Master Systèmes Dynamiques et Signaux

Mémoire

Acquisition multi-capteurs et traitement de données pour la modélisation 3D de fumée

Auteur :
M. Gaétan Pérez

Jury :
Pr. L. Hardouin
Pr. L. Jaulin

24 août 2025

Résumé

Chaque année, des centaines de milliers d'hectares sont ravagés par des feux sauvages, entraînant des impacts dévastateurs sur la biodiversité et des conséquences économiques majeures. Dans ce contexte, une réaction rapide et efficace est essentielle. Le projet **FIREFLIES**, mené conjointement par le LAAS-CNRS et l'ENAC, vise à développer des méthodes d'analyse des incendies à l'aide d'une flotte de drones, afin de fournir rapidement des informations précises pour une intervention optimale. Une partie de ce projet est dédiée à la reconstruction 3D des panaches de fumée et à la prédiction de leur évolution à partir de données issues d'images et de LIDAR 3D. Ce rapport présente le travail réalisé durant un stage de 24 semaines, portant sur la collecte de données LIDAR et images d'un panache de fumée, en vue de leur intégration dans un pipeline de reconstruction 3D développé par Lou Denis. Un setup expérimental a été conçu pour capturer des données en environnement contrôlé ou semi-contrôlé, et des méthodes de segmentation d'objets dynamiques ainsi que de prédiction de l'évolution de séquences de nuages de points ont été explorées.

Abstract

Every year, hundreds of thousands of hectares are devastated by wildfires, causing severe impacts on biodiversity and major economic losses. In this context, rapid and effective response is essential. The **FIREFLIES** project, jointly conducted by LAAS-CNRS and ENAC, aims to develop methods for wildfire analysis using a fleet of drones, in order to provide accurate and timely information for optimal intervention. A key component of this project focuses on the 3D reconstruction of smoke plumes and the prediction of their evolution from image and 3D LIDAR data. This report presents the work carried out during a 24-week internship, dedicated to collecting LIDAR and image data of smoke plumes for integration into a 3D reconstruction pipeline developed by Lou Denis. An experimental setup was designed to capture data in controlled or semi-controlled environments, and methods for dynamic object segmentation in static scenes as well as point cloud sequence evolution prediction were explored.

Glossaire

3D Gaussian Splatting Technique de reconstruction 3D d'une scène à partir de prises de vues 2D, s'appuyant sur l'optimisation de propriétés d'ellipsoïdes initialisées dans l'espace (position, couleur, transparence, matrice de covariance) pour recréer la scène voulue.. 9, 14

backpropagation La backpropagation est l'algorithme d'optimisation qui calcule, le gradient de la fonction de coût par rapport aux paramètres d'un réseau de neurones, afin de mettre à jour ses poids pendant l'entraînement d'un modèle d'apprentissage.. 45

Pointcloud Un point cloud (nuage de points) est un ensemble de points dans un espace 3D (ou 2D) représentant la forme et la géométrie d'un objet ou d'une scène. Chaque point possède des coordonnées spatiales, et éventuellement des attributs supplémentaires comme la couleur ou l'intensité. . 22

SfM Algorithme permettant de générer un nuage de point 3D à partir de plusieurs prises de vue 2D dont les coordonnées sont connues.. 13

voxel Un voxel est l'équivalent tridimensionnel d'un pixel : c'est une cellule élémentaire d'un volume 3D, définie par sa position dans une grille régulière et éventuellement associée à une valeur (couleur, densité, intensité, etc.).. 19

Acronymes

3DGS 3D Gaussian Splatting. 14, 15

DART Discrete Anisotropic Radiative Transfer. 19

DLT Direct Linear Transform. 21

GPU Graphics Processing Unit. 13

ICP Iterative Closest Point. 27

MLP Multi Layer Perceptron. 13, 14

ROS Robot Operating System. 39

Remerciements

Je tiens tout d'abord à remercier toute l'équipe RIS pour l'accueil et l'ambiance chaleureuse qui y règne tous les jours. Cet environnement de travail m'aura permis de me sentir à l'aise dès le début de mon stage, et de m'intégrer sans souci à l'équipe, dont les membres, toujours ouverts, sont toujours prêts à aider.
Particulièrement, je souhaite remercier Harold Murcia pour son soutien technique et amical tout au long de ce stage avec de précieux conseils.

Évidemment, ce stage n'aurait pas été le même sans le groupe de stagiaires soudé qui s'est créé pendant ces 6 mois avec Sami, Shriram, Kaweena, Elfie, Titouan, Ciaran, Matthias, Myriam, Alessandro et Jamila. Merci infiniment à eux tous pour le soutient, avec une mention spéciale pour Alessandro, Myriam et Jamila qui ont été là du début jusqu'à la fin de ce stage et qui se sont transformés en de très précieux amis.

Finalement, je tiens à remercier Lou, doctorante au LAAS sur le projet FIREFLIES, avec qui j'ai travaillé pendant ce stage. Lou aura été un soutien perpétuel, toujours disponible pour me conseiller et m'aiguiller pendant ce stage en parallèle de sa thèse, même dans des situations difficiles.

Enfin, je souhaite rendre hommage à Simon Lacroix, ex-directeur de l'équipe RIS et tuteur de ce stage.

Table des matières

Table des figures

Chapitre 1

Introduction

1.1 Présentation du projet global

A cause du dérèglement climatique et de la hausse des températures pendant l'été, battant des nouveaux records chaque année, les incendies sauvages sont de plus en plus fréquents. Qu'ils soient criminels, accidentels ou naturels, ils sont extrêmement dévastateurs et semblent difficilement prévisibles. Ces feux extrêmes ont un impact direct sur la nature et la santé des êtres vivants (dont les humains), car très difficiles à maîtriser et dégageant des fumées toxiques affectant fortement la qualité de l'air [6]. Dans ce contexte de feux de plus en plus présents, il faut être préparé et tenter de gérer au mieux ces feux pour minimiser leur impact socio-écologique [7]. Dans le cadre de la gestion des incendies, le projet Fireflies (dans lequel s'inscrit ce stage) conjointement mené par le LAAS-CNRS et l'ENAC a pour ambition d'analyser les feux pour être capable de les modéliser précisément (évolution de la fumée dans l'atmosphère, propagation du feu selon le type de terrain, ...). A travers l'utilisation d'une flotte de drones équipés de caméras et LIDAR, l'objectif est ainsi de prévoir l'évolution à court terme des feux et de leur fumée pour être capable de réagir le mieux possible, avec des moyens accessibles et déployables rapidement.

1.2 Reconstruction du panache de fumée

Le projet dans lequel s'inscrit ce stage fait donc partie du plus large projet Fireflies. Celui-ci, mené par le LAAS autour de la thèse de Lou Denis consiste à créer un pipeline de reconstruction 3D d'un panache de fumée à partir d'images et de nuages de points provenant des capteurs embarqués. Cette reconstruction doit aussi respecter les propriétés physiques de la fumée (densité, évolution dans l'espace 3D, etc...), et a pour but d'améliorer les modélisations actuelles de panaches de fumée. En effet, la fumée est un phénomène très complexe et très coûteux à modéliser numériquement puisqu'il nécessite une discrétisation de l'espace et en temps afin d'appliquer la célèbre formule de Navier-Stokes [8]. De plus cette formule demande d'avoir accès aux données de température et de pression qui sont difficiles à obtenir. Pour éviter d'avoir à faire des hypothèses fortes sur les données qui impacteraient la précision du modèle, la méthode choisie est de passer par des techniques de *deep learning* et d'optimiser les paramètres de la reconstruction via des méthodes de rendu différentiable. Le modèle serait aussi utilisé pour prédire l'évolution physique de la fumée (évolution de la position, de la densité, ...). L'avantage de passer par le *deep learning* est de pouvoir reconstruire les phénomènes observés sans forcément connaître les propriétés de l'environnement. On est dans une logique de reconstruction "inverse" au lieu d'une simulation "forward" (recréer le phénomène à partir d'un environnement connu).

1.3 Problématique du stage

Pour reconstruire en trois dimensions le panache, et comparer le résultat avec des prises de vues réelles une technique dérivée du Gaussian Splatting (détaillée plus tard dans le rapport) est utilisée car elle permet une représentation plus avantageuse de l'espace (coût de stockage et contrôlabilité de la scène)[9]. De plus, l'approche *point based* rend cette méthode plus facile à connecter à la physique Lagrangienne et aux données LIDAR. Cette technique est basée sur le rendu visuel d'un mélange d'ellipsoïdes (ou Gaussiennes 3D) disposant de nombreuses propriétés dont leur position, couleur, transparence, etc... Cependant, le Gaussian Splatting est un algorithme purement visuel, présentant une faible précision physique et volumétrique [10], ce qui rend impossible son utilisation pour une analyse du panache de fumée. Par conséquent, certaines contraintes géométriques et physiques doivent être intégrées dans le pipeline de reconstruction afin d'atteindre la précision souhaitée dans les deux cas. Pour cela, Lou Denis a proposé d'intégrer des contraintes inspirées de la physique Lagrangienne lors de l'optimisation des paramètres des Gaussiennes, ainsi que d'exploiter les données LIDAR en complément des images de la scène. En effet, si on ne prend en compte que leur position, les Gaussiennes 3D peuvent s'apparenter à un nuage de points. En pratique, il s'agit d'évaluer la complémentarité entre trois aspects présentés dans la figure 1.1 :

- le rendu visuel (*Gaussian Splatting*) ;
- l'intégration de modèles physiques inspirés de la physique Lagrangienne ;
- l'apport de données LIDAR comme réalité terrain.

L'hypothèse est que ces trois dimensions combinées permettront d'améliorer la précision de la reconstruction 3D d'un panache de fumée, et d'en prédire l'évolution dynamique.

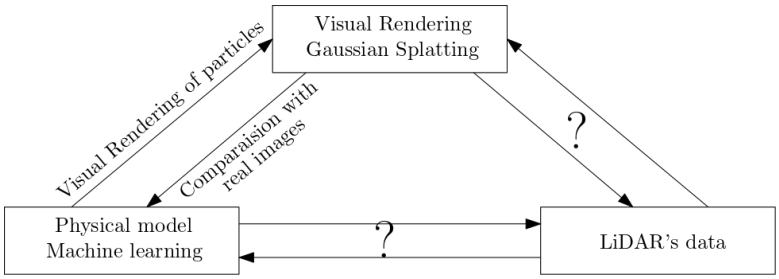


FIGURE 1.1 – Complémentarité des 3 aspects du projet de reconstruction 3D de panache de fumée. Les points d'interrogations n'ont pas encore été étudiés par manque de données.

Afin d'améliorer la précision du modèle, l'objectif de ce stage est de déterminer si, et comment des données LIDAR peuvent compléter le pipeline de reconstruction du panache. Cette problématique soulève plusieurs thèmes que nous traiterons dans ce rapport : l'étude du comportement des faisceaux lumineux du LIDAR dans des environnements dégradés, le traitement et l'analyse de données LIDAR et caméras, ainsi que la mise en place de *setups* expérimentaux.

Chapitre 2

Travail préliminaire et état de l'art

Comme mentionné dans l'introduction, ce stage s'inscrit dans un projet de reconstruction 3D de panache de fumée, et de prévision de son évolution. Afin de mieux comprendre le sujet, il est primordial de bien comprendre les principaux rouages du projet global avant de se plonger pleinement dans le sujet. On distingue alors deux parties (mais reliées comme représenté dans la figure 1.1) :

— La reconstruction 3D à partir de prises de vue 2D ;
— La prévision de l'évolution du panache ;

On détaillera alors premièrement ces deux premiers points avant de se concentrer sur l'observation de panaches de fumée avec des LIDAR et l'intégration de nuages de points dans le processus de reconstruction.

2.1 Reconstruction 3D

2.1.1 Principe et utilité

La reconstruction 3D permet une meilleure visualisation de l'environnement à partir de prises de vues 2D soit par différentes caméras calibrées entre elles, soit en utilisant une seule caméra dont on connaîtrait la position et ses déplacements. Le domaine commence avec des premiers algorithmes à partir des années 90 reconstruisant les rayons lumineux pour une position quelconque d'une caméra [11]. Mais c'est avec l'apparition de l'algorithme de Structure-From-Motion (SfM) [12] et de son dérivé le multiview-stereo (MVS) permettant d'extraire un nuages points à partir de plusieurs captures d'images, et d'y construire une surface basée sur la position des points obtenus [13]. Basés sur ces principes, de nombreux algorithmes ont pu atteindre des performances très impressionnantes, mais c'est surtout avec l'émergence fulgurante du *deep learning* et de l'amélioration des performances des Graphics Processing Unit (GPU), que la demande dans les technologies de reconstruction 3D a pris plus d'importance. Elle est présente dans de nombreux domaines tels que l'interaction homme-machine, la médecine, la perception de l'environnement par des systèmes autonomes ou la simulation. Certains algorithmes peuvent aussi y intégrer d'autres informations pour améliorer certains aspects comme la précision géométrique ou la densité de l'objet reconstruit ou même des modèles physiques pour des rendus dynamiques [10] [14] [15].

2.1.2 Gaussian Splatting

Récemment [16] a introduit les Neural Radiance Field (NeRF), qui basés sur le même principe de [11], permettent de cette fois de prédire de manière continue la couleur et la densité d'un point avec la position (x, y, z) et la direction (Θ, Φ) de la caméra sont données. Pour initialiser le modèle, on utilise l'algorithme SfM. Puis un Multi Layer Perceptron (MLP) est entraîné à prédire les rayons lumineux et

la géométrie d'une scène 3D en comparant avec des prises de vue 2D réelles grâce au *volumetric ray marching* (une méthode de rendu 2D d'un objet 3D). Malgré le progrès fulgurant dans la qualité des rendus apportés par cette technique de reconstruction, seulement 2 ans plus tard la technique de 3D Gaussian Splatting a marqué encore un nouveau tournant [9]. En utilisant une technique de rendu 2D introduite par [17] qui consiste à représenter l'espace non par par des points mais par des ellipsoïdes (Gaussiennes 3D) comme montré sur la figure 2.1. En optimisant les paramètres de ces Gaussiennes, [9] parvient à recréer une scène 3D avec la même qualité et des temps d'entraînement bien inférieurs en raison de l'absence du MLP utilisé par les NeRF pour prédire le rendu visuel du rayon [1].

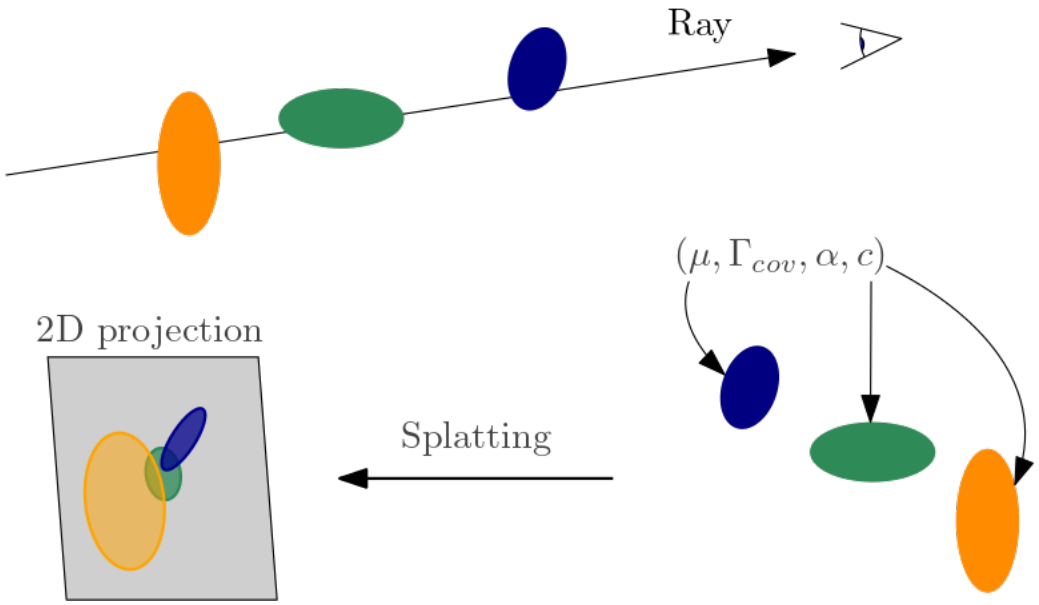


FIGURE 2.1 – Simple représentation du 3D Gaussian Splatting. Les Gaussiennes 3D sont projetées dans le plan 2D de l'image, et leurs paramètres sont optimisés pour faire correspondre la projection avec l'image de référence. Les paramètres sont : μ la position du centre de la Gaussienne, Γ_{cov} la matrice de covariance (équivalente à l'échelle et rotation de l'ellipsoïde), α l'opacité de la Gaussienne et c la couleur. Schéma inspiré de [1]

2.2 État de l'art de reconstruction de fumée

Le *3D Gaussian Splatting* (3DGS) est une technique efficace pour la reconstruction 3D de scènes, mais son objectif est porté sur la qualité et le réalisme de l'image 2D, et non pas sur la représentation 3D sous-jacente. La fumée est un milieu présentant de nombreux challenges pour la reconstruction : c'est un milieu semi-opaque, où de nombreux phénomènes lumineux (absorption, diffusion) se produisent. De plus, la fumée est un phénomène dynamique. La méthode de reconstruction 3DGS initiale ne permet donc pas de reconstruire la description volumétrique du phénomène. La méthode 3DGS telle quelle ne permet donc pas de remplir les objectifs de reconstruction et de prédiction.

2.2.1 Méthode forward

Traditionnellement pour modéliser les écoulements des fluides numériquement, il s'agit d'utiliser la célèbre équation incompressible de Navier-Stokes décrivant l'évolution temporelle et spatiale du champ de vitesse d'un fluide en fonction de la pression, de sa viscosité et de la densité du fluide [8]. On la retrouve généralement sous cette forme :

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \nabla p = \vec{g} + \nu \nabla \cdot \nabla \vec{u} \tag{2.1a}$$

$$\nabla \cdot \vec{u} = 0 \tag{2.1b}$$

Où $\vec{u}$ représente le champ de vitesse du fluide, ρ la densité, ν la viscosité et p la pression. On parle alors de méthode forward car on par d'un environnement connu pour modéliser l'évolution du fluide dans cet environnement.

2.2.2 Reconstruction inverse

Si le problème de la simulation est maintenant bien connu, il existe peu de méthodes de reconstruction inverse, c'est à dire retrouver des quantités pendant l'écoulement à partir de d'une observation partielle du fluide. Ainsi, ces modèles se basent sur des observations permettant de caractériser des paramètres du panache. De nombreuses techniques d'ingénierie permettant de caractériser les fluides en 3 dimensions dans des environnements contrôlés ont été développées. Il existe globalement deux grandes catégories : avec ou sans traceur passif dans le fluide. Par exemple, la technique du Particle Image Velocimetry (PIV) permet de suivre avec une caméra les déplacement d'une particule dans un fluide donnant ainsi des informations sur les champs de vitesse et de vortex [18]. Plus récemment, la tomographie (une méthode améliorée du PIV) a permis d'obtenir ces données en 3D grâce à une association de plusieurs caméras synchronisées [19].
A plus grande échelle, de nombreux capteurs peuvent être utilisés pour l'observation et la caractérisation de panaches de fumées. Des moyens très importants peuvent être utilisés comme de l'imagerie satellite, de l'imagerie par drone, ou l'imagerie infrarouge prise depuis un avion (plutôt utilisée pour caractériser les fronts de flammes). Des radars et LIDAR adaptés peuvent aussi être utilisés pour obtenir les champs de vitesses et la densité mais ces données sont complexes à implémenter et très coûteuses à obtenir.

Récemment des modèles de machine learning ont permis d'avoir des résultats très satisfaisants avec des observations partielles. En se basant sur des données tomographiques et des caméras monochromes et couleurs, [19] est capable de reconstruire la surface extérieure de la fumée par photogrammétrie (multiples prises de vues pour reconstruire la géométrie d'un environnement), ainsi qu'une carte de la densité depuis des points de vue quelconques et peut en déduire le champ de vitesse de la fumée. Seulement, la photogrammétrie seule implique de nombreux défis lorsqu'il s'agit de la fumée car cette dernière est très peu texturée et très sensible aux perturbations de l'environnement. Dans ce contexte, l'émergence des techniques de rendu différentiable, telles que NeRF ou 3DGS, a profondément transformé le domaine de la reconstruction de panaches de fumée. Par exemple, [20] démontre qu'en apprenant un modèle physique du comportement d'un panache, puis en ne reconstruisant que son état initial pour en simuler ensuite l'évolution, on obtient des résultats plus fidèles à la réalité. Une telle approche n'est rendue possible que grâce à l'utilisation du rendu différentiable, qui permet d'optimiser simultanément les paramètres visuels et dynamiques du modèle. [21] exploite également les NeRF pour le rendu visuel de panaches de fumée reconstruits, en minimisant l'écart entre les images synthétiques et les observations réelles. Parallèlement, un réseau de neurones est entraîné à estimer le champ de vitesse du panache en s'appuyant sur des fonctions de coût issues des équations de Navier-Stokes ((2.1a), (2.1b)), afin de garantir la cohérence physique du fluide simulé. En s'appuyant sur ces travaux, L. Denis travaille sur la reconstruction et l'évolution de densité 3D (donc pas seulement la surface) de panaches de fumée en utilisant le 3DGS.

2.3 Données caméras et LIDAR

2.3.1 Scalarflow

Pour pouvoir obtenir des résultats satisfaisants, les modèles basés sur le machine learning ont besoin de beaucoup de données. Il est possible d'utiliser des données simulées, mais ces dernières sont forcément moins fiables et moins représentatives d'une situation réelle. Ainsi, le *dataset Scalarflow* [22] constitue un pilier dans la communauté de reconstruction de panache de fumée, il est à ce jour le seul jeu de données disponible. Il contient 104 séquences de 150 frames de panaches de fumée prises avec 5 caméras autour de la source. Mais malgré la qualité et la quantité des données, celles-ci ne représentent que des panaches dans des conditions contrôlées (pas de perturbations extérieures, pas d'obstacles, éclairage contrôlé). Ainsi, l'un des buts de ce projet est également de créer un ensemble de données plus représentatif d'un panache de fumée "naturel" et plus diversifié pour venir compléter *Scalarflow*

2.3.2 Manque de données

Par ailleurs, des travaux se sont intéressés à l'intégration de nuages de points dans le processus de reconstruction 3D pour venir compléter des données images [23] [24] [25]. En revanche, ces travaux ne sont pas menés dans le cadre de la reconstruction de fumée, et il n'existe donc pas de données LIDAR dans ce domaine. Ainsi, on s'intéresse à la capture de données type nuages de points pour venir compléter les données d'images. Les données LIDAR peuvent servir de réalité terrain venant compléter les données images, avec une précision géométrique supplémentaire. De plus, à terme les collectes de données ont pour ambition d'être menées par des drones autonomes qui devront veiller à ne pas rentrer dans la fumée. Ceux-ci peuvent donc utiliser les données LIDAR, plus pratiques que des images pour estimer la distance par rapport au panache.

Chapitre 3

Mise en place du protocole et acquisitions

3.1 Comportement d'un LIDAR pour l'observation de la fumée

Si l'idée d'observer de la fumée avec un LIDAR paraît intéressante pour donner des informations sur la géométrie du panache et peut-être sur sa densité, il est important de comprendre précisément comment ce type de capteur fonctionne et comment l'utiliser dans notre contexte.

3.1.1 Principe global du LIDAR

Un LIDAR (Laser Imaging Detection and Ranging) est un appareil permettant de déterminer des distances par rapport à une cible ou un obstacle visé. Il utilise un ou plusieurs faisceaux laser et des récepteurs pour mesurer le temps de vol du rayon rétrodiffusé et ainsi estimer la distance d'un objet. Les lasers utilisés ont en général des longueurs d'ondes appartenant aux domaines des ultrasviolet ou infrarouge, mais certains émettent dans le domaine du visible. Il existe en effet de nombreux types de LIDAR utilisés dans des domaines très variés tels que l'archéologie, la sismologie, la physique de l'atmosphère, la robotique, etc... En sortie du capteur et après filtrage, on obtient des données sous forme de nuage de points offrant une description précise de l'environnement autour du LIDAR (en général en 3D même s'il existe aussi des LIDAR 2D, on se concentrera ici sur les LIDAR 3D). Afin de rendre un résultat en 3 dimensions, les LIDAR sont équipés de pièces mécaniques et optiques permettant d'étendre les possibilités des lasers. Par exemple, les LIDAR de l'entreprise Livox qui sont utilisés pendant le stage, utilisent deux prismes de Risley en rotation pour permettre à un seul laser fixe de pointer dans plusieurs directions comme illustré sur la figure 3.1.
Certains LIDAR appelés LIDAR Doppler sont capables de comparer le rayon émis et le rayon reçu afin de déterminer le décalage en fréquence. Cette information donne directement la vitesse de l'objet touché. Ce genre d'appareil est particulièrement utile pour l'observation de panaches de fumées car il permet d'observer de nombreux phénomènes au sein du panache [26]. Cependant, ce type de LIDAR est assez volumineux et ne peut pas être transporté par drone. Par conséquent, dans notre contexte, on doit se servir d'un LIDAR 3D standard moins sophistiqué, mais qui est plus approprié pour les vols de drones et pour l'examen rapproché du panache.

3.1.2 Interaction LIDAR et fumée

En robotique, les LIDAR 3D sont très utilisés pour les véhicules autonomes. Ainsi, la plupart des recherches faites dans le contexte des environnements visuels dégradés (fumée, brouillard, pluie, ...) visent à supprimer le bruit engendré par ces conditions. Notre cas est un peu différent, puisqu'on

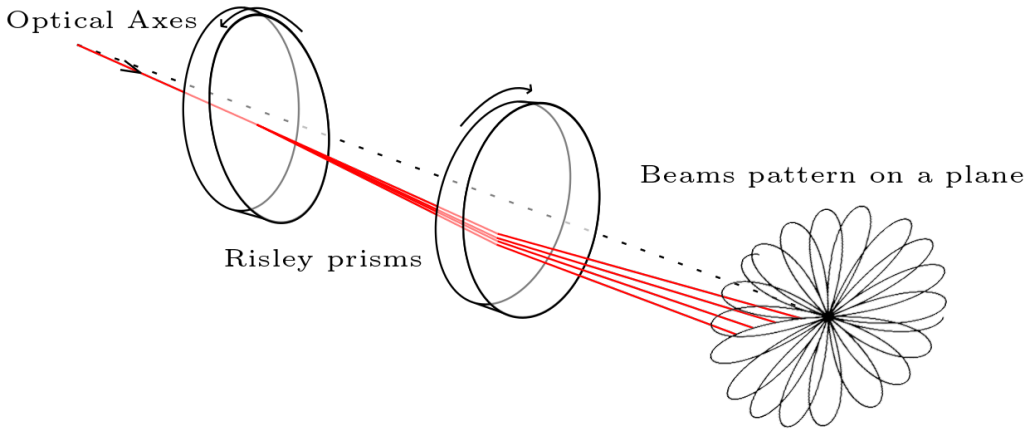


FIGURE 3.1 – Illustration du mécanisme avec 2 prismes de Risley pour couvrir un pattern avec un laser fixe.

cherche justement à caractériser le panache avec les données LIDAR, donc pas à le supprimer.
Lorsque de la lumière passe à travers une zone d'aérosols, il peut se produire plusieurs phénomènes [2] :

- **L'absorption :** la lumière est en partie absorbée par le nuage d'aérosol et perd donc en intensité.
- **Le out-scattering :** les photons interagissent avec les aérosols et sont déviés de leur trajectoire initiale.
- **Le in-scattering :** des photons provenant d'une autre source lumineuse interagissent avec les aérosols et sont déviées de leur trajectoire initiale comme pour le out-scattering. Mais leur nouvelle trajectoire atteint maintenant l'observateur.
- **L'émission :** dans certains cas (feu, explosion, bioluminescence), les particules d'aérosols émettent également de la lumière, qui s'ajoute à la source principale.

Ces phénomènes sont résumés dans la figure 3.2.

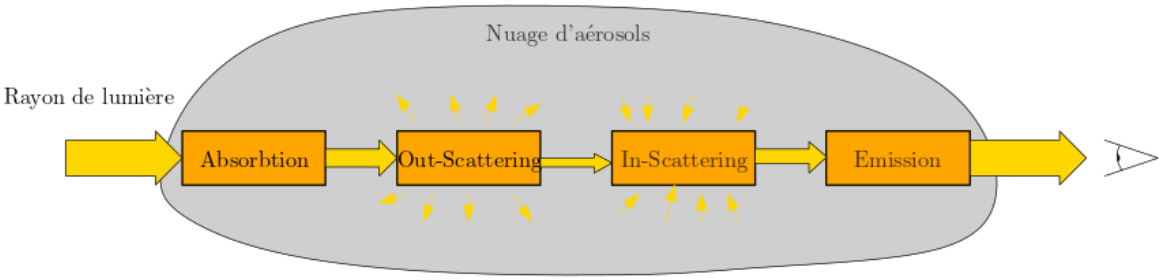
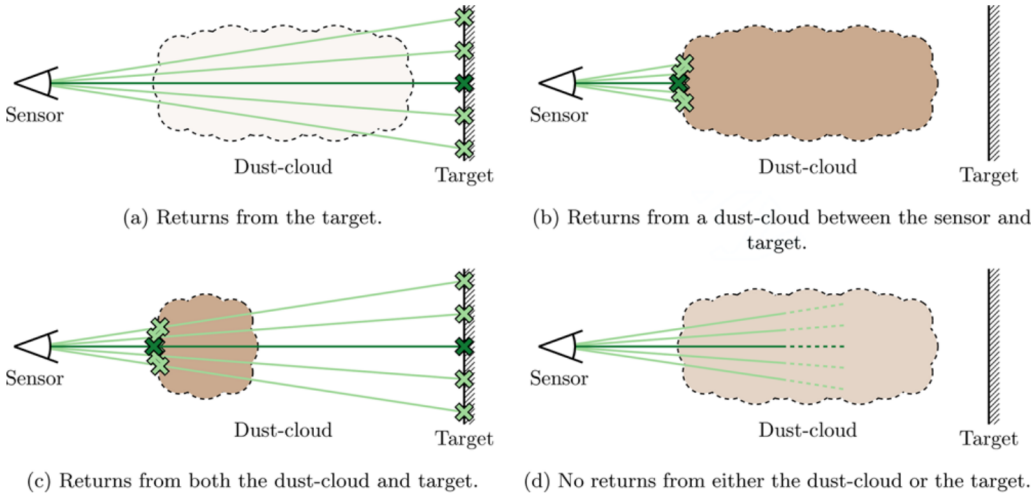


FIGURE 3.2 – Illustration des différents phénomènes possibles lorsqu'un faisceau lumineux traverse un nuage d'aérosols. Figure inspirée de [2]

Le LIDAR, qui utilise des faisceaux laser, devrait être soumis aux mêmes phénomènes bien que leur identification précise puisse être complexe. Cela donne tout de même une indication des défis auxquels on peut faire face avec les données LIDAR en ce qui concerne la fumée. Le principal étant la perte des rayons dans le panache à cause de l'absorption et du out-scattering. On peut aussi s'attendre à des faux points positifs dus au phénomène de l'émission lumineuse du panache (peu probable puisqu'un panache de fumée n'émet en théorie pas de lumière) ou du in-scattering qui serait indiscernable d'un véritable point. Dans le contexte des environnements visuels dégradés, et dans le cas spécifique des LIDAR (notre cas ici) la figure 3.3 résume les cas que l'on peut retrouver.
On remarque alors, qu'aucun cas ne permette avec un LIDAR 3D d'avoir des retours de l'intérieur du panache. Ainsi, d'après [3] si certains faisceaux traversent le nuage et renvoient un retour de la cible,



(a) Returns from the target.

(b) Returns from a dust-cloud between the sensor and target.

(c) Returns from both the dust-cloud and target.

(d) No returns from either the dust-cloud or the target.

FIGURE 3.3 – 4 cas répertoriés par [3] dans le cadre de nuage de poussière, comparable à un panache de fumée

il semble que les seules informations qu'on puisse avoir sur le panache ne soient que des points de la surface. Le reste semble être soit absorbé soit diffusé dans le panache. Les effets de in-scattering ou d'émission ne paraissent pas visibles.

3.2 Simulation de LIDAR dans la fumée

On a vu précédemment à quoi on pourrait s'attendre avec l'observation d'un panache de fumée avec un LIDAR 3D. Des données 3D de la surface du panache, même si elles peuvent être légèrement erronées dans le cas où il y aurait des faux-positifs, sont intéressantes car elles décrivent la géométrie du panache et peut-être que la quantité de points à certains endroits peut aussi donner un indicatif sur l'intérieur du panache. Nous avons choisi d'utiliser DART (Discrete Anisotropic Radiative Transfer), un modèle 3D basé sur la physique, simulant les interactions entre le rayonnement terrestre et l'atmosphère, des longueurs d'onde visibles aux longueurs d'onde infrarouges thermiques [27]. Il dispose également d'un mode "LIDAR" dans lequel il est possible de simuler des données LIDAR dans un environnement paramétré par ses propriétés optiques. La figure 3.4 montre le nuage de points obtenu dans un environnement contenant un LIDAR, un cône de fumée et un mur homogène.

On retrouve le résultat attendu, avec la surface du cône très bien décrite par les retours du LIDAR positionné devant. Comme attendu, il n'y a pas de points à l'intérieur du panache. Le nuage de points obtenu donne une bonne représentation de LIDAR en milieu homogène. En effet, ici on ne représente pas un vrai panache de fumée hétérogène mais un cône fixe homogène (mêmes propriétés optiques dans tout le cône). En effet, la fumée a généralement une composition et une densité hétérogène, ce qui n'est pas le cas dans la simulation DART. Il est possible de créer à la main une grille de voxel avec des coefficients optiques différents et de simuler pour chaque instant du panache afin de créer une scène dynamique, mais cela prendrait trop de temps à simuler et le modèle n'est pas optimisé pour ce genre de tâches. Ainsi, le passage à des expérimentations pour obtenir des données réelles semble nécessaires pour palier aux problèmes liés aux données simulées.

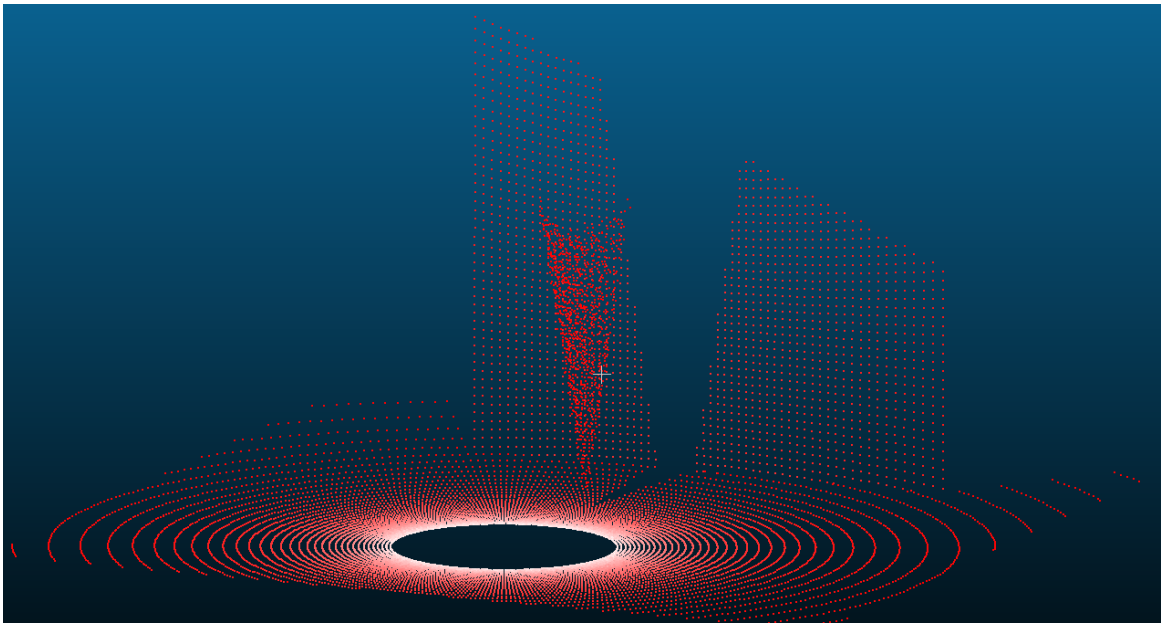


FIGURE 3.4 – Nuage de point obtenu avec le modèle DART dans un environnement simple avec un cône de fumée et un mur.

3.3 Robot MINNIE

En suivant la même logique que pour les données simulées, on cherche cette fois à visualiser à quoi ressemblent des données réelles prises avec un LIDAR 3D. Pour cette expérimentation, on génère un panache de fumée artificielle et on enregistre les données obtenues avec un LIDAR 3D ainsi qu'une caméra grand angle, tous deux présents sur un robot déjà existant développé au LAAS : MINNIE [28].

3.3.1 *Setup* expérimental

Le panache est généré par une machine à fumée utilisée en général dans des évènements ou des discothèques, elle peut être déclenchée à distance ce qui facilite les expérimentations. Ce genre de machine est aussi utilisé dans le *setup* de Scalarflow, la fumée expulsée est composée de glycol et d'eau, un mélange inoffensif pour la santé.
Du côté capteur, le LIDAR utilisé est un OUSTER OS2. Pour les expérimentations finales et le jeu de données, ce n'est pas ce type de LIDAR qui sera utilisé car peu adapté pour des vols embarqués sur des drones dans de la fumée. Quant à la caméra, elle est de la marque The Imaging Source modèle DMK33UP200 équipée d'une lentille grand angle MY110M. La lentille grand angle permet d'avoir une vue sur le panache entier.

3.3.2 Calibration caméra-LIDAR

Un autre intérêt d'utiliser le LIDAR en parallèle de l'image est la possibilité de projeter les points dans l'image. Comme illustré sur la figure 3.12 on peut ainsi déterminer les pixels correspondant au panache dans l'image, en se basant sur les données LIDAR. L'objectif de la calibration entre une caméra et LIDAR est de trouver la transformation entre le repère de l'image et le repère du LIDAR. Les travaux sur la calibration caméra-LIDAR ont été réalisés par Emna Adhar au LAAS lors d'un précédent stage, et repris pendant ce stage.
Pour trouver la transformation entre les deux capteurs, il faut trouver la transformation entre la caméra et une cible, ainsi que la transformation entre le LIDAR et cette même cible.

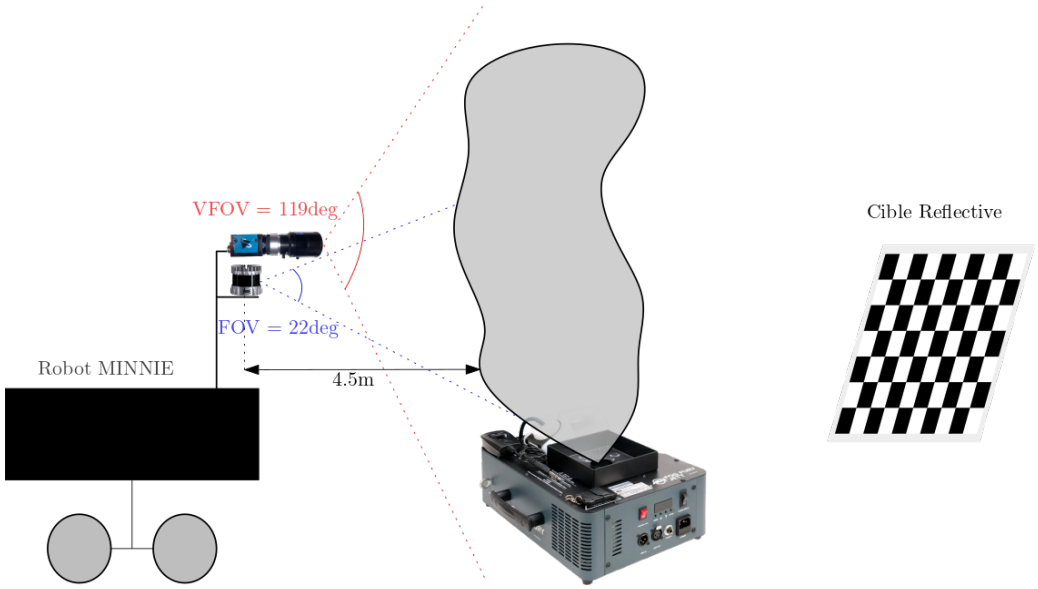


FIGURE 3.5 – Illustration du *setup* expérimental avec le Robot MINNIE.

3.3.2.1 Transformation Caméra-cible

On considère le modèle d'une caméra "pinhole", on doit donc trouver la transformation entre le repère image et le repère objet comme illustré sur la figure 3.6. En premier lieu, un algorithme d'inversion non linéaire permet d'estimer les coefficients de distorsion pour passer de S_d à S_c (repère image). On a ensuite la relation qui relie le repère image au repère de l'objet en coordonnées homogènes :

$$\begin{bmatrix} X_c \\ Y_c \\ 1 \end{bmatrix} = K \begin{bmatrix} R & | & t \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{3.1}$$

Il faut donc trouver les paramètres intrinsèques à la caméra (distances focales, position du centre du capteur) et extrinsèques pour reconstruire la transformation complète. On utilise un damier pour estimer d'abord la matrice de calibration :

$$K = \begin{bmatrix} f_x & 0 & x_c \\ 0 & f_y & y_c \\ 0 & 0 & 1 \end{bmatrix}$$

Puis, un algorithme DLT (Direct Linear Transform) pour estimer les paramètres extrinsèques de la caméra et estimer la position du repère du damier.

3.3.2.2 Transformation LIDAR-Cible

Le principe est plus simple que pour la caméra, puisque les données obtenues par le LIDAR sont déjà en 3D. Ainsi, il suffit de trouver les points correspondant au damier, et d'estimer le repère. Comme le damier est particulièrement réflectif, il est facilement repérable dans le nuage de point. Un algorithme de RANSAC est utilisé pour trouver le plan réunissant tous les points du damier, et à partir de ce plan est déduite l'origine du repère (coin en haut à droite du damier).
Une fois ces deux transformations obtenues, on peut en déduire la transformation entre le LIDAR et la caméra :

$$T_{c->L} = T_{c->0} \cdot T^T_{L->0} \tag{3.2}$$

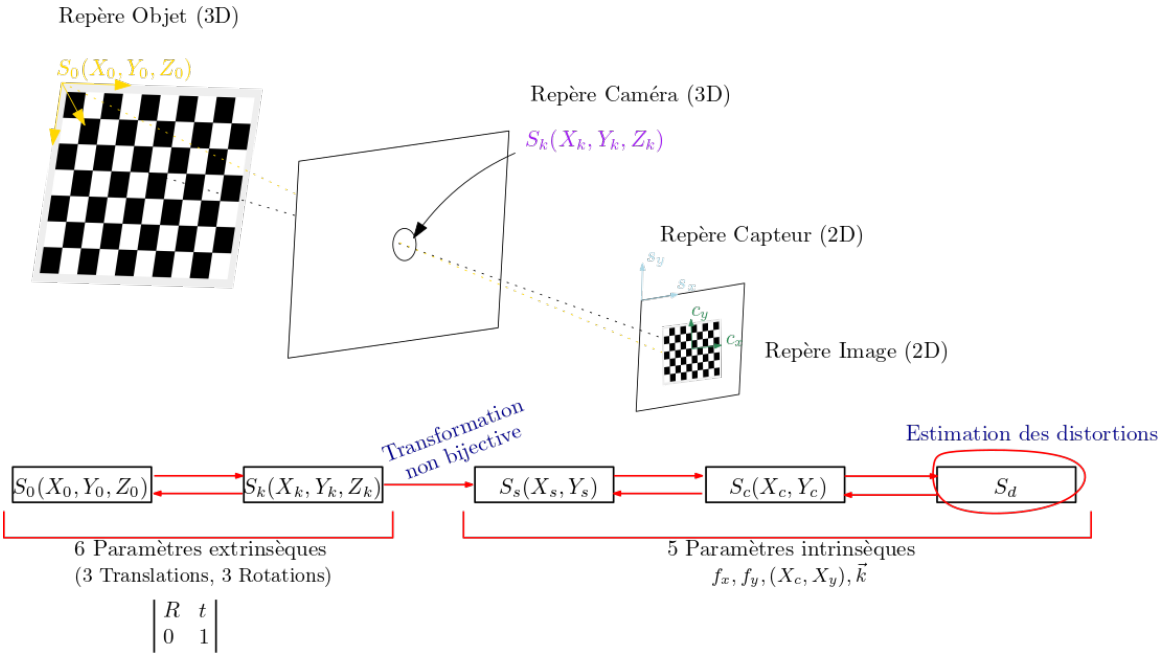


FIGURE 3.6 – Différents repères du modèle pinhole d'une caméra.

Avec $T_{c->0}$ et $T^T_{L->0}$ les transformations illustrées sur la figure 3.7.

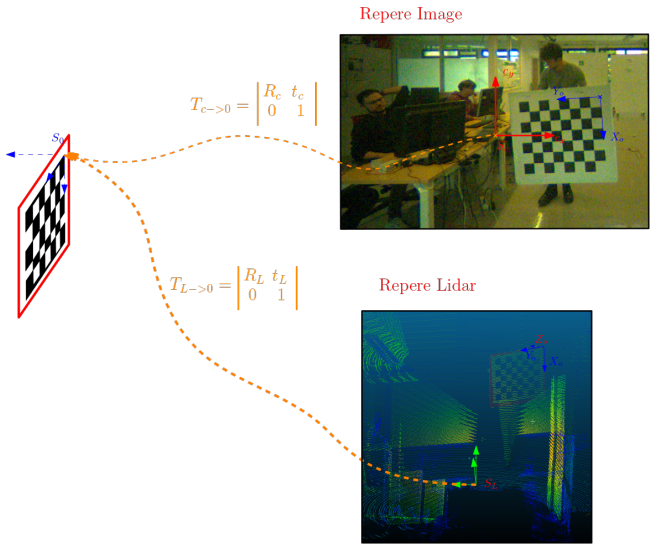


FIGURE 3.7 – Calibration LIDAR-Caméra avec un damier intermédiaire.

Ainsi, une fois la Transformation $T_{c->L}$ obtenue on projette tous les points du *Pointcloud* dans l'image correspondante à l'aide des paramètres intrinsèques. Les principales étapes de la projection sont décrites dans l'algorithme 1.

Algorithm 1: Projection d'un nuage de points LIDAR dans une image

Input: Nuages de points LIDAR, images, matrice intrinsèque **K**, distorsion **d**, transformation **T**

Output: Image avec points LiDAR projetés

images ← `loadAndSortImages()` ; `// Charger et trier les images`
nuages de points ← `loadAndSortLIDAR()` ; `// Charger et trier les fichiers LIDAR`
(P, I, img) ← `selectClosestData(images, nuages de points)` ; `// Associer nuage de points et image la plus proche temporellement`
P_h ← `toHomogeneous`(P) ; `// Convertir les points en coordonnées homogènes`
P_c ← `applyTransformation`(P_h, **T**) ; `// Transformer les points dans le repère caméra`
P_{img} ← `projectToImage`(P_c, **K**, **d**) ; `// Projeter les points sur le plan image`
I_n ← `normalizeIntensity`(I) ; `// Normaliser les intensités dans [0,255]`
`drawPointsOnImage`(img, P_{img}, I_n) ; `// Tracer les points sur l'image`
`displayImages`() ; `// Afficher l'image originale et celle annotée`

3.4 Données Obtenues

Le LIDAR Ouster OS2 permet de visualiser 2échos. En effet, lorsqu'un LIDAR émet un rayon lumineux il reçoit en général plusieurs échos et un filtrage (en général interne à l'appareil) permet de ne garder que les échos qui correspondent à un obstacle assez réflectif. La figure 3.8 illustre le principe des échos. On peut voir que deséchos intermédiaires sont en général moins intenses qu'un vrai obstacle.

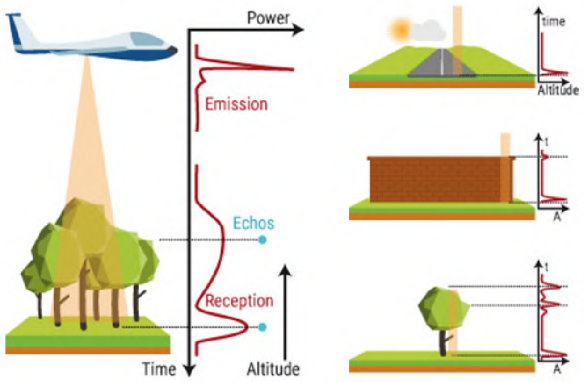


FIGURE 3.8 – Illustration des différentséchos qu'une même émission peut générer. Figure tirée de la thèse de Karl Montalban [4]

La figure 3.10 est un nuage de points tiré des enregistrements du LIDAR, on y retrouve les deux échos. La couleur des point évolue en fonction de la réflectivité de l'obstacle calculée à partir de l'intensité du signal reçu, en allant du rouge étant le moins intense vers le violet. On observe le panache de fumée au centre du nuage de points apparaissant avec une intensité assez faible par rapport à d'autres points de la frame. Derrière le panache, on voit distinctement la cible dont les bords sont réflectifs apparaissant en violet (donc la valeur maximale). En séparant les points du premier et du second écho, on se rend compte que tous les points correspondant au panache de fumée sont en réalité issus du second écho. Si on compare par rapport aux cas de la figure 3.3, on se retrouve clairement ici dans le cas (c) avec des points qui traversent le panache et atteignent la cible, mais aussi des retours de la surface du panache. Cette caractéristique est d'autant plus visible lorsqu'on extrait les points correspondants au panache

du reste du nuage de points (extraction expliquée dans la section 4) comme on peut le voir sur les figures 3.11b et 3.11c. On distingue sur la figure 3.11c des variations précises de la surface de la fumée. En analysant les valeurs de réflectivité, on constate un motif présent sur toutes les données observées : une variation de la réflectivité au sein du panache. On remarque une plus forte réflectivité vers le centre du panache, avec une décroissance qui semble varier en fonction de la distance au centre et à la source du panache. En sélectionnant une portion du panache selon l'axe z, on peut afficher les points du panache en fonction de leur intensité. La figure 3.9 présente à gauche le panache de fumée vu de face, avec une coloration en fonction de la réflectivité du retour. A droite, on sélectionne tous les points appartenant à une certaine tranche du panache (tous les points entre $z = 0.5 \pm 0.2m$ pour la tranche du milieu par exemple), on affiche en ordonnée leur intensité et en abscisse leur position selon l'axe transversal du panache. On remarque alors une sorte de répartition Gaussienne de l'intensité en fonction de la position, qui semble être validée lorsqu'on approxime ces données par une Gaussienne. On remarque que plus la tranche est proche de la source (environ $z = -0.7$) plus l'amplitude est grande et l'écart-type petit, et plus on s'éloigne de la source plus l'amplitude baisse et l'écart-type grandit.

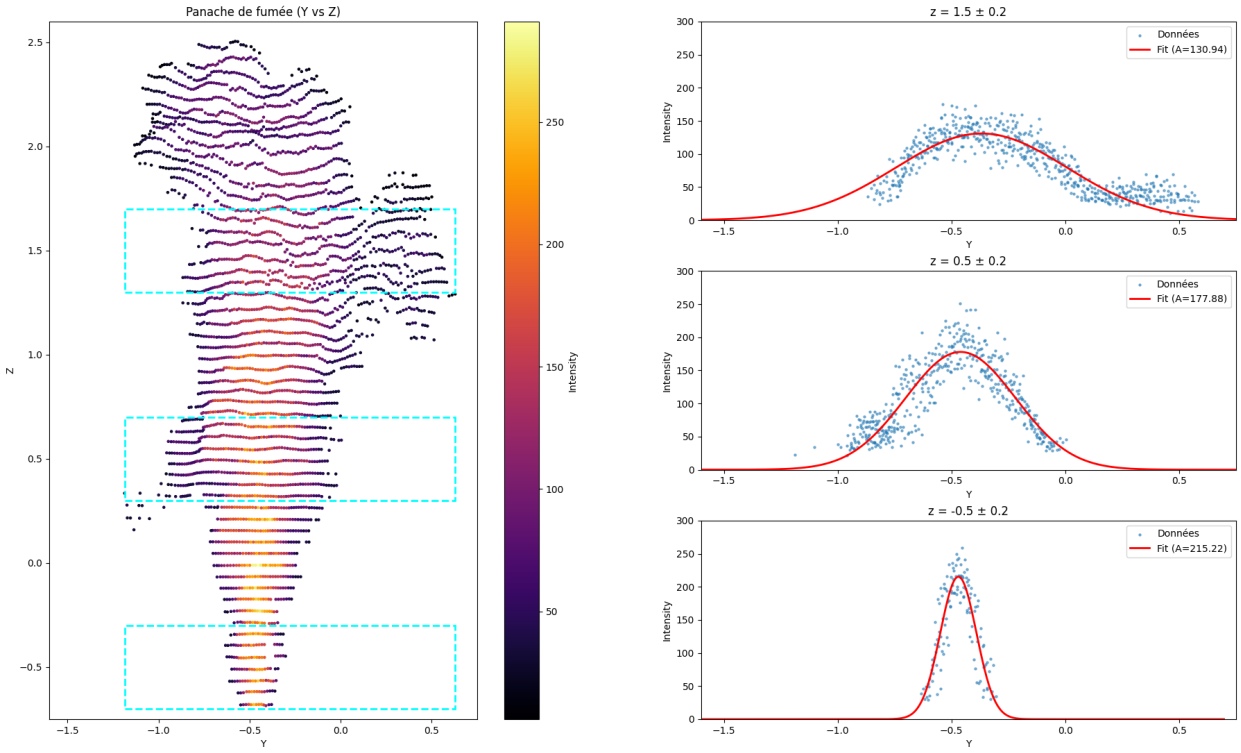


FIGURE 3.9 – Fit Gaussien de la répartition de la réflectivité en fonction de la hauteur z. On observe une variation par rapport au centre du panache et à la source.

En utilisant des modèles physiques tels que le modèle de dispersion de panache Gaussien [29], il pourrait être possible d'établir une corrélation entre la forme Gaussienne de l'évolution du panache et la concentration du panache. Cette concentration pourrait ensuite être une base pour la reconstruction 3D de la fumée puisque des informations à la fois sur la surface et la densité pourraient être extraites des données LIDAR.

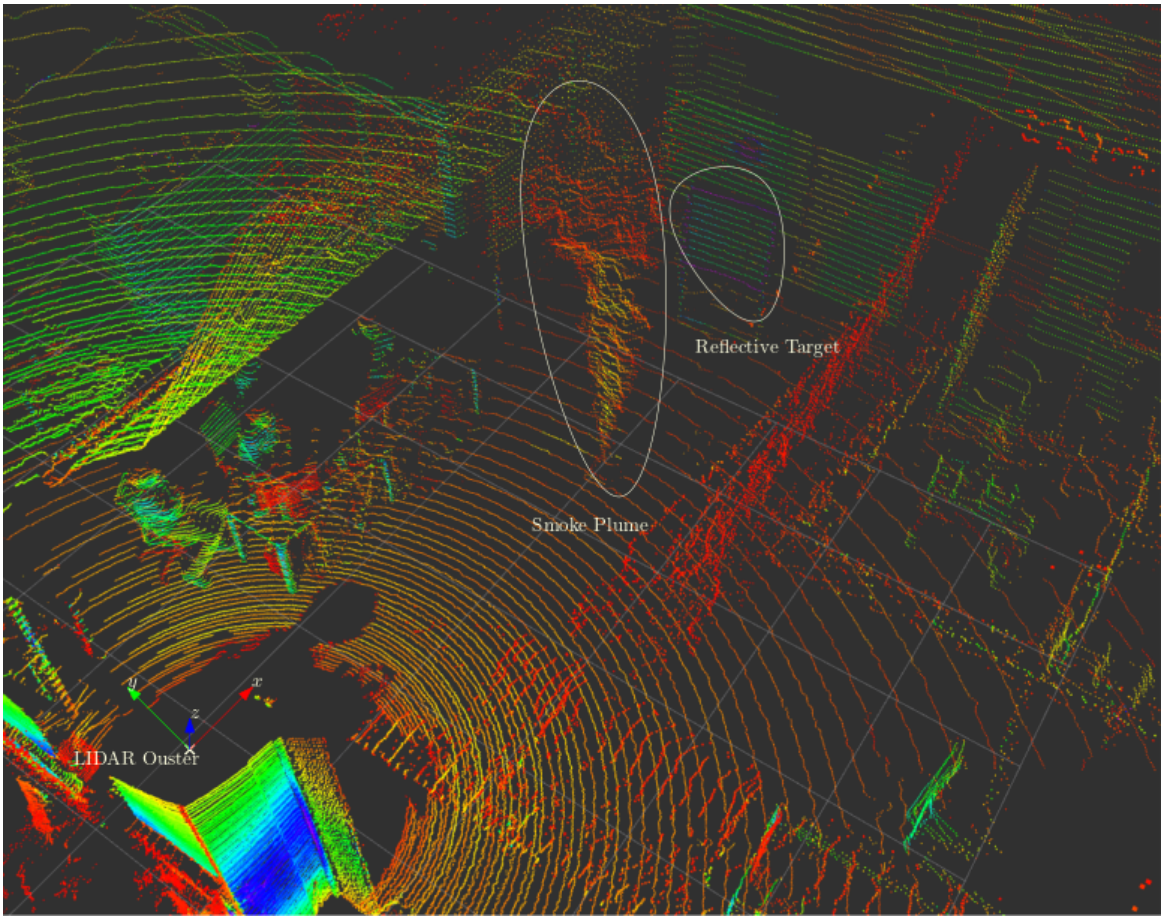
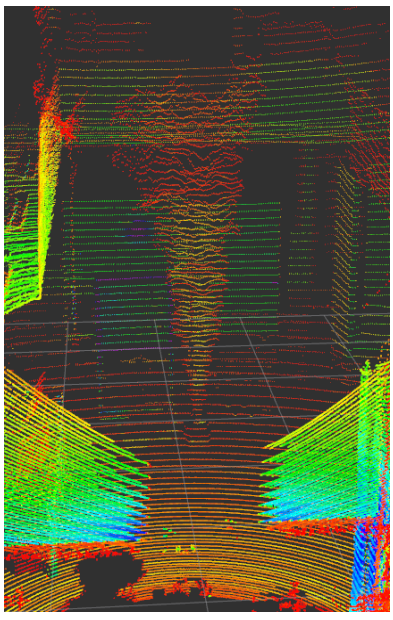
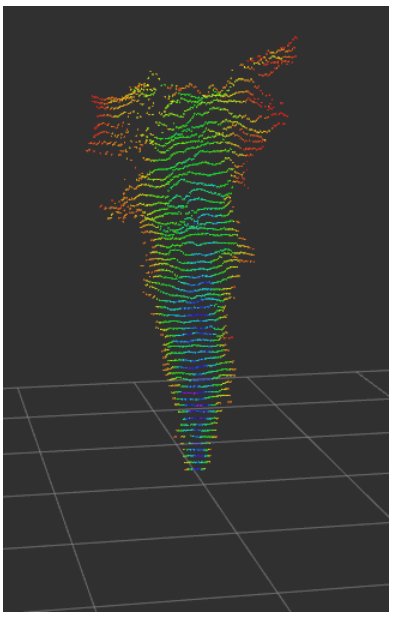


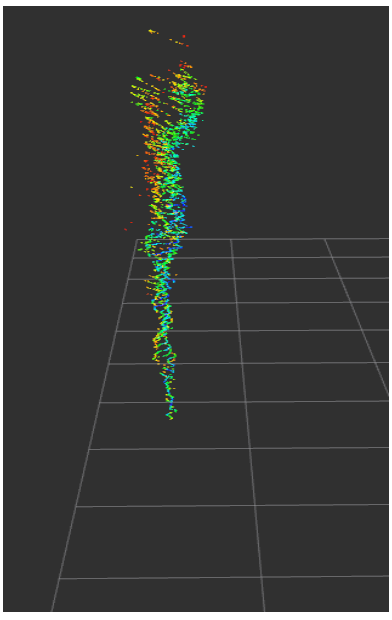
FIGURE 3.10 – Image tirées des données enregistrées par le LIDAR Ouster OS2 lors de l'expérience menée avec le robot MINNIE visualisée avec Rviz2.



(a) Panache de fumée vue de face dans la frame complète



(b) Panache de fumée extrait vue de face



(c) Panache de fumée extrait vue de profil

FIGURE 3.11 – Panache de fumée sous tous ses angles

3.4.1 Conclusions de l'expérimentation

Les données obtenues avec le LIDAR Ouster permettent de se rendre compte que la réflectivité des points transporte sûrement une information reliée à la densité du panache. Elles confirment également que les faisceaux ne permettent pas de voir l'intérieur du panache, mais seulement la surface d'où la nécessité d'utiliser plusieurs LIDAR pour avoir une idée plus précise de la géométrie globale du panache.
Les images obtenues sont quant à elles moins parlantes, à cause du fond non homogène et peu contrasté par rapport à la fumée, cette dernière apparaît très transparente et difficile à extraire avec du traitement d'image classique. Les points projetés aident à visualiser la forme du panache, mais on peut voir sur la figure 3.12 que la partie haute du panache n'est pas visible sur les nuages de points. Pour palier à ce problème dans le cadre de la création du dataset, un fond uniforme et assez contrasté par rapport au panache doit être utilisé. La luminosité doit aussi être améliorée pour faire ressortir la fumée dans l'image (sans la saturer).
Ces premières acquisitions ont mis en avant l'intérêt de la mesure de réflectivité par LIDAR et l'importance de contrôler l'environnement dans les images dans un premier temps.

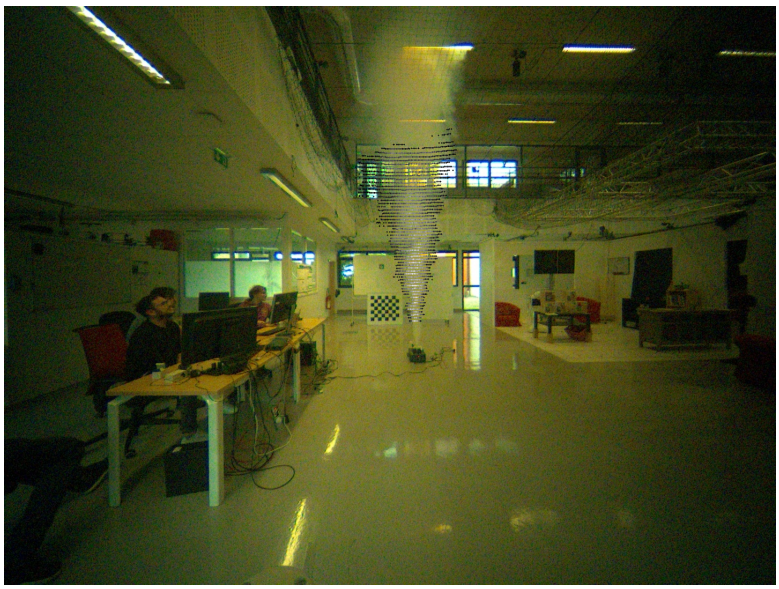


FIGURE 3.12 – Rendu de l'image de la fumée avec les points LIDAR projetés.

3.5 Acquisition Multi-LIDAR

Comme abordé dans la section 3.3 les données LIDAR obtenues avec un seul capteur sont assez plates puisqu'on ne perçoit que la surface visible par le LIDAR du panache. Afin d'obtenir des données complètes, il est donc nécessaire de faire des acquisitions avec plusieurs LIDAR répartis autour du panache. Deux tests ont été réalisés avec pour but de réaliser une acquisition multi-LIDAR, un en intérieur et un en extérieur.

3.5.1 Calibration des LIDAR

Pour être capable de visualiser la fumée en un seul nuage de point, il faut assembler les 3 nuages de points envoyés par les LIDAR. Pour ce faire il faut faire une calibration spatiale avant l'expérimentation, en choisissant un LIDAR de référence et en calculant les transformations entre les repères des deux

(a) *setup* for multi-LIDAR acquisition with 3 Livox MID-360 en intérieur

(b) *setup* for multi-LIDAR acquisition with 3 Livox MID-360 en extérieur

FIGURE 3.13 – Deux acquisition multi-LIDAR effectuées à l'intérieur et à l'extérieur avec deux types de fumée différente.

autres vers la référence. L'objectif d'une calibration entre deux LIDAR est de pouvoir réaligner un *pointcloud* généré par un LIDAR sur le *pointcloud* généré par le LIDAR de référence. Puisqu'on utilise deux capteurs différents, et que les points d'un *pointcloud* ne sont pas ordonnés, on ne connaît pas la correspondance entre les points d'un LIDAR avec ceux d'un autre. Il faut donc utiliser un algorithme ICP (Iterative Closest Point) pour la calibration.

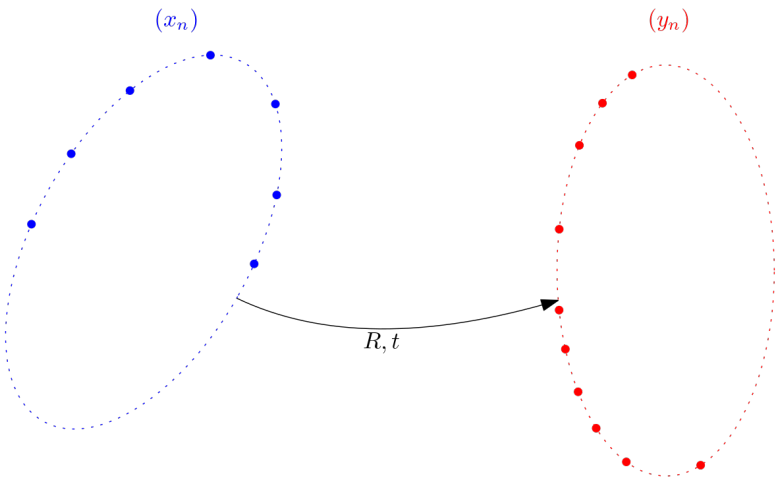


FIGURE 3.14 – Deux nuages de points théoriques représentant une partiellement la même ellipse. L'objectif est de trouver la transformation qui permette de recaler l'ellipse bleu sur la rouge sans correspondances entre les points.

En reprenant les notations de la figure 3.14 illustrant l'objectif d'un algorithme de recalage dans lequel on cherche à trouver les transformations R et t tel que :

$$\bar{x}_n = Rx_n + t \tag{3.3a}$$

$$\bar{x}_n = y_n \tag{3.3b}$$

Pour ce faire, on initialise à une certaine valeur R et t. On trouve ensuite une association entre les points des deux nuages de points avec un algorithme de **NearestNeigbors**. Puis on calcule la différence entre les deux nuages en comparant les associations de points avec l'erreur $\Phi = \Sigma\|y_i - \bar{x}_i\|^2$. En remplaçant avec l'équation 3.3b, on obtient alors $\Phi = \Sigma\|y_i^2 - Rx_i - t\|^2$. On minimise cette erreur après plusieurs itérations et en mettant à jour à chaque fois R et t jusqu'à atteindre une erreur minimale et trouver la transformation entre les 2 nuages de points et par conséquent entre les deux LIDAR. Pour implémenter la calibration, on utilise le package développé par Livox [30]. Comme pour tous les algorithmes de minimisation d'une erreur, la valeur initiale est déterminante dans la réussite du processus, il faut donc initialiser avec des valeurs pour R et t estimées proches de la réalité (une estimation de la rotation et translation à l'oeil suffit).

3.5.2 Premier test

Le premier test a été réalisé avec 3 LIDAR Livox MID-360 répartis au sol autour de la même machine à fumée utilisée pour le premier test. Il avait aussi pour objectif de tester ces LIDAR, particulièrement petits et donc très adaptés pour le vol en drone. Chaque LIDAR est connecté à un PC central en Ethernet. On peut voir sur le *setup* montré sur la figure 3.13a les 3 LIDAR répartis autour de la machine.

3.5.2.1 Données obtenues

Une fois les LIDAR calibrés et les nuages de points fusionnés, on peut visualiser le panache complet en 3 dimensions comme montré sur les figures 3.15 et 3.19. Les LIDAR MID-360, à la différence du OS2 utilisé pour les premiers tests, ont une méthode de scan très particulière non répétable. Cela se ressent lorsqu'on observe les points représentant le panache sur les figures 3.19, on ne distingue aucune "ligne" de points ou autre motif précis comme on pouvait observer sur la figure 3.11. Cela permet d'avoir des points plus aléatoires dans l'environnement pour en donner une bonne représentation avec un nombre de scans inférieur (maximum 200,000 points par secondes pour le Livox MID-360 contre maximum 2.62M points par secondes pour le OS2). On remarque néanmoins que le résultat est moins dense, mais la représentation reste satisfaisante avec près d'un millier de points correspondant à la fumée au maximum du panache. La réflectivité des points du panache semble aussi bien plus faible sur ce scan (un maximum de 6/255 de réflectivité obtenu contre des valeurs allant jusqu'à 250/255 avec le OS2). Cette différence est liée à la différence de qualité des deux LIDAR, mais aussi à la façon dont est calculée la réflectivité. On y distingue quand même des propriétés similaires sur les données, notamment avec des points plus réflectifs au centre du panache comme il avait été observé sur les figures 3.11. Ainsi, en reprenant la même logique que lors de la première expérience, on peut tracer la répartition de la réflectivité dans le panache pour une hauteur sélectionnée. Sur la figure 3.17 on peut estimer une Gaussienne pour représenter la répartition de la réflectivité dans le panache, et observer que les points les plus réflectifs sont en effet au centre du panache. Cette représentation Gaussienne peut être utile pour le vol de drones, avec une représentation simplifiée et formalisée de l'obstacle.

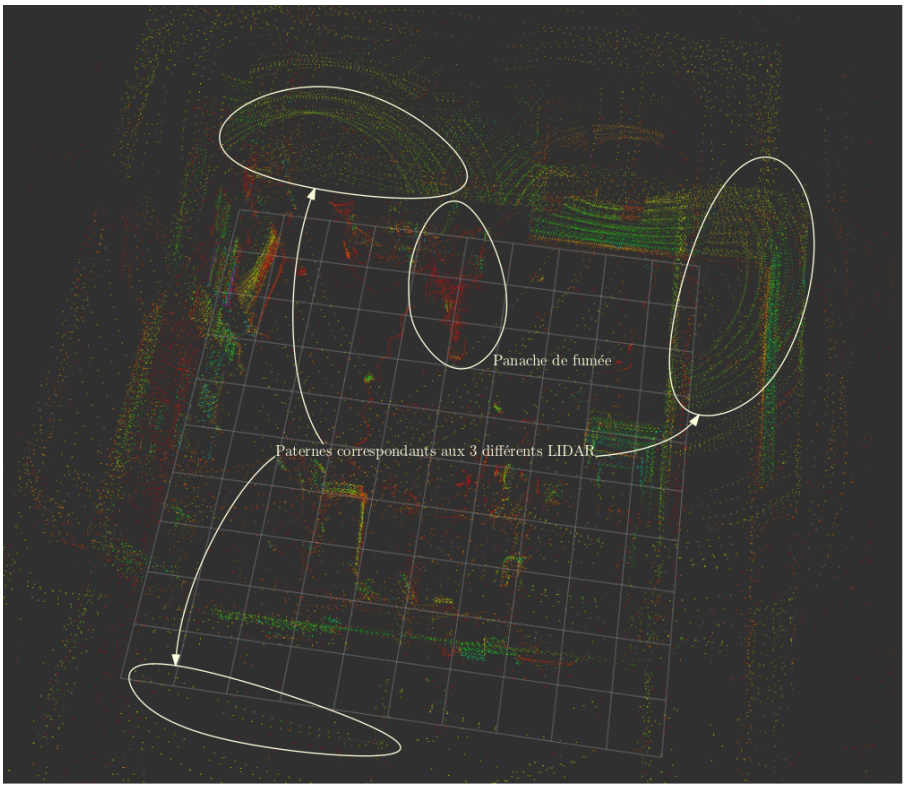
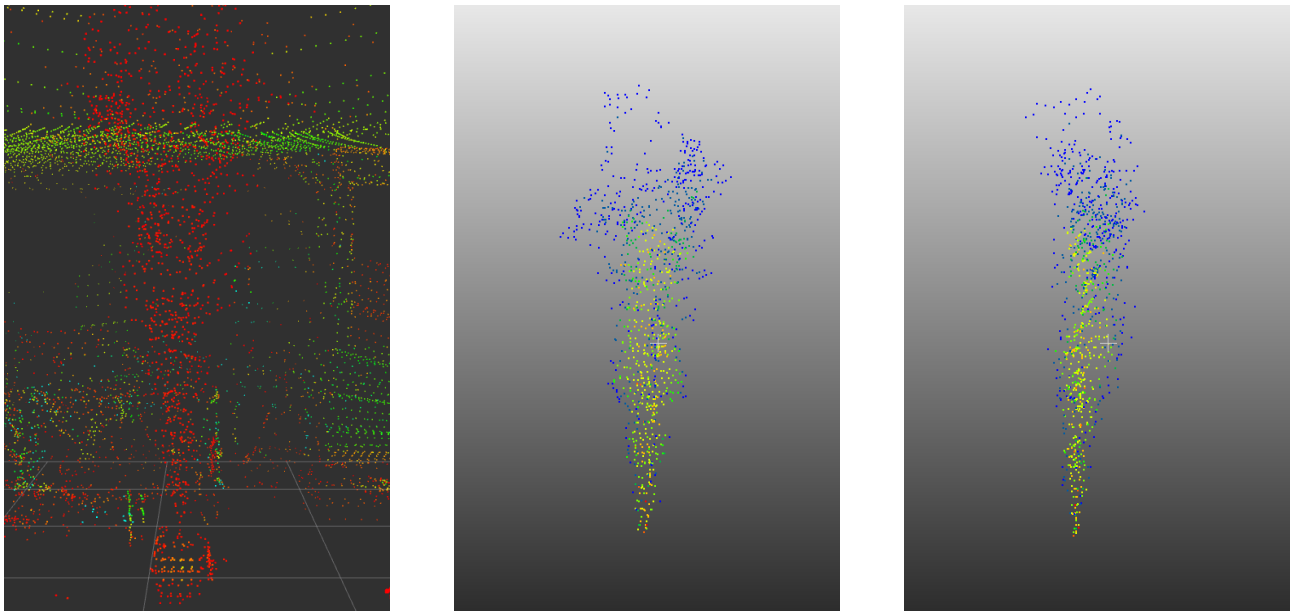


FIGURE 3.15 – Image tirée des données enregistrées avec 3 LIDAR Livox MID-360 calibrés. Visualisation sur Rviz2



(a) Panache de fumée vue de face dans la frame complète

(b) Panache de fumée extrait vue de face

(c) Panache de fumée extrait vue de profil

FIGURE 3.16 – Panache de fumée observé avec 3 LIDAR sous tous ses angles

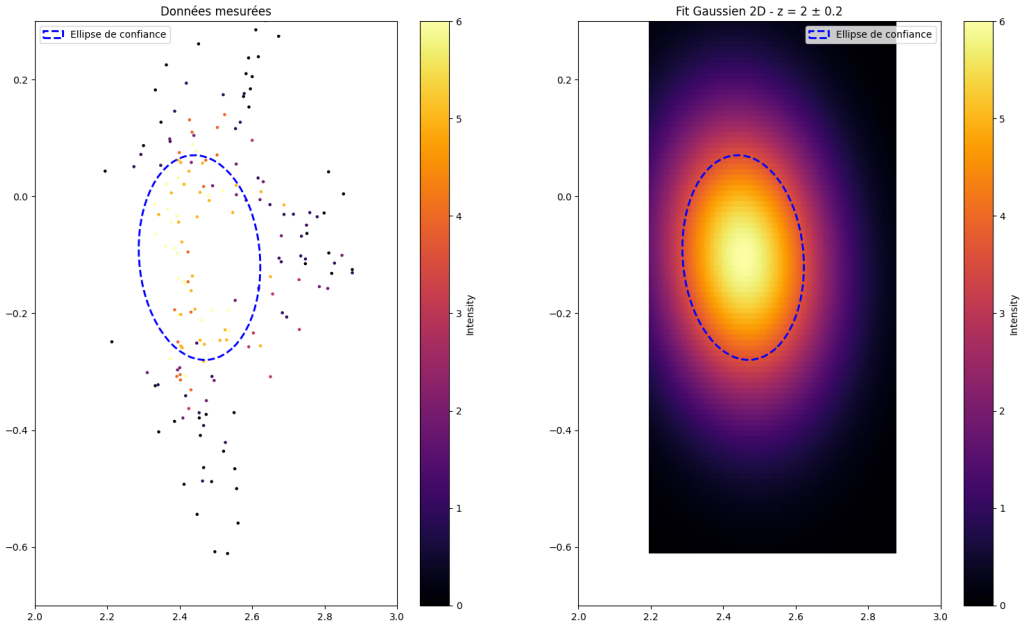


FIGURE 3.17 – Estimation d'une loi Gaussienne 2d en fonction de la réflectivité des points

3.5.2.2 Conclusions du test

Cette expérimentation avait pour but de tester les LIDAR Livox MID-360 (qui équiperont les drones du projet) en vérifiant que les données obtenues étaient satisfaisantes. En comparant avec la première expérience qui sert de mètre étalon, on se rend compte que les nuages de points obtenus sont moins denses et avec des données moins précises (notamment au niveau de la réflectivité). Il a été noté plus tard que la baisse de qualité dans les valeurs de la réflectivité pouvait aussi être due à la distance des LIDAR par rapport au panache. En effet, il est conseillé d'être à au moins 2 mètres d'un obstacle, car en dessous de ce seuil de nombreuses erreurs sont susceptibles d'apparaître selon la documentation du MID-360. Dans le cas de ce test, le panache était à une distance d'environ 2.3m de chaque LIDAR. Néanmoins, on constate la même répartition de la réflectivité dans le panache, renforçant ainsi l'hypothèse d'une corrélation entre la réflectivité et la concentration du panache. Ces données, peuvent aussi déjà servir pour commencer la réflexion sur leur intégration dans le pipeline de reconstruction, et pour évaluer les techniques de vol des drones (non traité dans ce stage).

3.5.3 Deuxième test

Le deuxième test réalisé reprend quasiment la même disposition que le premier, avec 3 LIDAR placés autour du panache de fumée mais cette fois-ci en extérieur. La raison pour laquelle cette expérience est faite à l'extérieur est d'une part pour pouvoir évaluer la qualité des données dans des conditions réelles plus proches de la réalité notamment avec la présence de vent. D'autre part, la machine à fumée est remplacée par un fumigène. Le panache généré est plus proche d'une fumée d'incendie car il s'agit d'une vraie combustion et non de vapeur d'eau. Le fumigène permet aussi d'avoir des séquences de données plus longues, ce qui peut permettre à un modèle d'apprentissage de mieux prédire des effets à long terme. Le fumigène est placé au centre des 3 LiDARs sur une plaque de plexiglas pour éviter

qu'une étincelle ne vienne brûler l'herbe et démarrer un incendie. Pour plus de sécurité et éviter un début d'incendie, les alentours de la plaque sont humidifiés. Les LIDAR sont placés cette fois-ci plus loin du panache, à une distance d'environ 5m. Les LIDAR sont aussi rehaussés afin de ne pas être gênés par l'herbe qui pourrait masquer les faisceaux laser. Si le *setup* est globalement le même, le fait d'être à l'extérieur rajoute de nombreuses contraintes matérielles. Pour alimenter et connecter les LIDAR on utilise le conteneur associé à l'arène de vol extérieur évoquée en introduction **??**. Depuis ce conteneur, les câbles de données et d'alimentation sont tirés jusqu'aux LIDAR. Le *setup* est visible sur la figure 3.13b.

3.5.3.1 Données Obtenues

Malgré la relative simplicité du *setup* sur le papier, les longues distances de câble et le fait d'être à l'extérieur rendent les choses plus compliquées à gérer. L'utilisation de fumigènes force aussi à prendre plus de mesures de sécurité ce qui alourdit un peu plus la tâche. Cependant, des données utiles ont quand même pu être enregistrées et donnent une bonne idée de l'impact que peut avoir un environnement non contrôlé (avec du vent par exemple) sur la fumée et comment cela se répercute sur les données. Aussi, l'augmentation de la distance entre les LIDAR et la fumée se rapproche plus d'un réel cas d'utilisation avec des drones (il est peu probable qu'une flotte de drones puisse rester coordonnée et stable à moins de 2m d'un panache généré par un incendie). Ainsi, l'impact de ces deux principaux changements est clairement visible sur les données.
Les nuages de points semblent moins denses comme on peut le remarquer sur la figure 3.19a, mais on distingue quand même assez bien la forme de la fumée et son déplacement. Des trous sont visibles dans le panache comme sur la figure 3.19b, si la raison n'est pas claire, il peut tout de même s'agir d'une fumée trop dispersée, ou même des phénomènes d'absorption ou de out-scattering comme évoqué dans la section 3.1.2. Il existe aussi la possibilité que certains morceaux de la fumée aient été supprimés à cause du filtrage lors de l'extraction du panache (détaillé dans la section 4.1). On remarque aussi, que même lorsque le panache est extrait de la frame complète, la réflectivité du panache est très faible sûrement due à la dissipation rapide de la fumée à cause du vent rendant le panache très diffus et donc très peu réflectif. On ne distingue pas de motif comme sur les précédentes expérimentations avec des points plus réflectifs au centre du panache par rapport aux extrémités.
On note tout de même des points plus réflectifs proches de la source, où la fumée est plus concentrée confirmant la corrélation entre la concentration de la fumée et la réflectivité des points perçus par le LIDAR.

3.5.3.2 Conclusions de l'expérimentation

Les données recueillies sont intéressantes de par leur durée, et leur caractère plus représentatif d'une situation naturelle bien que ces mêmes propriétés dégradent la qualité des points et rendent plus complexe l'estimation précise du comportement de la fumée. Si une calibration spatiale et une synchronisation temporelle entre chaque LIDAR avec une caméra peuvent être réalisées, une fusion de données permettant d'utiliser à la fois les caméras et LIDAR pourrait être utilisée afin de mieux percevoir les contours de la fumée.

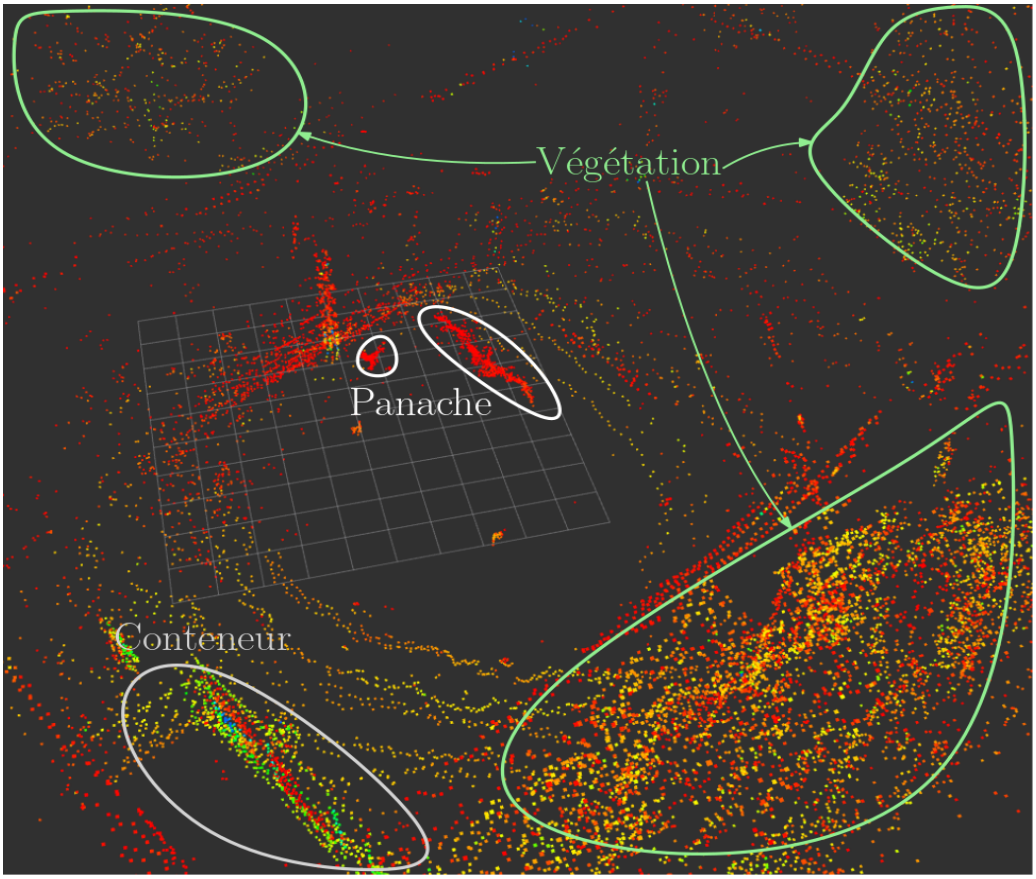
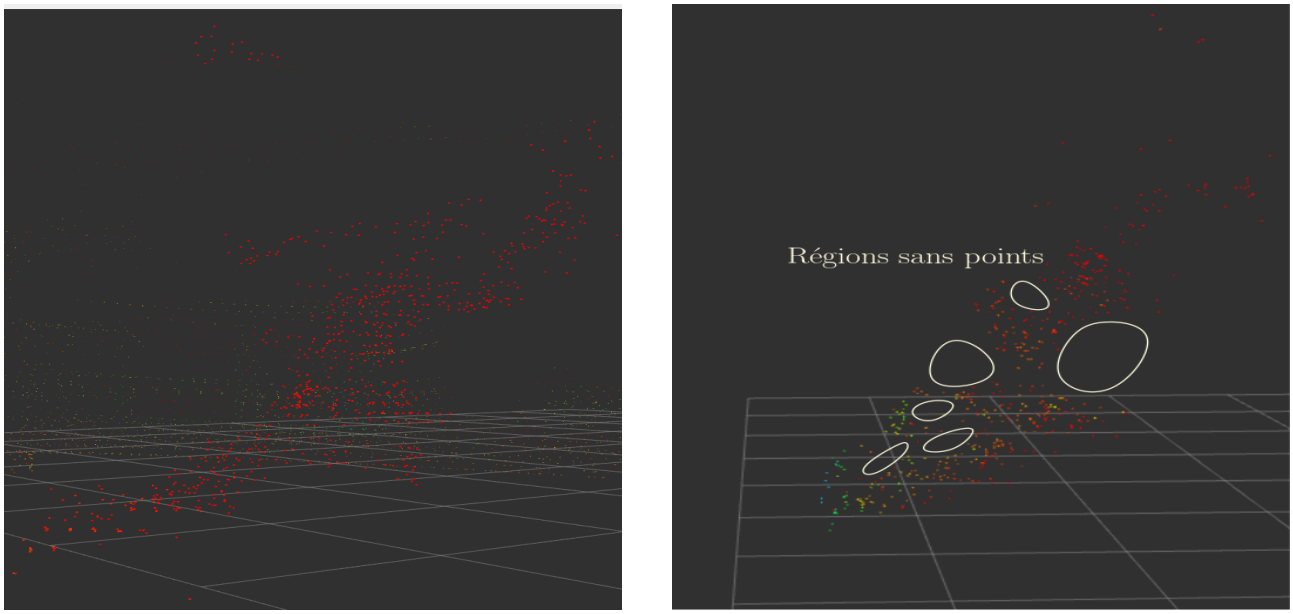


FIGURE 3.18 – Image tirée des données enregistrées avec 3 LIDAR Livox MID-360 calibrés en extérieur avec un panache généré par un fumigène. Visualisation sur Rviz2



(a) Panache de fumée dans la frame complète

(b) Panache de fumée extrait

FIGURE 3.19 – Panache de fumée observé avec 3 LIDAR sous tous ses angles

3.6 Acquisitions multi-vues et multi-LIDAR

Afin de construire un *dataset* cohérent pour développer un pipeline de reconstruction 3D par *3D Gaussian Splatting* , on cherche donc à avoir des données images et nuages de points de panaches de fumée. Ces dernières peuvent être utilisées si les capteurs sont calibrés entre eux (transformation spatiale disponible pour pouvoir assembler et fusionner les données) et synchronisés temporellement. Pour le *setup* complet, on s'inspire du *dataset* Scalarflow déjà évoqué plus tôt dans le rapport, qui sert de référence dans la communauté. Le *dataset* Scalarflow a été réalisé en environnement contrôlé avec 5 caméras placées en demi-lune autour du panache de fumée [22]. Pour plus de flexibilité dans l'optique d'ajouter des données prises en extérieur, on cherche à avoir un *setup* relativement facilement déployable aussi sur des grandes distances (10-20m entre la caméra/LIDAR la plus éloignée du centre de contrôle). Pour la prise de données, il a été décidé d'utiliser 5 caméras et 3 LIDAR. Ainsi, on s'éloigne un peu du cas de la prise de données par drones, mais cela permet d'avoir des données propres pour le développement du pipeline dans un premier temps. D'autre part ces données seront rendues publiques, et potentiellement utiles à la communauté en venant compléter Scalarflow.

3.6.1 Choix du setup

Comme indiqué, on souhaite que ce *setup* soit adaptable à la fois pour des conditions intérieures avec des courtes distances, et pour un environnement extérieur, où les tests menés (décrits dans la section 3.5.3) ont montré que la longueur des câbles peut atteindre jusqu'à 15m (voire plus). Ces grandes distances soulèvent alors 2 problèmes principaux :

- **L'alimentation** ;
- **Le transfert des données.**

Pour le premier point, l'utilisation d'une batterie est une solution directe et relativement peu encombrante (il faut tout de même penser à l'étanchéité en conditions extérieures). La principale problématique est de gérer le transfert de données. Dans le cas du LIDAR, on passe par une connexion Ethernet, dont les câbles peuvent atteindre aisément des distances de 20m donc ce n'est pas un problème. En revanche des caméras USB3.0 sont à disposition pour ce projet, or les câbles USB3.0 doivent être rallongés via des répéteurs pour atteindre ces distances (au minimum un tous les 5 mètres) augmentant la durée de communication et le risque de pertes de données.
De plus, si les appareils connectés en Ethernet supportent en général des protocoles de synchronisation fonctionnant en réseau, ce n'est pas le cas des appareils USB3.0. Le seul moyen de synchroniser ces appareils est d'utiliser des trigger hardware en passant par des PIN GPIO.
Pour palier aux problèmes de distances de câbles de transferts de données, et de synchronisation temporelle, la solution proposée est d'associer un ordinateur embarqué à chaque caméra. Ce dernier, équipé de ports Ethernet, de ports USB3.0 ainsi que de GPIO est idéal pour répondre à toutes ces contraintes puisqu'il répond à toutes les problématiques : le connecteur Ethernet permet de le synchroniser et de l'alimenter via PoE (Power Over Ethernet), le port USB3.0 permet l'alimentation et le transfert des données de la caméra et ses PIN permettent d'activer le trigger de la caméra. Un inconvénient de cette solution, est l'ajout de 5 câbles Ethernet (un par caméra) à brancher sur un même PC central en plus des 3 LIDAR déjà présents. Deux solutions sont possibles, ajouter une carte Ethernet au PC central, ou passer par un switch. Pour plus de simplicité, on décide de passer par un switch. Ce dernier doit permettre l'alimentation par PoE on veut utiliser l'alimentation via Ethernet. La figure 3.20 présente un schéma de principe du *setup* réalisé avec les liens entre ses composants.

3.6.2 Synchronisation temporelle

Pour observer un phénomène physique, il s'agit d'observer l'évolution du système au cours du temps. Si plusieurs capteurs sont utilisés, ils doivent être synchronisés pour pouvoir fusionner les données en

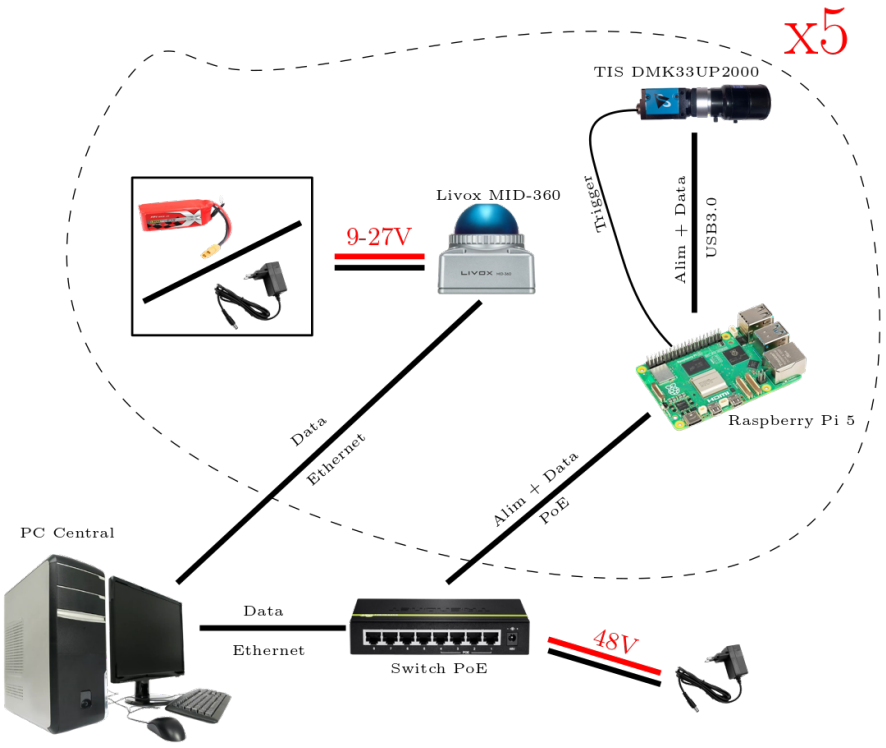


FIGURE 3.20 – Schéma de principe du *setup* final pour une acquisition multi-vues/multi-LIDAR

étant certains que c'est le même phénomène par chaque capteur. D'autant plus pour un panache de fumée qui est très sensible à la moindre perturbation.

3.6.2.1 Trigger les caméras

Dans un *setup* avec seulement des caméras comme Scalarflow, afin de synchroniser les prises de vue, [22] utilise un même trigger hardware pour toutes les caméras. Dans notre cas, comme on veut un *setup* adaptable aux grandes distances, il semble que la solution d'un câble pour trigger toutes les caméras ne soit compliquée à gérer. On choisit donc de paralléliser les signaux de Trigger, c'est-à-dire que chaque Raspberry Pi associée à une caméra aura la mission d'envoyer le signal. Pour que tous les signaux soient envoyés en même temps, il faut donc d'une part que toutes les Raspberry Pi soient synchronisée (c'est-à-dire qu'elles aient la même référence temporelle), et qu'elles reçoivent une instruction indiquant à quel moment le signal doit être lancé. Tout ceci, toujours en prenant en compte qu'il faut aussi synchroniser les prises de vues avec les scans LIDAR. Ces derniers fonctionnent à 10Hz et on un temps de scan d'un peu moins de 0.1s. On souhaite pouvoir prendre 3 photos pour chaque scan, donc une fréquence de 30Hz pour les caméras. Basé sur ces contraintes, on construit l'algorithme 2. On se base sur l'écart réel entre 2 scans LIDAR pour estimer quand est-ce qu'il faut envoyer un signal pour trigger les caméras. L'algorithme 3 décrit comment obtenir la valeur moyenne des timestamps des 3 LIDAR, et le calcul de l'écart entre 2 scans.
Pour que cet algorithme fonctionne bien, il faut donc synchroniser tous les appareils pour qu'ils aient tous la même référence temporelle. On choisit une horloge centrale sur laquelle tous les autres appareils (Raspberry Pi et LIDAR) pourront se synchroniser.

Algorithm 2: LiDARs - Cameras synchronisation using PTP and Raspberry Pies

Inputs :
$\Delta T_{\text{camera}_j}$: Time Exposure of the camera i
N : number of LiDAR
K : number of cameras S

Temporal Variables :
$T_{\text{ts\_lidar}_i}$: LiDAR timestamp i
T_{capt} : Mean Timestamp between each LiDAR's timestamp
ΔT_{LiDAR} : Mean duration between 2 scans
t : Current Time of the computer

Outputs : Trigger signal sent from each raspberry Pi to their corresponding camera

Initialisation :
Start PTP protocol from the computer (master) to synchronise the devices clock (LiDARs and Raspberry Pies)
Wait until time is synchronised (indicated in LiDAR message)
For each LiDAR read $T_{\text{ts\_lidar}_i}$
for $t \leftarrow t_0$ **to** $t_0 + 1s$ **do**
  $T_{\text{capt}}, \Delta T_{\text{LiDAR}} \leftarrow$ `get_mean_timestamp(`t, N`)`
end
Starting Acquisition :
while *Aquisition active* **do**
  for $j \leftarrow 1$ **to** K **do**
    Compute $T_{\text{trigger}_j} \leftarrow T_{\text{capt}} - \frac{\Delta T_{\text{camera}_j}}{2} + \Delta T_{\text{LiDAR}}$ `// Estimate next time to Trigger each camera corresponding to its time of exposure`
  end
  $T_{\text{capt}}, \Delta T_{\text{LiDAR}} \leftarrow$ `get_mean_timestamp(`t, N`)`
  for $j \leftarrow 1$ **to** K **do**
    while `PTP_synced_clock()` $< T_{trigger_j}$ **do**
      Wait
    end
    Send Trigger `// This must be done in parallel thanks to the PTP calibration between the Raspberries`
  end
end

Algorithm 3: get_mean_timestamp

Inputs : t : Current Time of the computer
N : number of LiDAR

Outputs : T_{capt} : Mean Timestamps between all LiDARs
ΔT_{LiDAR} : Mean duration between 2 scans

for $i \leftarrow 1$ **to** N **do**
  Read $T_{\text{ts\_lidar}_i}$ from ROS2 message of LiDAR i `// recall that LiDARs and computer's clocks are synchronised`
end
Compute $T_{\text{capt}} \leftarrow mean(T_{\text{ts\_lidar}_i})$ `// Compute mean of timestamps`
Compute $\Delta T_{\text{LiDAR}} = T_{\text{capt}}(t) - T_{\text{capt}}(t - \Delta T_{\text{LiDAR}})$
Return :
$T_{\text{capt}}, \Delta T_{\text{LiDAR}}$

3.6.2.2 Protocole de synchronisation

Un protocole de synchronisation est un protocole permettant de synchroniser une horloge locale avec celle d'un serveur relié à une horloge physique (en général une horloge atomique). Par exemple, le protocole NTP (Network Time Protocol) permet de se synchroniser à un serveur accessible dans le monde entier. Ces protocoles de synchronisation fonctionnent en général avec des protocoles de communication UDP d'où la nécessité d'une connexion Ethernet. Pour une synchronisation en référence à une horloge physique locale, on utilise le protocole de synchronisation PTP (Precision Time Protocol). Le principe est le même, mais l'horloge de référence peut être celle de l'ordinateur central par exemple. Le mécanisme de synchronisation repose sur un système hiérarchique maître-esclave. Quatre types de messages sont envoyés : *Sync*, *Follow up*, *Delay Req*, et *Delay Res*. Le mécanisme est illustré sur la figure 3.21.

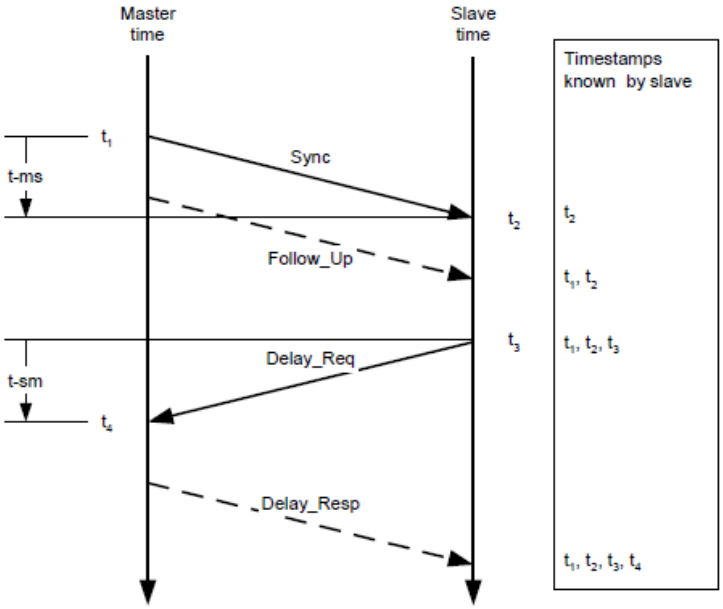


FIGURE 3.21 – Illustration du mécanisme du PTP

L'horloge esclave peut alors calculer le délai de transmission et l'offset entre les deux horloges et ajuster

son propre datage :

$$TransmissionDelay = \frac{(t4 - t1) - (t3 - t2)}{2} \tag{3.4}$$

$$Offset = (t2 - t1) - TransmissionDelay = \frac{(t2 - t1) + (t3 - t4)}{2} \tag{3.5}$$

Dans notre cas, le maître est le PC central, il est lui même connecté et synchronisé au serveur temporel du LAAS. L'architecture de synchronisation temporelle est décrite sur la figure 3.22, avec les Raspberry Pi et LIDAR synchronisés avec l'horloge du PC central.

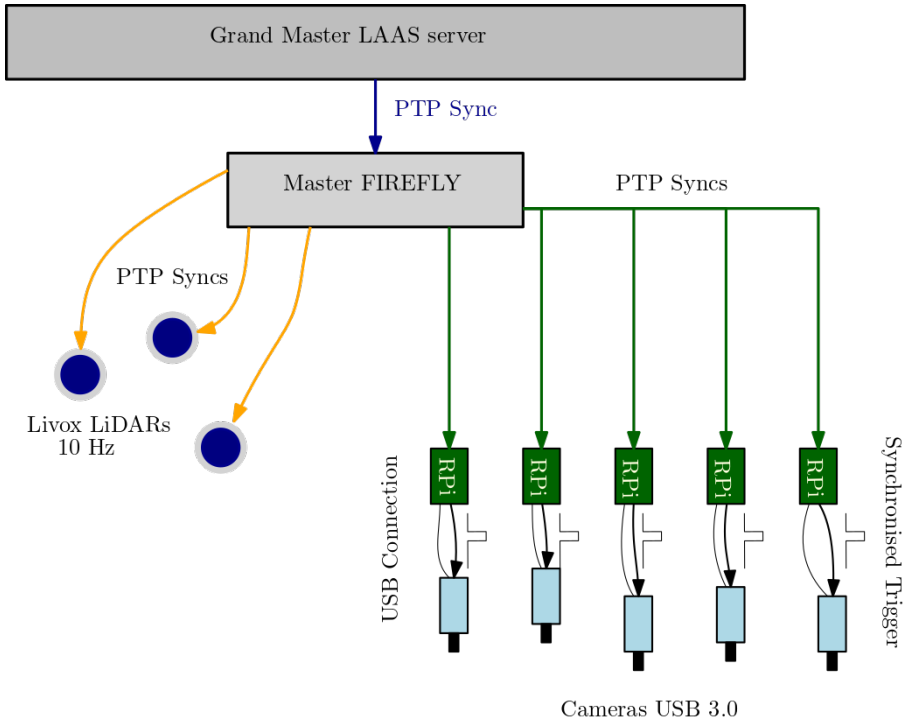


FIGURE 3.22 – Architecture de la synchronisation via PTP

3.6.2.3 Hardware ou Software

Lorsqu'un message UDP est reçu par un appareil, un timestamp (ou horodatage en français) est automatiquement généré. Il peut l'être de deux manières différentes, soit un timestamping hardware, soit un software. Le premier permet d'atteindre une précision de synchronisation de l'ordre de la dizaine de nanosecondes avec une horloge physique directement sur la carte Ethernet associée au port qui reçoit les paquets, tandis que le second génère le timestamp en passant par un driver qui récupère la date de l'appareil lui-même (celui qui reçoit). Cette petite manœuvre lui coûte une perte de précision, qui est alors de l'ordre du millier de nanosecondes, autrement dit quelques microsecondes. Le timestamping hardware est donc bien plus précis, mais il demande de synchroniser aussi l'horloge de la carte Ethernet avec celle de l'appareil. De plus, en mode hardware, il faut aussi s'assurer que le switch soit synchronisé. Pour éviter de trop complexifier la mise en place du protocole, il est choisi d'utiliser le timestamping software, déjà bien suffisamment précis.

Pour appliquer le protocole de synchronisation, on utilise le package ptp4l qui offre de nombreuses possibilités de configuration [31]. Les configurations utilisées pour les Raspberry Pi et pour les LIDAR sont disponibles en annexe 5.

3.6.2.4 Test de la synchronisation des caméras

Pour vérifier de manière certaine que la synchronisation des Raspberry Pi est bonne, il faut vérifier que les signaux de trigger sont envoyés exactement en même temps. Pour cela, on applique l'algorithme 2 présenté précédemment, estimant quand envoyer les signaux à travers les PIN des Raspberry Pi. Pour ce test, on prend 2 Raspberry Pi, dont on va observer les signaux de trigger à l'oscilloscope. Si la synchronisation fonctionne comme prévu, on devrait voir se confondre les deux signaux sur l'oscilloscope. Le résultat de ce test est disponible sur la figure 3.23. On voit bien que les signaux sont bien alignés, ce qui confirme que la synchronisation fonctionne bien.

FIGURE 3.23 – Preuve de la synchronisation entre deux Raspberry Pi en observant à l'oscilloscope des signaux envoyés à des instants précis

3.6.3 Calibration Multi-Vues

Afin de pouvoir utiliser les données images indépendamment des données LIDAR, par exemple pour reconstruire la scène en 3D en utilisant un algorithme SfM (Structure From Motion), il faut connaître la position relative entre les caméras. Il convient alors de réaliser une stéréo-calibration qui permet de calculer la transformation d'une caméra par rapport à une autre.

Lorsqu'on effectue une calibration de caméra avec un damier comme détaillé en section 3.3.2, on obtient aussi les transformations du repère caméra vers le repère du damier. Ainsi, en réalisant la calibration de nos 2 caméras en même temps avec le même damier, on obtient les transformations (R_1, t_1) et (R_2, t_2) vers le damier. Ainsi, soit (R, t) les transformations relatives de la caméra 1 vers 2, on a alors :

$$R = R_2 R_1^T \tag{3.6a}$$

$$t = t_2 - R t_1 \tag{3.6b}$$

La librairie openCV permet de facilement réaliser cette stéréo-calibration. Cependant, la calibration ne nous donne pas la distance par rapport à un objet quelconque. Ici cela fonctionne car on est capable d'estimer la distance au damier grâce à sa régularité et à ses dimensions connues.

3.6.4 Écriture des images

Un des problèmes qui se pose en stockant les données reçues dans une Raspberry Pi est sa vitesse d'écriture. En effet, il faut que les données soient stockées plus rapidement que la vitesse d'arrivée des données. Cela dépend de plusieurs paramètres comme la taille de l'image, son encodage et la fréquence de capture des images.
Par exemple, pour une image de taille 1920 × 1200, avec un encodage `GRAY8` (niveaux de gris, codé sur 8 bits) et une fréquence de 30 images par seconde, on obtient une vitesse d'écriture requise de 553 Mbit/s, soit 69 MB/s (en faisant bien la distinction entre *Mbits* en français et *MBytes* en anglais, qui signifie en fait Moctets).

$$requiredSpeed = \frac{width \times height \times encodingSize \times channels \times frequency}{8} \tag{3.7}$$

En général, une carte microSD a une vitesse d'écriture maximale d'environ 90 MB/s, ce qui est en théorie suffisant. En revanche, c'est une capacité maximale établie dans certaines conditions spécifiques, donc elle n'est pas garantie. De plus, comme on peut le voir avec l'équation ci-dessus, la vitesse requise dépend de l'encodage mais aussi du nombre de canaux (par exemple pour l'encodage `RGB8`, on encode un canal sur 8 bits, mais il y en a 3 au total : Rouge, Vert et Bleu).
Ainsi, si l'on veut passer en couleurs, on doit multiplier cette vitesse par 3, et on obtient une vitesse requise de 207 MB/s, ce qui surpasse largement la capacité d'écriture d'une carte microSD.
Donc, pour booster un peu les capacités des Raspberry Pi, on leur ajoute un module `M2 PoE Hat +` qui permet d'ajouter un disque NVMe ayant des capacités de stockage et une rapidité d'écriture bien plus élevées qu'une carte microSD. Cependant, malgré une vitesse d'écriture de 486 MB/s avec le disque NVME, des pertes de données apparaissent toujours. Il s'agit en réalité de la fonction utilisée avec OpenCV lors de l'acquisition, qui met trop de temps à enregistrer les images et bloque le processus. Ainsi, pour palier à ce problème, on enregistre les données dans un processus parallèle pour éviter tout blocage. Cela résout le problème, et il n'y a plus de pertes de données.

3.6.5 Intégration ROS2

ROS2 (Robot Operating System 2) est un middleware utilisé pour le développement de systèmes robotiques. Il permet de transmettre des informations entre plusieurs programmes appelés des nœuds via différents moyens standardisés (topics, services, actions). Ainsi, pour de l'expérimentation multisensorielle, c'est un outil qui peut s'avérer très pratique, surtout lorsque les différents capteurs sont reliés d'une manière ou d'une autre (par exemple lorsqu'une synchronisation est nécessaire). De plus, ROS2 offre des outils de visualisation tels que Rviz2 ou RQT facilitant la compréhension globale du système.

3.6.5.1 Nodes principaux

Pour mener à bien l'acquisition multi-vues plusieurs programmes sont nécessaires :

- **livox_ros_driver2 :** Ce package directement développé par l'entreprise Livox permet de lire les données envoyées par les LIDAR. Il publie ces données sur les topics de la forme `/livox_lidar/ip_lidar`, avec `ip_lidar` l'adresse IP permettant de se connecter au LIDAR. Ce package est écrit en C++.
- **pointcloud_fusion :** Ce package prend en paramètre un fichier de configuration contenant les matrices de transformation entre les divers LIDAR branchés. Il s'abonne aux topics `/livox_lidar/ip_lidar` et s'occupe de fusionner les nuages de points en les alignant pour recréer la scène 3D vue par tous les capteurs. Il publie le nuage complet sur le topic `/full_cloud`. Ce package est écrit en C++.

— **smoke_extraction :** Ce package a pour but d'extraire le panache de fumée du reste de la scène 3D en temps réel. Le détail est décrit dans la section 4.1. Il s'abonne au topic `/full_cloud` et publie sur le topic `/smoke_pc` le panache extrait. Ce package est écrit en C++.
— **trigger_estimation :** Ce package a pour but d'envoyer le timestamp exact où le trigger des caméras doit être envoyé. Il s'abonne aux topics `/livox_lidar/ip_lidar` afin de récupérer les timestamps des scans LIDAR et en calculer la valeur moyenne et le "delta" entre deux scans. Grâce à ce delta, il estime quand le prochain scan aura lieu, et publie le timestamp sur le topic `/trigger_time`. Ce package est écrit en C++.
— **trigger_camera :** Ce package fait parti d'un autre workspace ROS2, situé dans la Raspberry Pi. Il s'abonne au topic `/trigger_time` et envoie un signal de trigger à travers les PIN de l'appareil vers la caméra branchée. Il publie aussi à travers le topic `/last_image` le timestamp correspondant au moment où le dernier trigger a été envoyé. Ce pakage est écrit en Python.
— **image_saver :** Ce package fait parti du même workspace situé dans la Raspberry Pi. Il créé un pipeline Gstreamer afin de se connecter à la caméra branchée à la Raspberry Pi et sauvegarde une image dès lors qu'il en reçoit une. Pour dater les images, il utilise soit le timestamp publié sur `/trigger_time`, soit le timestamp du moment exact où il reçoit l'image. Le timestamp sert ensuite de nom pour l'image.

3.6.5.2 Discovery Server

Lorsque ROS est utilisé dans un laboratoire avec potentiellement d'autres projets utilisant ce framework, il est important que les messages publiés ne soient pas visibles par l'entièreté du laboratoire pour éviter d'interagir avec d'autres programmes sans le vouloir. Pour cela, il existe la variable *ROS_LOCALHOST_ONLY* permettant de garder les topics en local. Or dans notre cas, on doit fonctionner en réseau puisque les topics publiés depuis l'ordinateur principal doivent être perçus par les programmes sur les Raspberry Pi. Ainsi, on utilise ce qu'on appelle un *Discovery Server*. C'est un protocole qui permet la connexion entre différentes entités DDS. DDS est une norme pour l'échange de données via un réseau, typiquement une architecture ROS2 est une entité DDS. Donc le Discovery Server permet de connecter plusieurs environnements ROS2 ensemble, dans notre cas le workspace sur la machine principale avec les workspaces sur chaque Raspberry Pi. La machine principale sert de serveur, et tous les clients y sont connectés. Si les clients veulent communiquer entre eux via l'échange de topics ou de services par exemple, cela passe par le serveur et est redistribué vers les clients comme le montre cette figure 3.24.

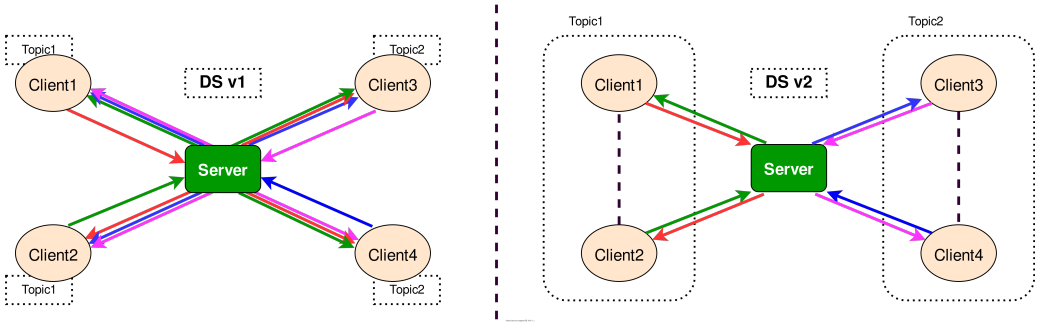


FIGURE 3.24 – Comparaison entre la première et deuxième version de discovery server implémenté dans ROS2. Figure tirée de la documentation de ROS2 [5]

Ainsi la figure 3.25 montre le "node graph" du système lorsque deux modules LIDAR+Caméra+Raspberry Pi sont branchés.

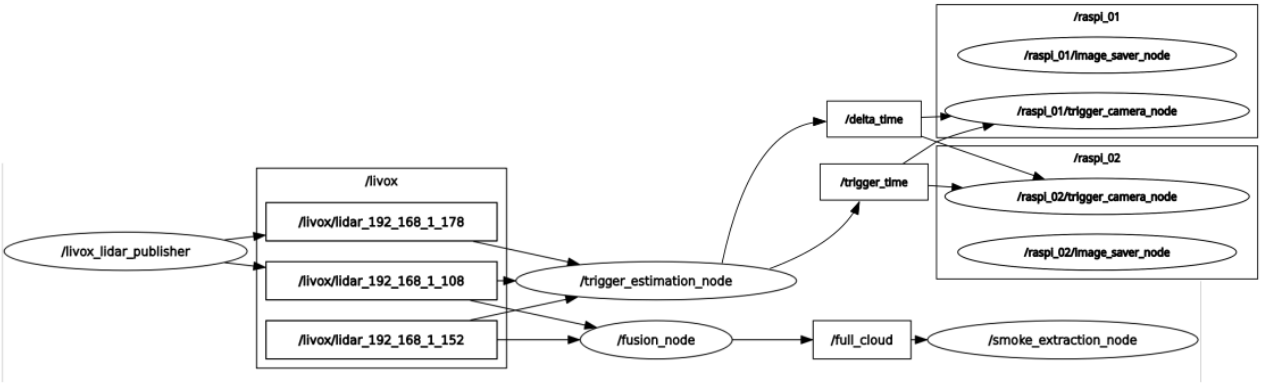


FIGURE 3.25 – Graphe montrant les liens entre les différents nodes ROS2. Les topics sont dans les boîtes rectangulaires, et les nodes dans les ovales. Les boîtes rectangulaires englobant les nodes sont des namespaces, pour éviter de mélanger deux topics avec le même nom comme le topic `/last_image` qui est publié sur deux Raspberry Pi différentes par exemple.

3.6.6 Acquisitions

Afin de se concentrer sur l'écriture de ce rapport, et de la documentation des apports dans ce projet, aucune acquisition de fumée n'a pu être réalisée à temps. Cependant, un protocole d'expérimentation décrivant chaque étape est disponible en annexe.

3.7 Conclusion sur le *setup* expérimental

L'objectif de cette partie était de présenter les différentes étapes et expérimentations qui ont permis de construire peu à peu un *setup* et un protocole expérimental qui permettraient de faire un *dataset* complet de données nuages de points et images d'un panache de fumée pour entraîner des modèles de reconstruction 3D. Afin de faciliter la mise en place des expérimentations, des modules de prise de vue visibles sur la figure 3.26 ont été réalisés pendant ce stage. De plus, grâce à ces modules, une fois les calibrations LIDAR-caméras réalisées, il n'y a, à priori, plus besoin de les refaire.

FIGURE 3.26 – Un module expérimental combinant un LIDAR, une caméra et une Raspberry Pi

Le travail présenté dans cette section pourra ainsi être réutilisé dans les travaux de L.Denis, afin de tester ses pipelines de reconstruction 3D sur des données réelles.

Chapitre 4

Traitement de données et Algorithmes

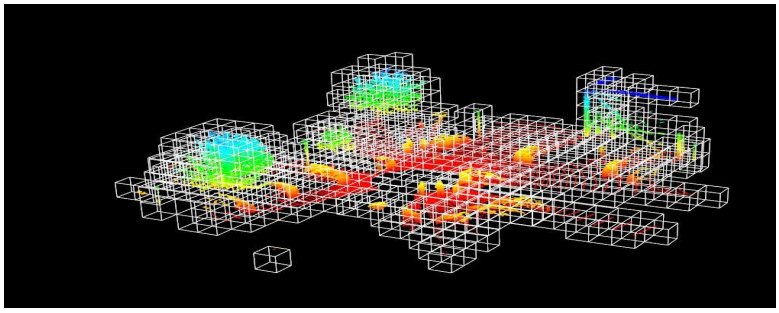
La section précédente avait pour but de présenter les travaux menés pendant ce stage sur de la prise de données à l'aide de LIDARs et caméras. Nous discuterons ici des programmes développés pour traiter les données recueillies pour divers objectifs.

4.1 Extraction de la fumée

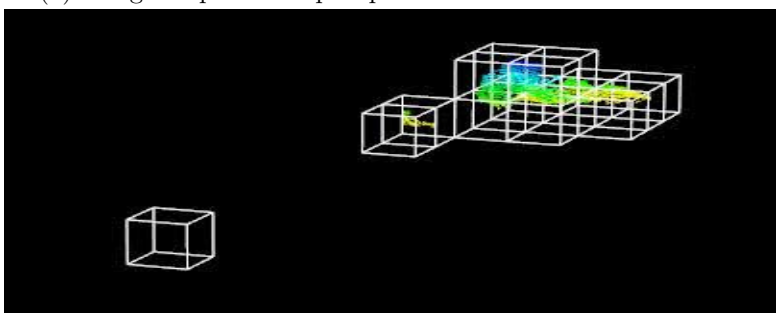
Le premier défi de traitement des données LIDAR est la segmentation dans un nuage de points. Plus précisément, ce qui nous intéresse est de déterminer quels sont les points qui correspondent au panache de fumée afin de les extraire du reste, comme illustré sur les figures 3.11 et 3.19. Il existe plusieurs algorithmes de segmentation, par exemple basés sur l'algorithme de RANSAC qui tente de retrouver des formes spécifiques dans le nuage [32]. D'autres méthodes basées sur du machine learning permettent d'améliorer les performances de segmentation en repérant aussi les objets dynamiques [33]. Dans notre cas, on cherche à extraire un objet dynamique avec une forme potentiellement très changeante, rendant peu efficaces les algorithmes basés sur RANSAC. De plus, étant donné le peu de données de fumée LIDAR (voire aucune) disponibles, utiliser des algorithmes basés sur du machine learning semble peu adapté.
Ainsi, le choix a été fait d'utiliser un principe plus simple, mais par conséquent moins adaptable à des situations réelles. En considérant que le panache de fumée est le seul objet dynamique de la scène, on peut comparer chaque *pointcloud* à une map de référence réalisée au préalable avec des données de l'environnement sans la fumée.

4.1.1 Division en octree

Pour que la comparaison entre les nuages soit rapide pour pouvoir la réaliser en temps réel, on choisit de représenter chaque nuage de points sous forme d'octree. Un octree est une représentation de données sous forme d'arbre où le nuage de points est partitionné en voxels. Ainsi, pour faire une comparaison entre deux nuages de points sous forme d'octree, il suffit de repérer quels sont les voxels qui sont apparus sur le deuxième nuage et qui ne sont pas présents sur le premier. Dans notre cas, on considère que ces nouveaux voxels correspondent alors forcément à la fumée, le seul objet dynamique de la scène. Pour effectuer ces comparaisons, on utilise la bibliothèque C++ PCL intégrant de nombreux algorithmes pour le traitement de données *pointcloud* [34] .
En revanche, comme on peut l'observer sur le panache extrait sur la figure 4.1, certains points peuvent apparaître et ne pas correspondre à la fumée. Pour éviter ces points aberrants, on applique des filtres pour retirer les points isolés.



(a) Nuage de point complet partitionné sous forme d'Octree



(b) Nuage de point après la comparaison avec la map de référence

FIGURE 4.1 – *pointcloud* sous forme d'octree

4.1.2 Données très bruitées

Dans le cas de scènes très contrôlées, comme lors d'expérimentations en intérieur, cette technique d'extraction fonctionne très bien car aucun élément extérieur ne vient perturber la scène. Cependant, cette méthode est moins adaptée aux expérimentations en extérieur et encore moins aux prises de données avec des drones. Comme on peut l'observer sur la figure 3.18, en extérieur de nombreux points correspondent à la végétation environnante, ce qui vient complexifier le traitement. Ces éléments peuvent aussi être dynamiques et donc ajouter des points qui ne correspondent pas au panache. Pour palier à ce problème, il est toujours possible d'ajouter des "bounding box" à la main ou d'augmenter le filtrage, mais ce qui peut aussi altérer la qualité des données (la fumée pouvant sortir de la bounding box, et le filtrage peut altérer aussi les données du panache surtout quand il est déjà très éparse).

4.1.3 Discussion de la méthode d'extraction

Pour ce stage, la méthode des octrees était suffisante, mais dans le futur, il pourrait être intéressant d'améliorer la technique d'extraction du panache. Une possibilité est d'utiliser aussi les données de la caméra et la calibration inter-capteurs présentée en section 3.3.2. Une autre possibilité pourrait être de concevoir un modèle d'apprentissage qui pourrait repérer les panaches de fumée basé sur les données créées à l'aide des modules d'acquisition de la première section.

4.2 Prévision de l'évolution du nuage de points

La mise en place d'un *setup* expérimental a permis de recueillir des données utilisables de nuages de points (et dans le futur de nuages de points combinés aux images) pour entraîner des modèles d'apprentissage. Une partie de ce stage a été dédiée à la conception de modèles d'apprentissage permettant de prédire l'évolution des nuages de points de panaches de fumée au cours d'une séquence temporelle à partir du début de cette même séquence. Et tout particulièrement, en s'intéressant à

l'utilisation des données qui impliquent de nombreux défis pratiques : taille des nuages différente, taille des séquences différente, le type d'encodage. Étant donné le peu de données disponible, les travaux menés ont plus pour intérêt de faire un état de l'art des modèles d'apprentissage existants et notamment comment encoder les nuages de points pour en tirer les meilleurs résultats.

4.2.1 Principaux modèles existants

Le *deep learning* appliqué aux nuages de points a été initié par des travaux pionniers comme PointNet [35] et PointNet++ [36], qui ont été les premiers à présenter un traitement direct des nuages de points bruts sans subdivision de l'espace avec une voxelisation ni à une projection 2D. PointNet introduit un mécanisme garantissant les propriétés des nuages de points, comme l'invariance à la permutation des points, en appliquant des MLP (Multi Layer Perceptron) indépendants suivis d'une fonction symétrique (dans ce cas, la fonction `max_pooling`) qui permet d'extraire des caractéristiques sans dépendre de l'ordre des points. Le modèle prend également en compte l'invariance aux transformations rigides (translations, rotations) grâce à un sous-réseau appelé T-Net, chargé d'estimer une matrice de transformation pour réaligner le nuage de points avant traitement. Toutefois, PointNet les caractéristiques extraites par l'encodage de PointNet représentent seulement le nuage de points de manière globale et ne permettent pas d'obtenir des relations plus précises entre les points dans différentes régions locales. Pour dépasser cette contrainte, PointNet++ introduit une structure hiérarchique inspirée des CNN sur images, capable de capturer des motifs locaux à différentes échelles qui sont ensuite agrégés de manière hiérarchique afin de former une représentation complète de la frame.
Dans leur prolongement, d'autres approches ont permis de dégager plus précisément les relations spatiales entre les points, notamment via les Graph Neural Networks comme PointGNN [37] ou DGCNN [38], qui représentent les nuages de points par des graphes de voisinage dynamiques (les nœuds et les arêtes évoluent).
Dans notre cas, on cherche à prédire l'évolution d'une séquence dynamique de nuages de points. Afin de garder en mémoire des informations extraites sur des frames précédentes, on peut s'appuyer sur des architectures récurrentes de type LSTM, capables de capturer les dépendances temporelles. Plus récemment, des modèles basés sur l'attention explorent de nouvelles manières de représenter conjointement les relations spatiales et temporelles.

4.2.2 Défis des données réelles

Comme évoqué en introduction, le fait d'utiliser des données réelles induit quelques défis pour l'apprentissage de modèles, dont notamment la différence de taille entre les données ou du bruit potentiel. On veut en entrée de notre modèle un nuage de point (ou une séquence de nuages de points), or tous les nuages de points et séquences ont des tailles différentes. Il n'est pas possible d'enregistrer des données de tailles différentes dans des tenseurs de la bibliothèque Python utilisée **Pytorch**. Il faut donc uniformiser les tailles de toutes les données utilisées. Une première solution est de mettre un seuil de nombre de points égal au nombre minimum parmi toutes les frames (de même pour le nombre de frame pour chaque séquence).
Cependant, les données obtenues contiennent très peu de points, et surtout les nuages de points correspondant au début du panache ne contiennent pour certains même pas 100 points. Ainsi, la solution choisie a été de faire du 0-padding afin d'uniformiser la taille des données. En revanche, il ne faut pas que ces données artificielles soient prises en compte dans la backpropagation lors de la phase d'apprentissage du modèle. Pour cela, on garde l'information de quels sont les points réels sous forme de masque, pour les supprimer lors du calcul de l'erreur. La figure 4.2 illustre le 0-padding et les masques réalisés à la fois sur le nombre de points par nuages de points, et le nombre de *pointcloud* par séquences.
On est donc aussi confronté au problème du trop petit nombre de points pour certaines frames,

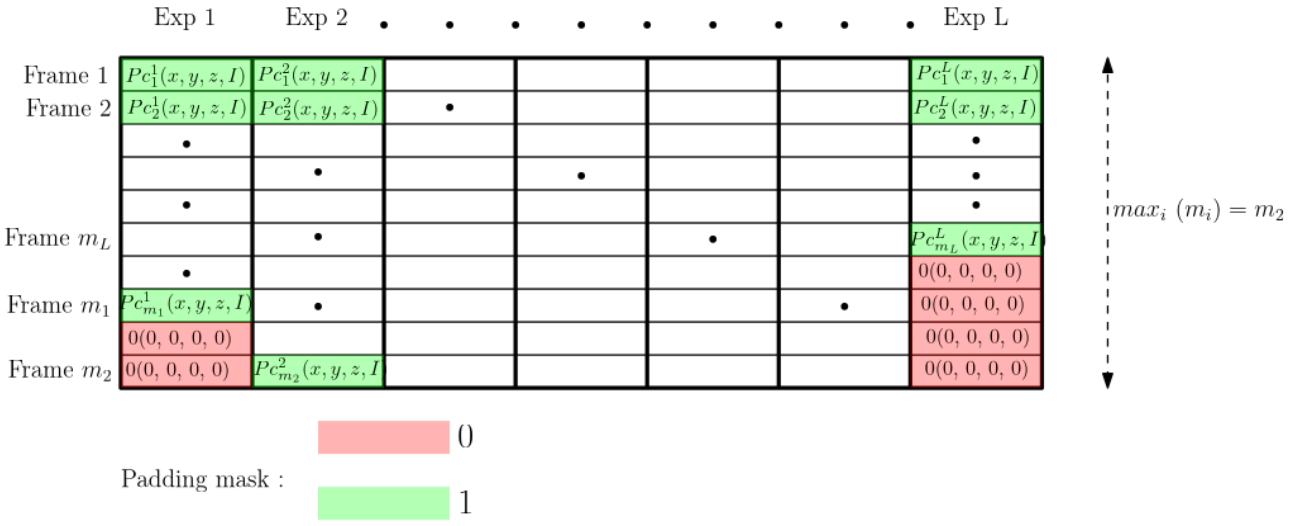


FIGURE 4.2 – Illustration du 0-padding des données nuages de points. En rouge les données artificielles, et en vert les données réelles. m_1, m_2 et m_L sont le nombre de *pointcloud* pour les expériences 1, 2 et L.

notamment au début et à la fin des séquences, c'est-à-dire lorsque le panache émerge et lorsqu'il se dissipe quand il n'y a plus de source. Il existe des algorithmes de densification de nuages de points trop peu denses qui pourraient permettre de compléter les données, et de leur donner une taille fixe.

4.2.3 Tests de modèles

Des premiers tests ont été effectués basés sur les modèles déjà existants d'une part afin de tester les traitements présentés dans la partie 4.2.2 (0-padding notamment) mais aussi pour comprendre comment extraire des features performantes des données.

4.2.3.1 Modèle simple MLP

Dans l'idée de faire un premier pas dans le domaine du machine learning pour les données nuages de points, un MLP adapté d'un code réalisé par L. Denis dans le cadre de sa thèse a été entraîné pour estimer une frame $t+1$ à partir de la frame t. Pour entraîner le modèle, on compare la frame estimée avec une frame réelle avec une fonction de coût de Chamfer :

$$Loss_{CD}(P_t, P_{t+1}) = \sum_{x \in P_t} \min_{y \in P_{t+1}} ||x-y||^2 + \sum_{y \in P_{t+1}} \min_{x \in P_t} ||y-x||^2 \tag{4.1}$$

Cette fonction de coût permet de comparer la distance entre 2 nuages de points qui n'ont pas forcément la même taille, en sommant les distances entre un point et son voisin le plus proche dans le deuxième nuage de points. On la retrouve dans la plupart des modèles d'apprentissage de nuage de points.
Le MLP prend en entrée un point à 4 dimensions (X, Y, Z, I) puis passe par une couche intermédiaire avec 128 nœuds pour donner en sortie une position absolue pour le point à la frame suivante. C'est une approche naïve car elle ne permet pas de prendre en compte les relations entre les points, et elle considère qu'un point dans la frame t correspond à un point dans la frame $t+1$, ce qui est forcément faux étant donné que les tailles de nuages de points varient d'une frame à l'autre. Cependant, cette première approche permet d'obtenir des premiers résultats qui permettent de comprendre comment ajuster certains paramètres. Par exemple, on se rend compte qu'en remplaçant le vecteur de sortie par

une position relative, on obtient des résultats légèrement meilleurs. Ainsi, avec

$$output = \begin{bmatrix} dx \\ dy \\ dz \\ I \end{bmatrix}$$

on obtient alors comme position prédite pour le point dans la frame $t+1$:

$$predPosition = prevPosition + output \tag{4.2}$$

4.2.3.2 Modèle PointNet + LSTM

Afin de complexifier légèrement le modèle et d'y ajouter des éléments vus dans des modèles existants pour avoir une sortie qui respecte les règles des nuages de points (voir section 4.2.1), on crée un modèle avec une extraction de features basée sur le modèle PointNet qui prend cette fois-ci en entrée une séquence de nuages de points. On n'utilise pas PointNet++ car ce dernier nécessite une grande quantité de points par frame pour extraire des caractéristiques locales, ce qui n'est pas le cas avec les données LIDAR seules. À la suite de cet encodage (à savoir l'extraction de features), on ajoute une couche de réseau LSTM avec des nœuds qui bouclent sur eux-mêmes. Ce bouclage permet de garder "en mémoire" des caractéristiques de frames précédentes. Enfin, on y ajoute une "decoding layer" qui nous permet d'obtenir en sortie la frame $t+1$. Pour la fonction de coût, comme pour le MLP simple, on utilise une loss de Chamfer. Des résultats visuels sont disponibles sur la figure 4.3.

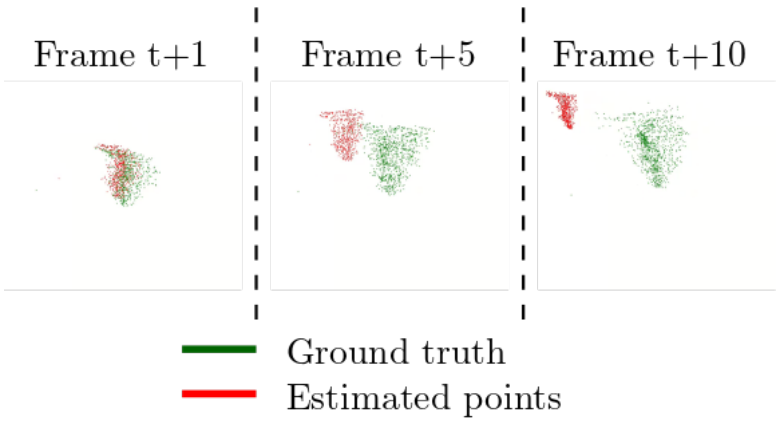


FIGURE 4.3 – Pointclouds estimés et réels à $t+1$, $t+5$ et $t+10$.

On observe le nuage estimé s'écarter peu à peu de la réalité terrain et rapetisser. L'erreur vient ici du fait que le modèle a été construit pour estimer seulement $t+1$, il est donc normal qu'il diverge au-dessus de cette frame. En revanche, on remarque tout de même un léger décalage dès la frame $t+1$, sûrement causé par de faux points en $(0,0,0)$ présents dans le nuage d'origine et qui n'ont pas été filtrés dans les données utilisées pendant l'entraînement du modèle (ces points ont été remarqués plus tard). La présence de ces points vient créer un décalage dans la position du nuage dès $t+1$. Des figures en annexe montrent plus précisément ces points en $(0,0,0)$.

4.2.3.3 Conclusions sur les modèles

Les résultats en tant que tels ne sont pas très satisfaisants, mais ils ont mis en évidence des problèmes majeurs liés aux données, notamment concernant la quantité de points ou la longueur des séquences. Par conséquent, le travail effectué peut aider à comprendre dès la collecte de données comment les rendre aussi adaptées que possible pour garantir une performance optimale des modèles d'apprentissage.

Chapitre 5

Limites et Perspectives

Les objectifs de ce stage consistaient à évaluer dans quelle mesure des données LIDAR de panaches de fumée pouvaient contribuer à améliorer la qualité d'une reconstruction 3D issue de données images. Les recherches et expérimentations réalisées pendant cette période ont mis en évidence plusieurs apports potentiels des données LIDAR. Leur précision permet d'obtenir une estimation précise de la surface du panache. De plus, une corrélation a été remarquée entre la concentration de la fumée et la réflectivité des points : les zones proches de la source, où la fumée est plus dense, présentent en effet une intensité plus élevée.
L'utilisation combinée de trois LIDAR a également montré qu'il était possible d'estimer, à une altitude donnée, une répartition gaussienne des points (plus ou moins fiable en fonction de la qualité des points). Ces informations pourraient être exploitées, par exemple, pour l'initialisation de Gaussiennes 3D dans le cadre de la reconstruction du panache avec le *3D Gaussian Splatting* développée par L. Denis.
Cependant, plusieurs limites sont apparues au cours des expérimentations. Les tests ont été effectués dans des conditions relativement contrôlées, alors que les capteurs sont destinés à être embarqués sur des drones. Dans ce contexte, les vibrations et les mouvements risquent d'affecter la qualité des données et de compliquer les calibrations, en particulier la calibration multi-LIDAR. De plus, en conditions extérieures, lorsque les capteurs sont éloignés du panache, la densité du nuage de points diminue fortement et les valeurs de réflectivité deviennent inutilisables. Une piste d'amélioration consisterait à augmenter le temps de scan des LIDAR, comme indiqué dans leur documentation [39], afin de densifier les nuages de points.
Pour ce qui est de la calibration multi-LIDAR dans un scénario de vol, une stratégie envisageable serait d'utiliser la position du drone, supposée connue (même avec une certaine incertitude), comme point de départ. Les positions relatives des capteurs pourraient ensuite être affinées a posteriori, par exemple grâce à l'algorithme testé lors de ce stage produit par l'entreprise Livox [30]. Néanmoins, cette calibration reste un véritable défi car ni l'environnement ni les drones ne sont statiques.
Un autre résultat important du stage a été la conception d'un *setup* expérimental réutilisable et simple à mettre en place, permettant d'acquérir des données exploitables pour des modèles d'apprentissage. Ce dispositif sera particulièrement utile dans la constitution d'un futur *dataset* associant images et nuages de points, dont l'élaboration complète n'a pas pu être menée à terme par manque de temps. Malgré cela, les données déjà collectées ont permis de réaliser quelques premiers essais avec des modèles simples, visant à prédire l'évolution dynamique des points. Bien que les modèles ne soient restés simples et les résultats peu concluants, ces expériences ont fourni des enseignements précieux sur le traitement de séquences de nuages de points et sur les propriétés spécifiques d'un nuages de points.
Enfin, au-delà des aspects techniques, ce stage aura été l'occasion de mieux comprendre le monde de la recherche et le sens du mot sérendipité, qui, bien qu'utile pour stimuler l'exploration, peut se révéler piégeuse si elle remplace la rigueur méthodologique.

Annexes

Fichiers de synchronisation des Raspberry Pi et Livox MID-360

Pour transmettre les messages de synchronisation PTP à toutes les Raspberry, il faut passer par une adresse multicast à laquelle chaque Raspberry pourra s'inscrire. Les paquets sont envoyés avec le protocole UDPv4 (peut changer selon la configuration du switch) à travers un seul port du PC et sont reçus par tous les appareils qui se connectent. Pour que la synchronisation fonctionne correctement, il faut désactiver sur chaque appareil client la potentielle synchronisation NTP s'il est connecté à un autre réseau (filaire ou wifi). Pour les LIDAR, l'ordinateur utilisé dispose d'un port Ethernet par LIDAR, tous reliés par un bridge. Dans le cas d'un timestamp hardware, il est primordial de synchroniser l'horloge interne du client (son horloge physique) avec les timestamps reçus par la connexion Ethernet. La librairie LinuxPTP donne ainsi 2 programmes à lancer : ptp4l (et phc2sys dans le cas d'une synchronisation hardware). Une fois la synchronisation enclenchée, on peut voir en sortie ce type de message après avoir lancé ptp4l :

ptp4l[22226.394] : rms 2738 max 5058 freq +1695 +/- 455 delay 50230 +/- 0

Le premier terme 'rms' signifie 'Root Mean Squared' et indique la moyenne de l'offset temporel par rapport au maître en nanosecondes. Avec cet exemple, on peut voir qu'on a un décalage temporel d'environ 2,7 µs, ce qui est très satisfaisant pour notre application, sachant qu'il y a d'autres sources d'incertitudes potentiellement plus grandes. Le deuxième terme est le décalage maximal recensé. Le terme 'freq' est la correction de fréquence de l'horloge locale (horloge physique). Par exemple ici on voit une correction positive par rapport au maître, ce qui signifie que l'horloge locale de la Raspberry Pi est plus lente que le maître. Si elle n'était pas corrigée, elle dériverait de 1.695µs chaque seconde (avec une erreur estimée à +- 455 ns). Enfin, le terme de delay mesure le temps de communication entre le serveur et le client (aller-retour).

Synchronisation Raspi

Côté maître

```
#
# Profile configuration for PTP synchronisation
# with a Raspberry Pi 5. This config file is for the master.
# See the file, default.cfg, for the complete list of available options.
#
[global]
# Options carried over from PTP.
gmCapable               1
priority1               248
priority2               248
logSyncInterval         -3
syncReceiptTimeout      3
```

```
neighborPropDelayThresh  800
min_neighbor_prop_delay  -20000000
network_transport        UDPv4
delay_mechanism          E2E
domainNumber             0
time_stamping            software
step_threshold           0.00002
serverOnly               1
logging_level            6
verbose                  1
transportSpecific        0
[enp0s31f6]
ptp_dst_ipv4             224.0.0.51
```

Côté Slave

```
#
# Automotive Profile configuration for PTP synchronisation
# with a Raspberry Pi 5. This config file is for the slave.
# See the file, default.cfg, for the complete list of available options.
#
[global]
gmCapable                1
priority1                248
priority2                248
logSyncInterval          -3
syncReceiptTimeout       3
neighborPropDelayThresh  800
min_neighbor_prop_delay  -20000000
clientOnly               1
domainNumber             0
network_transport        UDPv4
delay_mechanism          E2E
time_stamping            software
transportSpecific        0
# Makes convergence faster
step_threshold           0.00002
# logging_level          6
# verbose                1
[eth0]
ptp_dst_ipv4             224.0.0.51
```

Synchronisation LIDAR

```
#
# Automotive Profile example configuration for master containing those
# attributes which differ from the defaults.  See the file, default.cfg, for
# the complete list of available options.
```

```
#
[global]
# Options carried over from gPTP.
gmCapable                 1
priority1                 248
priority2                 248
logSyncInterval           -3
syncReceiptTimeout        3
neighborPropDelayThresh   800
min_neighbor_prop_delay   -20000000
assume_two_step           1
path_trace_enabled        1
follow_up_info            1
transportSpecific         0
ptp_dst_mac               01:80:C2:00:00:0E
network_transport         L2
delay_mechanism           P2P
ptp_minor_version         0
time_stamping             software
#
# Automotive Profile specific options
#
BMCA                      noop
serverOnly                1
inhibit_announce          1
asCapable                 true
inhibit_delay_req         1
```

Protocole acquisition multi-capteurs

Objectif :

Prendre des mesures d'un panache de fumée dans un environnement contrôlé ou non, avec 5 caméras et 3 LiDARs synchronisés. Ce protocole a pour but de rendre le plus facile possible l'expérimentation, et de la rendre le plus rapidement possible reproductible dans des conditions similaires afin de construire un large dataset.

Première étape : Démarrage de la synchronisation et Discovery Server

1→Pour commencer, il faut brancher tous les capteurs et se connecter en SSH à partir de l'ordinateur principal à toutes les Raspberry. Il y en a normalement 5, une par caméra.
2→ Une fois fait, la première étape est de lancer la synchronisation PTP avec la librairie PTP4L et les fichiers de synchronisation disponibles. Cela prend un petit temps avant que la synchronisation ne fasse effet.
3→Lancer aussi le PTP pour les LiDARs.
Si le PTP est en mode Hardware, il faut aussi synchroniser toutes les horloges physiques à une horloge de référence en utilisant la bibliothèque phc2sys (/ ! Ne pas oublier les horloges internes du switch s'il y en a un).
4→Lancer le discovery server et sourcer les variables d'environnement sur tous les terminaux ouverts qui interagissent avec ROS (sur la machine et sur les Raspberry Pi).

Deuxième étape : Calibration des capteurs

Pour commencer, placer tous les capteurs à l'endroit voulu, ainsi que la source de fumée, et on peut commencer les calibrations. Cette étape est certainement la plus longue et la plus éprouvante mais aussi la plus importante pour obtenir des données de qualité. Alors il faut rester concentré jusqu'au bout, et ne pas se presser pour éviter d'oublier une calibration.
1→ Calibration des caméras 2 à 2 pour obtenir la transformation entre toutes les caméras.
2→ Calibration des LiDAR avec leur caméra correspondante. La synchronisation doit être lancée.
3→ Calibration des LiDAR entre eux.
Si les calibrations sont bien faites, si on boucle les transformations on retrouve la matrice identité. Une fois les calibrations faites, il ne faut plus toucher au set-up. Un petit mouvement pourrait détériorer les données donc si un des éléments bouge, il faut refaire la calibration.
4→ Sauvegarder toutes les calibrations entre les différents composants sans ambigüité.

Troisième étape : Démarrage du recording

Maintenant que tout est fixé et tout calibré, on peut commencer.
1→ Lancer le driver Livox
2→ Lancer le package de fusion des nuages de point (MACHINE)(ATTENTION à ce que les timestamps restent les mêmes que reçus par le driver Livox)
3→ Lancer les Rosbag pour les LiDARs (MACHINE)
4→ Lancer le package de sauvegarde des images (RASPI)
5→ Lancer le package de Trigger (RASPI)
6→ Lancer le package de Trigger Estimation (MACHINE)
7→S'assurer que tous les records fonctionnent, lancer la fumée
Recommencer ces étapes autant de fois que nécessaire.

Quatrième étape : Post-processing

Pour tous les rosbags obtenus :
1→ Lancer le package d'extraction du panache de fumée
2→ Lancer le Rosbag
3→ Sauvegarder soit sous forme de Rosbags soit sous forme de .PCL les nuages extraits.

Quelques résultats de modèles de machine learning

Le modèle décrit en section 4.2.3.1 avait donc pour but de visualiser des premiers résultats. Une visualisation de ces résultats est disponible sur la figure 5.1, on voit que le nuage est décalé et que les points estimés semblent être attirés par un point en $(0, 0, 0)$ qui ne semble pas appartenir au panache. Il s'agit en réalité de faux points dans le nuages de points (erreurs de lecture ou autre) qui n'avaient pas été filtrés comme mentionné dans la section 4.2.3.2. C'est ce qui semble causer le décalage du panache.

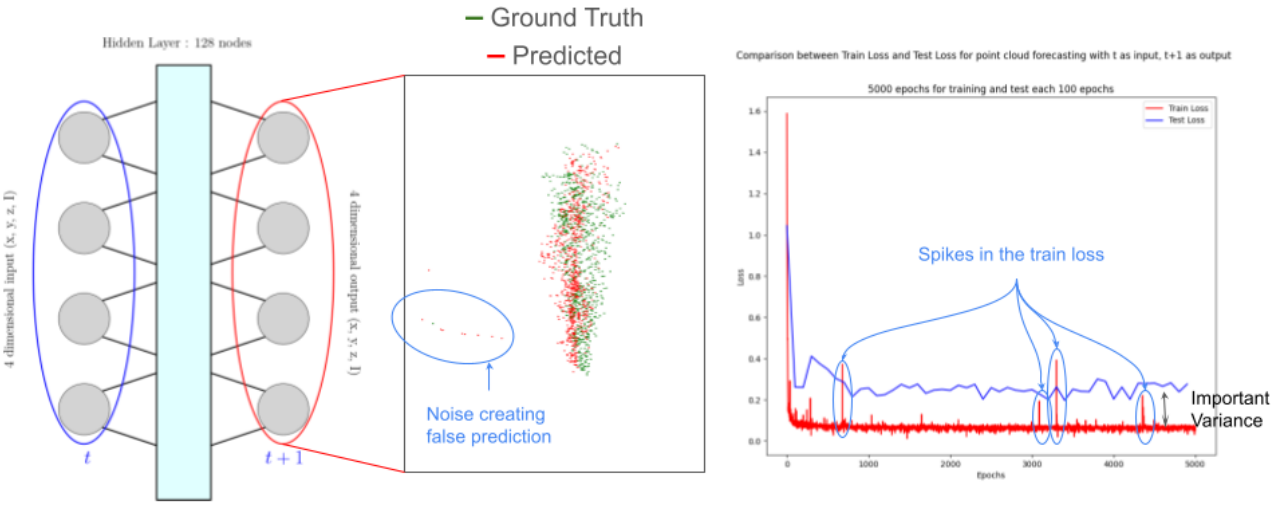


FIGURE 5.1 – Structure du réseau de neurone utilisé pour estimer la position de chaque point dans la frame $t + 1$ à partir de la position dans la frame t. Au centre une visualisation d'un nuage estimé (en rouge) et la comparaison avec la vérité terrain (en vert). A droite la fonction de coût en fonction des *epochs*.

On remarque aussi sur la courbe à droite de la figure 5.1 de nombreux piques. Ceux-ci peuvent être dus à plusieurs choses : une descente de gradient bruitée à cause de l'*optimiseur* choisi ou de la taille des *batch*, des données de mauvaise qualité ou un *learning rate* trop haut. Mise à part ce point en $(0, 0, 0)$, les données utilisées ne sont pas bruitées et le panache est bien représenté sans trop de divergence entre 2 frames. L'*optimiseur* utilisé est **Adam**, très populaire sur les algorithmes de machine learning, avec différents *learning rates* testés sans changement au niveau de la courbe de la fonction de coût. En revanche, comme on peut l'observer sur la figure 5.2, la variation dans la taille des *batch* a de fortes conséquences sur le bruitage de la courbe. Lorsqu'on augmente leur taille, la courbe est moins bruitée mais l'apprentissage plus long.

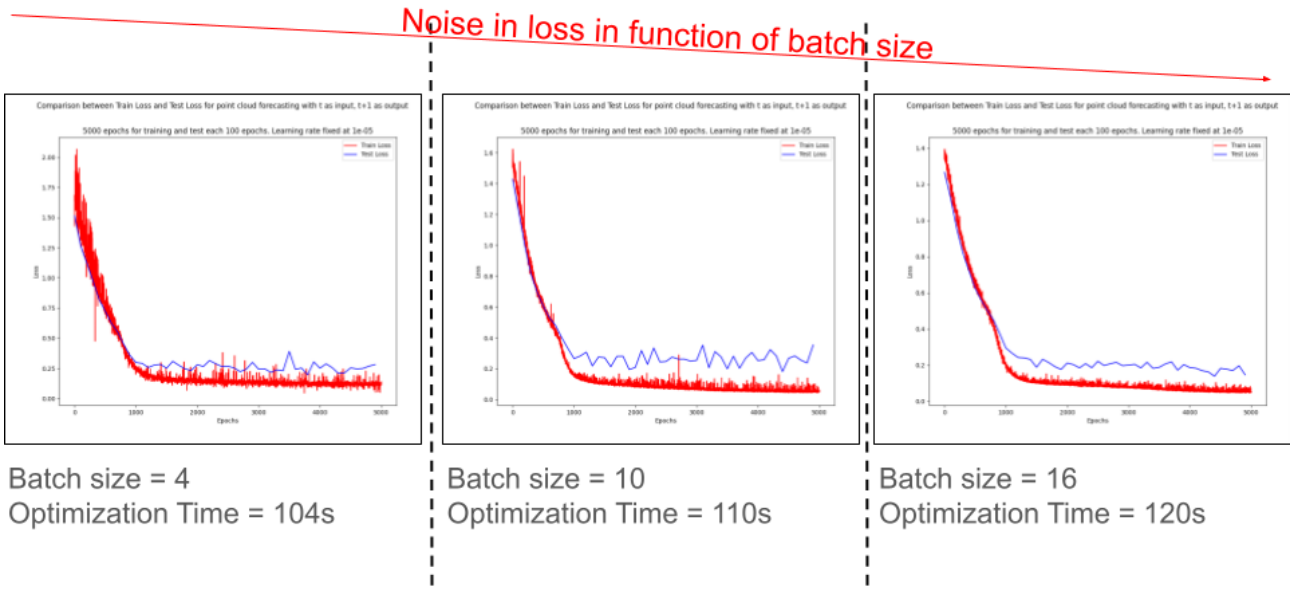


FIGURE 5.2 – Évolution des fonctions de coût en fonction de l'augmentation de la taille des *batch* pendant l'apprentissage

Pseudo-code extraction du panache

Voici le pseudo code correspondant à l'extraction du panache de fumée d'un pointcloud comme présenté dans la section 4.1. Il utilise un système de double *buffer* permettant de garder en mémoire deux nuages de points et ainsi rendant plus rapide la comparaison entre deux nuages de points.

Algorithm 4: Extraction d'un panache de fumée à partir de nuages de points

Input: Flux de nuages de points P_t issus des LiDARs
Output: Nuage de points extrait correspondant au panache de fumée S_t

foreach *nuages de points reçus* P_t **do**
  $P_t \leftarrow$ `FilterByAngle`(P_t);
  ; `// Conserver uniquement l'angle d'intérêt`
$F_t \leftarrow$ `FusionPointClouds`(*buffers de nuages de points*);
; `// Fusionner tous les nuages dans le cas de plusieurséchos reçus`
if *Map de référence non constituée* **then**
  $P_{ref} \leftarrow$ `BuildReferenceCloud`(F_t);
  ; `// Construire la carte de référence (statique)`
  Stocker l'octree de référence à partir de P_{ref} dans la première cellule du double buffer;
else
  Mettre à jour l'octree actuel à partir de F_t, et le stocker dans la deuxième cellule du double buffer;
  $N_t \leftarrow$ `DetectNewPoints`(F_t, P_{ref});
  ; `// Détection des points nouveaux par rapport à la référence`
  $S_t \leftarrow$ `FilterOutliers`(N_t);
  ; `// Filtrer les points aberrants`
  Publier S_t comme panache de fumée détecté;

Bibliographie

[1] Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. Pgsr : Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 2024. 9, 14

[2] Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz. Monte carlo methods for volumetric light transport simulation. In *Computer graphics forum*, volume 37, pages 551–576. Wiley Online Library, 2018. 9, 18

[3] Tyson Phillips, Nicky Guenther, and Peter Mcaree. When the dust settles : The four behaviors of lidar in the presence of fine airborne particulates. *Journal of Field Robotics*, 34, 02 2017. 9, 18, 19

[4] Karl Montalban. *Advancing LiDAR perception in degraded visual environments : a probabilistic approach for degradation analysis and in inference of visibility.* Theses, ISAE Université de Toulouse, September 2023. 9, 23

[5] Ros 2 humble documentation. https://docs.ros.org/en/humble/index.html. 10, 40

[6] Brauer M. Johnston F. H. Jerrett M. Balmes J. R. Elliott Reid, C. E. Critical review of health impacts of wildfire smoke exposure. *Environmental health perspectives*, 2016. 11

[7] François Pimont Julien Ruffault Eric Rigolot, Jean-Luc Dupuy. Les incendies de forêt catastrophiques. *Annales des mines - Série Responsabilité et environnement*, 2020. 11

[8] Matthias Muller-Fischer Robert Bridson. Fluid simultaion, 2007. 11, 14

[9] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. 12, 14

[10] Ruyi Zha, Tao Jun Lin, Yuanhao Cai, Jiwen Cao, Yanhao Zhang, and Hongdong Li. R^2-gaussian : Rectifying radiative gaussian splatting for tomographic reconstruction, 2024. 12, 13

[11] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. *The lumigraph.* Association for Computing Machinery, 2023. 13

[12] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism : exploring photo collections in 3d. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, page 835–846, New York, NY, USA, 2006. Association for Computing Machinery. 13

[13] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M Seitz. Multi-view stereo for community photo collections. In *2007 IEEE 11th international conference on computer vision*, pages 1–8. IEEE, 2007. 13

[14] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf : Point-based neural radiance fields, 2023. 13

[15] Mengyu Chu, Lingjie Liu, Quan Zheng, Aleksandra Franz, Hans-Peter Seidel, Christian Theobalt, and Rhaleb Zayer. Physics informed neural fields for smoke reconstruction with sparse data. *ACM Transactions on Graphics*, 41(4) :1–14, July 2022. 13

[16] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf : Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 13

[17] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3) :223–238, 2002. 14

[18] Jerry Westerweel Ronald J Adrian. *Particle image velocimetry*. Cambridge university press, 2011. 15

[19] B. Wieneke et B. W. van Oudheusden G. E. Elsinga, F. Scarano. Tomographic particle image velocimetry. *Experiments in Fluids*, 2006. 15

[20] Aleksandra Franz, Barbara Solenthaler, and Nils Thuerey. Global transport for fluid reconstruction with learned self-supervision, 2021. 15

[21] Yiming Wang, Siyu Tang, and Mengyu Chu. Physics-informed learning of characteristic trajectories for smoke reconstruction. In *ACM SIGGRAPH 2024 Conference Papers*, SIGGRAPH '24, New York, NY, USA, 2024. Association for Computing Machinery. 15

[22] Marie-Lena Eckert, Kiwon Um, and Nils Thuerey. Scalarflow : a large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *ACM Trans. Graph.*, 38(6), November 2019. 16, 33, 34

[23] Konstantinos Rematas, Andrew Liu, Pratul P Srinivasan, Jonathan T Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. Urban radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12932–12942, 2022. 16

[24] Changjian Jiang, Ruilan Gao, Kele Shao, Yue Wang, Rong Xiong, and Yu Zhang. Li-gs : Gaussian splatting with lidar incorporated for accurate large-scale reconstruction, 2024. 16

[25] Georg Hess, Carl Lindström, Maryam Fatemi, Christoffer Petersson, and Lennart Svensson. Splatad : Real-time lidar and camera rendering with 3d gaussian splatting for autonomous driving. *arXiv preprint arXiv :2411.16816*, 2024. 16

[26] Neil Lareau Mount Washinton Observatory. Science in the mountains : The science of fire weather. https://www.youtube.com/watch?v=osIAgms95RU. 17

[27] CESBIO. Dart. 19

[28] François Félix Ingrand. https://redmine.laas.fr/projects/minnie. 20

[29] Christian Seigneur. Modélisation de la pollution atmosphérique. https://cerea.enpc.fr/fich/support_cours/SGE_M2_modelisation/SGE-Modelisation-Dispersion.pdf. 24

[30] Zheng Gong, Chenglu Wen, Cheng Wang, and Jonathan Li. A target-free automatic self-calibration approach for multibeam laser scanners. *IEEE Transactions on Instrumentation and Measurement*, 67(1) :238–240, 2018. 28, 49

[31] The linux ptp project. https://linuxptp.nwtime.org/about/. 37

[32] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, pages 214–226. Wiley Online Library, 2007. 43

[33] Xieyuanli Chen, Shijie Li, Benedikt Mersch, Louis Wiesmann, Jürgen Gall, Jens Behley, and Stachniss Cyrill. Moving object segmentation in 3d lidar data : A learning-based approach exploiting sequential data. *IEEE Robotics and Automation Letters*, 6(4) :6529–6536, 2021. 43

[34] Radu Bogdan Rusu and Steve Cousins. 3D is here : Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011. IEEE. 43

[35] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet : Deep learning on point sets for 3d classification and segmentation, 2017. 45

[36] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++ : Deep hierarchical feature learning on point sets in a metric space, 2017. 45

[37] Weijing Shi, Ragunathan, and Rajkumar. Point-gnn : Graph neural network for 3d object detection in a point cloud, 2020. 45

[38] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5) :1–12, 2019. 45

[39] Livox products documentation. https://livox-wiki-en.readthedocs.io/en/latest/index.html. 49