Reconstruction de la géométrie des tirs d'un Système Lidar Mobile

Isabella Wokam

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

ECOLE SUPERIEURE D'INGENIEURS GEOMETRES ET TOPOGRAPHES

MEMOIRE

présenté en vue d'obtenir

le DIPLOME D'INGENIEUR CNAM

SPECIALITE : Géomètre et Topographe

par

Isabella WOKAM

Reconstruction de la géométrie des tirs d'un Système Lidar Mobile

Soutenu le 30 septembre 2024

JURY

Monsieur Stéphane DURAND	Président du jury
Monsieur Philippe VERLEY	Maître de stage
Monsieur Jérôme VERDUN	Enseignant référent

Remerciements

La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes.

Je remercie particulièrement M. Jérôme Verdun, pour sa disponibilité, son œil aguerrit, ses idées et ses interventions éclairées et originales sur mes travaux.

Un grand merci à AMAP et en particulier à M. Philippe Verley mon maître de stage, pour sa disponibilité, sa bonne humeur, ses idées et conseils sur le fond et la forme de ce mémoire.

Je tiens à remercier l'ingénieur métrologue KOUAME Maya Bi Axel Gerald Goneti pour son soutien émotionnel et sa patience.

Un grand merci à ma famille, mon oncle, mes aami.es pour leur soutien.

Enfin, je remercie tout particulièrement ma mère, mon père et mon grand-père.

Liste des abréviations

ALS : **A**irbone **L**iDAR **S**ystem

DLS : Drone **L**iDAR **S**ystem

feo : **f**réquence d'**e**chantillonnage d'**o**rigine du scanner

fet : **f**réquence d'**e**chantillonnage des **t**rous détectés

fef : **f**réquence d'**e**chantillonnage de la **f**usion des nuages de points

GPS : **G**lobal **P**ositioning **S**ystem

INTERP : **Interp**olation

LaDAR : **La**ser **D**etection **A**nd **R**anging

LAI : **L**eaf **A**rea **I**ndex

LiDAR : **Li**ght **D**etection and **R**anging

MLS : **M**obile **L**iDAR **S**ystem

MNT : Modèle **N**umérique de **T**errain

MNS : Modèle **N**umérique de **S**urface

ndp : **n**uage **d**e **p**oints

ndm : **n**uage **d**e points avec **m**ille valeurs en **m**oins

ndv : **n**uage **d**e points de **v**érification

vdm : **v**ecteurs **d**irecteurs **m**anquants

vdv : **v**ecteurs **d**irecteurs **v**érification

SLAM : **S**imultaneous **L**ocalization **A**nd **M**apping

TLS : Terrestrial **L**iDAR **S**ystem

RaDAR : **Ra**dio **D**etection **A**nd **R**anging

Glossaire

AMAP : Laboratoire de recherche

AMAVox : Outil de recherche du laboratoire de recherche

dt : Intervalle de temps entre valeurs de temps consécutives

dt_min : Le plus petit intervalle de temps entre valeurs de temps consécutives

Loi normale : La loi normale est une fonction de densité de probabilité d'une variable aléatoire continue qui suit une distribution symétrique en forme de cloche autour de sa moyenne. Elle est caractérisée par la moyenne qui détermine la position centrale de ka courbe, et l'écart-type qui contrôle la dispersion autour de cette moyenne.

Loi uniforme : La loi uniforme est une loi de probabilité discrète pour laquelle la probabilité de réalisation est identique (équiprobabilité) pour chaque tir.

Table des matières

Introduction

Les forêts équatoriales sont d'importants lieux d'échanges, d'activités mais aussi de biodiversité qu'il est crucial de préserver. Dans un contexte de changement climatique, il est important de bien étudier et de caractériser les différents étages du couvert forestier qui concentre l'essentiel des échanges entre la biosphère et l'atmosphère. Dans ce composant de la surface terrestre les phénomènes d'échange biosphère-atmosphère les plus importants tels que l'absorption de la radiation solaire, l'interception de la pluie et l'activité photosynthétique ont lieu. La caractérisation de ces couverts végétaux est donc un élément clé, dans le cadre de la compréhension des changements climatiques. Dans ce but, la détermination de l'indice de surface foliaire (« LAI » pour Leaf Area Index) qui donne une mesure de la densité de la couverture végétale et le potentiel photosynthétique, est devenue aujourd'hui un des éléments critiques dans la caractérisation des phénomènes d'échanges entre la biosphère et l'atmosphère.

La technologie LiDAR (Light Detection and Ranging), basée sur le principe de la télédétection active par émission-réception d'impulsions laser, est un outil puissant de plus en plus utilisée pour caractériser la structure tridimensionnelle et la dynamique de la végétation. À partir de ce type de donnée, il est notamment possible d'extraire l'indice foliaire des couverts forestiers qui caractérise particulièrement bien lesdits échanges entre la végétation et l'atmosphère. Son utilisation pour estimer la quantité de surface foliaire à l'échelle de l'arbre, de la parcelle, ou de la forêt nécessite le développement et la mise en œuvre d'outils de traitement de signal appropriés. Le laboratoire AMAP a développé le logiciel AMAPVox dans cette optique. L'outil permet de traiter des données de LiDAR terrestre et de LiDAR aéroporté.

Le LiDAR aéroporté permet de caractériser à l'échelle régionale le couvert des canopées tropicales mais la densité du couvert empêche une bonne pénétration du laser et la qualité des estimations de LAI sous-couvert est généralement insatisfaisante. Le LiDAR terrestre permet d'apporter un point de vue complémentaire, sous canopée, mais son utilisation est limitée à cause des difficultés d'accès et d'acquisition en sous-bois de forêt tropicale humide (matériel lourd, encombrant et lent à déplacer avec la mise en place du trépied). Les gammes de LiDAR MLS (mobile lidar system) permettent d'envisager des levers de plus grande emprise sous-couvert forestier, et l'UMR AMAP a entrepris de tester ces systèmes et d'adapter ses outils à cette "nouvelle" technologie.

Récemment, lors d'une campagne en Guyane, une équipe de chercheurs AMAP souhaitant estimer le LAI d'une parcelle de la forêt amazonienne, s'est servie des acquisitions faites à l'aide d'un MLS (le système HoverMap ST-X) dont les données incluent trois paramètres essentiels à sa détermination : l'intensité lumineuse, la température et l'humidité. Contrairement au LiDAR aéroporté où tous les tirs d'intérêts génèrent des échos (i.e des impulsions laser retour), les acquisitions faites depuis le sol, possèdent des tirs non interceptés et donc sans échos quand le rayon laser ne rencontre aucun obstacle. Cette présence de tirs vides ou cette absence de tirs nécessite la mise en place d'une reconstruction de la géométrie des tirs manquants car ces tirs sans échos contiennent aussi des informations précieuses telles qu'une absence d'obstacle (absence de végétation dans notre cas) et indispensable à l'estimation du LAI. Le système HoverMap ST-X qui inclut une centrale inertielle et le capteur LiDAR dispose pour chaque acquisition deux fichiers. Le fichier XYZ qui est celui de la trajectoire et qui contient toutes les données de la centrale inertielle (temps, position, vitesse et angles de tangage, roulis et lacet) et le fichier LAZ qui est celui du nuage de points et qui contient les échos (positions dans l'espace, numéro de faisceau et temps).

L'objectif principal du stage consiste donc à implémenter un algorithme de reconstruction de la géométrie des tirs d'un système LiDAR MLS. L'algorithme devra détecter précisément les tirs manquants, inférer leur direction et générer des pseudo-échos à l'infini dans le nuage de points corrigé.

L'objectif secondaire de ce stage consiste à implémenter un algorithme de détection des faux tirs vides MLS. Un faux tir vide est généré par le capteur LiDAR lorsque le rayon de rencontre entre l'impact du laser et la matière est très faible. Par exemple, le système HoverMap ST-X ne détecte pas les obstacles à moins de 50 cm et la détection est incertaine entre 50 cm et 1 m. Dans le cadre d'une acquisition sac-à-dos, cette détection incertaine concerne donc l'opérateur et toute la végétation proche rencontrée sur le chemin. Cet algorithme de détection des faux tirs vides devra permettre d'écarter systématiquement tous les faux tirs vides de nos nuages de points.

Pour présenter les travaux réalisés pour atteindre ces deux objectifs, nous avons décidé d'organiser ce mémoire en 3 grandes parties. Tout d'abord l'état de l'art présentera le LiDAR en général (origine, histoire, applications) et plus particulièrement le LiDAR en foresterie. Cette partie servira de base aux deux autres. Ensuite la deuxième partie de ce mémoire sera consacrée à l'implémentation des algorithmes en lien avec nos deux objectifs. Nous aborderons

la problématique en la testant sur 2 cas théoriques simplifiés : l'un utilisant un système Lidar statique, l'autre, un système Lidar dynamique qui suit une trajectoire rectiligne avant de l'appliquer sur un échantillon de données d'un cas réel. Enfin la troisième et dernière partie nous permettra d'appliquer et de tester nos algorithmes sur un cas réel. Dans cette partie, nous confronterons donc notre protocole aux réalités du terrain à travers une application à un cas pratique en foresterie afin de le valider.

I Etat de l'art

I.1 Les scanners LiDAR en foresterie

I.1.1 La technologie LiDAR

Le scanner laser ou LiDAR, acronyme de l'expression anglaise « Light Detection And Ranging » (Détection de la lumière et mesure à distance) est une technique optique de Mesure Electronique de Distance (MED) qui découle de celui du LASER (Light Amplication by Stimulated Emission of Radiation) signifiant "Lumière Amplifiée par Simulation d'Émission de Rayonnement". Le LiDAR, parfois aussi appelé LaDAR (Laser Detection and Ranging), utilise des ondes de lumière dans le domaine optique, de l'ultraviolet au proche infrarouge, se rapproche d'une autre technique de mesure : le RaDAR (Radio Detection And Ranging) à la seule différence que celui-ci utilise une autre gamme de longueur d'onde : les ondes radiométriques. Le Lidar est une technique de télédétection dite active. Elle utilise sa propre source d'énergie lumineuse pour éclairer les objets étudiés. Une source laser est utilisée pour la collecte à distance de données sur la scène observée. Elle se distingue des méthodes dites passives basées sur le rayonnement naturel (lumière solaire) telles que la spectroscopie ou la photographie aérienne [POPULUS J., 2022] ou le rayonnement thermique infrarouge émis par des objets, mesurable par une caméra multispectrale. Cette méthode d'analyse optique permet donc des mesures en l'absence de lumière solaire [Flamant P.H. 2019].

Le développement du LiDAR remonte aux années 1960, parallèlement à l'essor de la technologie laser. La technologie LiDAR a permis de réaliser de nombreuses avancées dans le domaine scientifique. La première utilisation importante du LiDAR fut le calcul de la distance Terre-Lune pour le projet *« Luna See »* en 1962. Le LiDAR a aussi permis de cartographier la Lune en 1971 lors de la mission Apollo 15 comme le montre la figure 1.

Figure 1 : Faisceaux lidars montés sur des télescopes afin de mesurer la distance Terre-Lune Source : Revue NASA

Par la suite, son utilisation dans le domaine militaire et spatial s'est développée avec l'apparition du système de radiopositionnement par satellite américain GPS. Au fil des décennies, cette technologie s'est diversifiée et perfectionnée, on remarque une grande diversité des LiDARs trouvant ses applications en topographie, géosciences, patrimoine et même archéologie. Il est utilisé pour la détection de sites enfouis sous la végétation ou l'eau. Son utilisation a permis à une équipe de chercheurs du Guatemala de découvrir en février 2018 une ancienne Cité Maya enfouie de plus de 2 000 km^2. En sciences de l'environnement avec les études forestières notamment, la technologie lidar a plusieurs utilités. Elle permet de calculer la quantité de dioxyde de carbone, prévenir les risques incendie à très grandes échelles, mesurer la hauteur de la canopée ou la couronne des arbres, quantifier la biomasse et la quantité de lumière interceptée, déterminer l'indice de surface foliaire pour établir des bilans de couverture végétale etc.

Le principe de la mesure LiDAR requiert généralement l'utilisation d'un laser impulsionnel. Le laser impulsionnel produit des flashs de lumière régulièrement espacés dans le temps et très brefs, que l'on appelle impulsions. Le télémètre laser qui a pour fonction la mesure de distance, fonctionne selon le principe d'un radar, « l'appareil laser génère, à intervalles réguliers, une série d'impulsions lumineuses intenses. Ces impulsions sont émises en direction de l'objet d'étude, lequel peut être une surface solide, liquide ou gazeuse (sol nu, sol recouvert de végétation, etc.) » [POPULUS J., 2002] puis réfléchie par l'objet. Le signal retour est ensuite détecté puis traité de manière à avoir une distance entre le télémètre et la cible [DU CHAPELET M.et al., 2016]. « L'écart en temps entre l'instant d'émission et l'instant de réception du signal est représentatif de la distance parcourue par ce dernier, donc de la distance séparant la source de rayonnement [de l'objet] » [POPULUS J., 2002]. Le télémètre laser permet donc d'obtenir la distance en un point précis de l'espace. Pour pouvoir obtenir une multitude de points, le télémètre à balayage utilise un miroir rotatif permettant l'envoi d'une sorte de nappe laser et ainsi la mesure de plusieurs points. Aujourd'hui des calculs de distances sont toujours utilisés par la technologie LiDAR et ne cessent de se développer et de s'améliorer […] en robotique, automobile, etc. [DU CHAPELET M. et al., 2016].

I.1.2 Types de LiDAR : LiDAR aéroporté et LiDAR terrestre

« [De nos jours, la méthode LiDAR constitue] une puissante technique pour étudier de nombreux aspects de l'environnement » [POPULUS J., 2002]. Les recherches se multiplient et les observations s'effectuent aussi bien depuis le sol, d'une plateforme aéroportée que de

l'espace. On effectue donc une distinction du LiDAR en fonction de sa prise de vue : les LiDAR terrestres et les LiDAR aéroportés.

Le LiDAR aéroporté, communément désigné par ALS acronyme de « Airbone LiDAR System » qui signifie système LiDAR aérien, utilise un scanner laser embarqué à bord d'aéronefs tels que des avions, des hélicoptères, des drones ou encore des satellites. Au cours de sa progression le laser balaie latéralement la surface terrestre (figure 2) en envoyant des impulsions à très haute fréquence (plusieurs centaines de milliers par seconde). Pour chaque impulsion le capteur enregistre le temps écoulé entre le moment où il émet une impulsion et celui où il reçoit le signal réfléchi par un objet se trouvant dans la trajectoire de l'impulsion. On en déduit la distance entre l'émetteur (scanner) et le récepteur (l'objet), connaissant la trajectoire de l'avion et son orientation. Cette technique permet de représenter les objets se trouvant à la surface de la Terre par des points d'impact du laser avec la matière sous la forme d'un nuage de points en trois dimensions.

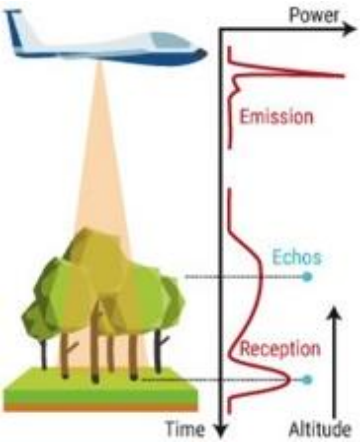

Figure 2 : Exemple de lidar aéroporté, LiDAR embarqué à bord d'un avion Source : yellowscan.com

En règle générale, l'acquisition de données LiDAR depuis un véhicule en mouvement se fait selon l'équation de géoréférencement (**e1**) suivante qui utilise la trajectoire et l'orientation du porteur.

$$\overrightarrow{r_p^g(t)} = \overrightarrow{r_{GPS}^g(t)} + R_b^g(t).R_s^b\left[\overrightarrow{r_p^s(t)} + \overrightarrow{a_{GPS}^s}\right]$$

Équation 1 : Equation de géoréférencement, source : cours de relevé 3D, Cali José, p.12)

où r le vecteur vitesse, g un repère, p un point de l'espace, t le temps, b est le bras de levier, s le repère du satellite, et a le vecteur accélération.

R^g_b : la matrice de rotation entre le repère g et le repère b

R^b_s : la matrice de rotation entre le repère b et le repère du satellite s

L'avantage d'un ALS est sa capacité à couvrir rapidement de vastes étendues, touffues (figure 3), marécageuses, accidentées, difficiles d'accès à l'homme par voie terrestre. Cependant les opérations aériennes restent coûteuses (car nécessitant des prestataires de services en plus du matériel) et tributaires des conditions atmosphériques. En effet, le LiDAR est sensible aux conditions atmosphériques. Par temps de pluie ou de brouillard le faisceau peut être perturbé lorsque le rayon traverse un milieu différent (comme une goutte d'eau), la vitesse de l'onde change, et ainsi la superposition de différentes couches de différents milieux rend approximatif l'évaluation de la distance [DU CHAPELET M. et al., 2016].

Les LiDAR terrestres, plus souvent appelés TLS (Terrestrial LiDAR System) pour système lidar terrestre, ou MLS (Mobile LiDAR System) pour système LiDAR mobile, sont tous les deux des scanners laser terrestres qui utilisent la technologie LiDAR. Lorsque le scanner laser est installé sur un appui tel qu'un trépied (figure 3), on parle de TLS. En revanche lorsque le scanner laser est monté sur un véhicule, un sac à dos ou tout simplement porté en main, on parle de MLS. TLS et MLS ont le même principe de fonctionnement à savoir qu'ils utilisent la technique du scanner laser terrestre pour effectuer des relevés tridimensionnels. Le laser émet une impulsion qui est réfléchie lorsque celle-ci rencontre la matière (une branche d'arbre, un tronc, une feuille, etc.). Connaissant la vitesse de la lumière et le temps retour de l'onde lumineuse, la distance entre l'émetteur laser et la cible est obtenue. Son principal avantage est la précision accrue de son nuage de points (~ 50 et 10 000 pts/m²). Plus l'acquisition faite est lente voire statique, plus les informations spatiales sont détaillées et le nuage de points dense. Ce précieux avantage entraîne toutefois un inconvénient, à savoir que sa portée est limitée et couvre des zones de petites emprises, ce qui rend le déploiement et la collecte des données plus lents et laborieux.

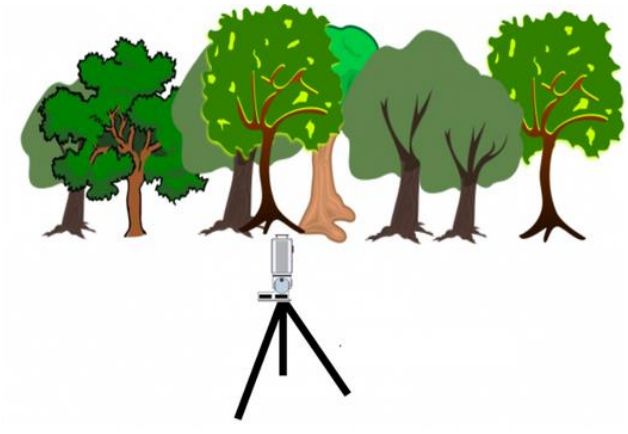

Figure 3: Exemple de lidar terrestre : Lidar fixé sur un trépied Source : eo.belspo.be

« Le résultat d'un lever LiDAR, qu'il soit terrestre ou aéroporté est un nuage de points géo-référencés, plus ou moins dense [avec une résolution plus ou moins fine] » [COCCIA S. et al., 2020]. La densité dépend de la cadence d'émission du laser (fréquence d'échantillonnage ou de tir), des dimensions et des propriétés de l'objet numérisé, et la résolution dépend de la résolution linéaire et la résolution angulaire. « Ces deux paramètres influent la résolution du levé et détermine le niveau de détail qu'on pourra observer dans le nuage de points » [COCCIA S. et al., 2020].

I.1.3 Méthodes d'acquisitions des scanners laser : scanner statique et scanner dynamique

Un scanner laser est un appareil de numérisation et d'acquisition 3D qui repose sur l'utilisation d'ondes lumineuses (ondes électromagnétiques visibles) générées par laser. Celles-ci sont projetées vers la cible à imager, qui les réfléchit [COCCIA S. et al., 2020]. Utiliser un laser suppose un éclairage actif de l'objet à mesurer. Le capteur étant indépendant des sources de lumière externes, il peut être utilisé avec la même efficacité la nuit, le jour, dans les tunnels, et aussi bien en extérieur qu'à la lumière artificielle. La puissance lumineuse réfléchie d'une impulsion laser dépend directement de la distance et de la nature de l'objet à mesurer [WEBER H., 2018]. Les scanners lasers 3D se distinguent en deux catégories principales : les scanners laser statiques et les scanners laser dynamiques. Ces deux méthodes d'acquisition possèdent des caractéristiques ayant des avantages et des inconvénients.

Les scanners statiques sont également connus sous le nom de scanners fixes. Leur nom provient du mode opératoire nécessaire à la prise de mesure. Ils sont montés sur une structure fixe, généralement un trépied pour scanner leur environnement et restent immobiles durant la capture des données. En effectuant une rotation sur un ou plusieurs axes « le scanner réalise un balayage laser [horizontal et vertical] selon le paramétrage (fréquence d'acquisition ou d'échantillonnage, résolution angulaire) défini par l'opérateur sur le terrain » [COCCIA S. et al., 2020]. Grâce à un mécanisme rotatif (figure 4), le scanner peut couvrir une zone complète autour de lui, capturant ainsi un nuage de points 3D détaillé de son environnement.

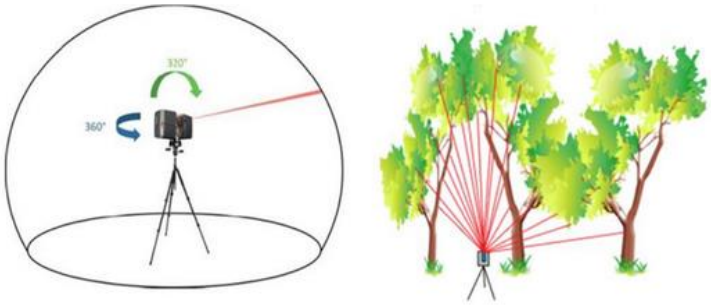

Figure 4: Exemple de méthode d'acquisition statique Source : eo.belspo.be

Les scanners lasers dynamiques, également connus sous le nom de scanners mobiles, ont la particularité de pouvoir facilement changer de support sur lequel ils sont posés pour faire une acquisition. Du satellite au cheval en passant par le vélo, une voiture ou un drone (figure 5), cette méthode d'acquisition est bien plus souple que la précédente. Conçus pour acquérir des données en mouvement et adaptés aux levés de grandes zones telles que des zones forestières, les scanners mobiles permettent une acquisition rapide et flexible d'environnements difficiles d'accès. Leur principe de mesure est similaire à celui des scanners statiques à la seule différence qu'ils sont équipés de capteurs LiDAR dont le fonctionnement repose sur une mesure de la distance permettent d'obtenir des informations aussi bien sur l'écart et la position sur l'axe X que sur les positions sur les axes Y et Z. Cependant, leur précision est inférieure à celle des scanners statiques. Si l'on considère les incertitudes maximales pour chaque technologie, 50 mm pour un MLS contre 5 mm pour un TLS. L'incertitude du Lidar mobile peut-être jusqu'à dix fois plus grande que celle du Lidar statique selon les conditions de mesure.

Figure 5 : Exemple de méthode d'acquisition dynamique à l'aide d'un scanner laser poser sur un drone Source : eo.belspo.be

Dans le cadre des travaux menés en forêt amazonienne ou équatoriale et dirigés par les chercheurs du laboratoire de recherche AMAP, la méthode d'acquisition dynamique est la plus courante car elle est la plus appropriée aux missions. En effet, par les airs ou depuis le sol, acquérir des données à l'aide d'un scanner laser mobile offre une souplesse et un gain de temps considérable tout en assurant un nuage de points de détail. Les données utilisées pour réaliser ce mémoire ont été acquises à l'aide d'un scanner laser terrestre dynamique : le système backpack (« sac à dos ») HoverMap ST-X. Sous couvert forestier, difficile d'obtenir des coordonnées GPS par triangulation, trop dense. Avec le ALS/DLS on utilise un GPS

différentiel, positionné dans une clairière à proximité, et ensuite il s'agit d'effectuer un recalage de la position du scanner en utilisant le GPS différentiel et le GPS embarqué. Cette étape est très importante et doit être réalisée avant tout les autres. Tandis qu'avec un TLS, on est statique, on arrive toujours à se localiser précisément grâce aux points de référence. Utiliser un MLS en sous-bois ne pourrait pas fonctionner à l'aide d'un GPS différentiel seul, car avec la végétation il y aurait trop d'interférence. En revanche, un MLS embarqué de l'algorithme SLAM, résout le problème car le calcul de la position du scanner se fait presque simultanément.

I.2 Présentation de la géométrie des tirs d'un système LiDAR Mobile

La géométrie des tirs d'un système LiDAR mobile est essentielle pour comprendre comment les données tridimensionnelles de l'environnement sont reconstituées. Cette partie détaille les principes de base de la géométrie des tirs et les implications de ces mécanismes sur la précision et la densité des nuages de points obtenus. En termes de géométrie des tirs, la principale différence entre un système statique et un système dynamique repose sur le mouvement du scanner lors de l'acquisition. L'étude de la géométrie des tirs d'un système LiDAR mobile se fera donc par l'étude de la géométrie des tirs d'un système statique.

La géométrie des tirs d'un système LiDAR statique vient de son acquisition et peut se décomposer en plusieurs parties (figure 6).

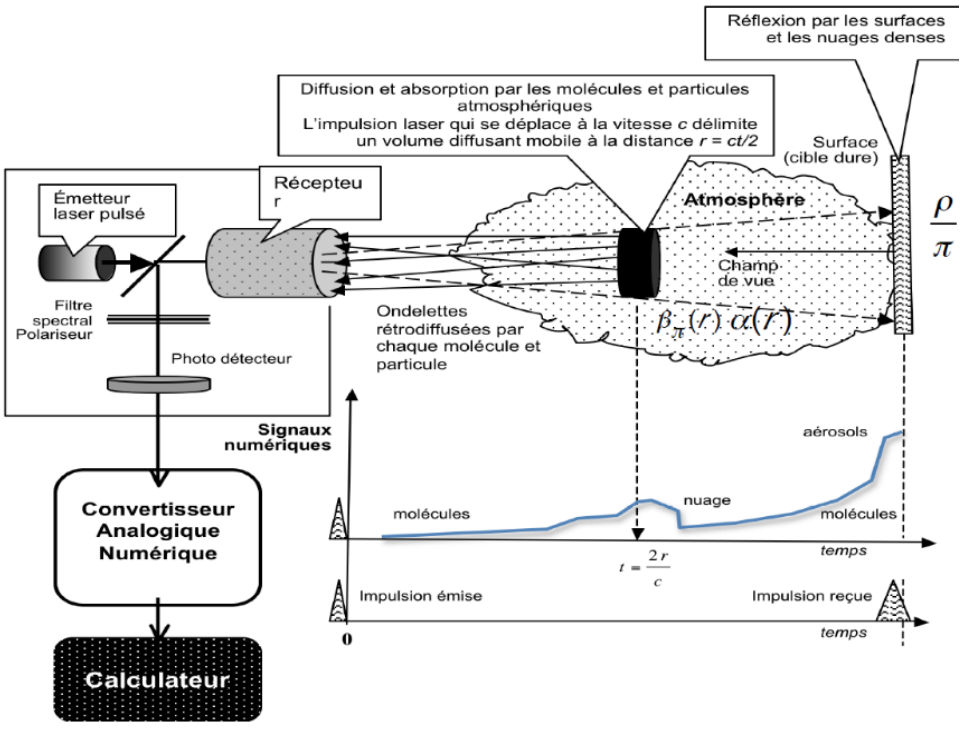

Figure 6 : Principe de fonctionnement général du LiDAR, Source : Cours 'Lidar Géophysique' [FLAMANT H., 2010]

Tout d'abord, l'émetteur laser va émettre une impulsion laser modulée en amplitude en fonction des besoins qui, en fonction de la dimension (1D, 2D ou 3D) du capteur utilisé, va permettre d'obtenir plus d'informations sur notre environnement. Pour mesurer un point, un capteur 1D va permettre, par utilisation du principe d'expansion linéaire unidimensionnelle, de déterminer la distance au point grâce au temps de vol (figure 7). Si l'impulsion est émise dans la même direction et de manière régulière, il sera aussi possible de mesurer un déplacement.

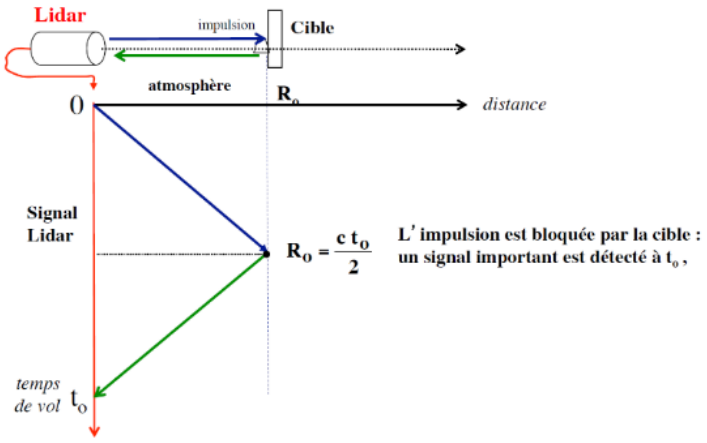

Figure 7 : Impulsion pulsée et calcul de distance, Source : Cours 'Lidar Géophysique' [FLAMANT H., 2010]

Pour mesurer une surface (en 2D), le même principe est utilisé à la différence qu'un pas angulaire souvent appelé résolution angulaire (capacité de l'appareil à mesurer de façon indépendante deux objets situés sur des lignes de visées adjacentes) va permettre d'émettre des impulsions dans deux directions de l'espace. En ce qui nous concerne, nous utilisons les capteurs 3D pour effectuer les mesures en forêt. Il en existe plusieurs types dont les plus répandus sont les multicouches et les capteurs à déviation d'angles de vue. Les capteurs à déviation d'angles (tel que HoverMap) sont basés sur le même principe que les capteurs 2D à la différence qu'un système opto-mécanique (optique et mécanique) vient dévier la ligne de visée. Les capteurs multicouches quant à eux possèdent plusieurs émetteurs et récepteurs d'impulsions ce qui leur permet de balayer simultanément plusieurs couches et sous différents angles. La diversité de capteurs 3D existante va entraîner une diversité de balayage et par conséquent une diversité du maillage des tirs (figure 8).

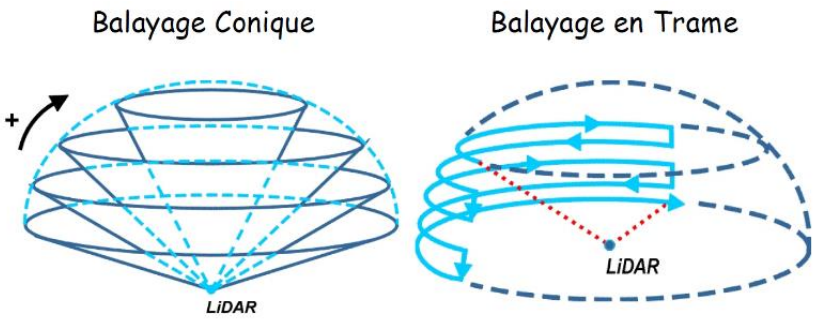

Figure 8 : Exemple de balayage LiDAR multicouches, Source : Cours 'Lidar Géophysique' [FLAMANT H., 2010]

Ensuite, le faisceau laser émis dans l'étape 1 va passer par un télescope, une optique, un photodétecteur et un convertisseur analogique numérique qui vont respectivement collecter la puissance diffusée, la filtrer, pour la mettre en forme et la convertir pour la rendre utilisable par un calculateur. Enfin, pendant la dernière étape, le calculateur utilise le signal numérique afin de restituer les informations voulues (coordonnées, distances, angles de vue, etc.) en combinant ces distances avec les angles de tir du faisceau laser.

La géométrie des tirs dans un système LiDAR mobile se réfère à la disposition spatiale des faisceaux laser et aux angles de tirs relatifs utilisés pour scanner l'environnement. Elle tient compte donc des angles de tirs et d'inclinaison, de la modélisation et de la densité des points. Les angles de tirs sont pris en compte car ce sont eux qui définissent la mécanique de rotation du capteur et la configuration des faisceaux laser inclinés à des angles spécifiques pour maximiser la couverture verticale et horizontale. Une fois nos informations obtenues, il est possible d'avoir une indication sur la précision des points. Cette précision va dépendre de plusieurs facteurs tels que, le pas d'échantillonnage angulaire (angle séparant deux tirs laser consécutifs), les conditions météorologiques et de la durée de l'impulsion. Le pas d'échantillonnage angulaire va permettre d'affiner le niveau de détails du levé. Elle est conditionnée par la densité de la forêt à mesurer et de la taille du vide à déterminer. Les conditions météorologiques vont par principe de réfraction modifier le milieu traversé par le faisceau laser et engendrer des points parasites et des imprécisions. Quant à la durée de l'impulsion, elle est proportionnelle à l'erreur engendrée, donc plus l'impulsion dure dans le temps, plus l'incertitude sur la détermination du temps de l'impulsion retour augmente et plus l'erreur est grande (figure 9).

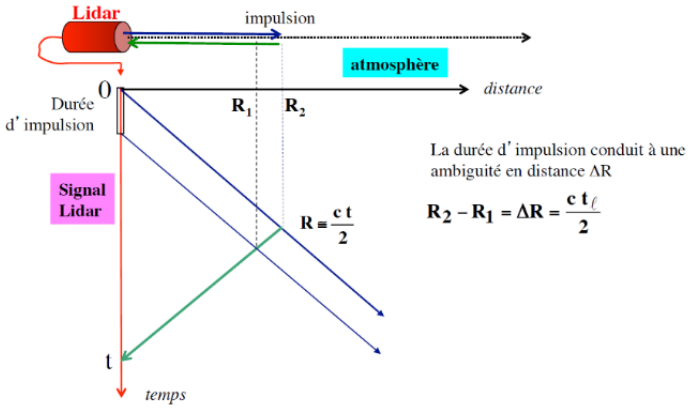

Figure 9 : Erreur due à la durée de l'impulsion laser, Source : Cours 'Lidar Géophysique' [FLAMANT H., 2010]

La géométrie des tirs d'un système LiDAR mobile est également un élément fondamental pour la précision et la fiabilité des données acquises. Les systèmes modernes, comme le HoverMap ST-X, exploitent des technologies avancées de tir et de traitement des données pour offrir des solutions robustes pour une variété d'applications industrielles et scientifiques. En combinant une couverture large avec une haute fréquence de tirs, ces systèmes permettent de capturer des nuages de points détaillés et précis. Basé sur cette étude détaillée de l'acquisition LiDAR statique, nous avons pu effectuer une simulation du maillage (avec un pas d'échantillonnage angulaire d'environ $2 \times 10^{-2}\ rad$ pour φ et $9 \times 10^{-2}\ rad$ pour θ) et des tirs réalisés lors d'un levé au LiDAR statique (figure 10).

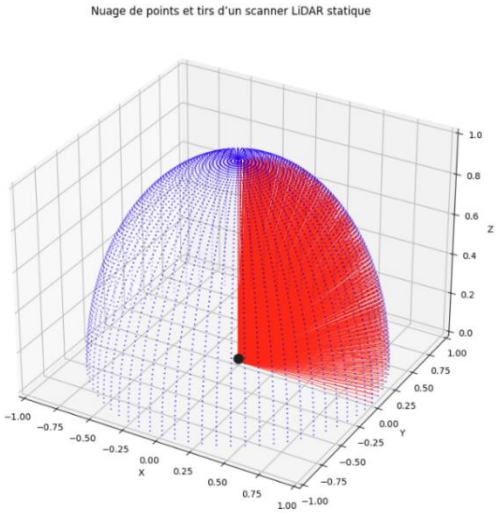

Figure 10 : Nuage de points et tirs générés par un système LiDAR statique

Les coordonnées des points dans l'espace sont déterminées en combinant ces distances avec les angles de tir du faisceau laser. La géométrie des tirs dans un système LiDAR mobile se réfère à la disposition spatiale des faisceaux laser et aux angles de tir relatifs utilisés pour scanner l'environnement. Elle tient compte des angles de tir et d'inclinaison, de la

modélisation et de la densité des points. Les angles de tir sont pris en compte car ce sont eux qui définissent la mécanique de rotation du capteur et la configuration des faisceaux laser inclinés à des angles spécifiques pour maximiser la couverture verticale et horizontale de l'espace. Quant à la modélisation des points, chaque élément dans le nuage de points est déterminé par la combinaison de la distance mesurée et des angles de tir. Leur précision dépend de la résolution angulaire (la précision avec laquelle les angles de tir sont mesurés) et de la précision sur les distances mesurées. Enfin la densité des points est aussi importante car elle dépend de la fréquence des tirs, de la vitesse de déplacement du capteur, et de la distance entre le capteur et les objets scannés.

I.3 Le système HoverMap ST-X mobile de Emesent en forêt amazonienne

I.3.1 Présentation générale de l'appareil

Le système HoverMap ST-X mobile est un scanner dynamique développé par la société australienne Emesent fonctionnant à l'aide des technologies LiDAR et SLAM (Simultaneous Localization And Mapping) qui veut dire « cartographie et localisation en simultanées ». Conçu pour offrir des solutions de cartographie et de numérisation 3D, il est particulièrement adapté aux environnements complexes et difficiles d'accès. Son dispositif intègre capteur et technologies de pointe permettant à la fois de capturer des données précises dans l'espace et de calculer sa trajectoire. Le HoverMap ST-X se distingue des autres appareils par sa capacité à être monté sur tout support, aérien comme terrestre. Cette flexibilité d'utilisation permet d'assurer une collecte de données optimale dans des conditions variées. Le mode « Backpack » du HoverMap ST-X est l'une des utilisations de ce système qui a été retenu pour l'acquisition des données sur lesquelles nous allons travailler. Il permet à l'opérateur de porter le scanner comme un sac à dos, offrant ainsi une mobilité accrue et la possibilité de cartographier des environnements inaccessibles aux véhicules ou aux drones (figure 11).

Figure 11 : Scanner HoverMap ST-X de Emesent porté sur le dos de l'opérateur, Source : emesent.com

Les paramètres qui suivent sont les caractéristiques machines du système HoverMap ST-X nécessaires à l'algorithme de reconstruction de la géométrie des tirs. Les caractéristiques du ST-X dont nous nous sommes servis sont les suivantes : le capteur LiDAR est un HesaiXT32M2X multi échos de 32 faisceaux capables d'acquérir jusqu'à plus d'un million de points par seconde à une fréquence de 0.5 Hz en extérieur pendant une durée de stockage de près de 4 heures. Il enregistre les données à 360° avec un angle d'ouverture de 30° dans un rayon de 300 m autour de l'appareil. Toutefois, il ne détecte pas les obstacles à moins de 50 cm de lui. Le système inclut également une centrale inertielle dont les capteurs permettent d'obtenir la position et l'orientation du scanner en mouvement dans le temps.

L'utilisation de l'appareil en milieu forestier n'est pas différente de son utilisation en milieu urbain. L'utilisation backpack du système HoverMap ST-X est porté sur le dos de l'utilisateur et enregistre son environnement le long de la trajectoire. Cependant, en forêt et forêt équatoriale en particulier, il faut faire face à de nombreux défis tels que : la topographie terrain qui peut être très irrégulière, la densité de végétation qui empêche de se mouvoir, les zones difficiles d'accès (coins reculées, accidentées, inondés) qu'il faut contourner, etc.

I.3.2 Principe de fonctionnement de l'appareil

Le LiDAR HoverMap ST-X fonctionne à l'aide de deux technologies de pointe : la technologie LiDAR et la technologie SLAM.

La technologie SLAM de son acronyme « Simultaneous Localization and Mapping » signifie localisation et la cartographie simultanées. C'est une technologie que l'on utilise lorsque l'on souhaite localiser et construire la carte d'un environnement situé dans une zone où le signal GNSS est très faible, indisponible voire inaccessible. Son principe de fonctionnement peut être divisé en plusieurs étapes.

La première étape est l'initialisation. A l'instant t1, le système a une position et une orientation initiale estimées. Le capteur lidar et celui de la centrale inertielle collectent des données sur l'environnement. Les données collectées forment un nuage de points représentant l'environnement. Le SLAM vient de construire une première carte approximative de son environnement.

La seconde étape est l'acquisition des données. A l'instant t2, le système se déplace à la position suivante. A mesure qu'il se déplace, il utilise les capteurs de la centrale inertielle (composé de trois accéléromètres et de trois gyromètres) pour estimer les changements de position et d'orientation. Une nouvelle position est estimée. Cette nouvelle estimation est meilleure que la première étant donné que le SLAM dispose de plus de données.

Puis à la troisième étape une correspondance de points et correction de la trajectoire est faite. L'algorithme compare les nouveaux points avec les points précédemment enregistrés afin d'identifier des correspondances. Cela aide à corriger les erreurs de position, affiner la carte et ajuster la trajectoire du système.

Enfin la quatrième et dernière étape sert à la mise à jour de la carte. La carte de l'environnement est continuellement mise à jour avec les nouvelles données, améliorant la précision au fur et à mesure que le système explore plus d'espace.

Cette procédure est répétée à chaque nouvelle position jusqu'à la fin de l'acquisition pour une carte complète et une trajectoire optimale. La particularité du HoverMap de Emesent est qu'une version rapide du SLAM tourne sur l'appareil, directement pendant l'acquisition. Le conseil du fabricant est de scanner en étoile ou un « fleur ». Pour un premier lever, on part d'un point central, on crée un premier pétale en revenant au point de départ. Au deuxième lever une deuxième pétale : on repart du même centre pour faire une autre boucle puis on revient au départ, etc. Ensuite en post traitement, on fait tourner le SLAM sur l'ensemble des « pétales » afin d'augmenter la qualité du positionnement sur l'ensemble de l'acquisition. Cependant, les calculs restent lourds et difficiles. De plus, nous ne sommes pas souverains étant donné que l'on dépend du logiciel propriétaire AURORA de Emesent.

I.3.3 Donnée de sortie du Hovermap ST-X : le nuage de points

« Nuage de points » est le terme définissant les données acquises par LiDAR [Pierre S., 2022][7]. Comme évoqué précédemment, un nuage de points est obtenu après une acquisition LiDAR ; c'est la donnée brute de sortie de toutes acquisitions LiDAR. Les données brutes

représentent souvent une énorme quantité d'informations dues à la densité des millions de points constituant les éléments de la scène scannée. C'est pourquoi le nuage de points est livré au format LAZ. LAZ est la version compressée du format binaire standard LAS (plus populaire) ayant la capacité de restaurer entièrement les données LAS d'origine sans aucune perte d'informations. Les informations attributaires attendues, sont :

- les coordonnées X, Y, Z,
- l'intensité,
- le temps GPS enregistré en temps absolu ou en temps GPS à la semaine,
- la nature et le rang de l'écho,
- l'angle de lever.

Ces données acquises par un scanner, constituent une représentation numérique précise permettant de visualiser et d'analyser des formes et des structures parfois complexes. Généralement géoréférencés, les nuages de points peuvent être exploités dans de nombreux logiciels afin de faire des MNT, MNS, etc., mais aussi différents types de calculs plus ou moins poussés. Par exemple, l'outil AMAPVox développé au sein de laboratoire qui permet de traiter et d'exploiter les données LiDAR acquises en forêts équatoriales traite déjà des données TLS, ALS, DLS (Drone LiDAR System) et plus récemment des données MLS.

II Implémentation de deux algorithmes de reconstruction pour l'outil AMAPVox

II.1 Reconstruction de la géométrie des tirs du MLS backpack HoverMap ST-X en forêt amazonienne

Dans cette partie nous allons réaliser un algorithme de reconstruction de la géométrie des tirs vides d'un système lidar mobile. Pour appréhender le problème et concevoir l'algorithme, nous proposons deux cas théoriques préliminaires : le premier une acquisition statique et le second une acquisition dynamique le long d'une trajectoire rectiligne.

II.1.1 Reconstruction de la géométrie des tirs d'un scanner lidar statique

II.1.1.1 Objectif et Acquisition

L'objectif de ce premier cas pratique consiste à reconstruire la géométrie d'un faisceau laser émis par un système lidar statique.

On considère que le levé a été réalisé avec un scanner lidar statique posé sur un trépied à 1m de hauteur avec pour caractéristiques une vitesse de rotation et une fréquence tir constante (figure 12). En partant de ce principe et à l'aide du langage de programmation Python, nous avons effectué une simulation simplifiée (sous forme de demi-cercle) d'un nuage de points obtenu après un levé statique. L'équation **(1)** est celle d'un cercle de centre (0, 0, 1), de rayon égale à 2.

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{2\,cos(\pi t)} \\ \boldsymbol{y} &= \boldsymbol{0} \\ \boldsymbol{z} &= \boldsymbol{1 + 2\,sin(\pi t)} \end{aligned} \qquad (\mathbf{1})$$

où t correspond à l'intervalle de temps durant lequel le faisceau laser parcourt le demi-cercle, il est de 1 seconde. Une impulsion est donc émise toutes les dixièmes de seconde pendant 1 seconde avec une vitesse angulaire constante de π rad.sec^{-1} (radian par seconde).

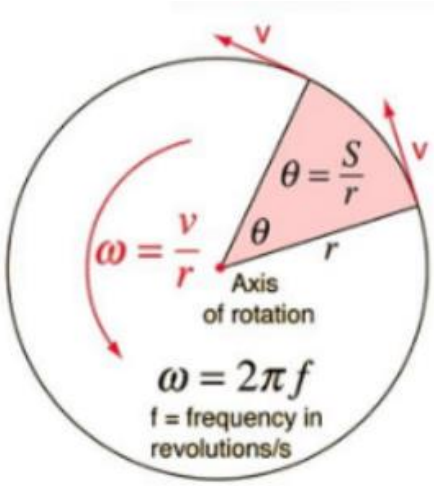

Figure 12 : Illustration de la vitesse angulaire lors d'une rotation complète (360°), Source : Javapoint.com

Pour simuler l'absence de tir, nous avons retiré aléatoirement 30% des tirs générés suivant la loi uniforme. Ces tirs supprimés représentent les impulsions laser émises et n'ayant pas rencontrés d'obstacles (les tirs vides). Ensuite, nous nous sommes servis des 70% de points restants pour retrouver par interpolation linéaire les coordonnées des vecteurs directeurs, les vecteurs directeurs des 30% de tirs vides supprimés. Cette méthode d'interpolation permet de modéliser les données avec plus de continuité et de précision. Par soucis de représentation nous avons fait le choix de projeter à « l'infini » ces points interpolés ceci afin de bien les distinguer des autres, mais surtout pour rester au plus proche de la réalité. Enfin, pour vérifier que notre interpolation fonctionne, nous avons calculé la distance euclidienne (c'est-à-dire la longueur qui sépare deux points consécutifs) entre les 30% des points dits « manquants »

mais présents à l'origine (ceux volontairement supprimés) et ceux retrouvés par interpolation et reprojetés sur le cercle de rayon 2.

II.1.1.2 Résultats et Analyse

La figure 13 nous montre le résultat de la projection à « l'infini » des tirs manquants interpolés. Nous allons présenter en détail comment a-t-elle été obtenue.

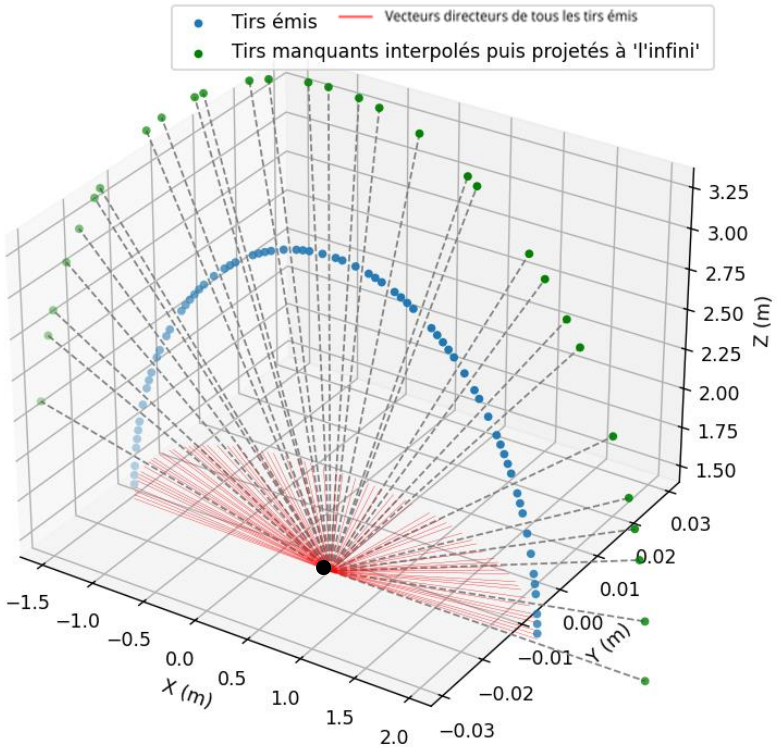

Figure 13 : Représentation de la reconstruction de la géométrie des tirs d'un système lidar statique

L'algorithme de reconstruction interpole les coordonnées x, y, z des vecteurs directeurs des tirs présents sur les temps des tirs vides. Pour valider la qualité de l'interpolation, on peut calculer à partir des vecteurs directeurs interpolés les coordonnées des points manquants sur le demi-cercle de rayon 2 et les comparer avec les coordonnées "vraies" obtenues avec l'équation paramétrique. La distance euclidienne entre les coordonnées interpolées et les coordonnées "vraies" sera la mesure de la qualité de l'interpolation.

Pour cet exercice, nous faisons le choix d'une interpolation cubique. Pour rappel, l'interpolation est une opération qui consiste à déduire mathématiquement une valeur entre deux valeurs explicitement connues. L'interpolation cubique est une interpolation par parties, c'est-à-dire que l'on utilise un polynôme de degré 3 pour modéliser localement la courbe puis déterminer la valeur ou les valeurs manquantes entre 2 données connues. Avec cette méthode

d'interpolation, la distance moyenne entre points interpolés et points "vrais" est de l'ordre de 3 micromètres.

II.1.1.3 Conclusion

Dans ce premier cas d'étude, nous avons cherché à représenter la reconstruction de la géométrie des tirs d'un système lidar statique sur un nuage de points « réduit » (semblable à un demi-cercle) et à vérifier les écarts entre les tirs reconstruits et les tirs intentionnellement supprimés. La vérification de la qualité des écarts des tirs reconstruits est nécessaire pour la validation de la méthode mathématique employée et de son application sur un cas réel par la suite. Nous avons obtenu des écarts très satisfaisant de 3.5 μm après interpolation cubique. Le résultat graphique obtenu et illustré à la figure 13 fait suite à l'implémentation de l'algorithme de reconstruction de la géométrie des tirs d'un scanner lidar statique sur Python.

II.1.2 Reconstruction de la géométrie des tirs d'un scanner lidar dynamique

II.1.2.1 Objectif et Acquisition

Ce second cas théorique reprend le cadre du premier cas théorique et consiste à ajouter une vitesse de déplacement constante le long de l'axe Y.

On suppose que le levé a été réalisé avec un scanner lidar dynamique porté par l'utilisateur à 1 mètre du sol et qui se déplace en ligne droite à la vitesse constante d'1 mètre par seconde. Nous conservons une vitesse de rotation et une fréquence d'échantillonnage constante, comme pour le premier cas théorique. La modélisation du nuage de point est obtenue avec l'équation paramétrique **(2)** d'un cercle de rayon $r = 2$ avec un déplacement d'1m/s suivant l'axe Y, Z borné à zéro, placée dans un repère tridimensionnel d'origine fixe $(0, t, 1)$, on obtient :

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{2\,cos(\pi t)} \\ \boldsymbol{y} &= \boldsymbol{t} \\ \boldsymbol{z} &= \boldsymbol{max(1 + 2\,sin(\pi t)\,, 0)} \end{aligned} \qquad (\boldsymbol{2})$$

où t correspond à l'intervalle de temps du premier au dernier tir émis, il vaut 10 secondes. Un tir est envoyé toutes les dixièmes de seconde pendant 10 secondes, avec une vitesse angulaire constante de π rad.sec^{-1} (radian par seconde).

Nous retirons aléatoirement 30% de points et nous appliquons sensiblement le même algorithme de reconstruction des tirs vides élaborés pour le cas théorique 1. L'étape nouvelle dans ce second cas théorique consiste à retrouver la position du scanner aux temps

manquants. Nous utiliserons pour ce faire une simple interpolation linéaire des coordonnées de la trajectoire du scanner (y = t, trivial dans notre cas théorique, mais applicable ultérieurement à tout type de déplacement) pour les temps manquants.

II.1.2.2 Résultats et Analyse

La figure 14 représente en bleu le nuage de point modélisé, en forme de spirale, en violet les vecteurs directeurs unitaires et en vert les points manquants interpolés et projetés “à l’infini”, en tenant compte de toutes les caractéristiques précitées.

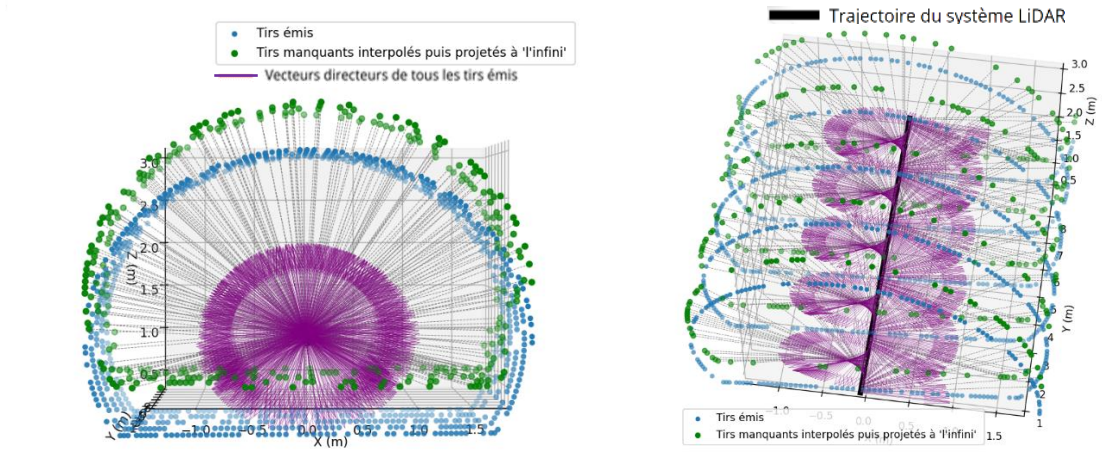

Figure 14 : Représentation de la reconstruction de la géométrie des tirs d’un système Lidar dynamique

L’évaluation de l’algorithme consiste là aussi à calculer la distance euclidienne entre les points “vrais” (obtenus à partir de l’équation paramétrique) et les points projetés à 2 m à partir des vecteurs directeurs interpolés. Les écarts moyens sont de l’ordre de 0.5 μm, ce qui est plus que satisfaisant.

Les résultats sont inférieurs à ceux du premier cas d’étude, ce qui est plus que satisfaisant.

Nous avons fait le choix d’une trajectoire rectiligne pour simplifier au mieux la problématique de la reconstruction de la trajectoire des tirs d’un système lidar mobile et pour la clarté des représentations. Une acquisition en forêt ne peut se faire du début jusqu’à la fin en ligne droite en raison notamment de la topographie de son terrain. Toutefois le plus important ici était de parvenir à l’objectif de reconstruction en prenant en compte dans les paramètres cette action de mouvement. Cela nous aidera donc à mieux aborder et à résoudre la reconstruction de la géométrie des tirs du HoverMap ST-X.

II.1.2.3 Conclusion

Dans ce second cas d'étude, nous avons cherché aussi à reconstruire la géométrie des tirs d'un système lidar dynamique à partir d'un nuage de points simplifié et à vérifier aussi les écarts entre les tirs reconstruits et les tirs intentionnellement supprimés. L'apport de ce 2nd cas théorique est la prise en compte du mouvement du scanner pendant l'acquisition. Cela implique de retrouver les coordonnées du scanner au temps des tirs manquants. Les écarts obtenus sont plus que satisfaisants puisqu'ils sont nettement inférieurs au micron et le résultat graphique obtenu figure 14, illustre bien la forme « spirale » de l'ensemble du nuage de points qu'on pourrait obtenir.

- **Synthèse**

Dans cette deuxième partie nous nous sommes attachés à la reconstruction de la géométrie des tirs d'un système lidar d'abord statique puis mobile, dans un cadre théorique simplifié, pour lequel nous connaissons les coordonnées de tout point dans l'espace et dans le temps de façon analytique. Nous avons procédé à la reconstruction de la géométrie des tirs de chaque système. D'abord une présentation de la méthode ainsi que la procédure d'acquisition ont été faites puis les différentes étapes mathématiques (équation paramétrique et interpolation cubique) ont été détaillées. Puis, nous avons effectué une vérification pour parer à d'éventuelles erreurs de calcul basées sur la distance euclidienne. La vérification s'est avérée plus que satisfaisante. A présent que nous savons qu'il est possible de parvenir à de tels résultats, nous confronterons ces procédures dans la troisième partie de ce manuscrit aux réalités du levé en foresterie en les appliquant à un nuage de points acquis avec un système lidar mobile en forêt équatoriale guyanaise.

II.2 Détection des « faux tirs vides » du nuage de points du MLS backpack HoverMap ST-X

Dans cette partie, il est question d'aborder l'objectif secondaire du stage. L'objectif est de produire un algorithme de détection des faux tirs vides du MLS HoverMap ST-X. Nous avons décomposé cette partie en deux sous parties. La première sous-partie donnera la définition des "faux tirs vides" et dans la deuxième sous-partie nous implémenterons son algorithme en respectant les paramètres fixés par le constructeur.

II.2.1 Qu'est-ce qu'un « faux tir vide » ?

Un faux tir vide correspond à tout tir dont le faisceau heurte un obstacle trop proche de la source pour pouvoir être détecté. Le système HoverMap ST-X ne détecte pas les obstacles à moins de 0.5 m et la détection entre 0.5 m et 1 m est incertaine. Dans le cadre d'une acquisition sac-à-dos en milieu forestier, cela concerne donc l'opérateur et toute la végétation proche rencontrée sur le chemin.

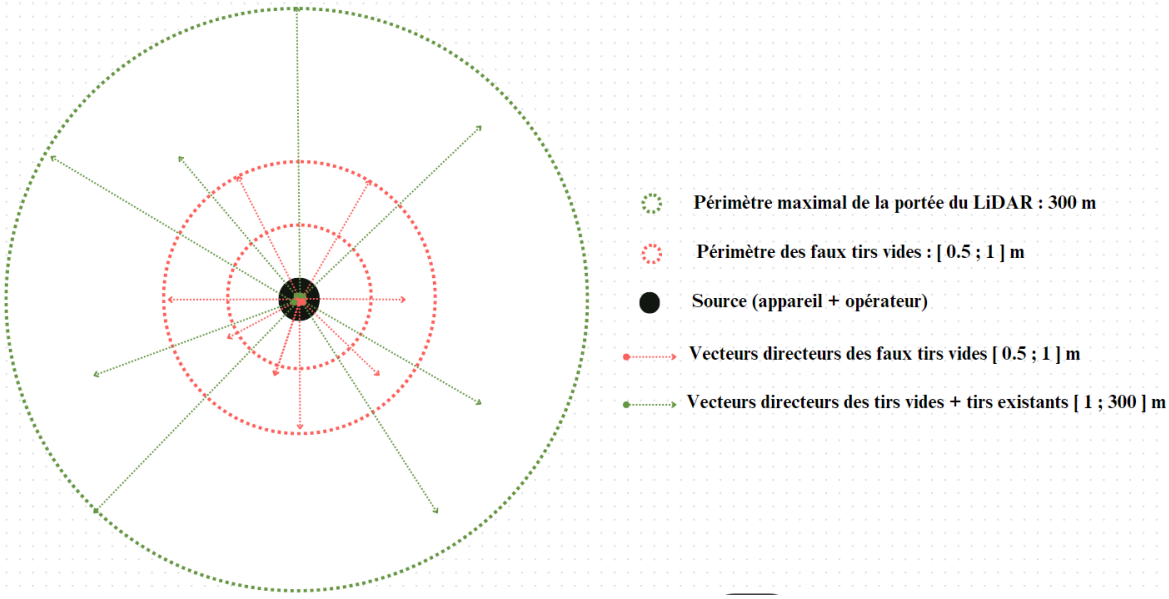

Figure 15 : Représentation du nuage de points, des tirs vides et des faux tirs vides (non à l'échelle)

En pratique, un tir sera considéré "faux tir vide" quand il traversera une zone qui compte une certaine densité de points interceptés à proximité de la source. L'algorithme de détection des faux tirs vides va devoir expliciter et quantifier des critères discriminants.

II.2.2 Implémentation d'un algorithme de détection des faux tirs vides

L'algorithme de reconstruction de la géométrie des tirs va générer les vecteurs directeurs des tirs LiDAR à tous les temps manquants. Ces tirs manquants sont considérés "vrais" s'ils correspondent vraiment à des tirs qui n'ont rencontré aucun obstacle sur leur chemin, ou "faux" s'ils ont rencontré un obstacle à moins d'un mètre du scanner et non détectés par l'appareil.

La seule information dont nous disposons est l'ensemble des points situés à proximité du scanner et les tirs qui auront traversé ces zones de proximité et qui ont généré des échos plus éloignés. A l'aide de ces points nous devons trouver un moyen de qualifier l'environnement direct d'un tir vide pour décider si ce tir vide traverse une zone relativement vide (donc un

vrai tir vide) ou au contraire une zone avec beaucoup de points donc avec des obstacles (donc un faux tir vide). Une approche statistique / probabiliste serait certainement adaptée pour un tel problème, mais nous proposons à ce stade une approche plus simple pour évaluer le potentiel de la méthode et ne pas engager des moyens de calculs trop conséquents. Le voisinage d'un tir vide sera caractérisé par le nombre de points présents dans son cylindre situé entre 50 et 150 cm de la source ; son diamètre lui, sera à définir empiriquement. La qualification du voisinage en "vide" ou "encombré" sera déterminée par une valeur seuil du nombre d'échos présents dans le cône entre 50 cm et 1 m du scanner. Cette valeur seuil sera définie empiriquement.

Cet algorithme sera précédé par trois opérations indépendantes :

1. La première consistera à éliminer tous les points situés à moins de 50 cm du scanner, considérés comme du bruit.
2. La seconde consiste à éliminer tous les tirs vides émis en direction du sol : les chercheurs de AMAP utilisent ces données principalement pour étudier le couvert forestier et la canopée, et la végétation au sol est généralement filtrée avant analyse.
3. La troisième éliminera tous les tirs vides émis en direction de l'opérateur. L'opérateur sera représenté en première approche par une parois opaque de 180 cm de haut et 80 cm de large, placée 40 cm devant le scanner. Tous les tirs vides intersectant cette paroi virtuelle seront éliminés.

i) Algorithme de suppression des points trop proches

Pour chaque point, nous calculons la distance au laser, en interpolant la trajectoire du scanner au temps T du point. Tous les points pour lesquels la distance est strictement inférieure à 50 cm sont éliminés.

- **ii) Suppression des tirs en direction du sol**

Chaque tir vide est caractérisé par un vecteur directeur $V(x_{dir}, y_{dir}, z_{dir})$. Tous les tirs vides pour lesquels z_dir est inférieur ou égal à zéro sont éliminés.

- **iii) Algorithme de suppression des tirs en direction de l'opérateur**

Dans cet exercice, les tirs émis en direction de l'opérateur désignent les tirs qui interceptent un demi-disque de 40 cm de rayon positionné orthogonalement à la trajectoire 40 cm devant

le scanner (rappelons-nous que tous les tirs orientés “vers le bas” ont déjà été éliminés à l’étape précédente).

Pour tous les temps correspondants aux tirs vides (missing_times) :

- nous calculons le vecteur directeur de la trajectoire (à l’aide d’un 2ème point de la trajectoire interpolé à *missing_times* + ε) que nous appelons n. Pour les calculs nous prendrons $\varepsilon = 0.1$ seconde,
- nous plaçons le centre du disque filtrant à 40 cm devant le scanner dans la direction du vecteur directeur de la trajectoire, le point $p(x_p, y_p, z_p)$,
- nous déterminons le point d’interception entre les tirs vides et le plan (P) défini par le point p et le vecteur normal n, que nous appelons $p_{intersect}$,
- nous calculons la distance euclidienne d entre p et $p_{intersect}$,
- nous éliminons tous les tirs pour lesquels la distance d est inférieure ou égale à 40 cm.

Pour calculer les points d’intersections entre les rayons reconstruits et le plan (P), nous écrivons l’équation du plan (P) et calculons les points d’intersections avec les tirs manquants r.

L’équation générale d’un plan est de la forme :

- $n_x(x - x_p) + n_y(y - y_p) + n_z(z - z_p) = 0$, avec $n(n_x, n_y, n_z)$ le vecteur normal au plan et $p(x_p, y_p, z_p)$ le point p du plan (P) défini ci-avant.

L’équation des tirs manquants est de la forme :

- $r(t) = p_{traj} + t \cdot v_{dir}$, où $p_{traj}(x_{traj}, y_{traj}, z_{traj})$ est la position du scanner, t un réel et $v_{dir}(x_{dir}, y_{dir}, z_{dir})$ le vecteur directeur du tir manquant.

L’intersection du rayon et du plan s’obtient après avoir déterminé t :

- $t = \frac{n_x(x_p - x_{traj}) + n_y(y_p - y_{traj}) + n_z(z_p - z_{traj})}{n_x \cdot x_{dir} + n_y \cdot y_{dir} + n_z \cdot z_{dir}}$

Une fois t trouvé, on calcule les points d’intersection $p_{intersect}$:

- $x_{intersect} = x_{traj} + t \cdot x_{dir}$
- $y_{intersect} = y_{traj} + t \cdot y_{dir}$
- $z\_intersect = z_{traj} + t \cdot z_{dir}$

II.2.3 Résultats, Analyse et Conclusion

i) Algorithme de suppression des points trop proches

La figure 16 est le résultat de l'implémentation de l'algorithme de suppression des points trop proches.

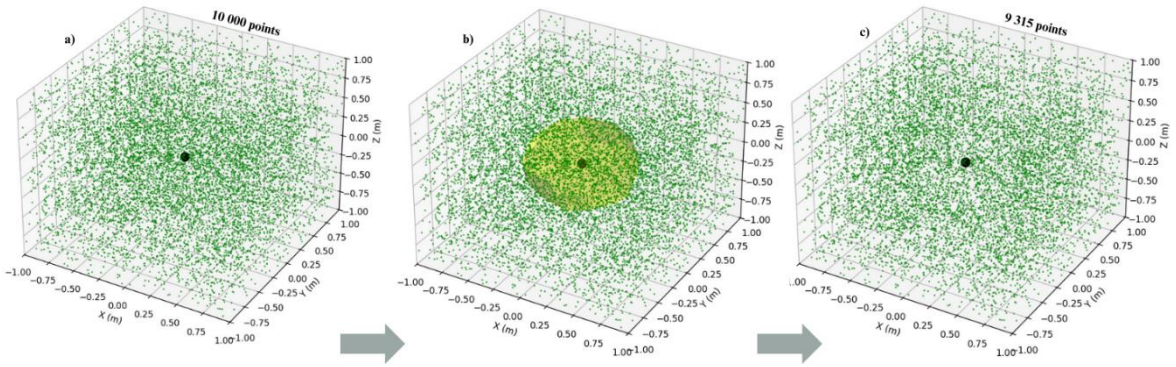

Figure 16 : Les différentes étapes d'un nuage de points jusqu'à suppression des points (en verts) trop proches du scanner (en noir).

ii) Suppression des tirs en direction du sol

L'algorithme de suppression des tirs en direction du sol permet de ne conserver que les tirs en direction de la végétation haute de la forêt, c'est l'unique partie qui fait l'objet de l'étude.

- **Conclusion**

La suppression des points trop proches du scanner, de ceux correspondants à la végétation basse ou encore ceux en direction de l'opérateur est importante car elle permet d'améliorer la qualité des données. Pour des études environnementales comme celle-ci, la précision des mesures est cruciale pour ne pas entraîner des erreurs d'estimations de paramètres (telles que la biomasse, l'indice de surface foliaire, etc.), des erreurs de calculs de volume d'objets ou encore des erreurs de modélisation de la canopée. De plus, le traitement des données LiDAR sont très souvent chronophage, en filtrant cet ensemble de points avant analyse, le volume de données à traiter est considérablement réduit, les étapes de calcul ultérieurs sont plus rapides et les estimations fidèles à la réalité. Leur implémentation aboutit à un nuage 'propre', conforme à la réalité avec des données plus précises et fiables.

III Application des deux algorithmes pour l'outil AMAPVox

III.1 Présentation du cas d'étude

Le présent cas d'étude concerne une acquisition faite sur une parcelle de la forêt amazonienne dont les données ont été acquises par le MLS HoverMap ST-X porté sur le dos de l'opérateur (figure 17).

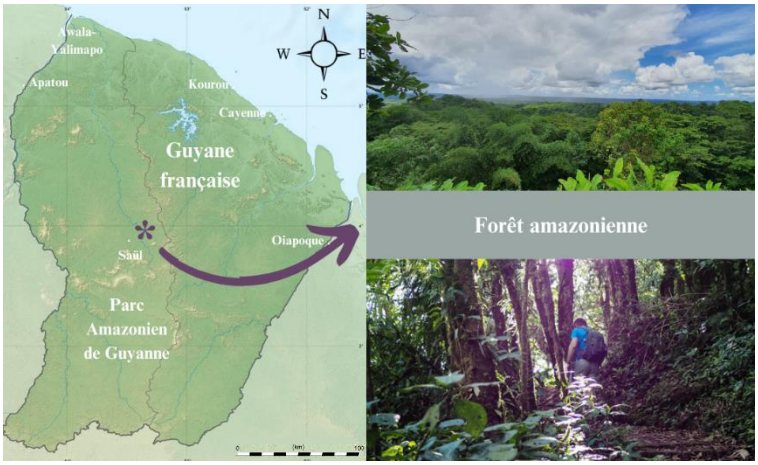

Figure 17 : Acquisition des données en forêt amazonienne à l'aide du MLS HoverMap ST-X

Comme pour les deux cas pratiques, le but de cette partie est de reconstruire tous les tirs émis par le scanner, mais en plus, de détecter les faux tirs vides afin de vérifier si les trous présents et visibles sur le nuage de points, les « troués », représentent ou non un manque d'informations sur ces parties scannées de la forêt. Pour y parvenir, nous appliquerons l'algorithme de reconstruction de la géométrie des tirs et l'algorithme de détection des faux tirs vides sur un nuage de points réel.

III.2 Acquisition des données

La phase d'acquisition a été faite par les chercheurs sur une parcelle de la forêt Amazonienne en terre guyanaise. Elle a débuté par le réglage de l'appareil puis par l'acquisition. Au cours de cette acquisition, le scanner et le LiDAR enregistrent respectivement les coordonnées de leurs positions et les coordonnées de leurs points, mais aussi leur temps GPS dit « *gpstime* ». L'acquisition a produit un nuage de points de la forêt ainsi que la trajectoire parcourue par l'utilisateur (figure 18)

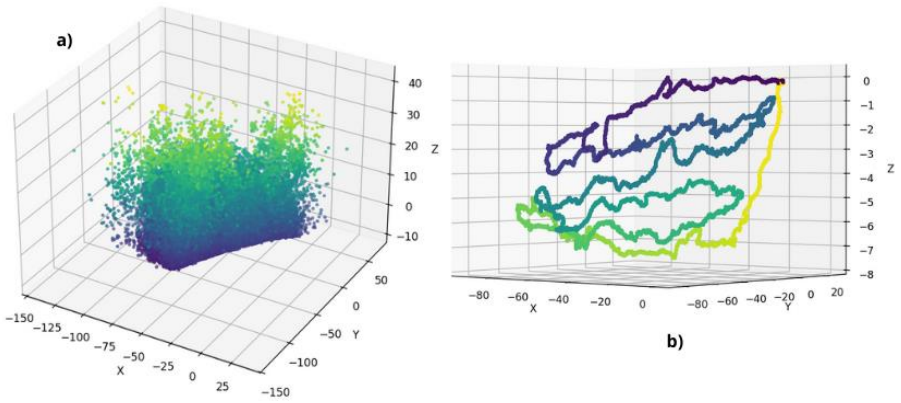

Figure 18 : a) Nuage de point d'une parcelle de la forêt amazonienne dans un système de coordonnées local ; b) Trajectoire suivie par le scanner sur le dos de l'opérateur ; (le système de coordonnées utilisé est local)

Avant application notons les différences notables :

- Le scanner est multi-échos, il faudra donc séparer les points dans le nuage pour les traiter par faisceau,
- La trajectoire du scanner n'est pas rectiligne,
- Le nuage de points et la trajectoire proviennent de deux fichiers distincts, enregistrés à des valeurs de temps différentes du fait d'une fréquence d'échantillonnage différente.

Comme données en entrée, nous disposons des deux fichiers de données : le fichier XYZ, qui contient les coordonnées de la position du scanner tout au long de l'acquisition et le fichier LAZ, qui contient notamment le temps et les coordonnées des points du nuage de points et le numéro du faisceau.

En vue de simuler la reconstruction de la géométrie des tirs sur notre cas d'étude, nous procédons en plusieurs étapes :

- Il faut d'abord identifier "les trous", i.e. les valeurs du GPS time manquantes, pour lesquelles il y a des tirs manquants.
- Nous calculons ensuite la position du scanner à tous les points GPS des tirs existants et manquants.
- On peut alors définir les vecteurs directeurs des tirs existants et interpoler les coordonnées des vecteurs directeurs des tirs manquants aux valeurs de temps manquantes.

- Enfin nous projetons à l'infini (quelques centaines de mètres en pratique) les tirs manquants pour obtenir des coordonnées des points manquants

III.3 Application des deux algorithmes et Résultats

III.3.1 Application et Résultats de l'algorithme de reconstruction de la géométrie des tirs du MLS

L'application de cet algorithme se fera en plusieurs étapes. Des paramètres de calcul ont été mis en place à chaque phase pour vérifier l'exactitude des résultats. Nous commencerons par présenter ces paramètres, puis, les résultats issus de l'expérimentation.

Mais avant cela, notons deux remarques préliminaires importantes :

1. Pour chacune de ces étapes nous traitons de façon séparée et indépendante chaque faisceau. Le nuage de point final sera obtenu grâce à une boucle de l'algorithme sur les 32 faisceaux.
2. Les raisonnements s'appuient sur l'hypothèse - raisonnable puisque c'est une donnée constructeur - que les différentes rotations du scanner se font à vitesse angulaire constante.

➢ **Identification des temps GPS manquants**

Cette partie consiste à identifier et quantifier les temps GPS des tirs manquants. Elle se décompose en plusieurs étapes :

- Identifier les « trous » dans le vecteur *gpstime*, c'est-à-dire les intervalles de temps dt supérieurs à la fréquence d'échantillonnage de temps originale (***feo***) du scanner. Nous le traduisons mathématiquement par la formule :

$$trou = dt > 1.2 \times dt_{\min} \qquad (\boldsymbol{3})$$

  avec $dt_{\min}$ le plus petit intervalle de temps (ou la moyenne des 100 plus petits intervalles de temps) qui correspond nécessairement à un intervalle de temps entre deux tirs consécutifs.

- Calculer ***feo***, la fréquence d'échantillonnage entre tirs émis consécutifs :

$$feo = \frac{\sum \ (dt < 1{,}2 * dt_{\min})}{effectif\ total}$$

- Déterminer la valeur entière du nombre de tirs manquants dans chaque trou :

$$n_{trou} = \left(\frac{dt_{trou}}{feo}\right) - 1$$

- Estimer les temps manquants :

$$t_{trou} = t_0 + i \times feo_{loc}$$

avec t_0 la valeur de *gpstime* précédent le trou,

i un entier variant de 1 à n_{trou} ,

et feo_{loc} la fréquence d'échantillonnage recalculée localement

$$feo_{loc} = \frac{t_1 - t_0}{n_{trou}}$$

- Calculer f et, la fréquence d'échantillonnage entre tirs manquants consécutifs :

$$fet = \frac{\sum \quad (dt_{trou} < 1{,}2 \times dt_{\min} \quad)}{effectif\ total}$$

- Fusionner les temps *gpstime* et les temps manquants t_{trou} et calculer la fréquence d'échantillonnage ***fef*** du vecteur temps fusionné.

- Calculer les erreurs relatives de ***fet*** et ***fef*** par rapport à ***feo***.

Nous attendons que ces fréquences d'échantillonnages soient quasi égales et que les erreurs relatives soient inférieures à 10^{-3}.

➢ Calcul de la position du scanner à tous les temps LiDAR

Pour l'étape du calcul de la position du scanner sur le temps du LiDAR nous utilisons la méthode d'interpolation '*interp*' de la bibliothèque numpy. La fonction 'np.interp' de numpy est une interpolation linéaire unidimensionnelle pour des points d'échantillonnage augmentant de manière monotone. Elle prend en entrée un ensemble de points connus et une série de valeurs à interpoler, puis elle renvoie les valeurs interpolées correspondantes. Cette interpolation nous permet d'avoir les coordonnées de la position de la trajectoire du scanner au temps GPS du LiDAR.

- **Calcul des vecteurs directeurs**

Lorsque l'interpolation de la trajectoire est correctement réalisée et que les coordonnées de la trajectoire sont bien au temps GPS du LiDAR, on procède au calcul des vecteurs directeurs unitaires de tous les points existants du nuage de points.
Nous interpolons ensuite les coordonnées des vecteurs directeurs pour les valeurs de temps GPS manquantes.

- **Vérification de la qualité de l'interpolation**

Dans la dernière étape on isole un nuage de points de vérification (***ndv***) de 1 000 pris aléatoirement dans le nuage de points. Nous appelons ***ndm*** le nuage de points avec ces 1 000 valeurs en moins. Nous appliquons l'algorithme de reconstruction des tirs manquants au nuage de points *ndm* et nous pourrons comparer les vecteurs directeurs (***vd_verif***) des points du nuage de points de vérification *ndv* avec les valeurs issues de l'interpolation (***vd_verif_i***). Nous regarderons en particulier si les vecteurs directeurs du *ndv* (calculés 'normalement') et les vecteurs directeurs du *ndm* (calculés comme des tirs 'manquants' par notre algorithme) sont colinéaires. Le troisième et dernier critère est donc celui de la vérification de colinéarité entre les vecteurs directeurs *vd_verif* et *vd_verif_i*.

- **Résultats**

Après application de l'algorithme nous obtenons les résultats rassemblés dans le tableau 1.

PARAMETRES DE CALCUL	FORMULES	RESULTATS
1er paramètre **Erreur relative sur les trous**	$\frac{\|\|feo - fet\|\|}{feo}$ $(\boldsymbol{e2})$	6.2×10^{-9}
2nd paramètre **Erreur relative sur la fusion des temps**	$\frac{\|\|feo - fef\|\|}{feo}$ $(\boldsymbol{e3})$	2.2×10^{-10}
3e paramètre **Colinéarité entre deux vecteurs :** ***collinearity***	$\|\| 1 - v1 \cdot v2 \|\|$ $(\boldsymbol{e4})$	1.9×10^{-10}
Angle θ entre deux vecteurs	$\arccos\left(\frac{v1 \cdot v2}{\|\| v1\|\| \cdot \|\| v2 \|\|}\right)$ $(\boldsymbol{e5})$	10^{-3} (degré)

Tableau 1 : Résultats de l'application de l'algorithme de reconstruction de la géométrie des tirs du HoverMap ST-X

III.3.2 Application de l'algorithme de détection des faux tirs vides

- **iv) Algorithme de suppression des faux tirs vides**

Comme nous l'avons expliqué en début de section, les faux tirs vides désignent des tirs manquants pour lesquels on considère qu'il est probable que le "vide" corresponde plus à un obstacle non détecté qu'une absence d'obstacle. La discrimination opère par analyse du voisinage du tir vide, en regardant les points détectés par le scanner à proximité.

Nous définissons le proche voisinage du tir manquant par le cylindre circulaire droit d'axe le vecteur directeur du tir $v_{dir}(x_{dir}, y_{dir}, z_{dir})$, de rayon $r = 5\ cm$ (arbitraire) et de hauteur $h = 1\ m$ pris entre 50 cm et 150 cm du scanner.

Pour les 32 faisceaux tirés à un même temps t (manquants ou pas), nous calculons la densité φ_i du voisinage du tir manquant comme étant le nombre de points dans le cylindre, divisé par le volume du cylindre.

Nous calculons également la densité moyenne φ_{moy} des points dans la demi-sphère centrée sur le scanner et de rayon $r = 150\ cm$.

Nous calculons ensuite l'écart type σ des densités des voisinages par rapport à la densité moyenne.

Pour chacun des tirs manquants, nous réalisons un tirage aléatoire φ_{alea} à partir de la loi Normale (φ_{moy}, σ) et nous disons que si φ_{alea} est inférieur à φ_i, alors le tir est conservé.

III.4 Analyse des résultats de l'algorithme de reconstruction de la géométrie des tirs du MLS

III.4.1 Analyse des résultats de l'algorithme de reconstruction

Les calculs sur les fréquences d'échantillonnage montrent que nous détectons correctement les tirs vides avec des erreurs relatives inférieures à 10^{-3} sur la fréquence d'échantillonnage des temps des tirs vides.

Le troisième critère consistait à la vérification de la qualité de notre interpolation des vecteurs directeurs. Pour cela, nous avons calculé les angles et le degré de colinéarité entre les vecteurs directeurs *vd_verif* et *vd_verif_i* de 1000 points pris aléatoirement dans le nuage origine.

Lorsque le degré de non-colinéarité (**e4**) vaut 0, cela indique que les vecteurs sont colinéaires et pointent dans la même direction. En revanche, lorsqu'il vaut 1 ou s'en approche, alors les

vecteurs sont orthogonaux. Après calcul du degré de colinéarité, nous avons représenté les résultats pour un sous-échantillonnage du nuage de points de vérification (figure 19).

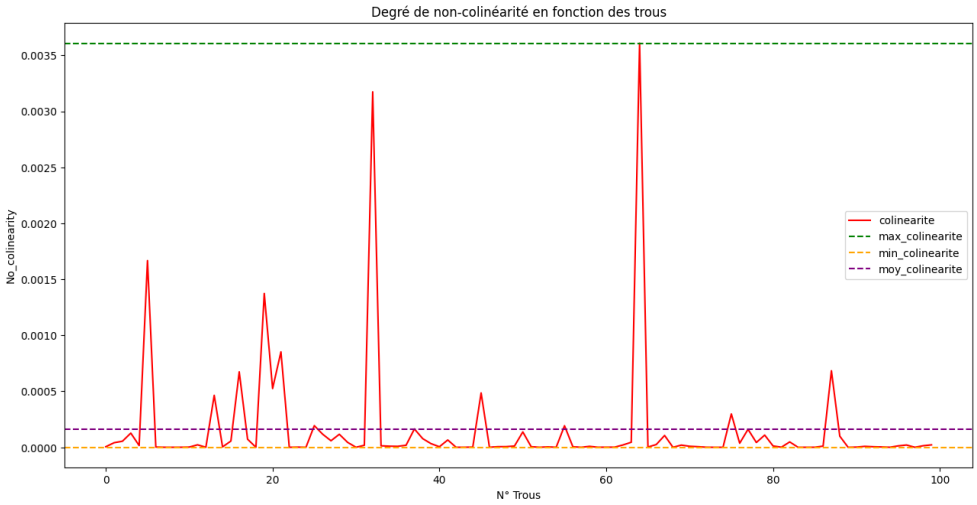

Figure 19 : Représentation du degré de non-colinéarité en fonction d'une partie des trous détectés

Le graphique illustre la variation du degré de non-colinéarité en fonction des trous détectés par l'algorithme de reconstruction des tirs. Comme l'illustre la figure, la majorité des valeurs sont proches de 0, ce qui indique que les vecteurs directeurs calculés et interpolés sont quasiment colinéaires. Ces résultats sont conformes aux exigences fixées à 10^{-3} par le commanditaire.

Malgré la conformité des résultats, on observe tout de même quelques pics importants. En effet, aux des indices 6, 33 et 65 la non-colinéarité augmente significativement jusqu'à 3.6×10^{-3}. Ces pics représentent les zones où l'erreur d'interpolation des vecteurs directeurs et des trous est la plus marquée. Les pics observés pourraient indiquer des régions du nuage de points où l'interpolation est moins précise dû à des irrégularités dans la distribution des points ou à des points qui seraient mal détectés car trop proche du scanner. Cependant, même dans ces cas, la non-colinéarité reste en deçà du seuil de tolérance dans la grande majorité des points.

Le fait que la plupart des valeurs soient proches de 0 confirme la fiabilité de l'interpolation utilisée pour reconstruire la géométrie des tirs. Quant à la présence des pics, elle pourrait induire le besoin d'explorer des méthodes d'interpolation alternatives ou des ajustements locaux pour ces régions particulières. Une investigation sur la distribution des trous et leur corrélation avec les pics pourrait également offrir des connaissances sur les facteurs

influençant la précision de l'interpolation. Une validation visuelle de notre méthode d'interpolation donne des résultats satisfaisants (figure 20).

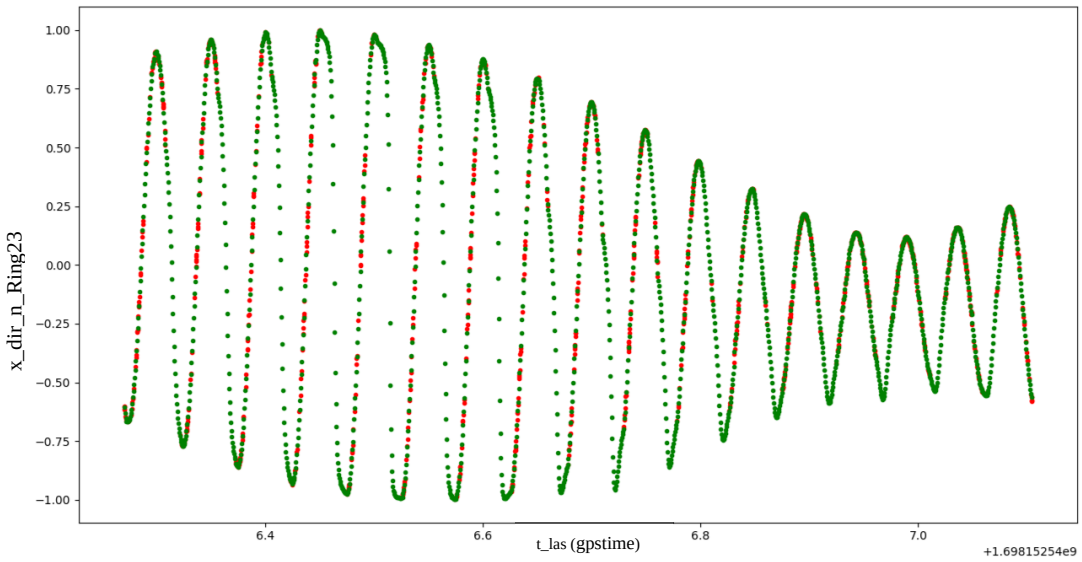

Figure 20 : Visualisation de l'interpolation de la coordonnée x des vecteurs directeurs pour le faisceau 23

La dernière étape de l'algorithme consiste à représenter les tirs vides par des points projetés "à l'infini", en pratique nous les projetons à 500 m du scanner pour la production du nuage de point final et pour la clarté de l'illustration nous les projetons à 50 m du scanner. Voici à la suite une représentation du nuage de points d'origine et les tirs vides projetés à 50 m.

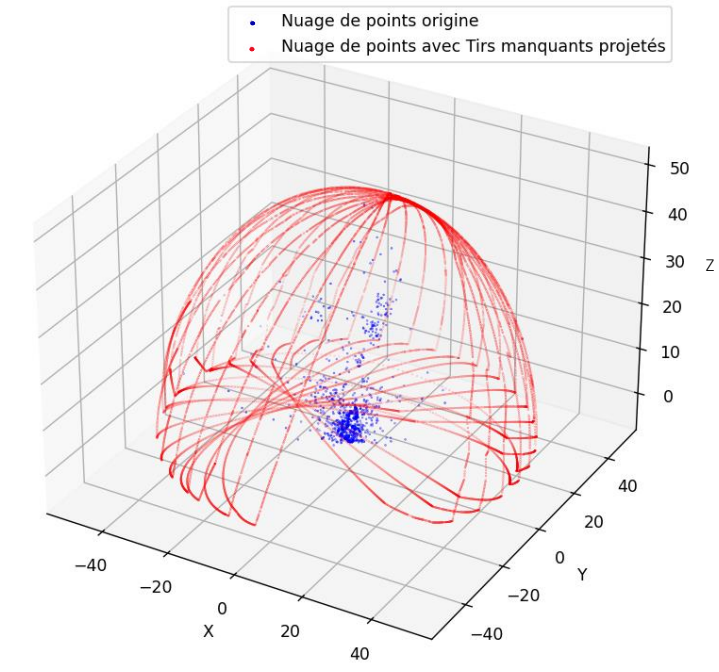

Figure 21 : Nuage de points d'origine et les tirs vides projetés à 50 m

III.5 Conclusions et Perspectives des deux algorithmes

III.5.1 Conclusions

La section **III.1** avait pour but d'appliquer notre processus de reconstruction de la géométrie des tirs à un cas réel de levé en forêt amazonienne. Suite à cette application, il en ressort que le processus permet de générer des tirs manquants avec des erreurs faibles (**i**) sur les temps d'émission (erreur relative de la fréquence des tirs inférieure 10^{-3}) et (**ii**) les directions des tirs (degré de colinéarité inférieure 10^{-3}).

L'application du premier algorithme a permis de définir avec précision les fréquences d'échantillonnage, de détecter les trous de manière fiable, et de calculer les vecteurs directeurs correspondants. Ce processus s'est avéré à la fois robuste et fiable, confirmant ainsi sa pertinence pour des applications de recherche. En conclusion, cet algorithme peut être efficacement utilisé dans les projets du laboratoire UMR AMAP.

Dans un second temps, ce travail de fin d'études visait à mettre en place un algorithme de détection de faux tirs vides. Dans le temps imparti, nous avons réalisé deux étapes de nettoyage préliminaires des tirs vides : l'une qui permet la suppression des points trop proches de l'opérateur et l'autre qui permet la suppression des tirs vides émis en direction de l'opérateur. L'implémentation en python pour la détection des faux tirs vides reste à faire, sur la base de l'algorithme proposé dans la partie méthode.

III.5.2 Perspectives

Les algorithmes de reconstruction des tirs ont été appliqués à un nuage de point issu du scanner MLS HoverMap ST-X. Une première perspective consisterait à appliquer ces algorithmes à d'autres acquisitions avec le même scanner puis, dans un second temps, de généraliser la reconstruction des faux tirs vides à d'autres scanners, statiques et mobiles. En particulier le logiciel AMAPVox intègre présentement la reconstruction de la géométrie des tirs manquants pour des LiDAR TLS statiques FARO et LEICA. C'est une étape coûteuse en temps de calcul, réalisée à la volée et donc répétée à chaque simulation du logiciel AMAPVox. Notre algorithme appliqué à ces données permettrait de sortir la reconstruction des tirs manquants du cœur de calcul d'AMAPVox et d'y rajouter la détection des faux tirs vides.

L'algorithme de détection des faux tirs vides proposés repose sur une approche statistique très simple basée sur la variabilité spatiale de la densité des points proches du scanner. Le sujet mériterait d'être revisité avec un statisticien, en particulier pour évoquer la pertinence d'une approche bayésienne.

Les implémentations réalisées en python dans le cadre de ce stage étaient de nature exploratoire car tout était à faire. Une autre perspective consisterait à formaliser les développements sous la forme d'un module python, documenté et mis à disposition de la communauté.

Conclusion

Le LiDAR est une technologie qui est en constante évolution. Basée sur le principe de la télédétection active par émission-réception d'impulsions laser, elle est de plus en plus utilisée pour représenter la structure tridimensionnelle de la végétation. Au vu des conditions particulières qui régissent les levés LiDAR en foresterie de ces dernières années, les chercheurs du laboratoire de recherche AMAP de ce secteur se retrouvent contraints d'utiliser des méthodes de mesures de plus en plus variées. Récemment, le laboratoire s'est doté du nouveau MLS HoverMap ST-X dont il s'est muni lors de sa dernière campagne en Guyane. Ce scanner dynamique a suscité un vif intérêt auprès des scientifiques en raison de sa facilité d'utilisation (très pratique pour des couverts forestiers tropicaux humides tel que la forêt amazonienne) et sa capacité à se déplacer aussi bien dans les airs que sur la terre. Cependant, malgré ces avantages, il présente un inconvénient. Les tirs sans échos ne sont pas stockés dans les sorties de HoverMap et cela représente pour les chercheurs un manque d'informations qui peut entraîner des erreurs d'estimations des paramètres de caractérisation des couverts forestiers.

Nous avons produit un premier algorithme qui permet de reconstruire la géométrie de tous les tirs du MLS pour pouvoir restaurer de façon exhaustive tous les tirs n'ayant pas reçu d'échos. Dans un deuxième temps nous avons écarté tous les tirs vides qui n'ont pas de réalité physique ou pas de pertinence dans le cadre des analyses écologiques menés par les chercheurs à AMAP : les tirs vides dirigés vers le sol et les tirs dirigés vers l'opérateur. Pour terminer nous avons proposé un algorithme de détection des faux tirs vides basé sur l'analyse spatiale de la densité de point à proximité du scanner.

Nous espérons que ces travaux constituent une première étape - une preuve de concept - dans la réalisation d'une chaîne de traitement des données LiDAR MLS acquises régulièrement en forêt guyanaise, pour leur utilisation en écologie forestière tropicale.

Bibliographie

DOCEUL C., Caractérisation de la végétation de Rennes Métropole par relevé LiDAR en vue de sa modélisation. Mémoire ingénieur Cnam : ESGT, 2017, 84 p.

MORDRANT G., Seuils de détection des anciens sites de l'architecture Maya basés sur l'imagerie LiDAR : identification des sites sous la canopée. Mémoire ingénieur Cnam : ESGT, 2015, 95 p.

LARDILLEUX J., Mise en place d'un protocole pour découvrir l'architecture des anciens sites Maya basé sur les images LiDAR : modèles d'utilisation du paysage dans la forêt. Mémoire ingénieur Cnam : ESGT, 2014, 77 p.

QUERE G., Détection des objets dans un environnement urbain à partir de données Lidar. Mémoire ingénieur Cnam : ESGT, 2020, 89 p.

SCHOLKOPL V., Estimation de la biomasse forestière du Brésil à partir des données LiDAR. Mémoire ingénieur Cnam : ESGT, 2020, 68 p.

FOUCHE G., Le scanner mobile ZEB-HORIZON : nouvel outil pour la réalisation des PCRS. Analyse du couplage de méthodes de levés pour identifier et déterminer les différentes erreurs du LIDAR. Mémoire ingénieur Cnam : ESGT, 2020, 82 p.

DIOMANDE M., Etude de la technologie SLAM du Backpack de Leica et optimisation de la chaine de traitement des données 3D. Mémoire ingénieur Cnam : ESGT, 2024, 62 p.

AMADIEU N., Étude comparative du système d'acquisition dynamique BLK-To-Go avec d'autres systèmes de cartographie mobile d'intérieur. Mémoire ingénieur Cnam : ESGT, 2022, 80 p.

RICARD B., ZEB-REVO : étude d'un scanner dynamique mobile et des ses applications au sein d'un cabinet. Mémoire ingénieur Cnam : ESGT, 2018, 61 p.

POREBA M., Qualification et amélioration de la précision de systèmes de balayage laser mobiles par extraction d'arêtes. Paris : Mines ParisTech, 2014, 179 p.

HOUSSEM N., Affinement de relevés laser mobiles issus de LIDARs multi-couches. Paris : Mines ParisTech, 2017, 142 p.

BEN HMIDA S., Inversion des formes d'ondes LiDAR pour l'estimation des caractéristiques des cultures et des forêts par des techniques probabilistes et variationnelles. Mémoire de thèse Université de Toulouse, 2018, 158 p.

CEZARD N., systèmes LIDAR pour la caractérisation à distance des propriétés physico-chimiques de l'atmosphère. Mémoire de thèse : Université de Toulouse, 2021, 153 p.

Flammant P., 2019, Comprendre la science du lidar. Sorbonne Université, 97 p.

Articles

ERZHUO C. et al., An Efficient Framework for Mobile Lidar Trajectory Reconstruction and Mo-norvana Segmentation. In : MDPI. [en ligne]. Disponible sur <https://www.mdpi.com/search?q=An+Efficient+Framework+for+Mobile+Lidar+Trajectory+Reconstruction+and+Mo-norvana+Segmentation>. (consulté le 05/05/2024)

Monnet J-M. et al., La télédétection aéroportée pour la gestion des territoires forestiers de montagne. [en ligne]. Disponible sur <https://revue-set.fr/article/view/6926>. (consulté le 08/05/2024)

POPULUS J. 2002, Altimétrie par Lidar aéroporté Et Modèles Numériques de Terrain. Science. DEAL

COCCIA S. et al., 2020. LiDAR terrestre à longue portée : retour d'expérience dans le contexte d'instabilités de pente, 48 p.

WEBER H., 2018. Fonctionnement et variantes des capteurs LiDAR. SICK AG, Allemagne. 16 p.

DU CHAPELET M. et al., 2016. Etat de l'art, application en perception de l'environnement pour le véhicule autonome. INSA, 24 p.

Sites web

Fonctionnement du LiDAR. In : Natural solutions. [en ligne]. Disponible sur <https://www.natural-solutions.eu/blog/fonctionnement-du-lidar>. (consulté le 10/05/2024)

What is a LAS file? In : Laspy. [en ligne]. Disponible sur <https://laspy.readthedocs.io/en/latest/intro.html>. (consulté le 10/05/2024)

Scipy Interpolate.LinearNDInterpolator. In : SciPy documentation [en ligne]. Disponible sur <https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.LinearNDInterpolator.html#scipy.interpolate.LinearNDInterpolator>. (consulté le 12/05/2024)

Scipy Interpolate.1D. In : SciPy documentation [en ligne]. Disponible sur < https://docs.scipy.org/doc/scipy/tutorial/interpolate/1D.html>. (consulté le 12/05/2024)

Interpolate any trajectories from waypoints in python. In : Python. [en ligne]. Disponible sur <https://python.plainenglish.io/interpolate-any-trajectories-from-waypoints-in-python-d2127727baa4>. (consulté le 06/05/2014)

3d-FOREST mieux connaitre les forets tropicales grace au liDAR. In : Belgian Earth Observation. [en ligne]. Disponible sur <https://eo.belspo.be/fr/actualites/3d-forest-mieux-connaitre-les-forets-tropicales-grace-au-lidar>. (consulté le 02/06/2024)

LiDAR Trajectory Reconstruction. In : Oregon State University. [en ligne]. Disponible sur <https://oregonstate.technologypublisher.com/tech/LiDAR_Trajectory_Reconstruction>. (consulté le 03/06/2024)

Table des annexes

Annexe 1
Code de l'algorithme de Reconstruction de la géométrie des tirs d'un Système Lidar Mobile

```
import numpy as np
import laspy
import time
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Lecture du fichier du scanner et import de ces données
fichier_xyzt = 'P16_C14152_01_traj.xyz'
data = np.loadtxt(fichier_xyzt, delimiter=' ')
t_traj = data[:, 1]      #GPSTIME
x_traj = data[:, 2]      #X
y_traj = data[:, 3]      #Y
z_traj = data[:, 4]      #Z
start_time = time.time()

# Lecture du fichier du LiDAR et import de ces données
las = 'P16_C14152_01_subsampled_laz1_4.laz'

"""     boucle des faisceaux    """
# liste des tirs manquants projetés pour tous les Ring
x_proj_all_ring = []
y_proj_all_ring = []
z_proj_all_ring = []

for ring_number in range(32):
```

```
# Filtrer les points correspondant au faisceau actuel
filter_ring = las.Ring == ring_number
t_las = las.gps_time[filter_ring]
t_las, u_index = np.unique(t_las, return_index=True)

x_las = las.x[filter_ring][u_index]
y_las = las.y[filter_ring][u_index]
z_las = las.z[filter_ring][u_index]
"""     ***     """

"""     1 seul faisceau     """
# ring_number = 23
# t_las = las.gps_time[las.Ring == ring_number]
# # u, indices = np.unique(a, return_index=True)
# u = unique pr éliminer les doublons
# t_las, u_index = np.unique(t_las, return_index= True)
"""         ****         """

# Détermination des fréquences d'échantillonnage
dt = np.diff(t_las)
dt = np.round(dt * 100000) / 100000
min_dt = np.min(dt)

# Fréquences d'échantillonnage origine ‘feo’
# Recherche des intervalles de temps ('dt') sans trous
dt_sans_trou = dt[dt < 1.2 * min_dt]
dt_mean_sans_trou = np.mean(dt_sans_trou)

# Histogramme de dt
histo_dt = np.round(dt / dt_mean_sans_trou)
histo_dt = histo_dt.astype(int)
```

```
n_missing_t = histo_dt - 1

# Recherche des dt avec trous
dt_avec_trou = dt[dt >= 1.2 * min_dt]
dt_avec_trou_indices = np.where(dt >= 1.2 * min_dt)[0]
dt_mean_avec_trou = np.mean(dt_avec_trou)
tps_avec_trou = t_las[dt_avec_trou_indices]
l_avec_trou = list(tps_avec_trou)

# Détermination des ‘gpstime’ des tirs manquants
#  dt local
missing_times = []
for i, n_missing in enumerate(n_missing_t):
    t0 = t_las[i]
    t1 = t_las[i+1]
    dt_loc = (t1-t0) / (n_missing + 1 )
    for j in range(1, int(n_missing) + 1):
        missing_times.append(t0 + j * dt_loc)

# Vérification des missing times
missing_times = sorted(set(missing_times))
missing_times = missing_times[0:1000]
missing_times_dt = np.diff(missing_times)
min_missing_times_dt = np.min(missing_times_dt)
dt_mean_missing_times = np.mean(missing_times_dt[missing_times_dt < 1.2 * min_missing_times_dt])

# Fusionner et trier les 2 listes de temps GPS
fusion_times = sorted(set(l_avec_trou + missing_times))
dt_fusion_times = np.diff(fusion_times)
min_dt_fusion_times = np.min(dt_fusion_times)
dt_mean_fusion_times = np.mean(dt_fusion_times[dt_fusion_times < 1.2 *  min_dt_fusion_times])
```

```
dt_mean_fusion_times_arrondi = np.round(dt_mean_fusion_times * 100000)

"""     interpolation de la trajectoire du scanner     """
t_las = las.gps_time      # modifie obligé de conserver cette ligne sinon pb de dim
x_traj_i = np.interp(t_las, t_traj, x_traj)
y_traj_i = np.interp(t_las, t_traj, y_traj)
z_traj_i = np.interp(t_las, t_traj, z_traj)

x_dir = las.x - x_traj_i
y_dir = las.y - y_traj_i
z_dir = las.z - z_traj_i
norme_dir = np.sqrt(x_dir**2 + y_dir**2 + z_dir**2)

x_dir_n = x_dir/norme_dir
y_dir_n = y_dir/norme_dir
z_dir_n = z_dir/norme_dir
"""         *****           """

"""     interpolation de la trajectoire du scanner pour les tirs manquants     """
x_traj_missing_i = np.interp(missing_times, t_traj, x_traj)
y_traj_missing_i = np.interp(missing_times, t_traj, y_traj)
z_traj_missing_i = np.interp(missing_times, t_traj, z_traj)

# vecteurs directeurs interpolés (v.d.i)
x_dir_missing_i = np.interp(missing_times, t_las, x_dir_n)
y_dir_missing_i = np.interp(missing_times, t_las, y_dir_n)
z_dir_missing_i = np.interp(missing_times, t_las, z_dir_n)
norme_dir_missing = np.sqrt(x_dir_missing_i**2 + y_dir_missing_i**2 + z_dir_missing_i**2)

# vecteurs directeurs interpolés unitaires (v.d.u)
x_dir_n_missing = x_dir_missing_i / norme_dir_missing
```

```
    y_dir_n_missing = y_dir_missing_i / norme_dir_missing
    z_dir_n_missing = z_dir_missing_i / norme_dir_missing

    # projection des tirs manquants à 50m au-dessus du nuage
    x_missing = x_traj_missing_i + x_dir_n_missing * 50
    y_missing = y_traj_missing_i + y_dir_n_missing * 50
    z_missing = z_traj_missing_i + z_dir_n_missing * 50
    z_missing[z_missing < 0] = 0

    x_proj_all_ring.extend(x_missing)
    y_proj_all_ring.extend(y_missing)
    z_proj_all_ring.extend(z_missing)


""""   affichage du nuage de points reconstitué   """"
# nombre de points à sélectionner pour l'affichage
num_points = 100000
indices = np.random.choice(len(las.x), num_points, replace=False)
x = las.x[indices]
y = las.y[indices]
z = las.z[indices]

fig = plt.figure(figsize=(12, 12))
ax = fig.add_subplot(111, projection='3d')

# Affichage d'une partie du nuage
ax.scatter(x, y, z, c='blue', marker='.', s=1, label="Nuage de points origine")

# Affichage de tout le nuage
ax.scatter(las.x, las.y, las.z, c='blue', marker='o', s=0.1, label="Nuage de points origine")
```

```
# Affichage de tous les tirs manquants projetés des 32 faisceaux
ax.scatter(x_proj_all_ring, y_proj_all_ring, z_proj_all_ring, c='g', marker='.', s=1, label="Tous les tirs manquants projetés")

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('Nuage de points reconstitué')
ax.legend()

plt.show()
print('\n', "fin du programme!!")


"""          Vérification de la qualité de l'algorithme de reconstruction     """
import numpy as np
import laspy
import time
import random
import matplotlib.pyplot as plt

# Lecture du fichier du scanner et import de ces données
fichier_xyzt = 'P16_C14152_01_traj.xyz'
data = np.loadtxt(fichier_xyzt, delimiter=' ')
t_traj = data[:, 1]  # GPSTIME
x_traj = data[:, 2]
y_traj = data[:, 3]
z_traj = data[:, 4]

# Lecture du fichier du LiDAR et import de ces données
las = 'P16_C14152_01_subsampled_laz1_4.laz'
ring_number = 23
```

```
t_las = las.gps_time[las.Ring == ring_number]
# on élimine les doublons
t_las, u_index = np.unique(t_las, return_index= True)

# on élimine les points qui ont un même gpstime
x_las = las.x[las.Ring == ring_number]
x_las = x_las[u_index]
y_las = las.y[las.Ring == ring_number]
y_las = y_las[u_index]
z_las = las.z[las.Ring == ring_number]
z_las = z_las[u_index]

# sélection des 1000 premières valeurs de t_las
t_las = t_las[0:1000]
x_las = x_las[0:1000]
y_las = y_las[0:1000]
z_las = z_las[0:1000]

# possibilité de changer la valeur 100 (‘size’) pour l’affichage sur le graph
t_las_verif = np.sort(np.random.choice(t_las, 100, replace=False))

"""     Calcul des vd ‘normalement’    """
indices_verif = np.isin(t_las, t_las_verif)
x_verif = x_las[indices_verif]
y_verif = y_las[indices_verif]
z_verif = z_las[indices_verif]

x_traj_verif = np.interp(t_las_verif, t_traj, x_traj)
y_traj_verif = np.interp(t_las_verif, t_traj, y_traj)
z_traj_verif = np.interp(t_las_verif, t_traj, z_traj)
```

```
x_dir_verif = x_verif - x_traj_verif
y_dir_verif = y_verif - y_traj_verif
z_dir_verif = z_verif - z_traj_verif

norme_dir_verif = np.sqrt(x_dir_verif**2 + y_dir_verif**2 + z_dir_verif**2)
x_dir_verif_n = x_dir_verif/norme_dir_verif
y_dir_verif_n = y_dir_verif/norme_dir_verif
z_dir_verif_n = z_dir_verif/norme_dir_verif

# Matrice de vd_verif
vd_verif = np.vstack((x_dir_verif_n, y_dir_verif_n, z_dir_verif_n)).T

# Coordonnées et temps restants de t_las sans les 1000 valeurs de t_las_verif
x_remaining = x_las[~indices_verif]
y_remaining = y_las[~indices_verif]
z_remaining = z_las[~indices_verif]
t_las_remaining = t_las[~indices_verif]

# Vérification de la taille des tableaux
assert len(x_remaining) == len(y_remaining) == len(z_remaining) == len(t_las_remaining)

"""     Calcul des vd avec algorithme d'interpolation       """
# a) interpolation de la traj
x_traj_i = np.interp(t_las_remaining, t_traj, x_traj)
y_traj_i = np.interp(t_las_remaining, t_traj, y_traj)
z_traj_i = np.interp(t_las_remaining, t_traj, z_traj)

# vd
x_dir = x_remaining - x_traj_i
y_dir = y_remaining - y_traj_i
z_dir = z_remaining - z_traj_i
```

```
# vdu
norme_dir = np.sqrt(x_dir**2 + y_dir**2 + z_dir**2)
x_dir_n = x_dir/norme_dir
y_dir_n = y_dir/norme_dir
z_dir_n = z_dir/norme_dir

# vd_verif interpolés
missing_times = t_las_verif
x_dir_missing_i = np.interp(missing_times, t_las_remaining, x_dir_n)
y_dir_missing_i = np.interp(missing_times, t_las_remaining, y_dir_n)
z_dir_missing_i = np.interp(missing_times, t_las_remaining, z_dir_n)
norme_dir_missing = np.sqrt(x_dir_missing_i**2 + y_dir_missing_i**2 + z_dir_missing_i**2)

# v.d.u interpolés
x_dir_n_missing = x_dir_missing_i / norme_dir_missing
y_dir_n_missing = y_dir_missing_i / norme_dir_missing
z_dir_n_missing = z_dir_missing_i / norme_dir_missing

# Matrice des v.d.u.i
vd_verif_i = np.vstack((x_dir_n_missing, y_dir_n_missing, z_dir_n_missing)).T

"""     Visualisation de la qualité de l'interpolation       """
fig = plt.figure(figsize=(12, 12))
ax = fig.add_subplot(111)
# X
ax.scatter(t_las_remaining, x_dir_n, c= 'r', marker='.', label= 'x_dir_n')
ax.scatter(missing_times[1:len(missing_times)], x_dir_n_missing[1:len(missing_times)], c= 'g', marker='.', label= 'x_dir_n_missing')
ax.scatter(missing_times[1:len(missing_times)], x_dir_verif_n[1:len(missing_times)], c= 'b', marker='.', label= 'x_dir_verif_n')
# Y
ax.scatter(t_las_remaining, y_dir_n, c= 'r', marker='.', label= 'y_dir_n')
```

```
ax.scatter(missing_times[1:len(missing_times)], y_dir_n_missing[1:len(missing_times)], c= 'g', marker='.', label= 'y_dir_n_missing')
# Z
ax.scatter(t_las_remaining, z_dir_n, c= 'r', marker='.', label= 'z_dir_n')
ax.scatter(missing_times[1:len(missing_times)], z_dir_n_missing[1:len(missing_times)], c= 'g', marker='.', label= 'z_dir_n_missing')
# plot
ax.set_xlabel('t_las')
ax.set_ylabel('z_dir_n Ring23')
plt.show()
# exit()
"""       *****         """


"""     Vérification du degré d’angle entre vd_verif et vd_verif_i     """
def calculer_angle(v1, v2):
    cos_angle = np.dot(v1, v2) / (np.linalg.norm(v1) * np.linalg.norm(v2))
    angle = np.arccos(cos_angle)
    return np.degrees(angle)
angles = [calculer_angle(vd_verif[i], vd_verif_i[i]) for i in range(len(vd_verif_i))]
print("angles (en degrés) :\n", angles)
print(max(angles))
print(np.mean(angles))
print(min(angles))

"""     Vérification de la colinéarité entre vd_verif et vd_verif_i     """
def colinearity(v1, v2):
    return abs(1 - abs(np.dot(v1, v2)))
collinearite = [colinearity(vd_verif[i], vd_verif_i[i]) for i in range(len(vd_verif_i))]
print("colinearité :\n", collinearite)
print(max(collinearite))
print(np.mean(collinearite))
print(min(collinearite))
```

```
# garph colinéarité
x_range = [i for i in range(len(collinearite))]
plt.figure(figsize=(12, 12))
plt.plot(x_range, collinearite, color='r', label='collinearite')

# Ajout des valeurs statistiques
plt.axhline(max(collinearite), color='orange', linestyle='--', label='max_collinearite')
plt.axhline(min(collinearite), color='g', linestyle='--', label='min_collinearite')
plt.axhline(np.mean(collinearite), color='purple', linestyle='--', label='moy_collinearite')

# Plot
plt.xlabel("N° Trous")
plt.ylabel("No_colinearity")
plt.title("Degré de non-colinéarité en fonction des trous")
plt.legend()
plt.show()
print("Fin du programme !")
```

Annexe 2
Code de l’algorithme de Détection des faux tirs vides

i) Algorithme de suppression des points trop proches

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy.spatial import ConvexHull

# Fonction pour tracer une sphère autour d'une position donnée
def tracer_sphere(ax, centre, rayon):
    """
    Fonction pour tracer une sphère autour d'une position donnée dans une figure 3D.

    Arguments :
    ax -- l'objet Axes3D de matplotlib sur lequel la sphère sera tracée
    centre -- tuple (x, y, z) représentant la position du scanner
    rayon -- rayon de la sphère
    """
    phi, theta = np.mgrid[0:np.pi:100j, 0:2*np.pi:100j]
    x = centre[0] + rayon * np.sin(phi) * np.cos(theta)
    y = centre[1] + rayon * np.sin(phi) * np.sin(theta)
    z = centre[2] + rayon * np.cos(phi)
    ax.plot_surface(x, y, z, color='red', alpha=0.3, rstride=5, cstride=5)

# Fonction pour trouver les points à l'intérieur de la sphère
def trouver_points_interieur_sphere(points, centre, rayon):
    distances = np.sqrt((points[:, 0] - centre[0])**2 +
```

```
                    (points[:, 1] - centre[1])**2 +
                    (points[:, 2] - centre[2])**2)
    points_interieurs = points[distances <= rayon]
    points_restants = points[distances > rayon]
    return points_interieurs, points_restants

# Fonction pour tracer l'enveloppe convexe autour d'un ensemble de points
def tracer_enveloppe_convexe(ax, points, color='yellow', alpha=0.3):
    if len(points) >= 4:  # Nécessite au moins 4 points pour former un tétraèdre 3D
        hull = ConvexHull(points)
        for simplex in hull.simplices:
            ax.plot_trisurf(points[simplex, 0], points[simplex, 1], points[simplex, 2],
                            color=color, alpha=alpha)

# Génération du nuage de points 3D
n_points = 10000
length = 2
points = np.random.uniform(-length/2, length/2, (n_points, 3))

# Centre et rayon de la sphère
centre_scanner = (0, 0, 0)
#rayon_sphere = 1
rayon_sphere = 0.5

# Trouver les points à l'intérieur et à l'extérieur de la sphère
points_interieurs, points_restants = trouver_points_interieur_sphere(points, centre_scanner, rayon_sphere)

# Afficher le nombre de points avant suppression
print(f"Nombre de points avant suppression: {len(points)}")

# --- Première figure avant suppression des points ---
```

```
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(points[:, 0], points[:, 1], points[:, 2], s=1, color='green', alpha=0.6)
ax.scatter(centre_scanner[0], centre_scanner[1], centre_scanner[2], s=100, color='black')
# tracer_sphere(ax, centre_scanner, rayon_sphere)
ax.set_xlim(-1, 1)
ax.set_ylim(-1, 1)
ax.set_zlim(-1, 1)
ax.set_xlabel('X (m)')
ax.set_ylabel('Y (m)')
ax.set_zlabel('Z (m)')
#plt.show()

# --- Deuxième figure avec points en jaune et enveloppe convexe avant suppression ---
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(points[:, 0], points[:, 1], points[:, 2], s=1, color='green', alpha=0.6)
ax.scatter(points_interieurs[:, 0], points_interieurs[:, 1], points_interieurs[:, 2], s=1, color='yellow', alpha=0.6)
tracer_enveloppe_convexe(ax, points_interieurs, color='yellow', alpha=0.3)
ax.scatter(centre_scanner[0], centre_scanner[1], centre_scanner[2], s=100, color='black')
ax.set_xlim(-1, 1)
ax.set_ylim(-1, 1)
ax.set_zlim(-1, 1)
ax.set_xlabel('X (m)')
ax.set_ylabel('Y (m)')
ax.set_zlabel('Z (m)')
#plt.show()

# Afficher le nombre de points après suppression
print(f"Nombre de points après suppression: {len(points_restants)}")
```

```
# --- Troisième figure après suppression des points ---
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(points_restants[:, 0], points_restants[:, 1], points_restants[:, 2], s=1, color='green', alpha=0.6)
ax.scatter(centre_scanner[0], centre_scanner[1], centre_scanner[2], s=100, color='black')
# tracer_sphere(ax, centre_scanner, rayon_sphere)
ax.set_xlim(-1, 1)
ax.set_ylim(-1, 1)
ax.set_zlim(-1, 1)
ax.set_xlabel('X (m)')
ax.set_ylabel('Y (m)')
ax.set_zlabel('Z (m)')
plt.show()
```

ii) Algorithme de suppression des vrais tirs vides

```
import numpy as np
import laspy
import time
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

fichier_xyzt = ‘P16_C14152_01_traj.xyz'
data = np.loadtxt(fichier_xyzt, delimiter=' ')
t_traj = data[:, 1]      #GPSTIME
x_traj = data[:, 2]      #X
y_traj = data[:, 3]      #Y
z_traj = data[:, 4]      #Z
start_time = time.time()

las = ‘P16_C14152_01_subsampled_laz1_4.laz'
ring_number = 23
```

```
t_las = las.gps_time[las.Ring == ring_number]
t_las, u_index = np.unique(t_las, return_index= True) # élimine les doublons

x_las = las.x[las.Ring == ring_number]
x_las = x_las[u_index]                # on élimine les points qui ont un même temps gps
y_las = las.y[las.Ring == ring_number]
y_las = y_las[u_index]
z_las = las.z[las.Ring == ring_number]
z_las = z_las[u_index]

# t_las = t_las[0:10000]
# x_las = x_las[0:10000]
# y_las = y_las[0:10000]
# z_las = z_las[0:10000]

# Déterminer frequence d'echantillonnage ‘feo’ du gps_time
dt = np.diff(t_las)
dt = np.round(dt * 100000) / 100000
min_dt = np.min(dt)

# Recherche des dt sans trous
dt_sans_trou = dt[dt < 1.2 * min_dt]
dt_mean_sans_trou = np.mean(dt_sans_trou)

# Histogramme de dt
histo_dt = np.round(dt / dt_mean_sans_trou)
histo_dt = histo_dt.astype(int)
n_missing_t = histo_dt - 1

# Recherche des dt avec trous
dt_avec_trou = dt[dt >= 1.2 * min_dt]
```

```
        dt_avec_trou_indices = np.where(dt >= 1.2 * min_dt)[0]
        dt_mean_avec_trou = np.mean(dt_avec_trou)
        tps_avec_trou = t_las[dt_avec_trou_indices]
        l_avec_trou = list(tps_avec_trou)

        # Détermination des gps time des tirs manquants
        missing_times = []
        for i, n_missing in enumerate(n_missing_t):
            t0 = t_las[i]
            t1 = t_las[i+1]
            dt_loc = (t1-t0) / (n_missing + 1 )
            for j in range(1, int(n_missing) + 1):
                missing_times.append(t0 + j * dt_loc)

        # Vérification des missing times
        missing_times = sorted(set(missing_times))
        #missing_times = missing_times[0:1000]
        missing_times_dt = np.diff(missing_times)
        min_missing_times_dt = np.min(missing_times_dt)
        dt_mean_missing_times = np.mean(missing_times_dt[missing_times_dt < 1.2 * min_missing_times_dt])        # on ne prend en compte que
les dt correspondants au temps qu'on a recréé

        """     interpolated_traj  """
        x_traj_i = np.interp(t_las, t_traj, x_traj)
        y_traj_i = np.interp(t_las, t_traj, y_traj)
        z_traj_i = np.interp(t_las, t_traj, z_traj)
        #
        x_dir = x_las - x_traj_i
        y_dir = y_las - y_traj_i
        z_dir = z_las - z_traj_i
        norme_dir = np.sqrt(x_dir**2 + y_dir**2 + z_dir**2)
```

```
x_dir_n = x_dir/norme_dir
y_dir_n = y_dir/norme_dir
z_dir_n = z_dir/norme_dir
"""         *****          """
#
x_traj_missing_i = np.interp(missing_times, t_traj, x_traj)
y_traj_missing_i = np.interp(missing_times, t_traj, y_traj)
z_traj_missing_i = np.interp(missing_times, t_traj, z_traj)
# v.d interpolés
x_dir_missing_i = np.interp(missing_times, t_las, x_dir_n)
y_dir_missing_i = np.interp(missing_times, t_las, y_dir_n)
z_dir_missing_i = np.interp(missing_times, t_las, z_dir_n)
norme_dir_missing = np.sqrt(x_dir_missing_i**2 + y_dir_missing_i**2 + z_dir_missing_i**2)
# v.d.u
x_dir_n_missing = x_dir_missing_i / norme_dir_missing
y_dir_n_missing = y_dir_missing_i / norme_dir_missing
z_dir_n_missing = z_dir_missing_i / norme_dir_missing


"""     faux tir vides (ftv)    """
# supprimer tous les tirs qui heurtent l'opérateur pour cela :

# calculer pr tous les missing_times le vd de la trajectoire
epsilon = 0.1       # secondes
missing_times_eps = np.array(missing_times) + epsilon

x_traj_eps = np.interp(missing_times_eps, t_traj, x_traj)
y_traj_eps = np.interp(missing_times_eps, t_traj, y_traj)
z_traj_eps = np.interp(missing_times_eps, t_traj, z_traj)

x_traj_dir_missing_i = x_traj_missing_i - x_traj_eps
```

```
y_traj_dir_missing_i = y_traj_missing_i - y_traj_eps
z_traj_dir_missing_i = z_traj_missing_i - z_traj_eps
norme_dir = np.sqrt(x_traj_dir_missing_i**2 + y_traj_dir_missing_i**2 + z_traj_dir_missing_i**2)

x_traj_dir_missing_i_n = x_traj_dir_missing_i/norme_dir
y_traj_dir_missing_i_n = y_traj_dir_missing_i/norme_dir
z_traj_dir_missing_i_n = z_traj_dir_missing_i/norme_dir

# position du plan :
# calcul d’un point qui est la prolongation de la ligne du vd à 40 cm plus loin dans la direction de la trajectoire 'xp', et qui appartient au plan
x_p = x_traj_missing_i + 0.4 * x_traj_dir_missing_i_n
y_p = y_traj_missing_i + 0.4 * y_traj_dir_missing_i_n
z_p = z_traj_missing_i + 0.4 * z_traj_dir_missing_i_n

# Boucle sur tous les points de missing_times pour trouver les intersections avec le plan
x_intersect_list = []
y_intersect_list = []
z_intersect_list = []
ftv = []

x_filter_list = []
y_filter_list = []
z_filter_list = []
vtv = []

for i in range(len(missing_times)):
    # Composantes du vecteur normal au plan
    A = x_traj_dir_missing_i_n[i]
    B = y_traj_dir_missing_i_n[i]
    C = z_traj_dir_missing_i_n[i]
```

```
# Composantes de la trajectoire au point i
x_traj_curr = x_traj_missing_i[i]
y_traj_curr = y_traj_missing_i[i]
z_traj_curr = z_traj_missing_i[i]

# Calcul du paramètre t pour l'intersection avec le plan
num = A * (x_p[i] - x_traj_curr) + B * (y_p[i] - y_traj_curr) + C * (z_p[i] - z_traj_curr)
denom = A * x_dir_n_missing[i] + B * y_dir_n_missing[i] + C * z_dir_n_missing[i]

# Vérifier si le rayon intersecte le plan
if denom != 0:  # Éviter la division par zéro
    t = num / denom

    # Calcul des points d'intersection
    x_intersect = x_traj_curr + t * x_dir_n_missing[i]
    y_intersect = y_traj_curr + t * y_dir_n_missing[i]
    z_intersect = z_traj_curr + t * z_dir_n_missing[i]

    # Calcul de la distance entre le point d'intersection et le point p
    distance = np.sqrt((x_intersect - x_p[i]) ** 2 + (y_intersect - y_p[i]) ** 2 + (z_intersect - z_p[i]) ** 2)

    # Filtrage des points selon la distance
    if distance > 0.4:  # Si distance > 40 cm
        # x_intersect_list.append(x_intersect)
        # y_intersect_list.append(y_intersect)
        # z_intersect_list.append(z_intersect)
        ftv.append((x_intersect, y_intersect, z_intersect))
    else:
        # x_filter_list.append(x_intersect)
        # y_filter_list.append(y_intersect)
        # z_filter_list.append(z_intersect)
```

```
            vtv.append((x_intersect, y_intersect, z_intersect))
# print(f"Nombre de points filtrés (distance > 40 cm) : {len(ftv)}")
# print(f"Nombre de points filtrés (distance < 40 cm) : {len(vtv)}")

# Trouver les index des points à supprimer
vtv_array = np.array(vtv)
to_remove_indices = []
for point in vtv_array:
    # Comparer chaque point VTV avec les points du nuage original
    idx = np.where((x_las == point[0]) & (y_las == point[1]) & (z_las == point[2]))[0]
    to_remove_indices.extend(idx)

# Supprimer les vtv du nuage de points d'origine
x_las_filtered = np.delete(x_las, to_remove_indices)
y_las_filtered = np.delete(y_las, to_remove_indices)
z_las_filtered = np.delete(z_las, to_remove_indices)
print(f"Nombre de points après suppression des VTV: {len(x_las_filtered)}")

# affichage
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')

#ax.scatter(x_las, y_las, z_las, c='blue', marker='.', s=10, label="Nuage de points origine")
ax.scatter(x_las_filtered, y_las_filtered, z_las_filtered, c='b', marker='.', s=10,label="Nuage de points apres suppression des vtv")

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('ndp sans vrais tirs vides')

plt.legend()
```

```
plt.show()
```

Liste des figures[1]

RESUME

Dans le cadre des changements climatiques, l'institut AMAP utilise la technologie LiDAR mobile (MLS) pour analyser le couvert végétal et déterminer l'indice de surface foliaire (LAI), essentiel pour évaluer le potentiel photosynthétique. La précision de cette estimation repose sur la détection des tirs vides, qui révèlent l'absence de végétation. Le stage a pour objectif de développer un algorithme capable de reconstruire la géométrie des tirs manquants en générant des pseudo-échos infinis dans le nuage de points. Deux cas théoriques ont été étudiés, l'un avec un scanner statique, l'autre avec un scanner dynamique. L'application de cet algorithme sur des données réelles de la forêt amazonienne a validé son efficacité. Un second algorithme distingue les faux tirs vides des vrais, garantissant ainsi des résultats optimaux. Le tout a été développé en Python avec des indicateurs pour vérifier la qualité de la reconstitution.

Mots-clés : LiDAR, MLS, LAI, tirs vides, algorithme, interpolation, couvert végétal.

SUMMARY

In the context of climate change, the AMAP institute uses mobile LiDAR technology (MLS) to analyze vegetation cover and determine the Leaf Area Index (LAI), essential for evaluating photosynthetic potential. The accuracy of this estimation depends on detecting empty laser shots, which indicate the absence of vegetation. The internship aims to develop an algorithm capable of reconstructing the geometry of missing shots by generating infinite pseudo-echoes within the point cloud. Two theoretical cases were studied: one with a static scanner and the other with a dynamic scanner. The application of this algorithm on real data from the Amazon rainforest confirmed its effectiveness. A second algorithm was developed to distinguish false empty shots from real ones, ensuring optimal results. The algorithms were developed in Python, with specific indicators to verify the quality of the reconstruction.

Keywords: LiDAR, MLS, LAI, empty shot, algorithm, interpolation, vegetation cover.