Détection des objets dans un environnement urbain à partir de données Lidar

Gwénaël Quéré

le cnam
École supérieure des géomètres et topographes esgt

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

ECOLE SUPERIEURE DES GEOMETRES ET TOPOGRAPHES

MEMOIRE

présenté en vue d'obtenir

le DIPLOME D'INGENIEUR CNAM

SPECIALITE : Géomètre et Topographe

par

Gwénaël QUÉRÉ

Détection des objets dans un environnement urbain à partir de données Lidar

Soutenu le 06 Septembre 2019

JURY

Monsieur José CALI	Président du jury
Monsieur Cyril MICHON	Maître de stage
Monsieur Jérôme VERDUN	Enseignant référent

Remerciements

Je tiens tout d'abord à remercier mon maître de stage, Cyril Michon pour son investissement, ses conseils et son encadrement durant ce travail de fin d'études. Je remercie également les équipes des services Topographie et Lidar de Geofit qui m'ont accompagné durant ces quelques mois et qui ont pris du temps pour répondre à mes questions. Je remercie tout particulièrement les chefs de projet qui ont pu m'apporter une aide technique précieuse durant ce stage.

Je remercie également M. Verdun, mon professeur référent, pour son accompagnement et pour ses conseils avisés durant ce travail de fin d'études.

Enfin, je remercie ma famille qui m'a apporté son soutien tout au long de mes études et qui a toujours été derrière moi pendant ces années.

Liste des abréviations

LIDAR : acronyme de “LIght Detection And Ranging” ou “Laser Detection And Ranging“, il s’agit d’une technique d’acquisition de données basée sur l’analyse du faisceau lumineux renvoyé par l’objet.

MMS : acronyme de “Mobile Mapping System”, il s’agit d’un processus pour acquérir des données à partir d’un véhicule mobile généralement équipé de capteurs tels que des appareils photos, des radars, des lasers, des Lidar.

CAO : acronyme de “Conception Assistée par Ordinateur”. Terme regroupant l’ensemble des logiciels et des techniques permettant de concevoir, réaliser des produits à l’aide d’un ordinateur.

GNSS : acronyme de “Global Navigation Satellite System”, il s’agit d’un ensemble de composants reposant sur une constellation de satellites permettant de fournir la position 3D d’un récepteur.

IMU : acronyme de “Inertial Measurement Unit” ou unité de mesure inertielle. Il s’agit d’un appareil de mesure permettant d’estimer les mouvements et l’orientation d’un mobile.

MLS : acronyme de “Mobile Lidar System”, équivalent au MMS.

RANSAC : acronyme de “RAndom SAmple Consensus”. Il s’agit d’une méthode itérative pour estimer les paramètres de certains modèles mathématiques comme les lignes, les plans...

SVM : acronyme de “Support Vector Machine”. Il s’agit d’une technique d’apprentissage supervisé destiné à résoudre des problèmes de discrimination et de régression.

Glossaire

Cluster : Terme anglais correspondant à un groupe de points issus d'une méthode de regroupement. Dans le cas des nuages de points, on utilise ce terme pour regrouper les points en plusieurs groupes selon certains critères.

Deep Learning : Terme anglais pour l'apprentissage profond. C'est un type d'intelligence artificielle dérivé du machine learning où la machine est capable d'apprendre par elle-même, contrairement à la programmation où elle se contente d'exécuter les instructions à la lettre.

Loopy Belief Propagation (LBP) : La propagation des convictions, aussi connu comme la transmission de message somme-produit est un algorithme à passage de message pour effectuer des inhérences sur des modèles graphiques. C'est un algorithme couramment utilisé en intelligence artificielle.

Machine learning : Terme anglais signifiant l'apprentissage automatique. C'est un champ d'étude de l'intelligence artificielle qui se fonde sur des approches statistiques pour donner aux ordinateurs la capacité "d'apprendre" à partir de données.

Markov Random Field (MRF) : Un champ aléatoire de Markov est un ensemble de variables aléatoires vérifiant une propriété de Markov relativement à un graphe non orienté. Ils sont utilisés, entre autres, pour la classification en fouille de données spatiales, en analyse d'images…

Nuage de points : Ce terme désigne un ensemble discret de points. Dans cette étude ces points sont obtenus par relevé Lidar. Chaque point est défini au minimum par ses coordonnées X, Y, Z. D'autres attributs peuvent s'ajouter à ces points comme l'intensité, le nombre de retour, la direction du scan, une classe…

Viewer : Terme anglais pour "Visionneuse". Il s'agit d'un logiciel qui permet d'afficher, de lire des objets informatiques que ce soit des images, des nuages de points ou d'autres objets selon son utilité.

Table des matières

Introduction

A l'heure où les données géoréférencées ont une place cruciale dans les prises de décisions aussi bien privées que publiques, la quantité et la qualité de ces données est une problématique très importante, notamment pour ceux qui créent cette donnée. L'arrivée des systèmes d'acquisitions tridimensionnelles bouleverse petit à petit l'acquisition et la façon de traiter les données. En effet avec l'arrivée de la technologie Lidar et du Mobile Mapping System (MMS) dont le rapport quantité de données par rapport à la vitesse d'acquisition est inégalé, l'approche des données n'est plus la même. L'avantage des systèmes d'acquisitions tridimensionnelles est de pouvoir obtenir une donnée exhaustive, et de choisir la donnée adéquate au livrable en post-traitement. En effet, à partir d'un même produit qu'est le nuage de points il est possible de produire plusieurs livrables.

Concernant l'acquisition des données 3D par technologie Lidar, la littérature s'accorde sur le fait que de nos jours l'acquisition sur le terrain est globalement maîtrisée. Cependant concernant la phase de traitement, l'adjectif chronophage revient régulièrement dans les articles. En effet si l'acquisition est rapide, de nombreux facteurs comme la quantité importante de données, le bruit dans les données, les zones d'ombre ou la consolidation parfois complexe rendent le traitement long et fastidieux. Depuis l'apparition de ces technologies, de nombreuses équipes de scientifiques travaillent sur le développement de méthodes automatiques et fiables afin d'améliorer le traitement de ces données. On peut notamment citer des recherches sur l'extraction automatique du sol, la détection de points isolés ou de points bas, la reconnaissance de formes géométriques ou d'objets urbains…

Ce mémoire a pour objectif d'identifier des méthodes de traitement et des processus potentiellement applicables dans un cabinet de géomètres-experts ou de topographie afin d'améliorer le traitement de nuages de points, de réduire le temps de ce traitement et d'en automatiser une partie. On s'intéressera principalement à la détection d'objets dans les nuages de points acquis par MMS dans divers environnements. Ces acquisitions de nuages de points sont réalisées en majeure partie afin de produire des plans sur des emprises importantes, que ce soit à l'échelle d'une ville, d'un quartier, sur une portion d'autoroute ou de ligne SNCF. Ce travail de recherche a été réalisé au sein du cabinet Geofit Expert. Il s'agit d'une société de services en géomatique et droits des sols comptant plusieurs domaines d'activité divers et variés. L'entreprise compte plus de 900 collaborateurs répartis sur 13 établissements (sept en

France et six à l'étranger). L'entreprise dispose depuis plusieurs années d'un pôle MMS ainsi qu'un pôle Topographie/Lidar, ce qui lui a permis d'être une des premières entreprises en France à développer les applications relatives à l'acquisition par MMS. L'entreprise est donc dans une démarche constante d'évolution et d'amélioration de leurs procédés afin de suivre au mieux les technologies existantes et améliorer leur rendement. Cette démarche s'inscrit donc dans une démarche de cycle PDCA (Plan, Do, Check, Act) schématisé en Figure 1.

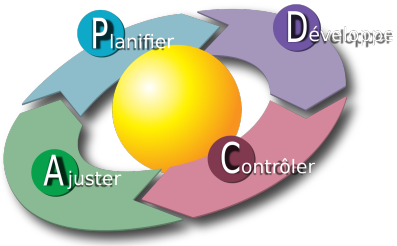


Figure 1 : Représentation du cycle PDCA

La réforme anti-endommagement des réseaux incite depuis 2012 les collectivités territoriales à s'équiper d'un Plan de Corps de Rue Simplifié (PCRS) [1]. Ceux-ci sont un très bon exemple d'application pour les levés mobiles. Auparavant ce type de plan de corps de rue était réalisé à l'aide d'instruments de topographie classique comme le tachéomètre équipé d'une visée laser. Avec l'arrivée des systèmes d'acquisition 3D, le choix de la donnée et la manière de la traiter par la suite est totalement différente. Si auparavant la donnée utile était sélectionnée sur le terrain au moment de l'acquisition, ce n'est plus le cas avec les nouveaux systèmes. Il y a donc de nouvelles méthodes à trouver afin d'optimiser ces traitements et faciliter voire automatiser certains procédés.

Pour un cabinet de géomètre, automatiser certains traitements par le biais d'algorithmes peut s'avérer très utile. L'utilisation d'algorithmes est un gain de temps non négligeable sur des dossiers aux objectifs similaires. On peut également s'attendre à une homogénéité dans la précision des résultats entre différents dossiers car le traitement sera réalisé de façon mathématique, donc similaire d'un dossier à l'autre, tandis que les résultats des opérateurs peuvent légèrement varier d'un dossier à un autre avec notamment des erreurs de pointés susceptibles d'être produites.

Ce travail de fin d'études est structuré en trois parties. La première partie présentera le contexte de l'étude de ce mémoire, en décrivant d'une part les méthodes employées à ce jour par l'entreprise pour l'acquisition des données et le traitement de celles-ci, et d'autre part un résumé de ce que les logiciels proposent en matière de segmentation, classification et vectorisation d'objets dans un nuage de points. Dans une seconde partie, les méthodes utilisées pour réaliser l'extraction d'objets dans un nuage de points issu de différents environnements seront détaillées ainsi que les recherches scientifiques qui y sont associées. Enfin nous mettrons en application ces méthodes afin d'analyser les résultats.

I Contexte de l'étude

I.1 Problématique et objectifs

L'entreprise Geofit Expert est consciente depuis plusieurs années des enjeux et des opportunités apportées par les technologies Lidar, qu'elles soient statiques ou mobiles, et a su intégrer ces outils récents pour élargir leur éventail de compétences. Le choix d'utiliser ces technologies de façon quasi systématique montre bien l'avantage sur le terrain de ces outils. Cependant il ressort de la phase de traitement de nombreuses problématiques, aussi bien en ce qui concerne la consolidation des nuages, le temps de traitement, le volume très important des données ou encore l'utilisation judicieuse de celles-ci. La problématique des logiciels à utiliser est également complexe. En effet les logiciels de CAO capables de gérer les nuages de points sont relativement récents, ou ont intégré récemment des modules supplémentaires pour la gestion de ceux-ci. Ces logiciels intègrent donc petit à petit des fonctions permettant l'automatisation de certaines tâches comme la segmentation et la classification. Mais la question est également de savoir si ces fonctions sont efficaces et adaptées aux besoins de l'entreprise. D'autant plus que l'utilisation de ce genre de logiciels peut s'avérer onéreuse dans une production importante, et l'entreprise préfère s'assurer de la viabilité de cette utilisation.

Selon le type de livrable demandé, l'échelle de représentation et l'environnement dans lequel le levé a été réalisé, de nombreux éléments peuvent apparaître sur un plan. S'il s'agit d'une zone urbaine dense, le nombre d'objets comme les panneaux de signalisation, le mobilier urbain, les éléments de réseaux sont bien plus présents que dans un environnement rural. Les mêmes types d'objets peuvent être également très différents d'un levé à un autre et d'un environnement à un autre puisqu'on ne cherchera pas à détecter les mêmes objets dans un environnement urbain qu'à proximité d'une voie de chemin de fer.

L'objectif de ce travail de détection d'objets sera dans un premier temps de tester et de comprendre les algorithmes et fonctions proposés par les logiciels capables de traiter les nuages de points. Une analyse sur l'efficacité et la pertinence de ces fonctions sera développée. Étant donné que ces méthodes montrent quelques faiblesses, des algorithmes plus efficaces seront donc présentés et analysés. Enfin nous verrons si ces algorithmes peuvent être utiles lors de la digitalisation et la création de plan.

La multiplication des logiciels et des applications nous force également à nous poser la question des formats de fichiers utilisés. En effet, de plus en plus de types de formats sont disponibles et entre les formats libres et les formats propriétaires il y a une multitude de possibilités sachant que chaque format offre des fonctionnalités différentes. Cependant certains formats peuvent être plus adaptés que d'autres à certaines utilisations. C'est pourquoi certains formats seront privilégiés lors des traitements.

Ce mémoire a pour objectif de réaliser une analyse systématique des méthodes de détection automatique d'objets dans différents environnements urbains issus de nuages de points acquis par Lidar mobile. Une conclusion sur la pertinence de cette façon de traiter ces données sera présentée et les perspectives d'évolution de cette approche seront également envisagées.

I.2 État de l'art

L'état de l'art ci-dessous a pour but de définir les méthodes utilisées aujourd'hui en partant de l'acquisition des données jusqu'au livrable. Il sera donc question du matériel utilisé pour le MMS, des traitements du nuage de points brut, des logiciels utilisés, des segmentations et classifications réalisables et enfin de la vectorisation.

I.2.1 Acquisition et traitement des données

I.2.1.1 Les systèmes de cartographie mobile

Les systèmes de cartographie mobile sont communément composés de deux parties. La première étant un système de positionnement et d'orientation permettant le géoréférencement et la seconde étant un ou plusieurs capteurs d'acquisition.

- Le système de position et d'orientation

  Pour connaître la position et l'orientation d'un véhicule en tout point de sa trajectoire, on utilise un système composé d'un récepteur GNSS, d'une centrale inertielle (IMU) et d'un odomètre.

  Pour déterminer le positionnement en cartographie mobile on utilise le mode cinématique différentiel en temps différé. Si les conditions d'acquisitions sont bonnes, la précision de géoréférencement est de l'ordre de quelques centimètres mais celle-ci

peut rapidement être dégradée si la restitution des coordonnées par le récepteur GNSS est altérée à cause des masques ou une mauvaise résolution des ambiguïtés entières...

Pour déterminer l'orientation du véhicule, la centrale inertielle (Inertial Measurement Unit, IMU) est constituée de trois gyromètres, et de trois accéléromètres. Ces capteurs sont associés à un calculateur réalisant l'intégration en temps réel des angles d'attitude (roulis, tangage, lacet), de la vitesse et de la position. La centrale inertielle devient un système de navigation inertiel (Inertial Navigation System, INS).

Le système de position et d'orientation est un élément très important de la cartographie mobile car la précision des points 3D acquis est grandement dépendante de la précision du géoréférencement.

- Les capteurs d'acquisition
  - Les scanners

Les scanners laser permettent de mesurer la distance qui les sépare d'une surface en utilisant la lumière comme source d'information. L'avantage de cette technologie est l'obtention rapide et automatique de points en 3D avec une densité très élevée contrairement à un tachéomètre comme montré en Figure 2.

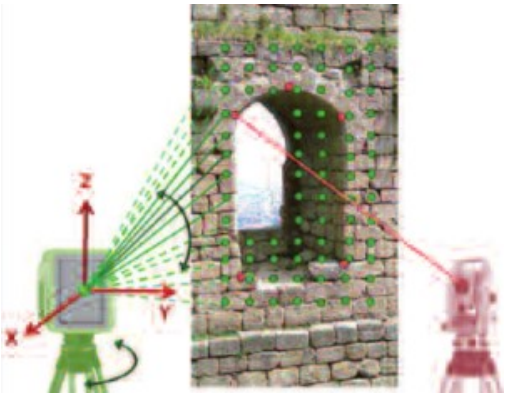


Figure 2 : Relevé d'une ouverture à partir d'un scanner laser terrestre (à gauche) et au tachéomètre sans réflecteur (à droite) (Landes et Grussenmeyer, 2011) [2]

Il existe principalement trois catégories de scanners laser (Boulaassal. H, 2010) [3]:

- Les scanners à temps de vol : Le scanner laser émet un faisceau laser pulsé et la distance est déterminée par la mesure du temps que met le train d'impulsions pour faire l'aller-retour.
- Les scanners laser à différence de phase : Le scanner laser émet un faisceau laser continu modulé et la distance à l'objet est déterminée par la différence de

phase entre la modulation de lumière reçue et envoyée. Cette méthode est plus adaptée pour les objets proches.

- Les scanners laser à triangulation : La distance entre le scanner laser et l'objet est calculée à partir d'une résolution de triangle, méthode adaptée pour numériser des objets à une courte distance.

- Les caméras

De manière générale, le scanner laser est associé à plusieurs caméras. Cela permet la prise continue d'images dans toutes les directions et de façon panoramique.
Ces images servent principalement à coloriser le nuage de points acquis par le laser scanner. Elles peuvent également servir à appliquer une texture si un maillage est réalisé à partir du nuage de points. Dans le cas de systèmes basés sur la photogrammétrie, des mesures de stéréovision sont aussi possibles.

Les caméras utilisées sont généralement des caméras numériques comprenant des résolutions relativement faibles et un temps d'exposition faible permettant d'optimiser le nombre d'images.

I.2.1.2 Les systèmes existants et leurs précisions

Le premier système de cartographie mobile opérationnel, appelé GPSVan, a été développé en 1991 par le centre de cartographie à l'université d'état de l'Ohio [Ellum et El-Sheimy, 2002]. Depuis, la demande a fortement augmenté concernant la modélisation urbaine 3D et les systèmes se sont développés petit à petit.

De nombreuses entreprises proposent des systèmes de MMS, notamment les sociétés 3DLM avec les StreetMapper, Leica avec les Pegasus, Optech avec les Lynx et les Maverick et Riegl avec leurs systèmes VMX. La société Geofit Expert utilise principalement la dernière version du système développé par 3DLM, le StreetMapper IV.

Ces systèmes s'étant rapidement développés et se démocratisant de plus en plus, une étude nommée "EuroSDR Mobile Mapping-Road Environment Mapping using Mobile Laser scanning" a été menée en 2013 afin de comparer les précisions des différents systèmes lors d'une acquisition avec de bonnes conditions GNSS (Kaartinen et al, 2013) [5].

Les nuages de points acquis par cinq systèmes (RIEGL VMX-250, le Streetmapper 360, l'Optech Lynx, le FGI Roamer et le FGI Sensei) ont été comparés à un nuage de points (acquis par scanner statique) qui a auparavant été géoréférencé à l'aide d'un réseau géodésique. Un test sur la qualité des données en altimétrie et un autre sur la qualité des données en planimétrie a ensuite été effectué.

Le premier test sur l'altimétrie consiste en une comparaison d'une grille d'environ 3000 points extraite des nuages de points en comparaison avec le nuage de référence. En utilisant l'outil de TerraScan "Output control report", les résultats sont donnés sur la Figure 3.

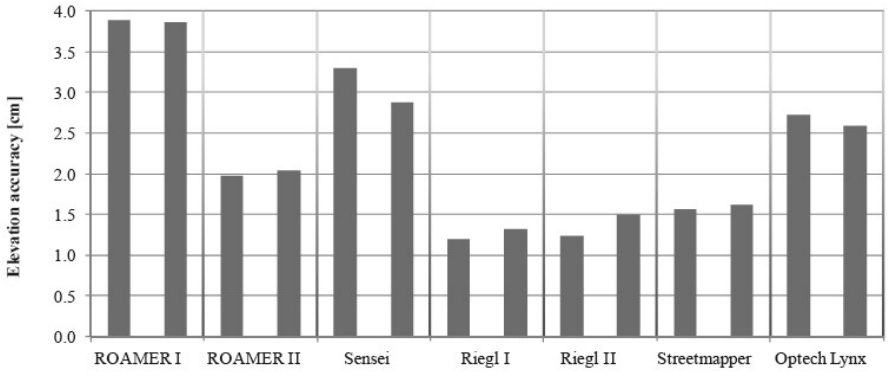


Figure 3 : Valeurs des précisions altimétriques (écart-types en centimètres) pour les systèmes MLS testés. La colonne de gauche correspond à un sens de marche antihoraire (CCW) et celle de droite à un sens horaire (CW)

On peut remarquer avec ces résultats que les meilleurs systèmes peuvent atteindre une précision comprise entre un et deux centimètres, pour des distances allant jusqu'à 35 mètres.

Le second test concernant les précisions planimétriques s'effectue sur 273 points situés près du sol (entre 10 cm en dessous et 50 cm au-dessus du sol) définis en planimétrie. Une comparaison avec les nuages de points à tester est réalisée et l'on obtient les résultats présentés sur la Figure 4 ci-après.

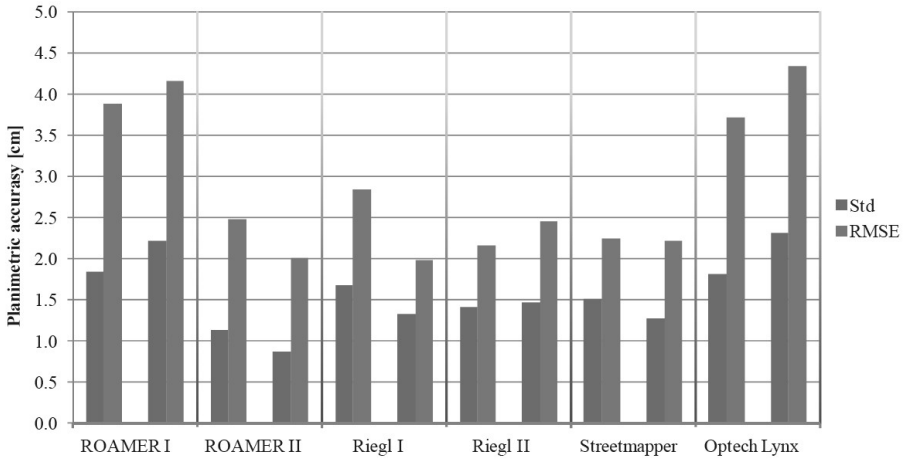


Figure 4 : Précision planimétrique des systèmes MLS testés dans deux directions. La colonne de gauche correspond à un sens de marche antihoraire (CCW) et celle de droite à un sens horaire (CW)

On peut voir que les précisions planimétriques sont du même ordre de grandeur que les précisions altimétriques. Bien que les technologies aient légèrement évolué depuis l'étude, l'ordre de grandeur des précisions est le même pour les systèmes actuels. Il est important de connaître les précisions des points 3D mesurés puisque cela va affecter directement les traitements qui vont suivre et peuvent jouer sur la détection des objets.

La grande différence entre un nuage de points acquis par scanner mobile comparé à un nuage acquis par scanner fixe réside dans le bruit et la densité variable des points compte tenu du mouvement et de la multiplicité des passages.

I.2.1.3 Le prétraitement des données

Les nuages de points acquis par MMS nécessitent plusieurs opérations avant d'être exploitables pour le dessin des plans (Figure 5). En premier lieu un calcul des trajectoires est nécessaire à l'aide des coordonnées GNSS mesurées et des données de la centrale inertielle afin de réaliser le géoréférencement. Il faut donc vérifier la cohérence des mesures GNSS et de la centrale inertielle. En milieu urbain, la réception GNSS peut s'avérer très mauvaise (masques, tunnels…) et la centrale peut rapidement dériver. Il est donc primordial de vérifier le géoréférencement, sans quoi la précision peut se dégrader.

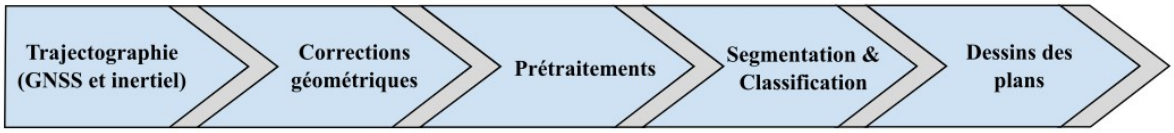


Figure 5 : Chaîne de traitement simplifiée des nuages de points au sein de Geofit Expert

Une fois le géoréférencement effectué, il faut réaliser la consolidation des nuages ("matching"). Si plusieurs nuages de points sont présents sur les mêmes zones, on va ajuster la position des nuages pour qu'ils coïncident entre eux. Effectivement même si le géoréférencement est correct, au vu de la précision GNSS et de la centrale inertielle, certains nuages peuvent être décalés l'un par rapport à l'autre de plusieurs centimètres. Cette consolidation s'effectue généralement avec des éléments (points de contrôle, marquages au sol, mobilier urbain…) communs aux deux nuages, appelés "Tie lines" sous Terrascan. Ces "Tie lines" peuvent être représentées sous différentes formes, que ce soit un point connu en coordonnées cartésiennes géocentriques (point GNSS par exemple), un point au sol, une ligne présente dans une section verticale ou horizontale (comme le mur d'un bâtiment ou le sol)...

Ces nombreuses formes permettent de s'adapter selon les données acquises et les éléments présents dans le nuage.

Après ces étapes, le nuage est exploitable pour réaliser les plans. Cependant on peut voir la limite de précision du Lidar embarqué sur de nombreux nuages avec un bruit de quelques centimètres qui peut apparaître. Ce bruit sera à prendre en compte pour la suite de nos recherches.

Ce rapport va se concentrer sur les dernières étapes de traitement des nuages de points dans l'entreprise. L'objectif de ce travail est de faciliter la vectorisation des éléments apparaissant sur les différents plans issus de nuages de points.

I.2.2 Les logiciels utilisés

Afin de traiter les nuages de points, l'entreprise Geofit Expert utilise plusieurs logiciels différents.

- Microstation : Il s'agit d'un logiciel propriétaire d'ingénierie collaborative édité par la société Bentley Systems. Il est utilisé dans de nombreux domaines et peut être assimilé à un logiciel de CAO. Il possède plusieurs outils pour le dessin 2D et 3D avec notamment les primitives de dessins (lignes, arcs, cercles, splines…), les opérations géométriques (projection, rotation…), les visualisations 2D et 3D. Il comprend également des outils pour développer des macros ou des modules (VBA ou MDL). Ces outils permettant à Bentley Systems ou à des sociétés tierces de développer des plugins particuliers pour le logiciel, à l'instar de la suite Terrasolid.

- La suite Terrasolid : Cette suite de modules est spécialisée dans le traitement et l'utilisation de données Lidar et photos. Ce sont des modules compatibles avec Microstation.
    - Terrascan : Ce module est dédié au traitement de données brutes acquises par Lidar aéroporté ou terrestre. Il permet de lire de nombreux formats de fichiers (.las, .fbi, .xyz…) et d'afficher jusqu'à un milliard de points. Il propose de nombreuses fonctions avec notamment la visualisation des nuages de points selon différents attributs (intensité, élévation, par classe…) et des outils de segmentation et de classification automatiques et manuels.

- TerraMatch : Ce plugin comprend plusieurs fonctions destinées à corriger les données et améliorer la qualité des données brutes des nuages. Il permet notamment de réaliser la consolidation de nuages de points à l'aide de ses outils "Tie lines". Il permet également la gestion des trajectoires et la correction des angles d'attitude (roulis, tangage, lacet) de la centrale inertielle.
- TerraModeler : La création, visualisation et modification des modèles de surface sont gérées par ce module. En plus de la création de MNT, il permet de gérer des calculs de profils, de volumes, ou de réaliser différentes opérations sur ceux-ci.

- Topodot : Il s'agit d'une application de CAO développée par Certainty3D pour l'extraction d'éléments, de la topographie ou de modèles 3D issus de données Lidar. De la même façon que Terrascan il s'intègre à l'environnement de Microstation. Cette application est très efficace pour vectoriser de façon semi-automatique des éléments spécifiques d'un nuage de points.

- CloudCompare : Il s'agit d'un logiciel libre et gratuit qui comme son nom l'indique a été créé à l'origine pour comparer deux nuages de points. Il s'est ensuite développé et des plugins se développent progressivement sur celui-ci ce qui en fait un logiciel de traitement de nuage de points très utilisé aujourd'hui.

I.2.3 Les méthodes de segmentation et de classification

Lorsqu'il s'agit de traitement et d'analyse de nuages de points, la segmentation est une étape incontournable. Landes et Grussenmeyer, 2011 [2] donnent une définition précise de ce processus : "La segmentation d'un nuage de points est un partage/subdivision de l'ensemble des points 3D en sous-ensembles (sous-nuage de points) homogènes, suivant des critères prédéfinis". Cette étape consiste donc à séparer le nuage en différents éléments afin de les traiter individuellement par la suite. Les critères prédéfinis cités précédemment sont des critères d'homogénéité dépendant de l'utilisateur, il peut aussi bien s'agir de l'intensité du laser ou bien la forme (planéité, courbure, linéarité…) que prend le voisinage d'un point (Boulaassal. H, 2010) [3].

Il existe de nombreuses techniques de segmentation de données 3D, pour la plupart inspirées de méthodes de segmentation d'images (2D). Cependant, là où les algorithmes de

segmentation d'images s'appuient sur la radiométrie de l'image, les algorithmes concernant les nuages de points s'appuient sur la position spatiale des points.

- Les outils de Terrascan :

Le logiciel Terrascan propose deux fonctions afin de classifier le sol ("Hard Surface" et "Ground"). Étant donné que l'on définit un sous-ensemble du nuage de points il s'agit en réalité de fonctions de segmentation. On s'intéresse ici à la fonction "Hard surface" plus adaptée pour les levés terrestres. Celle-ci va créer de façon itérative un modèle de surface et créer un sous-ensemble avec les points qui coïncident avec la surface médiane dominante du nuage. Cette fonction est très efficace pour classer la majorité des points du sol et ainsi réaliser des calculs de distance/hauteur par rapport à celui-ci. Cette fonction ressemble fortement à l'algorithme de RANSAC (Marchand M. 2018) [4] avec des contraintes qui peuvent être imposées comme la tolérance sur le plan médian, la pente maximale, et le pas à prendre en compte ("Minimum detail" et "Minimum triangle").

Le logiciel Terrascan propose également une fonction "Assign Groups" afin de créer des "Groups" qui correspondent aux sous-ensembles cités précédemment. Il est possible de créer ces groupes selon trois algorithmes différents.

- "Group planar surfaces" : Cet algorithme permet de définir un sous-ensemble pour tous les points contenus dans un même plan. Il est possible de fixer une tolérance de distance par rapport au plan, une aire minimale pour le sous-ensemble ainsi qu'une hauteur minimum par rapport au sol. Cet algorithme est similaire à l'algorithme RANSAC. Cette fonction est très efficace pour créer des sous-ensembles de bâtiments, de murs et autres éléments plans lorsqu'ils sont isolés. Il est en revanche très difficile de créer des sous-ensembles cohérents lorsque de la végétation se trouve à proximité de l'élément plan par exemple (voir exemple sur la Figure 6).

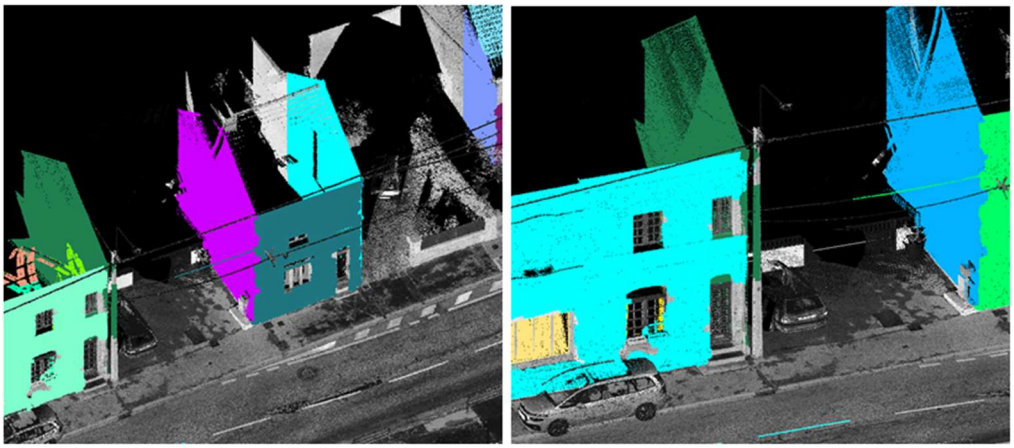


Figure 6 : Exemple de bonne segmentation des différents plans du bâtiment (à gauche) et de mauvaise segmentation (à droite) avec la prise en compte du poteau dans le plan, issues de l'algorithme "Group planar surfaces" de Terrascan

- “Group by tree logic” : Cet algorithme est basé sur le “watershed algorithm” (Ligne de partage des eaux) qui est un algorithme issu de la segmentation d’image à l’origine. Il consiste à considérer une image à niveau de gris comme un relief topographique et de définir les lignes de partage des eaux de ce relief à l’aide du gradient pour définir les sous-ensembles (S.Beucher, 1991) [6]. Comme son nom l’indique cet algorithme est intéressant pour segmenter les arbres et la végétation. Cependant il arrive très régulièrement que celui-ci ne s’arrête pas en présence d’un autre objet contigu (voir exemple sur la Figure 7).

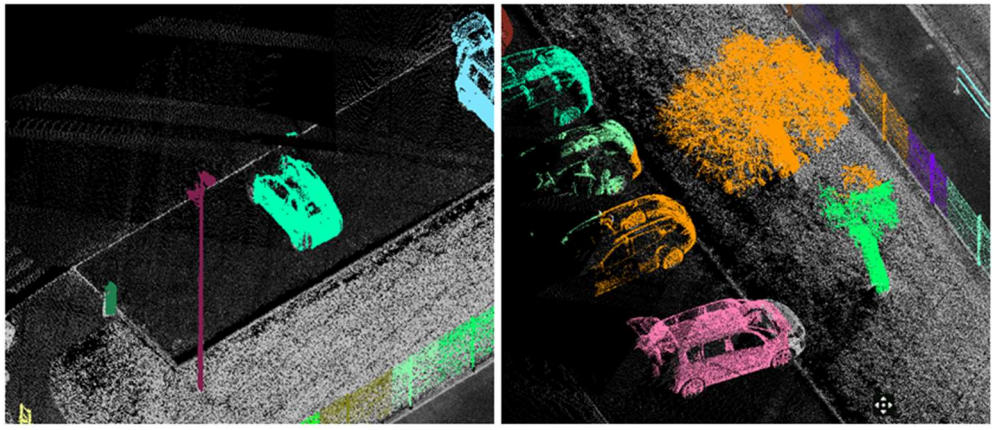


Figure 7 : Exemple de bonne segmentation (à gauche) avec des sous-ensembles définis par objet et de mauvaise segmentation (à droite) avec des sous-ensembles définis sur plusieurs objets, issues de l'algorithme "Group by tree logic" de Terrascan

- “Group by density” : Ce dernier algorithme de segmentation inclus dans la fonction “Assign Groups” va regrouper les points à la manière d’un kNN Tree (L.Di Angelo, 2011) [7]. Ainsi, à chaque point, les points voisins vont être sondés et si la distance entre le point origine et le point voisin est inférieure à une distance déterminée, ces

deux points seront inclus dans le même sous-ensemble. Un peu de la même manière que l'algorithme précédent, si deux objets sont contigus l'algorithme ne fera pas la différence entre les deux (voir exemple sur la Figure 8).

-

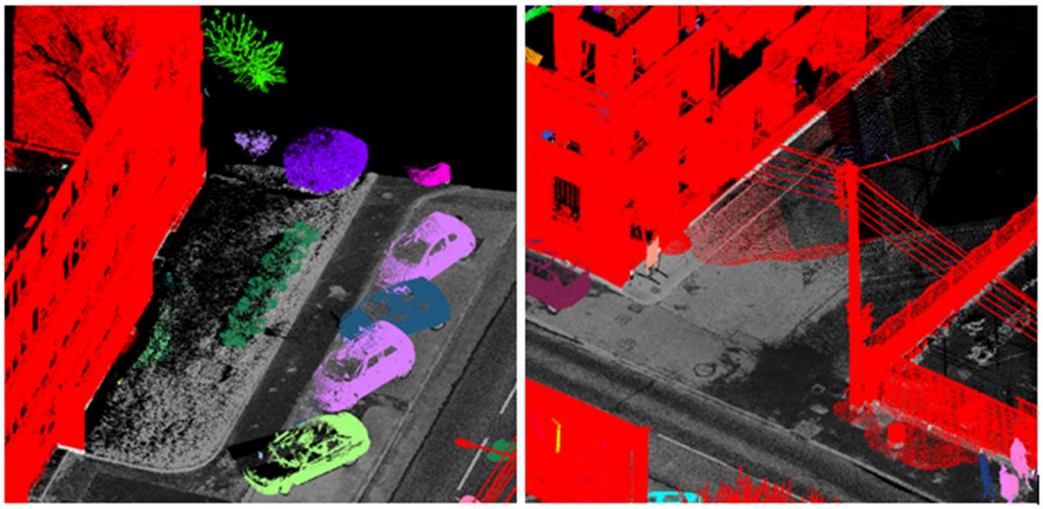


Figure 8 : Exemple de bonne segmentation (à gauche) avec des sous-ensembles définis par objet et de mauvaise segmentation (à droite) avec un sous-ensemble regroupant plusieurs objets, issues de l'algorithme "Group by density" de Terrascan

Une fois cette segmentation réalisée il est possible de regrouper ces sous-ensembles dans une même classe à l'aide de plusieurs fonctions. Il est possible de réaliser une classification supervisée avec la fonction "Test parameters" en isolant les éléments que l'on souhaite classifier dans un premier temps et en calculant différents paramètres (comme l'intensité, la couleur, la linéarité, la planéité, la hauteur par rapport au sol, le ratio largeur/hauteur…) permettant de différencier les éléments isolés du reste de la scène. Il est également possible de classer les groupes avec leur probabilité d'appartenance à une classe d'objets prédéfinis par Terrascan avec la fonction "By best match". Terrascan va attribuer une classe à un groupe si la probabilité que celui-ci soit un objet défini (comme une voiture, un arbre…).

Ces fonctions peuvent en théorie être très pratiques, mais comme il a été dit précédemment, les algorithmes de la fonction "Assign Groups" ne sont pas très fiables pour réaliser une segmentation efficace. De nombreuses tentatives ont été réalisées, dans divers environnements et avec différents paramétrages (densité du nuage de points), pour automatiser cette segmentation et classifier le nuage de points par la suite avec l'outil Macro permettant d'enchaîner plusieurs fonctions du logiciel. Il s'est avéré que dans les meilleurs cas, la segmentation était considérée réussie à hauteur de 70 % (en basant cette segmentation sur le fait que le logiciel attribue chaque sous-ensemble à un seul objet). Ce résultat n'est pas suffisant

pour considérer la segmentation réussie et ce chiffre était très variable d'un nuage à l'autre. Les fonctions citées précédemment ne pouvaient donc pas être utilisées à bon escient à cause de cette mauvaise segmentation.

Il y a également d'autres types de fonctions proposées par Terrascan qui proposent de détecter automatiquement des poteaux ou des arbres notamment. Cependant ces fonctions sont basées sur des modèles définis en amont à la manière d'une classification supervisée. Pour un type particulier de poteau par exemple, il faut au préalable l'isoler, créer un groupe manuellement et l'insérer dans la bibliothèque d'objets de Terrascan. Une fois cela fait en assignant des groupes au nuage et en appliquant la fonction avec le modèle, le logiciel devrait reconnaître les poteaux (groupes) similaires au modèle. Malheureusement on retrouve le problème cité au-dessus, à savoir que la segmentation doit être réussie pour que la fonction détecte l'objet.

Le principal inconvénient de ces algorithmes est le fait de ne pas prendre en compte à la fois des descripteurs locaux comme la verticalité du voisinage et la forme globale d'un ensemble (voir partie II). Le fait de prendre en compte uniquement le voisinage proche de chaque point induit de nombreuses erreurs sur la segmentation et entraîne une sur-segmentation ou une sous-segmentation.

- Les outils de Topodot :

Contrairement à Terrascan, Topodot propose relativement peu d'outils dédiés à la segmentation et la classification de nuage de points. Comme on le verra dans la partie suivante, ce logiciel est conçu pour aider à la vectorisation. On peut tout de même noter un outil pour extraire le sol similaire à l'outil proposé par Terrascan.

- L'outil CANUPO (CloudCompare) :

Il s'agit d'un plug-in développé lors d'une thèse par Nicolas Brodu qui permet de réaliser une classification supervisée. Cet outil est assez simple d'utilisation et permet de réaliser une classification sur plusieurs critères (dimensionnalité, nombre de points…) sur la base d'un échantillon défini (échantillon d'apprentissage). L'inconvénient de cet outil est premièrement, qu'il ne permet qu'une séparation du nuage en deux classes seulement, et

deuxièmement que les échantillons d'apprentissage sont rarement efficaces sur plusieurs nuages différents. Ces deux critères n'en font pas un bon outil pour une classification complète ou pour la détection d'objet.

I.2.4 Les méthodes de vectorisation

Pour le dessin d'un plan topographique on peut distinguer deux types d'objets qui seront représentés, les objets linéaires et les objets ponctuels. Les bords de chaussée, les fils d'eau, les bandes blanches, les bâtiments ou les murs sont représentés par des éléments linéaires sur le plan. Tandis que les poteaux, les panneaux, les regards seront représentés par des éléments ponctuels (accompagnés de symboles pour la plupart). Pour la représentation de ces éléments on retrouve globalement les mêmes méthodes adaptées à chaque type. Il existe à l'heure d'aujourd'hui deux façons de procéder, à savoir la façon manuelle et la façon semi-automatique.

I.2.4.1 La méthode manuelle

Pour les éléments linéaires, la méthode manuelle consiste à avoir une vue de dessus du nuage de points et représenter les éléments souhaités en accrochant les sommets du vecteur (ligne) à la classe sol (déterminée auparavant à l'aide des fonctions "Hard Surface" ou "Ground" sur Terrascan) ce qui va définir l'élément en planimétrie. Il sera nécessaire de visualiser l'élément en coupe également pour vérifier l'altimétrie. L'opérateur peut s'appuyer sur différents types de visualisations proposées par Terrascan comme la visualisation de l'intensité, la hauteur par rapport au sol, les normales aux points etc.

Pour les éléments ponctuels, la démarche consiste à réaliser une coupe sur l'élément à représenter, à placer un point au centre de l'objet en vue de dessus et d'ajuster l'altimétrie si nécessaire dans la coupe. Il est ensuite possible de placer un symbole pour définir l'objet représenté.

Cette méthode est plutôt fastidieuse pour l'opérateur car elle nécessite de nombreux déplacements dans le nuage de points pour repérer le ou les éléments dans un premier temps, les dessiner dans une vue dans un second temps et finalement à vérifier l'altimétrie. De plus, il n'est pas rare que la classe « sol » soit mal définie et que l'accrochage s'active sur des points mal classés qu'il faut corriger par la suite.

I.2.4.2 La méthode semi-automatique

En plus de petites méthodes qui permettent d'optimiser le temps de dessin, comme le fait de décaler des éléments linéaires parallèles d'une distance fixe (comme le fil d'eau et la bordure de trottoir par exemple), certains logiciels comme Topodot ont mis en place des fonctions pour faciliter le dessin de plans topographiques à partir de nuages de points. On peut prendre l'exemple de trois outils présents dans Topodot avec l'extraction par intensité, l'extraction d'éléments linéaires et l'identification d'éléments verticaux.

Le premier outil sert principalement à vectoriser les marquages au sol. En indiquant la continuité ou non du marquage, la taille minimale des bandes, l'intervalle minimum entre deux bandes et la direction de la ligne, l'outil permet de vectoriser ce marquage au sol de manière relativement efficace. En effet l'outil est efficace pour les marquages linéaires et où l'intensité peut être isolée. Cependant si le marquage observe des courbes et que la différence d'intensité avec les éléments alentours la vectorisation sera raccrochée à des éléments n'appartenant pas au marquage et peut rapidement dériver. Enfin avec cet outil on touche à un problème majeur de la vectorisation automatique des éléments linéaires à savoir l'emprise du vecteur. Si l'opérateur peut rapidement voir où commence et où se termine un élément linéaire, il est bien plus complexe de façon automatique de déterminer des paramètres pour définir cette emprise. Avec cet outil on se retrouve régulièrement avec un vecteur qui dérive à la première courbe que décrit le marquage et le vecteur finit sa course en dehors de la route (voir Figure 9 ci-après).

Figure 9 : Vectorisation réussie (à gauche) et mauvaise vectorisation (à droite) issues de l'outil "Extraction by intensity" de Topodot

Le second outil permet d'optimiser la méthode manuelle pour tracer certains linéaires comme les bordures de trottoir (voir Figure 10). En définissant une ligne guide, l'outil va créer des profils à intervalle déterminé en suivant ce guide. L'opérateur pourra définir un modèle au niveau du fil d'eau en indiquant plusieurs points (haut de la bordure, caniveau…) ce qui formera plusieurs lignes au fur et à mesure des profils. Cet outil est très efficace pour optimiser la méthode manuelle puisqu'elle permet de définir rapidement la planimétrie avec la ligne guide et de définir avec précision l'altimétrie de plusieurs éléments en peu de temps. Le principal inconvénient de cet outil est sa flexibilité. L'opérateur peut choisir de ne pas mettre de contrainte entre les différents points du modèle et à chaque profil le modèle ne suivra pas le nuage, ce qui engendre de petites retouches à effectuer manuellement et une hétérogénéité entre les profils. L'opérateur peut également choisir de contraindre le modèle, ce qui va engendrer une homogénéité entre les profils mais celui-ci aura des difficultés à s'adapter aux changements qui peuvent être fréquents (présence de grilles, abaissement de trottoir, courbe…). Ces changements demanderont également des retouches manuelles. L'autre inconvénient de cet outil est qu'il faut donc prévoir en amont de placer des profils aux emplacements des éléments de réseaux, des abaissements de trottoir, de courbes puisque l'outil ne le détecte pas automatiquement.

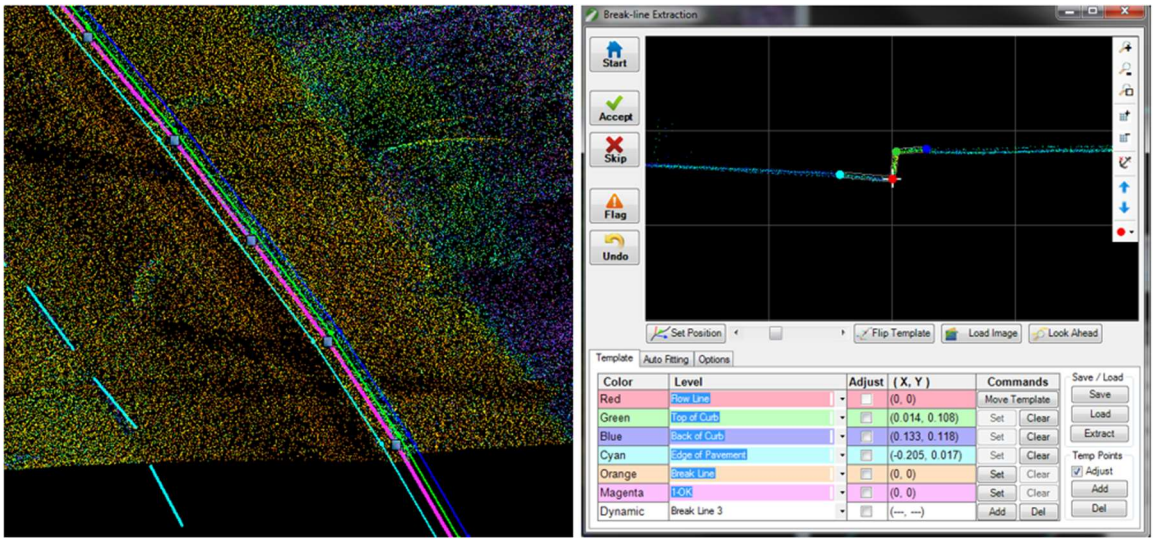


Figure 10 : Vectorisation d'une bordure de trottoir avec l'outil "Breakline extraction" de Topodot

Enfin le dernier outil permet d'optimiser la méthode manuelle pour vectoriser les éléments ponctuels comme les arbres et les poteaux. Celui-ci détecte de manière semi-automatique des éléments verticaux, l'opérateur aura à valider ces éléments et aura la possibilité de placer un symbole (cellules dans Microstation) en s'aidant de deux coupes verticales définies par l'outil. Il faut cependant avoir une classification définie auparavant, ou nettoyer le nuage pour isoler les éléments en amont.

Toutes ces méthodes de vectorisation semi-automatique sont bien plus efficaces que les méthodes manuelles mais nécessitent des opérations ou des retouches manuelles quasi systématiques. Elles permettent en général d'obtenir un gain de temps, mais sont loin d'être automatiques. Cependant on peut déduire des observations précédentes quelques points importants pour la détection et la vectorisation d'objets. Pour les objets ponctuels le plus important est de pouvoir distinguer un objet de son entourage afin de définir sa planimétrie avec le plus de précision possible. Son altimétrie sera dans la majeure partie des cas rattachée au sol, d'où l'importance d'avoir une segmentation au niveau du sol fiable. Concernant les objets linéaires, de même que pour les ponctuels, il est important de distinguer l'objet de son entourage. À cela s'ajoute des problématiques d'emprises puisqu'il est question de savoir où commence et se termine cette ligne. Et enfin ces linéaires peuvent comporter plusieurs sections avec des primitives différentes comme des droites, des courbes régulières (portions de cercle) et des courbes irrégulières (clothoïdes[1]).

C'est sur cette problématique d'optimisation, de gain de temps et dans le but de réduire les opérations réalisées par l'opérateur que la suite de ce rapport va se focaliser en essayant de détecter au mieux les objets utiles aux plans topographiques pour finalement les vectoriser.

[1] Courbe plane caractérisée par la propriété que sa courbure en un point est proportionnelle à l'abscisse curviligne du point.

II Les algorithmes de détection d'objets

Après avoir testé les outils que proposent différents logiciels, on se rend compte que les outils orientés pour la segmentation d'objets spécifiques sont assez limités et nécessitent de nombreuses corrections manuelles par la suite. Dans cette partie nous verrons donc les caractéristiques spécifiques que peuvent présenter les différents objets que nous souhaitons segmenter afin de se concentrer sur celles-ci. Puis nous verrons les algorithmes mis en place pour réaliser cette segmentation.

La programmation de ces algorithmes a été réalisée sous forme de scripts Python. Quelques bibliothèques de fonctions ont été utilisées comme la bibliothèque Laspy, permettant de lire et d'exploiter les nuages de points au format *.las* et également la bibliothèque Pyntcloud permettant de créer des voxels, de calculer des voisinages de points… Plus de détails sur ces bibliothèques et sur les fonctions utilisées sont développés en **Annexe 1**. Le logiciel CloudCompare a également été utilisé en tant que viewer. Pour comparer rapidement, et de façon visuelle les résultats, les nuages de points étaient exportés au format *.pts* et importés sous CloudCompare.

II.1 Description des objets

Intuitivement, la première chose à laquelle on pense est d'utiliser directement les attributs des points comme leurs coordonnées, l'intensité (et la couleur si disponible). Cependant cette façon de penser ne fonctionne pas, elle ne permet pas de faire de lien entre chaque point. Or un objet est constitué d'une multitude de points dans le nuage et ce sont ces relations entre les points qui permettent de définir un objet dans un nuage.

Dans la littérature on retrouve de nombreuses manières de définir les caractéristiques des points dans un nuage en prenant en compte un voisinage défini. Ils sont généralement appelés les descripteurs et séparés en deux grandes catégories, les descripteurs locaux et les descripteurs globaux. Un article de X-F.Han et al. (2018) [8] propose une liste non exhaustive de ces types de descripteurs.

Cette partie proposera donc dans un premier temps une explication sur les descripteurs locaux qui seront utilisés par la suite, puis l'équivalent pour les descripteurs globaux.

II.1.1 Les descripteurs locaux

La majorité des descripteurs locaux se base sur un point du nuage pour le caractériser et calcule ce descripteur à l'aide du voisinage du point. Ce voisinage est généralement défini à l'aide des k plus proches voisins (KNN algorithm[2]) ou à partir des points compris dans un rayon défini ayant pour centre le point considéré.

II.1.1.1 Point Feature Histogram

Le premier descripteur présenté se nomme Point Feature Histogram, il a été présenté par Rusu et al. (2008) [9]. Il utilise la relation entre les paires de points d'une région définie (voisinage du point considéré). Il calcule dans un premier temps la normale à la surface estimée (calcul de PCA[3] sur la matrice contenant les k plus proches voisins) pour définir un repère de Darboux[4] (u, v, w) entre chaque paire de points. Dans la formule (1), p_t et p_s correspondent aux coordonnées des points T et S respectivement et n_t , n_s aux vecteurs normaux associés à ces mêmes points T et S.

$$\boldsymbol{u = n_s \ ; \ v = u \times \frac{(p_t - p_s)}{|p_t - p_s|^2} \ ; \ w = u \times v} \qquad (1)$$

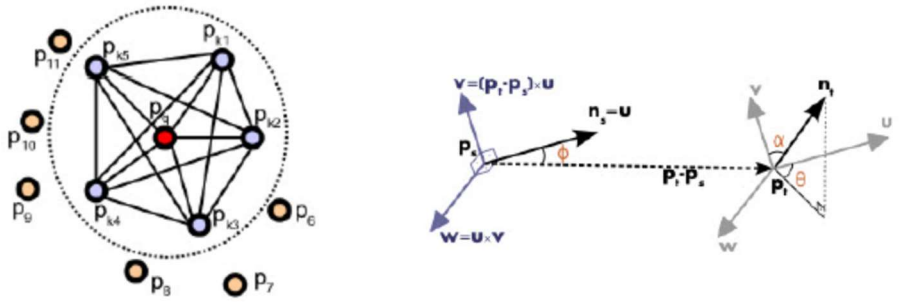


Figure 11 : Point Feature Histogram - Voisinage d'un point p_q (à gauche), Repères de Darboux entre les points p_s et p_t (à droite)

Trois angles sont ensuite calculés ($\boldsymbol{\alpha}$, $\boldsymbol{\Phi}$, $\boldsymbol{\theta}$) exprimant la différence entre les normales n_t et n_s (voir Figure 11). La distance est également calculée pour chaque paire de points compris dans le voisinage. Ces angles sont ensuite stockés dans des histogrammes.

Ce descripteur permet d'obtenir un bon indicateur de la courbure d'une surface. Dans notre cas un dérivé de ce descripteur sera appliqué pour la détection de poteaux.

---

[2] Acronyme de "k-Nearest Neighbours", traduit par les "k plus proches voisins"
[3] Acronyme de Principal Component Analysis, traduit par Analyse en Composantes Principales est une méthode qui consiste à transformer des variables corrélées en variables décorrélées.
[4] Repère mobile utile pour l'étude des courbes tracées sur une surface de l'espace euclidien orienté à trois dimensions.

II.1.1.2 Spin Image

Le second descripteur est le Spin Image introduit par Johnson et al, (1998) [10]. Utilisé à la base pour réaliser des correspondances entre surfaces dans un nuage de points complexe, celui-ci introduit un système de coordonnées cylindriques local pour chaque point (voir Figure 12), avec comme repère les coordonnées du point *p* et sa normale n. Chaque point du voisinage se voit attribuer une paire de coordonnées ($\boldsymbol{\alpha}$, $\boldsymbol{\beta}$). La première coordonnée $\boldsymbol{\alpha}$ correspond à la distance perpendiculaire (guidée par le vecteur n_q perpendiculaire au vecteur normal *n*) entre le point *p* et le point *q*. Tandis que la seconde coordonnée $\boldsymbol{\beta}$ correspond à la distance entre le plan de vecteur normal *n* et le point *q*. Ces coordonnées sont calculées par la relation (2) ci-après.

$$\boldsymbol{\alpha} = \boldsymbol{n_q}\ ;\ \boldsymbol{\beta} = \sqrt{|\boldsymbol{p}-\boldsymbol{q}|^2 - \boldsymbol{\alpha}^2} \qquad (2)$$

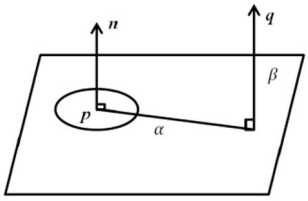


Figure 12 : Spin image - Système de coordonnées cylindrique local

II.1.1.3 Local Elevation Difference

Ce descripteur permet assez simplement de calculer la différence d'altitude locale (voir Figure 13). Cela permet de mettre en avant les changements d'altitude très marqués. Il est calculé en prenant le voisinage de chaque point. La moyenne des altitudes des points voisins est soustraite à l'altitude du point concerné. On retient ensuite la valeur absolue de cette différence. Si le relief est localement plat, la valeur retenue pour le point sera proche de 0. Si localement il y a un changement d'altitude marqué, la valeur retenue sera supérieure à 0.

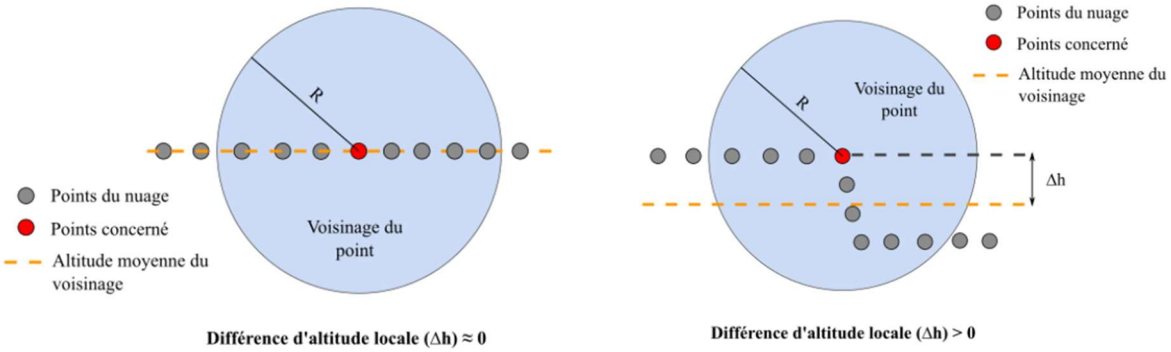


Figure 13 : Local Elevation Difference - Exemple d'un relief plat (à gauche) et d'un relief marqué (à droite)

Le voisinage à prendre en compte est donc très important puisque celui-ci doit approximer la taille de l'élément que l'on veut mettre en valeur. Dans notre cas ce descripteur sera utilisé pour détecter les bordures de trottoirs, le voisinage sera donc compris dans un rayon de 20 centimètres.

II.1.2 Les descripteurs globaux

Pour avoir une analyse à plus grande échelle, les descripteurs globaux ne se basent plus sur une petite zone du nuage de point (comme le voisinage proche) mais sur tout le nuage ou sur un cluster entier. Dans notre cas les descripteurs globaux serviront principalement à confirmer ou à rejeter les clusters définis par les descripteurs locaux. Ces descripteurs sont généralement des dérivés de descripteurs locaux appliqués à une zone d'étude plus large.

II.1.2.1 Global Fourier Histogram

Ce descripteur est dérivé du descripteur Spin Image et expliqué dans l'article de X-F.Han et al. (2018) [7] Au lieu de prendre en compte le voisinage d'un point comme dans le descripteur local, le calcul s'effectue sur un cluster entier. De manière générale, le point au centre du cluster est utilisé comme point origine et une paire de coordonnées est attribuée à chaque point du cluster de la même manière que pour le Spin Image. Cela permet de confirmer que le cluster s'inscrit bien dans un cylindre par exemple. Dans cette étude il est utilisé pour confirmer ou infirmer les clusters retenus pour la détection des poteaux.

II.1.2.2 Successives Angles

Ce descripteur est assez simple à mettre en place puisqu'il va consister à calculer l'angle qui est formé entre un vecteur défini (comme l'axe x) et le vecteur formé par les centres de deux clusters. Cela permet de s'assurer de la cohérence dans la forme que prend plusieurs clusters. Ce descripteur permet d'exclure les clusters qui s'éloignent de la direction prises par les clusters précédents. Ainsi une tolérance est définie, lorsque la différence entre l'angle calculé sur le cluster concerné et l'angle du cluster précédent est supérieure à la tolérance, celui-ci est exclu.

II.1.3 Caractéristiques particulières des objets

Dans cette partie on va se concentrer sur les caractéristiques propres que présentent les objets afin de pouvoir les identifier par la suite. Comme dit précédemment, on peut dans un premier temps séparer les objets ponctuels et linéaires dans leur représentation sur un plan. On se focalise sur les objets les plus représentés dans un plan topographique, à savoir les poteaux, les grilles, plaques, bordures de trottoirs, végétation, marques de peinture…

On regroupe dans le Tableau 1 ci-après les caractéristiques qui sont propres à ces objets sur lesquels on pourra s'appuyer pour leur détection et également les caractéristiques sur lesquelles la distinction peut être complexe d'un point de vue géométrique.

Type d'objet	*Objet*	*Caractéristiques principales*	*Similarités*
Ponctuel	Poteaux	• Objet linéaire et vertical • Compris dans le sursol • Diamètre régulier	• Similaire aux troncs d'arbres
	Arbres	• Composé d'une partie linéaire et verticale (tronc) et d'une partie en volume (feuillage) • Compris dans le sursol	• Partie linéaire similaire aux poteaux
	Plaques et grilles	• Compris dans le sol • Intensité légèrement différente du sol localement • Forme géométrique (cercle, carré)	• Non différenciable du sol géométriquement
Linéaire	Trottoirs	• Objet linéaire • Compris dans le plan du sol • Forme une rupture avec la route	• Même retour (intensité, couleur) que la route et le trottoir
	Marques de peinture	• Objet linéaire • Compris dans le sol • Intensité et couleur différente de la route	• Intensité variable si peinture détériorée

Tableau 1 : Caractéristiques des objets

Ce tableau permet de résumer les caractéristiques principales des objets et on se basera sur cela pour définir les algorithmes de détection. En grande partie les caractéristiques principales serviront à déterminer les descripteurs locaux à utiliser et les similarités serviront à définir les descripteurs globaux. On procédera donc de la manière suivante pour détecter les objets. Dans un premier temps on calculera un descripteur local basé sur une caractéristique principale de l'objet afin d'obtenir une première classification du nuage. Dans un second temps on regroupera les points similaires afin d'obtenir des clusters de points homogènes (selon le premier descripteur). Enfin ces clusters seront analysés à l'aide d'un descripteur global pour vérifier les similarités avec d'autres objets.

Cette étude va se focaliser sur des exemples au vu des algorithmes relativement complexes à mettre en place. Nous avons choisi pour cela des éléments fréquents dans les deux types d'objets décrits au-dessus à savoir les poteaux et les bordures de trottoir.

II.2 Objet ponctuel : Les poteaux

Pour les objets linéaires et verticaux que sont les poteaux, il a été décidé de suivre en grande partie l'algorithme proposé par F.Tombari et al. (2014) [11]. Cet article propose une méthode pour la détection automatique d'objets ressemblant à des poteaux dans un nuage de points acquis par MMS. Cette détection étant relativement complexe due à la forme fine que peuvent avoir ces éléments (exemple avec les supports de panneaux de signalisation), le bruit et les occlusions fréquentes dans ce type de nuage. Il propose donc une méthode basée sur des descripteurs locaux, une "clusterisation" et un descripteur global pour valider ces groupes (comme le montre la Figure 14).

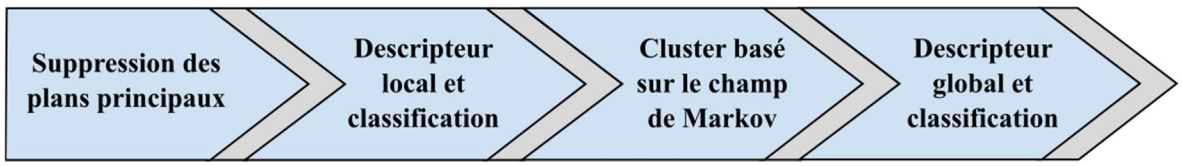


Figure 14 : Chaîne de traitement de l'algorithme

Un organigramme détaillant les différentes étapes de l'algorithme est présent en **Annexe 2**. Certaines parties de scripts écrites en Python seront également explicitées en pseudo-code au fur et à mesure de l'algorithme.

II.2.1 Prétraitements

Avant de commencer l'algorithme, le nuage est traité en retirant les points isolés et le bruit. On applique ensuite un sous-échantillonnage aléatoire puis une voxelisation consistant à créer une grille 3D, avec un pas défini dans notre cas (voir Figure 15). Ce pas sera de quelques centimètres pour conserver au mieux la précision du nuage. Un algorithme de RANSAC sera ensuite appliqué pour filtrer les plans principaux, à savoir le sol et les bâtiments sachant que les éléments recherchés ne se situent pas dans ces plans. (S. Gebhardt, 2009) [12].

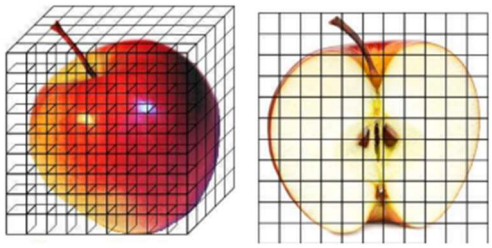


Figure 15 : Schéma du Gridded Voxel Model (Source : [12])

II.2.2 Point Feature Histogram et Support Machine Vector

Il propose dans un premier temps d'utiliser un descripteur local dérivé du Point Feature Histogram décrit auparavant. Celui-ci propose pour chaque point de choisir un voisinage situé entre un rayon minimum et un rayon maximum (entre 0.5 m et 1.5 m par exemple), le rayon minimal correspondant approximativement au diamètre d'un poteau. Il calcule ensuite l'angle décrit entre le point concerné et chaque point compris dans ce voisinage et la verticale. Ces angles sont ensuite regroupés dans un histogramme avec un nombre d'intervalles définis (10 intervalles dans notre cas). Le fait de ne pas choisir un voisinage directement proche au point permet de conserver la principale caractéristique d'un poteau à savoir sa verticalité, comme le montre le schéma de la Figure 16.

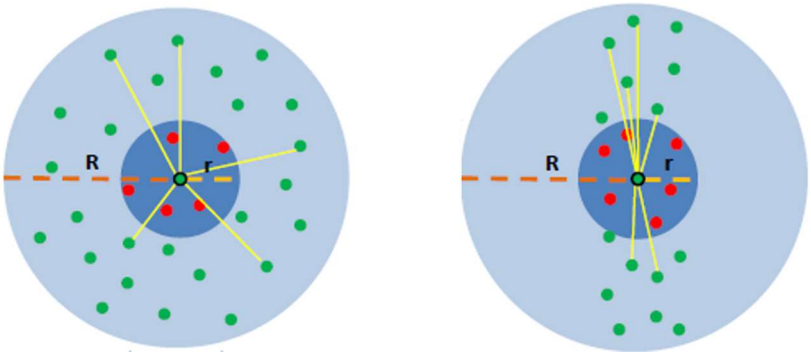


Figure 16 : Coupe verticale d'un nuage, cas d'un objet en volume (à gauche) et d'un poteau (à droite) (Source : [11])

En parallèle, deux échantillons sont choisis, l'un contenant uniquement des poteaux et l'autre contenant d'autres objets comme du sol, de la végétation, des bâtiments… Ces échantillons sont des échantillons d'apprentissage pour effectuer une méthode d'apprentissage supervisée appelée SVM (Support Vector Machine). En ayant labellisé les échantillons en tant que "poteau" et "autre", celle-ci va calculer la probabilité pour que chaque point appartienne à une classe ou à une autre. Les histogrammes des échantillons et du nuage de point seront donc comparés pour calculer ces probabilités. Plus de détails sur la méthode SVM sont disponibles en **Annexe 1**.

Sur ces histogrammes représentés dans le Tableau 2, on peut voir que ceux liés aux poteaux ont des valeurs de cosinus regroupées majoritairement entre 0.5 et 1. Ce qui semble logique puisque les angles sont calculés par rapport à la verticale, les angles associés aux poteaux doivent être compris entre 0° et 30°. En revanche, les valeurs sur les histogrammes des autres points sont plus dispersées et répartis de manière plus aléatoire.

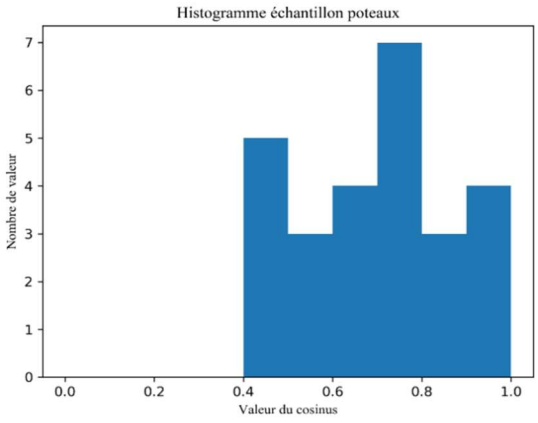
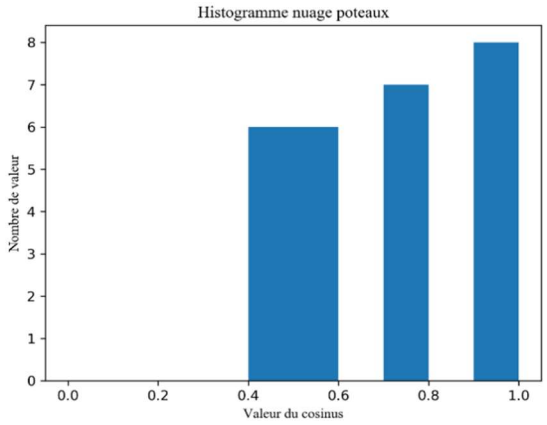
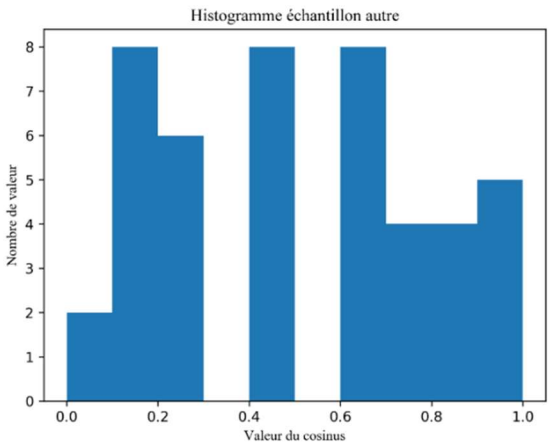
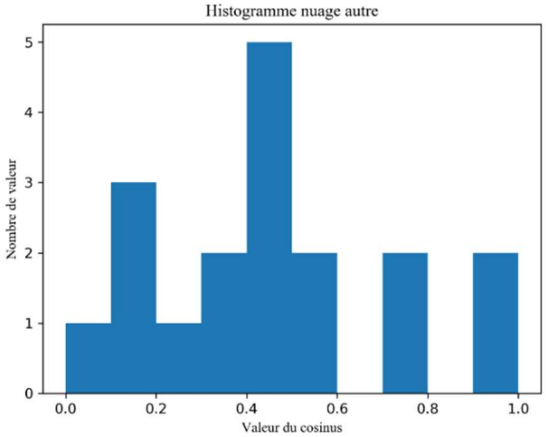
	Échantillons d'apprentissage	*Nuage testé*
Poteau		
Autres		

Tableau 2 : Comparaison des histogrammes entre les échantillons et le nuage. Les histogrammes sont composés des angles entre le vecteur vertical et les vecteurs entre le point concerné et chaque point de son voisinage.

Un pseudo-code développant de façon plus précise la façon dont a été implémenté cette partie d'algorithme est présent en **Annexe 3** et **Annexe 4**.

II.2.3 Homogénéisation de la classification

Après cette première classification on se rend compte que de nombreux points sont mal classifiés, que ce soit des points compris dans un poteau qui ne sont pas classés comme tel (faux positif), ou inversement, des points classés comme des poteaux qui n'en sont pas en réalité (faux négatif). Pour corriger en partie ces erreurs, on utilise un Markov Random Field (MRF) couplé à un Loopy Belief Propagation (LBP). Le MRF va permettre de créer un graphique (voir Figure 17) qui lie les points les uns avec les autres, on utilisera pour cela un kd-tree. Cela va permettre d'apporter plus de cohérence entre les points voisins dans le nuage.

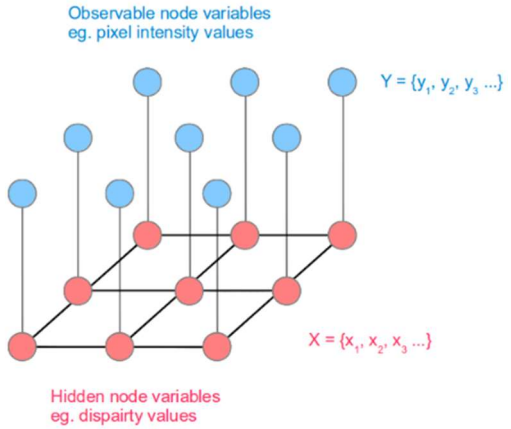


Figure 17 : Illustration du MRF. En bleu les points du nuage, en rouge le kd-tree (MRF) liant chaque point à ses voisins

Une fonction (appelée fonction d'énergie E) va ensuite être calculée pour le LBP à partir des distances entre chaque point pour déterminer l'influence que va avoir chaque voisin sur le point concerné. Pour vulgariser le LBP, la comparaison est souvent faite avec des messages que s'envoient les points, ces messages contenant le label du point et un autre paramètre qui permet de déterminer l'influence du voisin sur le point. Dans l'équation (3), la fonction est nommée *E(X)* représentant l'énergie de chaque point considérés *X*. x_i correspond au label du point i, x_j correspond au label associé au voisin j. La première partie de la fonction est appelée la fonction coût (Datacost) et la seconde partie est appelée la fonction de lissage (Smoothness) (F.Tombari, 2011) [13].

$$E(X) = \sum_i \varphi_i(x_i) + \sum_i \varphi_{i,j}(x_i, x_j) \qquad (3)$$

On va essayer de minimiser cette fonction pour chaque point. Au final si un point considéré comme n'étant pas un poteau est entouré de trois voisins étant classé comme un poteau et d'un point n'étant pas classé comme un poteau, son voisinage va suggérer que ce point est mal classé et on lui fera changer de classe (comme montre le schéma de la Figure 18).

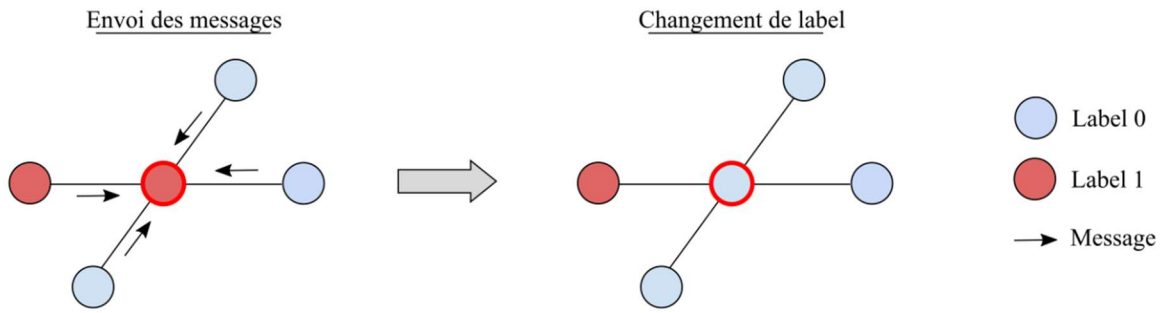


Figure 18 : Schéma des envois de message pour le LBP

Un pseudo-code de la façon dont a été programmée cette partie d'algorithme est disponible en **Annexe 5**.

Étant donné la manière dont a été programmé l'algorithme, une seule itération sera réalisée (nombre de fois où l'on applique l'algorithme expliqué ci-dessus pour minimiser l'équation). Après avoir testé plusieurs valeurs d'itérations, lorsque le nombre était supérieur à 1, la qualité de la classification se dégradait fortement et de nombreux points bien classifiés changeaient de labels. On estime donc qu'une itération est suffisante.

On peut remarquer dès cette étape que les poteaux sont bien détectés mais sont sous-segmentés. Autrement dit, tous les points qui composent le poteau ne sont pas tous bien classés. C'est la voxelisation et le descripteur local en lui-même qui induisent ce phénomène. En effet, la voxelisation réduit le nombre de points et le descripteur, de par sa nature ne prend pas tous les points en compte. Si un câble est accroché au milieu du poteau, celui-ci n'est localement pas vertical et n'est donc pas pris en compte. Cependant dans ce cas on va préférer un algorithme plus strict qui prendra en compte uniquement des poteaux plutôt qu'un algorithme plus laxiste qui prendra en compte d'autres éléments. Sur la Figure 19 ci-après on voit que la base du poteau et certains niveaux ne sont pas pris en compte à cause de la présence d'un muret ou de câbles.

II.2.4 Clustering

Une fois que les points sont globalement bien classés, on utilise un algorithme pour regrouper les groupes de points et créer les clusters. Dans l'article les points sont regroupés grâce à un parcours de l'arbre (kd-tree) appelé algorithme "Breadth-First Search"[5]. Dans notre cas nous avons mis en place l'algorithme DBSCAN (Density-Based Spatial Clustering of Applications with Noise) que nous avons trouvé plus simple à mettre en place et plus efficace. Cette fonction est disponible dans la bibliothèque Scikit-learn et plus de détails sont disponibles en **Annexe 1**. La mise en place de cet outil suppose qu'aucun cluster n'est collé l'un à l'autre puisque cet algorithme va regrouper les éléments classés comme poteau entre eux si la distance entre chaque point est inférieure à un seuil donné. Il faut également un nombre minimum de point pour créer le cluster. Le résultat de cette clusterisation est visible sur la Figure 19.

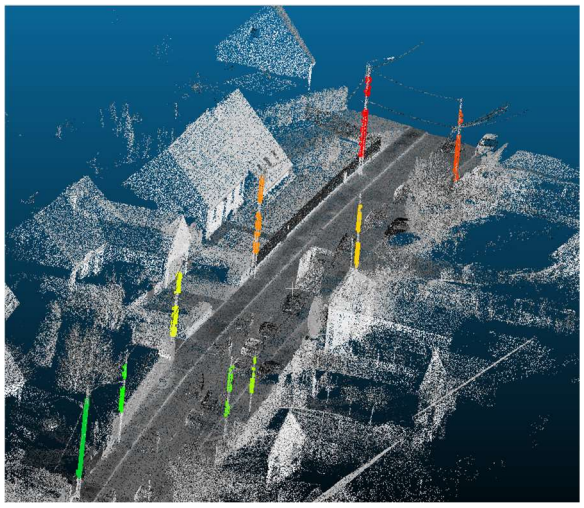


Figure 19 : Visualisation des clusters sous CloudCompare

Cette opération nous permet, comme on peut le voir sur la Figure 19, de séparer chaque poteau en un groupe distinct (cluster). Étant donné les valeurs de rayons entrés au début de l'algorithme, seul les poteaux avec un diamètre autour de 50 centimètres sont détectés. On retrouve également le phénomène de sous-segmentation cité au-dessus.

II.2.5 Validation globale des clusters

Comme énoncé précédemment dans la partie II.1.2, la validation globale permet une validation à l'échelle du cluster et non plus à l'échelle d'un point et de son voisinage. Les poteaux étant assimilés à des cylindres, on utilise un descripteur dérivé du Global Fourier

[5] Algorithme de parcours en largeur qui permet le parcours de graphe ou d'arbre

Histogram pour réaliser la validation de nos clusters. Cette validation est réalisée surtout pour éliminer les troncs d'arbre qui peuvent être détectés puisqu'ils ont la même forme cylindrique que les poteaux. On se base donc sur la principale différence entre les deux types d'objets à savoir le feuillage.

Dans un premier temps on calcule un point origine grâce à la moyenne des coordonnées X, Y, Z[6] des points contenus dans le cluster. À partir de cela on calcule les coordonnées cylindriques de chaque point sachant que la base de ce repère est le point origine calculé auparavant et le vecteur vertical. On s'assure que les points du cluster s'inscrivent bien dans un cylindre dont le rayon est défini par le rayon minimal défini à la partie II.2.2.

II.2.6 Création des points topographiques

L'objectif final étant de pouvoir utiliser cet algorithme pour dessiner les plans de façon automatique, on souhaite récupérer un point unique pour chaque poteau détecté afin de le représenter sur un plan. Pour cela, la méthode que nous avons trouvé la plus judicieuse a été de former des clusters dans le nuage de points à l'origine du traitement (sans sous-échantillonnage ou de voxelisation). Pour chaque cluster trouvé auparavant et à partir de chaque point, nous cherchons les points dans le nuage d'origine qui se situent proche des points détectés. Cette démarche est mise en place avec un système de voisins situés dans un rayon inférieur au pas de voxelisation.

Une fois cela réalisé, deux méthodes ont été testées. La première consistait sur chaque cluster à calculer une moyenne des coordonnées X et Y pour obtenir un point en planimétrie. Pour la coordonnée Z, le point le plus proche du point moyenné sur le nuage du sol (extrait par RANSAC) est récupéré puis on affecte sa coordonnée Z au point moyenné. La seconde méthode consiste à prendre une section du cluster située entre un et deux mètres au-dessus du point le plus bas du cluster. Une moyenne en planimétrie des points de cette section est calculée puis la même méthode était effectuée pour la composante Z. La seconde méthode permet en théorie de ne pas prendre en compte le défaut de verticalité des objets et les potentiels points résiduels situés à la base du cluster qui peuvent faire varier les coordonnées du point de plusieurs centimètres.

[6] Ces coordonnées sont à la base des coordonnées en Conique Conforme 9 zones. Mais celles-ci sont ramenées à un repère local dès le début de l'algorithme pour soulager la mémoire vive de l'ordinateur.

La méthode pour retrouver les points sur le nuage d'origine comporte le défaut de prendre des points supplémentaires dans les clusters (comme des câbles ou un peu de végétation) comme on peut voir sur la Figure 20 (droite). Cependant pour les câbles ils sont généralement symétriques, ce qui est annulé par la moyenne. Étant donné le nombre de points que comporte le cluster, la moyenne sera très peu affectée par quelques points légèrement séparés.

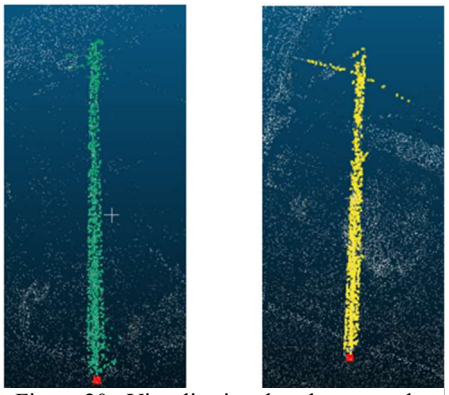


Figure 20 : Visualisation des clusters et des points topographiques (points rouges) sous CloudCompare

Les résultats de cet algorithme sont présentés en partie III.

II.3 Objet linéaire : Les bordures

Pour les bordures de trottoir qui sont des éléments inclus dans le sol qui forme néanmoins une rupture avec la route, il a été décidé de s'inspirer de différentes méthodes trouvées dans des articles scientifiques. Notamment l'article de B.Yang et al. (2016) [14] mais également d'utiliser la trajectoire qui est mesurée et calculée pour chaque levé MMS. Pour cela on propose d'utiliser un descripteur local basé sur la différence d'altitude comme descripteur local et la position de l'objet par rapport à la trajectoire. L'algorithme sera concentré sur la détection de la ligne de fil d'eau. Un organigramme est disponible en **Annexe 6** reprenant le déroulé global de l'algorithme.

II.3.1 Prétraitements

De la même manière que pour les poteaux, nous allons faire un tri dans le nuage dès le début de l'algorithme afin de ne pas alourdir les traitements. Dans un premier temps un sous-échantillonnage aléatoire est réalisé puis un algorithme de RANSAC afin de séparer le sol du sursol. On choisira une tolérance assez élevée (autour de 0.5 m) pour le plan afin de garder un maximum de points pour le sol. En effet les bordures étant comprises dans le sol mais sur un plan légèrement plus haut que le sol, cette tolérance est importante à préciser. Les traitements qui suivent seront donc réalisés sur la partie du nuage considérée comme le sol.

II.3.2 Local Elevation Difference

Comme expliqué dans la partie II.1.1, ce descripteur permet de savoir s'il y a une rupture de pente dans l'entourage de chaque point ou si celui-ci est plat. Pour ce calcul on prend un voisinage compris dans un rayon autour du point. Ce rayon ne doit pas être trop petit pour prendre un nombre de points assez important pour que le calcul soit représentatif, mais il ne doit également pas être trop grand pour que la différence entre deux plans soit visible. Les bordures de trottoir de "type T" dépassant de 10 à 15 centimètres du sol, on prendra 20 centimètres de rayon. On calcule ensuite la moyenne en altitude du voisinage pour la soustraire à l'altitude du point. Les résultats seront ajoutés comme attributs Δh à chaque point.

On ajoute ensuite des labels aux points étant potentiellement des bordures. Cette hypothèse est vérifiée si le Δh du point est compris entre deux et dix centimètres. Cet intervalle n'est pas choisi au hasard, puisque la borne inférieure correspond à la hauteur d'une bordure basse (environ cinq centimètres) divisée par deux, et la borne haute à la hauteur d'une bordure haute divisée par deux.

Un pseudo-code donnant plus de détails sur la façon dont a été écrite l'algorithme est disponible en **Annexe 7**.

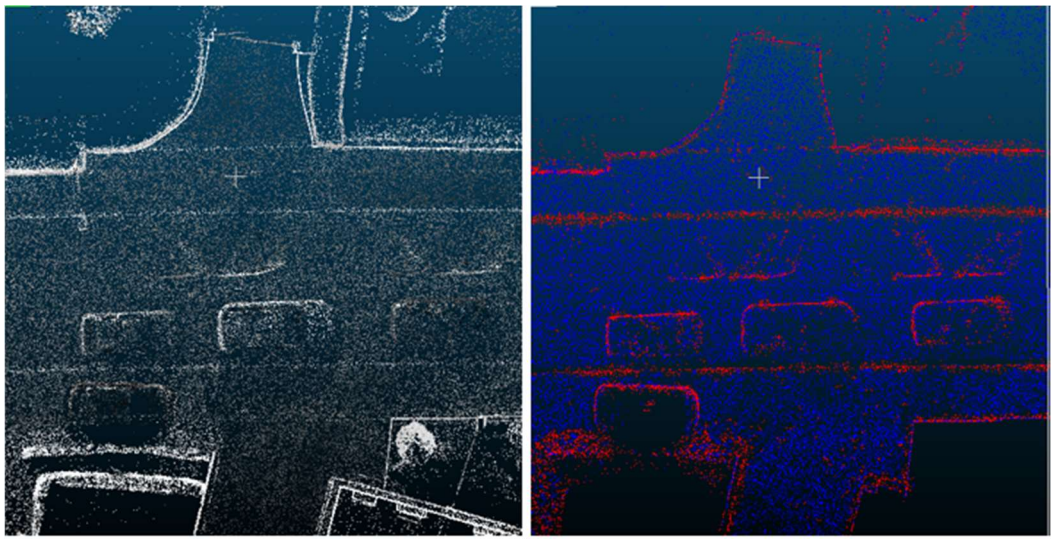


Figure 21: Visualisation d'un nuage de points en intensité (à gauche) et avec les labels (à droite)

On peut noter sur la Figure 21 que de nombreux véhicules sont détectés également. Il s'agit de véhicules en mouvement qui ont été acquis lors d'un passage du scanner. Afin d'éliminer un maximum le bruit créé par ces objets, un script a été écrit pour éliminer ces points du nuage en amont du traitement. Cet algorithme est développé en **Annexe 8**.

II.3.3 Homogénéisation de la classification

Après cette première classification, de nombreux points isolés sont classifiés en tant que bordures. Pour que la classification soit un peu plus homogène, un test est réalisé dans le voisinage de chaque point. Un peu de la même manière qu'un Markov Random Field (partie II.2.3), on va vérifier que les k-voisins de chaque point ont bien le même label que le point concerné. On réalisera une seule itération qui sera relativement stricte puisqu'on prendra k égal à huit pour ce test. Cela permettra d'éliminer la majorité des points isolés et de réduire l'emprise des points classifiés comme bordures (voir Figure 22). Le test va donc changer de label à un point classé comme bordures qui n'est pas entouré par huit points classés également comme bordures. Ce test est à sens unique, un point qui n'est pas classé comme une bordure à la base ne sera pas affecté par ce test. Un pseudo-code sur cette partie est disponible en **Annexe 9**.

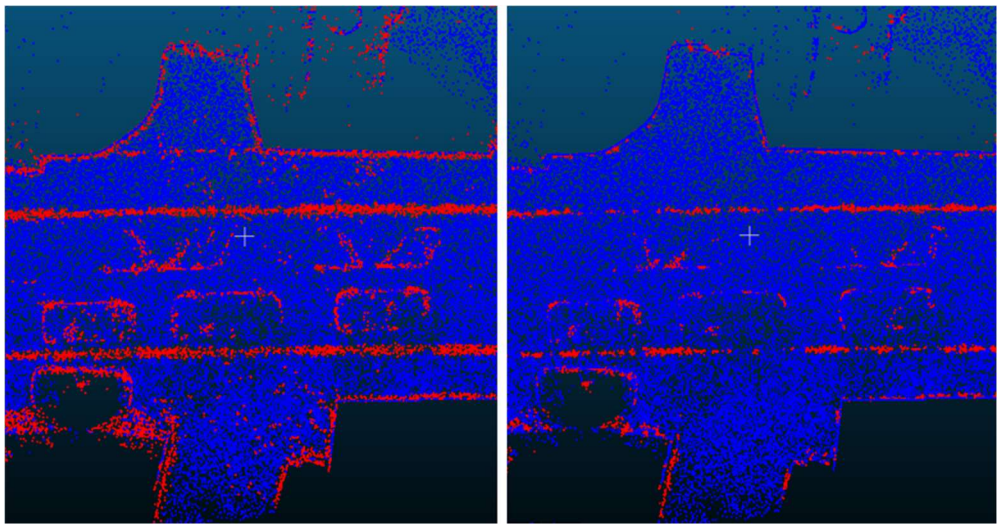
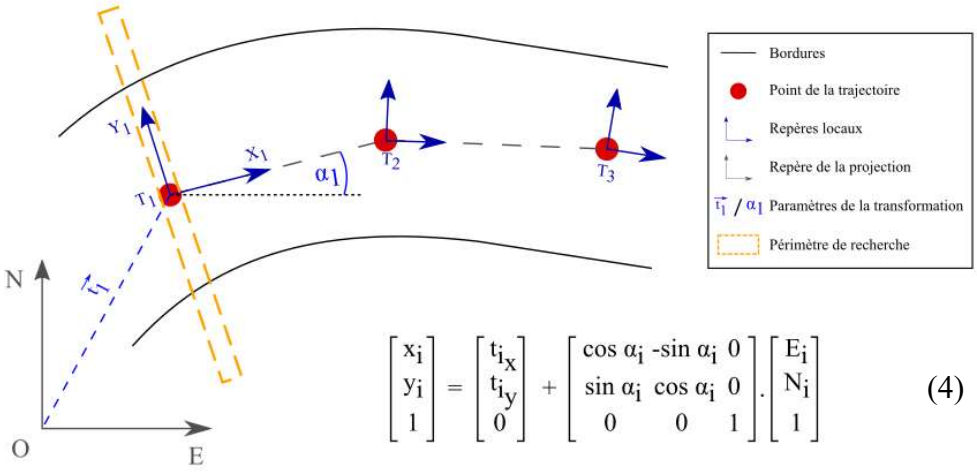


Figure 22 : Visualisation de la différence avant (à gauche) et après homogénéisation (à droite)

II.3.4 Points proches de la trajectoire

L'idée d'utiliser la trajectoire provient du fait que les trottoirs ont une position relativement fixe par rapport à celle-ci. En effet les trottoirs et donc leurs bordures se trouveront toujours de part et d'autre de la trajectoire, celle-ci se trouvant toujours d'un côté ou l'autre de la route. De plus, les coordonnées de la trajectoire sont obligatoirement calculées et corrigées pour géolocaliser le nuage de points. Les points définissant la trajectoire en sortie du calcul étant très denses (plusieurs points par centimètre de trajectoire), on effectue un sous-échantillonnage à 1 sur 100 dans un premier temps.

Dans un second temps on calcule un repère local pour chaque point de la trajectoire (voir Figure 23), en se basant sur une translation (t_i) entre le point origine du repère de la projection (O) et le point de la trajectoire (T_i). L'autre paramètre de cette transformation est une rotation d'angle α entre le vecteur x et le vecteur entre le point de la trajectoire et le point suivant $\overrightarrow{T_i T_{i+1}}$. À chacun de ces repères, on va appliquer la transformation aux points du sol, leur donnant des coordonnées locales. Ce système de repères locaux va permettre de créer des profils au niveau de chaque point de la trajectoire. On va donc retenir les points qui sont labellisés comme bordures et qui ont une coordonnée locale proche de 0 en abscisse à chaque repère local (cela correspond au périmètre de recherche sur la Figure 23). Ensuite les points sont séparés en deux parties, la première comprenant les points ayant des coordonnées négatives en ordonnées et la seconde avec des coordonnées positives. Cette séparation permet de créer la séparation entre la bordure gauche et la bordure droite dans le nuage. Finalement, les points de chacune de ces parties sont triés par ordre croissant en ordonnée, seuls les points les plus proches seront retenus. Le nombre de points à retenir par profil sera fixé à 20 de chaque côté de la route, cela permet de s'assurer d'un nombre suffisant de points pour définir un profil.



$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} t_{i_x} \\ t_{i_y} \\ 0 \end{bmatrix} + \begin{bmatrix} \cos\alpha_i & -\sin\alpha_i & 0 \\ \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} E_i \\ N_i \\ 1 \end{bmatrix} \quad (4)$$

Figure 23: Changement de repère entre le repère trajectoire et le repère de la projection

II.3.5 Clustering

On applique ensuite l'algorithme de segmentation DBSCAN vu précédemment (II.2.4). Cela permet de segmenter chaque profil et de supprimer les points isolés qui peuvent subsister (notamment à cause d'obstacles comme les voitures). Le seuil maximal (ε) correspondant à la distance maximale entre deux points pour laquelle ces deux points ne seront pas considérés dans le même cluster, a été défini à 20 centimètres.

II.3.6 Validation globale des clusters

Comme énoncé dans la partie II.1.3, on utilise les descripteurs locaux pour réaliser une première classification et les descripteurs globaux pour valider celle-ci. Dans ce cas, on pose l'hypothèse que deux points successifs définissant une bordure de trottoir (les points utilisés seront le centre des clusters), sont relativement proches et prennent globalement la même direction. Si ce n'est pas le cas, on considère soit que le cluster n'est pas bien classé, soit que la bordure forme une courbe ou encore que la bordure s'arrête à cet endroit.

On considère donc les points C_i comme le centre des clusters avec i défini comme le numéro du cluster et on va calculer la différence de gisement entre $\overrightarrow{C_i.C_{i-1}}$ et $\overrightarrow{C_i.C_{i+1}}$ (voir Figure 24). Si cette différence est comprise entre 0 et 10 grades ou entre 190 et 200 grades (à plus ou moins 200 grades), le cluster sera validé et l'algorithme passera au suivant. Si cette différence est comprise entre 10 et 190 grades, le cluster sera refusé et va clôturer un premier groupe ce qui va créer une première ligne. Le calcul recommencera entre le cluster où le groupe précédemment s'est arrêté et le cluster suivant, et ainsi de suite. Cette façon de faire permet en théorie de séparer les portions relativement droite ou en légère courbe et les portions courbes. Cela permet également d'arrêter la ligne si la bordure se termine.

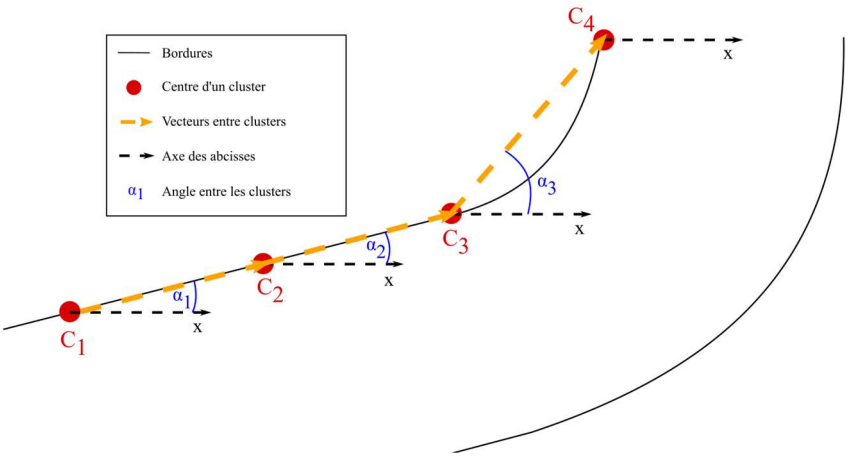


Figure 24 : Calcul des angles successifs

II.3.7 Création des points topographiques

Comme pour les poteaux l'objectif étant de faciliter le dessin de plans, il nous faut discrétiser ces bordures et former une primitive géométrique simple. Pour la ligne de fil d'eau on prendra le point le plus bas de chaque cluster validés. Pour le haut de bordure on conservera le point le plus haut dans un rayon de cinq centimètres autour du point du fil d'eau. Un fichier texte sera exporté en séparant chaque groupe de points formant chacun une ligne. Le tracé de la primitive sera géré directement par Microstation en indiquant le type d'objet souhaité. La primitive la plus adaptée pour ce genre de tracé semble être la « B-spline » qui gère bien les transitions entre sections droites et les sections courbes.

Les résultats de cet algorithme sont présentés en partie III.

II.4 Les autres algorithmes

En parallèle des algorithmes présentés, d'autres algorithmes ont également été proposés ou testés, notamment pour les objets en mouvement présenté plus en détails en **Annexe 8**. La même démarche avec les descripteurs locaux, la clusterisation, puis la validation par descripteur globaux a été suivie. Les bandes blanches peuvent notamment être détectées en isolant les intensités hautes du sol et en vérifiant que les clusters ont très peu de relief et qu'ils suivent globalement la même direction. Un test a été réalisé pour les bâtiments également en essayant d'isoler les formes localement planes dans le nuage du sursol. Ces tests étaient relativement concluants mais beaucoup de bruit subsistait au niveau de la végétation. Réciproquement, pour la végétation, des tests ont été réalisés en essayant de détecter les volumes et les clusters comprenant un taux de retours secondaires important. Cependant beaucoup de bruit subsistait au niveau des arêtes et des fenêtres des bâtiments. Un autre test a été réalisé pour les plaques de réseaux. Malgré le fait que l'intensité soit légèrement différente sur les bords des plaques comparés au sol entourant celles-ci, les histogrammes entre une plaque et du sol ne sont pas assez différent pour permettre une réelle séparation.

Les algorithmes concluants comme les objets en mouvement ou les lignes blanches ont été intégrés à l'interface graphique (cf II.5.3). Par manque de temps les autres algorithmes n'ont malheureusement pas pu être plus développés. Cependant en gardant cette démarche avec les descripteurs il est sans doute possible de détecter d'autres objets.

II.5 Gestion des données

Cette partie permet de développer certaines problématiques rencontrées au cours de l'étude et qui ont eu un impact important sur celle-ci.

II.5.1 Format des fichiers

La question du format de fichier à utiliser est importante puisqu'il existe de nombreux formats de fichiers différents, qu'ils soient libres ou propriétaires, ou bien binaire ou texte. On peut notamment citer les formats *.txt*, *.pts*, *.ply*, *.las*, *.pcd*… La norme pour les levés Lidar est d'exporter les fichiers bruts en format *.las*. La majorité des logiciels de traitement de nuages de points pouvant exploiter ce format, il a été décidé de conserver ce format. Cependant il s'agit d'un format binaire[7], il n'est pas exploitable directement en Python. La bibliothèque LasPy (**Annexe 1**) permet donc de lire les fichiers *.las* sous Python et de convertir les données sous forme de tableaux Numpy, très facilement exploitables. Les calculs réalisés dans les algorithmes sont donc effectués grâce à Numpy. Afin d'exploiter les données en sortie, les fichiers seront de nouveau convertis en format *.las* avec une nouvelle classification ce qui permettra de les lire notamment avec la suite Terrascan.

En revanche, les autres fichiers en sortie des algorithmes, ceux contenant les coordonnées des objets pour le dessin du plan, seront exportés en *.txt*. Ce format est le format le plus facile d'utilisation notamment pour l'écriture et l'import dans les logiciels. Il nous permet d'écrire très facilement des fonctions disponibles dans Microstation comme le tracé de points ou de lignes.

Lorsqu'elles sont utilisées, notamment dans la détection de bordures, les fichiers trajectoires sont exportés au format *.txt* avant d'être intégré dans l'algorithme. Les fichiers sont ensuite concaténés dans l'algorithme pour ne former qu'une seule donnée.

II.5.2 Gestion de la quantité de données

Les nuages de points relevés par Lidar sont des fichiers très volumineux. Ces fichiers contiennent une très grande quantité de points (un projet relativement grand peut se composer d'une centaine de dalles[8] contenant chacune entre un à vingt millions de points), qui eux-mêmes

[7] Type de fichier non interprétable par un éditeur de texte
[8] Ce terme désigne une partie d'un projet. L'emprise des dalles est définie par un opérateur afin que chacune d'entre elles ne soit pas trop volumineuse pour faciliter le traitement.

sont définis par de nombreux attributs (dans notre cas au nombre de 15 avec notamment les coordonnées X, Y, Z, l'intensité, la classe, l'angle…). Cela nous donne des fichiers très lourds, allant de centaines de Mégabits (Mb) à quelques Gigabits (Gb) pour une seule dalle. Pour des opérations relativement simples le temps de calcul reste raisonnable même avec un ordinateur peu puissant. Cependant, lorsque les opérations se complexifient comme avec de l'apprentissage supervisé, du machine learning ou encore la création de voxels ou le calcul de voisinage cela demande énormément de temps et de ressources. L'ordinateur principalement utilisé pour cette étude avait une mémoire vive de 12Go, ce qui est raisonnable. Cependant celle-ci a été de nombreuses fois mise à l'épreuve et a très souvent saturé pendant plusieurs dizaines de minutes, voire quelques heures pour au final stopper le calcul par manque de mémoire vive. C'est pour cette raison que les nuages sont sous-échantillonnés avant même de réaliser le moindre calcul. Dans la majorité des algorithmes, les nuages sont sous-échantillonnés à 1 million de points. Pour parler en densité de points (ce qui est plus parlant pour parler de définition d'un objet notamment), cela revient à passer d'une densité moyenne de 1500 points par mètre carré à une densité de 130 points par mètre carré en moyenne. Cela réduit drastiquement le nombre de calcul notamment quand il s'agit de voisinage d'un point dans un rayon donné. Cela affecte également la capacité de l'algorithme à détecter un objet puisque celui-ci sera moins bien défini. Lorsque c'était possible les résultats sont adaptés au nuage d'origine comme avec les poteaux. Si l'on pense à la production dans une entreprise, cette notion de temps de calcul peut ne pas poser de problèmes dans une moindre mesure puisqu'il s'agit d'un "temps machine" et qu'il ne mobilise pas d'opérateur. Et contrairement à un opérateur, un ordinateur peut réaliser des calculs en continu de jour comme de nuit. Cependant, si l'on prend un exemple concret, un algorithme qui met plusieurs heures à détecter des poteaux (avec un taux de réussite qui ne sera pas à 100 %) sur une dalle, soit 100 mètres de voirie, ce n'est pas viable puisque cette opération prendra quelques minutes à l'opérateur.

Il existe plusieurs méthodes qui n'ont pas été testées qui pourraient potentiellement améliorer ce problème comme programmer les algorithmes en C++ qui est un langage réputé comme plus performant que Python. Il serait possible également d'utiliser des calculs parallèles afin de mieux gérer les ressources à demander à l'ordinateur. Enfin, même si beaucoup d'efforts ont été dédiés à cette tâche, il est également fort probable que les algorithmes écrits pour cette étude ne soient pas optimisés. Ce travail reste en effet un travail exploratoire. Si celui-ci s'avère être intéressant, les algorithmes seraient revus par un développeur ce qui permettrait d'améliorer grandement l'optimisation de ceux-ci

II.5.3 Utilisation des algorithmes

L'objectif étant d'automatiser au maximum le processus, il est indispensable que l'algorithme soit simple d'utilisation et accessible au plus grand nombre. Pour ces raisons deux choses ont été mises en place.

Dans un premier temps les fichiers en sortie des algorithmes permettant de représenter les éléments détectés sont formatés sous forme de fichier texte avec des commandes issues de Microstation (appelées "Key-in"). Ce formatage permet d'importer directement le fichier via la barre de commande situé dans l'interface de Microstation. Ainsi pour les poteaux, en important le fichier issu de l'algorithme, des points seront tracés dans le calque courant. On utilise pour cela les commandes inscrites dans la Figure 25, la première ligne permettant d'indiquer le type d'objet à tracer, la deuxième et la troisième permettant d'indiquer les coordonnées des objets à tracer. La dernière ligne permettant de clore le tracé et de revenir au curseur de sélection sur Microstation. Pour les bordures, l'idée est similaire comme on peut le voir sur la Figure 26. La première ligne permet d'indiquer que l'on souhaite tracer un objet de type "B-spline". Cette spline débutera au premier point indiqué, se terminera au dernier point indiqué en passant par les points intermédiaires. Pour faire appel à ces fichiers sous Microstation, il suffit de taper le chemin du fichier précédé d'un "@" de la façon suivante "@C:\Projet\coordonnees_poteaux.txt" ou d'importer le fichier. L'inconvénient de cette méthode concerne le choix de l'opérateur face aux objets. Si un des objet est faux, l'utilisateur ne pourra pas décider de supprimer ou non celui-ci. Les commandes "Key-in" ne permettent pas de réaliser des choix, il faudrait pour cela réaliser des programmes en VBA[9] pour permettre ces choix à l'opérateur.

```
Place point string
xy=10,5,3
xy=4,7,2
reset
```

Figure 25 : Commandes de tracé de points sous Microstation

```
Place bspline curve points
xy=10,5,3
xy=4,7,2
xy=14.8.6
reset
```

Figure 256 : Commandes de tracé de b-spline sous Microstation

[9] Visual Basic for Applications : Langage de programmation compatible avec de nombreux logiciels

Ensuite, il a été décidé de réaliser une interface graphique pour les algorithmes proposés. Étant donné que peu de personnes sont à l'aise avec les langages de programmation et notamment Python, il semble nécessaire de réaliser une interface pour que ces algorithmes soient accessibles au plus grand nombre. Pour essayer de faciliter un maximum son utilisation l'interface a été divisée en plusieurs onglets. Chacun de ces onglets correspond à un algorithme de détection d'objet. Ces algorithmes peuvent en théorie être utilisés les uns à la suite des autres, ce qui affectera un numéro de classe aux points détectés comme l'objet recherché par l'algorithme sans affecter la classification précédente (voir II.5.4). Les algorithmes peuvent ainsi être utilisés de manière conjointe ou de manière séparée selon les besoins. Des captures d'écrans des interfaces graphiques réalisées se situent en **Annexe 10**. Ces interfaces graphiques ont été réalisées à l'aide de QtDesigner et PyQt[10]. Chaque onglet comprend les entrées et sorties des algorithmes, tandis que le dernier onglet comprend des notes et des détails sur le fonctionnement de l'interface.

II.5.4 Les codes de classification

Les numéros pour la classification (appelés également codes) sont soumis à une normalisation pour les fichiers *.las*. Cette normalisation a été fixée par l'ASPRS[11] depuis 2011 pour la version 1.4 du format de fichier. Bien qu'elle soit plus adaptée aux levés aériens, cette classification est utilisée par le service Topographie/Lidar de Geofit Expert. Ils classent leurs points à l'aide d'une macro réalisée sous Terrascan permettant de segmenter le sol et différentes couches du sursol en fonction de l'altitude. Le Tableau 3 ci-dessous présente un aperçu de cette normalisation.

Afin de ne pas perdre ces informations, les algorithmes créés garderont la classification pour les points qui ne sont pas concernés par la détection. Pour les points concernés ils se verront affectés d'un numéro de classe supérieur à 64 selon l'objet. Ces numéros sont réservés à l'utilisateur.

Classification Value	Meaning
0	Never Classified
1	Unclassified
2	Ground
3	Low vegetation
4	Medium vegetation
5	High vegetation
6	Building
…	…
64-255	User definable

Tableau 3 : Standard de classification ASPRS

[10] Outils permettant de créer des interfaces graphiques et de traduire celles-ci en langage Python
[11] American Society for Photogrammetry and Remote Sensing : https://www.asprs.org/

III Analyse des résultats

Après avoir écrit plusieurs algorithmes pour détecter des objets dans un nuage de points, cette partie proposera plusieurs indicateurs statistiques qui permettront de qualifier la qualité et la viabilité des algorithmes et de leurs résultats. Ces indicateurs permettront notamment de savoir si ces algorithmes peuvent être utilisés régulièrement pour faciliter le dessin des plans. Les indicateurs statistiques seront choisis en fonction du type d'objet détecté.

Les algorithmes ont été testés dans un premier temps sur un nuage de points correspondant à une dalle. Cette dalle fait partie d'un levé de commune au tissu urbain relativement dense composé principalement de maisons entourées de jardins ou de cours. Un visuel de cette dalle est disponible en **Annexe 11**.

III.1 Résultats de la détection de poteaux

Pour cet algorithme on proposera trois types d'indicateurs différents. Dans un premier temps un indicateur sur chaque poteau détecté sera proposé. Cet indicateur sera calculé en faisant le rapport entre le nombre de points qui définissent le poteau dans le nuage d'origine et le nombre de points contenus dans le cluster détecté. Ensuite, un indicateur calculera le pourcentage de poteaux détectés sur le nombre d'objets présents dans le nuage. Enfin on présentera les différences en planimétrie et en altimétrie entre les points définis par l'algorithme et les points qui auraient été définis par un opérateur.

III.1.1 Qualité de segmentation individuelle

Cet indicateur permet donc de qualifier la segmentation après avoir récupéré les points sur le nuage d'origine. Une première segmentation manuelle a été créée sur le nuage en créant un groupe pour chaque poteau, puis la même opération a été réalisée sur les poteaux détectés par l'algorithme en séparant chaque cluster. Le logiciel CloudCompare nous propose ensuite le nombre de points compris dans chaque nuage. Ces nombres ont été retranscrits dans le Tableau 4. On a ensuite calculé le pourcentage (visible dans le Tableau 4 et sur l'histogramme en Figure 27) entre les deux valeurs pour avoir une comparaison.

N° cluster	Nbre de pts nuage origine	Nbre de pts par cluster détecté	Pourcentage (%)
1	1407	1121	79.7
2	394	304	77.2
3	627	681	108.6
4	1433	1536	107.2
5	352	397	112.8
6	818	808	98.8
7	901	994	110.3
8	216	x	x
9	333	350	105.1
10	738	814	110.3
11	1412	1550	109.8
12	788	832	105.6
13	361	x	x
14	840	958	114.0
15	1607	1658	103.2

Tableau 4 : Segmentation individuelle des poteaux

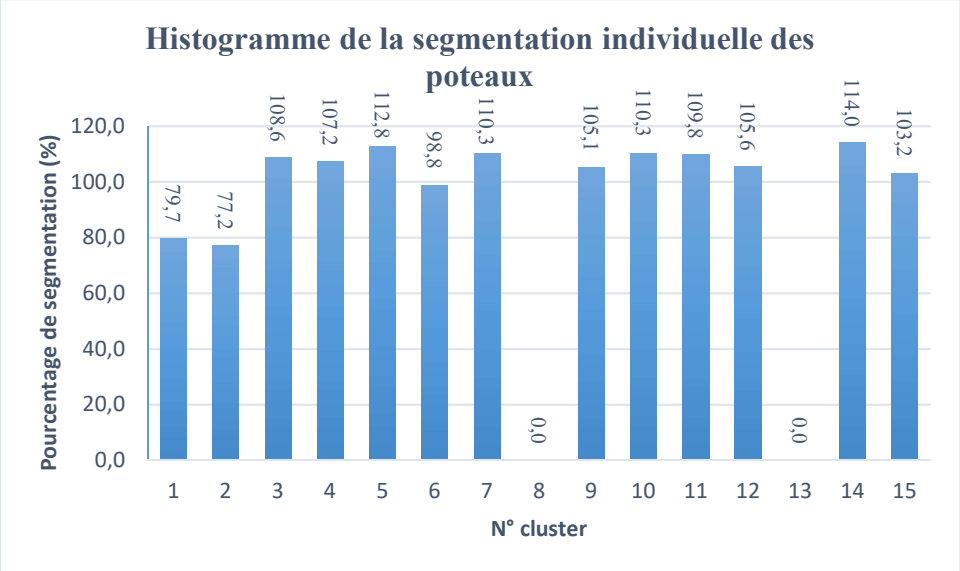


Figure 26 : Histogramme de segmentation individuelle

Il y a plusieurs détails que l'on peut noter à partir de ce tableau et de cet histogramme. On peut dans un premier temps voir que les poteaux sont définis avec un nombre de points qui varie beaucoup. Cela dépend notamment du type de poteau, de son placement par rapport à la route, mais il peut également y avoir des masques qui obstruent une partie des visées. Le second

détail à noter est le fait que certains poteaux ne sont pas détectés (notés par un "x" dans le tableau) par l'algorithme, nous en parlerons dans la partie suivante III.1.2. On peut ensuite regarder le pourcentage qui indique la qualité de segmentation de chaque objet. Les résultats montrent que la segmentation est correcte à ± 20 %. Comme il a été dit précédemment, nous ne cherchons pas un algorithme qui permet une segmentation fiable puisque nous travaillons en deux dimensions sur le plan par la suite, cependant une meilleure segmentation permettra en théorie une représentation en deux dimensions plus précise. Ce point sera vu plus en détail dans la partie III.1.3. Finalement, on peut remarquer que certains poteaux sont sur-segmentés par l'algorithme (pourcentage supérieur à 100%). Comme il a été précisé dans la partie II.2.6, il s'agit des points proches des clusters détectés mais qui ne font pas partie des objets comme des câbles électriques, de la végétation, des murets. Cette sur-segmentation n'est pas gênante dans la majorité des cas puisqu'elle est symétrique par rapport à l'axe de l'objet, elle va donc induire une erreur minime sur la digitalisation.

III.1.2 Pourcentage d'objets détectés

Cet indicateur est sans doute le plus important, il permet simplement de comparer le nombre d'objets présents dans le nuage et le nombre d'objets détectés par l'algorithme. En se basant sur le type de poteau que l'on souhaitait détecter, on compte 15 objets présents dans la dalle observée et 13 d'entre eux ont été détectés par l'algorithme. Ce qui nous donne un taux de réussite de 86.7 %. La raison pour laquelle les deux objets non détectés ne sont pas pris en compte est assez simple, On peut voir sur la Figure 28, dans le premier cas le poteau est entouré de végétation et sa section linéaire est très réduite, ce qui empêche le descripteur local de bien détecter l'objet. Dans le second cas l'objet est très proche d'un muret avec de la végétation de sa base jusqu'au deux tiers de sa hauteur et des câbles passent à son sommet ce qui rend également sa détection difficile pour le descripteur local. De plus, ces deux objets sont définis par relativement peu de points comparés à certains autres objets comme on peut le voir dans le Tableau 5.

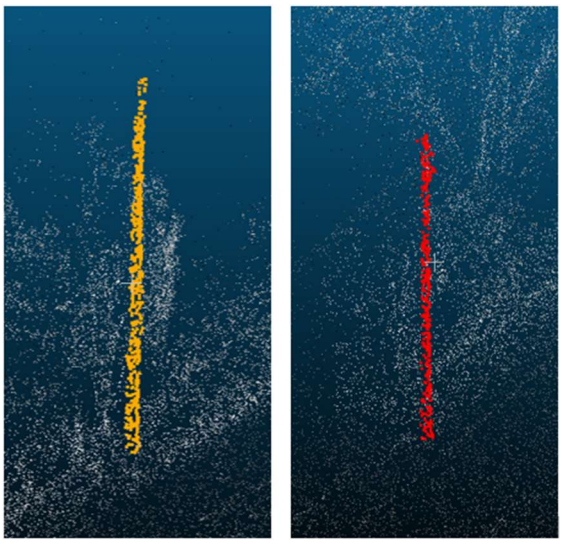


Figure 27 : Objets non détectés (en orange et rouge) par l'algorithme

Ces problèmes de détection seront développés dans la conclusion mais on estime que le taux de réussite est tout de même satisfaisant. Pour confirmer ce taux de réussite, l'algorithme a été testé sur d'autres dalles similaires et les résultats se situent entre 80 et 90 % pour les dalles testées pour des raisons similaires à celles explicitées au-dessus.

III.1.3 Précision planimétrique et altimétrique

Comme il a été expliqué précédemment, l'objectif final est de représenter l'objet par un point sur le plan. Pour tester la qualité et la précision de l'algorithme, on a créé les points représentant les objets manuellement, avec la même méthode qu'utilisent les opérateurs pour dessiner. Puis on a simplement calculé les différences en planimétrie et en altimétrie entre ces points placés manuellement et les deux jeux de points, correspondant aux deux méthodes explicitées en partie II.2.6, créés par l'algorithme. Les résultats se trouvent dans le Tableau 5. Un tableau plus complet est disponible en **Annexe 12**.

Méthode	Donnée	ΔX (m)	ΔY (m)	ΔZ (m)	ΔXY (m)
1	Moyenne	0.084	0.065	0.064	0.12
	Min / max	0.007 / 0.331	0.002 / 0.158	0.017 / 0.134	0.026 / 0.334
	Écart-type	0.099	0.046	0.037	0.091
2	Moyenne	0.04	0.068	0.042	0.082
	Min / max	0.004 / 0.144	0.006 / 0.167	0.000 / 0.094	0.007 / 0.221
	Écart-type	0.043	0.051	0.027	0.062

Tableau 5 : Comparaison des deux méthodes de vectorisation

En analysant les résultats on voit que la seconde méthode est globalement plus précise, aussi bien au niveau des moyennes que des écart-types. Ce qui semble logique, puisque la méthode 1 suppose que les poteaux sont parfaitement verticaux ce qui en réalité n'est pas le cas. On retrouve tout de même dans les deux méthodes des points qui s'écartent plus que les autres de leur position supposée. Sur les 13 points pris en exemple, 2 points (soit 15 %) ont une erreur supérieure à 15 centimètres en planimétrie. En revanche, 100 % des points testés ont une erreur en altimétrie inférieure à 10 centimètres. Ce nombre est à relativiser, puisqu'il arrive très régulièrement que les poteaux soient en dehors de l'emprise de la route et que leur base et le sol autour d'eux ne soit pas mesuré. De plus, avec le bruit qu'il peut y avoir au niveau du sol en raison du "matching", et au vu de la façon dont ont été créés les points, ceux-ci se situent à plus de 90 % en dessous des points placés par l'opérateur. On peut également analyser les écart-types, indicateur de la dispersion des résultats. Ceux-ci sont cohérents avec les résultats. En

planimétrie, on obtient un écart-type de 6 centimètres, ce qui est assez important. Cette dispersion est induite par plusieurs points avec un écart supérieur à 10 centimètres. Ces résultats ne sont pas suffisants pour la confection d'un plan topographique en revanche ils restent acceptables pour valider la précision de classe A exigée par un PCRS [1].

Pour conclure sur ces indicateurs, entre les erreurs de détection et les erreurs de position, on estime que 73 % des points testés sont placés à moins de 10 centimètres en planimétrie et en altimétrie de leur vraie position. Ce résultat est satisfaisant mais pas suffisant pour pouvoir automatiser de façon systématique ce processus.

III.1.4 Remarques générales sur l'algorithme

Ce premier algorithme pour la détection de poteaux donne des résultats plutôt satisfaisants. Même si les chiffres ne sont pas optimums et ne suffisent pas pour être exploités dans une production, au vu des raisons pour lesquelles il y a des erreurs cela semble logique et un opérateur est obligé d'être très attentif pour ne pas louper certains éléments comme certains poteaux entourés de végétation par exemple. Autant dire qu'avec ce type d'exemple il est très complexe d'automatiser ce type de problème. L'ajout d'autres attributs a tenté d'être ajouté à l'algorithme pour renforcer la détection. Un travail sur la différence d'intensité a notamment été développé sans que cela soit concluant.

Dans cette analyse, nous nous sommes focalisés sur des poteaux de diamètre relativement gros comme les poteaux électriques etc… L'algorithme a également été testé avec des éléments plus fins comme les panneaux de signalisation en changeant le rayon de détection. On retrouve globalement les mêmes résultats. On retrouve également les mêmes problèmes avec la proximité d'autres objets qui rend difficile la détection.

L'algorithme implémenté dans l'interface graphique ne prend en compte qu'une seule dalle. Un test a été réalisé en ajoutant une boucle générale sur l'algorithme afin de gérer plusieurs dalles voire tout un projet. Avec cet algorithme il est impensable de pouvoir gérer tout le projet simultanément. Après un test avec cette nouvelle boucle, l'algorithme a mis environ 24 heures pour réaliser ses calculs sur la moitié d'un projet (environ 50 dalles). On en revient donc au point vu au chapitre II.5.2, à savoir la gestion du temps machine. Ce temps de calcul est bien trop long au vu de l'efficacité de l'algorithme.

Un des points positifs de cet algorithme reste tout de même le fait qu'il ne prenne en compte que très peu, voire aucun, autres objets comme des troncs d'arbres par exemple. Sur les quelques dalles testées, on compte une cinquantaine de poteaux et seulement deux troncs d'arbres ont été détectés par l'algorithme. Cependant cela dépend beaucoup du choix du rayon en paramètre d'entrée de l'algorithme. L'autre point intéressant de cet algorithme c'est qu'il s'adapte très bien à d'autres environnements également, en particulier les levés de voies de chemin de fer où les poteaux sont isolés des autres éléments du nuage.

III.2 Résultats de la détection de bordures

Pour cet algorithme, nous avons choisi de tester de trois façons différentes les éléments créés, à savoir la ligne de fil d'eau et l'arête du trottoir. En premier lieu aura lieu un simple test sur le pourcentage de linéaire détecté. Ensuite sera testé les coordonnées des points créés en les comparants aux positions qu'ils auraient dû occuper. Les seconds points sont choisis arbitrairement et seront comme projetés sur la ligne définie par le fil d'eau et l'arrête du trottoir. Enfin, nous dessinerons les lignes (objet de type B-spline) définies par les points créés et les positions de profils équidistants seront comparés. Finalement des indicateurs statistiques seront étudiés en fonction des résultats et une réflexion sur la façon de faire sera proposée.

Cet algorithme a été testé dans un premier temps sur la même dalle que pour la détection de poteau. Un aperçu de cette dalle est disponible en **Annexe 11**.

III.2.1 Qualité de détection du linéaire

Dans ce premier test, on calcule simplement le pourcentage du linéaire détecté. Ayant procédé à un sous-échantillonnage et n'ayant pas la nécessité de prendre en compte tous les points pour définir correctement la bordure de trottoir, on considère que la détection d'un point tous les cinquante centimètres sur le linéaire sera amplement suffisant. En prenant ces critères, on estime à plus de 80% la détection du linéaire sur la dalle testée et cela pour les deux côtés de la route. Cette donnée est visible sur l'**Annexe 15**. De plus on remarque que les points forment des profils d'une dizaine de centimètres de large, ce qui correspond aux points de la trajectoire pris en compte. Les endroits correspondants aux 20 % non détectés sont, situés soit au niveau des abaissements de trottoir où la différence d'altitude entre la route et le trottoir n'est pas assez marquée, ou encore au niveau d'une voiture qui est soit sur la route, soit garée sur le trottoir ce qui induit une différence d'altitude trop élevée pour être retenu par l'algorithme.

III.2.2 Comparaison des coordonnées des points

Pour ce comparatif nous nous appuierons sur les moyennes et les écart-types obtenues sur les deux côtés de la route, mais des résultats plus complets sont disponibles en **Annexe 13**. Le Tableau 6 propose un récapitulatif des moyennes des écarts en planimétrie et en altimétrie des points du fil d'eau et des points de l'arête du trottoir. À titre indicatif pour cette dalle, le trottoir mesure environ 160 mètres de long et l'algorithme a créé 22 points de chaque côté pour les fils d'eau et les arêtes (soit un point tous les sept mètres environ). Ce premier indicateur nous permet de savoir si les points sont globalement bien placés et de voir le nombre d'erreur et la précision approximative de l'algorithme.

Côté	**Gauche**				**Droit**			
Linéaire	**Fil d'eau**		**Arête**		**Fil d'eau**		**Arête**	
	ΔXY (cm)	**ΔZ (cm)**	**ΔXY (cm)**	**ΔZ (cm)**	**ΔXY (cm)**	**ΔZ (cm)**	**ΔXY (cm)**	**ΔZ (cm)**
Moyenne	2.8	0.8	2.6	1.3	9.2	3.3	7.4	4.0
Min / Max	0.8 / 6.3	0.1 / 2.8	0.5 / 5.9	0.1 / 4.9	0.0 / 9.7	0.0 / 1.5	0.0 / 86.5	0.0 / 43.4
Ecart-type	1.7	0.6	1.6	1.3	20.1	10.8	18.1	9.2

Tableau 6 : Comparaison des coordonnées du fil d'eau et de l'arrête entre la position vraie et l'algorithme

En analysant les données plus précisément, on remarque sur le côté gauche qu'aucun point n'a un écart en planimétrie ou en altimétrie excédant 10 centimètres que ce soit pour le fil d'eau ou l'arête. En revanche, sur le côté droit 2 points sur le fil d'eau (soit 9 %) excèdent un écart de 10 centimètres. L'un en planimétrie ainsi qu'en altimétrie avec un écart d'un mètre en planimétrie et l'autre seulement en planimétrie. Ces erreurs s'expliquent simplement puisque le premier point est situé sur une voiture garée sur le trottoir et l'autre point est situé à l'intérieur d'un avaloir. Pour l'arête à droite on retrouve des erreurs liées aux mêmes points, ce qui montre une faiblesse de l'algorithme. En effet, la détermination des arêtes est directement liée au fil d'eau, si celui-ci est mal détecté, c'est le couple de points qui sera mal détecté. En supprimant le point aberrant situé sur la voiture, les moyennes baissent fortement (ΔXY=4.7 cm ; ΔZ=1.0 cm et ΔXY=3.4 cm ; ΔZ=2.1 cm). Les erreurs approchant les 10 centimètres quant à elles peuvent s'expliquer soit par une détection pas assez précise, soit par des objets comme des feuilles présentes dans le caniveau. On peut également ajouter à cela l'incertitude lié au bruit du "matching". Concernant les écart-types, ceux-ci montrent une dispersion relativement faible ce qui est plutôt rassurant. De même que pour les moyennes, on a supprimé le point aberrant du calcul ce qui donne de meilleurs résultats sur le côté droit (ΔXY=4.0 cm ; ΔZ=4.3 cm et

ΔXY=4.2 cm ; ΔZ=1.5 cm). À noter également que l'erreur sur l'axe Y est globalement plus élevée car, sur cette dalle, la route est presque parallèle à l'axe X.

Avec ces résultats on se rend bien compte que l'interpolation et le jugement de l'opérateur est indispensable pour obtenir une meilleure précision. Notamment sur les points qui peuvent être aberrants, si des objets sont présents au niveau du fil d'eau l'opérateur va approximer l'intersection entre la route et la bordure, ou encore pour le bruit au niveau du sol où il réalisera une moyenne visuellement. Néanmoins les résultats sont globalement satisfaisants avec une précision que l'on peut estimer à moins de 5 centimètres en éliminant les points aberrants.

III.2.3 Comparaison des profils

Pour cet indicateur, l'idée dans un premier temps est de confirmer le choix de l'objet de type "B-spline" sur Microstation pour le tracé des lignes représentant les fils d'eau et les arêtes. On rappelle que ce type a été choisi pour son adaptation entre les portions droites, légèrement courbes voire courbe. Ensuite la seconde idée derrière cet indicateur est de voir si les lignes représentent bien les objets sur toute leur longueur. Pour cela nous avons tracé dans un premier temps la ligne tel que l'aurait tracé un dessinateur, puis nous avons tracé la "B-spline" avec les points retenus pour chaque linéaire. Dans cet indicateur nous n'avons pas pris en compte les points faux. Ceux-ci engendrent une erreur bien trop importante sur la longueur pour pouvoir les prendre en compte. Ensuite des sections ont été créées tous les cinq mètres et les coordonnées du vecteur entre les deux lignes ont été notées. Cet indicateur n'a pas été choisi au hasard puisqu'il est très fréquent voire toujours demandé de représenter ce type de plan à l'aide de profils équidistants.

Les tableaux complets sont à retrouver en **Annexe 14**. Le Tableau 7 quant à lui reprend les moyennes et les écart-types des valeurs notées dans ces tableaux. La longueur des linéaires étant d'environ 160 mètres, nous nous sommes basés sur 30 sections pour obtenir ces résultats.

Côté	Gauche				Droit			
Linéaire	**Fil d'eau**		**Arrête**		**Fil d'eau**		**Arrête**	
	ΔXY (cm)	**ΔZ (cm)**	**ΔXY (cm)**	**ΔZ (cm)**	**ΔXY (cm)**	**ΔZ (cm)**	**ΔXY (cm)**	**ΔZ (cm)**
Moyenne	2.1	1.9	2.9	2.5	6.3	0.0	3.4	0.3
Min / Max	0.1 / 6.7	0.0 / 6.5	0.6 / 6.9	0.0 / 8.2	0.1 / 18.1	0.0 / 0.0	0.1 / 12.0	0.0 / 4.8
Ecart-type	2.0	1.6	1.6	2.5	4.4	0.0	3.0	1.0

Tableau 7 : Comparaison des sections de linéaires

Si l'on prend les valeurs indiquées dans le Tableau 7, les valeurs pour le côté gauche sont plutôt satisfaisantes. En revanche, pour le côté droit les écarts sont un peu trop élevés notamment sur le fil d'eau en planimétrie. En analysant de plus près les valeurs une à une, on remarque sur le côté gauche que plusieurs points ont des écarts supérieurs à 5 centimètres, et que les écarts se situent sur des profils adjacents. En effet il suffit qu'un point soit mal placé à la base pour reporter un écart de plusieurs centimètres sur plusieurs mètres. Ainsi, on peut remarquer pour le fil d'eau à droite, un écart de plus de 10 centimètres sur le profil 16 qui se répercute sur les profils 15 et 17 par exemple. On remarque également, comme pour l'indicateur précédent, que les écarts importants sur le fil d'eau se retrouvent en grande partie sur les profils des arêtes. Les résultats sur les écart-types sont quant à eux similaires à ceux trouvés dans la partie II.2.2.

Pour en revenir sur le choix d'objet "B-spline" pour linéariser ces éléments, il semble plutôt adapté. En effet d'autres primitives comme des arcs de cercle ou des droites ont été testés. Ceux-ci étaient assez complexes à mettre en place et donnaient des résultats bien moins concluants. L'inconvénient en revanche de la "B-spline" se trouve au niveau des points légèrement mal placés puisque l'objet va accentuer les écarts et les répartir sur la longueur.

III.2.4 Remarques générales sur l'algorithme

Malgré des résultats mitigés, cet algorithme de détection de bordure reste intéressant mais ne semble pas assez robuste pour être utilisé en production, notamment de plans topographiques. En plus des erreurs de détection, la part d'interprétation nécessaire est bien trop importante, notamment lorsque l'on prend l'exemple d'un trottoir abîmé, d'un objet présent au niveau du fil d'eau, du bruit dans le nuage… Cependant l'utilisation de cet algorithme peut rester intéressante pour obtenir une ligne guide par exemple pour la digitalisation avec certains outils de Topodot. Mais de la même manière que pour l'algorithme précédent, en théorie, les résultats obtenues sont suffisants pour avoir une précision de classe A d'un PCRS [1].

Un point non abordé, mais qui reste très important dans la digitalisation, est la gestion des éléments présents sur le trottoir comme les abaissements, les grilles, les avaloirs… Une tentative a été faite pour essayer de détecter les abaissements. Ceux-ci se différencient avec le changement d'altitude suivant une pente régulière mais cette pente peut-être similaire à celle de portions de trottoir donc difficilement détectable et la différence d'altitude est de quelques centimètres ce qui rend également difficile la détection. On ajoute à cela les trottoirs qui peuvent être abîmés et qui peuvent être confondus avec un abaissement et cela rend très complexe la détection de ces éléments.

L'autre point important d'un linéaire est le début et la fin de la ligne. La façon dont a été programmé l'algorithme avec la trajectoire, induit que les linéaires commencent légèrement après et finissent légèrement avant le début de la dalle. On suppose que cela provient du détecteur local. En effet, le voisinage autour des premiers et derniers points étant moins nombreux (à cause du "vide" en bout de dalle), on suppose que la première classification à cet endroit est moins bonne.

Finalement, de la même manière que pour le premier algorithme, celui-ci ne gère qu'une dalle à la fois. Une boucle serait donc à implémenter pour gérer plusieurs dalles à la fois.

III.3 Bilan et autres remarques

Pour faire un bilan bref de ces algorithmes, ceux-ci ne sont pas encore assez robustes pour être utilisés en production de plans topographique et le fait de devoir sous-échantillonner le nuage reste un problème pour la précision obtenue en sortie. Les résultats sont tout de même plus constants que les méthodes logicielles présentées dans la partie I. De plus, la problématique de digitalisation reste très complexe à aborder de manière automatique notamment au niveau des linéaires. Un aperçu de ce que donne la classification en sortie de ces algorithmes est disponible en **Annexe 15**, ainsi qu'un aperçu de ce que peut donner la digitalisation en **Annexe 16**. En revanche, les résultats obtenus restent intéressants pour la précision demandée pour un PCRS, ou encore pour faire un inventaire de mobilier urbain par exemple.

Conclusion et ouverture

À travers cette étude, une première approche sur la détection d'objets dans un nuage de points Lidar a pu être réalisée. Elle a notamment permis de constater, par le biais d'exemples, les possibilités et les limites que peut offrir une telle application. Notre objectif initial était, à partir d'un nuage de points obtenu par MMS, de détecter automatiquement des objets ou des éléments distinctifs de ce nuage afin de les représenter de façon vectoriel en deux dimensions pour faciliter la confection de plans topographiques.

Pour cela, les poteaux et les bordures de trottoir, qui sont des éléments récurrents des plans topographiques, ont été choisis comme exemple avec pour chacun de ces objets une méthodologie similaire mais une approche différente. Après un nettoyage du nuage et une segmentation du sol dans un premier temps, on a utilisé des descripteurs locaux basés sur les caractéristiques principales des objets pour les détecter. À savoir pour les poteaux un descripteur basé sur leur forme verticale (présenté en partie II.2.2), et un descripteur basé sur la différence locale d'altitude dans le second cas (présenté en partie II.3.2). Ces descripteurs permettent une première classification qui est confrontée à une homogénéisation (dans les parties II.2.3 et II.3.3). Une clusterisation permet ensuite de calculer un descripteur global basé sur d'autres caractéristiques des objets (ceux-ci étant présentés en partie II.2.5 et II.3.6). Ces méthodes, bien que les résultats soient imparfaits, se sont montrées plus efficaces et robustes que celles présentes dans les logiciels existants. Ces erreurs sont notamment dues à la proximité des objets entres eux. Le temps de calcul en revanche est similaire voire plus long que les logiciels selon les cas.

Du point de vue de la vectorisation, plusieurs méthodes ont été testées. Pour les éléments représentés ponctuellement, comme les poteaux, une méthode se rapprochant de la méthode manuelle a été préférée. En simulant une section située à un mètre au-dessus du sol et en prenant la moyenne en planimétrie on obtient un point assez représentatif de l'objet (méthode expliquée dans la partie II.2.6). On y ajoute ensuite l'altitude du sol proche de ces coordonnées. Concernant les linéaires, la méthode choisie a été de discrétiser l'objet en créant des points et en les reliant avec une primitive géométrique disponible dans le logiciel utilisé (détails expliqués dans la partie II.3.7). Cela comporte l'inconvénient de propager les erreurs si certains points sont faux. Ces méthodes ont des résultats discutables et dépendent directement de la

partie segmentation de l'algorithme. Ceux-ci sont globalement satisfaisants pour les objets ponctuels mais restent à améliorer pour les linéaires.

Ces algorithmes sont loin d'être parfaits que ce soit au niveau des résultats ou des temps de calcul. Le géomètre ayant l'obligation de garantir la précision de ses rendus, ils sont difficilement exploitables dans une production à grande échelle et ne rentrent pas dans la classe de précision d'un plan topographique. On se rend également bien compte que la part d'interprétation d'un opérateur est indispensable encore aujourd'hui pour bien analyser la donnée. De nombreuses améliorations seraient à apporter pour permettre de gérer les erreurs par exemple, d'avoir plus de contrôle sur les résultats ou encore de déterminer de quel type d'objet précisément il s'agit (par exemple différencier un panneau de signalisation d'un poteau électrique). En revanche, cette étude reste intéressante à d'autres points de vue.

Le développement de l'intelligence artificielle, du machine learning et des ressources informatiques en général permettront surement de trouver une solution automatique efficace à cette problématique dans un avenir proche. On peut notamment penser à de la reconnaissance d'objets ou de formes avec une bibliothèque d'objets déjà modélisés par exemple. Cela permettrait de reconnaitre un type d'objet particulier et de façon plus précise. Ces méthodes se développent de plus en plus pour les images, notamment dans le domaine de la robotique, de la téléphonie (reconnaissance faciale), ou encore de l'automobile (conduite autonome). Ces progrès laissent à penser que ces méthodes pourront se décliner sur les nuages de points dans un avenir proche et automatiser un bon nombre de procédés. Il y a notamment des travaux qui sont développés actuellement proposant de détecter les objets sur les photos prises au moment de l'acquisition grâce au Deep learning, pour ensuite les localiser dans le nuage de points.

Bibliographie

Documents et articles :

[1] Référentiel du PCRS rédigé par le Conseil National de l'Information Géographique (CNIG). Version du 21 Septembre 2017. Disponible sur : http://cnig.gouv.fr/

[2] LANDES T. et al, *Les principes fondamentaux de la lasergrammétrie terrestre: acquisition, traitement des données et applications (partie 2/2)*, Revue XYZ, 4e trimestre 2011, N°129

[3] BOULAASSAL H. *Segmentation et modélisation géométriques de façades de bâtiments à partir de relevés laser terrestres*, thèse présenté pour le titre de docteur spécialisé en Topographie-Géomatique, Université de Strasbourg, soutenu le 03/02/10

[4] MARCHAND M. *Optimisation du traitement de nuage de points pour la production de plan de façade au sein d'un cabinet de géomètre-expert*, mémoire présenté en vue d'obtenir le diplôme d'ingénieur CNAM spécialisé Géomètre et Topographe, ESGT, Juillet 2018, Partie I.3.1.5

[5] KAARTINEN H. et al, *Mobile Mapping – Road Environment Mapping using Mobile Laser Scanning*, Official Publication N062, EuroSDR, Mars 2013

[6] BEUCHER S. et al, *The Morphological approach to segmentation: The watershed algorithm*, Janvier 1993

[7] DI ANGELO L. et al, *An efficient algorithm for the nearest neighbourhood search for point clouds*, International Journal of Computer Science Issues, Septembre 2011, Vol. 8

[8] HAN X-F. et al, *A comprehensive review of 3D point cloud descriptors*, Tianjin University, Février 2018

[9] RUSU R B. et al, *Persistent Point Feature Histograms for 3D Point Clouds*, 2008

[10] JOHNSON A E. et al, *Surface Matching for Object Recognition in Complex 3D Scenes*, 1998

[11] TOMBARI F. et al, *Automatic Extraction of Pole-like Objects Using Point Cloud Library*, GIM International, Octobre 2014

[12] GEBHARDT S. et al, *Polygons, Point Clouds, and Voxels, a Comparison of High-Fidelity Terrain Representations*, extrait issu de *Simulation Interoperability Workshop and Special Workshop on Reuse of Environmental Data for Simulation Processes, Standards and Lessons Learned*, 2009

[13] TOMBARI F. et al, *3D Data Segmentation by Local Classification and Markov Random Fields*, May 2011

[14] YANG B. *Two-step adaptive extraction method for ground points and breaklines from lidar point clouds*, article issu de *ISPRS Journal of Photogrammetry and Remote Sensing*, 2016

Sites Web :

- Documentation en ligne de la suite Terrasolid : http://www.terrasolid.com/home.php , consulté en Février et Mars 2019
- Explications de fonctions de la suite Terrasolid : https://geocue.com/, consulté en Février et Mars 2019
- Documentation en ligne du logiciel Topodot : https://new.certainty3d.com/, consulté de Février à Juin 2019
- Documentation en ligne du logiciel CloudCompare: https://www.danielgm.net/cc/, consulté de Mars à Juin 2019
- Hébergement et documentation des bibliothèques Python: https://anaconda.org/, consulté de Mars à Juin 2019

Tables des annexes

Annexe 1

Détails sur les bibliothèques et les fonctions Python utilisées

Pour l'écriture des scripts Python, de nombreuses bibliothèques et fonctions déjà programmées ont été utilisées. Ci-dessous se trouve un peu plus de détails sur ces bibliothèques et ces fonctions.

- Numpy : Cette bibliothèque comprend de nombreuses fonctions pour réaliser du calcul scientifique, notamment des fonctions pour gérer des tableaux et des matrices à n dimensions. Elle comprend également un module d'algèbre linéaire avec le module "linalg", un module de gestion de nombres aléatoires avec "random"… Ce sont principalement les trois modules cités ci-dessus qui ont été utilisés dans les algorithmes. Plus de détails sur les modules et les fonctions sont disponible sur le site : https://www.numpy.org/ .

- Matplotlib : Il s'agit d'une bibliothèque de tracé. Elle permet donc de réaliser de nombreux graphiques, que ce soit des courbes, des histogrammes et bien d'autres objets notamment issus de la bibliothèque Numpy. Celle-ci a été utilisée pour obtenir des visualisations rapides des résultats calculés notamment dans le premier algorithme de cette étude. Plus d'informations sur cette bibliothèque sont disponibles sur le site : https://matplotlib.org/ .

- Pandas : Cette bibliothèque permet la manipulation et l'analyse de données. Elle comprend des fonctions et des objets similaires à Numpy. Cependant elle a été utilisée car contrairement à Python, sa fonction de création de tableau est compatible avec la bibliothèque de fichiers *.las* Pyntcloud. Plus de détails sur les fonctions incluses dans Pandas sur le site : https://pandas.pydata.org/ .

- Scikit-Learn : Cette bibliothèque de fonction est un outil relativement simple pour accéder à des fonctions de machine learning. Elle comprend plusieurs modules. Ceux qui ont été utilisés pour cette étude sont les modules de "Clustering" avec la fonction Density Based Spatial Clustering of Applications with Noise (DBSCAN) et de

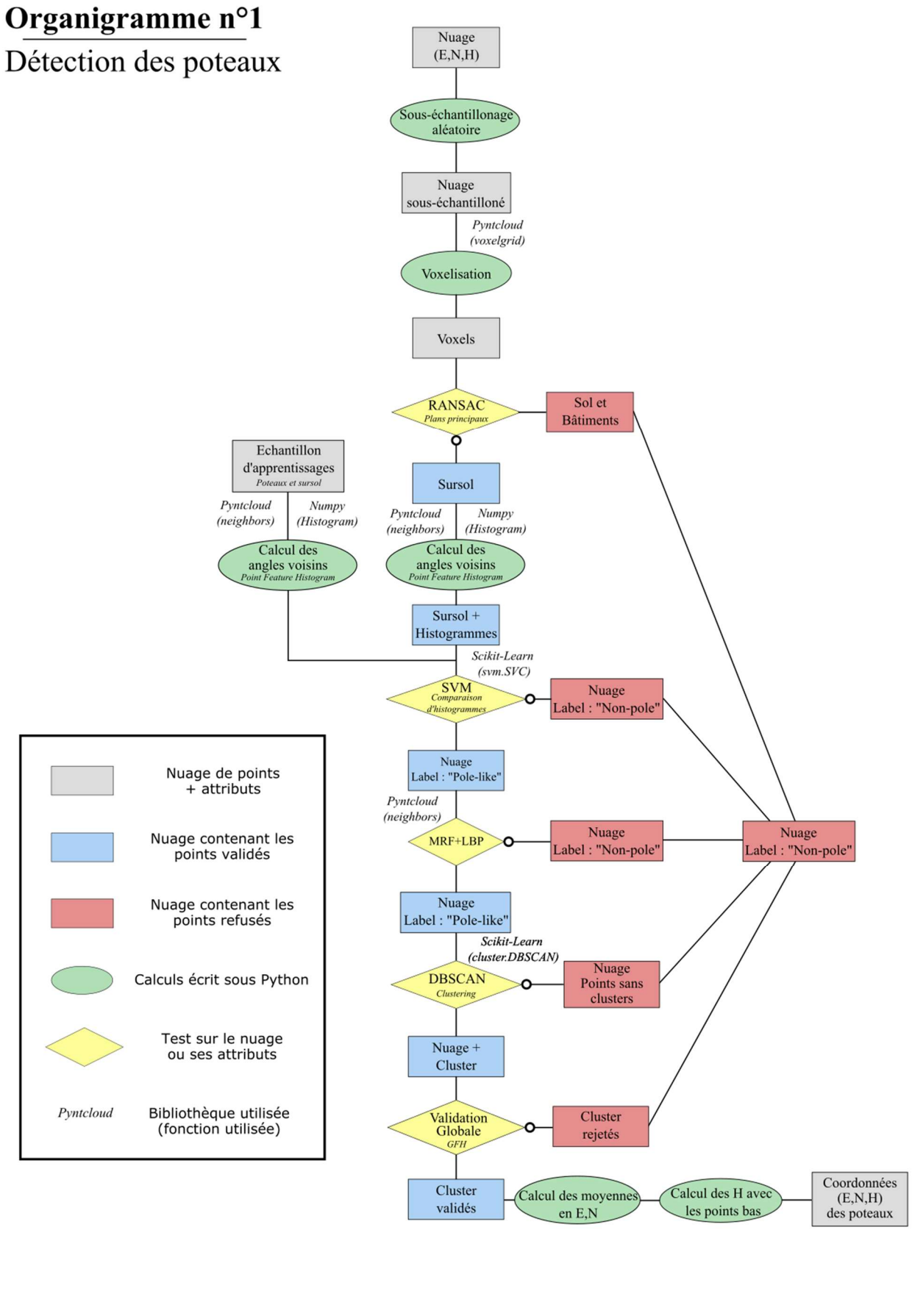
"Classification" avec l'algorithme de Support Vector Machine (SVM). On peut retrouver plus d'informations et de détails de fonctionnement sur les pages suivantes:

- SVM : https://scikit-learn.org/stable/modules/svm.html#svm-classification
- DBSCAN : https://scikit-learn.org/stable/modules/clustering.html#dbscan

- LasPy : Cette bibliothèque a été développée pour lire, modifier et créer des fichiers *.las* ou *.laz* (forme compressée du format *.las*). On peut très facilement obtenir les données comprises dans les fichiers *.las* sous forme de tableau Numpy à l'aide de cette bibliothèque. Étant donné que le format *.las* est un format binaire et donc non traduisible par un éditeur de texte, cette bibliothèque est très utile. Il existe d'autres bibliothèques pour cette usage comme Liblas ou encore la Point Cloud Library. Plus de détails et des exemples sont disponibles sur le site : https://pypi.org/project/laspy/ .

- Pyntcloud : Il s'agit d'une bibliothèque de fonctions permettant de réaliser plusieurs calculs sur les nuages de points. Elle comprend notamment des outils de voxelisation, de calcul de voisinage (r-voisins ou k-voisins). Ce sont principalement ces fonctions qui ont été utilisées pour cette étude. Plus de détails sur le site de la bibliothèque et sur le GitHub qui lui est dédié : https://pyntcloud.readthedocs.io/en/latest/ / https://github.com/daavoo/pyntcloud .

Annexe 2


Organigramme n°1
Détection des poteaux
Nuage (E,N,H)
Sous-échantillonage aléatoire
Nuage sous-échantilloné
Pyntcloud (voxelgrid)
Voxelisation
Voxels
RANSAC
Plans principaux
Sol et Bâtiments
Echantillon d'apprentissages
Poteaux et sursol
Sursol
Pyntcloud (neighbors)
Numpy (Histogram)
Pyntcloud (neighbors)
Numpy (Histogram)
Calcul des angles voisins
Point Feature Histogram
Calcul des angles voisins
Point Feature Histogram
Sursol + Histogrammes
Scikit-Learn (svm.SVC)
SVM
Comparaison d'histogrammes
Nuage Label : "Non-pole"
Nuage Label : "Pole-like"
Pyntcloud (neighbors)
MRF+LBP
Nuage Label : "Non-pole"
Nuage Label : "Non-pole"
Nuage Label : "Pole-like"
Scikit-Learn (cluster.DBSCAN)
DBSCAN
Clustering
Nuage Points sans clusters
Nuage + Cluster
Validation Globale
GFH
Cluster rejetés
Cluster validés
Calcul des moyennes en E,N
Calcul des H avec les points bas
Coordonnées (E,N,H) des poteaux
Nuage de points + attributs
Nuage contenant les points validés
Nuage contenant les points refusés
Calculs écrit sous Python
Test sur le nuage ou ses attributs
Pyntcloud
Bibliothèque utilisée (fonction utilisée)

Annexe 3

Pseudo-code - Calcul des angles du voisinage des points:

```
Donnée en entrée : nuage de points correspondant au sursol avec attributs (x,y,z)
appelé "sursol"


Fonctions utilisées : "get_neighbors" de la bibliothèque Pyntcloud , "histogram"
et fonctions de manipulations de tableaux de la bibliothèque Numpy
                        ----------------------------
### Variables

Sursol = nuage
rayon mini = 0.5m
rayon maxi = 1.5m
n=nombre de points compris dans sursol

### Pseudo-code

pts voisins rayon mini = Pyntcloud.get_neighbors(sursol, rayon mini)
pts voisins rayon maxi = Pyntcloud.get_neighbors(sursol,rayon maxi)

# Définition du voisinage des points pour un rayon entre 0.5m et 1.5m

Pour i de 1 jusqu'à n avec un pas de 1:
    voisinage des points i=liste(pts voisins rayon maxi de i)-(pts voisins
    rayon mini de i)

voisins des points i = tableau (voisinage des points i)

# Calcul des angles pour chaque point voisins

angles voisinage = tableau de la même forme que voisins des points i
vertical = [0,0,1]

Pour j de 1 jusqu'à n avec un pas de 1:
    v = nombre de voisins pour le point j
    liste angles = [ ]
    Pour k de 1 jusqu'à v avec un pas de 1:
        vecteur = (coordonnées du point j) - (coordonnées du voisin k de j)
        scal = produit scalaire (vertical , vecteur)
        liste angles = liste angles + scal

    angles voisinage [j] = liste angles


# Calcul des histogrammes
histogrammes = tableau de la forme (n , 10)

Pour i de 1 jusqu'à n avec un pas de 1:
    histogrammes[i]=Numpy.histogram(angles voisinage[i],nb intervalles = 10)
```

Annexe 4

Pseudo-code - Support Vector Machine (SVM) sur les histogrammes:

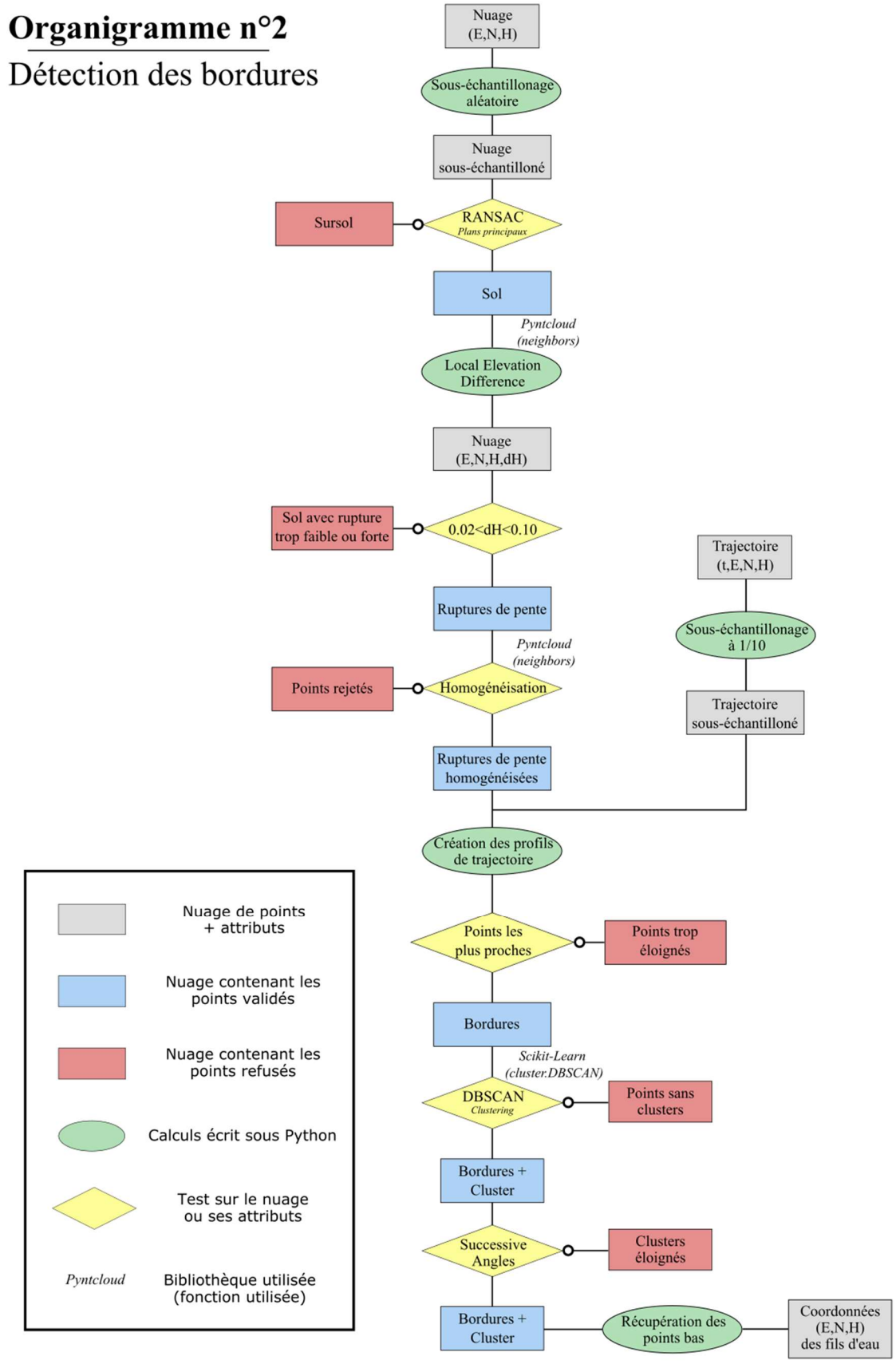
```
Donnée en entrée : histogrammes du nuage et des échantillons

Fonctions utilisées : "svm.SVC" de la bibliothèque Scikit learn, fonctions de manipulations de tableaux de Numpy

                    ----------------------------

### Variables

sursol = nuage
échantillon poteaux = échantillon d'apprentissage correspondant aux poteaux
échantillon autre = échantillon d'apprentissage correspondant au sursol sans poteaux
n poteaux = Nombre de points dans échantillon poteaux
n autre = Nombre de points dans échantillon autre
histogramme poteaux = tableau des histogrammes d'échantillon poteaux
histogramme autre = tableau des histogrammes d'échantillon autre
histogrammes = tableau des histogrammes de sursol

### Pseudo-code

# Création des labels

label poteaux = tableau de la forme (n poteaux , 1) rempli de "1"
label autre= tableau de la forme (n autre , 1) rempli de "0"
sample = concaténation des tableaux (histogramme poteaux , histogramme autre)
labels sample = concaténation des tableaux (label poteaux , label autre)

# Calcul de la fonction de prédiction

fonction SVM = svm.SVC (kernel= "linear", C=1.0)
fonction SVM.fit (sample, labels samples)

labels = fonction SVM.predict (histogrammes)

sursol labellisé = concaténation des tableaux (sursol , labels)
```

Annexe 5

Pseudo-code -Calcul du Markov Random Field:

```
Donnée en entrée : sursol labellisé
Fonctions utilisées : "get_neighbors" de la bibliothèque Pyntcloud, fonctions
de manipulations de tableaux et "lin.norm" de Numpy
                        ----------------------------

### Variables
sursol labellisé = cf annexe 5
n = nombre de points de sursol labellisé
nombre itérations = 1       # Nombre d'itérations à réaliser
itérations = 0
lambda = 1
k = 4             # Nombre de voisins à prendre en compte pour chaque point

### Pseudo-code
# Calcul des k-voisins
pts voisins = Pyntcloud.get_neighbors(sursol labellisé, k)

# boucle MRF
Pendant que itérations est inférieur à nombre itérations:
      phi i = tableau de la forme (n,1) rempli de "0"        # Datacost
      phi ij = tableau de la forme (n,4) rempli de "0"       # Smoothness
      E = tableau de la forme (n,1) rempli de "0"            # Fonction énergie

      Pour j de 1 jusqu'à n avec un pas de 1:
      |     |Si la norme de (sursol labellisé[j] - pts voisins [j][1]) est
      |     |différente de 0:
      |     |      v xi = 1 - 0.5*(norme de (sursol labellisé[j] - pts voisins
      |     |      [j][0]) / (norme de (sursol labellisé[j] - pts voisins [j][1])
      |     |Sinon:
      |     |      v xi = 0
      |     phi i[j]=lambda*(1-v xi)

            Pour m de 1 jusqu'à 4 avec un pas de 1:
            |     |Si le label de  sursol labellisé [j] est égal au label de pts
            |     |voisins [j][m]:
            |     |      phi ij [j,m] = 0
            |     |Sinon:
            |     |      phi ij[j,m] = exponentielle de (norme de (sursol labellisé
            |     |      [j] - pts voisins[j,m])

            Somme phi ij = Somme des phi ij du point j
            E[j] = phi i[j] + phi ij[j]

# Changement des labels
      Pour j de 1 jusqu'à n avec un pas de 1:
      |     Si E[j] est supérieur à 1.5:
      |     |     |Si label de sursol labellisé [j]=0:
      |     |     |      label de sursol labellisé [j]=1
      |     |     |Si label de sursol labellisé [j]=1:
      |     |     |      label de sursol labellisé [j]=0
      itérations = itérations  + 1
```

Annexe 6


Organigramme n°2
Détection des bordures
Nuage (E,N,H)
Sous-échantillonage aléatoire
Nuage sous-échantilloné
Sursol
RANSAC
Plans principaux
Sol
Pyntcloud (neighbors)
Local Elevation Difference
Nuage (E,N,H,dH)
Sol avec rupture trop faible ou forte
0.02<dH<0.10
Trajectoire (t,E,N,H)
Sous-échantillonage à 1/10
Trajectoire sous-échantilloné
Ruptures de pente
Pyntcloud (neighbors)
Points rejetés
Homogénéisation
Ruptures de pente homogénéisées
Création des profils de trajectoire
Points les plus proches
Points trop éloignés
Bordures
Scikit-Learn (cluster.DBSCAN)
DBSCAN
Clustering
Points sans clusters
Bordures + Cluster
Successive Angles
Clusters éloignés
Bordures + Cluster
Récupération des points bas
Coordonnées (E,N,H) des fils d'eau
Nuage de points + attributs
Nuage contenant les points validés
Nuage contenant les points refusés
Calculs écrit sous Python
Test sur le nuage ou ses attributs
Pyntcloud
Bibliothèque utilisée (fonction utilisée)

Annexe 7

Pseudo-code - Calcul de la différence d'altitude locale :

```
Donnée en entrée : nuage de points correspondant au sol avec attributs (x,y,z)
appelé "sol"

Fonctions utilisées : "get_neighbors" de la bibliothèque Pyntcloud, outils de
manipulation de tableaux de la bibliothèque Numpy

                        ----------------------------

### Variables

sol = nuage
n = nombre de points dans sol

### Pseudo-code

# Calcul du voisinage

pts voisins rayon = Pyntcloud.get_neighbors(sol, r = 0.20)
pts k voisins = Pyntcloud.get_neighbors(sol, k = 4)

# Calcul de la différence et test

différence = tableau de la forme (n,1) rempli de "0"
points validés = [ ]

Pour i de 1 jusqu'à n avec un pas de 1:
    delta = (altitude de sol[i]) - (moyenne des altitudes de pts voisins
    rayon[i])

    Si la valeur absolue de (delta) est comprise entre 0.02 et 0.10:
        ajouter i à points validés

bordures = sol [points validés]
```

Annexe 8

Algorithme de détection d'objets en mouvement

Lors d'un levé par MMS, plusieurs passages sont effectués sur une même zone. Lors d'un levé avec un véhicule en milieu urbain, il y a au moins deux passages qui sont effectués, un premier dans un sens puis un autre sur l'autre voie. Cela permet deux choses. En premier lieu cela permet de combler quelques zones puisqu'il arrive régulièrement qu'il y ait des masques dus au sens de déplacement du véhicule et à l'inclinaison de l'appareil qui peuvent être comblé dans l'autre sens. Ensuite cela permet d'avoir plus de données et d'avoir un contrôle sur la donnée. Après post-traitement les nuages sont concaténés mais les points gardent un attribut correspondant au numéro de session de scan. Sur une dalle, un numéro de session est donc équivalent à un passage dans un sens et un autre numéro sera équivalent au passage dans l'autre sens. De plus lors de ces levés sur route, il est inévitable de lever des voitures qui circulent sur les voies ce qui créé un bruit très gênant dans le nuage. Cependant étant en mouvement, elles n'apparaissent que sur un passage et donc une session de scan comme on peut le voir sur la Figure 29. On voit en bleu les points levés lors d'une session de scan et en jaune les points levés lors d'une autre session (pour la visibilité le sol est grisé). Tous les éléments alentours sont composés de points jaunes et bleus, tandis que les voitures sont d'une seule couleur. On va donc utiliser cette particularité pour éliminer en grande partie ce bruit.

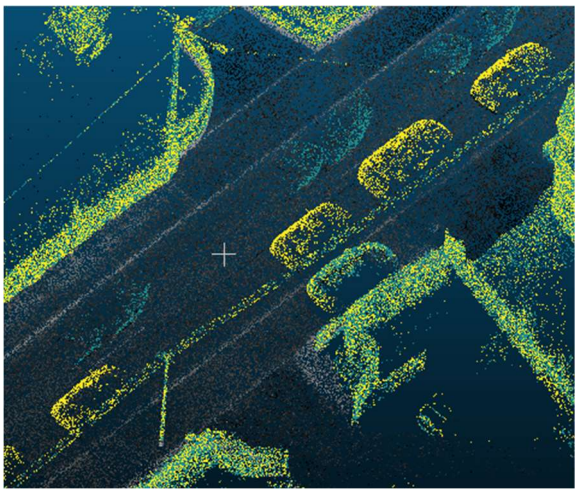


Figure 28 : Visualisation des différents passages du MMS

Figure 29 : Classification des objets en mouvement

Une première étape consiste à classer le sol avec un algorithme de RANSAC. Sur les points du sursol on applique ensuite un algorithme DBSCAN. Étant donné que les voitures sont en majorité isolées, le sol étant à part. Cet algorithme forme des groupes où chaque voiture est un cluster. Finalement on réalise un test sur chaque cluster. Si les points d'un cluster ont un numéro de session de scan unique, les clusters sont considérés comme en mouvement et donc stockés dans une classe à part. On obtient les résultats indiqués sur la Figure 30.

Annexe 9

Pseudo-code -Homogénéisation de la classe bordures

```
Donnée en entrée : nuage de points labellisés bordures

Fonctions utilisées : "get_neighbors" de la bibliothèque Pyntcloud, outils de
manipulation de tableaux de la bibliothèque Numpy

                        ----------------------------

### Variables

bordures = cf annexe 7
n = nombre de points de bordures

### Pseudo - code

# Calcul du voisinage
pts k voisins = Pyntcloud.get_neighbors(bordures, k = 8)

# Test
points validés = [ ]
Pour i de 1 jusqu'à n avec un pas de 1:
    voisins = bordures [ pts k voisins [ i ]]
    Si les labels des 8 voisins est égal à "1":
        ajouter i à points validés

labels bordures [points validés] = 0
```

Annexe 10

Captures d'écran des interfaces graphiques

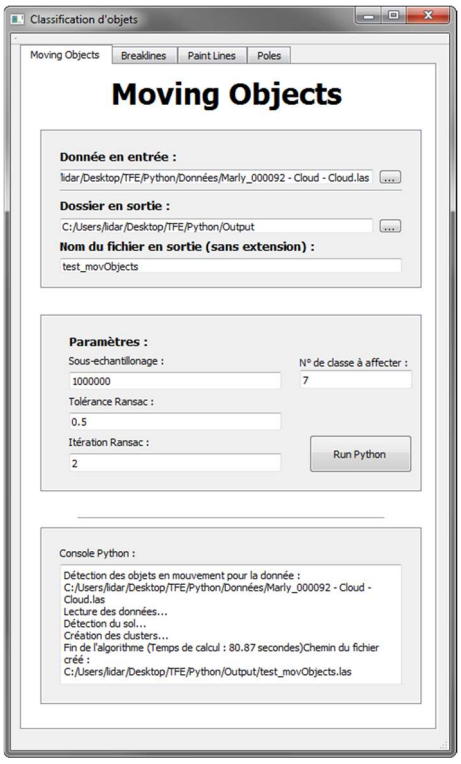


Figure 31 : Interface graphique pour les objets en mouvement

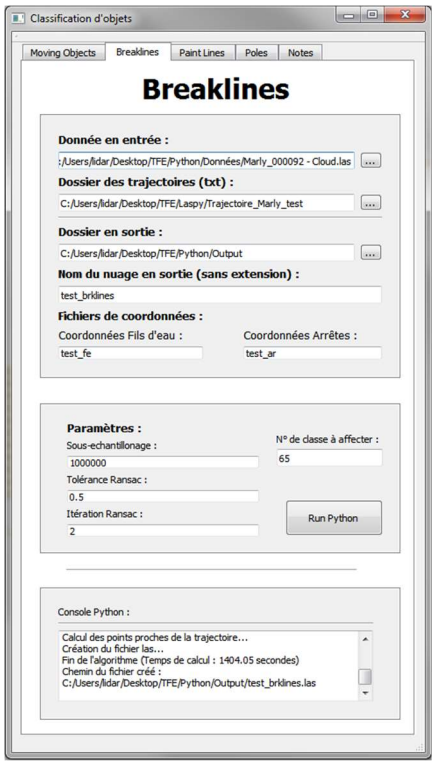


Figure 32 : Interface graphique pour la détection de bordures

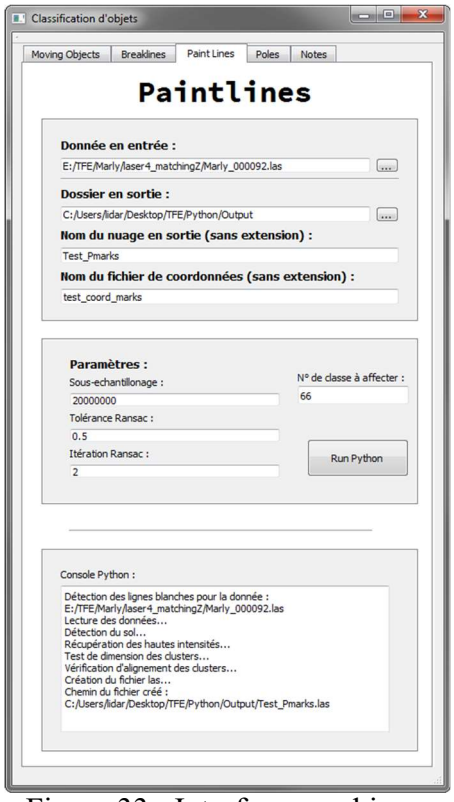


Figure 33 : Interface graphique pour les lignes blanches

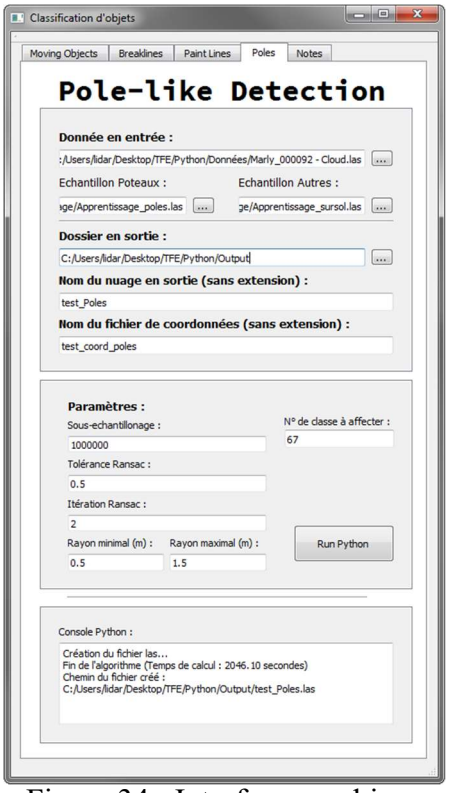
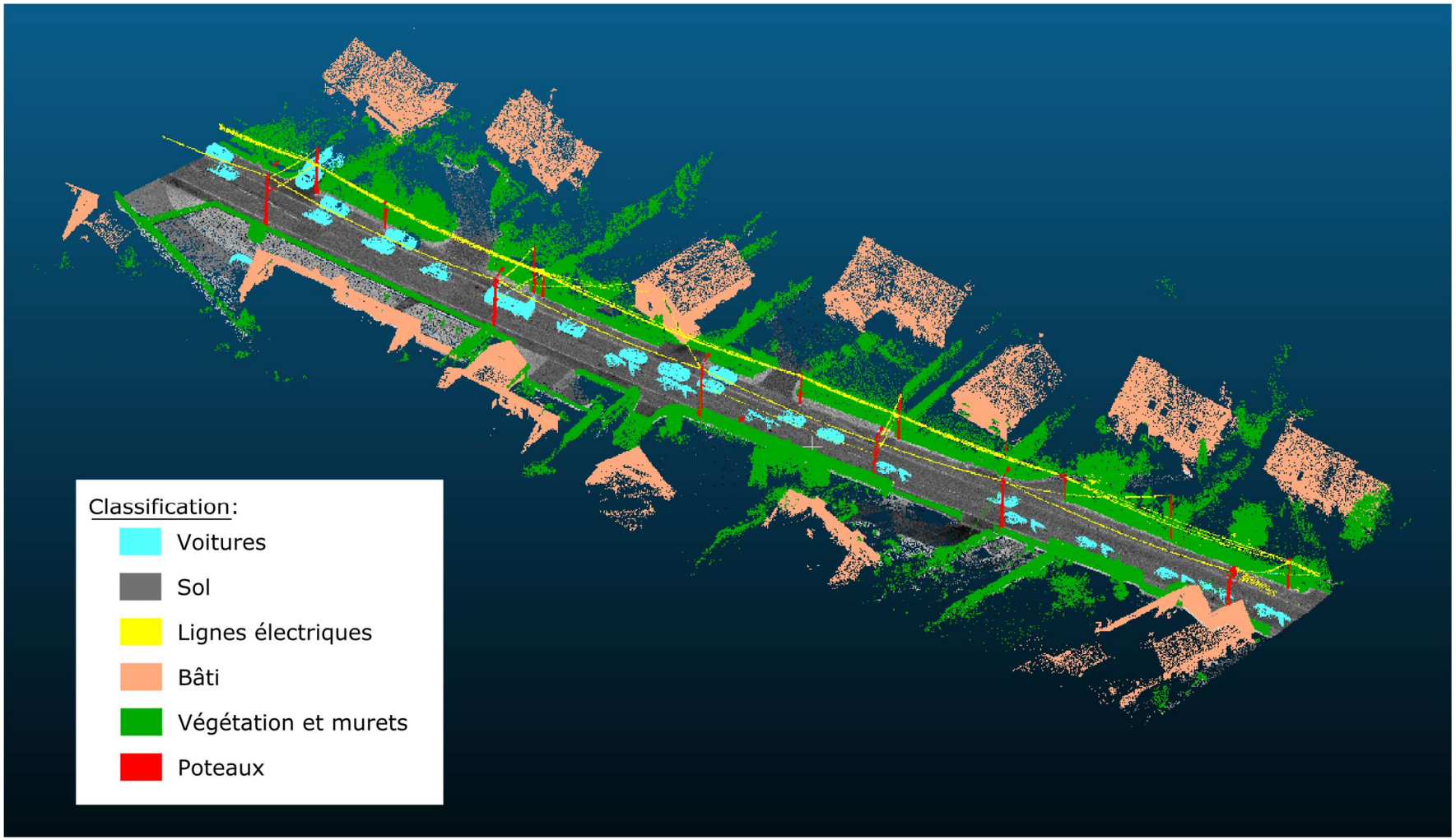


Figure 34 : Interface graphique pour la détection de poteaux

Annexe 11

Visuel de la dalle n°92 du dossier de Marly (Classification manuelle)

Capture d'écran du logiciel CloudCompare



Annexe 12

Tableau comparatif des positions planimétriques et altimétriques des poteaux

Méthode 1

N° Objet	Coordonnées opérateur			Coordonnées algorithme				ΔX (m)	ΔY (m)	ΔZ (m)	ΔXY (m)	ΔXYZ (m)
	Xo (m)	Yo (m)	Zo (m)	Xa (m)	Ya (m)	Za (m)						
1	1739190.613	9240464.420	54.238	1739190.561	9240464.578	54.176		0.052	0.158	0.062	0.166	0.178
2	1739208.836	9240472.382	54.131	1739208.824	9240472.359	54.090		0.012	0.023	0.041	0.026	0.049
3	1739206.082	9240462.451	54.284	1739206.068	9240462.489	54.250		0.014	0.038	0.034	0.040	0.053
4	1739230.368	9240467.918	54.213	1739230.468	9240468.045	54.144		0.100	0.127	0.069	0.162	0.176
5	1739250.561	9240463.272	54.208	1739250.561	9240463.352	54.167		0.000	0.080	0.041	0.080	0.090
6	1739255.599	9240452.790	54.283	1739255.823	9240452.713	54.204		0.224	0.077	0.079	0.237	0.250
7	1739284.369	9240455.262	54.136	1739284.308	9240455.27	54.067		0.061	0.015	0.069	0.063	0.093
8	1739311.763	9240449.028	53.939	1739311.756	9240448.997	53.869		0.00	0.031	0.070	0.032	0.07
9	1739339.290	9240442.921	53.924	1739339.246	9240442.923	53.790		0.044	0.002	0.134	0.044	0141
10	1739343.360	9240432103	54.143	1739343.291	9240432.020	54.010		0.069	0.083	0133	0.108	0.171
11	1739358.303	9240438.620	53.773	1739358.323	9240438.521	53.756		0.020	0.099	0.017	0.101	0.102
12	1739357.331	9240428.990	53.926	1739357.175	9240428.930	53.866		0.156	0.060	0.060	0.167	0.178
13	1739380.378	9240433.170	53.715	1739380.709	9240433.124	53.698		0.331	0.046	0.017	0.334	0.335
						Moyennes		**0.084**	**0.065**	**0.064**	**0.120**	**0.145**
						Min		**0.007**	**0.002**	**0.017**	**0.026**	**0.049**
						Max		**0.331**	**0.158**	**0.134**	**0.334**	**0.335**
						Ecart-type		**0.099**	**0.046**	**0.037**	**0.091**	**0.082**

Méthode 2

N° Objet	Coordonnées opérateur			Coordonnées algorithme				ΔX (m)	ΔY (m)	ΔZ (m)	ΔXY (m)	ΔXYZ (m)
	Xo (m)	Yo (m)	Zo (m)	Xa (m)	Ya (m)	Za (m)						
1	1739190.613	9240464.420	54.238	1739190.532	9240464.584	54.203		0.081	0.164	0.035	0.183	0.186
2	1739208.836	9240472.382	54.131	1739208.819	9240472.331	54.090		0.017	0.051	0.041	0.054	0.068
3	1739206.082	9240462.451	54.284	139206.087	9240462.510	54.276		0.005	0.059	0.008	0.059	0.060
4	1739230.368	9240467.918	54.213	1739230.339	9240467.974	54.253		0.029	0.056	0.040	0.063	0.075
5	1739250.561	9240463.272	54.208	1739250.565	9240463.266	54.172		0.004	0.006	0.036	0.007	0.037
6	1739255.599	9240452.790	54.283	1739255.743	9240452.623	54.283		0144	0.167	0.000	0.221	0.221
7	1739284.369	9240455.262	54.136	1739284.339	9240455.194	54.093		0.030	0.068	0.043	0.074	0.086
8	1739311.763	9240449.028	53.939	1739311.751	9240448.973	53.879		0.012	0.055	0.060	0.056	0.082
9	1739339.290	9240442.921	53.924	.1739339.307	9240442.917	53.830		0.017	0.004	0.094	0.017	0.096
10	1739343.360	9240432103	54.143	1739343.319	9240431.996	54.056		0.041	0.107	0.087	0.115	0.144
11	1739358.303	9240438.620	53.773	1739358.321	9240438.650	53.741		0.018	0.030	0.032	0.035	0.047
12	1739357.331	9240428.990	53.926	1739357.230	9240428.926	53.884		0101	0.064	0.042	0.120	0.127
13	1739380.378	9240433.170	53.715	1739380.399	9240433.113	53.689		0.021	0.057	0.026	0.061	0.066
						Moyennes		**0.040**	**0.068**	**0.042**	**0.082**	**0.099**
						Min		**0.004**	**0.006**	**0.000**	**0.007**	**0.037**
						Max		**0.144**	**0.167**	**0.094**	**0.221**	**0.221**
						Ecart-type		**0.042**	**0.051**	**0.027**	**0.062**	**0.055**

Annexe 13

Tableau comparatif des coordonnées des points pour les bordures / Côté Gauche – Fil d'eau

N° Objet	Coordonnées opérateur			Coordonnées algorithme				ΔX (cm)	ΔY (cm)	ΔZ (cm)	ΔXY (cm)	ΔXYZ (cm)
	Xo (m)	Yo (m)	Zo (m)	Xa (m)	Ya (m)	Za (m)						
1	1739824.421	9240354.161	50.385	1739824.420	9240354.112	50.357		0.1	4.9	2.8	4.9	5.6
2	1739833.470	9240354.418	50.314	1739833.477	9240354.356	50.303		0.7	6.2	1.1	6.2	6.3
3	1739839.443	9240354.737	50.285	1739839.442	9240354.723	50.273		0.1	1.4	1.2	1.4	1.8
4	1739850.041	9240355.387	50.179	1739850.049	9240355.372	50.172		0.8	1.5	0.7	1.7	1.8
5	1739855.429	9240355.714	50.143	1739855.437	9240355.695	50.141		0.8	1.9	0.2	2.1	2.1
6	1739860.208	9240356.013	50.128	1739860.213	9240356.005	50.111		0.5	0.8	1.7	0.9	1.9
7	1739872.104	9240356.736	50.083	1739872.104	9240356.730	50.077		0.0	0.6	0.6	0.6	0.8
8	1739875.005	9240356.892	50.065	1739875.010	9240356.870	50.062		0.5	2.2	0.3	2.3	2.3
9	1739887.092	9240357.438	49.993	1739887.105	9240357.439	49.994		1.3	0.1	0.1	1.3	1.3
10	1739899.647	9240357.894	49.971	1739899.640	9240357.855	49.968		0.7	3.9	0.3	4.0	4.0
11	1739907.258	9240358.122	49.930	1739907.268	9240358.113	49.923		1.0	0.9	0.7	1.3	1.5
12	1739911.755	9240358.228	49.901	1739911.774	9240358.224	49.899		1.9	0.4	0.2	1.9	2.0
13	1739912.993	9240358.244	49.892	1739913.006	9240358.251	49.900		1.3	0.7	0.8	1.5	1.7
14	1739918.050	9240358.309	49.876	1739918.052	9240358.274	49.875		0.2	3.5	0.1	3.5	3.5
15	1739926.365	9240358.318	49.859	1739926.367	9240358.297	49.855		0.2	2.1	0.4	2.1	2.1
16	1739930.588	9240358.312	49.834	1739930.599	9240358.270	49.823		1.1	4.2	1.1	4.3	4.5
17	1739940.666	9240358.226	49.774	1739940.675	9240358.169	49.758		0.9	5.7	1.6	5.8	6.0
18	1739943.400	9240358.192	49.744	1739943.411	9240358.175	49.736		1.1	1.7	0.8	2.0	2.2
19	1739951.945	9240358.048	49.778	1739951.949	9240357.987	49.773		0.4	6.1	0.5	6.1	6.1
20	1739979.540	9240357.038	49.879	1739979.534	9240357.024	49.871		0.6	1.4	0.8	1.5	1.7
21	1739982.111	9240356.902	49.886	1739982.120	9240356.873	49.883		0.9	2.9	0.3	3.0	3.1
22	1739988.250	9240356.566	49.905	1739988.253	9240356.545	49.900		0.3	2.1	0.5	2.1	2.2
						Moyennes		**0.7**	**2.5**	**0.8**	**2.8**	**2.9**
						Min / Max		**0.0 / 1.9**	**0.1 / 6.2**	**0.1 / 2.8**	**0.6 / 6.2**	**0.8 / 6.3**
						Ecart-type		**0.5**	**1.9**	**0.6**	**1.7**	**1.7**

Côté Gauche – Arrête

N° Objet	Coordonnées opérateur			Coordonnées algorithme				ΔX (cm)	ΔY (cm)	ΔZ (cm)	ΔXY (cm)	ΔXYZ (cm)
	Xo (m)	Yo (m)	Zo (m)	Xa (m)	Ya (m)	Za (m)						
1	1739824.463	9240354.181	50.450	1739824.452	9240354.194	50.448		1.1	1.3	0.2	1.7	1.7
2	1739833.385	9240354.435	50.365	1739833.399	9240354.431	50.364		1.4	0.4	0.1	1.5	1.5
3	1739839.457	9240354.755	50.342	1739839.459	9240354.808	50.357		0.2	5.3	1.5	5.3	5.5
4	1739850.092	9240355.415	50.258	1739850.092	9240355.462	50.275		0.0	4.7	1.7	4.7	5.0
5	1739855.432	9240355.737	50.215	1739855.430	9240355.728	50.183		0.2	0.9	3.2	0.9	3.3
6	1739860.255	9240356.044	50.194	1739860.258	9240356.036	50.189		0.3	0.8	0.5	0.9	1.0
7	1739872.181	9240356.778	50.144	1739872.183	9240356.794	50.152		0.2	1.6	0.8	1.6	1.8
8	1739874.963	9240356.912	50.121	1739874.974	9240356.917	50.126		1.1	0.5	0.5	1.2	1.3
9	1739887.192	9240357.467	50.119	1739887.192	9240357.516	50.123		0.0	4.9	0.4	4.9	4.9
10	1739899.638	9240357.900	50.038	1739899.634	9240357.917	50.069		0.4	1.7	3.1	1.7	3.6
11	1739907.170	9240358.145	50.007	1739907.193	9240358.172	50.006		2.3	2.7	0.1	3.5	3.5
12	1739911.714	9240358.246	49.943	1739911.716	9240358.199	49.910		0.2	4.7	3.3	4.7	5.7
13	1739913.044	9240358.266	49.941	1739913.047	9240358.306	49.940		0.3	4.0	0.1	4.0	4.0
14	1739918.004	9240358.326	49.969	1739918.000	9240358.354	49.990		0.4	2.8	2.1	2.8	3.5
15	1739926.385	9240358.343	49.901	1739926.396	9240358.371	49.903		1.1	2.8	0.2	3.0	3.0
16	1739930.677	9240358.336	49.869	1739930.666	9240358.326	49.850		1.1	1.0	1.9	1.5	2.4
17	1739940.706	9240358.246	49.831	1739940.700	9240358.203	49.782		0.6	4.3	4.9	4.3	6.5
18	1739943.332	9240358.230	49.820	1739943.332	9240358.235	49.821		0.0	0.5	0.1	0.5	0.5
19	1739951.910	9240358.063	49.848	1739951.915	9240358.054	49.827		0.5	0.9	2.1	1.0	2.3
20	1739979.466	9240357.063	49.951	1739979.478	9240357.053	49.939		1.2	1.0	1.2	1.6	2.0
21	1739982.169	9240356.925	49.945	1739982.161	9240356.920	49.950		0.8	0.5	0.5	0.9	1.1
22	1739988.251	9240356.591	49.994	1739988.261	9240356.628	49.992		1.0	3.7	0.2	3.8	3.8
						Moyennes		**0.7**	**2.3**	**1.3**	**2.6**	**3.1**
						Min / Max		**0.0 / 2.3**	**0.4 / 4.9**	**0.1/4.9**	**0.5 / 5.9**	**0.5/ 6.5**
						Ecart-type		**0.6**	**1.7**	**1.3**	**1.6**	**1.7**

Côté Droit – Fil d'eau

N° Objet	Coordonnées opérateur			Coordonnées algorithme				ΔX (cm)	ΔY (cm)	ΔZ (cm)	ΔXY (cm)	ΔXYZ (cm)
	Xo (m)	Yo (m)	Zo (m)	Xa (m)	Ya (m)	Za (m)						
1	1739972.626	9240351.832	49.842	1739972.600	9240351.833	49.838		2.6	0.1	0.4	2.6	2.6
2	1739962.247	9240352.232	49.779	1739962.232	9240352.298	49.773		1.5	6.6	0.6	6.8	6.8
3	1739951.568	9240352.530	49.729	1739951.570	9240352.561	49.727		0.2	3.1	0.2	3.1	3.1
4	1739942.741	9240352.749	49.686	1739942.733	9240352.774	49.682		0.8	2.5	0.4	2.6	2.7
5	1739939.423	9240352.750	49.732	1739939.414	9240352.765	49.718		0.9	1.5	1.4	1.7	2.2
6	1739935.244	9240352.806	49.725	1739935.241	9240352.820	49.723		0.3	1.4	0.2	1.4	1.4
7	1739929.704	9240352.832	49.782	1739929.690	9240352.857	49.774		1.4	2.5	0.8	2.9	3.0
8	1739917.064	9240352.784	49.855	1739917.027	9240352.874	49.840		3.7	9.0	1.5	9.7	9.8
9	1739915.077	9240352.749	49.856	1739915.070	9240352.773	49.852		0.7	2.4	0.4	2.5	2.5
10	1739912.448	9240352.731	49.861	1739912.432	9240352.802	49.851		1.6	7.1	1.0	7.3	7.3
11	1739908.815	9240352.678	49.888	1739909.195	9240351.779	50.400		38.0	89.9	51.2	97.6	110.2
12	1739905.506	9240352.600	49.911	1739905.477	9240352.634	49.898		2.9	3.4	1.3	4.5	4.7
13	1739894.818	9240352.244	49.959	1739894.809	9240352.318	49.956		0.9	7.4	0.3	7.5	7.5
14	1739891.612	9240352.101	50.038	1739891.587	9240352.101	50.023		2.5	0.0	1.5	2.5	2.9
15	1739887.887	9240351.981	50.013	1739887.866	9240351.792	49.943		2.1	18.9	7.0	19.0	20.3
16	1739870.261	9240351.108	50.090	1739870.248	9240351.158	50.078		1.3	5.0	1.2	5.2	5.3
17	1739866.207	9240350.869	50.104	1739866.219	9240350.905	50.107		1.2	3.6	0.3	3.8	3.8
18	1739863.051	9240350.671	50.094	1739863.039	9240350.753	50.097		1.2	8.2	0.3	8.3	8.3
19	1739859.198	9240350.432	50.097	1739859.202	9240350.446	50.102		0.4	1.4	0.5	1.5	1.5
20	1739843.639	9240349.488	50.204	1739843.638	9240349.546	50.193		0.1	5.8	1.1	5.8	5.9
21	1739821.115	9240348.639	50.326	1739821.119	9240348.658	50.319		0.4	1.9	0.7	1.9	2.1
22	1739822.449	9240348.647	50.326	1739822.486	9240348.660	50.325		3.7	1.3	0.1	3.9	3.9
						Moyennes		**3.1**	**8.3**	**3.3**	**9.2**	**9.9**
						Min / Max		**0.0 / 3.7**	**0.0 / 9.0**	**0.0 / 1.5**	**0.0 / 9.7**	**0.0 / 9.8**
						Ecart-type		**7.9**	**18.7**	**10.8**	**20.1**	**22.8**

Moyenne et écart-type sans N°11:

Données	ΔX	ΔY	ΔZ	ΔXY	ΔXYZ
Moy (cm)	1.4	4.2	1.0	4.8	4.9
Ecart-type (cm)	1.1	4.3	1.5	4.0	4.2

Côté Droit – Arrête

N° Objet	Coordonnées opérateur			Coordonnées algorithme				ΔX (cm)	ΔY (cm)	ΔZ (cm)	ΔXY (cm)	ΔXYZ (cm)
	Xo (m)	Yo (m)	Zo (m)	Xa (m)	Ya (m)	Za (m)						
1	1739972.558	9240351.801	49.882	1739972.552	9240351.787	49.877		0.6	1.4	0.5	1.5	1.6
2	1739962.261	9240352.213	49.849	1739962.256	9240352.296	49.776		0.5	8.3	7.3	8.3	11.1
3	1739951.546	9240352.518	49.770	1739951.55	9240352.502	49.781		0.4	1.6	1.1	1.6	2.0
4	1739942.709	9240352.677	49.784	1739942.694	9240352.687	49.773		1.5	1.0	1.1	1.8	2.1
5	1739939.318	9240352.736	49.806	1739939.333	9240352.705	49.818		1.5	3.1	1.2	3.4	3.6
6	1739935.244	9240352.786	49.775	1739935.270	9240352.784	49.789		2.6	0.2	1.4	2.6	3.0
7	1739929.683	9240352.812	49.819	1739929.685	9240352.795	49.826		0.2	1.7	0.7	1.7	1.8
8	1739917.047	9240352.768	49.879	1739917.043	9240352.792	49.850		0.4	2.4	2.9	2.4	3.8
9	1739915.036	9240352.736	49.894	1739915.057	9240352.729	49.899		2.1	0.7	0.5	2.2	2.3
10	1739912.437	9240352.710	49.942	1739912.449	9240352.722	49.942		1.2	1.2	0.0	1.7	1.7
11	1739909.229	9240352.643	49.966	1739909.195	9240351.779	50.400		3.4	86.4	43.4	86.5	96.7
12	1739905.524	9240352.558	49.941	1739905.508	9240352.540	49.942		1.6	1.8	0.1	2.4	2.4
13	1739894.791	9240352.214	50.030	1739894.776	9240352.226	49.960		1.5	1.2	7.0	1.9	7.3
14	1739891.514	9240352.087	50.068	1739891.508	9240352.010	50.090		0.6	7.7	2.2	7.7	8.0
15	1739887.709	9240351.800	49.993	1739887.817	9240351.815	50.095		0.8	1.5	10.2	10.9	14.9
16	1739870.313	9240351.085	50.113	1739870.313	9240351.085	50.113		0.0	0.0	0.0	0.0	0.0
17	1739866.287	9240350.828	50.149	1739866.294	9240350.847	50.144		0.7	1.9	0.5	2.0	2.1
18	1739863.095	9240350.652	50.146	1739863.103	9240350.830	50.096		0.8	17.8	5.0	17.8	18.5
19	1739859.237	9240350.421	50.148	1739859.242	9240350.407	50.165		0.5	1.4	1.7	1.5	2.3
20	1739843.629	9240349.475	50.278	1739843.636	9240349.463	50.289		0.7	1.2	1.1	1.4	1.8
21	1739821.095	9240348.598	50.367	1739821.104	9240348.595	50.375		0.9	0.3	0.8	0.9	1.2
22	1739822.422	9240348.595	50.371	1739822.425	9240348.607	50.367		0.3	1.2	0.4	1.2	1.3
							Moyennes	**1.5**	**6.5**	**4.0**	**7.4**	**8.6**
							Min / Max	**0.0 / 10.8**	**0.0 / 86.4**	**0.0 / 43.4**	**0.0 / 86.5**	**0.0 / 96.7**
							Ecart-type	**2.2**	**18.3**	**9.2**	**18.1**	**20.3**

Moyenne et écart-type sans N°11:

Données	ΔX	ΔY	ΔZ	ΔXY	ΔXYZ
Moy (cm)	1.4	2.7	2.2	3.6	4.4
Ecart-type (cm)	2.3	4.1	2.8	4.2	4.9

Annexe 14

Tableau comparatif des coordonnées des sections de linéaires

Côté Gauche – Fil d'eau

N° Profil	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	Moy	Min/Max	Std[12]
ΔX (cm)	0.1	0.2	0.1	0.0	0.0	0.1	0.2	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.1	0.2	0.0	0.2	0.0	0.0	0.6	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0	**0.1**	**0.0 / 0.2**	**0.1**
ΔY (cm)	2.2	3.3	1.2	1.0	0.2	2.4	2.9	0.6	0.5	1.3	0.2	0.5	0.3	1.3	0.5	4.5	4.5	0.6	3.4	6.0	6.7	5.0	1.5	1.1	1.5	5.4	4.4	1.0	0.3	0.1	**2.1**	**0.1 / 6.7**	**2.0**
ΔZ (cm)	0.0	0.0	0.0	2.8	2.9	1.7	1.2	1.7	1.7	2.3	4.0	5.8	6.5	5.1	2.8	1.3	1.3	1.4	1.4	1.1	0.7	1.1	1.5	0.7	1.3	2.3	1.0	0.9	1.0	0.0	**1.9**	**0.0 / 6.5**	**1.6**
ΔXY (cm)	2.2	3.3	1.2	1.0	0.2	2.4	2.9	0.6	0.5	1.3	0.2	0.5	0.3	1.3	0.5	4.5	4.5	0.6	3.4	6.0	6.7	5.0	1.5	1.1	1.5	5.4	4.4	1.0	0.3	0.1	**2.1**	**0.1 / 6.7**	**2.0**

Côté Gauche – Arrête

N° Profil	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	Moy	Min/Max	Std
ΔX (cm)	0.1	0.3	0.3	0.2	0.0	0.2	0.0	0.1	0.3	0.2	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.2	0.0	0.0	0.2	0.1	0.2	0.1	0.2	0.1	**0.1**	**0.0 / 0.3**	**0.1**
ΔY (cm)	2.3	3.2	2.8	1.3	1.0	2.4	1.3	2.5	3.9	2.4	1.1	0.8	1.9	3.2	3.3	2.1	0.6	0.9	2.1	3.7	4.9	4.0	1.2	4.7	3.6	3.7	4.7	4.8	6.9	5.3	**2.9**	**0.6 / 6.9**	**1.6**
ΔZ (cm)	2.8	3.0	0.2	1.1	1.0	3.5	2.9	0.2	2.4	1.6	1.0	0.0	0.3	0.3	1.4	2.6	4.4	5.9	6.3	6.8	7.9	8.2	4.7	0.0	0.3	0.3	1.8	0.1	1.0	1.6	**2.5**	**0.0 / 8.2**	**2.5**
ΔXY (cm)	2.3	3.2	2.8	1.3	1.0	2.4	1.3	2.5	3.9	2.4	1.1	0.8	1.9	3.2	3.3	2.1	0.6	0.9	2.1	3.7	4.9	4.0	1.2	4.7	3.6	3.7	4.7	4.8	6.9	5.3	**2.9**	**0.6 / 6.9**	**1.6**

[12] Std : Écart-type

Côté Droit – Fil d'eau

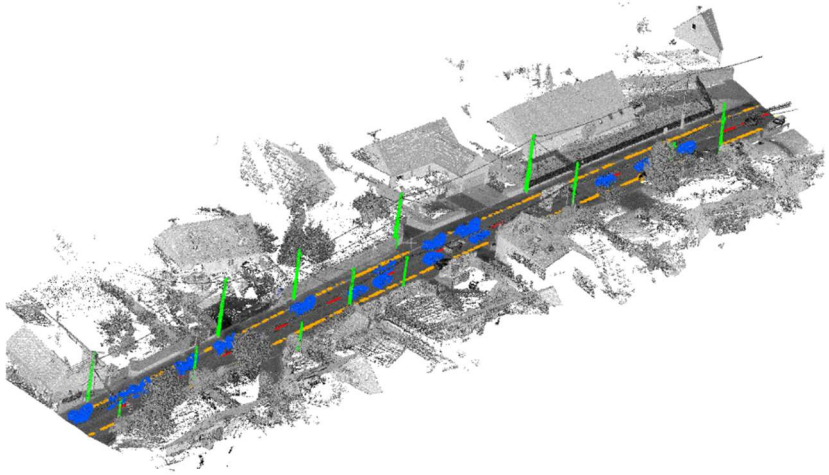
N° Profil	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	Moy	Min/Max	Std
ΔX (cm)	0.0	0.2	0.3	0.5	1.3	1.0	0.1	0.4	0.4	0.2	0.1	0.5	0.5	0.4	0.4	0.2	0.1	0.0	0.2	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.3	0.3	0.1	0.3	**0.3**	**0.0 / 1.3**	**0.3**
ΔY (cm)	1.0	7.6	7.0	7.9	16.2	11.8	2.3	5.5	6.4	3.3	2.0	9.6	10.9	9.1	9.1	4.8	1.7	8.8	3.6	7.3	1.0	3.6	5.4	0.1	3.3	2.2	9.3	18.1	3.6	5.4	**6.3**	**0.1 / 18.1**	**4.3**
ΔZ (cm)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	**0.0 / 0.0**	**0**
ΔXY (cm)	1.0	7.6	7.0	7.9	16.3	11.8	2.3	5.5	6.4	3.3	2.0	9.6	10.9	9.1	9.1	4.8	1.7	8.8	3.6	7.3	1.0	3.6	5.4	0.1	3.3	2.2	9.3	18.1	3.6	5.4	**6.3**	**0.1 / 16.3**	**4.4**

Côté Droit – Arrête

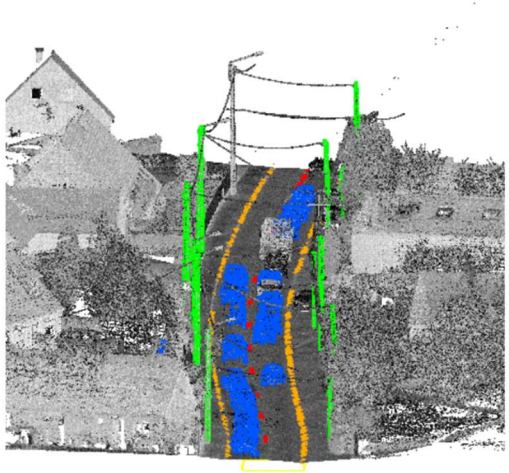
N° Profil	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	Moy	Min/Max	Std
ΔX (cm)	0.1	0.3	0.3	0.2	0.0	0.2	0.0	0.1	0.3	0.2	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.1	0.1	0.1	0.2	0.0	0.0	0.2	0.1	0.2	0.1	0.2	0.1	**0.1**	**0.0 / 0.7**	**0.2**
ΔY (cm)	2.3	3.2	2.8	1.3	1.0	2.4	1.3	2.5	3.9	2.4	1.1	0.8	1.9	3.2	3.3	2.1	0.6	0.9	2.1	3.7	4.9	4.0	1.2	4.7	3.6	3.7	4.7	4.8	6.9	5.3	**2.9**	**0.4 / 12.0**	**3**
ΔZ (cm)	2.8	3.0	0.2	1.1	1.0	3.5	2.9	0.2	2.4	1.6	1.0	0.0	0.3	0.3	1.4	2.6	4.4	5.9	6.3	6.8	7.9	8.2	4.7	0.0	0.3	0.3	1.8	0.1	1.0	1.6	**2.5**	**0.0 / 4.8**	**1**
ΔXY (cm)	2.3	3.2	2.8	1.3	1.0	2.4	1.3	2.5	3.9	2.4	1.1	0.8	1.9	3.2	3.3	2.1	0.6	0.9	2.1	3.7	4.9	4.0	1.2	4.7	3.6	3.7	4.7	4.8	6.9	5.3	**2.9**	**0.1 / 12.0**	**3**

Annexe 15

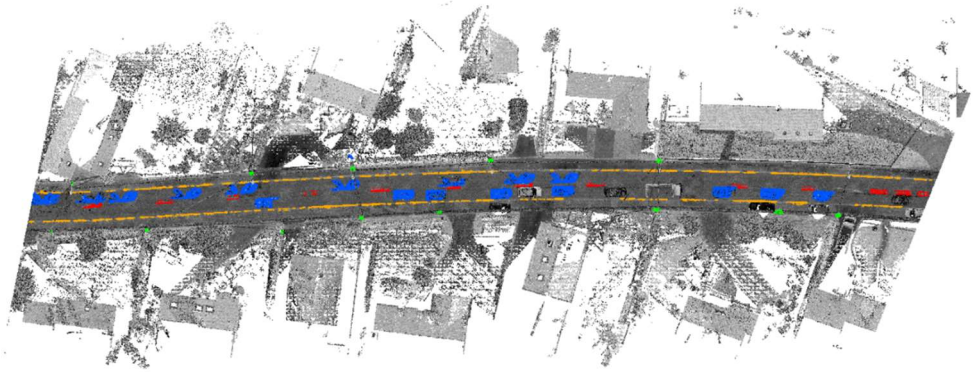
Visuel de la dalle n°92 du dossier de Marly classifiée grâce aux algorithmes



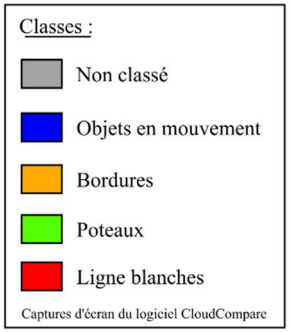
Vue isométrique



Vue de côté



Vue de dessus

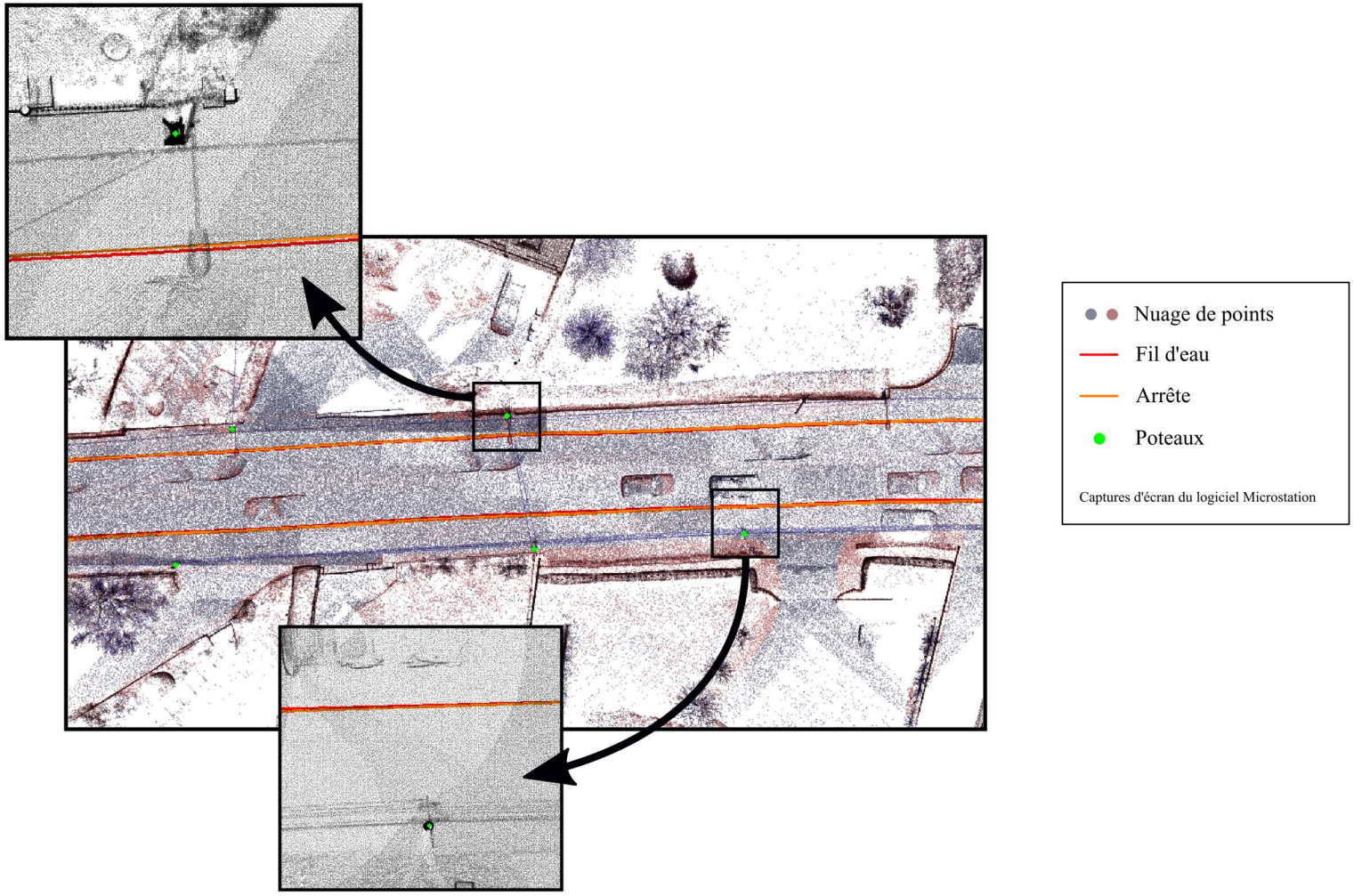


Captures d'écran du logiciel CloudCompare

Annexe 16

Visuel de la dalle n°92 du dossier de Marly digitalisé grâce aux algorithmes

Les objets (points et lignes) ont été volontairement agrandis pour des raisons de visibilité.



Liste des figures

Liste des tableaux

Détection des objets dans un environnement urbain à partir de données Lidar

Mémoire d'Ingénieur C.N.A.M., Le Mans 2019

RÉSUMÉ

La technologie de Mobile Mapping System (MMS) se démocratise depuis quelques années et s'installe petit à petit dans les entreprises de géomatique. Les nuages de points qui découlent de cette technologie renouvellent la manière de traiter la donnée et de réaliser des plans. Ces nuages de points proposent une donnée exhaustive de l'environnement et le choix de la donnée pour la réalisation de plan est réalisé en post-traitement. L'objectif de ce mémoire est de proposer des solutions et des méthodes de traitement pour automatiser un maximum la création de plans.

Pour cela, cette étude est focalisée sur deux types d'objets que sont les poteaux et les bordures de trottoir afin de représenter les objets dessinés ponctuellement et linéairement sur un plan. Des méthodes utilisant des descripteurs locaux et globaux ainsi que des algorithmes de segmentation sont proposées pour détecter et vectoriser ces éléments de manière robuste et optimisée. Une analyse sur la précision et l'efficacité de ces méthodes est ensuite proposée.

Mots clés : 3D, nuage de points, MMS, détection automatique, segmentation, descripteurs, cluster

---

ABSTRACT

The Mobile Mapping System technology has become common since the last few years in geomatic companies. Point clouds resulting from this technology bring a new way of treating the data and drawing topographic plans. Point clouds offer a comprehensive data of the environment. Thereby, the choice of the useful data for drawing the plans is made in post treatment. The purpose of this paper is to propose solutions for an automation of the drawing.

For this purpose, this study is focused on two types of objects, poles and curbs. These objects represent the punctual and the linear elements drawn on plans. Methods using local and global descriptors along with segmentation algorithm are proposed to detect and vectorize elements in a robust and optimized way. An analysis of the accuracy and the efficiency of those methods is then explained.

Key words : 3D, point cloud, MMS, automatic detection, segmentation, descriptors, cluster