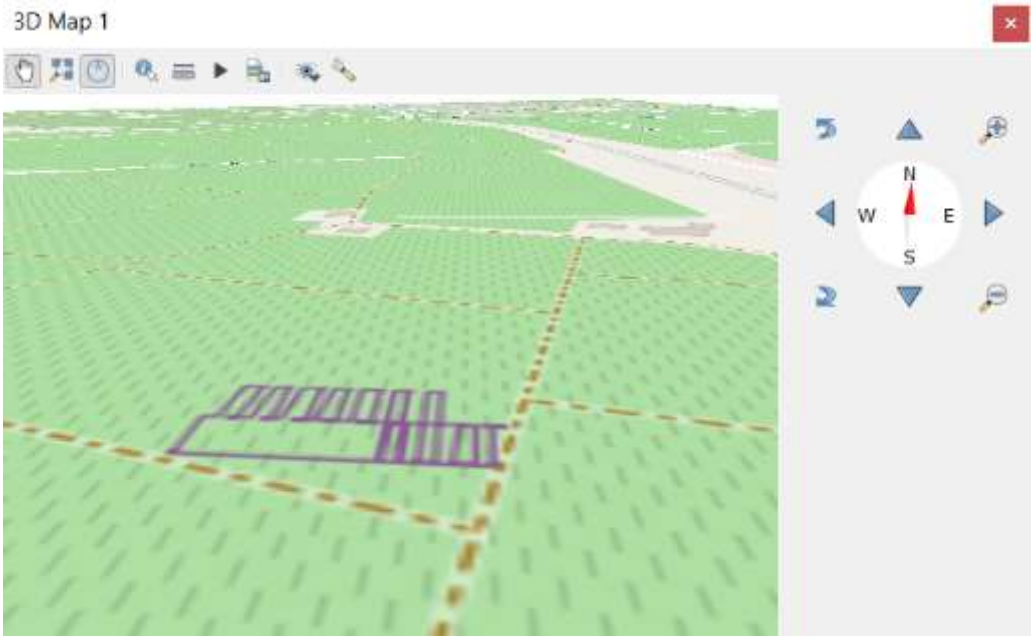

Figure 20: résultat de la visualisation sur QGIS sans relief, image de l'auteur

4.3.2. Avantages et inconvénients

Nous constatons que QGIS est un programme gratuit et open source avec un haut potentiel pour la création de cartes en 2D et 3D. Néanmoins, il n'est pas possible d'obtenir une visualisation rapide des données de vols dont nous disposons. Nous estimons que ce programme ne permet pas de remplir nos objectifs de manière efficace.

Il est intéressant de relever que ce programme est une référence dans le milieu des SIG open source et qu'il est toujours utilisé et mis à jour par une grande communauté. Il est facile de trouver de la documentation et de l'aide en ligne et la librairie de plugins comporte de nombreux outils qui offrent un vaste choix d'actions à l'utilisateur.

4.4. ArcGIS

ArcGIS est un logiciel SIG professionnel développé par Esri qui permet l'analyse et le « mapping » de données géographiques en 2D et 3D (ArcGIS, 2020). En plus de proposer une interface utilisateur intuitive, ArcGIS propose plusieurs programmes adaptés à différentes situations techniques. Ceux-ci sont inter-connectés et offrent la possibilité de partager et collaborer facilement sur des projets au sein d'une organisation. Esri propose une version d'essai sur la suite ArcGIS Desktop qui contient les logiciels principaux ArcGIS. Il est ensuite possible d'obtenir une licence adaptive en fonction des besoins, de la taille et du nombre d'utilisateurs de l'entreprise.

Nous avons sélectionné ce programme dans le but de comparer les programmes gratuits à un logiciel professionnel. Notre but final est de s'orienter vers une solution gratuite, mais nous souhaitons avoir une idée générale de ce que peux offrir un programme propriétaire et déterminer s'il est nécessaire d'utiliser un logiciel payant.

Nous allons tester en particulier la partie ArcGIS Pro qui permet une visualisation 3D sur une carte avec reliefs sous la même forme que Google Earth Pro.

Pour obtenir et installer ArcGIS Pro, il faut créer un compte et/ou une organisation sur le site internet de Esri[3]. Il est ensuite nécessaire d'utiliser leur interface utilisateur en ligne afin d'ajouter votre compte à la liste des utilisateurs autorisés pour ArcGIS Pro. L'installation se fait à l'aide d'un assistant classique et rapide.

L'écran d'accueil est intuitif et permet d'ouvrir les projets existants ou de créer une nouvelle carte. Lorsque l'on sélectionne « New » -> « Global Scene », une nouvelle fenêtre apparaît avec le projet vide. L'écran principal est divisé en plusieurs parties globalement similaires à Google Earth Pro. La carte ainsi que le terrain sont accessibles par défaut et l'utilisateur peut librement modifier son angle de vue. Les données à visualiser peuvent être importées et sont visibles sur l'onglet « Content » à l'aide d'un simple glisser-déposer.

[3] https://www.esri.com/en-us/arcgis/products/esri-redistricting/trial

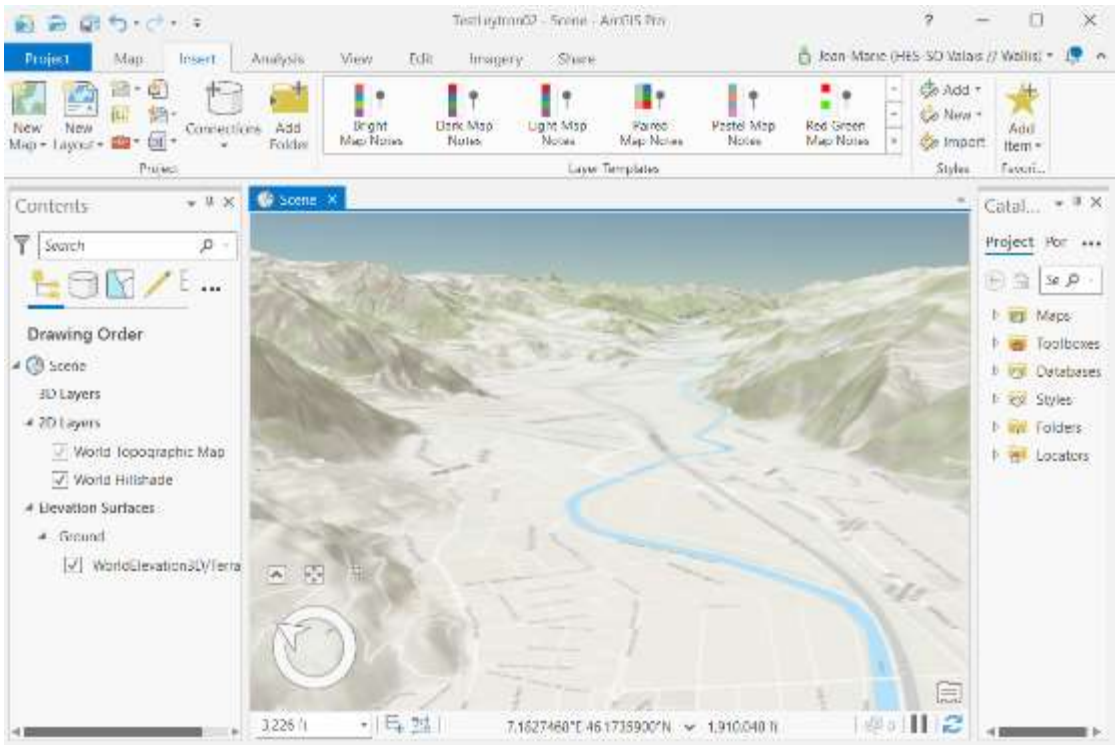

Figure 21: interface utilisateur de ArcGIS Pro avec terrain en 3D, image de l'auteur

ArcGIS est développé de manière propriétaire par Esri et ne peut pas être modifié librement. Le prix est disponible sur demande et varie selon le nombre d'utilisateurs, au minimum CHF 1'500.

ArcGIS a été conçu pour fonctionner sur Windows mais il est possible de l'utiliser sur d'autres systèmes d'exploitation en passant par un « dual boot » ou une machine virtuelle.

Il est possible d'intégrer la plupart des types de fichier dans ArgGIS. Une section du logiciel nommée « Toolbox » offre un vaste choix d'outils permettant l'intégration et la transformation des données géographiques. Nous avons été capables d'importer des fichiers kml ainsi que des coordonnées XY contenues dans un fichier csv.

4.4.1. Qualité des résultats

Les résultats obtenus sont concluants et nous avons pu observer rapidement les données désirées. Il est facile de trouver les informations nécessaires dans la documentation et les outils à disposition sont complets et intuitifs. De plus, le programme dispose d'une bulle d'aide pour chaque champ à remplir lors de phases de configuration et d'importation.

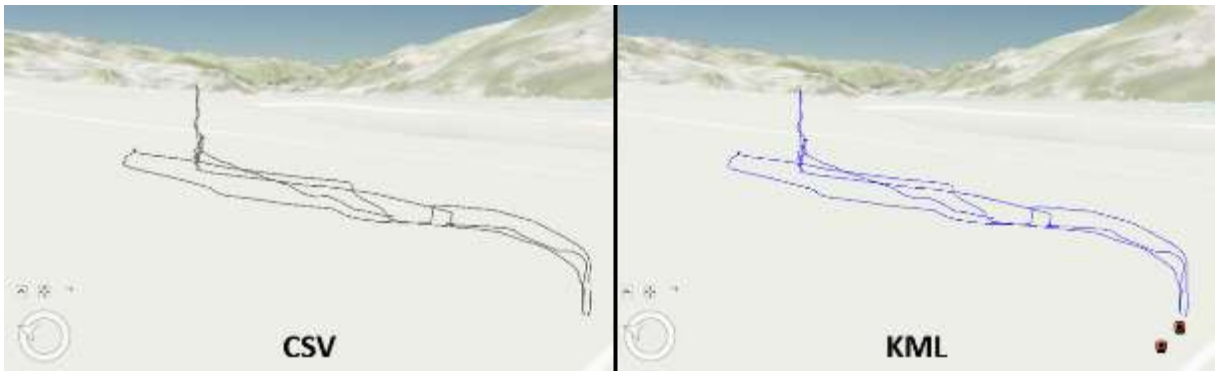

Figure 22: résultat de la visualisation sur ArcGIS pro, image de l'auteur

4.4.2. Avantages et inconvénients

ArcGIS est un logiciel performant et précis pour la représentation de données géographiques. Cette solution offre la possibilité d'importer rapidement et facilement les types de fichiers les plus utilisés par les SIG. L'interface utilisateur est intuitive et permet de gagner beaucoup de temps durant les opérations.

ArcGIS est le seul logiciel payant de la série et demande un investissement financier important. Nous souhaitons nous focaliser sur les solutions moins coûteuses dans la mesure du possible et nous avons vu qu'il est possible d'atteindre nos objectifs avec des programmes gratuits. Il faut aussi préciser qu'ArcGIS est un logiciel lourd (4GB) qui demande des ressources « hardware » élevées et qui fonctionne uniquement sur un système Windows.

5. État de l'art – Détection d'obstacles en temps réel

Nous allons maintenant décrire différents systèmes et les technologies déjà présentes sur le marché qui permettent une détection d'obstacles en temps réel afin d'effectuer des manœuvres d'évitement avant un accident. Nous avons décidé de sélectionner des solutions qui utilisent la technologie radar ou lidar. Ces senseurs permettent d'obtenir des données qui décrivent l'environnement dans un périmètre donné afin de les traiter et de déterminer si un obstacle est présent ou non.

Le but de cette section est d'avoir une vision globale des procédés existants afin de mieux orienter nos recherches sur ce sujet. Nous souhaitons nous inspirer et non pas effectuer une comparaison de ces différents produits ou techniques.

5.1. Recherches technologiques

Nous allons tout d'abord dresser l'état de l'art des études effectuées sur la reconnaissance d'obstacles à l'aide d'une méthode d'apprentissage automatique. Nous allons nous focaliser sur les

techniques qui utilisent des senseurs radar ou lidar intégrés à un drone dans le but de détecter et d'éviter les objets à basse altitude sur une courte portée.

Une étude affirme que les différents composants considérés comme étant communément utilisés pour la reconnaissance d'obstacles sont un GPS, un système IMU (gyroscope) pour détecter les mouvements du drone sur les axes de rotation, une caméra ainsi que des senseurs laser (Fraga-Lamas, Ramos, Mondéjar-Guerra, & Fernández-Caramés, 2019). De plus, Wang et al. (2019) affirment que, durant l'exploitation d'un terrain agricole avec un drone, plusieurs facteurs comme les gouttelettes du traitement, la lumière et le vent peuvent altérer les mesures des senseurs. Il convient donc de prendre ces effets en compte lors de l'implémentation d'une reconnaissance d'obstacles. Ils affirment aussi qu'il est primordial d'identifier plusieurs paramètres qui décrivent le terrain d'opération, notamment la distance minimale de reconnaissance, la façon dont le drone est piloté et le temps de réponse de l'appareil (Wang, et al., 2019).

Des tests de détection d'obstacles ont aussi été effectués avec des senseurs à faible coût et ils ont révélé qu'il n'était pas nécessaire d'utiliser des senseurs coûteux et performants pour obtenir des résultats convaincants (Gageik, Benz, & Montenegro, 2015). Néanmoins, de nombreuses publications, notamment celle de Xie, Xu et Wang (2019) et de Asvadi, Premebida, Peixoto et Nunes (2016) estiment qu'il est nécessaire d'utiliser la technologie lidar en trois dimensions afin d'obtenir de meilleurs résultats car cela permet d'obtenir plus de données et de couvrir une surface plus conséquente qu'avec un simple laser à une dimension.

D'après les caractéristiques décrites ci-dessus, un système de classification à deux niveaux est proposé. La première classification porte sur la taille de l'objet détecté (micro, moyen-petit, large, non-fixé) et la deuxième se concentre sur la distance mesurée jusqu'à l'objet (courte, moyenne et longue) (Wang, et al., 2019). Il n'y est pas mentionné de distance spécifique pour chaque classe de distance et il est donc nécessaire de prendre en compte la vitesse du véhicule ainsi que le temps de réaction du contrôleur de vol lorsqu'un signal d'alarme est donné par le "companion computer".

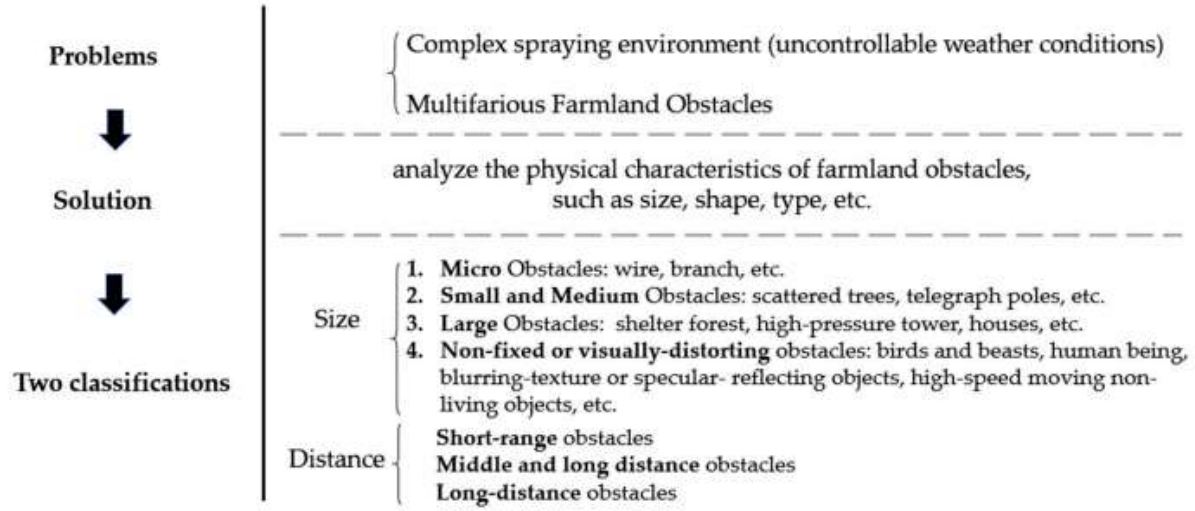

Figure 23: classification à deux niveaux des obstacles dans un milieu agricole. Source : Applications and Prospects of Agricultural Unmanned Aerial Vehicle Obstacle Avoidance Technology in China - Scientific Figure on ResearchGate. Available from : https://www.researchgate.net/figure/Analysis-summary-of-farmland-obstacles_fig1_330895032 [accessed 8 Jul, 2020]

Comme mentionné dans les publications de Cruz & Encarnação (2012) et de Wang et al. (2019), la détection d'obstacles doit se concentrer sur l'espace qui se trouve sur la trajectoire du drone, en général à l'avant. Il est donc primordial de définir cette zone et de toujours avancer dans la direction qui fait face au drone. Il est aussi nécessaire de définir l'angle de vue horizontal et latéral adapté en fonction des besoins et de la situation afin de garantir un coussin de sécurité autour de la machine.

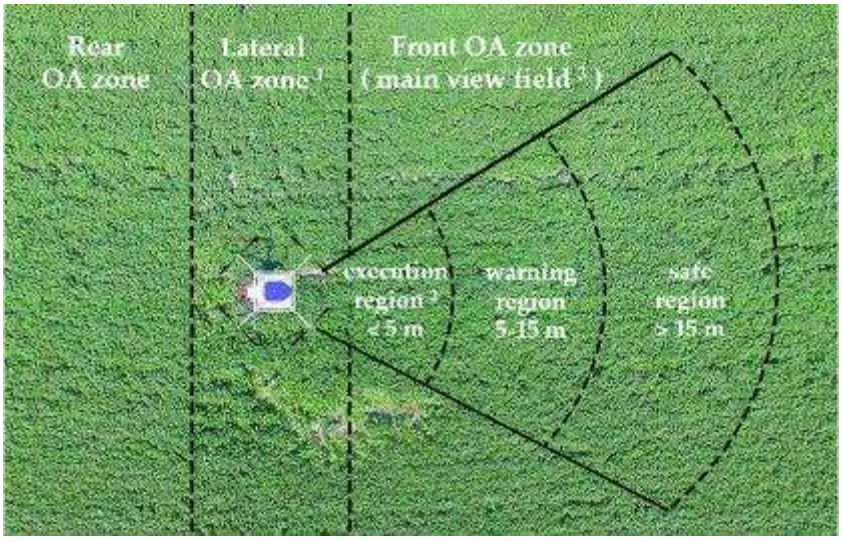

Figure 24: angle et distance de la zone de détection. Source : Applications and Prospects of Agricultural Unmanned Aerial Vehicle Obstacle Avoidance Technology in China - Scientific Figure on ResearchGate. Available from : https://www.researchgate.net/figure/Obstacle-avoidance-zone-of-agricultural-UAVs-1-This-figure-does-not-express-the-OA-zone_fig2_330895032 [accessed 9 Jul, 2020]

Pour ce qui est des mesures à prendre en compte dans le calcul qui détermine si un obstacle est dangereux dans la direction du drone, il est décrit par Gageik, Benz, & Montenegro (2015) qu'il faut tenir compte de la situation autour de l'appareil en récupérant les données du gyroscope (roll, pitch) et les distances mesurées par les senseurs. Cela permet de calculer la distance disponible effective qui se trouve en face du drone. Une étude menée par Xie, Yu et Wang (2019) propose un système de détection d'obstacle pour robots autonomes qui prend aussi en compte la direction (yaw) en plus des deux autres axes principaux afin d'obtenir des mesures lidar dans l'environnement plus précises.

Concernant les techniques d'apprentissage automatique utilisées, plusieurs études décrivent le « Deep Learning » (DL) comme étant la plus efficace dans le milieu de la robotique en utilisant différents types de senseurs (Carrio, Sampedro, Rodriguez-Ramos, & Campoy, 2017). Il est mentionné dans l'article de Rodríguez-Puerta et al. (2020) que le DL rencontre un fort succès dans la reconnaissance d'obstacle embarquée. Ils affirment que l'algorithme "Random Forest" (Forêt d'arbres décisionnels) permet de diminuer le temps du processus d'entraînement du modèle d'apprentissage automatique et ils affirment avoir obtenu les meilleurs résultats avec cet algorithme (Rodríguez-Puerta, et al., 2020). Ils finissent par conclure que les résultats varient peu entre les différentes méthode d'apprentissage automatique, mais que le temps de calcul est plus rapide avec un algorithme « random forest » et les résultats sont plus précis(Rodríguez-Puerta, et al., 2020).

5.2. Acconeer

Nous avons présenté en détail le radar XM112 de Acconeer dans la section 3.1.2. Nous rappelons que ce radar peut être utilisé à des fins de développement et qu'un outil d'exploration open source développé en python est disponible sur la page Github de l'entreprise. Cet outil propose des scripts qui permettent d'utiliser leur radar afin de détecter les obstacles et d'estimer leur position et leur distance par rapport au senseur (Acconeer AB, 2019).

Le script de détection d'obstacles effectue une procédure spécifique sur les données récoltées par le radar. Premièrement, les mesures sont traitées et transformées en matrice de balayage (sweep matrix). Chaque ligne de cette matrice consiste en plusieurs points qui représentent une distance spécifique pour chacun d'eux (Acconeer AB, 2019). Une différence dans les mesures des points entre deux lignes consécutives de la matrice représente un changement dans la mesure de distance qui signifie qu'un obstacle est en mouvement en face du senseur (Acconeer AB, 2019).

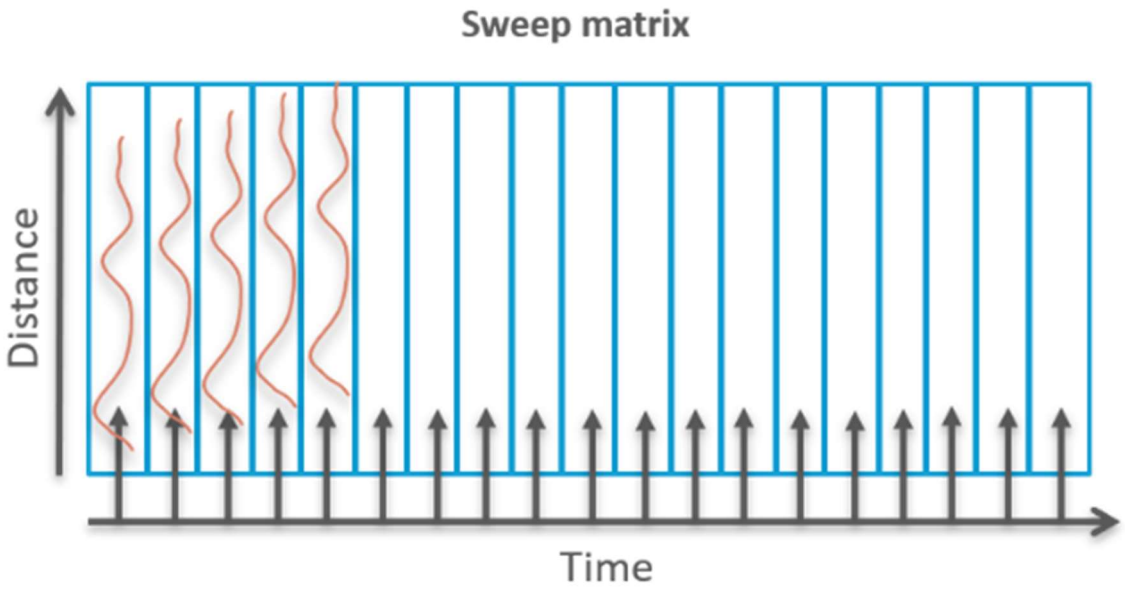

Figure 25: matrice de balayage qui récolte l'évolution de la distance en fonction du temps, image récupérée sur https://acconeer-python-exploration.readthedocs.io/en/latest/processing/obstacle.html

Après avoir récolté les données dans la matrice, le script proposé par Acconeer (2019) utilise un algorithme appelé “Transformation de Fourier rapide” (FFT) qui permet d’améliorer le signal radio des obstacles à proximité en évitant au mieux le bruit pour un meilleur calcul des angles et de la distance. Celui-ci retourne une enveloppe sous forme de ligne qui représente l’amplitude par rapport à la distance et une pointe au niveau de l’amplitude est formée à chaque obstacle détecté. Il est ensuite possible de faire une moyenne de plusieurs enveloppes afin de réduire le bruit. Il faut alors fixer un seuil qui détermine l’amplitude minimale à atteindre dans une enveloppe pour filtrer toutes les mesures qui ne contiennent pas d’obstacle. Sur les mesures restantes qui ont détecté un ou plusieurs obstacles, il est possible de les classer en fonction de la distance ou de l’amplitude. Dans notre cas de figure, nous allons utiliser la distance comme référence car nous souhaitons identifier en premier les obstacles les plus proches du drone. Nous devons alors trouver le premier pic d’amplitude et récupérer la distance mesurée au sommet de cette courbe pour obtenir la distance de l’obstacle le plus dangereux pour l’appareil au moment de la détection.

L'outil d’exploration en python permet ensuite une représentation graphique après avoir traité les données. La matrice de nettoyage peut ainsi être représentée graphiquement afin de visualiser les obstacles présents sur la trajectoire du radar. À noter que cette méthode permet d’identifier plusieurs objets simultanément.

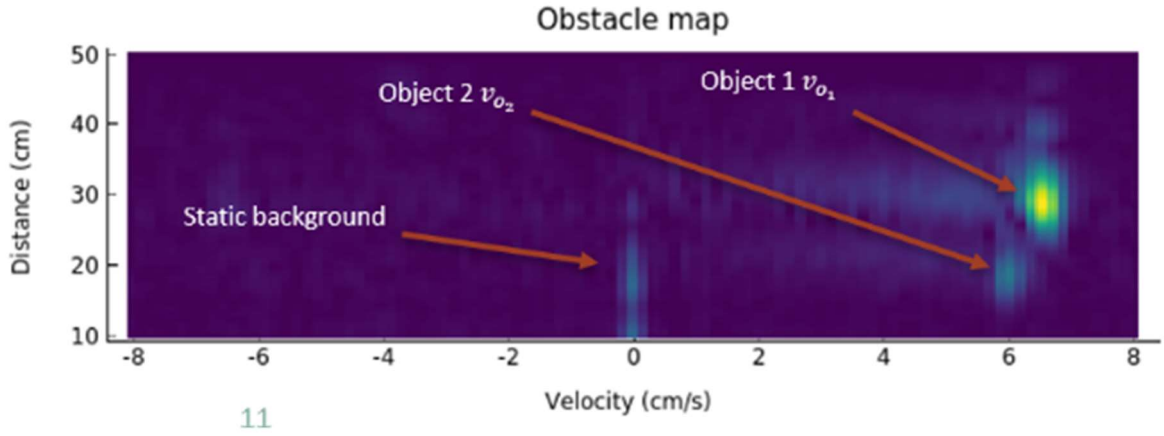

Figure 26: visualisation des obstacles détectés par le radar avec l'outil d'exploration, image récupérée sur https://acconeer-python-exploration.readthedocs.io/en/latest/processing/obstacle.html

Il est donc possible d'utiliser les données récupérées après l'algorithme FFT dans un système d'apprentissage automatique qui permet de déterminer si une manœuvre d'évitement doit être entreprise ou non.

5.3. Ainstein

Ainstein est une entreprise spécialisée dans la détection d'objets et l'analyse en temps réel des données récupérées par les radars, notamment pour les drones d'agriculture. Ils proposent une solution intelligente qui permet de garder le drone à une altitude précise constante par rapport au sol et aux plantations. Leur système est composé d'un radar dirigé vers le sol pour mesurer l'altitude et analyser le terrain ainsi qu'une caméra à l'avant pour la détection d'obstacles sur la trajectoire du drone.

Cette compagnie propose une intégration facile du drone à des logiciels open source qui permettent de planifier des missions ou de contrôler le sol. Il est aussi possible sur demande auprès de l'entreprise d'intégrer le framework « Robot Operating System » (ROS), un outil de développement open source spécialisé dans l'intégration des systèmes robotiques. Il est donc possible de développer une solution personnalisée qui peut être déployée sur le drone en vol, notamment la détection d'obstacles.

Nous n'avons pas trouvé d'autres caractéristiques sur la détection d'obstacle du système Ainstein, il reste néanmoins intéressant de noter que le fabricant affirme que leur drone est capable de maintenir une distance constante de 40 à 100 mètres par rapport au sol.

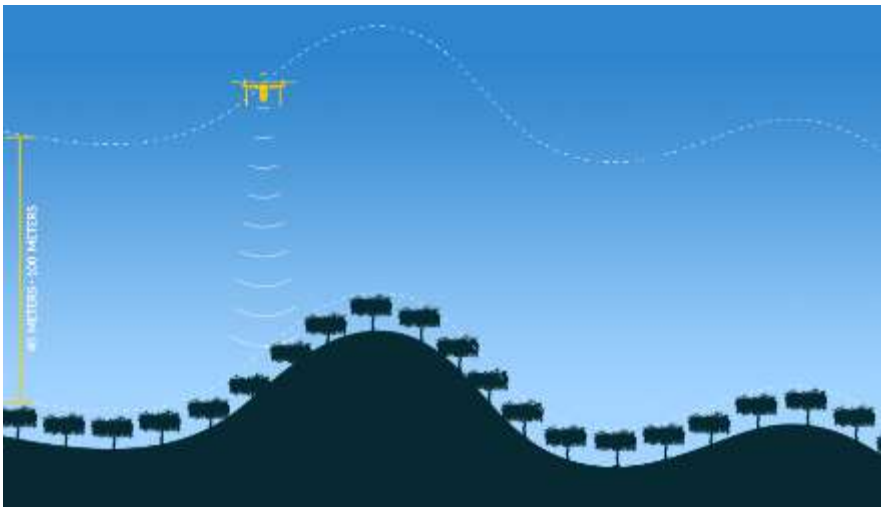

Figure 27: le drone Ainstein garde une distance fixe par rapport au sol, image récupérée sur https://ainstein.ai/precisionag/

5.4. PrecisionHawk

PrecisionHawk propose une solution commerciale de lidar pour drone capable de mesurer les distances des objets autour de la machine et de traiter ces données afin de visualiser l'environnement de manière précise. Ils utilisent un lidar rotatif capable d'obtenir plus de 100'000 mesures par seconde dans l'espace en 3D avec une marge d'erreur de 2 à 3 centimètres. Ils estiment qu'il est nécessaire d'obtenir le plus de données possibles afin d'obtenir de meilleurs résultats et une meilleure flexibilité sur les possibilités de traitement, notamment la reconnaissance d'obstacles (PrecisionHawk, 2020).

L'entreprise propose en plus des lidars un logiciel permettant de traiter les données et de les transformer en nuage de points. Ils recommandent néanmoins un traitement des données après le vol et ils estiment que les opérations sur les données durant le vol doivent uniquement être effectuées en cas d'urgence (PrecisionHawk, 2020). Il est néanmoins intéressant de constater qu'il est efficace d'utiliser un lidar rotatif à trois dimensions afin d'obtenir un grand nombre de données pour des calculs plus précis. Nous estimons qu'un traitement des données durant le vol en 3D peut potentiellement améliorer les résultats d'apprentissage automatique embarqué sur le drone.

6. Choix technologique

Cette section décrit les choix effectués concernant le matériel et les logiciels que nous avons utilisés durant nos tests.

6.1. Restrictions matérielles

Durant nos expériences, une grande partie du matériel utilisé nous a été imposé. Nous n'avons donc pas pu tester physiquement tous les senseurs étudiés. Néanmoins, nous nous sommes basés sur les fiches techniques et les caractéristiques des senseurs non-testés afin de comparer les performances de ceux-ci.

Les senseurs utilisés durant cette étude sont les suivants : trois lidars TFmini Plus avec une ligne de vision en une dimension pointés vers l'avant du drone contre le bas avec trois angles différents et un radar XM112 de la marque Acconeer à l'avant pour une détection en trois dimensions. Le « companion computer » installé sur le drone avec lequel nous avons effectué tous les tests est un Raspberry Pi 3b+.

6.2. Restrictions logicielles

Lors de cette étude, nous avons travaillé avec plusieurs restrictions sur les logiciels et « frameworks » à utiliser.

Premièrement, nous souhaitons nous orienter le plus possible sur des solutions gratuites ou disponibles avec un budget limité. Le critère du prix est donc l'un des éléments les plus importants à prendre en considération lors de la comparaison des programmes. La règle suivie est la suivante : s'il est possible d'atteindre nos objectifs et d'obtenir des résultats avec un logiciel gratuit, nous choisirons cet outil. De plus, nous souhaitons avoir une certaine liberté de modification du code source dans le cas où il serait nécessaire d'adapter le programme afin d'y intégrer nos propres données et processus. C'est pour cela que, si les conditions mentionnées ci-dessus sont remplies sur des logiciels « open source », alors nous mettrons ces solutions en avant.

6.3. Choix du type des senseurs

6.3.1. Tableau de comparaison

Nous avons mis en place des tableaux comparatifs pour l'état de l'art sur les types de senseurs ainsi que les logiciels permettant une visualisation dans l'espace. Ces deux tableaux regroupent tous les critères que nous avons jugé pertinents ainsi qu'une pondération en fonction de nos besoins. La

pondération varie de 1 pour les critères avec une importance modérée jusqu'à 3 pour les critères primordiaux que nous souhaitons mettre en avant.

Les notes données pour chaque catégorie vont de 1 à 5 et la note 5 signifie que le critère est entièrement rempli et nous sommes parfaitement satisfait des résultats.

Nous allons finalement calculer les résultats de chacun des systèmes testés ou étudiés pour créer un classement qui va nous aider à faire une décision sur le matériel et les logiciels que nous allons utiliser durant cette étude.

La figure ci-dessous représente le tableau de comparaison pour les senseurs que nous avons analysés. Nous avons mis l'accent sur la capacité du senseur à fonctionner correctement dans de mauvaises conditions atmosphériques, car il se peut que du liquide se trouve sur sa trajectoire durant les traitements. De plus, nous souhaitons pouvoir développer nous-mêmes les processus à effectuer avec les données récupérées par les senseurs. Il est donc important que le système soit personnalisable. Nous devons par ailleurs garder à l'esprit que le senseur doit être précis et qu'il doit être facilement intégré au drone sans prendre trop de place ni trop d'espace et que la consommation d'énergie doit être la plus faible possible.

Thème	Yanwu Tech FMK24-E	Note	DJI AGRAS Obstacle Avoidance Radar	Note	Pondération
Portée	20 mètres, 78° horizontal et 23° vertical	4	30 mètres, 50° horizontal et 10° vertical	4	1
Poids et dimensions	70*60*20 mm pour un poids de 70 grammes	4	406g (lourd) et environ 10-15 cm3	2	2
Précision	.+/- 10 cm de précision, fréquence de 24 GHz	4	marge d'erreur de 10 cm, fréquence 24 GHz	5	2
Conditions atmosphériques	Radar micro-ondes, performant lors de mauvais temps et pendant la nuit	5	Radar micro-ondes, performant lors de mauvais temps et pendant la nuit	5	3
Open source / propriétaire	Software personnalisable et facilement intégré à un système existant	4	DJI Propriétaire, peu de marche de manoeuvre pour le développement	2	3
Prix	CHF 70	4	CHF 900	1	3
Total		**4.21**		**3**	

Thème	Acconeer XM112	Note	TFmini Plus	Note	Pondération
Portée	10 mètres, vision de 40° de rayon	3	12 mètres	3	1
Poids et dimensions	carte de 24x16 mm	5	35 x 21 x 18.5 mm pour 11 g	4	2
Précision	Les objets sont détectés mais la précision est limitée	4	marge d'erreur de 5 cm, un seul point détecté par scan	3	2
Conditions atmosphériques	Performant lors de mauvaises conditions atmosphériques	4	Potentiellement altéré par les mauvaises conditions atmosphériques	3	3
Open source / propriétaire	Offre un kit de développement et favorise la modification du code	5	Software personnalisable	4	3
Prix	CHF 60	4	CHF 35	5	3
Total		**4.29**		**3.79**	

Figure 28: tableau comparatif des types de senseur, image de l'auteur

Nous constatons que la solution la mieux classée est le radar XM112 de Acconeer car, excepté pour la portée, il permet de répondre à nos besoins. Nous rappelons que la portée est négligeable car le drone en mission vole à environ deux mètres au-dessus des cultures. Nous avons observé que Acconeer

met à disposition un logiciel d'exploration open source qui peut être modifié en fonction de nos besoins et qui permet une intégration personnalisée en Python des mesures radar au « companion computer ».

Nous retrouvons en deuxième position un deuxième radar de nos échantillons de test. Il s'agit du radar à micro-ondes FMK24-E de Yanwu Tech. Nous constatons que la détection en utilisant les micro-ondes est la plus efficace lors de mauvaises conditions atmosphériques. Cela signifie que cette technologie est particulièrement adaptée à notre situation puisque le liquide de traitement peut se situer entre le senseur et l'obstacle à éviter.

En comparaison aux radars, nous avons constaté que les résultats obtenus avec la technologie lidar peuvent potentiellement être altérés par les conditions atmosphériques. Même si les résultats obtenus avec le lidar TFmini plus sont concluants, nous estimons qu'il n'est pas suffisant de détecter un seul point à la fois sur une dimension car un obstacle peut se trouver juste à côté du laser tout en étant toujours sur la trajectoire du drone. Nous estimons qu'il est intéressant dans une autre étude de tester des lidars rotatifs à deux ou trois dimensions afin d'obtenir une vision linéaire de l'environnement autour du drone et d'éviter les angles morts de l'appareil.

Finalement, nous avons constaté que le radar proposé par DJI n'est pas adapté à notre situation pour des raisons financières. De plus, ce senseur ne peut pas être installé sur des machines personnalisées. Il reste néanmoins le radar le plus performant du test et utilise aussi les ondes micro-ondes adaptées à la situation.

6.3.2. Choix final

D'après les résultats, nous estimons que le radar XM112 de Acconeer est le plus adapté pour une détection d'obstacles intégrée à un drone. Nous avons donc décider d'effectuer cette étude avec ce senseur ainsi que les lidars TFmini afin de pouvoir déterminer si les deux technologies sont efficaces dans notre cas de figure. Ce choix découle aussi du fait que nous étions en possession de ces deux modèles lors des tests en situation.

6.4. Représentation de l'environnement

6.4.1. Tableau de comparaison

En utilisant la même méthodologie que celle décrite pour le choix du type de senseur à la section 6.3.1, nous avons effectué un tableau comparatif pour identifier la meilleure solution logicielle permettant de visualiser les données géographiques récupérées sur une carte dynamique en trois dimensions. Il s'agit ici de trouver le SIG le plus adapté pour une visualisation simple et rapide des données récoltées par le drone.

Nous avons défini plusieurs critères comme étant plus pertinents dans la pondération afin de mieux cerner nos besoins. Le plus important dans notre situation est de trouver un logiciel rapide et qui peut se prendre en main sans devoir effectuer une formation spécifique au programme. Il n'est pas nécessaire d'utiliser un grand nombre de fonctionnalités car nous souhaitons effectuer un processus de visualisation simple sans avoir à modifier les données.

Nous pensons qu'il est crucial que les résultats soient à la hauteur de nos attentes. Cela signifie qu'ils doivent être précis et doivent refléter au mieux la réalité sur le terrain. L'utilisateur doit aussi être capable de visualiser les données de manière libre dans l'espace et d'obtenir des informations sur les mesures prises durant les vols.

Il existe un vaste choix de type de fichier pour stocker des données géographiques. Pour cette étude, nous avons sélectionné deux types standards que nous allons utiliser tout au long du processus de visualisation. Il s'agit du type csv (« comma-separated values ») car nous obtenons les données brutes sous cette forme ainsi que le format kml (« keyhold markup language »). Ce dernier est décrit comme étant un format standard mondial basé sur du XML (« Extensible Markup Language ») pour coder et représenter des données géographiques dans de multiples SIG qui permettent une visualisation sur une carte 2D ou un globe 3D (Open Geospatial Consortium, 2015).

Finalement, nous avons étudié la possibilité d'utiliser ces programmes sur différentes plateformes, notamment Windows, Mac et les distributions Linux. De plus, nous souhaitons favoriser les logiciels gratuits et open source si les résultats sont concluants.

Thème	Google Earth Pro	Note	Earth Enterprise	Note	Pondération
Facilité d'installation	Facile et rapide	5	Client: facile, Server + fusion: lent et complexe	2	2
Facilité d'utilisation	Intuitif	5	Server: complexe, demande du temps pour utiliser correctement	2	3
Open source / propriétaire	Propriétaire Gratuit	4	Open source, gratuit	5	2
Logiciel multiplateforme	Windows, Mac, Linux	5	Server + Fusion: limité à ubuntu 16.4, centOS et RedHat	2	1
Qualité des résultats	Excellent, vision "street view" disponible	5	Excellent sur le client	4	3
Intégration formats GPS	CSV + KML ok + styles kml visibles	5	csv + kml ok	4	3
Total		**4.86**		**3.29**	

Thème	QGIS	Note	ArcGIS	Note	Pondération
Facilité d'installation	Facile et rapide	5	Facile et intuitif	5	2
Facilité d'utilisation	User interface intuitive mais prise en main complexe	3	Intuitif, complet	5	3
Open source / propriétaire	Open source, gratuit	5	Propriétaire, payant	1	2
Logiciel multiplateforme	Windows, Mac, Linux	5	Windows uniquement	2	1
Qualité des résultats	Mauvais, résultats désirés pas obtenus	2	Résultats précis	4	3
Intégration formats GPS	csv + kml ok	4	CSV + KML ok, beaucoupd'autres formats disponibles	5	3
Total		**3.71**		**4**	

Figure 29: tableau de comparaison des logiciels de visualisation de données géographiques, image de l'auteur

Nous avons constaté que le logiciel le mieux noté de notre liste de tests est Google Earth Pro et il devance les autres de manière considérable. Cela est dû au fait que nous avons été entièrement satisfaits des résultats obtenus à l'aide d'un logiciel accessible, gratuit et facile d'utilisation. Nous estimons que ce programme est le plus adapté à notre cas de figure.

Le programme ArcGIS nous a permis de visualiser nos données de manière précise sans passer par une procédure complexe. Néanmoins, celui-ci reste un choix onéreux et dispose de beaucoup d'outils que nous ne souhaitons pas utiliser lors de nos phases de visualisation. Le programme reste néanmoins une solution pertinente dans le cas où nous souhaitons à l'avenir effectuer des modifications ou des processus complexes directement sur les données dans l'espace.

Le programme open source QGIS ne nous a pas permis d'obtenir les résultats désirés dans le temps imparti et nous avons été forcé à abandonner cette solution.

Finalement, Earth Entreprise se voit attribuer la note la plus basse car nous avons constaté au milieu de nos tests qu'il s'agit d'une solution destinée aux personnes souhaitant créer et héberger leur propre serveur de cartes 2D et de globes 3D.

6.4.2. Choix final

D'après les résultats obtenus lors de la comparaison, nous avons décidé d'utiliser Google Earth Pro pour nous différentes visualisations. Ce logiciel est le plus adapté à notre situation et nous avons constaté que les fichiers kml sont plus personnalisables que sur les autres programmes testés. Nous avons tout de même obtenu une visualisation globale des différents SIG disponibles sur le marché et leurs particularités et nous sommes en mesure de déterminer le meilleur logiciel pour un certain type d'utilisation.

7. Implémentation

Nous allons maintenant décrire le processus effectué pour implémenter la détection d'obstacles à l'aide de l'apprentissage automatique en passant par la transformation, la visualisation, la modélisation et l'évaluation des données.

7.1. Description du vol d'essai

Durant les premiers tests, nous avons planifié plusieurs vols de tests qui se rapprochent au mieux d'une situation concrète de traitement des vignes. Pour des raisons de sécurité, nous avons fixé l'altitude à environ 10 mètres du sol. À cette distance du sol, les senseurs ne devraient théoriquement pas détecter d'obstacle, sauf durant le décollage et l'atterrissage. La durée d'un vol était d'environ 5 minutes.

Lors des vols suivants, nous avons diminué progressivement l'altitude afin d'obtenir des mesures de distances lidar plus courtes pour obtenir des données à classer comme étant des situations à risque de collision. Nous avons donc piloté le drone manuellement à une distance d'environ 3 à 5 mètres au-dessus du sol avec quelques variations pour obtenir des données variées.

7.2. Récolte de données

Afin d'obtenir des données sur lesquels travailler et tester la reconnaissance d'obstacles, nous sommes aller faire voler le drone au-dessus des vignes. Durant ces tests, la machine était équipée d'un radar à l'avant orienté dans le sens de marche de manière horizontale ainsi que deux lidars à l'avant orientés vers le bas en diagonale à 45 degrés (voir ci-dessous). Les deux lidars sont orientés à l'avant gauche et l'avant droit de la machine.

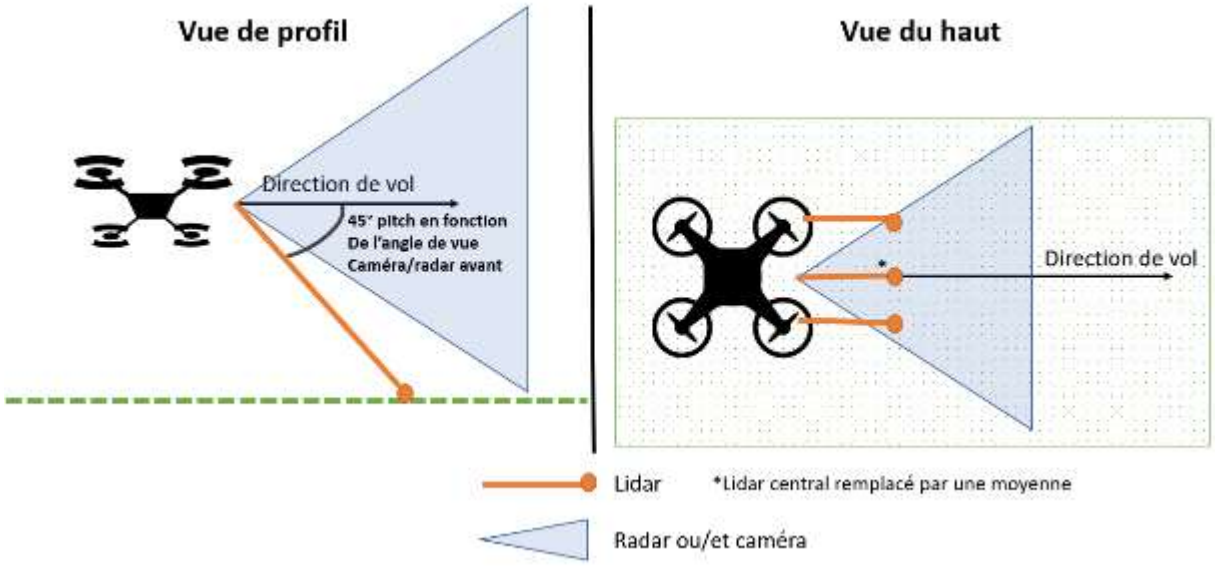

Figure 30: Position et direction des senseurs lors des premières récoltes de données avec la moyenne des deux lidars comme mesure centrale, image de l'auteur

Avec l'aide de deux scripts disponibles sur le GitHub du constructeur du radar Acconeer (https://github.com/acconeer/acconeer-python-exploration), nous avons récolté les données fournies par tous les senseurs sur la machine. Les scripts utilisés sont nommés basic.py et basic_continuous.py. Ils nous ont permis de récolter les données radar et lidar et de constituer un fichier .csv. Ces mesures sont prises toutes les 2.7 secondes environ. Après plusieurs essais, nous avons constaté que le délai entre chaque mesure était trop long, nous sommes donc passé à une mesure toutes les demi secondes, puis toutes les 0.2 secondes (5 mesures par seconde).

Avec cette étape, nous avons récolté plusieurs séries de données radar durant plusieurs vols d'essais. La figure ci-dessous montre un exemple d'une ligne de donnée contenant la matrice de points retournés par le radar ainsi que les mesures de points prisent par les lidars à un temps précis.

```
uid;timestamp;cnt;left;right;radar
1;2020-05-06-14:44:28.239;150;0;33;"[104. 106. 108. 110. 112. 114. 116. 118. 120. 122. 124. 126. 128. 130.
 132. 134. 136. 138. 140. 142. 144. 146. 148. 148. 148. 148. 148. 148.
 148. 148. 148. 148. 148. 148. 148. 148. 148. 148. 148. 148. 148. 146.
 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146.
 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146.
 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146.
 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146.
 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146. 146.
 146. 148. 148. 148. 148. 150. 150. 150. 150. 152. 152. 152. 154. 154.
 154. 154. 156. 156. 156. 158. 158. 160. 160. 160. 160. 162. 162. 162.
 162. 164. 164. 164. 164. 164. 164. 164. 166. 166. 166. 166. 166. 166.
 166. 166. 168. 168. 168. 166. 166. 166. 166. 168. 168. 168. 168. 168.
 168. 168. 170. 170. 170. 170. 170. 170. 170. 170. 170. 170. 170. 170.
 170. 170. 172. 172. 172. 172. 172. 172. 172. 172. 172.]";['1', '20200424-123942', '46.3211', '6.97338', '480.485', '1.18671', '-1.45473', '-43.1112']
```

Figure 31: Données radar et lidar brutes récoltées durant les tests, image de l'auteur

Nous constatons que la matrice de distance du radar est délimitée par des points et des retours à la ligne. Cette matrice représente une ligne qui représente la distance et l'amplitude des objets

détectés à un temps précis et si l'on compare une mesure à un temps T avec une autre à un temps T+1, nous pouvons identifier les obstacles sur la trajectoire du radar.

7.3. Traitement des données pour visualisations – Radar

Afin de mieux comprendre les données récoltées à l'étape précédente, nous avons utilisé le programme d'analyse de données Knime pour mieux les visualiser. Nous avons donc créé un « workflow » qui prend en paramètre d'entrée le tableau de données brutes et qui retourne un graphique de surface qui représente la distance mesurée entre le radar et les obstacles en face de lui à un moment donné. Au début de cette étude, nous avons eu quelques difficultés à comprendre les données retournées par le radar et nous les avons interprétées de la mauvaise manière. En effet, nous avions estimé que le tableau de valeurs retourné représentait une série de points dans l'espace en 3D avec chaque point qui représente une distance par rapport à l'objet détecté. Après plusieurs phases de recherches, nous avons constaté que cette matrice n'est pas une mesure en 3D mais une ligne en 2D qui représente la distance et l'amplitude à un temps donné et qu'il est possible d'identifier les mouvements sur la trajectoire du radar en comparant la mesure d'un temps T avec celle d'un temps T+1.

Après de nouvelles recherches sur la façon d'utiliser les données radar décrites en 5.2, nous avons retravaillé les données afin de les visualiser sous la bonne forme comme décrite dans le manuel de l'outil d'exploration Python de Acconeer (Acconeer AB, 2019). Les résultats que nous avons obtenus sont décrits en 7.5.1.

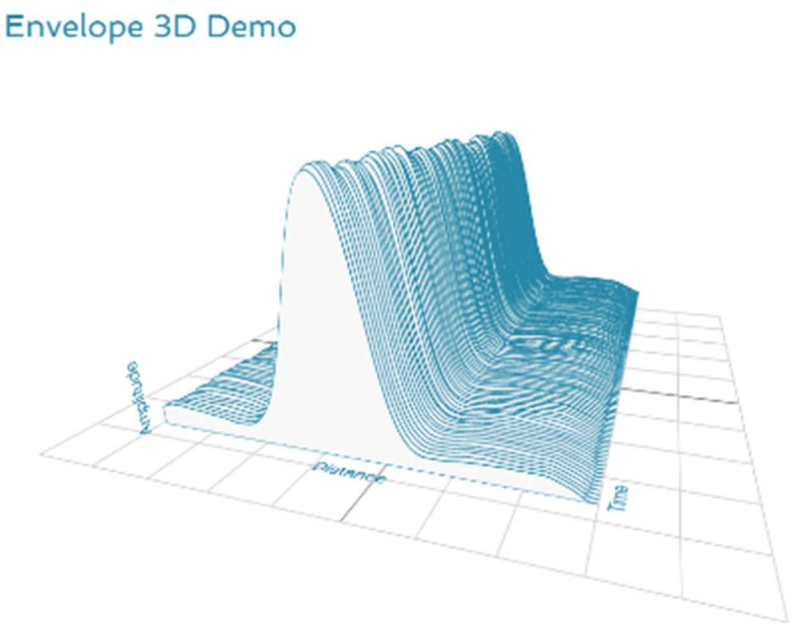

Figure 32: représentation de l'enveloppe des données radar, image récupérée sur https://acconeer-python-exploration.readthedocs.io/en/latest/sensor_introduction.html#sensor-intro

Nous avons implémenté un script en Python qui permet de visualiser les mesures sous la bonne forme en créant un graphique en 3D dont les trois axes représentent respectivement le temps en X, la distance en Y et l'amplitude en Z. Pour chaque mesure, une ligne est tracée sur le plan de l'axe du temps X.

Le script nommé « raw_to_radar_plot.py » est disponible en annexe I, et fonctionne de la manière suivante : l'utilisateur doit sélectionner un fichier csv sous la forme brut, c'est-à-dire un fichier de mesure qui a été extrait d'un vol du drone et qui n'a pas été modifié. Nous utilisons ensuite la librairie Python « Mathplotlib » pour créer un graphique en 3D. Une méthode nommée « plot_graph » permet de tracer une à une les lignes qui correspondent aux mesures radars en commençant par extraire toutes les mesures séparément du fichier brut, puis en les ajoutant au graphique. Cette méthode prend deux paramètres en entrée qui correspondent à l'index de la première et à la dernière mesure à afficher.

À l'aide de deux « sliders », il est possible de choisir la mesure de départ et de fin de manière dynamique. Un « listener » permet d'appeler à nouveau la méthode pour tracer les données avec la nouvelle fourchette de valeur. En faisant glisser le « slider » de la dernière valeur, il est possible de voir l'évolution des courbes au fil du temps.

7.4. Traitement des données pour visualisations – Lidars

7.4.1. Prévisualisation

Le drone possède un radar et deux lidars situés à l'avant de la machine comme indiqué sur la Figure 30. Contrairement au radar qui retourne une matrice de points dans l'espace, un lidar calcul uniquement la distance en face de lui et retourne un point unique. Nous avons donc à notre disposition deux mesures des lidars afin d'établir une visualisation du terrain.

Dans le but de comprendre les données retournées par ces trois senseurs, nous allons représenter les distances mesurées en fonction du temps lors d'un vol. Nous avons utilisé le programme Knime pour effectuer une prévisualisation rapide des données afin de trouver au plus vite les données utilisables et fiables. Pour cette tâche, la seule transformation nécessaire est la conversion de l'altitude en centimètres plutôt qu'en mètres pour obtenir la même échelle que les mesures lidar. Il ne reste plus qu'à récupérer toutes les données nécessaires depuis un fichier brut csv, puis de les insérer dans un graphique 2D qui représente la distance par rapport au temps.

La figure ci-dessous représente le résultat d'une prévisualisation des données lidars à l'aide du « workflow » Knime.

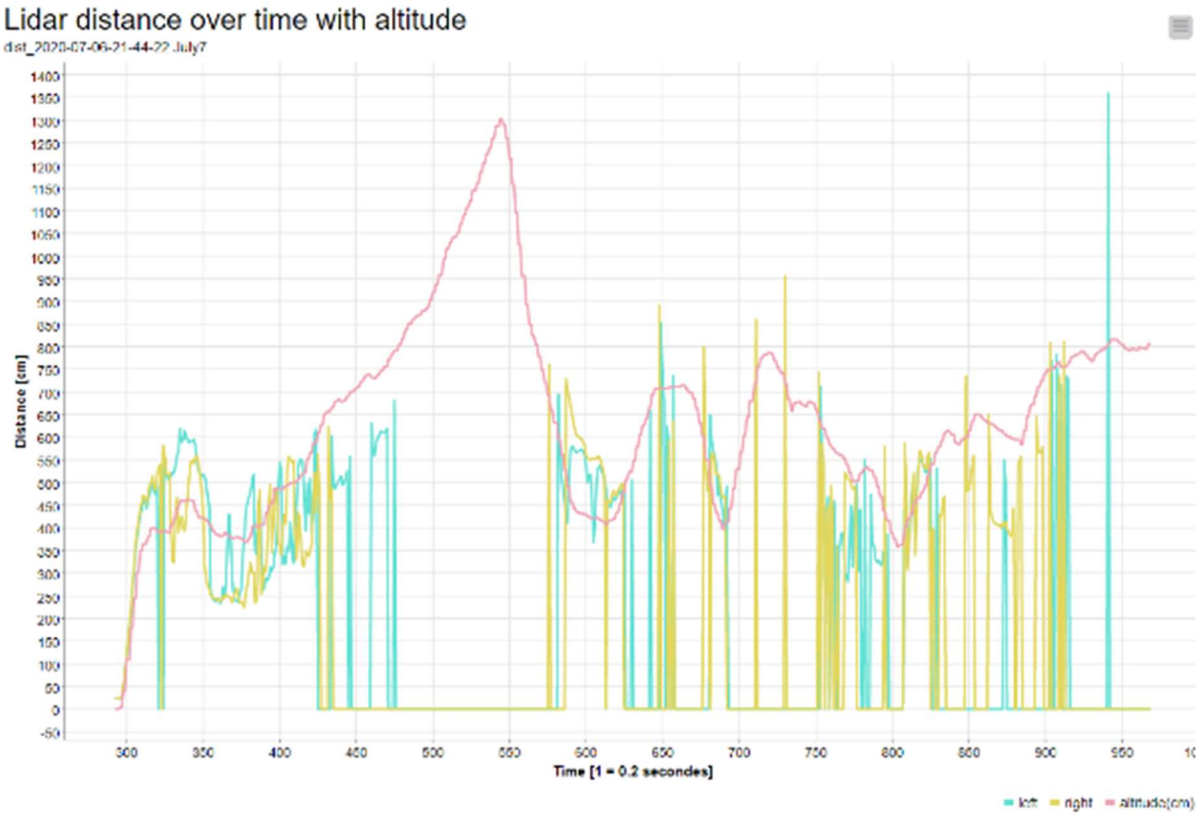

Figure 33: distances en centimètres mesurées par les deux lidars toutes les 0.2 secondes, image de l'auteur

Lors de la prévisualisation, nous constatons que le drone a décollé aux alentours du temps 300 et que les mesures lidar suivent de manière régulière l'altitude du drone. Puisque l'altitude est absolue par rapport au niveau de la mer et ne représente pas la véritable distance entre le drone et le sol, nous observons que les mesures lidar se situent parfois plus haut et plus bas que l'altitude. Cela est dû au fait que nous avons effectué les tests sur un terrain en pente et cela signifie que lorsque la mesure lidar est plus petite que celle de l'altitude, le sol sous le drone est plus haut que lors du décollage. Nous pouvons aussi constater que les senseurs retournent une mesure de zéro lorsque l'altitude du drone dépasse les huit mètres environ. Cela correspond à la portée maximale des lidars TFMini Plus.

7.4.2. Visualisation dans l'espace

Afin de représenter la trajectoire du drone ainsi que les points détectés par les lidars, nous avons transformé les données pour pouvoir créer un fichier de données géographiques sous le format kml. Il était nécessaire d'effectuer un processus géométrique complexe pour obtenir les coordonnées géographiques des points lidar car la seule information retournée par les lidars est la distance et non pas la position.

Nous sommes partis du principe que nous connaissons plusieurs informations qui peuvent nous permettre de calculer les coordonnées des points lidar. Premièrement, nous connaissons la position GPS et l'altitude du drone à un temps T, nous savons aussi que le lidar pointe dans le sens de marche du drone avec un angle de 45° en direction du sol. Les données concernant l'orientation du drone (roll, pitch et yaw) sont aussi connues et, finalement, nous connaissons la distance par rapport à l'obstacle. Il est donc possible de calculer la direction réelle du lidar à l'aide des angles d'Euler (voir 2.5.1), puis, en fonction du vecteur résultant ainsi que la distance de celui-ci, calculer les coordonnées du nouveau point en utilisant une formule de conversion entre distance, latitude, longitude et altitude. Afin de mieux visualiser le calcul des rotations du drone, nous avons créé un script Python « euler_to_vector.py » qui trace un graphique de la direction du drone et des deux lidars avant à la position neutre sans rotation et à la position finale après avoir appliqué la rotation en fonction du « roll, pitch, yaw ». Le graphique ci-dessous montre le résultat d'une rotation avec la trajectoire du drone représentée par les vecteurs épais et les lidars représentés par les vecteurs fins. Ce script nous a permis de confirmer que la matrice de rotation correcte est celle de « Tait-Bryan ZYX » comme mentionné en 2.5.1 et à la Figure 10.

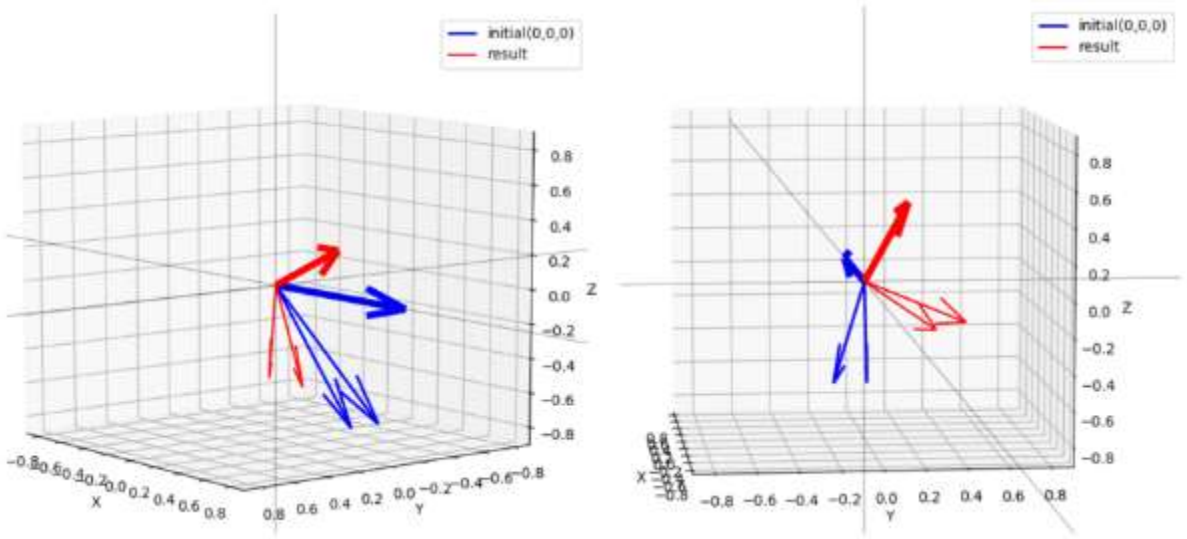

Figure 34: représentation sous deux angles différents d'une rotation d'euler de 20° en "yaw", -20° en "pitch" et 20° en "roll" avec la trajectoire neutre (bleu) et le résultat après rotation (rouge), image de l'auteur

Nous avons ensuite mis en place un script Python nommé « raw_to_kml.py » disponible en annexe III qui permet à l'utilisateur de sélectionner un fichier csv brut pour transformer les données et créer un fichier kml contenant une ligne représentant la trajectoire du drone et les points détectés par les lidars.

Dans un premier temps, le fichier brut est lu et toutes les données des les lidars, de la position GPS et de l'orientation du drone sont récupérées. Les tableaux de données contenant les coordonnées

des points détectés sont ensuite créés, respectivement un tableau pour la gauche et un pour la droite. Pour cela, une méthode spécifique pour chaque côté est appelée. Celle-ci va appeler la méthode générique « get_lidar_point » car nous ne pouvons pas utiliser une méthode itérative « apply() » avec plusieurs paramètres d'entrée, notamment le vecteur unitaire qui est différent du côté gauche ou droite ainsi que la distance mesurée par les deux lidars. Dans cette méthode, une rotation d'Euler est effectuée sur le vecteur du lidar pour trouver son orientation réelle à l'aide des axes « roll, pitch, yaw ». Le vecteur unitaire est alors multiplié à la distance mesurée par le lidar pour trouver le vecteur réel. Celui-ci comporte la distance en mètres à ajouter dans le sens de la latitude, la longitude et l'altitude.

Ces distances sont finalement transformées en différence de latitude, longitude à l'aide de formules mathématiques qui simulent la forme de la terre. Cette différence et ajoutée aux coordonnées GPS du drone pour trouver les coordonnées du point détecté. Il est important de noter qu'il n'existe pas de modèle mathématique qui représente la terre avec exactitude. Même s'il existe des modèles qui se rapprochent fortement de la réalité, nous avons décidé d'utiliser une formule plus simple qui permet d'obtenir des résultats précis sur des petites distances et lorsque l'on ne se rapproche pas des pôles. Cette méthode fonctionne parfaitement dans notre cas de figure puisque les mesures lidar ne dépassent pas huit à dix mètres.

```
r_earth = 6378137.  # constant for earth's radius according to WGS84 (6'378'137 meters)
pi = math.pi
x = vector_distance_x  # distance for longitude
y = vector_distance_y  # distance for latitude
z = vector_distance_z  # distance for altitude

startLON = initial_longitude
startLAT = initial_latitude
startALT = initial_altitude

# Calculs, retrieved at:
# https://stackoverflow.com/questions/7477003/calculating-new-longitude-latitude-from-old-n-meters
Longitude = startLON + (x / r_earth) * (180. / pi) / math.cos(startLAT * pi / 180.)
Latitude =  startLAT + (y / r_earth) * (180. / pi)
Altitude =  startALT + z
```

Figure 35: formules mathématiques pour calculer les nouvelles coordonnées GPS après l'addition d'une courte distance, image de l'auteur

Maintenant que le script a créé les tableaux contenant toutes les coordonnées des points lidars ainsi que le « timestamp » correspondant, il reste à tracer la ligne qui correspond à la trajectoire du drone en fonction de sa position GPS. Pour cela, nous allons créer une chaîne de caractères contenant plusieurs lignes qui correspondent à l'historique de la position du drone de manière à pouvoir l'ajouter en un bloc au fichier kml dans la balise « LineString » et « coordinates ».

Nous avons à présent toutes les informations géographiques nécessaires pour pouvoir créer le fichier désiré. Le script va donc créer et écrire un nouveau fichier kml en suivant les conventions mises en place par google pour ce type de données. Des librairies permettant la création de fichiers

kml sont disponibles, mais nous avons décidé de créer notre propre processus afin d'avoir une plus grande liberté sur les styles et sur la mise en forme choisis. Puisque nous souhaitons obtenir à chaque utilisation la même structure kml, nous avons créé des fichiers textes contenant chaque partie générique qui doit obligatoirement figurer dans le fichier (voir les exemples à la fin de l'annexe III). Entre deux parties de texte statique, le script écrit les données créées auparavant dans les balises correspondantes.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Placemark>
    <name>Simple placemark</name>
    <description>Attached to the ground. Intelligently places itself
       at the height of the underlying terrain.</description>
    <Point>
      <coordinates>-122.0822035425683,37.42228990140251,0</coordinates>
    </Point>
  </Placemark>
</kml>
```

Figure 36: exemple d'un fichier kml simple avec un point dans une balise "Placemark", image de Google récupérée sur https://developers.google.com/kml/documentation/kml_tut

7.4.3. Problèmes rencontrés

Durant une grande partie de l'étude, nous avons obtenu des résultats qui ne représentaient pas la réalité. Les mesures lidars ne correspondaient pas à l'altitude et le temps entre chaque mesure était trop élevé (2.7 secondes environ).

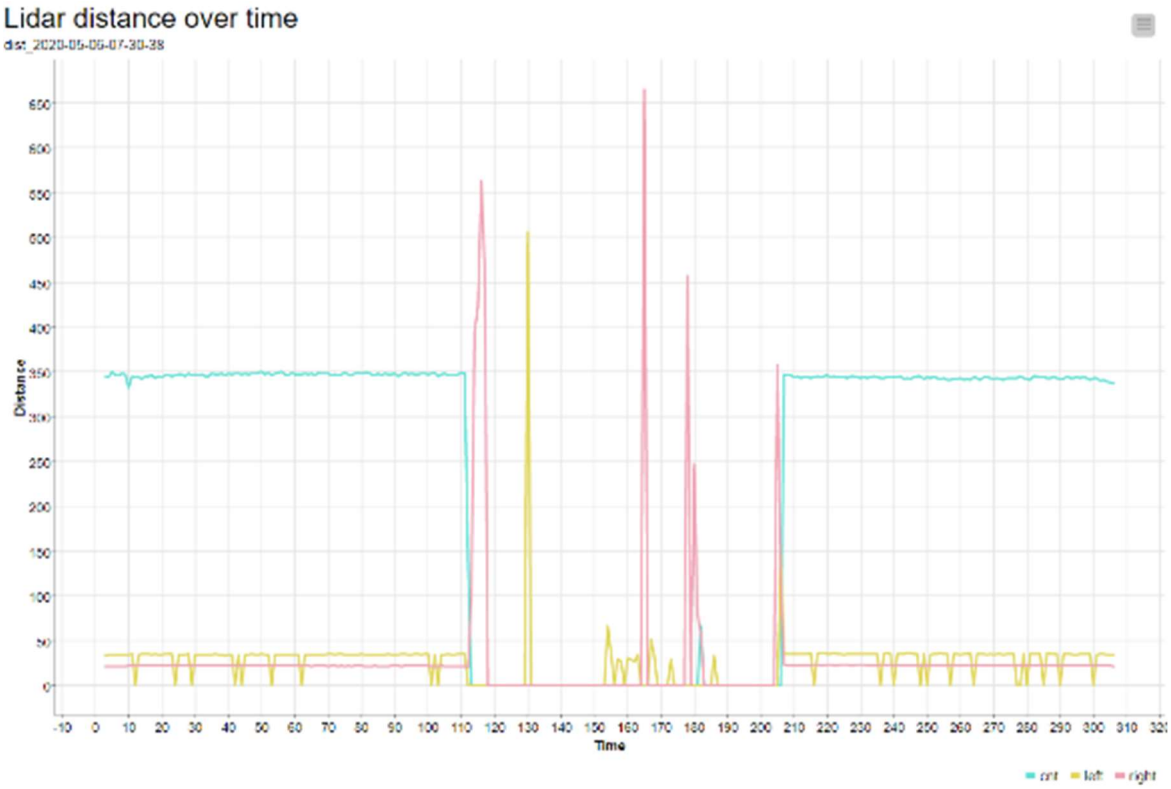

Figure 37: distances en cm mesurées par les trois lidars toutes les 2.7 secondes, image de l'auteur

Malgré la détection de certains lidars, notamment celui de droite, nous ne pouvons pas affirmer qu'un objet a été identifié car un seul point de mesure indique la distance avant de retourner à zéro. De plus, nous observons une détection potentielle d'obstacle entre 25 et 50 cm par le senseur gauche (en jaune sur la Figure 33) alors que l'appareil en vol se trouvait à 10 mètres d'altitude dans une zone dégagée.

Nous pensons que les différences entre les mesures et la réalité sont liés à l'un ou plusieurs facteurs. Le premier peut venir du fait que nous avions au départ un intervalle entre les mesures élevé (2.7 secondes) qui peut diminuer la qualité des données.

Concernant les détections peu fiables du radar gauche pendant le vol, il peut d'agir d'un disfonctionnement mécanique de celui-ci ou alors de la détection du spray en cours sous le drone lors du traitement test. Nous avons notamment observé des gouttes d'eau présentes sur les viseurs lidars lorsque celui-ci a terminé sa mission et après l'atterrissage. Il se peut donc que le liquide déversé par le drone crée des interférences sur les mesures lidar.

7.5. Résultats des visualisations et compréhension des données

7.5.1. Données radar

À l'aide du script « raw_to_radar_plot.py » décrit en 7.3 et disponible à l'annexe I, nous avons pu observer les mesures du radar sous la forme désirée.

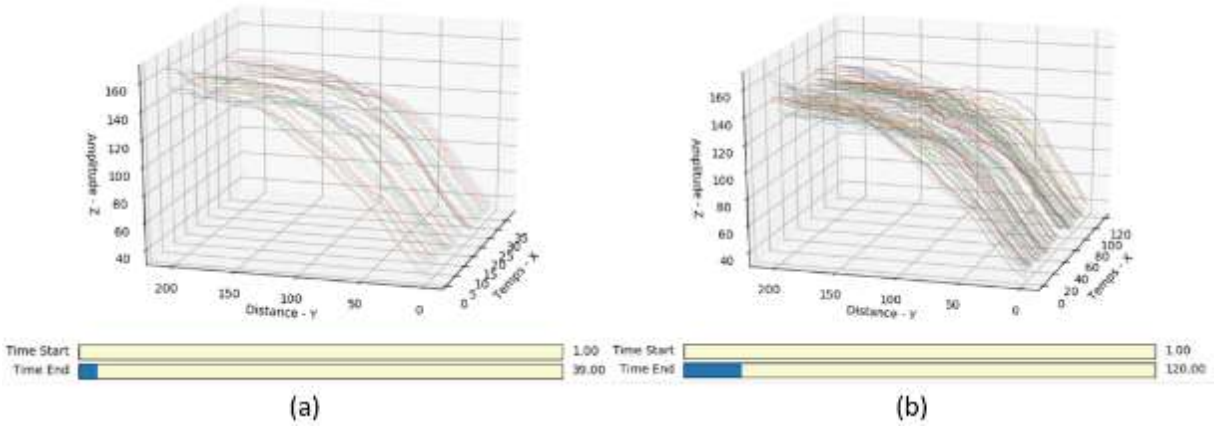

Figure 38: résultat de la visualisation des données radar. (a) 40 mesures avec faibles variations de courbre. (b) 120 mesures avec une variation importante. Image de l'auteur

Nous constatons qu'il est possible d'identifier les variations d'amplitude d'un vol sur une longue durée. La Figure 38 (b) permet d'afficher 120 mesures avec un intervalle de 0.2 seconde pour un total de 24 secondes. Il est aussi possible de partir du même temps de départ et de fin et de faire glisser le « slider » pour voir l'évolution de la courbe au fil du temps. Grâce à cette représentation, nous avons pu prendre connaissance des informations qui nous sont données par le radar ainsi que la manière dont nous pouvons les utiliser pour atteindre notre but de détection d'obstacle.

Afin de mieux comprendre les liens entre ces données et la réalité, nous avons effectué des tests statiques à l'intérieur avec le drone au sol pointant dans la direction d'un mur à une distance fixe de celui-ci. Nous sommes partis du principe que la courbe d'amplitude et de distance doit rester stable puisqu'aucun mouvement n'est effectué sur la trajectoire du radar. En visualisant les résultats de cette expérience, nous avons constaté que la courbe subit toujours des variations. Nous n'avons pas trouvé d'explications à ce phénomène, mais nous pensons qu'il peut s'agir de mouvements derrière le mur visé par le radar durant l'expérience étant donné que le radar peut traverser les surfaces fines.

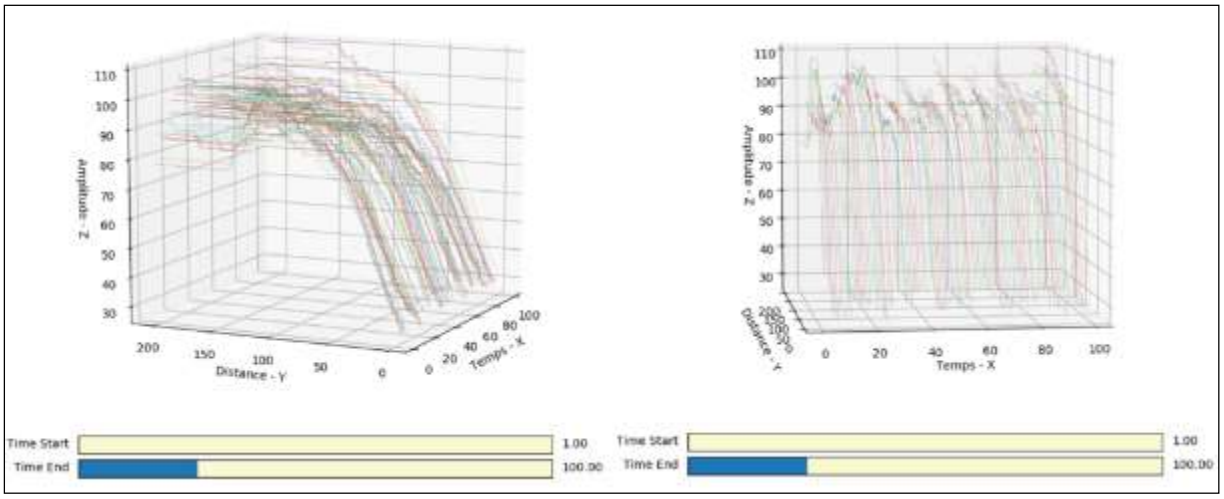

Figure 39: visualisation du test statique de 20 secondes à 130 cm du mur des données radar sous deux angles différents, image de l'auteur

7.5.2. Données lidar

Lorsque les fichiers contenant des données valides sont identifiés sur Knime à l'aide du graphique et de la méthode de prévisualisation décrits en 7.4.1, nous pouvons les utiliser pour créer une visualisation 3D du parcours sur Google Earth. À l'aide du script « raw_to_kml.py » détaillé en 7.4.2 et à l'annexe III, nous avons obtenu des résultats convaincants qui reflètent la situation réelle et qui nous permettent de valider le modèle mathématique utilisé pour placer les points lidar. La figure ci-dessous représente la visualisation finale du parcours sur Google Earth. Afin de pouvoir observer le résultat en mouvement, une vidéo avec différents angles de caméra est disponible à l'addresse https://youtu.be/pjCqYeSqizA.

Figure 40: résultat de la visualisation du parcours (ligne) du drone et des données lidar (points) sur Google Earth, image de l'auteur

Nous constatons une légère variation d'altitude pour les points lidar. Ce phénomène peut s'expliquer en prenant compte que nous sommes en terrain agricole et que les plantations ne sont pas à une hauteur régulière. Puisque chaque lidar ne pointe que dans une seule direction sur un point précis, il se peut que la plantation soit détectée à la hauteur du sol ou la hauteur de son sommet. Les points détectés les plus proches du drone nous intéressent car c'est en fonction de ceux-ci que le drone doit se positionner pour ne pas entrer en collision avec les cultures.

Nous constatons aussi que certains points sont plus éloignés et ne se trouvent pas en dessous du drone. Cela est dû au fait que les lidars sont pointés en diagonale à 45 degrés et que le « pitch » et le « roll » de l'appareil influence cette trajectoire.

7.6. Traitement pour données d'entraînement

À présent, nous allons procéder à un traitement des données pour effectuer leur classification. D'après les recommandations du « project owner » Prof. Dominique Genoud, nous avons décidé d'attribuer une classe nommée « Alarme » qui peut avoir deux valeurs différentes : « True » pour les mesures qui montrent que le drone est trop près de l'obstacle et « False » pour les mesures qui

montrent que la situation est stable et non dangereuse pour l'appareil. Le but étant que le drone puisse effectuer un freinage ou une manœuvre d'évitement si la mesure est alarmante (True).

Nous avons identifié plusieurs méthodes qui permettent une classification automatique des lignes de mesures à l'aide d'un script. Nous avons abandonné l'idée de les classer manuellement car il est difficile de représenter l'orientation du drone en lisant uniquement les valeurs des axes principaux. De plus, nous allons devoir classifier une grande quantité de donnée puisqu'un seul vol peut générer plus de mille mesures et que nous souhaitons utiliser plusieurs vols différents pour des résultats plus précis.

La méthode la plus intuitive et rapide est de prendre les distances des lidars séparément et, si l'une des mesures est plus petite qu'un certain seuil (environ 2.5 mètres dans notre cas), alors la ligne est classifiée comme étant une alarme (True). Une alternative à cette technique est de calculer la distance moyenne de tous les senseurs et de comparer le résultat au seuil pour classer l'échantillon. Ces deux méthodes sont intuitives, mais ne permettent pas d'avoir une représentation réelle du problème car nous ne prenons pas en compte l'angle du lidar par défaut ainsi que la direction et la position du drone.

Pour palier à ce problème, nous avons imaginé une technique qui prend en compte les rotations du drone et la position du lidar. Nous partons du principe que l'altitude du drone doit toujours être plus haute de 2.5 mètres par rapport à l'obstacle le plus haut sous l'appareil. En d'autres termes, si l'on dessine un vecteur qui part du drone et qui touche le sol de manière perpendiculaire, sur l'axe Z entre le sol et le drone doit être supérieure à 2.5 mètres. Il est possible d'utiliser le théorème de Pythagore dans un prisme rectangulaire droit (rectangle en 3D) étant donné que l'on connaît deux angles (angle de base + pitch et roll) ainsi que la mesure de l'un des côtés (2.5 mètres). La figure ci-dessous représente la formule entière à utiliser.

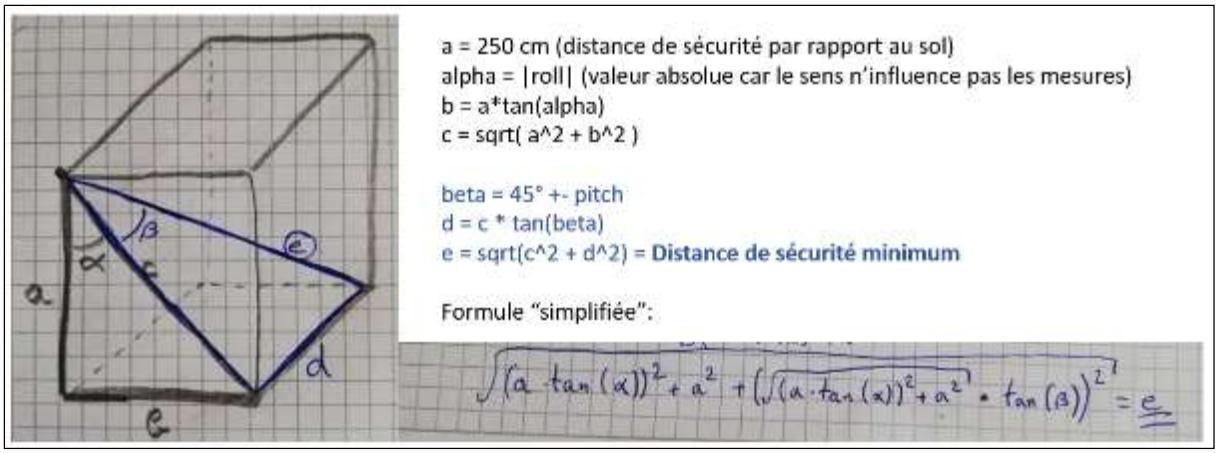

Figure 41: formule pour calculer la distance "e" (mesure lidar) minimale pour garder un seuil de 250 cm sous le drone (a) à l'aide de Pythagore, image de l'auteur

Le problème de cette méthode est que l'on calcule le roll sans prendre en compte les effets du pitch. Étant donné que la direction correcte du drone est calculée à l'aide d'une rotation d'Euler intrinsèque ZYX (chaque rotation modifie la position des axes), il est nécessaire de prendre en compte le pitch en premier puis de calculer la rotation du roll en fonction. Plus les angles de rotation sont élevés, plus le résultat va varier d'une méthode à l'autre.

Pour obtenir les résultats les plus précis possibles, nous avons réutilisé le principe du vecteur unitaire et de la rotation d'Euler. En théorie, la norme d'un vecteur unitaire est égale à 1 et chacune des 3 mesures d'un vecteur en 3D représente la distance effectuée par rapport à l'un des axes. La mesure Z correspond donc à la variation d'altitude. En calculant le vecteur unitaire en suivant la méthode décrite en 7.4.2 et en assumant que celui-ci mesure en réalité 1 cm, nous trouvons la distance Z pour 1cm. Nous souhaitons calculer le facteur de multiplication de la distance Z pour atteindre 250 cm (seuil d'altitude). Le facteur de cette multiplication sera égal à la distance minimale qui peut être mesurée par le lidar car il est égal à la norme du nouveau vecteur dont la distance Z est égale à 250 cm.

Nous avons donc mis en place un script Python nommé « raw_to_training_data_with_classification.py » disponible à l'annexe IV qui effectue la procédure décrite ci-dessus. En plus des calculs et de la classification, le script va comparer chaque ligne avec les deux suivantes et, si les trois ont une position identique, la ligne testée est supprimée pour ne garder que les lignes où le drone est en vol. Nous avons aussi décidé d'utiliser la valeur moyenne des lidars lors de la comparaison avec le seuil d'altitude de 250 centimètres. Puisque les lidars retournent la valeur zéro quand aucun obstacle n'est détecté dans la portée du senseur, il est possible qu'une seule valeur soit disponible. Dans ce cas, nous prenons uniquement la mesure positive sans prendre en compte la valeur zéro. Nous avons défini toutes les mesures où le drone se trouve théoriquement à moins de 250 centimètres des plantations comme étant des valeurs alarmantes. Pour les trouver, nous effectuons tout d'abord une rotation d'Euler sur la trajectoire du lidar central (moyenne des deux lidars latéraux) pour trouver le vecteur unitaire correspondant, puis nous divisons le seuil de 250 centimètres par la mesure « Z » du vecteur calculé pour trouver la norme du vecteur lorsque celui-ci a une valeur « Z » de 250 centimètres. Nous allons ensuite comparer cette norme avec les mesures lidar en utilisant la règle suivante : si l'une des mesures lidar est plus petite que la norme, alors la valeur est alarmante, sinon, le drone est à une altitude convenable.

Le script finit par écrire un fichier csv avec les colonnes suivantes : le temps, la mesure lidar gauche et droite, la mesure moyenne des lidars, l'altitude, l'angle roll, l'angle pitch, l'altitude calculée relative au sol et la classe alarme.

-------- Alarm == True ---------

	uid	timestamp	left	right	ALT	ROLL	PITCH	MaxAmpl	AvgDist	center	relativeAltitude	Alarm
159	160	2020-07-06-21:44:59.864	0	378	511.732	0.788327	11.576880	122.0	282.5	378.0	208.184698	True
160	161	2020-07-06-21:44:59.295	569	516	511.732	0.788327	11.576880	112.0	282.5	542.5	188.652961	True
161	162	2020-07-06-21:44:59.525	389	0	511.883	3.884520	12.540980	114.0	282.5	389.0	165.349512	True
162	163	2020-07-06-21:44:59.792	377	431	511.883	3.884520	12.540980	106.0	282.5	404.0	216.185122	True
164	165	2020-07-06-21:45:00.253	0	440	511.631	2.008770	10.964680	116.0	282.5	440.0	246.083993	True
165	166	2020-07-06-21:45:00.486	264	323	511.631	2.008770	10.964680	100.0	186.3	293.5	164.149289	True
166	167	2020-07-06-21:45:00.762	339	335	511.376	-0.723592	-8.580162	98.0	185.5	337.0	248.676638	True
167	168	2020-07-06-21:45:00.997	149	168	511.376	0.723592	-8.580162	104.0	185.5	158.5	113.196579	True
168	169	2020-07-06-21:45:01.230	163	232	511.372	0.778797	-3.828670	110.0	185.5	197.5	148.654149	True
169	170	2020-07-06-21:45:01.461	200	205	511.372	0.778797	-3.828670	114.0	282.5	202.5	152.417546	True
170	171	2020-07-06-21:45:01.718	254	249	511.951	1.266610	-1.485110	116.0	282.5	251.5	182.181251	True
171	172	2020-07-06-21:45:01.948	319	363	511.951	1.266610	-1.485110	118.0	282.5	341.0	246.984678	True

Figure 42: tableau de résultat des valeurs alarmantes après classification par rapport à la distance moyenne mesurée (au centre), image de l'auteur

Le tableau obtenu à l'aide du script de classification offre une bonne représentation de base des valeurs dangereuses. Cependant, nous souhaitons effectuer les vols au-dessus des vignes et nous rappelons que la distance sous le drone varie fortement par rapport au sol et au sommet des ceps. Nous allons donc avoir des cas de figure où il y a une alternance entre des situations alarmantes et des situations normales car la distance mesurée à un temps T peut être alarmante et la distance au temps T+1 ne l'est plus car le drone est passé au-dessus d'un cep, puis au dessus du sol (figure ci-dessous).

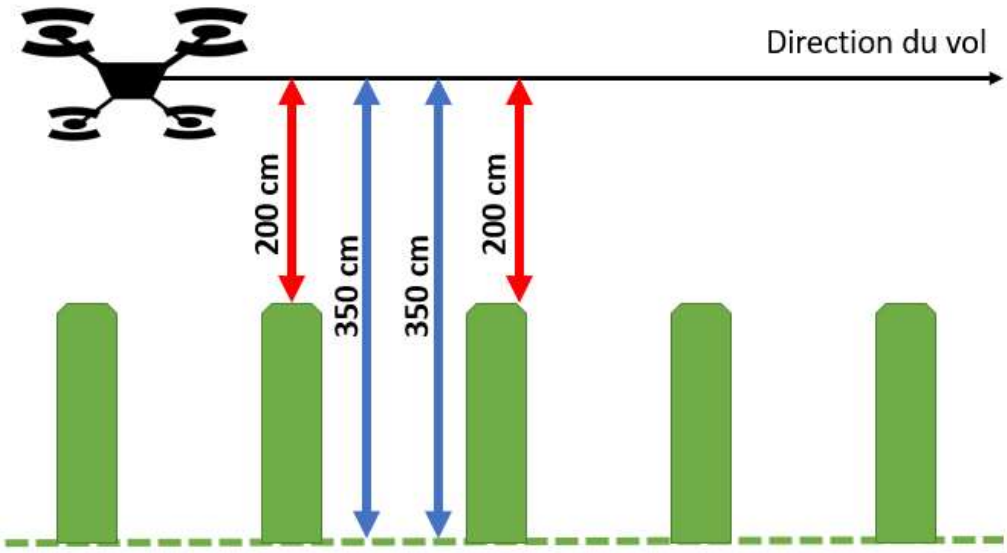

Figure 43: variation des distances mesurées sous le drone en fonction des plantations et du sol, image de l'auteur

Étant donné que le danger est toujours présent entre deux ceps, même lorsque la distance mesurée est plus grande que le seuil de 250 centimètres, nous allons aussi classer ces données comme étant alarmantes. Pour y remédier, nous avons repris les données qui viennent du script de classification et

nous avons manuellement changé les valeurs non-alarmantes qui se trouvent entre une ou plusieurs valeurs alarmantes.

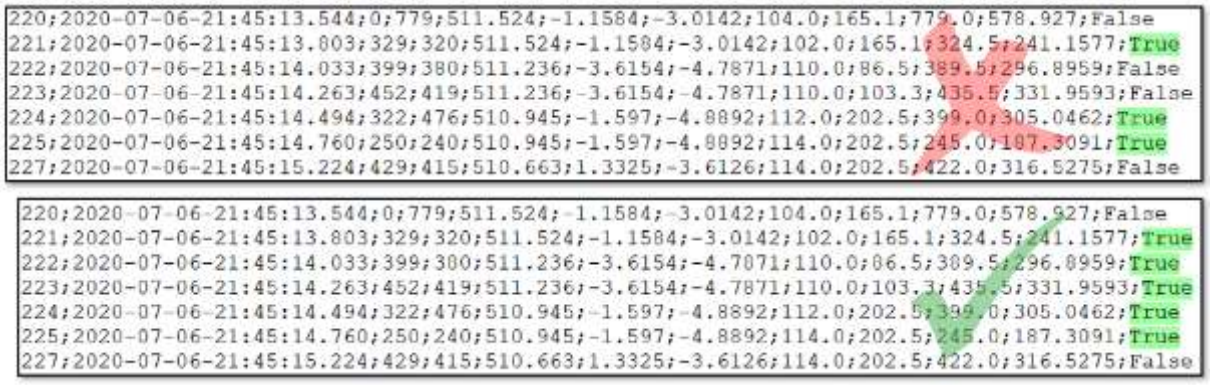

Figure 44: série temporelle de mesures avec la classe alarme "True" ou "False" avant et après modification manuelle pour résoudre de problème de variation de distance, image de l'auteur

Étant donné le peu de données disponible dans un seul vol, nous avons concaténé plusieurs vols différents en un fichier pour obtenir plus de mesures lors de la modélisation. Nous avons donc à présent un fichier csv prêt pour l'entraînement et la validation de plusieurs modèles d'apprentissage automatique.

7.7. Modélisation

Nous allons à présent tester différents modèles sur les données classifiées que nous avons à disposition. Pour cela, nous avons utilisé le programme Knime qui permet d'effectuer les dernières transformations nécessaires ainsi que la modélisation des données à l'aide de plusieurs algorithmes d'apprentissage automatique. Nous avons créé un « workflow » (flux de travail) qui suit le modèle de l'apprenant et du prédicteur (« Learner-Predictor Construct ») représenté sur la figure ci-dessous.

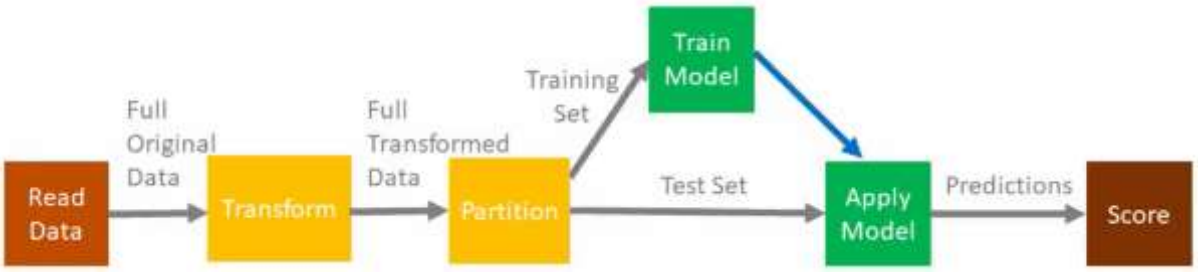

Figure 45: "workflow" général de l'apprenant et du prédicteur pour entrainer un modèle d'apprentissage automatique, image de Knime récupérée sur https://youtu.be/bKrJkdPvpeA

De plus, nous souhaitons utiliser l'historique des données mesurées et calculées avant le temps T, car celui-ci peut contenir des informations importantes pour aider à classifier les données. En effet, pour pouvoir identifier les mesures dangereuses qui se situent entre deux ceps, il est nécessaire de

regarder si les dernières valeurs sont alarmantes ou non. Nous allons donc travailler avec des séries temporelles et il est obligatoire de traiter les données de manière linéaire en fonction du temps.

À l'aide du script de classification (Annexe IV), nous avons déjà effectué la majorité des transformations nécessaires. Il reste le passage de la chaîne de caractère contenant l'heure et la date en objet « date time » pour pouvoir classer les données en fonction du temps. Nous avons aussi repris l'altitude absolue (par rapport au niveau de la mer) et nous avons soustrait l'altitude minimale de la liste pour que les valeurs soient plus proches de zéro. Cela permet à l'algorithme de travailler avec des valeurs plus petites et plus consistantes.

Pour le partitionnement des données en tableau de test et d'entraînement, nous avons identifié deux méthodes différentes. La première consiste à utiliser un unique fichier contenant les données de tous les vols réunis et de prendre un pourcentage de données d'un côté et de l'autre en prenant les données de manière chronologique sans les séparer (« take from top »). Nous avons estimé que le pourcentage de données pour l'entraînement idéal est de deux tiers (66%). La deuxième possibilité est de combiner plusieurs vols pour les données d'entraînement et de ne garder qu'un seul vol séparé pour les données de test. Cette méthode semble plus naturelle que la première et nous évitons une coupure entre les deux tables se fasse au milieu d'un vol.

Nous allons maintenant entraîner deux algorithmes de classification binaire différents ainsi que quatre séries temporelles variables. Avec ces deux paramètres combinés, nous allons tester au total huit modèles différents. Le premier algorithme choisi est le « decision tree » (arbre de décision). Celui-ci utilise un ensemble de données pour évaluer chaque variable afin de construire un arbre où les feuilles correspondent à la valeur cible et les embranchements sont la combinaison des paramètres d'entrée qui mènent à la valeur de prédiction finale (Wikipedia, 2020). En d'autres termes, il s'agit d'une suite de conditions binaires qui mènent à des embranchements et des feuilles différents en fonction du résultat. Le deuxième algorithme choisi est le « random forest » (forêt d'arbres décisionnels) et fonctionne sur la même base que le « decision tree ». La différence est que cet algorithme crée plusieurs arbres de décisions qui testent des valeurs différentes au hasard et, lorsque tous les arbres ont effectué leur prédiction, un vote est effectué et la classe avec la majorité est sélectionnée.

Nous souhaitons tester différents intervalles de temps, c'est-à-dire que nous allons comparer les résultats des prédictions lorsque l'on prend un nombre différent de données dans l'historique (voir figure ci-dessousFigure 46). Nous allons tester les intervalles suivants : les deux, trois, cinq et dix dernières valeurs de l'historique. Pour cela, nous avons utilisé dans le « workflow » Knime le nœud « lag column » qui permet de reporter sur la ligne actuelle un certain nombre de données sur les lignes précédentes. Nous avons estimé qu'il est nécessaire de reprendre toutes les mesures

disponibles ainsi que la classe de prédiction dans les lignes précédentes pour que l'algorithme puisse effectuer sa décision sur une quantité de données plus importante.

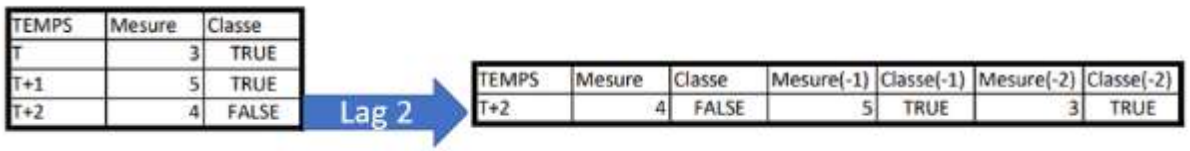

Figure 46: utilisation du "lag" pour obtenir l'historique des dernières données dans une série temporelle, image de l'auteur

En résumé, les modèles testés sont les suivants :

- « Decision tree », lag de 2, 3, 5 et 10
- « Random forest », lag de 2, 3, 5 et 10

Pour rappel, une mesure est effectuée toutes les 0.2 secondes. Un lag de 2 est équivalent à un intervalle de mesures de 0.4 secondes, un lag de 3 correspond à 0,6 secondes, un lag de 5 correspond à 1 seconde et un lag de 10 correspond à 2 secondes.

7.8. Évaluation

Après avoir entraîné et appliqué les modèles comme décrit à la section ci-dessus, nous avons effectué une évaluation des résultats des différentes prédictions. Nous rappelons que l'objectif de cette évaluation est d'identifier l'algorithme qui a obtenu les meilleurs résultats, c'est-à-dire la prédiction qui a fait le moins d'erreur. Dans notre cas de figure, puisqu'il est primordial d'éviter tout accident, la priorité est de trouver les situations alarmantes, c'est-à-dire les valeurs où la classe alarme est égale à « True ». En effet, si la prédiction ne trouve pas une situation alarmante et n'effectue pas de manœuvre d'évitement, le drone risque d'entrer en collision avec un obstacle. À l'inverse, si la prédiction classe une situation sans danger comme étant alarmante, le drone risque simplement de s'arrêter ou d'augmenter son altitude.

Nous avons tout d'abord effectué une « ROC curve » (courbe ROC, ou fonction d'efficacité du récepteur) pour comparer le niveau de confiance de chaque algorithme. Ensuite, nous allons calculer la zone sous la courbe (« area under curve ») et l'erreur type (« standard error ») de chaque algorithme pour identifier le plus précis d'entre eux. Nous allons aussi analyser la précision et la matrice de confusion du meilleur algorithme (« confusion matrix »). Finalement, nous allons analyser manuellement les résultats de la meilleure prédiction pour identifier les lignes où la prédiction est incorrecte ainsi que le détail des valeurs sélectionnées et le chemin effectué par l'algorithme pour prendre la décision finale.

7.8.1. ROC Curve

Nous avons complété le workflow Knime afin de tracer le graphique « ROC curve » avec les taux de prédiction (confiance) de « true » positifs de chaque algorithme afin de comparer leur efficacité. Pour cela, après avoir entraîné et testé le modèle, nous créons une table contenant la probabilité de la prédiction de chaque mesure pour tous les algorithmes. Nous avons utilisé le nœud Knime « ROC curve » en sélectionnant la classe alarme et la valeur positive comme étant « True » puisque c'est la valeur que nous devons trouver en priorité et nous avons donné en paramètre du graphique les taux de confiance récupérés précédemment.

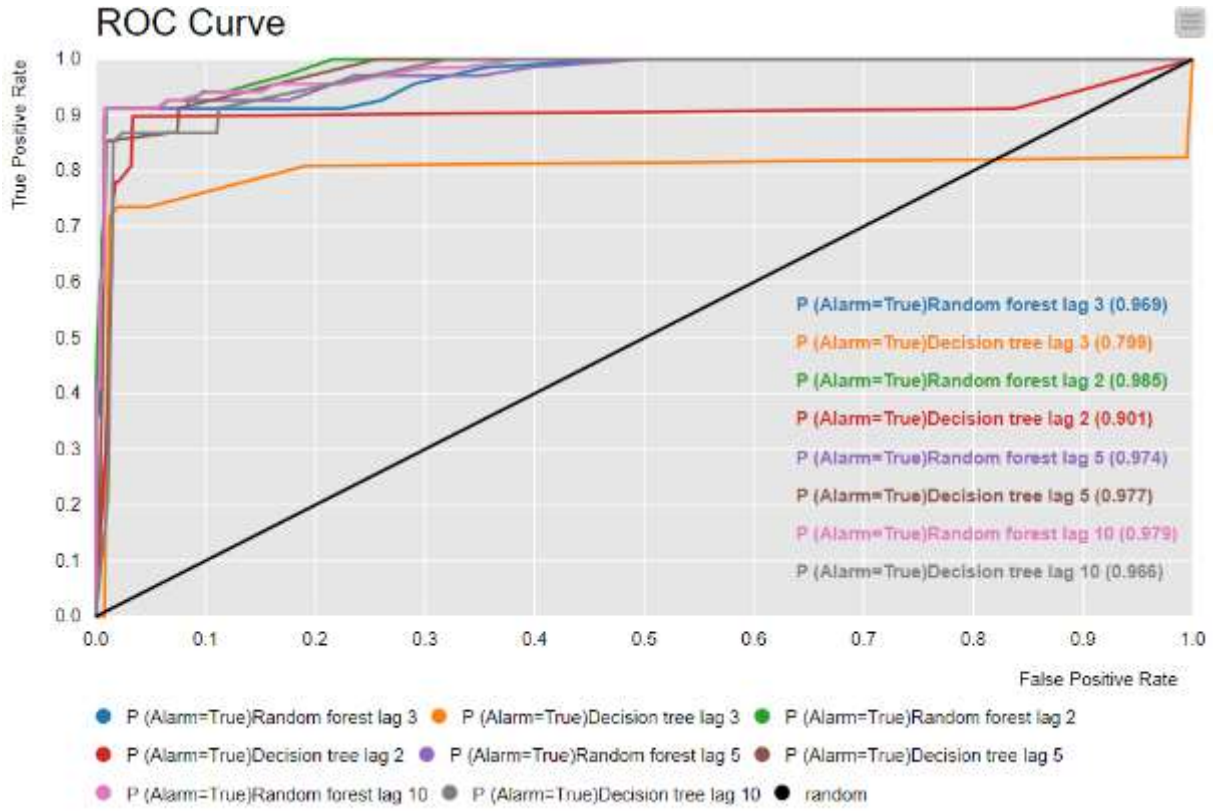

Figure 47: "ROC curve" des résultats des prédictions de chaque algorithme en fonction du taux de confiance, image de l'auteur

Nous constatons que le meilleur résultat des prédictions est la courbe du « random forest » avec un lag de 2 car c'est la courbe qui se rapproche le plus du point (0,1) du graphique et la zone sous la courbe est la plus grande (0.985).

7.8.2. Zone sous la courbe et erreur type

Maintenant que nous avons tracé la courbe ROC et que nous avons obtenu la zone sous la courbe (AUC), nous allons calculer l'erreur type (SE) afin d'avoir une idée de la dispersion des données dans notre échantillon de test. Plus l'erreur type est petite, plus les données sont unifiées et les clusters sont facilement identifiables. Pour cela, nous avons utilisé la méthode de Hanley & McNeil (1982), une formule qui permet de calculer l'erreur type en fonction de l'AUC. Pour finir, nous avons ajouté le nom des méthodes pour chaque résultat et nous les avons classé en commençant par l'AUC le plus élevé.

Method	Area Under Curve	Standard Error
Random Forest lag 2	98.45%	0.37%
Random Forest lag 10	97.91%	0.45%
Decision Tree lag 5	97.67%	0.48%
Random Forest lag 5	97.36%	0.52%
Random Forest lag 3	96.86%	0.58%
Decision Tree lag 10	96.58%	0.62%
Decision Tree lag 2	90.24%	1.31%
Decision Tree lag 3	79.84%	2.22%

Figure 48: tableau de résultat de l'AUC et de l'erreur type des différentes méthodes de prédiction utilisées, image de l'auteur

D'après les résultats, l'algorithme « random forest » avec un intervalle de deux est la combinaison la plus efficace dans la prédiction par rapport à l'AUC et l'erreur type. Cela signifie que cet algorithme est le plus apte à classer les données en fonction de leur taux de confiance sur la prédiction de la classe « true ».

7.8.3. Matrice de confusion

Maintenant que nous avons identifié l'algorithme le plus consistant par rapport à la « ROC curve », nous allons analyser la précision et la matrice de confusion de ces méthodes de prédiction afin d'identifier le pourcentage de bonnes prédictions pour les valeurs true et false. Pour rappel, le plus important est d'obtenir un maximum de valeurs dans la catégorie des vrais positifs pour éviter tout accident.

		Classe estimée (prédiction)	
		TRUE	FALSE
Classe réelle	TRUE	Situation alarmante identifiée correctement **vrais positifs**	Situation alarmante non identifiée **faux négatifs**
	FALSE	Situation non alarmante mal prédite **faux positifs**	Situation non alarmante prédite correctement **vrais négatifs**

Figure 49: matrice de confusion générale pour la classe alarme, image de l'auteur

Afin de résumer les matrices de confusion de chaque algorithme, nous avons utilisé le nœud Knime « Scorer » afin de récupérer les statistiques de chacun d'eux. Dans ces statistiques, nous avons extrait le pourcentage de vrais positifs et de vrais négatifs afin de les classer et de les afficher en commençant par le modèle qui a le meilleur score de vrais positifs. Les résultats figurent sur le tableau ci-dessous.

Method	% true positive	% true negative
Lag 3 Random Forest	91.04%	99.07%
lag 5 Random Forest	90.91%	98.94%
lag 10 Random Forest	90.91%	98.92%
lag 2 Random Forest	89.55%	98.94%
lag 5 Decision Tree	89.23%	98.67%
Lag 3 Decision Tree	83.05%	97.51%
lag 10 Decision Tree	82.86%	98.65%
lag 2 Decision Tree	77.94%	98.02%

Figure 50: tableau des pourcentages de réussite des algorithmes de prédiction en fonction de la classe (true ou false), image de l'auteur

Nous constatons que, cette fois-ci, le modèle qui utilise le « random forest » avec un intervalle de trois mesures obtient le meilleur résultat de vrais positifs et de vrais négatifs. La meilleure méthode par rapport à l'AUC et l'erreur type se retrouve uniquement à la quatrième place (« lag 2 random forest ») car, même si ce dernier a obtenu des taux de confiances plus élevés, il a détecté moins de situations alarmantes que le meilleur algorithme.

Nous pouvons aussi constater que l'algorithme « random forest » est plus performant que le « decision tree » de manière générale, car aucune méthode utilisant le « decision tree » est mieux classée que les méthodes utilisant le « random forest ».

Nous allons donc observer la matrice de confusion (ci-dessous) de la meilleure méthode en détail afin d'obtenir le nombre exact de mesures alarmantes que l'algorithme n'a pas détecté.

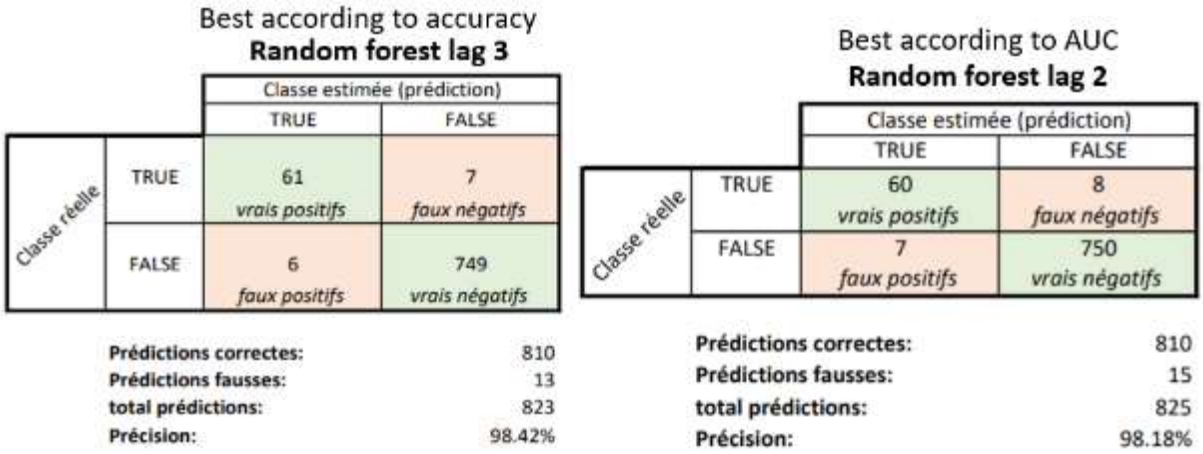

Figure 51: matrice de confusion pour des deux meilleures méthodes en fonction de l'AUC et de la précision, image de l'auteur

Sur la matrice de confusion du « random forest lag 3 » ci-dessus, nous constatons que les données d'entraînement contiennent un total de 68 valeurs alarmantes(true) et que l'algorithme s'est trompé 7 fois dans la classification de ces valeurs. Pour ce qui est des valeurs non-alarmantes, seulement 6 valeurs sur 755 ont été mal prédites. Étant donné que nous avons travaillé sur un set de données déséquilibré avec une grande majorité de valeurs non-alarmantes (alarme = false), la précision est une mesure trompeuse car le classificateur aura tendance à prédire la classe majoritaire en priorité.

7.8.4. Analyse manuelle

Maintenant que nous savons le nombre de fois où l'algorithme s'est trompé de prédiction, nous allons observer manuellement les prédictions par rapport aux classes réelles et identifier les lignes qui n'ont pas été classées correctement. Nous constatons que les algorithmes ont mal identifié la classe des valeurs alarmantes (true) dans deux cas de figure différents. Les premières erreurs apparaissent lorsqu'une seule valeur alarmante est isolée entre des séries de valeurs non-alarmantes. Dans cette situation, l'algorithme a tendance à ne pas trouver cette valeur ou alors à la trouver en retard à la mesure suivante. Le deuxième type d'erreur intervient lorsqu'une série de plusieurs valeurs est alarmante. Dans ce cas-là, l'algorithme ne détecte à aucune reprise la première valeur alarmante de la série et les valeurs alarmantes suivantes sont classées correctement. Cela signifie que, excepté quelques valeurs alarmantes isolées qui ne sont pas identifiées, l'algorithme a réussi dans l'ensemble à détecter toutes les situations dangereuses pour le drone avec un retard de 0.2 secondes. En plus de détecter les valeurs alarmantes avec un retard d'une mesure, l'algorithme classe toujours la première situation non-alarmante comme étant toujours dangereuse. La plupart des faux positifs viennent de ce phénomène, mais dans notre cas, ce phénomène n'est pas dangereux car il s'agit d'une mesure de sécurité supplémentaire plutôt que d'une mauvaise prédiction.

	Lag columns				
Timestamp	Alarm(-1)	Alarm(-2)	Alarm(-3)	Alarm	Prediction (Alarm)
T	FALSE	FALSE	FALSE	FALSE	FALSE
T+1	FALSE	FALSE	FALSE	FALSE	FALSE
T+2	FALSE	FALSE	FALSE	TRUE	FALSE
T+3	TRUE	FALSE	FALSE	TRUE	TRUE
T+4	TRUE	TRUE	FALSE	TRUE	TRUE
T+5	TRUE	TRUE	TRUE	TRUE	TRUE
T+6	TRUE	TRUE	TRUE	TRUE	TRUE
T+7	TRUE	TRUE	TRUE	FALSE	TRUE
T+8	FALSE	TRUE	TRUE	FALSE	FALSE
T+9	FALSE	FALSE	TRUE	FALSE	FALSE
T+10	FALSE	FALSE	FALSE	FALSE	FALSE

Legend:
Correct
Incorrect

Figure 52: extrait du résultat de la classification décalée d'une mesure durant un intervalle de temps alarmant, image de l'auteur

7.8.5. Analyse de l'algorithme de prédiction

Nous allons maintenant analyser en détail les paramètres les plus souvent utilisés par l'algorithme « random forest » dans sa démarche de classification. Nous avons fixé le nombre d'arbres différents à cent exemplaires. Cela signifie que chaque ligne va être classée cent fois avec des embranchements différents, c'est-à-dire des attributs et des conditions différents. Le tableau ci-dessous représente les attributs qui ont le plus souvent été testés sur les trois premiers niveaux de chacun des arbres.

Attribute	Level 0	Level 1	Level 2	Total occurences
Alarm(-1)	11	24	27	62
Alarm(-2)	16	15	11	42
Alarm(-3)	19	9	9	37
ALT(-2)	9	15	9	33
ALT(-3)	10	9	12	31
relativeAltitude(-2)	5	9	12	26
ALT(-1)	9	10	7	26
center(-2)	1	6	16	23
MaxAmpl(-3)	1	5	14	20
center(-1)	1	6	13	20
relativeAltitude(-1)	0	8	12	20
relativeAltitude(-3)	2	3	12	17
ROLL(-1)	0	3	13	16
right(-1)	1	9	5	15
right(-3)	2	1	11	14

Figure 53: classement des 15 attributs les plus utilisés sur les trois premiers niveaux des arbres pour l'algorithme « random forest » avec un intervalle de 3 mesures, image de l'auteur

Nous constatons que l'attribut le plus utilisé est la valeur de l'alarme des trois dernières mesures suivies des altitudes relatives et absolues ainsi que des mesures lidar (« center »). Ces résultats

démontrent que l'algorithme a accordé plus d'importance à l'historique de la classe alarme pour effectuer ses décisions. Cela explique potentiellement le fait que la première valeur « true » d'une série alarmante n'est pas détectée car l'algorithme a pris sa décision en regardant l'historique de l'alarme plutôt que l'historique de l'altitude et de la mesure lidar. Comme la première valeur alarmante n'a pas d'historique d'alarmes, l'algorithme estime que la valeur suivante n'est pas alarmante, c'est seulement lorsqu'il observe une valeur alarmante dans l'historique qu'il va classer la nouvelle valeur comme étant alarmante.

7.9. Remarques importantes

Durant cette étude, nous avons effectué un nombre de vols limité pour récolter des données d'entraînement et une partie de ces données était corrompue, donc inutilisable. Cela signifie que nous avons travaillé avec un volume de donnée très fin. Pour entraîner un modèle d'apprentissage automatique avec précision, il est primordial d'utiliser un set de données volumineux et complet. Nous ne pouvons donc pas garantir que les résultats sont précis et, pour obtenir un modèle fiable, il serait nécessaire d'effectuer un plus grand nombre de vols. Lors de la modélisation, nous avons travaillé avec un tableau contenant 1'834 lignes déjà classées.

Le deuxième problème que nous avons rencontré concerne la répartition des données dans la classe alarme. En effet, une grande majorité des données de notre set d'entraînement est classée comme étant non-alarmante (alarm = false). Sur un total de 1'834 lignes, 1'621 sont classées « false » (88.39%) et 213 sont classées « true » (11.61%). Cela aura un impact négatif sur la prise de décision du modèle car il sera influencé par cette répartition. L'algorithme aura tendance à classer les nouvelles valeurs comme étant « false » puisqu'il a appris et il sait que la plupart des valeurs précédentes étaient « false ». Pour résoudre ce problème, une possibilité est de récolter plus de données en ajoutant volontairement plus de valeurs alarmantes (« true »). La deuxième solution envisagée est d'utiliser la technique du « bootstrap », un procédé qui va imiter les valeurs d'une certaine classe, par exemple « alarme = true », à l'aide de moyennes et d'écarts types afin de peupler le tableau de données avec des données de remplacement (Brownlee, 2018).

Durant le temps imparti de l'étude, nous avons eu l'occasion d'effectuer une seule itération de modélisation des données. Afin d'obtenir de meilleurs résultats, il est nécessaire d'effectuer plusieurs itérations en effectuant plusieurs modifications sur les données et les algorithmes testés. Cela permet entre autres d'effectuer une optimisation des paramètres (« tuning ») qui contrôlent le processus d'apprentissage automatique des algorithmes. À l'aide de la technique du « tuning », il est possible, par exemple, de modifier le seuil de prédiction de la classe alarme. Par défaut, l'algorithme va attribuer un pourcentage de chance que la valeur soit classée comme « true » et, si le pourcentage est égal ou plus élevé que 50 %, alors la valeur est classée comme « true », sinon elle est classée comme « false ». En modifiant ce pourcentage, il est possible d'augmenter ou de diminuer la

probabilité que les valeurs soient classées comme « true ». Par exemple, avec un seuil de prédiction de 30 % toutes les valeurs qui ont un pourcentage de prédiction de la classe « true » égal ou plus élevé que 30 % seront classées comme « true ». Dans notre cas de figure, puisqu'il est primordial de détecter les valeurs alarmantes (« true »), nous avons intérêt à diminuer ce pourcentage afin d'identifier toutes les situations dangereuses même si cela peut augmenter la probabilité d'obtenir des faux négatifs.

8. Conclusion

8.1. Résultats obtenus

Dans un premier temps, nous avons obtenu des résultats concluants de visualisation dans l'espace des données géographiques du drone et des senseurs lidars. Nous avons démontré qu'il est possible d'automatiser la transformation des données brutes récupérées par le drone et les lidars en vol en un fichier de données géographiques à l'aide d'un script « Python ». La visualisation de ces données nous a permis de comprendre et d'analyser l'environnement et les dangers autour d'un drone en milieu agricole. Cela nous a permis de mieux appréhender le problème de la détection d'obstacles. De plus, nous pouvons affirmer que le format standard « Keyhold Markup Language » (KML) est le plus adapté à notre cas de figure et l'intégration de ceux-ci au logiciel gratuit « Google Earth Pro » est irréprochable.

Ensuite, nous avons testé différents modèles d'apprentissage automatique afin d'identifier l'algorithme de prédiction le plus adapté à une reconnaissance d'obstacles à basse altitude avec des senseurs. D'après les résultats des prédictions, la technique du « random forest » avec un intervalle de deux mesures est le modèle le plus sûr d'après la courbe ROC avec une AUC de 0.9845. En théorie, cet algorithme a obtenu un meilleur score de prédiction, car étant donné que le set de donnée d'entraînement est fortement déséquilibré, la mesure AUC a plus de sens que la précision, car il montre que l'algorithme est plus confiant dans ses décisions. En pratique, puisque la détection de valeurs alarmantes est impérative (vrais positifs), nous avons identifié l'algorithme « random forest » avec un intervalle de trois mesures comme ayant la meilleure précision sur la prédiction des vrais positifs avec une précision de 91.04 %. Ce résultat n'est pas idéal, mais si l'on prend en compte que l'algorithme a détecté toutes les situations qui contiennent une suite de valeurs alarmantes avec un retard d'une mesure, alors l'algorithme a véritablement détecté les situations où le drone est trop proche des plantations avec un délai de 0.2 secondes.

8.2. Recommandations

Les recherches et les résultats ont révélé qu'un système d'apprentissage automatique sur des données récupérées par des senseurs permet d'améliorer la reconnaissance d'obstacles embarquée sur un drone agricole. Néanmoins, nous avons constaté que le système développé durant cette étude nécessite des recherches et des tests supplémentaires si l'on souhaite l'utiliser sur les drones en production. Notre première recommandation est d'effectuer des tests avec des lidars rotatifs 2D ou 3D afin d'obtenir un plus grand nombre de mesures afin d'identifier plus facilement le haut des ceps de vigne. Le lidar 1D que nous avons testé aura tendance à ne pas identifier le sommet des ceps, car celui-ci peut potentiellement passer sur la trajectoire du lidar entre deux mesures. De plus, nous recommandons d'effectuer plus d'itérations de modélisation après avoir été récolté un plus grand

nombre de données de vols. Il est aussi important d'ajouter plus de valeurs alarmantes afin d'équilibrer la répartition des valeurs de la classe alarme.

8.3. Perspectives de recherches

D'après les résultats obtenus, nous estimons qu'il est convenable de continuer les recherches afin d'améliorer la reconnaissance d'obstacles proposée dans ce document. Il nous semble nécessaire d'effectuer à nouveau le cycle de la méthode CRISP-DM en reprenant à l'étape de la préparation et de la transformation des données. Lors de ces nouvelles itérations, il est pertinent de travailler sur le « tuning » des données, c'est-à-dire de modifier les paramètres qui gouvernent les algorithmes. Puisque nous avons constaté que certains algorithmes ont un taux de confiance élevé en fonction de l'AUC, nous estimons que des recherches pour trouver le seuil (« treshold ») de prédiction optimal permettraient d'améliorer la détection des valeurs alarmantes.

8.4. Conclusion personnelle

Ce travail m'a permis de découvrir le domaine des drones agricoles avec toutes ses particularités techniques. Étant donné mon niveau de connaissance limité au début de l'étude sur le fonctionnement d'un drone et des capteurs radars et lidars, j'ai eu l'occasion de développer mes compétences au fil de l'avancement de cette étude. De plus, cela m'a donné la possibilité d'appliquer mes connaissances de base en apprentissage automatique dans une situation avec un objectif réel et concret. Lors du choix de cette thèse de Bachelor, je ne pensais pas qu'un tel sujet pouvait avoir tant de particularités et de défis à relever. Il m'a parfois fallu une semaine entière pour comprendre certains mécanismes utilisés, notamment la compréhension des matrices de rotation avec les angles d'Euler. J'ai aussi constaté que le débogage peut prendre beaucoup de temps et doit être effectué fréquemment. Il m'a fallu faire preuve d'une grande patience et de beaucoup de persévérance afin de comprendre le problème et pour atteindre les objectifs fixés.

Arrivé au terme de la rédaction de ce document, je suis de manière générale satisfait du travail accompli. Étant donné les limitations matérielles et le temps imposé, j'estime que nous avons obtenu des résultats concluants et que cette étude est une bonne base pour l'intégration d'un système de reconnaissance d'obstacles à basse altitude pour drone.

Références

ABot. (2020). *Drone agricole : la technologie au service de l'agriculture*. Retrieved from ABot: https://www.abot.fr/agriculture-2779

Acconeer AB. (2019). *Obstacle detection*. Retrieved from acconeer-python-exploration: https://acconeer-python-exploration.readthedocs.io/en/latest/processing/obstacle.html

Acconeer AB. (2019, December 1). Radar sensor introduction.

Alaimo, A., Artale, V., Milazzo, C., & Ricciardello, A. (2013). Comparison between Euler and Quaternion parametrization in UAV dynamics. *AIP Conference Proceedings*, 1228-1231.

Arbellay, O. (2019, 07 30). Ajout de résultats de machine learning aux drones agricoles en utilisant un «Companion Computer». Sierre, Valais, Suisse.

ArcGIS. (2020, May). *About ArcGIS Pro*. Retrieved from ArcGIS Pro: https://pro.arcgis.com/en/pro-app/get-started/get-started.htm

Asvadi, A., Premebida, C., Peixoto, P., & Nunes, U. (2016). 3D Lidar-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes. *Robotics and autonomous Systems*, 299-311.

Bošnjak, Z., Grljević, O., & Bošnjak, S. (2014). CRISP-DM as a Framework for Discovering Knowledge. *BULETINUL STIINTIFIC al Universitatii "Politehnica" din Timisoara, ROMANIA*, 1-2.

Brownlee, J. (2018, May 25). *A Gentle Introduction to the Bootstrap Method*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/a-gentle-introduction-to-the-bootstrap-method/

Carrio, A., Sampedro, C., Rodriguez-Ramos, A., & Campoy, P. (2017). A Review of Deep Learning Methods and Applications for Unmanned Aerial Vehicles. *Hindawi*.

Champman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000, August). *CRISP-DM 1.0*. Retrieved from TMA, LCC: https://www.the-modeling-agency.com/crisp-dm.pdf

Chen, K. Y., Janz, K. F., Zhu, W., & Brychta, R. J. (2012). Redefining the Roles of Sensors in Objective Physical Activity Monitoring. *Medicine and Science in Sports and Exercise*, 13-12.

Cruz, G. C., & Encarnação, P. M. (2012). Obstacle Avoidance for Unmanned Aerial Vehicles. *Journal of Intelligent and Robotic Systems*.

DJI, A. (2018, Janvier 12). DJI Agras MG-1S Radar Module.

DJI, S. (2016, 06). AGRAS MG-1S.

Escobar, F., Hunter, G., Bishop, I., & Zerger, A. (1998). *Geogra*. Retrieved from Geogra: http://www.geogra.uah.es/patxi/gisweb/GISModule/

Expert System Team. (2017, March). *What is Machine Learning? A definition*. Retrieved from Expert System: https://expertsystem.com/machine-learning-definition/

Food and Agriculture Organization of the United Nations. (2018). *E-AGRICULTURE IN ACTION: DRONES FOR AGRICULTURE*. Bangkok: Food and Agriculture Organization of the United Nations and International Telecommunication Union.

Fraden, J. (2010). *Handbook of Modern Sensors*. New York: Springer.

Fraga-Lamas, P., Ramos, L., Mondéjar-Guerra, V., & Fernández-Caramés, T. M. (2019). A Review on IoT Deep Learning UAV Systems for Autonomous Obstacle Detection and Collision Avoidance. *Remote Sensing*.

Free Software Foundation. (1991, June). *GNU General Public License, version 2*. Retrieved from GNU Operating System: https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html

Gageik, N., Benz, P., & Montenegro, S. (2015). Obstacle Detection and Collision Avoidance for a UAV With Complementary Low-Cost Sensors. *IEEE Access*, 599-609.

Gatziolis, D., & Andersen, H.-E. (2008). *A Guide to LIDAR Data Acquisition and Processing for the Forests of the Pacific Northwest*. Portland: U.S. Department of Agriculture.

Goodchild, M., & Kemp, K. (1990). *NCGIA Core Curriculum in GIS*. Retrieved from NCGIA: https://ibis.geog.ubc.ca/courses/klink/gis.notes/ncgia/toc.html

Google. (2018, October 31). *KML Tutorial*. Retrieved from Google Developpers: https://developers.google.com/kml/documentation/kml_tut

Google. (2020). *Google Earth Enterprise - Open Source*. Retrieved from Github: https://github.com/google/earthenterprise

Hanley, J., & Mcneil, B. (1982). The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve. *Radiology*, 29-36.

Kannan, V., Jhajharia, S., & Verma, S. (2014). Agile vs waterfall: A Comparative Analysis. *International Journal of Science*, 1-2.

Kardasz, P., Doskocz, J., Hejduk, M., Wiejkut, P., & Zarzycki, H. (2016). Drones and Possibilities of Their Using. *Journal of Civil & Environmental Engineering*.

LaValle, S. M. (2012, April 20). *Yaw, pitch and roll rotations*. Retrieved from Planning Algorithms: http://planning.cs.uiuc.edu/node102.html

Mcgrath, M. J., & Scanaill, C. N. (2013). Sensing and Sensor Fundamentals. In M. J. Mcgrath, & C. N. Scanaill, *Sensor technologies: Healthcare, wellness, and environmental applications* (pp. 15-50).

NASA. (2015, May 5). *Aircraft Rotations Body Axes*. Retrieved from NASA: https://www.grc.nasa.gov/WWW/K-12/airplane/rotations.html

Open Geospatial Consortium. (2015, August 4). *OGC KML 2.3*. Retrieved from Open Geospatial Consortium: http://docs.opengeospatial.org/is/12-007r2/12-007r2.html#1

PrecisionHawk. (2020). *Advanced Sensors and Data Collection*. Retrieved from PrecisionHawk: https://www.precisionhawk.com/sensors/advanced-sensors-and-data-collection/lidar

QGIS project. (2020, June 24). *A Gentle Introduction to GIS*. Retrieved from QGIS Documentation: https://docs.qgis.org/3.10/en/docs/gentle_gis_introduction/

QGIS project. (2020, June 24). *QGIS User Guide*. Retrieved from QGIS Documentation: https://docs.qgis.org/3.10/en/docs/user_manual/index.html

Rodríguez-Puerta, F., Alonso Ponce, R., Pérez-Rodríguez, F., Águeda, B., Martín-García, S., Martínez-Rodrigo, R., & Lizarralde, I. (2020). Comparison of Machine Learning Algorithms for Wildland-Urban Interface Fuelbreak Planning Integrating ALS and UAV-Borne LiDAR Data and Multispectral Images. *Drones*.

Rouse, M. (2005). *radar (radio detection and ranging)*. Retrieved from TechTarget: https://searchmobilecomputing.techtarget.com/definition/radar

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. New York: Cambridhe University Press.

Simon, P. J., Lisa, W. M., & Bryan, C. (2014, May 21). LiDAR Applications. Silver Spring, Maryland, USA.

Skolnik, M. I. (1990). *Radar Handbook*. Boston: McGraw-Hill.

Smola, A., & Vishwanathan, S. V. (2008). *Introduction to Machine Learning*. Cambridge: Cambridge University Press.

The Apache Software Foundation. (2004, January). *APACHE LICENSE, VERSION 2.0*. Retrieved from THE APACHE SOFTWARE FOUNDATION: https://www.apache.org/licenses/LICENSE-2.0

Wandinger, U. (2005). *Introduction to Lidar*. Springer.

Wang, L., Lan, Y., Zhang, Y., Zhang, H., Tahir, M. N., Ou, S., . . . Chen, P. (2019). Applications and Prospects of Agricultural Unmanned Aerial Vehicle Obstacle Avoidance Technology in China. *Sensors*.

Wikipedia. (2020 , June 5). *Arbre de décision (apprentissage)*. Retrieved from Wikipedia: https://fr.wikipedia.org/wiki/Arbre_de_d%C3%A9cision_(apprentissage)

Xie, D., Xu, Y., & Wang, R. (2019). Obstacle detection and tracking method for autonomous vehicle based on three-dimensional LiDAR. *International Journal of Advanced Robotic Systems*.

Zhengyu, P., & Changzhi, L. (2019, March 6). Portable Microwave Radar Systems for Short-Range Localization and Life Tracking: A Review. Lubbock, Texas, USA.

I. Annexe - Script Python de visualisation des données Radar

```
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
from matplotlib.widgets import Slider
from Tkinter import Tk
from tkinter.filedialog import askopenfilename
import numpy as np


def get_file_name():
    Tk().withdraw()  # we don't want a full GUI, so keep the root window
from appearing
    filepath = askopenfilename()  # show an "Open" dialog box and return
the path to the selected file
    return filepath


file_data_path = get_file_name()

df = pd.read_csv(file_data_path, delimiter=';', quotechar="\"")
df = df["radar"].str[1:-1]  # delete brackets at the beginning and end of
data
first_val = 1
last_val = len(df)

fig = plt.figure(figsize=(8, 7))
ax = fig.add_subplot(111, projection='3d')

axcolor = 'lightgoldenrodyellow'
axtime_start = plt.axes([0.15, 0.05, 0.7, 0.03], facecolor=axcolor)
stime_start = Slider(axtime_start, 'Time Start', 1, last_val,
valinit=first_val, valstep=1)
axtime_end = plt.axes([0.15, 0.01, 0.7, 0.03], facecolor=axcolor)
stime_end = Slider(axtime_end, 'Time End', 1, last_val, valinit=last_val,
valstep=1)


def plot_line(points, time_tab):
    distance_tab = np.arange(0., float(len(points)), 1.)
    ax.plot(time_tab, distance_tab, points, linewidth=0.25)


def plot_graph(filter_from, filter_to):
    ax.clear()  # reset graph
    time = 1.
    for r in df.iloc[int(filter_from - 1):int(filter_to - 1)]:
        points_tab = []
        time_tab = []

        for row in r.split('\n'):

            for col in row.split():
```

```
                # for all radar values in one measurement
                points_tab.append(float(col))
                time_tab.append(time)

        plot_line(points_tab, time_tab)
        time += 1.
    init_graph()


def init_graph():
    ax.set_xlabel("Temps - X")
    ax.set_ylabel("Distance - Y")
    ax.set_zlabel("Amplitude - Z")


def update(val):
    # to update sliders, check that the last value is higher than the first
    if stime_start.val <= last_val and stime_end.val <= last_val:
        if stime_start.val > stime_end.val:
            stime_end.set_val(stime_start.val)
        plot_graph(stime_start.val, stime_end.val)


init_graph()

# add listeners for sliders
stime_start.on_changed(update)
stime_end.on_changed(update)

# default display shows all values
plot_graph(first_val, last_val)
plt.show()
```

II. Annexe - Script Python de classification pour les données d'entraînement

```
from Tkinter import Tk
from tkinter.filedialog import askopenfilename
import pandas as pd
from datetime import datetime
from scipy.spatial.transform import Rotation as R

# allow a bigger size when printing a df
pd.options.display.width = 0


def get_file_name():
    Tk().withdraw()  # we don't want a full GUI, so keep the root window
from appearing
    filepath = askopenfilename()  # show an "Open" dialog box and return
the path to the selected file
    return filepath


filename = get_file_name()

df = pd.read_csv(filename, delimiter=';', quotechar="\"")

# Retrieve coordinates from flight header
coord_data = pd.DataFrame(df["coord"].str.split(', ', expand=True).values,
                          columns=["uidcoord", "timestampcoord", "LA",
"LON", "ALT", "ROLL", "PITCH", "YAW"])

df["ALT"] = pd.to_numeric(coord_data["ALT"].str[1:-1])
df["LA"] = pd.to_numeric(coord_data["LA"].str[1:-1])
df["LON"] = pd.to_numeric(coord_data["LON"].str[1:-1])
df["ROLL"] = pd.to_numeric(coord_data["ROLL"].str[1:-1])
df["PITCH"] = pd.to_numeric(coord_data["PITCH"].str[1:-1])

df = df.drop(['uid', 'cnt', 'radar', 'coord'], axis=1)

# Delete all rows that are similar to the next two rows (comparing
altitude, lat and lon)
# to filter data when drone is not flying
index_table = []  # stores the index of rows to delete

for index, row in df.iterrows():
    try:
        second_row = df.iloc[index + 1]
        third_row = df.iloc[index + 2]
        if (row.ALT == second_row.ALT
                and row.ALT == third_row.ALT
                and row.LA == second_row.LA
                and row.LA == third_row.LA
                and row.LON == second_row.LON
                and row.LON == third_row.LON):
            print ("row " + str(index) + " deleted")
            index_table.append(index)
```

```
    except IndexError:
        print row
        index_table.append(index)

df.drop(index_table, inplace=True)

# Delete unused rows
df = df.drop(['LA', 'LON'], axis=1)


def get_middle_lidar_point(current_row):
    # if both lidar have values, then average, if not, we only take the non
zero value
    if current_row["left"] <= 0 and current_row["right"] <= 0:
        return 0
    elif current_row["left"] <= 0:
        return current_row["right"]
    elif current_row["right"] <= 0:
        return current_row["left"]
    else:
        return (current_row["right"] + current_row["left"]) / 2.


df["center"] = df.apply(get_middle_lidar_point, axis=1)

# variables to calculate alarm
base_unit_vector_center = [0.70710678, 0, 0.70710678]
alt_treshold = 250  # threshold in cm


def calculate_alarm(current_row):
    yaw_pitch_roll = [0, current_row["PITCH"], current_row["ROLL"]]  # yaw
always zero as it doesn't matter
    unit_vector = R.from_euler('ZYX', yaw_pitch_roll,
degrees=True).apply(base_unit_vector_center)
    # factor = treshold after drone rotation, equals the number of times we
have to multiply
    # Z value to reach altitude treshold.
    factor = alt_treshold / unit_vector[2]
    if current_row["center"] != 0 and current_row["center"] <= factor:
        return True
    else:
        return False


# Add class column
df["Alarm"] = False
df['Alarm'] = df.apply(calculate_alarm, axis=1)

print df
print df.columns
print "-------- Alarm == True ---------"
print df[df["Alarm"]]

# round values to 4 decimals
df["ALT"] = df["ALT"].round(4)
df["ROLL"] = df["ROLL"].round(4)
```

```
df["PITCH"] = df["PITCH"].round(4)

# write df to a csv file with current date time
now = str(datetime.now()).replace('.', '').replace(":", "-")
df.to_csv('newCsv_' + now + '.csv', index=False, sep=';')
```

III. Annexe - Script Python de transformation de données brutes en fichier kml

```
import math
import pandas as pd
from Tkinter import Tk
from tkinter.filedialog import askopenfilename
from scipy.spatial.transform import Rotation as R
from datetime import datetime

r_earth = 6378137.  # constant for earth's radius according to WGS84
(6'378'137 meters)
pi = math.pi


def get_file_name():
    Tk().withdraw()  # we don't want a full GUI, so keep the root window
from appearing
    filepath = askopenfilename()  # show an "Open" dialog box and return
the path to the selected file
    return filepath


def get_new_coordinate(new_real_vector, startPoint):
    # calculates coordinates according to lidar vector distance
    # param: lidar vector, coordinates of drone position

    # meters for longitude (positive number = -> East)
    x = new_real_vector[1]
    # meters for latitude (positive number = -> North
    y = new_real_vector[0]
    # meters for elevation (positive number = -> going up
    z = new_real_vector[2]

    # Calculs, retrieved at:
    # https://stackoverflow.com/questions/7477003/calculating-new-
longitude-latitude-from-old-n-meters
    newX = startPoint[1] + (x / r_earth) * (180. / pi) /
math.cos(startPoint[0] * pi / 180.)
    newY = startPoint[0] + (y / r_earth) * (180. / pi)
    newZ = startPoint[2] + z

    return str(newX) + ", " + str(newY) + ", " + str(newZ) # format matches
a line in the kml file


def get_lidar_point(initial_vector, row, side):
    if row[side] <= 0:
        return
    # Euler angle calculation for lidar trajectory
    new_unit_vector = R.from_euler('ZYX', [row["YAW"], row["PITCH"],
row["ROLL"]], degrees=True).apply(initial_vector)

    new_real_vector = []
    for i in new_unit_vector:
```

```
        new_real_vector.append(i * float(row[side]) / 100.)

    point_string = get_new_coordinate(new_real_vector, [row["LA"],
row["LON"], row["ALT"]])

    return point_string


def create_lidar_frame_left(row):
    # Method applied on df to create new left lidar frame
    v_left = [0.70183611, -0.08617464, -0.70710678]  # unit vector
    point_string = get_lidar_point(v_left, row, "left")
    return pd.Series([row["timestamp"], point_string])


def create_lidar_frame_right(row):
    # Method applied on df to create new right lidar frame
    v_right = [0.70183611, 0.08617464, -0.70710678]  # unit vector
    point_string = get_lidar_point(v_right, row, "right")
    return pd.Series([row["timestamp"], point_string])


filename = get_file_name()

try:
    df = pd.read_csv(filename, delimiter=';', quotechar="\"")
except IOError:
    print "No file selected, process terminated"
    exit(42069)

# Retrieves coordinates from flight header
coord_data = pd.DataFrame(df["coord"].str.split(', ', expand=True).values,
                          columns=["uidCoord", "timestampCoord", "LA", "LON",
"ALT", "ROLL", "PITCH", "YAW"])

df["ALT"] = pd.to_numeric(coord_data["ALT"].str[1:-1])
df["LA"] = pd.to_numeric(coord_data["LA"].str[1:-1])
df["LON"] = pd.to_numeric(coord_data["LON"].str[1:-1])
df["ROLL"] = pd.to_numeric(coord_data["ROLL"].str[1:-1])
df["PITCH"] = pd.to_numeric(coord_data["PITCH"].str[1:-1])
df["YAW"] = pd.to_numeric(coord_data["YAW"].str[2:-2])

df = df.drop(['uid', 'cnt', 'radar', 'coord'], axis=1)

# Df that holds all left lidar points (coordinates)
df_left_lidar = pd.DataFrame()
df_left_lidar[["timestamp", "point"]] = df.apply(create_lidar_frame_left,
axis=1)

# Df that holds all right lidar points (coordinates)
df_right_lidar = pd.DataFrame()
df_right_lidar[["timestamp", "point"]] = df.apply(create_lidar_frame_right,
axis=1)

# Creates and formats all coordinates of flight trajectory
string_for_lines = df[["LON", "LA", "ALT"]].to_string(header=False,
                                                      index=False,
```

```
index_names=False).replace("  ", ",")

string_for_lines = '\t\t\t' +
'\t\t\t'.join(string_for_lines.splitlines(True))

# Date and time to write on the new kml file's name
now = str(datetime.now())\
    .replace('.', '-')\
    .replace(":", "-")\
    .replace(' ', '-')

# Open and write a new kml file with new data and correct format
with open("flightpath-" + now + ".kml", 'w') as new_kml:

    def write_placemark_line(row, side):
        # Method to write all placemark (points)
        new_kml.write("\t\t<Placemark>\n")
        new_kml.write("\t\t\t<name>" + side + "-" + row["timestamp"] +
"</name>\n")
        new_kml.write("\t\t\t<styleUrl>#defaultStyle</styleUrl>\n")
        new_kml.write("\t\t\t<Point>\n")
        new_kml.write("\t\t\t\t<coordinates>" + str(row["point"]) +
"</coordinates>\n")
        new_kml.write("\t\t\t\t<altitudeMode>absolute</altitudeMode>\n")
        new_kml.write("\t\t\t</Point>\n")
        new_kml.write("\t\t</Placemark>\n")


    def write_placemark_line_left(row):
        if row["point"]:
            write_placemark_line(row, "left")


    def write_placemark_line_right(row):
        if row["point"]:
            write_placemark_line(row, "right")


    def write_linestring(rows):
        new_kml.writelines(rows + "\n")


    with open("kml_static/kml_head.txt", 'r') as kml_head:
        # Reads the static header of a kml file and writes it
        new_kml.write(kml_head.read() + "\n")

    # Writes the trajectory of the drone
    write_linestring(string_for_lines)

    with open("kml_static/kml_middle01.txt", 'r') as kml_middle01:
        # Reads and writes the middle part of the kml file (static for our
process)
        new_kml.write(kml_middle01.read() + '\n')

    df_left_lidar.apply(write_placemark_line_left, axis=1)
    df_right_lidar.apply(write_placemark_line_right, axis=1)

    with open("kml_static/kml_end.txt", 'r') as kml_end:
```

```
        # Reads and Writes the end of the kml file including styles and
icons
        new_kml.write(kml_end.read() + '\n')

    print "New Kml file created: " + new_kml.name
```

kml_static/kml_head.txt :

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
   <Folder>
      <name>GPS track</name>
      <open>1</open>
      <description>
         Folder that contains the flight line from GPS data
      </description>
      <Placemark>
       <name>Flight path</name>
       <LineString>
        <coordinates>
        <!-- longitude, latidude, altitude-->
```

kml_static/kml_middle01.txt :

```
         </coordinates>
     <altitudeMode>absolute</altitudeMode>
    </LineString>

    <Style>
     <LineStyle>
      <color>#ff0000ff</color>
      <width>3.5</width>
     </LineStyle>
    </Style>
   </Placemark>
</Folder>
<Folder>
   <name>Lidar data</name>
   <open>1</open>
   <description>
      Folder that contains all the points detected by lidars
   </description>
```

kml_static/kml_end.txt :

```
        <Style id="defaultIcon">
         <LabelStyle>
            <scale>0</scale>
         </LabelStyle>
         <IconStyle>
            <scale>0.4</scale>
```

```
            <Icon>
               <href>http://maps.google.com/mapfiles/kml/paddle/red-square-
lv.png</href>
            </Icon>
            <hotSpot x="15" y="1" xunits="pixels" yunits="pixels"/>
         </IconStyle>
      </Style>
      <Style id="hoverIcon">
         <IconStyle>
            <scale>0.5</scale>
            <Icon>
               <href>http://maps.google.com/mapfiles/kml/paddle/red-square-
lv.png</href>
            </Icon>
            <hotSpot x="15" y="1" xunits="pixels" yunits="pixels"/>
         </IconStyle>
      </Style>
      <StyleMap id="defaultStyle">
         <Pair>
            <key>normal</key>
            <styleUrl>#defaultIcon</styleUrl>
         </Pair>
         <Pair>
            <key>highlight</key>
            <styleUrl>#hoverIcon</styleUrl>
         </Pair>
      </StyleMap>

   </Folder>

</Document>
</kml>
```

IV. Annexe - Script Python de classification des données avec Euler et vecteur unitaire

```
from Tkinter import Tk
from tkinter.filedialog import askopenfilename
import pandas as pd
from datetime import datetime
from scipy.spatial.transform import Rotation as R

# allow a bigger size when printing a df
pd.options.display.width = 0


def get_file_name():
    Tk().withdraw()  # we don't want a full GUI, so keep the root window
from appearing
    filepath = askopenfilename()  # show an "Open" dialog box and return
the path to the selected file
    return filepath


filename = get_file_name()

df = pd.read_csv(filename, delimiter=';', quotechar="\"")

# Retrieve coordinates from flight header
coord_data = pd.DataFrame(df["coord"].str.split(', ', expand=True).values,
                          columns=["uidcoord", "timestampcoord", "LA",
"LON", "ALT", "ROLL", "PITCH", "YAW"])

df["ALT"] = pd.to_numeric(coord_data["ALT"].str[1:-1])
df["LA"] = pd.to_numeric(coord_data["LA"].str[1:-1])
df["LON"] = pd.to_numeric(coord_data["LON"].str[1:-1])
df["ROLL"] = pd.to_numeric(coord_data["ROLL"].str[1:-1])
df["PITCH"] = pd.to_numeric(coord_data["PITCH"].str[1:-1])

df = df.drop(['uid', 'cnt', 'radar', 'coord'], axis=1)

# Delete all rows that are similar to the next two rows (comparing
altitude, lat and lon)
# to filter data when drone is not flying
index_table = []  # stores the index of rows to delete

for index, row in df.iterrows():
    try:
        second_row = df.iloc[index + 1]
        third_row = df.iloc[index + 2]
        if (row.ALT == second_row.ALT
                and row.ALT == third_row.ALT
                and row.LA == second_row.LA
                and row.LA == third_row.LA
                and row.LON == second_row.LON
                and row.LON == third_row.LON):
            print ("row " + str(index) + " deleted")
            index_table.append(index)
```

```
    except IndexError:
        print row
        index_table.append(index)

df.drop(index_table, inplace=True)

# Delete unused rows
df = df.drop(['LA', 'LON'], axis=1)


def get_middle_lidar_point(current_row):
    # if both lidar have values, then average, if not, we only take the non
zero value
    if current_row["left"] <= 0 and current_row["right"] <= 0:
        return 0
    elif current_row["left"] <= 0:
        return current_row["right"]
    elif current_row["right"] <= 0:
        return current_row["left"]
    else:
        return (current_row["right"] + current_row["left"]) / 2.


df["center"] = df.apply(get_middle_lidar_point, axis=1)

# variables to calculate alarm
base_unit_vector_center = [0.70710678, 0, 0.70710678]
alt_treshold = 250  # threshold in cm


def calculate_alarm(current_row):
    yaw_pitch_roll = [0, current_row["PITCH"], current_row["ROLL"]]  # yaw
always zero as it doesn't matter
    unit_vector = R.from_euler('ZYX', yaw_pitch_roll,
degrees=True).apply(base_unit_vector_center)
    # factor = treshold after drone rotation, equals the number of times we
have to multiply
    # Z value to reach altitude treshold.
    factor = alt_treshold / unit_vector[2]
    if current_row["center"] != 0 and current_row["center"] <= factor:
        return True
    else:
        return False


# Add class column
df["Alarm"] = False
df['Alarm'] = df.apply(calculate_alarm, axis=1)

print df
print df.columns
print "-------- Alarm == True ---------"
print df[df["Alarm"]]  # prints all values that are alarming

# round values to 4 decimals
df["ALT"] = df["ALT"].round(4)
df["ROLL"] = df["ROLL"].round(4)
```

```
df["PITCH"] = df["PITCH"].round(4)

# write df to a csv file with current date time
now = str(datetime.now()).replace('.', '').replace(":", "-")
df.to_csv('newCsv_' + now + '.csv', index=False, sep=';')
```

V. Annexe – Product backlog

US Nr.	Thème	User Stories			Acceptance Criteria	Priority	Status	Size	Sprint	MoSCoW
		En tant que ...	Je veux ...	car/pour ...						
1	Préparation	Auteur	Préparer l'environnement de travail	Travailler plus facilement	Structure de dossier en place, Document word préparé, Modèle pour PV, Modèle pour tableau de bord et décompte hebdomadaire, product backlog.	1000		1	0	Must
2	Étude	Auteur	Apprendre les bases de Python et savoir utiliser des frameworks externes.	Pouvoir utiliser des scripts efficaces en machine learning et comprendre les scripts déjà implémentés	Créer un script permettant de récupérer des données d'un .csv et de les visualiser.	800		8	1	Must
4	Transformation	Auteur	Transformer les données radar, lidar et gps du drone fournies lors de la première prise de mesures	Pouvoir utiliser ces données facilement et les intégrer à des outils de visualisation.	Workflow Knime qui transforme le .csv de base dans un format réutilisable pour la visualisation.	970		13	1	Must
3	Visualisation	Auteur	Visualiser les données GPS, radar et lidar en 3D sur google earth	Nous donner une idée de ce que les capteurs peuvent repérer dans l'espace	créer un fichier .kml contenant la position du drone et une visualisation du terrain pour un vol (Leytron 2020-5-6-7-18-11).	950		5	0	Must
5	Visualisation	Auteur	Créer un script Python qui trace un graphique des vecteurs unitaires de la direction du drone et des lidars après rotation.	Avoir une idée de la situation dans l'espace du drone et de la zone visée par les lidars	Script python qui trace un graph des valeurs pour le drone à l'arrêt et le résultat après la rotation	900		13	1	Must
6	Transformation	Auteur	Créer un script Python qui prend un csv brut en entrée et crée un fichier avec les données classifiées	Entrainer le modèle	Script python qui retourne un nouveau fichier csv avec les données prêtes pour l'entrainement du modèle	890		8	2	Must
7	Modélisation	Auteur	Créer un workflow knime qui entraine plusieurs algorithmes différents et qui teste ceux-ci avec des données d'entrainement	Évaluer la possibilité d'utiliser du machine learning dans une situation de reconnaissance d'obstacles à basse altitude	Workflow Knime qui prend en entrée un fichier d'entrainement, qui partage les données et qui retourne les résultats de la classification de ceux-ci	880		21	2	Must

VI. Annexe – Journal de bord

Tableau de bord

Titre Détection d'obstacles à basse altitude pour drone en utilisant du machine learning

Étudiant Jean-Marie Alder

Période Du 04.05.2020 au 31.07.2020

	Sem. 1	Sem. 2	Sem. 3	Sem. 4	Sem. 5	Sem. 6	Sem. 7	Sem. 8	Sem. 9	Sem. 10	Sem. 11	Sem. 12	Sem. 13	**Total**
Analyse / Planification	5.50	2.00	1.00	0.50	0.50	0.50	1.50	0.50	2.00	1.00	0.50	1.00	4.00	20.50
Installation / Configuration	3.50	5.00	1.50	3.00	2.00	5.00	11.00	2.00	0.00	0.00	0.00	0.00	0.00	33.00
Programmation	0.00	2.00	0.00	0.00	0.00	0.00	22.50	0.00	18.00	0.50	5.00	17.00	5.00	70.00
Recherche / Lecture	9.00	12.00	11.50	5.00	0.00	25.50	8.50	5.50	5.50	7.50	12.00	0.00	0.00	102.00
Rédaction	1.00	0.00	11.50	10.50	13.50	0.00	5.00	10.50	13.50	30.00	20.00	24.50	18.00	158.00
Validation / Test	1.00	1.00	0.50	0.00	4.50	0.00	2.00	0.00	1.00	2.00	3.00	0.00	0.00	15.00
Total	20.00	22.00	26.00	19.00	20.50	31.00	50.50	18.50	40.00	41.00	40.50	42.50	27.00	398.50

Rapport Hebdomadaire

Titre: Détection d'obstacles à basse altitude pour drone en utilisant du machine learning
Étudiant: Jean-Marie Alder
Semaine: N°1 Du 04.05.2020 au 10.05.2020

	Lun	Mar	Mer	Jeu	Ven	Sam	Dim	**Total**
Analyse / Planification	1.50		4.00					5.50
Installation / Configuration				3.50				3.50
Programmation								0.00
Recherche / Lecture	4.00	3.00	2.00					9.00
Rédaction			1.00					1.00
Validation / Test				1.00				1.00
Total	5.50	3.00	7.00	4.50	0.00	0.00	0.00	20.00

Détails du Rapport	
Date	**Note**
04.05.20	Premier meeting, Discussions générales sur le déroulement du TB, Lecture du TB de l'année passée (Olivier).
05.05.20	Lecture du TB (Olivier)
06.05.20	Meeting avec Aero 41 et demonstration du matériel. Prise en main sur le terrain
07.05.20	Workflow knime pour préparer les données et premières visualisations (surface plot)

Commentaires

Rapport Hebdomadaire

Titre	Détection d'obstacles à basse altitude pour drone en utilisant du machine learning
Étudiant	Jean-Marie Alder
Semaine	N°2 Du 11.05.2020 au 17.05.2020

	Lun	Mar	Mer	Jeu	Ven	Sam	Dim	**Total**
Analyse / Planification	1.00	1.00						2.00
Installation / Configuration	4.00		1.00					5.00
Programmation			2.00					2.00
Recherche / Lecture	2.00	4.00	4.00			2.00		12.00
Rédaction								0.00
Validation / Test		1.00						1.00
Total	7.00	6.00	7.00	0.00	0.00	2.00	0.00	22.00

Détails du Rapport	
Date	**Note**
11.05.20	Meeting: discussion sur les résultats de la semaine. A faire: représentation dans le temps et utilisation des données des lidars. Tests knime et PowerBI pour visualisation.
12.05.20	Meeting: explications sur le script utilisé pour la récolte de données et sur les radars/lidars. Recherches sur le fonctionnement des radars.
13.05.20	Configuration et tests de lecture de données en python. Recherche d'articles scientifiques sur le sujet de la détection d'obstacle pour drones
16.05.20	Recherches de sources

Commentaires

Rapport Hebdomadaire

Titre	Détection d'obstacles à basse altitude pour drone en utilisant du machine learning
Étudiant	Jean-Marie Alder
Semaine	N°3 Du 18.05.2020 au 24.05.2020

	Lun	Mar	Mer	Jeu	Ven	Sam	Dim	**Total**
Analyse / Planification	1.00							1.00
Installation / Configuration					1.50			1.50
Programmation								0.00
Recherche / Lecture	3.00	2.50			4.00	2.00		11.50
Rédaction	2.00	4.50			3.00	2.00		11.50
Validation / Test	0.50							0.50
Total	6.50	7.00	0.00	0.00	8.50	4.00	0.00	26.00

Détails du Rapport	
Date	**Note**
18.05.20	Meeting: Présentation des résultats des lidars et des recherches pour état de l'art. A faire: Transposer ces données ainsi que les données GPS sur Google Earth et continuer l'état de l'art.
19.05.20	Création doc word, Product backlog, tableau de bord. Mise en forme en règle et optimale pour travailler plus rapidement à l'avenir. Continuation recherches pour état de l'art
22.05.20	Etat de l'art, Tests knime. Problème: les données gps sont éronnées et ne peuvent pas être utilisées pour une visualisation Google Earth.
23.05.20	Etat de l'art, recherches et rédaction.

Commentaires

Rapport Hebdomadaire

Titre: Détection d'obstacles à basse altitude pour drone en utilisant du machine learning
Étudiant: Jean-Marie Alder
Semaine: N°4 Du 25.05.2020 au 31.05.2020

	Lun	Mar	Mer	Jeu	Ven	Sam	Dim	**Total**
Analyse / Planification	0.50							0.50
Installation / Configuration	3.00							3.00
Programmation								0.00
Recherche / Lecture	1.50	2.00					1.50	5.00
Rédaction	1.50	4.00					5.00	10.50
Validation / Test								0.00
Total	6.50	6.00	0.00	0.00	0.00	0.00	6.50	19.00

Détails du Rapport	
Date	**Note**
25.05.2020	Meeting: objectif pour la sem. prochaine, continuer Etat de l'art (tableau comparatif des radars) et mettre au propre le workflow knime
26.05.2020	Fin rédaction première récolte de donnée + transformation pour visualisation 3d
31.05.2020	Comparaison des types de radars

Commentaires

Rapport Hebdomadaire

Titre Détection d'obstacles à basse altitude pour drone en utilisant du machine learning
Étudiant Jean-Marie Alder
Semaine N°5 du 01.06.2020 au 07.06.2020

	Lun	Mar	Mer	Jeu	Ven	Sam	Dim	**Total**
Analyse / Planification		0.50						0.50
Installation / Configuration		1.00		1.00				2.00
Programmation								0.00
Recherche / Lecture								0.00
Rédaction		5.00		4.00	4.50			13.50
Validation / Test				1.00		3.50		4.50
Total	0.00	6.50	0.00	6.00	4.50	3.50	0.00	20.50

Détails du Rapport	
Date	**Note**
02.06.2020	Meeting: définition des critères d'évaluation pour représentation de l'environnement. (SIG) et nouvelles données à tester
04.06.2020	Tests des données, analyse des résultats
05.06.2020	Fin Etat de l'art pour senseurs, début état de l'art pour SIG.
06.06.2020	Tests gps avec mavic pro sur google earth

Commentaires

Rapport Hebdomadaire

Titre Détection d'obstacles à basse altitude pour drone en utilisant du machine learning
Étudiant Jean-Marie Alder
Semaine N°6 du 08.06.2020 au 14.06.2020

	Lun	Mar	Mer	Jeu	Ven	Sam	Dim	**Total**
Analyse / Planification	0.50							0.50
Installation / Configuration	2.00		1.50				1.50	5.00
Programmation								0.00
Recherche / Lecture	3.00	6.00	6.00		6.50		4.00	25.50
Rédaction								0.00
Validation / Test								0.00
Total	5.50	6.00	7.50	0.00	6.50	0.00	5.50	31.00

Détails du Rapport	
Date	**Note**
08.06.2020	Meeting: Les données gps vont être améliorées et de nouveaux vols d'essais et de récolte de données à programmer. Continuer les recherches sur comment représenter les données lidars sur un SIG (google earth pro)
09.06.2020	Recherche de méthodes pour implémenter le roll,pitch,yaw de l'appareil
10.06.2020	Recherche de méthodes pour implémenter le roll,pitch,yaw de l'appareil, installation et tests de programmes sig open source.
12.06.2020	Lectures sur l'algèbre linéaire et sur les angles d'Euler
14.06.2020	Euler et les quaternions (compréhension et analyse de la méthode adaptée à notre problème. Visualisation des nouvelles données (GPS et lidar).

Commentaires

Rapport Hebdomadaire

Titre: Détection d'obstacles à basse altitude pour drone en utilisant du machine learning
Étudiant: Jean-Marie Alder
Semaine: N°7 du 15.06.2020 au 21.06.2020

	Lun	Mar	Mer	Jeu	Ven	Sam	Dim	**Total**
Analyse / Planification	1.50							1.50
Installation / Configuration					5.00	6.00		11.00
Programmation	5.00	3.00	6.50	4.50	3.50			22.50
Recherche / Lecture	2.00	3.00	1.50	2.00				8.50
Rédaction							5.00	5.00
Validation / Test	1.00		1.00					2.00
Total	9.50	6.00	9.00	6.50	8.50	6.00	5.00	50.50

Détails du Rapport	
Date	**Note**
15.06.2020	Meeting: Point sur les prochains objectifs. Les nouvelles données ont l'air d'être valides, il faut pouvoir utiliser la position gps ainsi que les "roll, pitch, yaw". Pour état de l'art de la représentation de l'environnement, essayer google earth open source et arcGIS (payant). Programmation en python d'un plot pour représenter le vecteur du lidar dans l'espace. tests concluants jusqu'ici.
16.06.2020	Recherches euler et programmation de scripts pour euler et la conversion de vecteur en latitude/longitude
17.06.2020	Programmation et debug de l'angle d'euler, tests avec le mavic pour une situation réelle
18.06.2020	Tests de manipulation de données avec pandas et création d'un script qui prend les données bruts du drone pour les transformer en coordonnées xyz.
19.06.2020	Script python pour visualiser en "point cloud" les données du radar à l'avant en fonction du temps. Installation d'une VM ubuntu 16.04.6 pour le SIG Earth Entreprise
20.06.2020	Fin de l'installation de Earth Entreprise
21.06.2020	

Commentaires

15.06.2020: Meeting avec Jérome Tréboux. Problème: difficultés à utiliser l'algèbre linéaire et la représentation des axes de rotation dans l'espace ainsi que l'ordre dans lequel effectuer les rotations. Doutes sur la technique à utiliser (euler ou quaternions). Solution: Utiliser Euler et trouver l'ordre de rotation qui convient à la situation, effectuer des tests dans l'espace pour voir si le comportement réel est proche de la simulation mathématique.

17.06.2020: Grosses difficultés dans la réalisation du script pour l'angle d'euler, différence entre rotation intrinsèque et extrasèque. Avec scipy, en utilisant from_euler(), il faut impérativement mettre les majuscules en premier paramètre, example: 'ZYX' pour l'ordre correcte pour un drone.

19.06.2020: L'installation de Earth Entreprise est très rigide et peu documentée, beaucoup de temps perdu lors de l'installation de ce logiciel.

Rapport Hebdomadaire

Titre	Détection d'obstacles à basse altitude pour drone en utilisant du machine learning
Étudiant	Jean-Marie Alder
Semaine	N°8 du 22.06.2020 au 28.06.2020

	Lun	Mar	Mer	Jeu	Ven	Sam	Dim	**Total**
Analyse / Planification	0.50							0.50
Installation / Configuration			2.00					2.00
Programmation								0.00
Recherche / Lecture	3.50	1.00					1.00	5.50
Rédaction		3.00	5.50				2.00	10.50
Validation / Test								0.00
Total	4.00	4.00	7.50	0.00	0.00	0.00	3.00	18.50

Détails du Rapport	
Date	**Note**
22.06.2020	Meeting: présentation du script pour point cloud, nous pouvons boserver une forte variation des mesures radar, prochain objectif est de faire une moyenne afin de trouver un facteur de conversion pour faire correspondre à la distance lidar.
23.06.2020	Etat de l'art Visualisation: Earth Enterprise
24.06.2020	Etat de l'art Visualisation: QGIS et ArcGIS
28.06.2020	Recherches sur les solutions de détection d'obstacles embarquées pour drones

Commentaires

Rapport Hebdomadaire

Titre Détection d'obstacles à basse altitude pour drone en utilisant du machine learning
Étudiant Jean-Marie Alder
Semaine N°9 du 29.06.2020 au 05.07.2020

	Lun	Mar	Mer	Jeu	Ven	Sam	Dim	**Total**
Analyse / Planification	1.00			1.00				2.00
Installation / Configuration								0.00
Programmation			5.00	1.00		6.00	6.00	18.00
Recherche / Lecture	2.00	2.00					1.50	5.50
Rédaction	2.00	4.50	1.50		5.50			13.50
Validation / Test							1.00	1.00
Total	5.00	6.50	6.50	2.00	5.50	6.00	8.50	40.00

Détails du Rapport	
Date	**Note**
29.06.2020	Meeting + recherches état de l'art machine learning embarqué + fin Etat de l'art visualisation
30.06.2020	Etat de l'art machine learning embarqué
01.07.2020	Création du script pour préparer les données de test
02.07.2020	Point sur la situation en vidéo, proposition de modification des colonnes pour les données test
03.07.2020	Rédaction méthodologie et partie d'introduction. Début rédaction choix technologique
04.07.2020	Intégration des scripts, améliorations script pour données de test
05.07.2020	Continuation de l'intégration + effectuer une boucle sur toutes les mesures pour calculer les points lidar nécessaire en parallèle à la ligne de trajectoire du drone

Commentaires
29.06.2020: Meeting PO: Faire la moyenne des deux points pour trouver l'altitude du drone. Pour chacun des points, on a une altitude des deux lidars. On veut ajouter une troisième colonne (classe) pour voir si ok ou pas ok. Fichier csv: (série temporelle, fenêtre temporelle) Temps; Altitude drone; Altitude mesurée des deux lidars; Alarme ou pas alarme (classe). Filter les données lorsque le drone est en standby (si les données gps et d'altitude sont similaires pendant 3 mesures)

Rapport Hebdomadaire

Titre	Détection d'obstacles à basse altitude pour drone en utilisant du machine learning
Étudiant	Jean-Marie Alder
Semaine	N°10 du 06.07.2020 au 12.07.2020

	Lun	Mar	Mer	Jeu	Ven	Sam	Dim	**Total**
Analyse / Planification	1.00							1.00
Installation / Configuration								0.00
Programmation				0.50				0.50
Recherche / Lecture			4.00		2.00		1.50	7.50
Rédaction	4.00	6.50	3.50	4.00	6.00		6.00	30.00
Validation / Test				2.00				2.00
Total	5.00	6.50	7.50	6.50	8.00	0.00	7.50	41.00

Détails du Rapport	
Date	**Note**
06.07.2020	Meeting avec PO, rédaction des choix technologiques
07.07.2020	Fin rédaction choix technologiques
08.07.2020	Recherches technologiques et rédaction de cette partie
09.07.2020	Rédaction définitions et test des nouvelles données
10.07.2020	Rédaction et recherches définitions
12.07.2020	Fin rédaction des définitions

Commentaires

06.07.2020: Meeting PO: Objectifs: obtenir des données valides. Imaginer une solution pour entrer la classe le plus automatiquement possible. Afficher sur google earth les classes différentes dans des couleurs différentes. + continuer le rapport
09.07.2020: premier test de visualisation réussi avec des données concluantes.

Rapport Hebdomadaire

Titre	Détection d'obstacles à basse altitude pour drone en utilisant du machine learning
Étudiant	Jean-Marie Alder
Semaine	N°11 du 13.07.2020 au 19.07.2020

	Lun	Mar	Mer	Jeu	Ven	Sam	Dim	**Total**
Analyse / Planification	0.50							0.50
Installation / Configuration								0.00
Programmation	4.00			1.00				5.00
Recherche / Lecture		4.00	8.00					12.00
Rédaction				8.00	5.50	6.50		20.00
Validation / Test	1.00		2.00					3.00
Total	5.50	4.00	10.00	9.00	5.50	6.50	0.00	40.50

Détails du Rapport	
Date	**Note**
13.07.2020	Meeting PO, Modification et amélioration du script pour plotter les résultats radar
14.07.2020	Recherches et géométrie pour trouver une méthode de classification des données.
15.07.2020	Continuation des recherches et mise en place d'un modèle mathématique pour calculer la distance minimale sous le drone avant une alerte. Création du script de classification et préparation des fichiers d'entrainement
16.07.2020	Rédaction implémentation (transformation et visualisation des données) et amélioration du script pour fichier kml
17.07.2020	Rédaction implémentation (transformation pour classification et résultats visualisation.
18.07.2020	Rédaction implémentation + corrections et modifications Définitions + état de l'art

Commentaires
13.07.2020: Meeting PO: Résultats concluants sur la visualisation. Objectifs pour la semaine: - ajouter une courbe de moyenne entre les deux lidars, aussi pour les kml. - quels sont les mesures dont j'ai besoin, comment je vais classer les données. - ajouter le pitch au graphique et faire une représentation graphique avec toutes les données utilisées.

Rapport Hebdomadaire

Titre	Détection d'obstacles à basse altitude pour drone en utilisant du machine learning
Étudiant	Jean-Marie Alder
Semaine	N°12 du 20.07.2020 au 26.07.2020

	Lun	Mar	Mer	Jeu	Ven	Sam	Dim	**Total**
Analyse / Planification	1.00							1.00
Installation / Configuration								0.00
Programmation	2.00	4.00	3.00	5.50	2.50			17.00
Recherche / Lecture								0.00
Rédaction	4.00	5.50	4.00	2.00	5.00	4.00		24.50
Validation / Test								0.00
Total	7.00	9.50	7.00	7.50	7.50	4.00	0.00	42.50

Détails du Rapport	
Date	**Note**
20.07.2020	Meeting PO, rédatcion développement
21.07.2020	Corrections sur l'orthographe, la syntaxe et suppression des parties non nécessaires, modification des scripts et améliorations. Création d'un script pour ajouter la mesure du centre au kml
22.07.2020	Corrections du rapport après les corrections du PO, modification des scripts et création du fichier pour données d'entrainement, création du workflow knime pour l'entrainement du modèle
23.07.2020	Suite workflow Knime pour entrainement du modèle, meeting avec Jérome Tréboux pour compléter les lacunes (justifications pour résultats et techniques pour améliorer le workflow)
24.07.2020	Remaniement des données d'entrainement et modification du workflow Knime. Rédaction introduction
25.07.2020	Ajout des corrections et astuces de Jérome Tréboux au workflow knime. Rédaction développement, résumé, avant-propos, product backlog et Poster A3
26.07.2020	Rédaction glossaire, remerciements et implémentation
07.07.2020	Corrections du rapport (grammaire, orthographe, suppression ou modification des phrases mal formulées

Commentaires
Meeting 20.07.2020: corriger à la main les csv en regardand sur earth pour vérifier si les mesures sont ok. prendre un set de points pour l'utiliser comme données d'entrainement (supposer partir au temps 0 et stocker 10 points). au lieu des points, faire des lignes (1 pour chaque côté) sur le kml. pour le radar: valeur max de l'amplitude, trouver la distance moyenne à laquelle il y a l'amplitude la plus grande. prendre les 10 points les plus grands et faire moyenne => mettre dans le fichier d'input 24.07.2020: Meeting avec Jérome Tréboux, corrections du workflow actuel pour predictions de la nouvelle valeur avec l'historique (lag).

Rapport Hebdomadaire

Titre: Détection d'obstacles à basse altitude pour drone en utilisant du machine learning
Étudiant: Jean-Marie Alder
Semaine: N°13 du 27.07.2020 au 31.07.2020

	Lun	Mar	Mer	Jeu	Ven	Sam	Dim	Total
Analyse / Planification	1.00			3.00				4.00
Installation / Configuration								0.00
Programmation		5.00						5.00
Recherche / Lecture								0.00
Rédaction	5.00	6.00	7.00					18.00
Validation / Test								0.00
Total	6.00	11.00	7.00	3.00	0.00	0.00	0.00	27.00

Détails du Rapport	
Date	**Note**
27.07.2020	Meeting PO, workflow pour la modélisation ok, quelques modifications : changer le tree ensemble avec le decision tree. Identifier les valeurs qu'il a pas détecté et vérifier si des valeurs ne jouent pas. Se penser sur le tuning (jouer avec le seuil de décision. Montrer la différence de schéma entre celui d'Olivier et le notre avec les capteurs. + continuation corrections rapport
28.07.2020	Fin du workflow pour la modélisation et fin de la rédaction de l'implémentation. Début de la conclusion
29.07.2020	Fin conclusion et corrections générales + tables des matières, références et autres parties générées + ajout des annexes (PB, journal de bord)
30.07.2020	Préparation du dossier virtuel avec tous les scripts, les workflows knime et les annexes effectuées

Commentaires

Déclaration de l'auteur

Je déclare, par ce document, que j'ai effectué le travail de Bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de Bachelor, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail et que je cite ci-après : Prof. Dominique Genoud, l'entreprise Aero 41.

Echichens, le mercredi 29 juillet 2020

Jean-Marie Alder

Étudiant, HES-SO Valais