Extraction et utilisation de connaissances métier structurées pour la classification de nuages de points 3D

Auteur : Lesecque, Florian
Promoteur(s) : Billen, Roland
Faculté : Faculté des Sciences
Diplôme : Master en sciences géographiques, orientation géomatique et géométrologie, à finalité spécialisée
Année académique : 2018-2019
URI/URL : http://hdl.handle.net/2268.2/7363

Faculté des sciences
Département de géographie

Extraction et utilisation de connaissances métier structurées pour la classification de nuages de points 3D

Mémoire présenté par : **Florian LESECQUE**

pour l'obtention du titre de

**Master en sciences géographiques,
orientation géomatique et géométrologie**

Année académique :	**2018-2019**
Date de défense :	**Septembre 2019**
Président de jury :	Pr. **Jean-Paul DONNAY**
Promoteur :	Pr. **Roland BILLEN**
Jury de lecture :	Pr. **Yves CORNET**
	Pr. **Serge SCHMITZ**

REMERCIEMENTS

Je tiens, tout d'abord, à remercier mon promoteur, Monsieur R. Billen, pour son implication et ses commentaires avisés ainsi que pour son suivi et ses indications qui m'ont aiguillé tout au long de ce travail.

Je tiens ensuite à remercier Monsieur F. Poux pour ses nombreux conseils judicieux et explications qui m'ont permis de mieux appréhender le sujet et d'aller de l'avant.

Je remercie aussi particulièrement Messieurs Y. Cornet et S. Schmitz, mes lecteurs, ainsi que les membres du jury pour l'intérêt porté à mon travail.

Mes remerciements vont également à A. Kharroubi et A. Tabkha, stagiaires au sein de l'Unité de Géomatique, pour leurs échanges intéressants et pertinents.

Je tiens aussi à adresser mes remerciements à mes parents et ma sœur qui m'ont toujours supporté et encouragé, dans les bons et les mauvais moments, et qui ont cru en moi tout au long de mes études.

Enfin, je remercie mes proches et mes amis pour leur soutien moral.

RESUME / ABSTRACT

Résumé

Parmi les traitements des nuages de points, la segmentation et classification sémantique sont à la base de nombreux travaux et algorithmes. Leur réalisation manuelle présente de nombreux défauts, c'est pourquoi beaucoup cherchent à l'automatiser. Ces opérations peuvent notamment être facilitées par des apports externes d'informations. Parmi les connaissances métier structurées, le plan d'intérieur 2D CAD (Computer Aided Design) constitue une source d'informations spatiales et sémantiques enregistrées dans des formats précis. Une analyse de la littérature et de travaux liés à ce sujet suggère que peu de recherches intègrent ces plans CAD comme source d'informations pour la segmentation et classification de nuages de points 3D.

Ce travail vise donc à développer une méthode de segmentation et classification sémantique d'un nuage de points 3D d'environnement intérieur, méthode qui se base sur des informations sémantiques et spatiales extraites du plan CAD 2D associé à cet environnement

Une méthodologie en neuf étapes détaillant le développement d'un tel algorithme est donc prosposée. Après certains prétraitements, la méthode extrait de l'information sémantique et spatiale des plans. Ces informations sont alors utilisées afin de segmenter, en différentes classes sémantiques, un nuage de points correspondant au plan. Cette segmentation se base sur la création de zones tampons associées aux éléments du plan et sur des statistiques de hauteur des points. Des opérations d'optimisation sont aussi réalisées.

Une fois développée, cette méthode est appliquée à différents nuages pour la valider et l'évaluer statistiquement. Les différents résultats obtenus sont alors présentés, analysés et discutés. Cette analyse prouve que la méthode est fonctionnelle et qu'elle peut fournir des résultats quasiment similaires à ceux de la méthode manuelle, mais de manière plus rapide. Enfin, certaines limitations et perspectives de développement sont mises en avant.

Abstract

Among possible treatments to address 3D point clouds, segmentation and semantic classification serve as a foundation for many works and algorithms. However, their manual implementation presents several shortcomings. Thus many are seeking to develop automatic implementations instead, which could particularly benefit from the importation of external information. Among structured professional data, 2D CAD* indoor floor plans contain some semantic and spatial information recorded in well-defined and structured file format. Yet, literature and related work analysis suggest that few researches integrate such CAD floor plans as a source of information for the segmentation and classification of 3D point clouds.

Hence this work aims to develop a 3D indoor point cloud segmentation and semantic classification method. This method should use semantic and spatial information extracted from the 2D CAD floor plan depicting the same environment as the point cloud.

The development of such an algorithm is presented through a nine steps methodology. After some pretreatments, the algorithm extracts semantic and spatial information from the 2D CAD floor plan.These informations are then used to segment in multiple semantic classes the 3D point cloud representing the same environment as the plan. This segmentation is based on the creation of buffer zones related to plan elements as well as on the analysis of points height statistics. Additionally, optimization processes are also carried out.

Once this algorithm is developed, it is then applied to various point clouds in order to validate and statistically evaluate it. The results are then presented, analysed and discussed. This analysis proves that the method is functional and can provide results that are nearly similar to the ones obtained by manual implementation, but in a faster way. At last, some limitations and different development perspectives are highlighted.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

CHAPITRE I. INTRODUCTION

Depuis quelques années déjà, les nuages de points 3D sont de plus en plus produits et utilisés dans de nombreux domaines : ingénierie, architecture, construction, topographie, robotique, patrimoine, archéologie, analyse de risques, transport, divertissement, … (Xiong & Huber, 2010 ; Poux et al., 2016 ; Poux et al., 2018). Cela peut notamment s'expliquer par les avancées récentes en termes de capacité de traitements informatiques, ainsi que par la démocratisation et diversification des méthodes et appareils de levé : scanners laser terrestres, mobiles (drones, robots, voitures, etc.), LiDAR, … (Nguyen & Le, 2013 ; Poux et al., 2016).

Un exemple récent et éloquent de leur utilisation concerne la reconstruction de la célèbre Cathédrale Notre-Dame de Paris, partiellement brulée le 15 avril 2019. En effet, la restauration de ce monument est envisageable grâce à une représentation 3D de celui-ci, basée sur l'utilisation de nuages de points réalisés auparavant. (Obadia, 2019)

Figure 1 Nuage de points de la Cathédrale Notre-Dame de Paris (Obadia, 2019)

Un levé au scanner laser 3D peut permettre d'acquérir un nuage de plusieurs millions de points rapidement (plusieurs dizaines de milliers de points à la seconde) et précisément (de l'ordre du millimètre). A ces points peuvent être associées bon nombre d'informations différentes. Un nuage de points est donc une source considérable et relativement exhaustive de données permettant de décrire un environnement 3D (Poux et al., 2017a ; Djemâ, 2018). Cependant, malgré ces avantages, les nuages de points ne récoltent pas les informations sémantiques liées aux objets représentés. L'interprétation des données et leur traitement sont donc nécessaires pour pouvoir les employer dans d'autres travaux. Toutefois, vu le volume des données et certains problèmes intrinsèques aux nuages de points, ces opérations constituent une tâche manuelle délicate, fastidieuse et complexe (Poux et al., 2016 ; Grilli et al., 2017). Parmi ces traitements, la classification et la segmentation du nuage constituent souvent une seule et même étape. Cette étape suscite de nombreuses recherches consacrées notamment à son automatisation (Nguyen & Le, 2013 ; Engelmann et al., 2017 ; Grilli et al., 2017). Elle va, en effet, permettre de segmenter un nuage selon certaines caractéristiques et de conférer de l'information sémantique à ces segments en leur attribuant une classe.

Pour réaliser une segmentation-classification d'un nuage de points, il faut déterminer et fournir des informations pertinentes et adaptées sur lesquelles se baser. C'est ce que nous appelons dans notre titre des « connaissances métier structurées ».

En 2010, Budroni & Boehm spécifiaient que l'introduction de techniques CAD (Computer Aided Design) avait représenté un tournant dans la manière de concevoir un projet de conception architecturale. De manière générale, nous pouvons considérer que ce tournant s'appliquait plus largement à d'autres domaines notamment celui de la construction. Ainsi, dans un contexte de représentation 3D de l'intérieur d'un bâtiment, les plans CAD d'intérieurs (plans d'étage) constituent des sources d'informations dignes d'intérêts. En effet, Lewis & Séquin (1998) déclaraient dejà en 1998 : *"The existence of such floor plans should be considered a significant investment"*, dans le cadre de la création de modèles 3D de bâtiments à partir de plans CAD d'architecte. Nous avons donc envisagé l'utilisation de plans d'intérieur au format CAD (plus spécifiquement au format structuré « .dxf »), comme « connaissances métier structurées » pour notre travail. En effet, ces plans contiennent beaucoup d'informations structurées et pertinentes pour la segmentation-classification d'un nuage de points 3D représentant l'intérieur du bâtiment associé.

Si beaucoup de recherches abordent le problème de segmentation-classification, peu utilisent ce type de données CAD bidimensionnelles dans leur méthode. Dans ce travail, nous avons donc essayé de répondre à la question de recherche suivante :

« Est-il possible de développer une méthode automatique et efficace (en termes de rapidité, précision et exactitude) de segmentation et classification d'un nuage de points 3D d'intérieur de bâtiment, à partir d'informations sémantiques et spatiales extraites du plan CAD associé à cet environnement ? »

Pour y répondre, nous commencerons par définir certains concepts importants et par présenter divers travaux associés à ce sujet (Etat de l'art, cf. Chapitre II). Nous poserons ensuite notre hypothèse de travail (cf. Chapitre III), puis présenterons les étapes de la méthodologie employée : depuis le prétraitement des données jusqu'à la validation des résultats (cf. Chapitre IV). Le développement de l'algorithme, les résultats obtenus et leur validation seront ensuite détaillés (cf. Chapitre V) avant d'être discutés et d'en aborder les limites (cf. Chapitre VI). Nous conclurons en rappelant les points pertinents de ce travail, en répondant à notre question-problème et nos hypothèses, et en prosposant des perspectives de développement (cf. Chapitre VII).

CHAPITRE II. ÉTAT DE L'ART

II.1. Définitions et informations de base

II.1.1. Nuages de points 3D et segmentation-classification

II.1.1.1. Nuages de points 3D

Un nuage de points 3D est une collection plus ou moins large de points localisés dans les 3 dimensions de l'espace et pour lesquels on dispose d'un certain nombre d'informations : couleurs (sous formes de valeur RVB), l'intensité, … (Djemâ, 2018). Ces collections de points sont ainsi utilisées pour représenter et travailler avec différents objets (ou scènes) en utilisant les informations qui y sont associées. Comme nous le verrons ci-dessous, ces collections possèdent en effet divers avantages et sont utilisées dans de nombreux domaines.

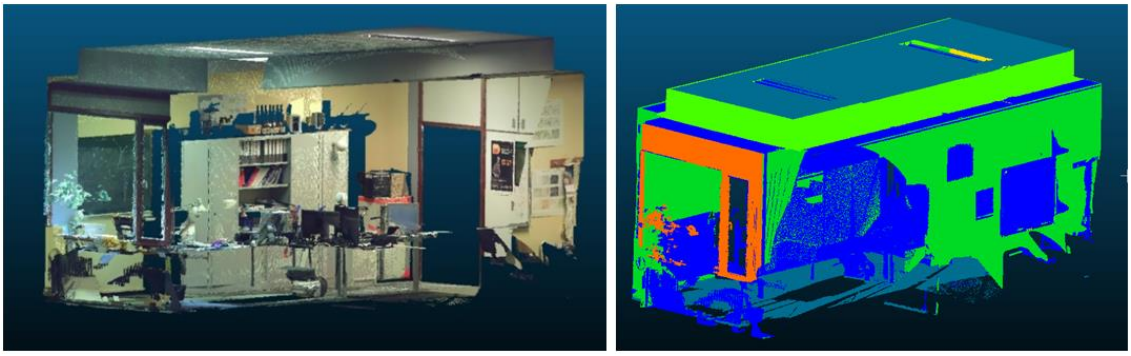


Figure 2 Exemple de nuage de points (tronqué) représentant un bureau (à gauche), et de sa segmentation-classification manuelle (à droite). Ce nuage sera utilisé par la suite sous le nom "Office_4" et la segmentation correspond à sa vérité terrain

Un nuage de points peut être relevé selon différents types de méthodes et d'appareils : scanners laser terrestres (Terrestrial Laser Scanning ou TLS), scanners laser mobiles (Mobile Laser Scanning ou MLS) tels que des scanners portables à la main ou montés sur un sac-à-dos, sur un robot, sur une voiture, … On trouve aussi des scanners laser aériens (Aerial Laser Scanning ou ALS), LiDAR, etc. (Xiong & Huber, 2010 ; Nurunnabi et al., 2016 ; Poux et al., 2017b).

Comme l'explique Djemâ (2018), le scanner va balayer la scène avec un rayon de haute fréquence et selon un pas de temps régulier. En connaissant la direction du rayon (via les angles vertical et horizontal) et la distance à laquelle se trouve l'objet mesuré, l'appareil peut créer un point positionné dans l'espace. Le calcul de la distance parcourue par le rayon peut se faire de différentes manières : mesure d'un temps de parcours, mesure d'un déphasage, triangulation. Selon le type de technique utilisée, la précision des mesures, la portée et la vitesse d'acquisition varient. Pour récupérer d'autres informations telles que les couleurs, d'autres capteurs (appareil photo par exemple) peuvent être utilisés en association avec le scanner. Nous pouvons remarquer que, au vu du principe d'acquisition de ces données, tous les objets ne seront pas représentés. En effet, comme le précisent Xiong & Huber (2010), *« Laser scanners can provide accurate 3D measurements of the visible surfaces of a facility… »* : seules les surfaces visibles de la scène sont mesurées et représentées dans le nuage. Certains objets peuvent donc être cachés par d'autres : on parlera alors d'occlusion.

Comme pour un plan 2D, le nuage de points peut être géoréférencé. Ce géoréférencement consiste à positionner le nuage de points dans un système de coordonnées de référence bien défini. Cette opération peut être directe ou indirecte. Dans le premier cas, le scanner est, sur terrain, directement positionné dans le système de coordonnées de référence choisi grâce à des points qui y sont déjà référencés (point de station et points visés). Ainsi, les coordonnées des points sont directement spécifiées dans le bon système. Le référencement sera indirect lorsque les points sont relevés dans un système local arbitraire et qu'une transformation leur est ensuite appliquée pour les faire passer de leur système vers le système choisi (recalage). Cette transformation correspond à une transformation de Helmert 3D linéarisée et ajustée par moindres carrés. (Djemâ, 2018).

Si plusieurs nuages de points ont été réalisés et doivent être combinés, on effectuera ce qu'on appelle une consolidation du nuage. Cette consolidation correspond à un ensemble de géoréférencements indirects permettant de regrouper les différents nuages. En effet, chacun de ces sous-nuages est exprimé dans un système local propre et il faut pouvoir les amener tous dans un seul et même système de référence. Cette opération pourra être réalisée grâce à des recalages (Transformation de Helmert 3D, telle que citée précédemment) entre tous ces nuages. Ces recalages seront possibles si les nuages se recouvrent suffisamment entre eux de sorte à inclure des points identiques et identifiables (cibles). Cette étape peut se faire manuellement ou automatiquement. (Djemâ, 2018).

Comme Poux et al. (2018) l'expliquent, les nuages de points sont employés dans de nombreuses techniques de modélisation 3D utilisées et nécessaires dans de nombreux domaines : architecture, construction, ingénierie, analyse de risque, gestion de bâtiment, robotique, mobilité, ainsi que, par exemple, dans l'industrie du divertissement. Leur utilisation permet de mettre en œuvre différentes simulations, de calculer des plans d'évacuation, de tester et contrôler l'avancée de la construction de bâtiments, ou encore de réaliser des représentations virtuelles d'éléments divers, etc.

Si ces nuages de points sont tant demandés, c'est qu'ils présentent des avantages certains. Ils permettent de collecter rapidement et précisément des nuages de points de taille importante (Tang et al., 2010 ; Hong et al., 2015). Ils constituent un moyen simple et pourtant puissant de représenter des objets et leurs caractéristiques associées (position, orientation, géométrie, etc.) (Grilli et al., 2017). Ils forment ainsi des ensembles de données exhaustifs sur lesquels de nombreux algorithmes et méthodes peuvent être appliqués notamment en ce qui concerne l'extraction d'informations (Poux et al., 2017a). Ils sont aussi, comme expliqué ci-dessus, compatibles avec d'autres capteurs, ce qui permet d'élargir les types d'information récoltés.

Toutefois, les nuages de points possèdent aussi un certain nombre de défauts comme Hong et al. (2015) le rappellent : ils sont sensibles aux occlusions, très lourds en termes de stockage (volume de données important) et la manipulation de ces données est fastidieuse. Nurunnabi et al. (2016) ajoutent que les nuages de points 3D sont souvent épars, non-organisés et peuvent contenir différents types de données aberrantes et de bruits. Enfin, les nuages de points sont qualifiés par Poux et al. (2017a) de complexes et hétérogènes.

Différents formats de fichiers (propriétaire ou open source) sont utilisés pour stocker les nuages de points. Dans le cadre de ce travail, nous utiliserons principalement le format « .las ». Il s'agit d'un format d'échange de données LiDAR mis au point par l'ASPRS (American Society for Photogrammetry and Remote Sensing). L'avantage de ce format est qu'il permet de conserver un grand nombre d'informations par point mais aussi concernant le levé. Le stockage de ces informations est réalisé via une structure binaire offrant une lecture et une importation efficace. On y retrouvera par exemple : l'étendue des données, les données de vol, le nombre de points, les coordonnées de chaque point, l'intensité de chaque point, des valeurs de classification des points, des données utilisateurs, etc. Différentes versions de ce format existent. (Esri, 2018)

II.1.1.2. Segmentation et classification

Ces deux opérations sont proches, dépendantes et donc souvent traitées et réalisées de paire. Grilli et al. (2017) définissent ces deux opérations séparément : la segmentation consiste à rassembler les points en sous-ensembles ayant un certain nombre d'attributs communs ; la classification, elle, correspond à la définition de classes et à l'utilisation de critères pour attribuer les points à ces classes. Les auteurs parlent aussi de segmentation sémantique pour désigner la combinaison de ces deux opérations.

Nurunnabi et al. (2016) abordent la segmentation de surfaces au sein des nuages de points et la définissent comme la séparation et la classification de points en un certain nombre de groupes ou de régions distincts correspondant chacun à une surface d'un objet. Ils utilisent donc le terme de segmentation pour englober les deux opérations. Ils ajoutent aussi que c'est une tâche complexe de par les différents défauts qui peuvent caractériser le nuage (présentés précédemment), notamment le manque potentiel d'informations. Ces défauts, principalement les données aberrantes et le bruit, peuvent rendre les résultats de segmentations inexacts et inconsistants.

Dans le même ordre d'idées, Nguyen & Le (2013) définissent la segmentation comme *"the process of classifying point clouds into multiple homogeneous regions, the points in the same region will have the same properties"*. Ils ajoutent aussi que chacune des régions créées doit avoir du sens, et que de tels segments sont utiles à l'analyse d'une scène ou encore à sa classification. Ainsi, cette définition se rapproche de celle émise par Grilli et al. (2017). Les auteurs reconnaissent aussi la complexité de cette opération, notamment due à la désorganisation et l'hétérogénéité, en termes de densité, du nuage.

Enfin, concernant ces opérations, et plus généralement l'interprétation d'un nuage de points, Poux et al. (2016) soulignent que ce sont des processus nécessitant une analyse et des connaissances spécifiques. Ils sont aussi très contextuels, chronophages, sujets aux erreurs et peuvent engendrer des pertes d'informations. Les auteurs proposent aussi une définition générale de la classification : elle consiste à déterminer les observations situées au sein d'un intervalle d'extension non seulement spatiale mais aussi sémantique.

Les méthodes de segmentation et de classification peuvent être séparées en plusieurs grands types selon les principes sur lesquels elles se basent. Nous allons détailler ces grands types et présenter différents travaux liés à ce sujet par après.

II.1.2. **« Connaissances métier structurées »**

Ce que nous appelons « connaissances métier structurées » englobe en certain nombre d'éléments. Il s'agit de toutes les informations (sémantiques, spatiales, statistiques, …) qui sont associées aux objets traités et qui sont enregistrées sous des formats structurés. Cela peut être : des plans d'intérieur (scannés ou dessinés à l'ordinateur), des photographies, des cartes techniques, des modèles 3D, des statistiques de densité, etc. Dans notre cas, ces informations sont liées à l'utilisation de nuages de points 3D d'intérieur de bâtiments.

Nous utiliserons dans ce travail des plans d'intérieur dessinés à l'ordinateur (CAD, Computer Aided Design) comme source d'informations sémantiques. Ces plans seront enregistrés sous le format structuré « .dxf » pour Drawing eXchange Format. Il s'agit d'un format propriétaire d'échange de fichiers de dessins, associé au logiciel AutoCAD. Les fichiers « .dxf » peuvent être soit sous forme binaire (plus petits et d'utilisation rapide) soit ASCII (lisibles et facilement modifiables). Il existe différentes versions de ce format. (Autodesk Help, 2019).

Nous retrouverons, tracés sur ces plans d'intérieur, les éléments importants d'un bâtiment tels que les murs, les portes, les fenêtres, etc. Ces éléments y seront représentés sous la forme de lignes, polylignes, polygones, etc., positionnés dans un système de coordonnées en deux dimensions. Ces lignes, polylignes, etc. formeront les entités du dessin. Chacune de ces entités est caractérisée par différents attributs (spatiaux, visuels, etc.) et notamment par son appartenance à un calque.

Un calque (ou couche) d'AutoCAD constitue, en quelque sorte, une version informatique d'un calque réel. Il s'agit d'une structure permettant d'enregistrer des entités du dessin possédant des caractéristiques sémantiques communes et ainsi d'organiser le plan. A la manière d'une feuille de style dans un logiciel de traitement de texte, un certain nombre de caractéristiques propres à ce calque peuvent être spécifiées : nom, couleur des entités, type de ligne, épaisseur de ligne, etc. Par exemple, un calque nommé « Mur » pourrait reprendre toutes les entités utilisées dans le plan pour représenter des murs, et les afficher en lignes rouges continues. De la même manière qu'avec des feuilles de calque transparentes, en superposant les différents calques AutoCAD dans lesquels les éléments du plan ont été enregistrés, on peut visualiser le plan en entier (cf. Fig. 3). La représentation d'un calque peut être activée ou non, de sorte à visualiser et manipuler ou non les éléments de ce calque. (Autodesk Help, 2019)

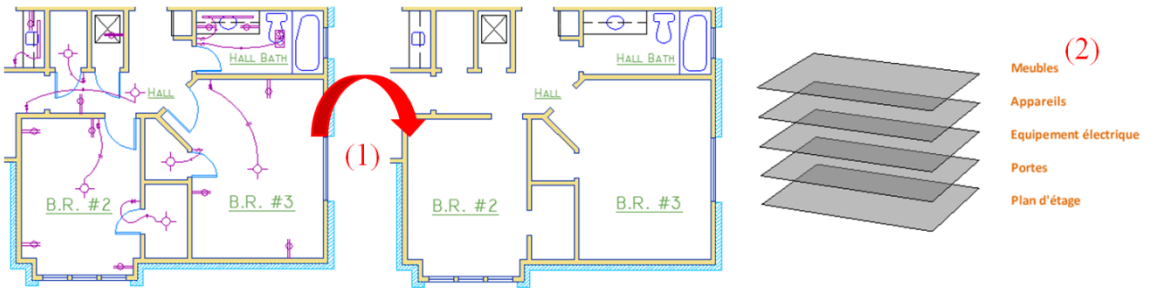


Figure 3 Utilisation (1) et organisation (2) des calques AutoCAD d'un plan d'intérieur : masquage des calques contenant les portes et l'équipement électrique repris sur le plan (1). (Autodesk Help, 2019, modifiées)

II.1.3. **Notions importantes associées : le BIM**

Parmi les travaux présentés par la suite, beaucoup s'inscrivent dans un thème plus large que la segmentation-classification de nuages de points. Ce thème correspond à la création et le développement de BIM ou Building Information Models. Comme l'expliquent Rajala & Penttilä (2006) et Tang et al. (2010), ces BIM permettent notamment aux professionnels du secteur AEC (Architecture, Engineering and Construction) de gérer, stocker et échanger, de manière informatisée, diverses informations liées à un bâtiment. Le BIM est, en effet, un modèle 3D fournissant une représentation sémantiquement riche d'un bâtiment. Les éléments y sont identifiés et représentés en 3D et de manière volumique (un mur est un bloc possédant plusieurs surfaces). Les relations de voisinage entre entités y sont aussi représentées et d'autres informations y sont stockées : matériaux, coût, etc. Le BIM peut constituer une représentation du bâtiment « tel que construit » (« as-built ») ou « tel qu'existant » (« as-is »). (Tang et al., 2010). Ces BIM sont donc très utiles, voire incontournables, pour ce qui est de la gestion des différentes phases de la construction d'un bâtiment (de la conception à la construction). Ils le sont aussi pour leur gestion, maintenance ou rénovation : détection d'erreurs, contrôles de qualités ou énergétiques, gestion de l'espace, simulations, etc. (Xiong et al., 2013 ; Hong et al., 2015 ; Anagnostopoulos et al., 2016 ; Macher et al., 2017).

Les nuages de points sont une des sources principales d'information pour ces BIM au vu de leurs nombreux avantages (rapidité, précision, quantité, 3D etc.). La transition entre le nuage et le BIM pose cependant problème. Ce processus est majoritairement manuel, délicat, chronophage et subjectif. Il est compliqué par les volumes importants de données à traiter, par la présence d'occlusions et par le manque d'informations sémantiques au sein des nuages. Ainsi, un grand nombre de recherches se concentrent sur l'automatisation de ce processus transitoire appelé "scan-to-BIM process". (Tang et al., 2010 ; Macher et al., 2017)

II.2. Techniques et algorithmes existants

II.2.1. **Segmentation - Classification**

Nguyen & Le (2013) distinguent cinq types de segmentations. Ils commencent par les segmentations basées sur la détection de bords. Ces méthodes permettent des segmentations rapides en se basant sur la détection des bords de différentes parties du nuage. En effet, les bords caractérisent la forme des objets et peuvent être détectés en repérant des changements importants au sein des caractéristiques des points (intensité, direction de la normale, …). Ces techniques sont sensibles au bruit et à l'hétérogénéité des nuages et sont donc peu précises.

Le second type de méthodes présenté par Nguyen & Le (2013) est celui dit « Region based ». Ces méthodes sont moins sensibles au bruit et vont comparer les points voisins pour former des régions en fonction de leurs similarités ou dissimilarités. Il existe deux approches : par « graine » (bottom-up) ou non (top-down). Dans le premier cas, les comparaisons démarrent depuis quelques « graines » et se propagent dans le nuage, ajoutant à la région les points satisfaisant les conditions de similarité. Le choix des points de départ aura un impact sur la segmentation en engendrant plus ou moins de sur- ou sous- segmentations. Le deuxième cas démarre d'une région et la subdivise en sous-régions tant que certaines conditions sont respectées. Le choix de ces conditions de subdivision est compliqué et peut engendrer de la sur-segmentation. Ce deuxième cas nécessite aussi de bonnes connaissances a priori.

Pour la 3ème classe, Nguyen & Le (2013) détaillent les méthodes basées sur le calcul d'attributs. Il s'agit de méthodes robustes, précises et fournissant des résultats homogènes. Des attributs sont calculés pour les points du nuage (en prenant en compte leur spatialité) et servent de critère de segmentation. Le calcul de ces attributs doit être très précis pour fournir de bonnes segmentations. Ce sont des méthodes souvent chronophages et qui dépendent de la définition des relations de voisinage et de la densité de points au sein du nuage.

L'avant dernière classe aborde les méthodes basées sur l'ajustement de modèles, c'est-à-dire segmentant des régions du nuage en leur ajustant des primitives géométriques (rectangles, cylindres, …). L'algorithme RANSAC (Random Sample Consensus) (Fischler & Bolles, 1981) est le plus connu et permet de détecter des éléments tels que des lignes droites et des cercles. Schnabel et al. (2007) l'ont notamment repris et optimisé pour segmenter les nuages de points. Ces méthodes sont purement mathématiques et robustes par rapport aux valeurs aberrantes mais parfois peu précises. (Nguyen & Le, 2013)

Enfin, Nguyen & Le (2013) finissent en présentant les méthodes considérant les nuages de points comme des graphes (« Graph-based »). Par exemple, chaque point du nuage correspond à un sommet et les bords relient différents points voisins. De nombreux algorithmes de ce type utilisent notamment les Conditional Random Fields (CRF). Ces méthodes sont précises et efficaces notamment dans des nuages bruités et hétérogènes. Un problème lié à ces méthodes est qu'elles nécessitent d'être entrainées.

A ces différents types de segmentation, Grilli et al (2017) ajoutent les segmentations hybrides, qui combinent des techniques différentes pour profiter des avantages de chacune, ou encore les segmentations par deep learning. Ces dernières utilisent l'intelligence artificielle pour prendre les décisions de segmentation en se basant sur des données d'entrainement. Les auteurs séparent aussi les méthodes de classification selon trois approches : supervisée, non-supervisée ou interactive. Dans le premier cas, l'utilisateur fournit les classes alors que, dans le second, elles sont auto-déterminées. Le dernier cas requiert une implication importante de l'utilisateur mais fournira de meilleurs résultats.

De nombreuses méthodes de segmentation de nuages de points ont été publiées et nous allons en détailler quelques-unes. Nurunnabi et al. (2016) présentent, par exemple, un algorithme par « region growing » (Region based) basé sur une analyse en composantes principales (ACP). Il intégre des conditions de minimisation des distances et de maximisation de la cohérence. Cet algorithme est plus rapide et plus précis que les méthodes basées uniquement sur RANSAC ou sur des ACP normales et résiste mieux aux valeurs aberrantes et au bruit.

Hermans et al. (2014), utilisent la labélisation d'une scène intérieure en 2D à partir d'images RGB-D (D pour « depth map » ou carte de disparité) pour segmenter et classer la même scène représentée dans un nuage de points 3D. Les auteurs vont donc utiliser la segmentation sémantique réalisée en 2D et l'appliquer via des champs aléatoires conditionnels (CRF) au nuage de points. Cette méthode se rapproche du but poursuivi dans ce travail : segmenter un nuage de points 3D à partir d'une source d'informations externes 2D.

Nan et al. (2012) ont mis au point une approche dite « search-classify » permettant de comprendre et modéliser une scène en intérieur depuis un nuage de points. Le principe de cette approche est de réaliser une succession de segmentations et de classifications en utilisant comme classificateur (entrainé auparavant) la probabilité qu'un segment appartienne à un objet. La méthode des auteurs sera aussi affinée en ajustant des modèles d'objets de la classe aux segments obtenus.

Armeni et al. (2016) ont développé une méthode de décomposition sémantique des nuages de points de grande échelle (bâtiments entiers). Tout d'abord, ils décomposent hiérarchiquement leur nuage en sous-nuages sémantiquement pertinents en se basant sur la détection de vides. Par exemple, un mur correspond à un vide limité par deux surfaces. Cette configuration formera une structure particulière (« peak-gap-peak ») au sein des histogrammes de densités. Les murs sont donc détectés et segmentés grâce à cette structure. Les auteurs abordent ensuite le reste de la segmentation comme un problème de détection et utilisent la récurrence caractérisant les espaces intérieurs. La détection prend en effet en compte l'objet dans son ensemble, plutôt que chaque point individuellement. Elle est donc moins sensible aux occlusions. Les auteurs utilisent des ensembles de boites 3D de référence propres à chaque classe (leurs caractéristiques dépendent de la classe) ainsi qu'un classificateur entrainé sur des données spécifiques. Des ensembles de boites candidats sont alors déterminés dans le nuage et comparés via le classificateur à l'ensemble de référence associé à chaque classe. Le classificateur décide alors, en fonction de la comparaison, de classer ou non le candidat testé. Enfin, les auteurs utilisent une approche graphe afin d'affiner les résultats.

Poux & Billen (2019) proposent une méthode basée sur l'utilisation de voxels, permettant de caractériser précisément et robustement le nuage de points. Ils déterminent ainsi des attributs de deux grands types : liés à la forme (basés sur les X, Y, Z des points) et liés aux connections entre éléments (basés sur les relations topologiques entre voxels). Cette méthode est notamment résistante au bruit, aux occlusions et aux variations de résolution, et fournira un support solide pour les classifications. Les auteurs détaillent aussi la mise en œuvre des attributs déterminés par leur algorithme au sein d'une méthode de segmentation sémantique opérant selon une approche de type « graph based » (5ème classe de Nguyen & Le (2013)). La voxelisation, comme l'expliquent les auteurs, correspond à une méthode de subdivision de l'espace en un ensemble de volumes (voxels) d'une certaine taille. Plusieurs niveaux de subdivisions engendrant des voxels plus au moins petits peuvent être mis en oeuvre. Cette voxelisation est liée au principe d'indexation spatiale. Pour faire l'analogie avec le 2D, le voxel pourrait s'apparenter à un équivalent tridimensionnel du pixel.

Poux & Billen (2019) vont aussi comparer leur méthode de segmentation aux méthodes par « deep learning » les plus efficaces. Parmi ces méthodes citées et utilisées, nous retrouvons celle de Engelmann et al. (2017), celle de Qi et al. (2017) présentant leurs réseaux PointNet, ou celle de Landrieu & Simonovsky (2018) utilisant le deep-learning pour déterminer l'organisation d'un nuage sous une structure dite « Superpoint Graphs ».

Pour continuer avec les méthodes utilisant des classificateurs entrainés, Xiong et al. (2013) soumettent une méthode d'extraction des murs, sols, plafonds et ouvertures importantes (portes et fenêtres). Elle se déroule en quatre étapes : voxelisation, extraction de surfaces planes depuis les voxels (par un algorithme « région growing »), classification par algorithme de machine learning, et enfin, affinage des résultats. Les auteurs prennent aussi fortement en compte les problèmes d'occlusion et de désordre présents dans les nuages. En effet, leur but est de mettre au point une méthode robuste à ces problèmes afin qu'elle puisse être utilisée dans la modélisation automatique de batiments. Leur travail s'inscrit effectivement dans la lignée des recherches sur l'automatisation du processus « scan-to-BIM ».

Poux et al. (2017a) ont aussi développé une méthode de segmentation hybride se basant notamment sur des attributs descriptifs calculés dans le nuage (spatiaux et sémantiques, notamment la couleur). En plus de ces attributs, leur méthode intègre de la reconnaissance de formes (RANSAC), les principes de propagation par région, ou encore l'utilisation de voxels. Cette méhode est appliquée dans le cadre de l'extraction d'informations archéologiques depuis des tesselles formant une mosaïque.

Afin d'améliorer la modélisation de modèles « as built », Hong et al. (2015) proposent une approche semi-automatique comprenant plusieurs segmentations. Tout d'abord, ils utilisent une structure 2D constituée de pseudo-points pour simplifier le nuage. La segmentation du sol et du plafond se base sur la distribution des hauteurs au sein de ces pseudo-points et sur l'algorithme RANSAC mentionné précédemment. La segmentation des murs se base, elle, sur une projection horizontale du nuage permettant d'en retracer les limites. Ces limites seront affinées par la suite (regroupement selon leurs caractéristiques).

Anagnostopoulos et al. (2016) présentent un travail aussi lié au processus « scan-to-BIM » et ayant pour but d'extraire les plans correspondant aux sols, plafonds et murs pour en récupérer de l'information contextuelle. Leur algorithme se base sur l'hypothèse « Manhattan-World Buildings » (Coughlan & Yuille, 1999) qui suppose que les bâtiments sont organisés selon trois directions orthogonales entre elles et que leurs composants sont liés par certaines relations. Les auteurs utilisent aussi l'agorithme RANSAC de Schnabel et al. (2007) pour segmenter les surfaces. Celles-ci seront projetées dans des plans XZ et YZ, de manière à les séparer en différentes classes selon leur orientation par rapport à ces plans. Une indexation spatiale (octree) et des « bounding box » sont aussi employés.

Previtali et al. (2014) se sont eux intéressés à la détection et modélisation de surfaces planes au sein des BIM. Leur méthode permet de déterminer les sols, plafonds et murs dans un nuage de points mais aussi de résoudre des problèmes de bruit et d'occlusion par détection d'ouvertures. Ils commencent par détecter les surfaces présentes dans le nuage en utilisant l'algorithme mis au point par Previtali et al. (2013) et basé sur l'algorithme RANSAC modifié (itératif et gérant les erreurs de sur-/sous- et fausse classification). Grâce à la hauteur et l'orientation des surfaces détectées, les auteurs segmentent les sols et plafonds. Les murs sont segmentés par projection des points sur un plan horizontal discrétisé en cellules carrées. Enfin, ils utilisent un algorithme de « ray tracing » se basant sur la distance scanner-point pour distinguer le bruit et les ouvertures présents dans le nuage.

Valero et al. (2012) s'intéressent à l'automatisation des opérations faisant partie du processus « scan-to-BIM ». Ils présentent une méthode permettant de représenter les limites des scènes intérieures scannées. Parmi les étapes, la segmentation des murs, sols et plafonds est abordée. Les auteurs proposent une méthode de segmentation mettant en œuvre une voxelisation et basée sur la densité au sein de ces voxels. Dans le même sujet, Macher et al. (2017) présentent une méthode de segmentation de ces mêmes éléments, employée pour reconstruire automatiquement des scènes intérieures (à intégrer dans des BIM). Cette méthode utilise la distribution des points selon l'axe de Z pour distinguer les sols et plafonds. Les murs sont détectés en réalisant des coupes horizontales dans le nuage, à proximité du plafond et sont segmentés sur base des bords ainsi déterminés. Les auteurs utilisent aussi des zones tampons pour affiner la segmentation de ces murs. Les segments sont alors regroupés selon leur classe en nuages qui seront utilisés pour reconstruire un modèle 3D.

Enfin, Budroni & Boehm (2010) présentent un algorithme pour reconstruire automatiquement, depuis un nuage de points, le modèle CAD 3D d'une scène intérieure. La segmentation mise au point dans leur algorithme se fait en deux étapes correspondant à l'extraction de plans verticaux et celle de plans horizontaux. Ces plans serviront à déterminer les murs, sols et plafonds du nuage. Cette extraction se fait par balayages linéaires (le long d'un vecteur normal) et rotationnels (autour d'un axe) d'un plan dans le nuage. Les surfaces planes sont donc, en quelque sorte, déterminées par « ajustement de modèles ». Sur base de certaines hypothèses associées aux classes, les surfaces obtenues sont testées et attribuées ou non à une classe. Les segments classés obtenus serviront à reconstruire le modèle 3D CAD.

II.2.2. **Nuages de points 3D et plans d'intérieur**

Il est aussi pertinent de s'intéresser aux méthodes de segmentation existantes et utilisant, comme nous, des plans d'intérieur comme sources d'informations. Nous avons notamment pu trouver quelques travaux sur le sujet.

Tout d'abord, Gimenez et al. (2015) réalisent un examen des différents outils et méthodes qui peuvent permettre de générer des modèles 3D de bâtiments (BIM) à partir de plans 2D. Après avoir passé en revue les méthodes plus classiques de créations de BIM (photographies aériennes, laser scan 3D, …), les auteurs détaillent les méthodes de générations de BIM depuis différents types de documentation 2D : dessins, plans scannés et plans CAD. Les dessins peuvent être traités comme des images (dessin papier) ou comme des séries temporelles s'ils sont dessinés l'aide d'outils digitaux (les mouvements successifs sont analysés par les logiciels). Les plans papiers peuvent, eux, être scannés, digitalisés et traités comme images. Pour être utilisées, les informations pertinentes devront être d'abord détectées et extraites. Enfin, concernant les plans CAD, les auteurs mentionnent différents avantages associés à leur utilisation : le fait que les éléments soient composés de primitives géométriques, l'organisation en calques séparant les différents types d'objets ou encore la précision et la netteté des entités fournies. Ils spécifient aussi qu'il existe des solutions pour convertir les plans CAD en BIM (Lewis & Séquin, 1998) mais que ce type de plan est rarement disponible pour les vieux bâtiments.

Lewis & Séquin (1998), présentent une approche semi-automatique permettant de créer un modèle 3D d'un bâtiment sur base de plans d'étage dessinés à l'ordinateur. Ils précisent, en effet, que « *The existence of such floor plans should be considered a significant investment* » (Lewis & Séquin, 1998) et qu'un tel plan est un bon point de départ pour la création d'un modèle 3D d'un bâtiment. Leur méthode se base sur ces plans et sur quelques interactions de l'utilisateur. Leur système, appelé « Building Model Generator », prend un plan d'intérieur 2D au format « .dxf » et le convertit selon une structure de données spécifique. Ce plan est alors analysé afin d'extraire de l'information sémantique. Cette information permet de reconstruire les murs et d'y ajouter les ouvertures aux endroits adéquats. Le modèle est ensuite ajusté à la main. Des escaliers sont aussi ajoutés lors de la phase de combinaison des différents étages, permettant de reconstruire le modèle 3D total. On notera bien sûr que, si la méthode se base bien sur des plans CAD, elle n'implique ni nuage de points, ni segmentation. La méthode d'extraction d'informations reste toutefois intéressante et pertinente.

Dans le cadre de l'extraction et l'utilisation de données issues de plans CAD, nous pouvons aussi mentionner le travail de Huang et al. (2008) qui propose une méthode de transformation automatique de dessins architecturaux en informations topologiques et spatiales liées à un bâtiment. Cette méthode va utiliser la théorie des graphes pour traduire le sens de la représentation des différents éléments au sein du plan.

Nous pouvons aussi citer la méthode mise au point par Okorn et al. (2010) qui automatise la création de plan 2D depuis un nuage de points. Ils utilisent notamment des histogrammes de hauteurs pour déterminer le sol et le plafond. Les murs et ouvertures sont détectés en projetant le reste des données dans un plan 2D et en appliquant la transformée de Hough sur les histogrammes de densités découlant de cette projection. Les auteurs réalisent donc la transformation inverse en passant du nuage 3D au plan 2D.

Nous avons aussi pu remarquer que, souvent, lorsque le terme CAD était associé aux nuages de points, il était fait mention de modèles CAD 3D (Ex. : Budroni & Boehm (2010)). Ces modèles s'apparentent aux BIM et sont notamment comparés aux nuages de points pour gérer et inspecter l'avancement de travaux (Kim et al., 2013 ; Nguyen & Choi, 2018).

II.3. Problèmes et difficultés

Comme nous avons pu le voir, les nuages de points ont quelques défauts. Parmi ceux-ci, les occlusions, la désorganisation et le bruit sont souvent cités par les auteurs. Que ce soit pour de l'extraction d'informations, de la segmentation, ou de la reconstruction de modèles 3D, la plupart des méthodes développées vont en tenir compte et essayer de s'en affranchir au maximum. Par exemple, Nurunnabi et al. (2016), constatant que les segmentations basées sur des ACP sont sensibles aux valeurs aberrantes, proposent une méthode robuste par rapport à ces valeurs. Nan et al. (2012), eux, détaillent une méthode permettant d'effectuer des reconstructions 3D d'environnements intérieurs malgré des problèmes d'occlusions et de bruit. Previtali et al. (2014) ont, eux aussi, développé une méthode répondant à ces mêmes besoins et problèmes. Pour continuer avec les occlusions, afin de réduire leur impact sur la segmentation, Armeni et al. (2016) abordent cette segmentation comme un problème de détection. Enfin, Poux & Billen (2019) présentent une méthode de segmentation sémantique résistant au bruit, aux occlusions et aux variations de résolution. D'autres exemples pourraient encore être énoncés.

Un autre problème, régulièrement abordé et pris en compte par les auteurs, découle de la taille des données traitées. En effet, plus le volume de données est important, plus les traitements seront chronophages. Diverses techniques sont ainsi utilisées par les chercheurs pour réduire le nombre de données à analyser et/ou pour accélérer les traitements.

Un premier moyen, assez brut, est de réduire simplement le nombre de données en n'en conservant qu'une partie. Le problème est que cet échantillonnage engendre des pertes d'informations. Une autre possibilité est l'utilisation de voxels. Cette voxelisation, notamment utilisée par Poux & Billen (2019), est décrite précédemment. Elle est assez bien utilisée, comme nous pouvons le constater, dans les travaux détaillés ci-dessus (Valero et al., 2012 ; Xiong et al., 2013 ; Poux et al., 2017a). Des structures semblables à une voxelisation sont aussi parfois mises en œuvre : Hong et al. (2015) utilisent, par exemple, une structure 2D de pseudo-points. Il s'agit en quelque sorte d'une voxelisation mais dans laquelle les voxels ne sont créés que s'ils contiennent un ou plusieurs points.

Des méthodes d'indexation spatiale peuvent aussi être appliquées aux nuages. Une indexation spatiale consiste en une subdivision récursive de l'espace, associée à un arbre décrivant la structure engendrée par ces divisions. C'est par exemple ce que réalisent Anagnostopoulos et al. (2016) dans leur travail. Un autre exemple est celui de Hackel et al. (2016) qui ont mis au point une méthode de segmentation permettant de s'affranchir de la désorganisation, de l'hétérogénéité et du volume des nuages. Pour ce faire, ils se basent sur une gestion soignée des relations de voisinage, grâce à l'utilisation d'un KD tree (un type d'index spatial). Nous pouvons aussi souligner que la voxelisation mise en place par Poux & Billen (2019) se base sur une indexation par octree (subdivision récursive de l'espace en huit parties).

Nous avons aussi pu mentionner des méhodes de segmentation par deep learning. Celles-ci constituent parfois des alternatives à la création de voxels, comme par exemple la méthode présentée par Qi et al. (2017).

CHAPITRE III. HYPOTHÈSES DE RECHERCHE

Suite à l'état de l'art, nous avons pu mettre en évidence les avantages et les inconvénients des nuages de points mais aussi ceux de l'opération de segmentation sémantique d'un nuage de points. Celle-ci, lorsqu'elle est réalisée manuellement est délicate, subjective, chronophage et complexe. Elle nécessite aussi un apport en informations sémantiques bien souvent non disponibles dans les données brutes. Nous avons aussi détaillé différents travaux récupérant et utilisant de l'information dans le but d'automatiser cette étape.

En outre, nous avons noté que les plans d'intérieur au format CAD constituaient des sources d'informations pertinentes (spatiales et sémantiques) relatives aux éléments de l'environnement représenté. Ces informations sont structurées au sein du plan notamment via des calques. Un travail a aussi déjà été réalisé pour créer le modèle 3D d'un bâtiment à partir de plans CAD mais sans faire intervenir de nuages de points ou de segmentations (Lewis & Séquin, 1998). Néanmoins, peu de méthodes mettent en oeuvre l'usage de plans CAD 2D d'intérieur dans la segmentation sémantique de nuages de points.

Pour rappel, notre question-problème est la suivante : « Est-il possible de développer une méthode automatique et efficace (en termes de rapidité, précision et exactitude) de segmentation et classification d'un nuage de points 3D d'intérieur de bâtiment, à partir d'informations sémantiques et spatiales extraites du plan CAD associé à cet environnement ? »

Nous pouvons formuler deux hypothèses découlant de cette question :

- La première est : « Il est effectivement possible d'extraire de l'information sémantique et spatiale depuis le plan d'intérieur au format CAD, et d'utiliser ces informations dans une opération de segmentation sémantique d'un nuage de points d'intérieur représentant le même environnement que le plan » ;

- La seconde est : « Cette méthode de segmentation sémantique est plus rapide qu'une méthode manuelle et va fournir des résultats aux moins aussi précis et exacts que cette méthode manuelle ».

L'information sémantique extraite de ces plans et intégrée dans notre méthode permettra, en effet, de réduire le temps de traitement et d'assurer un résultat précis et exact.

CHAPITRE IV. MÉTHODOLOGIE

Ce chapitre présente la méthodologie appliquée pour le développement et la mise en œuvre de ce travail. Cette méthodologie se veut générale et aisément reproductible dans d'autres circonstances. L'environnement de travail ainsi que les données et logiciels utilisés sont présentés en fin de chapitre. Cela permettra de poser clairement le contexte de notre travail avant d'aborder les résultats concrets obtenus. Les différentes grandes parties de la méthodologie sont traitées dans un ordre chronologique de bonne procédure. Il s'agit de l'ordre dans lequel il faut exécuter ces opérations en cas de reproduction de cette méthodologie. Cet ordre ne correspond pas nécessairement à l'ordre dans lequel ces opérations ont été exécutées au cours de notre travail.

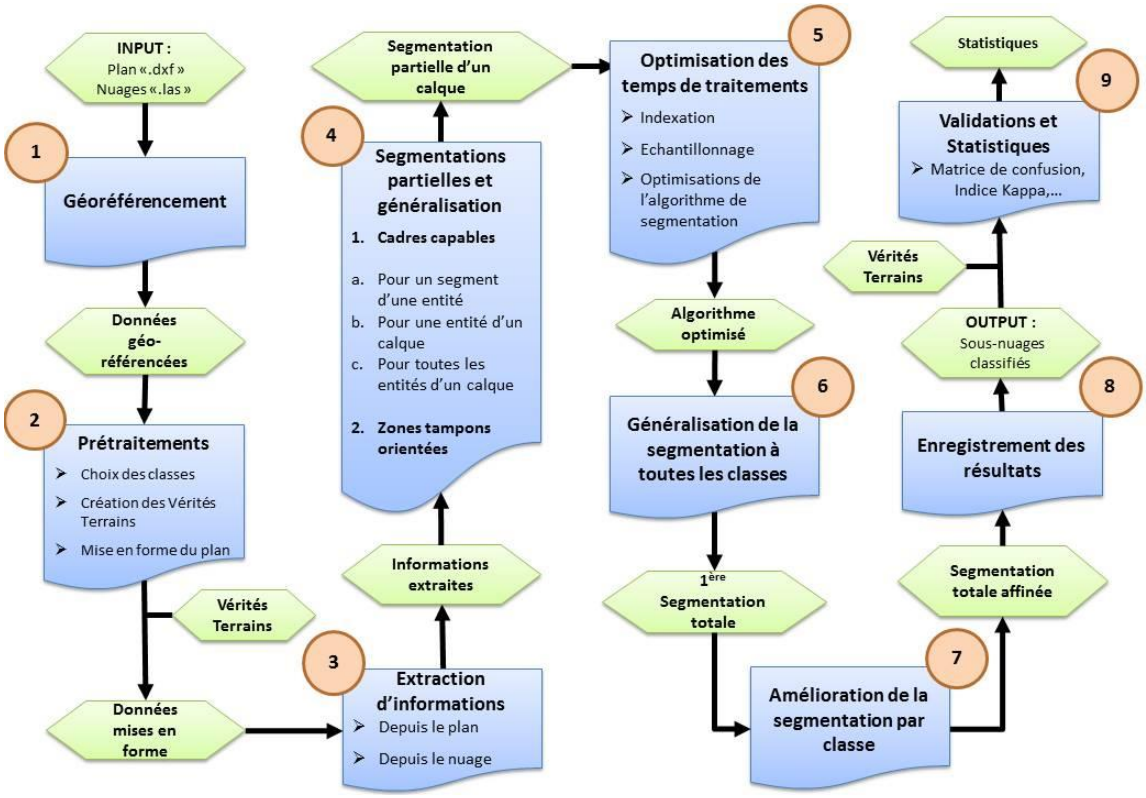


Figure 4 Schéma du déroulement de la méthodologie

Avant de commencer la description de la méthodologie, abordons brièvement une étape préalable : la prise en mains du sujet. Cette prise en mains s'effectue dans notre cas grâce à la visualisation et la manipulation de plans d'intérieur et de nuages de points. Par exemple, nous avons segmenté manuellement différents nuages. Cette étape favorise donc une meilleure compréhension du sujet, des problèmes associés et des solutions envisageables pour les résoudre.

Rappelons aussi que les données utilisées dans ce travail comme données de départ (inputs) sont :

- d'une part, un plan au format « .dxf »
- d'autre part, un nuage de points au format « .las »

IV.1. Géoréférencement

Tout d'abord, le nuage et le plan doivent partager un même système de coordonnées afin que les intersections et autres opérations entre les éléments des deux fichiers puissent être réalisées. Ce référencement est donc indispensable : si la mise en correspondance entre éléments du plan et du nuage n'est pas possible, l'algorithme ne fonctionnera pas.

Le géoréférencement consiste à appliquer une transformation aux différentes données utilisées par l'algorithme afin de les exprimer toutes dans un système de coordonnées unique (identique donc pour les deux fichiers). Pour ce qui est du choix de système, nous utiliserons celui du nuage puisque nous travaillons en 3D et que le système du plan est en 2D.

Il faut donc transposer le plan « .dxf » dans le système du nuage. Cette opération correspond à une transformation de Helmert. Cette transformation, dans le cas 2D, consiste en un système d'équations à 4 inconnues à résoudre : un facteur d'échelle (k), deux translations (t_x, t_y) et un angle de rotation (θ). Le système d'équations comprend aussi deux points du système 2D source ((X_1,Y_1) et (X_2,Y_2)) et les points correspondants dans le système de destination ((E_1,N_1) et (E_2,N_2)) (cf. Fig 5). Ces données vont nous permettre de résoudre le système et d'obtenir les équations de transformation permettant de passer du système source (plan) au système destination (nuage). (Billen, 2016).

Il s'agira bien, dans notre cas, d'une transformation 2D puisque le plan « .dxf » ne possède pas de dimension Z. Il faudra donc s'assurer que les points spécifiés dans le nuage (E,N) sont bien situés à une même hauteur pour former un plan perpendiculaire à l'axe Z. Ainsi, après la transformation, le plan « .dxf » sera lui aussi perpendiculaire à l'axe Z. Par exemple, nous utiliserons comme hauteur de référence celle à laquelle sont généralement réalisés les plans d'intérieur : 1,3m par rapport au sol.

Le résultat obtenu à la fin de cette étape est un système de coordonnées 3D unique, dans lequel sont exprimées les coordonnées des objets du plan et du nuage.

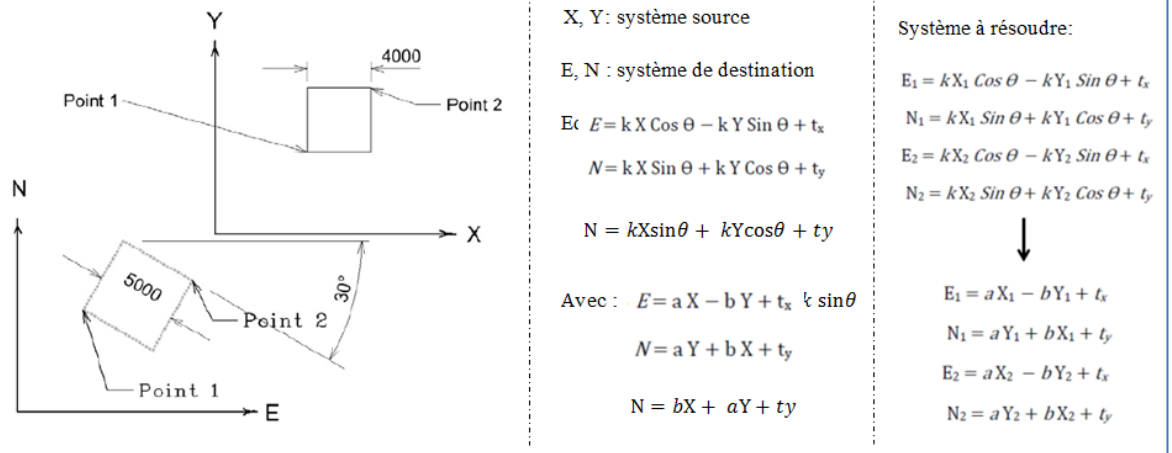


Figure 5 Schéma, équations de départ et système d'équations à résoudre pour réaliser une Transformation de Helmert 2D (Billen, 2016 ; modifiées)

IV.2. Prétraitements

Avant de commencer à extraire de l'information depuis le plan et le nuage, il faut réaliser certains prétraitements. Ces prétraitements vont éventuellement modifier les données pour les faire correspondre à un format précis. Ils sont nécessaires pour assurer l'obtention de résultats corrects avec l'algorithme. L'utilisation de notre algorithme est donc contrainte par ce formatage des données. De plus, ce formatage n'est pas automatisé dans la version actuelle de l'algorithme. Cette limitation sera discutée par la suite (cf. VI.3).

Tout d'abord, il est nécessaire de déterminer notre classification. Le choix de ces classes façonnera le reste des traitements. Pour réaliser ce choix, le plan constitue la première source d'informations à utiliser. En effet, le but du travail étant de réaliser une segmentation basée sur un plan « .dxf », certains calques de ce plan pourront constituer certaines de nos futures classes. D'autres sources d'informations peuvent être consultées pour éventuellement élargir la classification : objets visibles dans le nuage, classifications mentionnées dans la littérature, etc. La classification doit toutefois rester en adéquation avec le sujet (ici, l'utilisation du plan). Par exemple, dans notre cas, les classes doivent correspondre à des éléments représentés sur le plan et non à des meubles ou des décorations.

Une fois la classification choisie, des vérités terrains peuvent être créées en segmentant manuellement et selon cette classification les nuages de points de validation. Cette étape peut permettre aussi de mieux appréhender la manière de segmenter les différentes classes.

Enfin, les plans d'intérieur au format « .dxf » ne sont pas toujours sous une forme optimale pour l'algorithme : calques superflus, entités superflues, entités mal tracées, non reliées, inutilisables, etc. Or, ces problèmes peuvent alourdir les traitements ou engendrer des erreurs dans les résultats. Il faudra donc nettoyer les plans de ces différents défauts afin qu'ils soient utilisables par l'algorithme. De même, si l'algorithme se base sur des noms de calque spécifiques pour différencier les classes, il faudra modifier ces noms en conséquence. Les modifications varieront selon le calque, le type d'objet et ses caractéristiques, la manière de les segmenter, etc.

Suite à ces prétraitements, on obtient plusieurs résultats, notamment : une classification qui guidera la suite des opérations, des vérités terrains pour valider nos résultats et un plan « .dxf » mis en forme. Les traitements et modifications sont illustrés dans le chapitre suivant « Algorithme et Résultats ».

IV.3. Extraction d'informations

Une fois les données prétraitées et prêtes à être employées, le développement de l'algorithme peut commencer. La première étape consiste à importer les données et en extraire les informations importantes. Ces informations sont celles associées à chaque point du nuage (coordonnées, couleurs RVB, tag de classification, etc.) et celles associées aux calques et entités du dessin (nom de calque, type d'entité, coordonnées, etc.). Le type d'informations disponibles pour chaque point du nuage est lié à la version du format «.las » utilisée. Les informations ainsi extraites permettront de développer les méthodes de segmentation de notre algorithme. Le but de l'étape est donc de déterminer, d'extraire et de stocker de manière structurée, les connaissances métier sur lesquelles l'algorithme reposera.

IV.4. Segmentations partielles et généralisation

Avec les informations extraites, les premiers essais de segmentation peuvent être effectués. Ils ont pour but d'obtenir un premier résultat (servant de base pour l'étape suivante) sans se soucier, à ce stade, de la précision et de l'exactitude des résultats ou de l'optimisation du processus.

IV.4.1. **Cadres capables (Bounding Box)**

La première méthode est donc peu précise et inexacte mais aisément implémentable. Le principe est d'utiliser le cadre capable 2D de l'objet traité et de récupérer les points dont les coordonnées planes sont situées entre les limites de ce cadre. Les points sélectionnés sont segmentés et forment un nuage résultat. Les autres points sont utilisés pour reformer un nuage « reste ». Le cadre est caractérisé par deux points : (X_{min}, Y_{min}) et (X_{max}, Y_{max}). Ces points ne font pas nécessairement partie de l'objet mais sont formés des coordonnées limites de l'objet selon les axes X et Y. Les entités du plan étant en 2D, la limite en Z n'est pas fournie et ne sera pas intégrée dans l'algorithme à cette étape du développement.

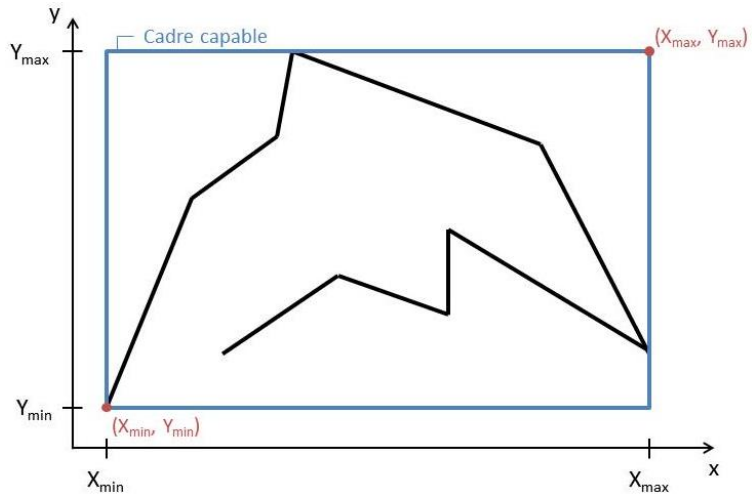


Figure 6 Illustration du cadre capable pour une entité aléatoire du plan

Nous commencerons par tester la méthode sur une partie restreinte du nuage total en essayant de segmenter un objet très spécifique : le segment d'une entité d'un calque. De cette manière, les tests seront rapides et les erreurs plus facilement identifiables. Une fois le code développé dans ces conditions, il est alors plus facile de généraliser. Nous testerons alors le code pour deux niveaux de généralisation supérieurs : l'entité entière d'abord, toutes les entités d'un calque ensuite. De cette manière, on divise et organise les étapes de développement, procédant d'un niveau de généralisation au suivant. Plus tard dans le développement, nous aborderons ainsi le niveau de généralisation global : toutes les entités de chaque calque.

Comme illustré ci-après, cette méthode de développement est aussi appliquée pour la précision et l'exactitude ainsi que pour les temps de traitement associés à la méthode. On évoluera d'une méthode spécifique, peu précise et lente vers des méthodes de plus en plus générales, précises et rapides et ce, par étapes successives.

Ce développement fournira une première méthode de segmentation simple, offrant des résultats relativement corrects, à un niveau de généralisation intermédiaire (pour un calque). Il servira aussi et surtout de point de départ pour les futures optimisations. La première de ces optimisations sera d'améliorer soit la précision et l'exactitude de la méthode soit sa rapidité.

IV.4.2. **Zones tampons orientées (Buffer)**

Dans notre cas, nous commencerons par optimiser sa précision et son exactitude sans prendre en considération le temps de traitement. La méthode de segmentation est donc modifiée de manière à mieux s'adapter aux objets à segmenter. Au lieu du cadre capable, nous utiliserons une zone tampon dessinée autour de l'entité. Cette zone correspond à l'ensemble des points situés en dessous d'une certaine distance seuil (*s*) de l'entité. Ce seuil caractérisera le tampon.

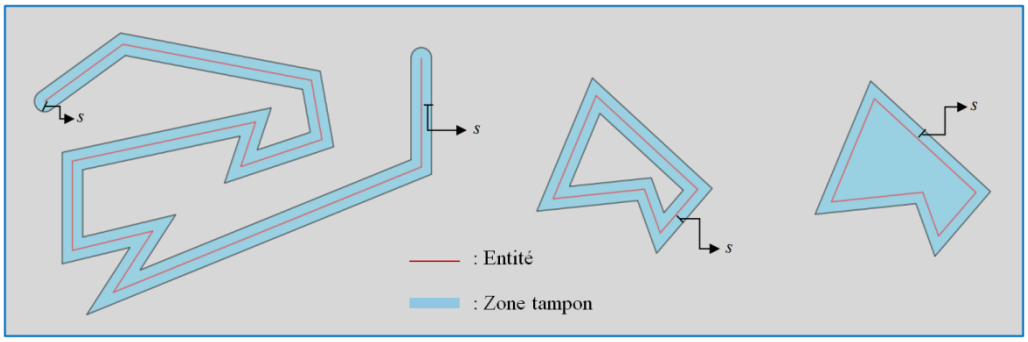


Figure 7 Exemple de zones tampons pour une ligne, le contour d'un polygone et l'ensemble de ce polygone. La fermeture en fin de ligne est ronde (telle qu' utilisée) et les jointures sont angulaires (rondes dans notre cas).

Cette zone tampon a l'avantage d'être orientée en fonction de l'entité et de suivre ses différents segments. Ainsi, la zone étant plus réduite et adaptée à l'entité, la sélection des points est plus juste et plus précise. De plus, les zones tampons sont facilement modulables et donc adaptables aux différentes classes. La création de ces zones peut, néanmoins, allonger le temps de traitement de la méthode. Le principe de l'utilisation de la zone tampon est le même que pour le cadre capable. On va déterminer la zone selon une valeur seuil arbitraire et sélectionner tous les points qui sont contenus dans cette zone. Les points sélectionnés et non sélectionnés vont former respectivement le nuage résultat et le nuage de reste. La zone est toujours en 2D (dans le plan XY), on ne teste donc que les coordonnées planes des points. La 3ème dimension n'est pas encore prise en compte à ce niveau du développement.

Le résultat de cette étape est donc une nouvelle méthode de segmentation éventuellement un peu plus complexe, paramétrée pour la segmentation d'un calque (généralisation intermédiaire) et offrant des résultats plus précis et plus exacts.

IV.5. Optimisation des temps de traitement

Une de nos hypothèses concerne la rapidité de la méthode développée. Dès lors nous avons ajouté différentes étapes d'optimisation des temps de traitement. Tout d'abord, il faut chronométrer les différentes étapes de la segementation afin d'identifier celles qui sont les plus chronophages. Nous pourrons alors les modifier, les remplacer ou éventuellement les contourner afin de réduire le temps de traitement.

Cette optimisation peut être aussi réalisée en exécutant certaines opérations comme une indexation ou un échantillonnage du nuage. Un échantillonnage du nuage consiste à en diminuer le nombre de points en ne conservant qu'un point sur un certain nombre. Par exemple, un échantillonnage « 1/10 » revient à ne conserver qu'un point sur dix (le nombre de points est donc divisé par cette valeur). L'indexation spatiale revient à diviser l'espace en sous-espaces de différents niveaux de manière à l'organiser. Cette division n'est réalisée que pour les sous-espaces non-vides. Cette organisation est enregistrée sous une structure d'arbre et permettra de cibler les parties de l'espace qui nous intéressent (Donnay, 2018). Ainsi, nous pourrons nous focaliser sur ces parties plutôt que sur l'ensemble du nuage. Par exemple, imaginons l'indexation mise en place. Nous commencerons par identifier les sous-espaces intersectant notre zone tampon en parcourant les feuilles (nœuds) de l'arbre depuis le niveau de subdivision le plus large vers le plus précis. Suite à cette recherche, nous obtiendrons un index des sous-espaces intersectés et la segmentation ne se fera que pour les points appartenant à ces sous-espaces. On réduit ainsi le nombre de points à tester ce qui accélère le processus. Plusieurs méthodes d'indexation existent : octree, KD tree, … Elles diffèrent par leur méthode de division de l'espace, de construction ou de parcours de l'arbre, etc.

L'exemple ci-dessous illustre un « Region Quadtree » (Donnay, 2018), correspondant à une version 2D de l'octree. L'espace et les sous-espaces sont divisés en quatre carrés identiques à chaque nouveau niveau de précision atteint. L'espace 3D est divisé en 8 volumes identiques avec l'octree. Le principe reste toutefois le même. La ligne correspondrait au contour de notre zone tampon, les quadrillages bleus correspondent aux divisions de la surface en sous-surfaces, les nombres identifient les niveaux de précision de ces sous-surfaces. La méthode de fonctionnement est identique : division de l'espace et construction de l'arbre (1), parcours de l'arbre en fonction des intersections entre la zone et les sous-divisions (2), construction de l'index (3). La sélection des points devrait donc se faire parmi ceux appartenant aux sous-surfaces spécifiées dans cet index : 0-10-12-300.

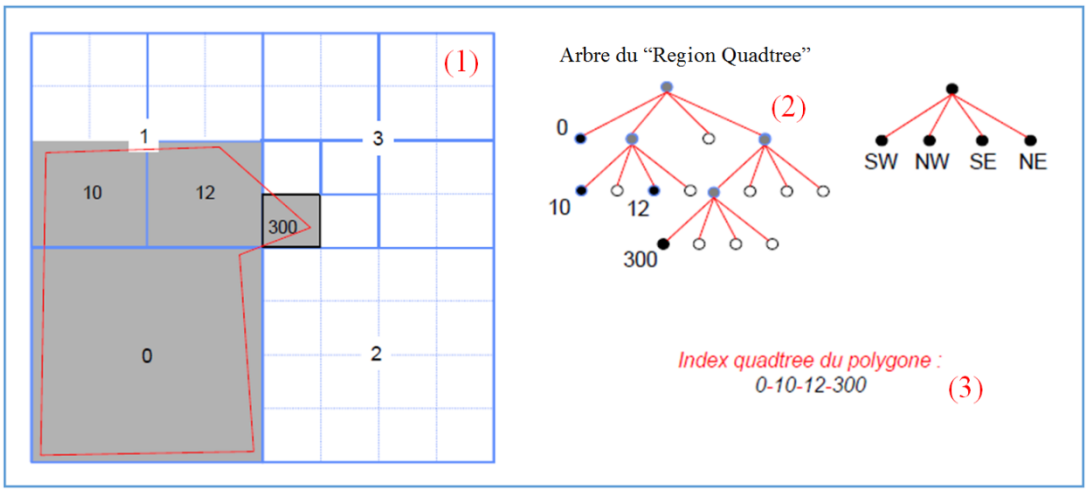


Figure 8 Exemple d'indexation en 2D avec un "Region Quadtree" (Donnay, 2018 ; modifiée)

La difficulté de cette étape sera de rendre la méthode plus rapide tout en conservant le niveau de précision et d'exactitude atteint précédemment. Ces deux étapes d'optimisation (4.2 et 5) sont liées et interfèreront l'une avec l'autre. Il faudra donc trouver un compromis entre justesse et rapidité. La généralisation va, elle aussi, modifier la précision et l'exactitude de la méthode ainsi que sa rapidité. Une méthode ne donnera pas nécessairement des résultats de même qualité pour toutes les classes (précision et exactitude). De même, le fait de passer d'une entité spécifique à toutes les entités d'un calque, va allonger le temps de traitement (impacte la rapidité). En réduisant ce temps de traitement, on permet aussi de réaliser les tests suivants plus aisément. Cette partie fournit donc une méthode de segmentation des entités d'un calque (généralisation intermédiaire) qui est plus complexe, mais plus rapide, tout en offrant des résultats précis et exacts.

IV.6. Généralisation de la segmentation à toutes les classes

Maintenant que nous avons une méthode rapide, précise et exacte, il faut augmenter son niveau de généralisation à un niveau global. Cela signifie appliquer la méthode de segmentation aux entités de chaque calque du plan. Il faudra aussi adapter la méthode à chacune des classes. Par exemple, il faudra décider si les zones tampons des entités d'une classe reprendront l'intérieur des entités ou seulement leurs contours (cf. Tab. 3). Cette généralisation engendrera, comme expliqué ci-dessus, une augmentation des temps de traitement et une diminution de précision et d'exactitude si la méthode ne s'adapte pas convenablement à la classe. Le résultat de cette étape est donc un algorithme généralisé, prenant en compte toutes les classes et adaptant la méthode de segmentation à chacune. Cet algorithme nous fournira les premiers résultats de segmentation totale du nuage.

IV.7. Amélioration de la segmentation par classe

Si la méthode de segmentation a été adaptée aux différentes classes lors de l'étape précédente, cela n'est pas nécessairement suffisant pour certaines classes. Celles-ci peuvent nécessiter certains compléments ou certaines améliorations. Cette étape va donc permettre de compléter et affiner les segmentations des différentes classes en prenant en compte leurs besoins propres. C'est dans cette étape que la coordonnée Z des points du nuage va être prise en compte. Nous y développerons la méthode de segmentation de la classe Sol-Plafond qui ne peut pas être basée sur le plan. Pour cette méthode nous utiliserons des histogrammes de fréquences (relatives) d'apparition des coordonnées Z par classe de 10cm pour déterminer les pics correspondant aux sol et plafond. Nous utiliserons au sein de ces classes de pic des paramètres caractérisant la dispersion des données. Nous y appliquerons une distribution normale de manière à déterminer les bornes d'un intervalle de confiance qui seront utilisées dans la segmentation. Les points de cette classe sont ceux compris dans les intervalles :

Intervalle sol = $[Z_{min}, BS]$ et Intervalle plafond = $[BI, Z_{max}]$

Où BS est la borne supérieure de l'intervalle calculé sur les données de la classe de pic sol et BI la borne inférieure de l'intervalle calculé sur les données de la classe de pic plafond. Soit n la taille de l'échantillon, X_i les valeurs de l'échantillon et a un coefficient dépendant du pourcentage de confiance utilisé (par exemple 95% = 1.96)

Moyenne (μ) =	$\sum_{i=1}^{n} Xi/\ n$	Ecart-type (σ) =	$\sqrt{\sigma^2}$
Variance (σ^2) =	$\sum_{i=1}^{n} (Xi - \mu)^2/\ n-1$	Intervalle de confiance =	$\mu \pm a * \sigma$

La hauteur et les paramètres de dispersion seront aussi utilisés pour resegmenter d'autres classes afin d'affiner le résultat. Il s'agit donc, en quelque sorte, d'une seconde amélioration de la précision et de l'exactitude générales de l'algorithme. De la même manière, une seconde optimisation des temps de traitement pourrait éventuellement être mise en œuvre.

Ces généralisation et optimisations constituent en quelque sorte un cycle d'amélioration de l'algorithme. Il faut répéter ce cycle jusqu'à atteindre les niveaux de généralisation, précision, exactitude et rapidité voulus. Le résultat de cette phase correspond au résultat final de la segmentation totale du nuage sur base du plan.

IV.8. Enregistrement des résultats (Outputs)

Le résultat de l'étape précédente correspond au résultat final de la segmentation mais ne constitue pas l'output de l'algorithme. Ces résultats obtenus doivent encore être enregistrés sous format « .las » dans un dossier approprié. Cette sauvegarde des résultats peut se faire de multiples manières : en créant un fichier par classe, en créant un fichier général dans lequel les classes sont distinguées grâce à des codes de classification, etc. Nous, nous combinerons les deux techniques citées. Ces fichiers constituent les résultats finaux de l'algorithme.

IV.9. Validations et évaluations statistiques

Si l'algorithme est terminé, il reste encore une étape : la validation des résultats. Pour valider, il faut comparer les résultats fournis par l'algorithme à des classifications de référence. Cette comparaison se fait pour différents nuages de validation. Les vérités terrains créées lors des prétraitements vont jouer le rôle des nuages de référence. Le but de l'étape est donc d'analyser statistiquement la justesse de la segmentation réalisée par l'algorithme. Un certain nombre de métriques statistiques peuvent être calculées pour quantifier cela.

Nous utiliserons tout d'abord des matrices de confusion et les métriques associées. Elles prennent en entrées le nuage de résultat et le nuage de référence correspondant et vont comparer les informations pour chaque point. Il faut donc s'assurer que les deux points comparés correspondent bien au même point dans les deux nuages. Pour ce faire, les points des deux nuages seront triés similairement afin d'être enregistrés dans le même ordre. Le résultat de la matrice de confusion est une matrice de contingence carrée, de dimension égale au nombre de classes. Elle reprend notamment sur la diagonale le nombre de points qui ont été bien classés (en vert sur l'illustration). Il s'agit des vrais positifs, c'est-à-dire les points appartenant réellement à la classe et attribués par notre classification à celle-ci. De base, la matrice fournit les données en tant que nombre de points (valeur absolue) mais pourrait être normalisée, ce qui n'a pas été réalisé ici.

L'exactitude globale correspond au pourcentage total de points bien classés dans le résultat. Soit, *n* le nombre de points total du nuage (taille du nuage), X_{ii} l'élément de la ligne *i*, colonnne *i* (diagonale pour la i[ème] classe), et *c* le nombre de classes :

$$\text{Exactitude globale} = \frac{\sum_{i=1}^{c} Xii}{n}$$

		Classes du résultat obtenu						
		Classe 1	Classe 2	Classe 3	Classe 4	Classe 5	Classe 6	Classe 7
Classes de la vérité terrain	Classe 1							
	Classe 2							
	Classe 3				(1)			
	Classe 4							
	Classe 5			(2)				
	Classe 6							
	Classe 7							

Figure 9 Exemple d'une matrice de confusion vide.

Dans notre cas, les classes de référence seront situées en ordonnée et celles des résultats en abscisse. De ce fait, nous retrouverons dans les cases (hors diagonale) d'une ligne choisie les faux négatifs (en rouge sur l'illustration) et, dans celles de la colonne correspondante, les faux positifs (en bleu). Les faux-négatifs (ou omissions) sont les points qui appartiennent réellement à la classe de cette ligne (« *classe 3*») mais qui ont été faussement attribués à la classe de la colonne (par exemple : « *classe 4* », (1)). Les faux-positifs (ou commissions), sont les points qui ont été faussement attribués à la classe traitée (« *classe 3*») alors qu'ils appartiennent en réalité à la classe de la ligne (par exemple, « *classe 5* », (2)). (Cornet, 2016 ; Harris Geospatial Solutions, 2018)

Le taux d'omission (*TO*) pour la classe de la ligne *i* correspond au taux de faux négatifs total (Harris Geospatial Solutions, 2018 ; Cornet, 2016) :

$$TO = 1 - \frac{Xii}{\left(\sum_{j=1}^{c} Xij\right)}$$ (Précision du producteur = 1 – *TO*, non utilisée)

Le taux de commission (*TC*) pour une classe de la colonne *j* correspond au taux de faux positifs total (Cornet, 2016 ; Harris Geospatial Solutions, 2018) :

$$TC = 1 - \frac{Xjj}{(\sum_{i=1}^{c} Xij)} \qquad \text{(Précision de l'utilisateur = 1 – } TC\text{, non utilisée)}$$

En plus de la matrice de confusion, nous utiliserons aussi le KIA (Kappa Index of Agreement) (Cornet 2016) basé sur l'utilisation du Kappa de Cohen. Ce coefficient fournit un degré de concordance général entre deux « juges », deux « méthodes de sélection », ici nos deux classifications. On ne compare donc plus un résultat à une référence mais simplement deux classifications. Ce degré sera fourni sous la forme d'un score (standardisé) variant de -1 à 1. Le cas nul signifierait que la concordance entre nos méthodes n'est pas différente de la concordance obtenue par une classification aléatoire (concordance due à la chance). Si le score est négatif (peu probable), il n'y a pas de concordance. (McHugh, 2012).

Le score est calculé en utilisant la probabilité observée sur la diagonale (P_o) et la probabilité calculée sur la diagonale si l'échantillon était distribué aléatoirement dans la table (P_c). Ainsi, soit *c* la dimension de la matrice de confusion, n_{ij} un élément de cette matrice situé à la ligne *i*, colonne j et *n* le nombre de points total du nuage (Cornet, 2016):

$$P_o = \sum_{i=1}^{c} P_{ii} \qquad P_c = \sum_{i=1}^{c} P_{i+}P_{+j}$$

$$\text{Avec } P_{i+} = \sum_{j=1}^{c} P_{ij} \quad ; \quad P_{+j} = \sum_{i=1}^{c} P_{ij} \quad et \quad P_{ij} = n_{ij}/n$$

Le coefficient Kappa vaut alors (Cornet, 2016) :

$$\kappa = \frac{p_0 - p_c}{1 - p_c}$$

Le coefficient Kappa peut aussi être calculé pour une seule classe. Toujours avec *n*, la taille de l'échantillon (nuage), X_{ii} un élément de la diagonale de la matrice de confusion et $X_{i.}$, $X_{.i}$ respectivement la somme en ligne et en colonne pour la classe dans cette matrice de confusion, (Cornet, 2016) :

$$\hat{\kappa}_s = \frac{n X_{ii} - X_{i.} X_{.i}}{n X_{i.} - X_{i.} X_{.i}}$$

L’interprétation du score peut se faire de la façon suivante (McHugh, 2012) :

Value of Kappa	Level of Agreement	% of Data that are Reliable
0-.20	None	0-4%
.21-.39	Minimal	4-15%
.40-.59	Weak	15-35%
.60-.79	Moderate	35-63%
.80-.90	Strong	64-81%
Above .90	Almost Perfect	82-100%

Figure 10 Niveau de concordance entre les méthodes selon le Kappa de Cohen obtenu (McHugh, 2012)

A la fin de la validation, nous obtenons ainsi des données statistiques à analyser et à discuter afin de critiquer l’algorithme développé.

Environnement de travail et données

Ce point ne fait pas exactement partie de la méthodologie puisqu’on y présente les conditions dans lesquelles notre algorithme a été développé. La méthodologie se veut, en effet, générale or ces conditions sont propres à notre travail. Ces informations constituent toutefois une suite logique à la méthodologie et doivent être évoquées avant la présentation de nos résultats. Nous présenterons donc : les logiciels employés, les différentes données utilisées et l’environnement de développement.

Au niveau des logiciels, nous avons utilisé le logiciel AutoCAD (Autodesk help, 2017) pour visualiser et manipuler les plans au format « .dxf » et des nuages de points au format « .rcp ». Le logiciel CloudCompare (CloudCompare, 2016), a, lui, été utilisé pour visualiser et manipuler les nuages de points sous format « .las ». Enfin, le développement de l’algorithme s’est fait via le logiciel Anaconda (distributeur et gestionnaire de packages Python). Nous avons utilisé Anaconda Navigator (Graphical User Interface) (Anaconda Documentation, 2019), pour la gestion de l’environnement de développement et le logiciel Spyder (Integrated Development Environment) (Spyder Docs, 2018) pour la production des codes. Si CloudCompare, Anaconda et Spyder sont tous des logiciels open source, AutoCAD est lui un logiciel propriétaire qui a été utilisé sous les licences de l’ULiège (Unité de Géomatique).

Quant aux données, plusieurs jeux ont été utilisés. Tout d’abord, un certain nombre de nuages relevés par l’Unité de Géomatique de l’ULiège ont été mis à disposition pour la prise en mains du sujet. Malheureusement, par absence des plans « .dxf » associés, ils n’ont pu être

utilisés par la suite. Avant d'obtenir les données réelles de travail, certains tests ont été réalisés sur un jeu de données levé sur le domaine de l'ULiège au Sart Tilman. Ces données sont constituées d'un nuage de points et d'un plan « .dxf » reprenant notamment des murs. Les tests ont ainsi pu être effectués sur un échantillon du nuage et sur les éléments « Murs » du plan. Toutefois, ces données représentent une situation en extérieur, ce qui ne correspond pas aux conditions du travail : elles n'ont pu être utilisées comme données de travail.

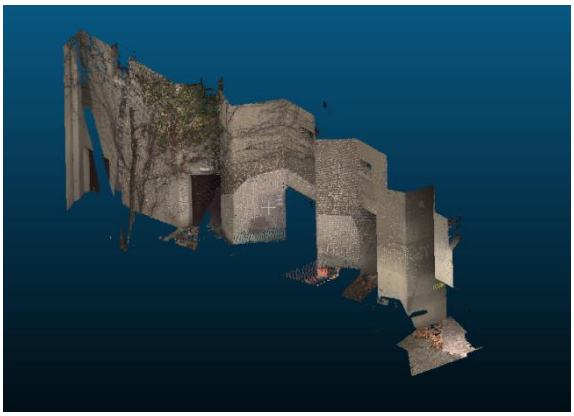


Figure 11 Nuage de points utilisé pour les premiers tests de segmentation (jeu de données test)

Enfin, les données utilisées pour développer l'algorithme sont constituées :

- D'un nuage de points réalisé par l'Unité de Géomatique de l'ULiège et réprésentant une partie du 2ème étage du bâtiment B5a (Institut de Physique) de l'ULiège (Sart-Tilman). Ce nuage a été redivisé en plusieurs sous-nuages dont quatre correspondent aux différents bureaux scannés. Ce sont ces 4 nuages que nous utiliserons :

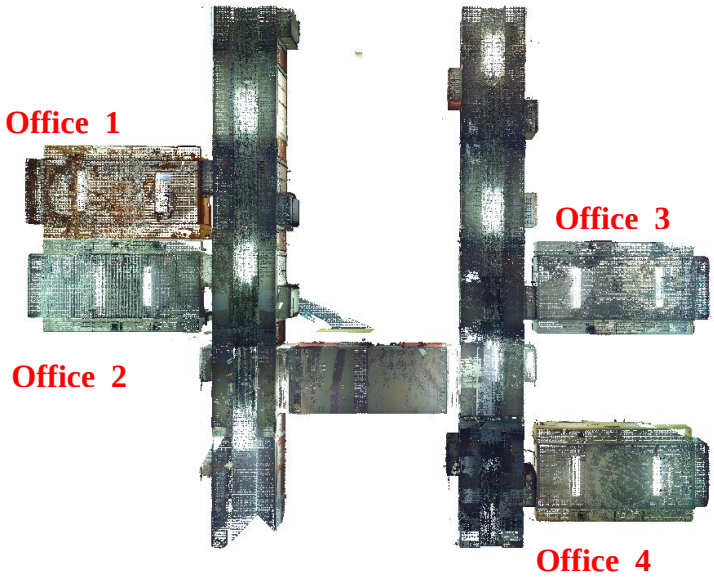


Figure 12 Nuage Total, vu du haut

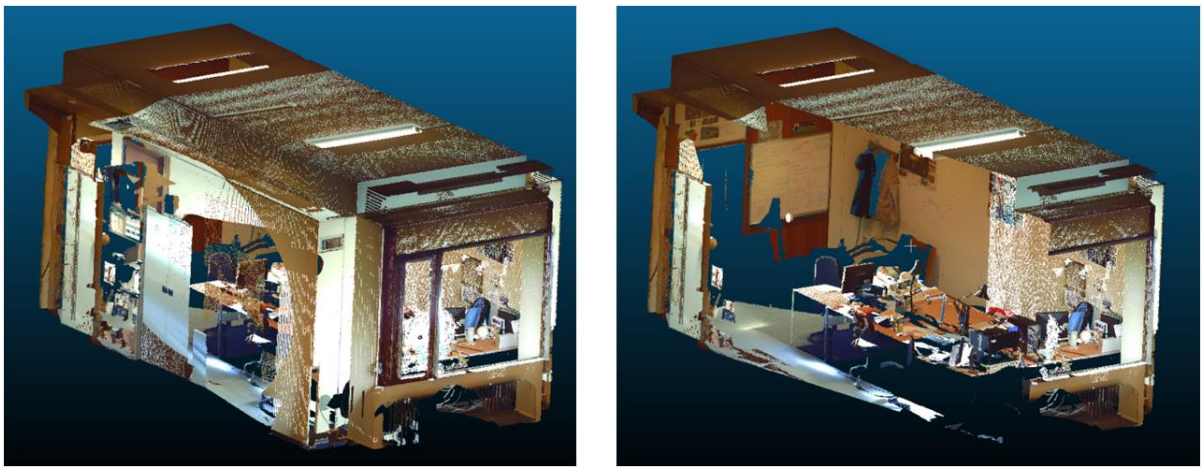


Figure 13 Nuage "Office_1, vue totale (à gauche) et vue tronquée d'une partie (à droite)
Les nuages des bureaux 2, 3 et 4 sont disponibles en annexes.

- D'un plan de ce deuxième étage fourni par l'ARI (Administration des Ressources Immobilières de l'ULiège). Le plan total sera, lui aussi, redivisé pour obtenir un plan associé à chaque bureau.

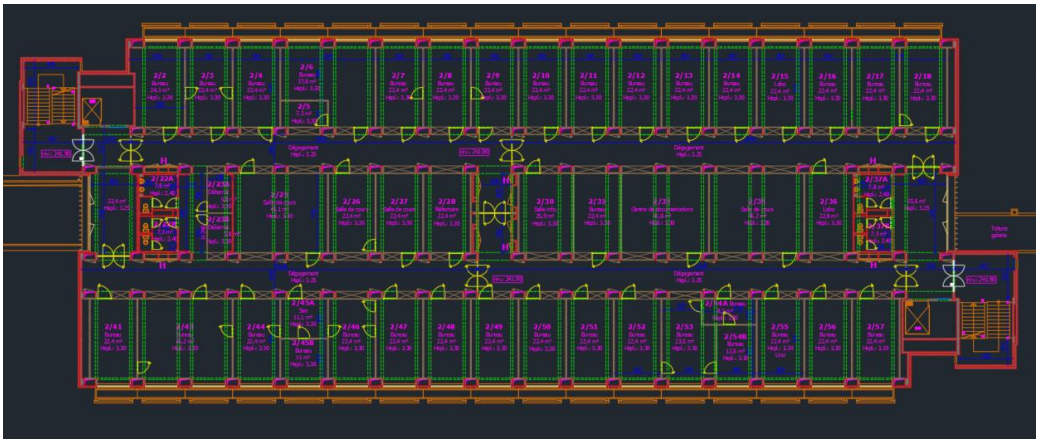


Figure 14 Plan total du 2ème étage

Parmi les bureaux segmentés, « Office_1 » est utilisé comme nuage d'entrainement pour paramétrer l'algorithme et les méthodes de segmentation. Les 3 autres nuages sont utilisés comme nuages de validation. Ces données ont divers avantages :

- leur disponibilité permettait d'éviter une phase de levé,
- elles sont de qualité et facilement compréhensibles,
- le plan et le nuage de départ uniques permettent d'éviter de répéter certaines opérations notamment au niveau des prétraitements,
- le milieu est connu et accessible ce qui favorise une meilleure visualisation et compréhension des scènes,
- ...

Les données de paramétrage et de validation proviennent d'un même nuage. Cela impacte la pertinence de la validation et sera discuté par la suite (cf. VI.3).

Pour terminer avec les données, nous avons pu utiliser les vérités terrains réalisées par M. Abderrazzaq Kharroubi (stagiaire au sein de l'Unité de Géomatique) pour chacun des sous-nuages. Ces vérités terrains ont dû être adaptées à notre travail puisqu'elles ont été construites en fonction des besoins de la recherche de M. Kharroubi et non de notre travail. Quelques modifications ont donc dû être réalisées et sont présentées dans le chapitre « Algorithme et Résultats » (cf. V.2.2). Bénéficier de ces vérités terrains comme point de départ nous a permis de résoudre plus rapidement une étape quelque peu fastidieuse. Nous remercions sincèrement M. Kharroubi pour cela.

Enfin, pour ce qui est de l'organisation de l'environnement de développement, Anaconda a été utilisé pour gérer les librairies et les versions de cet environnement de développement. Spyder (3.3.4, sous Python 3.6) a été installé dans ces différentes versions et utilisé pour développer le code et le tester. Plusieurs librairies ont ainsi été installées dans cet environnement et utilisées dans le code :

- Dxfgrabber (Moizi, 2018) : extraction d'informations depuis les « .dxf »
- Laspy (Brown & Butler, 2014) : édition et manipulation de fichiers « .las »
- Matplotlib (Hunter et al., 2019): construction de graphiques (visualisation de données)
- Numpy (The SciPy community, 2019): manipulation de matrices à N dimensions
- Shapely (Gillies, 2018) : création et manipulation d'objets géométriques plans
- Sklearn (Scikit-learn, 2019) : librairie associée au Machine Learning, utilisée ici dans le cadre de la validation. Elle permet de calculer différentes métriques statistiques
- Scipy (The SciPy community, 2019): librairie multi-usage destinée aux travaux scientifiques et utilisée, elle aussi, pour calculer différentes statistiques
- Time : librairie intégrée de Python permettant de mesurer des temps de traitement

AutoCAD et CloudCompare sont indépendants d'Anaconda et donc externes à l'environnement de développement.

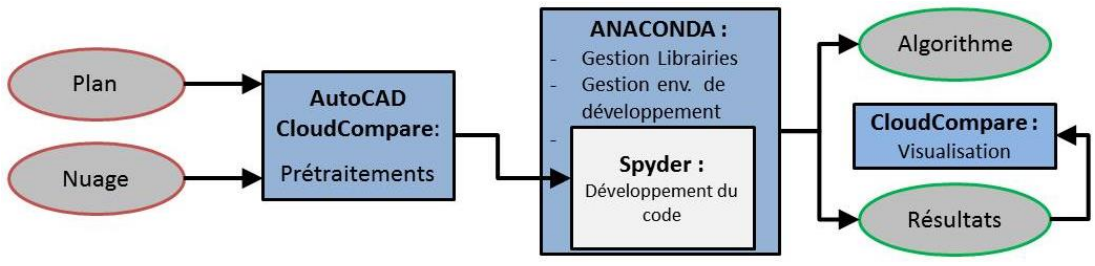


Figure 15 Représentation schématique de l'environnement de travail et de son utilisation

CHAPITRE V. ALGORITHME ET RÉSULTATS

Ce chapitre présente et explique les différents développements et résultats obtenus tout au long du travail. Ils sont présentés selon l'ordre établi dans la méthodologie. Cela permet de présenter l'évolution du travail, les problèmes rencontrés et les solutions trouvées ou envisagées pour y répondre.

En ce qui concerne la prise en mains, nous avons pu nous familiariser à la manipulation et visualisation des nuages de points grâce aux nuages mis à notre disposition par l'Unité de Géomatique. Nous avons réalisé des segmentations manuelles de ces différents nuages en utilisant différentes techniques offertes par CloudCompare : champs scalaires selon un axe du système de coordonnées, boites de segmentation 3D modulables, découpages 2D, utilisation de filtres correctifs, algorithmes existants, etc. Manipuler ces données nous a permis de mettre en avant un certain nombre de faits et de problèmes avant la phase de codage :

- les limitations des PC notamment pour supporter l'affichage de nuages de points de grande ampleur ou certaines versions récentes de logiciels,
- la précision relative associée aux techniques de segmentation manuelle,
- le coté fastidieux, chronophage et la subjectivité de la segmentation,
- la grande variété des milieux représentés (gestion d'éléments complexes tels que des jonctions de mur arrondies ou d'éléments plus rares tels que des moulures, etc.),
- la gestion du bruit et des occlusions
- …

Ces manipulations ont aussi permis d'énoncer un premier essai de classification et de faire ressortir un ordre de segmentation pour ces classes. On commence d'abord par segmenter le sol et le plafond. Ensuite, on segmente les murs, portes et fenêtre incluses. On segmentera alors en une classe « ouvertures » ces portes et fenêtres avant de segmenter les éléments fixés aux murs, par exemple des armoires (« éléments fixes »). Enfin, parmi les points restants, nous reprendrons certains éléments au sein d'une classe « éléments mobiles » et nous constituerons une classe « reste et bruit » pour ce qui n'a pas été segmenté. Les résultats de ces diverses classifications manuelles ne seront pas illustrés ici. En effet, il ne s'agit pas vraiment du résultat réel de notre travail mais d'une simple opération de familiarisation. Il semble donc peu pertinent d'inclure de telles illustrations ici.

Comme expliqué dans la méthodologie, nous avons utilisé au départ des données illustrant un mur scanné sur le domaine de l'ULiège au Sart-Tilman. Nous avons ensuite utilisé les données correspondant au 2ème étage du bâtiment B5a (Institut de Physique) de l'Uliège (Sart-Tilman). Les données ont été traitées dans leur ensemble (nuage et plan total) pour les deux phases suivantes (Géoréférencement et Prétraitements). La suite du développement a été réalisée sur les sous-nuages de bureaux extraits du nuage total et sur les sous-plans correspondants (issus des données de travail). Pour rappel, le nuage « Office_1 » a servi de nuage de paramétrage de l'algorithme tandis que les trois autres ont servi de nuages de validation. (cf. IV.Environnement de travail et données).

V.1. Géoréférencement

Le géoréférencement du plan et des nuages a été réalisé via AutoCAD. Les deux fichiers étant initialement exprimés dans deux systèmes de coordonnées locaux différents, il a fallu les exprimer dans un système commun. Comme expliqué dans la méthodologie, nous avons utilisé le système de coordonnées du nuage puisqu'il nous faut un système 3D et que le plan est en 2D.

Dans notre cas, réaliser le référencement dans ce sens nous assure aussi que les coordonnées du plan correspondent directement aux coordonnées du nuage et de tous les sous-nuages. En effet, les sous-nuages étant déjà créés, nous aurions dû les référencer chacun individuellement si le sens de référencement était inversé. Pour éviter de répéter cette opération, nous avons géoréférencé l'entièreté du nuage et du plan avant de les subdiviser.

Le nuage de points total a dû d'abord être mis en forme. En effet, étant très volumineux, il était plus pratique de n'en utiliser qu'un échantillon. Pour cela, nous avons réalisé une coupe horizontale dans le nuage. Les plans d'intérieur étant généralement réalisés à une hauteur de 1,3m, la coupe a été effectuée à cette hauteur, calculée depuis le point le plus bas du nuage, et sur une épaisseur de 30cm. Ce fichier « .las » a été ensuite converti au format « .rcp », format propre d'AutoCAD, pour pouvoir être manipulé au sein du logiciel. Le plan, lui, n'a pas subi de transformation particulière et a été utilisé tel quel afin d'être intégralement référencé.

Figure 16: Coupe horizontale dans le plan total (de 1,30 à 1,60m)

La mise en forme terminée, les deux fichiers ont été importés sur AutoCAD. Pour ce faire, nous avons d'abord ouvert le plan avant d'y insérer le nuage. Lors de l'insertion du nuage dans l'espace 3D d'AutoCAD, plusieurs paramètres de projection étaient fournis et notamment un facteur d'échelle de 39,3701. Cela signifie que lors de la visualisation du nuage dans AutoCAD, le système de coordonnées de ce nuage était modifié et de ce fait, il ne correspondait plus au système exact des données. Le sytème de destination était donc biaisé et le géoréférencement compromis. En effet, en cet état, les points du plan auraient été exprimés dans un système agrandi par rapport au système réel du nuage. Afin d'éviter cette erreur, le système de coordonnées réel du nuage a été rétabli en appliquant au nuage un facteur d'échelle inverse :

$$\frac{1}{39.3701} = 0.0253999863$$

Une fois cette correction appliquée, nous avons pu réaliser le référencement du plan dans le système de coordonnées réel du nuage. Pour ce faire, nous avons utilisé la fonction ADETRANSFORM d'AutoCAD (Autodesk Help, 2016) qui permet de réaliser une transformation de Helmert en 2D telle que décrite dans la méthodologie. Elle prend en entrées (« inputs ») deux points sources et deux points de destinations correspondants. En sortie, l'ensemble des objets sélectionnés ont été déplacés, tournés et mis à l'échelle de sorte que les points sources se trouvent aux mêmes coordonnées que leur point de destination respectif. Le géoréférencement est alors effectué.

Nous avons commencé par déterminer deux points sources sur le plan et les points de destinations correspondants sur le nuage en créant une ligne sur chacun d'eux :

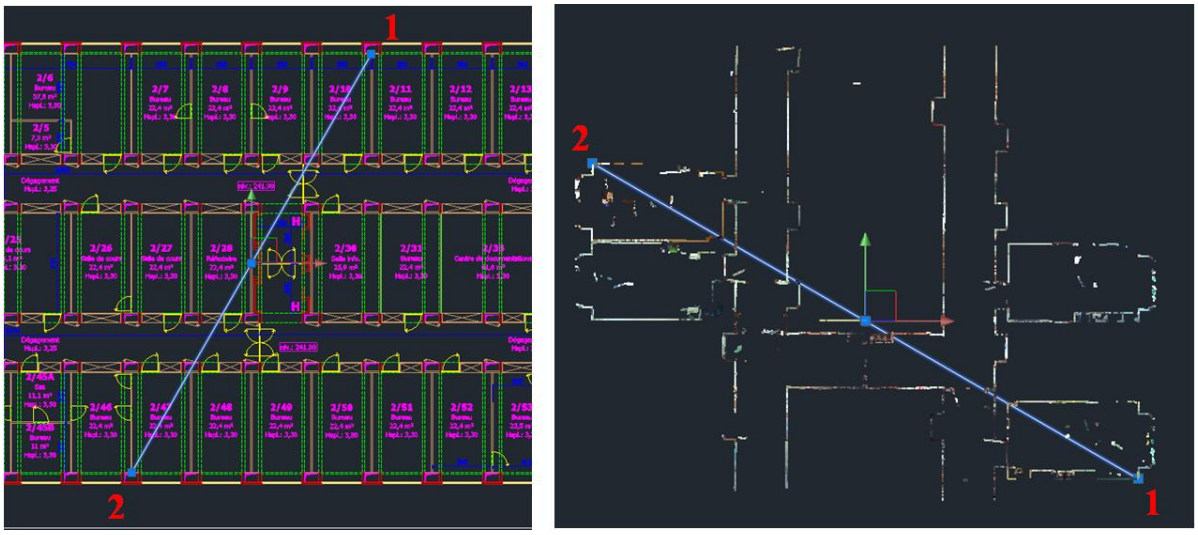


Figure 17 Création de deux lignes dont les points correspondent aux points sources (plan) et aux points destinations (nuage)

Une fois ces points déterminés, nous avons lancé la fonction en sélectionnant tous les éléments du plan et en spécifiant les points sources et destinations tout juste créés. Comme expliqué dans la méthodologie, il s'agit d'une transformation en 2D puisque le plan est en 2D. Afin que le plan soit bien perpendiculaire à l'axe Z du nuage après transformation, les points spécifiés comme destinations sur le nuage ont été sélectionnés à une hauteur identique. Vu que les points du plan sont tous de même hauteur, la tranformation leur attribuera à tous cette hauteur de destination. Le plan sera donc perpendiculaire à l'axe Z et bien référencé.

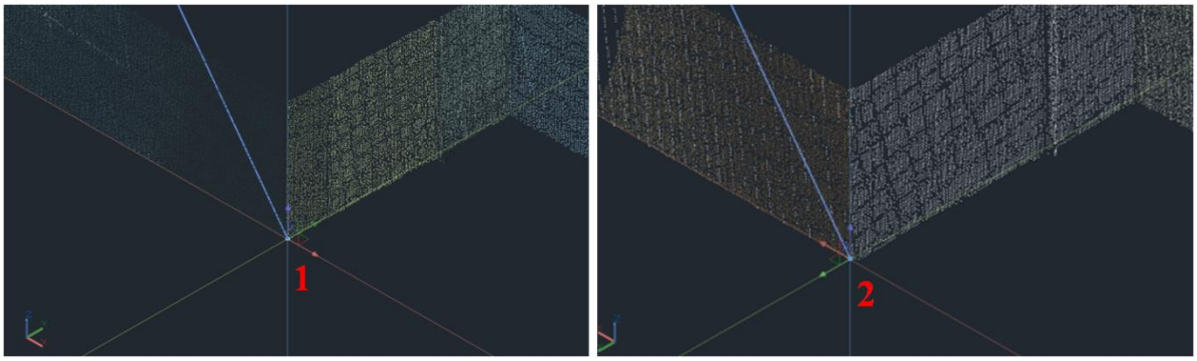


Figure 18 Création des deux points destinations sur le nuage, à une hauteur identique

Ainsi, nous obtenons le résultat attendu : les éléments correspondants dans le plan et le nuage possèdent les mêmes coordonnées. Il peut toutefois exister des écarts entre le plan et le nuage pouvant engendrer des erreurs par la suite. Des illustrations de ce résultat sont disponibles en annexes (cf. IX.3).

V.2. Prétraitements

V.2.1. **Choix de la classification**

Avant de se lancer dans la suite des opérations, il est nécessaire de choisir une classification sur laquelle se basera notre algorithme. Pour nous aider dans ce choix, nous avons utilisé trois sources d'informations différentes : la classification formulée lors de la prise en mains, la littérature (cf. II.2.1), et surtout les calques présents dans notre plan « .dxf ».

Pour ce qui est de la classification formulée lors de la prise en mains du sujet, diverses classes avaient été déterminées durant cette étape :

- « Eléments séparateurs » : murs, sol et plafond
- « Poutres et colonnes »
- « Ouvertures » : portes et fenêtres
- « Eléments fixes (ou fixés) » : armoires, étagères, radiateurs, lampes, etc.
- « Eléments mobiles » : décorations, meubles, etc.
- « Bruit et non classé »

Pour la littérature, plusieurs types de classification ressortent selon le type et le but du travail présenté. Ces différents travaux sont cités et présentés plus en détails dans l'état de l'art (cf. II.2.1).

- Valero et al (2012) Macher et al. (2017), Hong et al. (2015) Anagnostopoulos et al. (2016) s'intéressent à la création de modèles 3D à partir de scans (BIM) et ont utilisé trois classes : « Murs », « Sol » et « Plafond ».
- Previtali et al. (2014) ont présenté une méthode automatique de transformation de nuages d'intérieur en modèles « as-built ». Ils utilisent les classes « Murs », « Sol », « Plafond » ainsi que « Portes », « Fenêtres » et « Occlusions ».
- Rajala & Penttilä (2006) ont, eux, étudié l'utilisation des BIM 3D pour la rénovation de bâtiments. Ils utilisent une classification plus détaillée : « Murs porteurs », « Murs externes », « Murs internes non-porteurs », « Plancher », « Colonnes » et « Poutres ».
- Nan et al. (2012) présentent une approche pour modéliser une scène intérieure à partir d'un nuage de points 3D. Ils utilisent une classification reprenant plutôt des meubles et décorations. Günther et al. (2017) utilisent eux aussi des classes de meubles.
- Hermans et al. (2014) emploient, eux, la labélisation de scènes en 2D (images RGB-D) pour labéliser et reconstruire des scènes en 3D à partir de nuages. Ils utilisent des classes standards (« Murs », « Sol », « Plafond ») ainsi que des classes de mobilier (« Chaises », « Tables », « Lits », « Décorations », etc.).
- Silberman & Fergus (2011) se sont intéressés à la segmentation de scènes intérieures en utilisant un Microsoft Kinect. Ils ont déterminé 13 classes : « Background » (objets rares), « Lits », « Sol », « Stores », « Fenêtres », « Tables », « Armoires », « Télévisions », « Murs » et « Etagères ».

- Enfin, Armeni et al. (2016), ont étudié la décomposition sémantique d'un nuage de points 3D d'un bâtiment entier. Ils ont dégagé 12 classes reprenant à la fois des éléments structurants et de mobilier : « Sol », « Plafond », « Murs », « Colonnes », « Poutres », « Fenêtres », « Portes », « Tables », « Chaises », « Canapés », « Etagères/Armoires » et « Tableaux ».

Voici donc un échantillon de classifications que l'on peut trouver dans la littérature associée à notre sujet. Certaines classifications sont plus liées au thème des BIM et comprennent des classes plus centrées sur les éléments structurels (Sol, Plafond, Murs, etc.). D'autres classifications sont plus liées à la reconnaissance de scènes d'intérieur et seront constituées des classes orientées vers le mobilier et la décoration. Enfin, certaines classifications sont mixtes et associent des classes d'éléments structurels et des classes de mobilier/décorations.

Enfin, pour ce qui est du plan, il s'agit de la source d'informations impactant le plus ce choix. En effet, la classification formée par les calques constituera l'information qui sera extraite du plan et utilisée dans la segmentation. La majorité de nos classes proviendront donc du plan. Il sera par ailleurs compliqué et moins pertinent, par rapport à notre méthode, d'ajouter des classes qui n'y sont pas présentes. Une classe « Tables » sera par exemple compliquée à segmenter sur base du plan puisque les tables n'y sont pas représentées. Les calques du plan auront ainsi un effet limitant. Certaines classes devront tout de même être ajoutées, telles que « Sol-Plafond » ou une classe pour le reste (tous les points doivent être classés). Tous les calques ne sont pas non plus à utiliser dans la classification choisie. Nous utiliserons : 'Baies', 'Coupé murs', 'Dessus' (poutres), 'Menuiseries Intérieures' (armoires encastrées, etc.) et 'Portes' (cf. Fig 19). Au final, sur base des informations obtenues depuis les trois sources citées ci-dessus, la classification choisie comprend 7 classes (cf. Tab. 1).

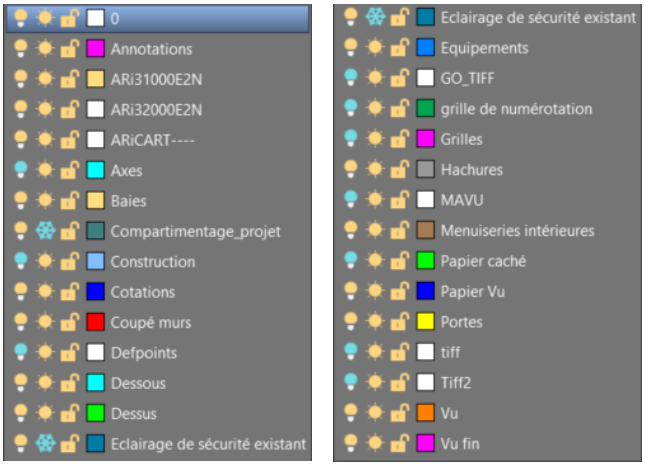


Figure 19 Calques du plan

CLASSES
« Remains » (les restes des segmentations)
« Sol-Plafond »
« Murs »
« Poutres »
« Meubles Encastrés »
« Portes »
« Fenêtres »

Tableau 1 Classes utilisées dans ce travail

V.2.2. **Vérités Terrains**

La classification établie, nous pouvons déjà réaliser les vérités terrains qui nous serviront de référence lors de la validation des résultats. Pour ce faire, il faut réaliser une segmentation manuelle et fondée sur notre classification pour les nuages qui seront utilisés lors de cette évaluation. Comme expliqué dans la méthodologie, nous avons pu utiliser les segmentations réalisées par M. Abderrazzaq Kharroubi, stagiaire au sein de l'Unité de Géomatique. Nous sommes donc repartis de ces données et les avons adaptées à notre classification.

- Il a fallu rassembler les sous-nuages selon nos classes. Les murs et les colonnes ont été rassemblés sous la classe « Murs », le sol et le plafond sous « Sol-Plafond », les poutres et portes sous leur classe éponyme respective et tous les éléments non classifiés dans « Remains ». Nous avons ainsi reconstitué nos sept classes.

- Il a aussi fallu adapter certaines segmentations à notre manière de segmenter. Par exemple, les murs séparant le bureau et le couloir avaient été segmentés d'un seul tenant. Or, dans notre classification, la différence est réalisée entre le mur, la porte et le meuble encastré. Nous avons donc resegmenté ces éléments :

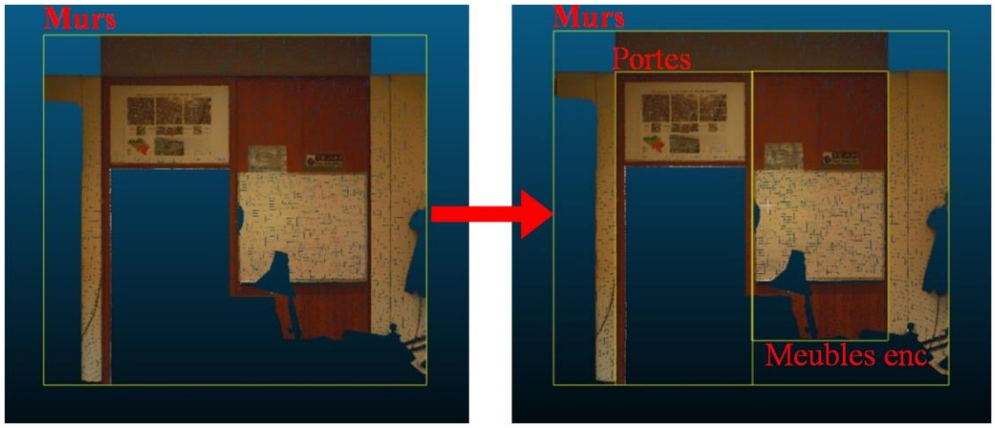


Figure 20 Modification (Murs) de la vérité terrain fournie par M. Kharroubi

- Une autre différence réside dans la segmentation des poutres. Dans les différents bureaux, l'espace entre les murs et les poutres est fermé, formant un conduit. Dans les segmentations de M. Kharroubi, ce conduit est repris en tant que poutre. De plus, dans le cas d'un faux plafond, les poutres n'étaient pas distinguées de ce faux-plafond. Cependant, au sein de notre classification, ces conduits ne sont pas repris comme poutres et les poutres sont distinguées du plafond. Nous avons donc resegmenté les poutres et segmenté les conduits dans « Remains ».

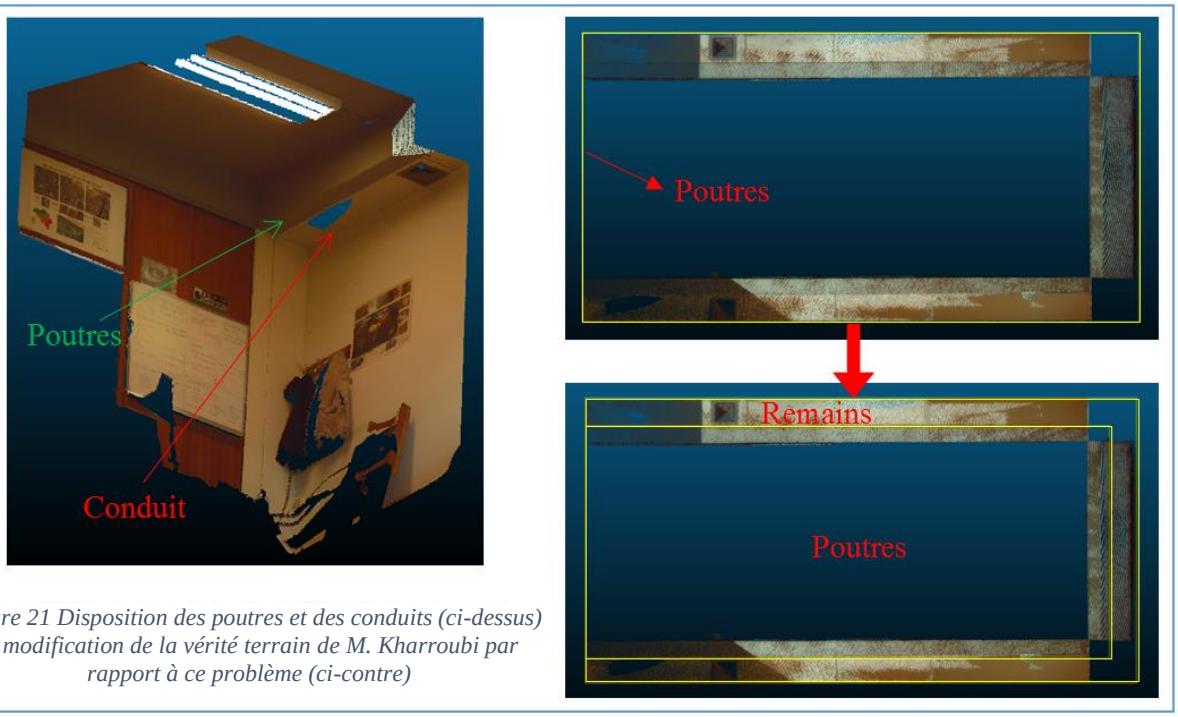


Figure 21 Disposition des poutres et des conduits (ci-dessus) et modification de la vérité terrain de M. Kharroubi par rapport à ce problème (ci-contre)

Nous obtenons donc, à la fin de cette étape, quatre vérités terrains, une pour chaque nuage de points de bureau utilisé. Chaque vérité terrain correspond à un ensemble de sept fichiers « .las » comprenant les points appartenant à une même classe. (cf. IX.5).

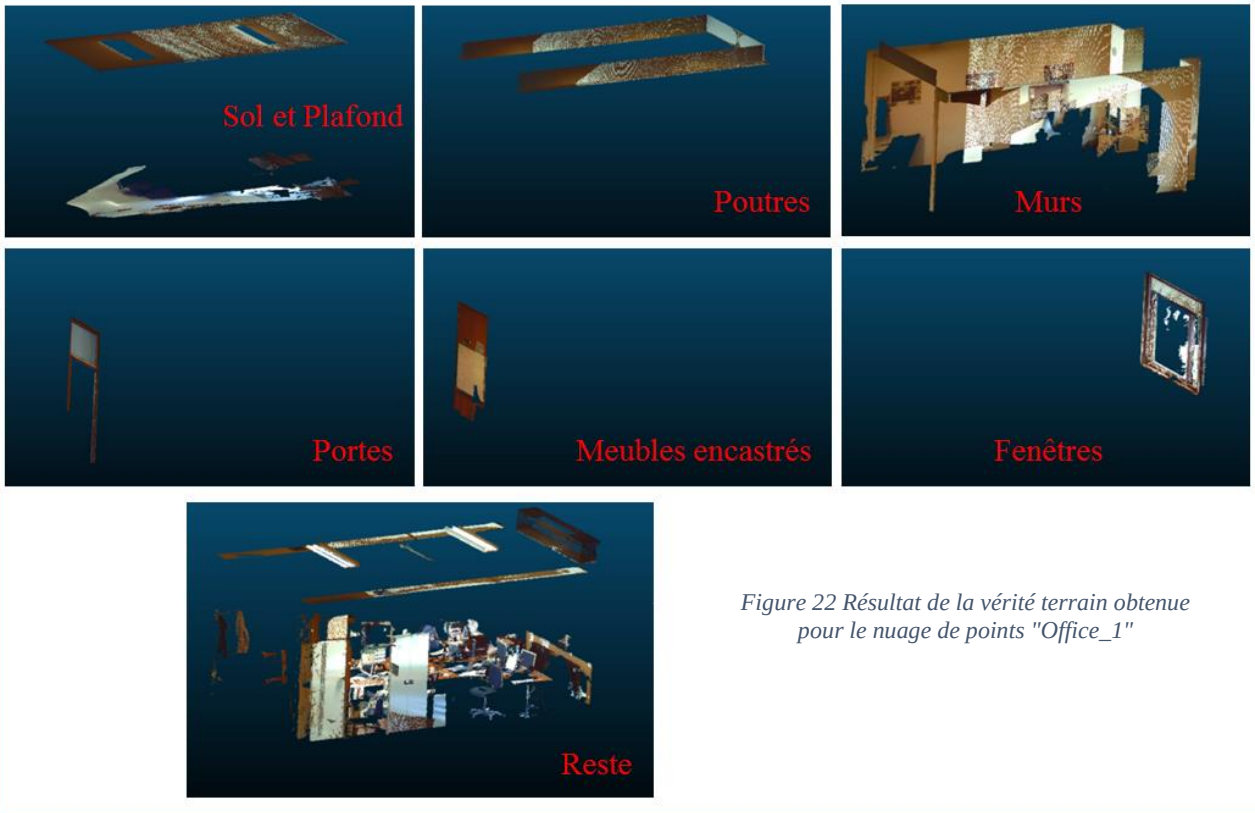


Figure 22 Résultat de la vérité terrain obtenue pour le nuage de points "Office_1"

V.2.3. **Mise en forme du plan « .dxf »**

Le plan que nous utilisons doit subir un certain formatage pour pouvoir être utilisé par l'algorithme. Tout d'abord, notre algorithme se basera sur des noms de calques précis, c'est pourquoi les calques que nous utilisons (baies, coupé murs, dessus, menuiseries intérieures et portes) doivent être renommés. Le nom de chaque calque doit commencer par le préfixe « PCS_ » (pour Point Cloud Segmentation) suivi de la dénomination du calque. Cette dénomination doit obligatoirement comprendre la traduction anglaise du nom de la classe correspondante (et éventuellement d'autres mots) :

CLASSES	CALQUES
« Sol-Plafond »	/
« Murs »	'PCS_ Wall'
« Poutres »	'PCS_Beam'
« Meubles Encastrés »	'PCS_Embedded'
« Portes »	'PCS_Door'
« Fenêtres »	'PCS_Window'
« Remains » (non-classifié)	/

Tableau 2 Classes et leur calque associé

En plus de ce formatage des noms de calques, le plan a aussi été nettoyé des différents calques qui ne seront pas utilisés. Certains calques ont toutefois été conservés parce qu'ils renferment des informations intéressantes (annotation sur le numéro des bureaux par exemple) ou parce qu'ils permettent une meilleure visualisation et représentation de la scène.

En outre, l'algorithme va passer en revue et utiliser les entités appartenant à ces calques. Cependant, le plan représente l'entièreté de l'étage tandis que le nuage n'en représente qu'une partie. Les entités de ces calques situées hors des limites de notre nuage seront donc analysées inutilement. En réduisant le plan à l'emprise du nuage, les performances de l'algorithme seront améliorées : pas d'analyse inutile.

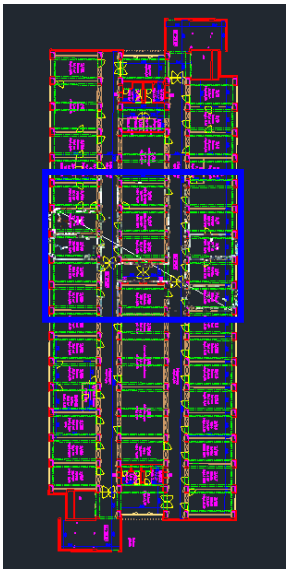


Figure 23 Emprise du nuage (cadre bleu) sur le plan total de l'étage.

La réduction du plan sera aussi appliquée pour les bureaux : à chaque sous-nuage de bureau correspondra le sous-plan de la pièce. Le plan a ainsi été nettoyé des calques et des entités non utilisées et on obtient un résultat intermédiaire.

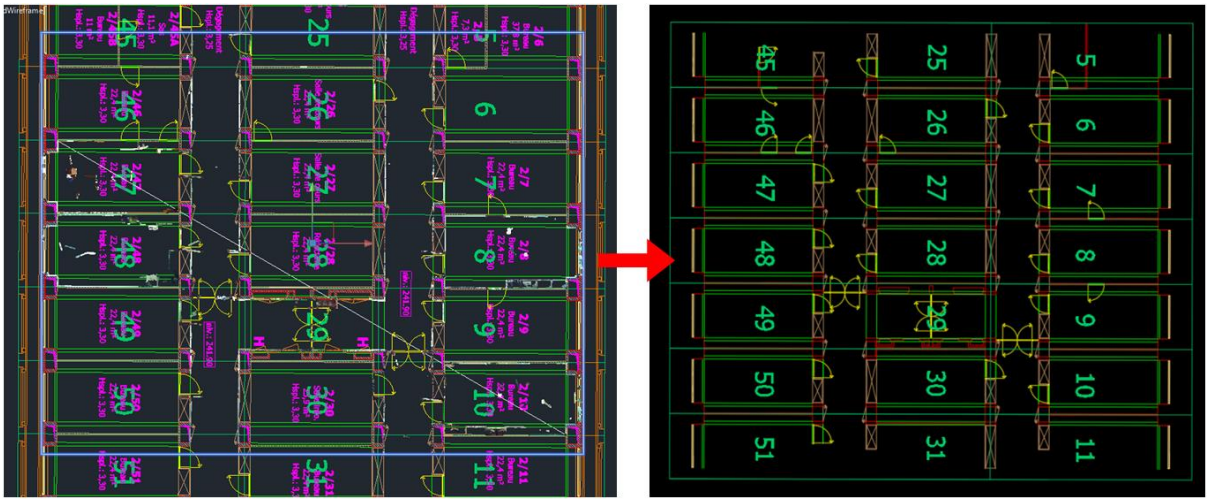


Figure 24 Résultat d'une première mise en forme des calques et du plan. La grille de numérotation a été laissée pour une meilleure visualisation. (Le cadre bleu correspond à l'emprise du nuage)

La deuxième partie de la mise en forme va aussi consister à nettoyer le plan de certains de ses éléments mais aussi à en retracer d'autres. En effet, trois problèmes se posent.

Tout d'abord, un grand nombre d'entités du plan sont formées de segments discontinus et/ou parfois superflus ou alors sont représentées via des « blocs » (entités caractérisées par un seul point d'accroche). C'est notamment le cas des armoires encastrées ou des portes qui sont représentées par des symboles. Il existe aussi d'autres segments superflus dont il faut se débarrasser : les segments invisibles par le scanner 3D. Un nuage de points 3D n'est, en effet, qu'une représentation des éléments visibles de la scène, c'est-à-dire de l'interface visible des objets (Xiong & Huber, 2010). Seuls les éléments du plan et qui ont été effectivement scannés (visibles dans le nuage) seront donc conservés. Les autres éléments (superflus) n'engendreront que des segmentations nulles : l'objet qu'ils représentent dans le plan ne l'est pas dans le nuage.

Chacun de ces segments superflus et non-liés devra être analysé indépendamment. Or, les segments superflus n'engendreront que des segmentations nulles et les segments non-liés formant un seul et même objet pourraient n'être analysés qu'une seule fois. Sachant que chaque analyse nécessite de tester l'ensemble des points du nuage, de tels segments ralentiront l'algorithme.

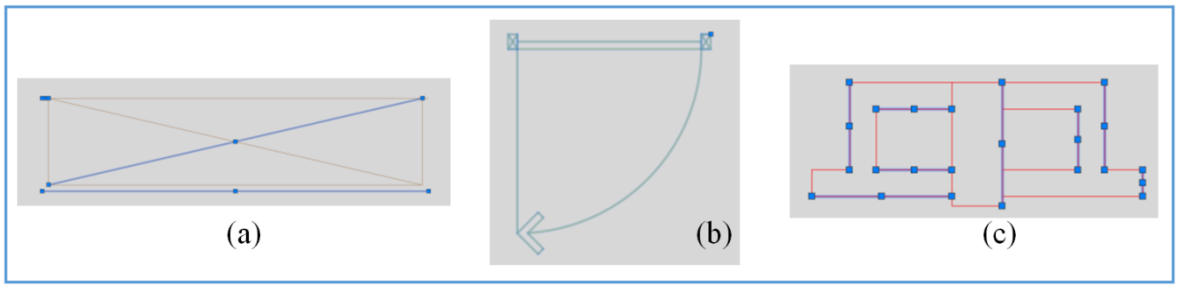


Figure 25 Exemples de segments superflus et discontinus au sein d'un symbole (a), d'un « bloc » représentant une porte (b) et de segments superflus et discontinus dans la représentation d'un mur et de colonnes (c)

Un second problème découle de l'utilisation de la librairie Shapely lors de la deuxième technique de segmentation. Cette librairie va être utilisée pour transformer les entités du plan en objets planaires 2D définis par la librairie : Polylignes et Polygones. Grâce à cette transformation, nous pourrons utiliser une fonction de cette librairie pour calculer une zone tampon autour de ces objets (et donc de notre entité). Cette zone tampon ne sera, par contre, pas la même pour une polyligne (intérieur non compris dans le tampon) ou un polygone (intérieur compris). De plus, ce calcul de zone tampon ne donnera un résultat concluant qu'avec des entités tracées de certaines manières :

1) Si l'entité est représentée sur le plan par un **bloc**, elle ne sera caractérisée que par son point d'accroche et la zone tampon ne pourra pas être calculée.

2) Si l'entité est représentée sur le plan par une **polyligne fermée** ou par une **forme géométrique** (rectangle par exemple), le résultat du calcul de tampon variera. La zone tampon sera correctement calculée si l'entité est transformée en polygone mais le dernier segment sera absent du calcul si elle est transformée en polyligne.

3) Si l'entité est représentée sur le plan par une **polyligne non fermée**, alors la zone tampon pourra être calculée correctement dans le deux cas (polyligne et polygone).

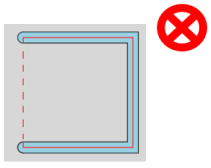
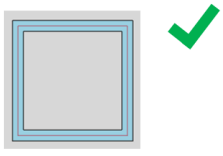
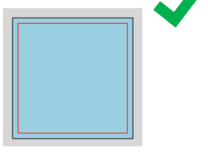
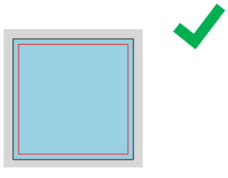
	Ligne fermée	**Ligne non fermée/Forme**
Polyligne (intérieur non-compris)		
Polygone (intérieur compris)		

Tableau 3 Zone tampon selon le type d'objet Shapely et la méthode de tracé

Enfin, le dernier problème est l'absence de certains éléments sur le plan, à savoir des portes et armoires, au niveau du couloir central :

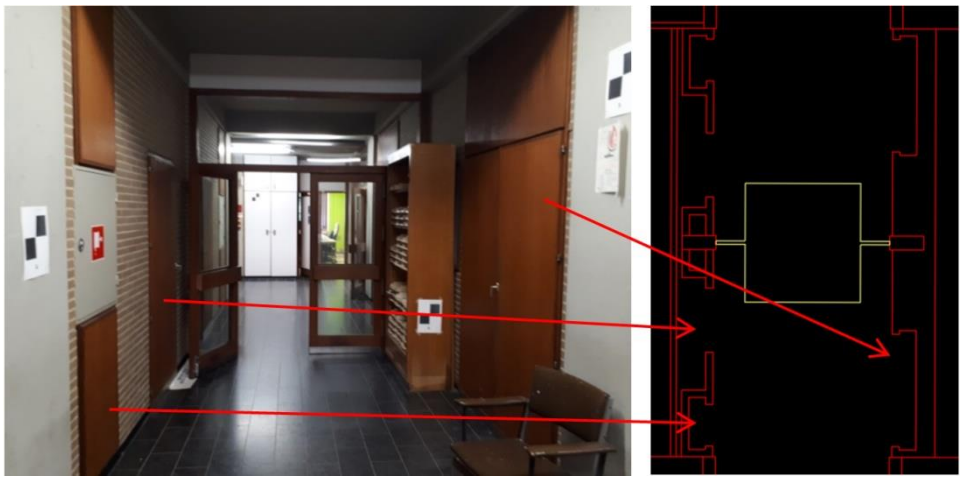


Figure 26 Illustration de l'absence de certains éléments sur le plan (ici situés dans le couloir central)

Pour résoudre ces trois problèmes, nous avons :

- supprimé les entités représentant des objets non-visibles dans le nuage de points
- retracé toutes les entités conservées, de manière à supprimer les segments superflus et à lier les segments discontinus
- ajouté les éléments absents de manière à compléter le plan.

Lors du traçage nous avons pris en compte les besoins de la librairie Shapely et utilisé des polylignes non-fermées. Ainsi, peu importe la transformation, la zone tampon est bien calculée. Pour chaque objet nous avons gardé uniquement les parties visibles de son contour :

- les murs (en rouge) deviennent des lignes continues,
- les poutres (en vert) et les fenêtres (en bleu) deviennent des rectangles (chassis inclus pour les fenêtres),
- les meubles encastrés (en blanc) deviennent soit des rectangles soit des lignes simples selon que le fond est visible dans le scan ou pas,
- et les portes (en jaune) deviennent des polygones, englobant le chassis et adaptés selon l'angle d'ouverture de la porte.

Cela donne le résultat général illustré en annexes (cf. IX.4, Fig. 55 et 56). Puisque nous travaillions avec les sous-nuages de bureaux, il a fallu reprendre hors de ce plan général, les sous-plans correspondants. Ceux-ci sont présentés en annexes (cf. IX.4, Fig. 57 à 62).

V.3. Extraction d'informations

Les données étant prétraitées, le développement de l'algorithme peut commencer. Il faut d'abord importer les données puis en extraire et structurer les informations utiles. Pour réaliser cela, nous utilisons deux librairies. (cf. IX.1).

- La librairie Laspy a été utilisée pour lire le nuage « .las » et en extraire les données. Les différentes informations de chaque point sont isolées dans des listes propres à chaque attribut. Ces listes sont enregistrées sous une matrice Numpy car il est plus facile de manipuler les données des points sous cette forme. L'ensemble des informations de chaque point est conservé durant les segmentations. Elles seront nécessaires pour transformer les matrices résultats obtenues en fichiers « .las ».

- La librairie Dxfgrabber est utilisée pour lire le plan « .dxf » et en extraire les informations. Parmi ces informations, nous avons enregistré dans deux listes de données distinctes les noms des calques et les entités présentes dans le dessin. Avec ces informations, nous avons alors créé un dictionnaire. Il s'agit d'une structure de données fonctionnant par paire clé-valeur. Dans notre cas, la clé correspondait au nom d'un calque et la valeur associée correspondait à une liste des entités appartenant à ce calque. Lors des premiers tests, tous les calques sont traités. A partir de la généralisation à toutes les classes, seuls ceux correspondant à nos classes sont traités.

V.4. Segmentations partielles et généralisation

V.4.1. **Cadres capables**

Comme détaillé dans la méthodologie, la première méthode de segmentation mise au point a été basée sur l'utilisation du cadre capable de l'objet traité. Pour conserver le nuage de points initial, les segmentations ont été réalisées sur une copie de celui-ci nommée '*remains*'. Cette matrice sera aussi utilisée pour conserver les points non-classés d'une segmentation à l'autre. Une liste '*segmented*' était aussi définie pour enregistrer les points segmentés. La première version de la méthode permettait de traiter un segment d'une entité du calque 'mur' du jeu de données de test. Une fois mise au point, la méthode a été d'abord généralisée pour segmenter tous les segments d'une entité puis toutes les entités du calque.

V.4.1.1. Segmentation d'un segment d'une entité du calque 'Mur'

La première étape a consisté à récupérer, depuis le dictionnaire créé précédemment une des entités du calque 'Mur'. De cette entité, nous avons extrait les coordonnées des deux sommets : elles constituent en effet les coordonnées limites du cadre capable du segment. Les points du nuage sont alors passés en revue un à un et testés pour déterminer s'ils sont compris au sein du cadre capable :

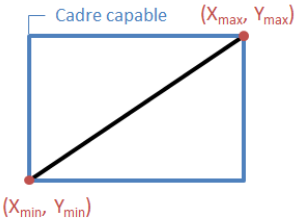


Figure 27 Cadre capable d'un segment

$$X_{min} \leq X \leq X_{max} \quad \textbf{et} \quad Y_{min} \leq Y \leq Y_{max}.$$

Si les deux conditions sont respectées alors le point est compris dans le cadre et segmenté. Le point et toutes ses informations sont ajoutés à la suite d'une liste *'segmented'*. Les points non-segmentés sont, eux, ajoutés à la suite d'une liste *'remains_list'*. Une fois tous les points testés, les deux listes sont remplies et constituent nos résultats. En les aditionnant, on obtient bien tous les points du nuage initial : rien n'est perdu, tout est classé. Ces deux listes sont alors transformées en matrice Numpy de manière à pouvoir être enregistrées en tant que nuage de points. On obtient ainsi notre premier résultat de segmentation, pour un segment d'une entité du calque 'mur' :

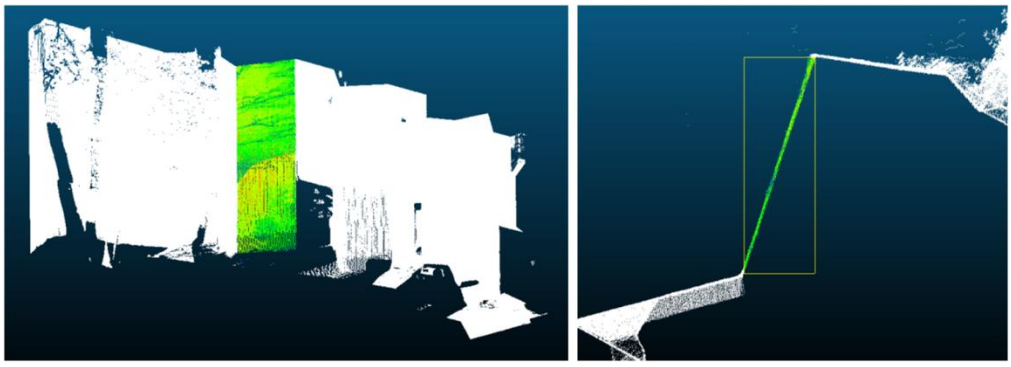


Figure 28 : Premier résultat de segmentation (en couleurs) par cadre capable (en jaune). Vue 3D à gauche et vue du haut à droite.

V.4.1.2. Segmentation d'une entité du calque « Mur »

La technique a été ensuite généralisée à une entité toute entière de ce calque. Pour conserver une certaine précision, la segmentation de l'entité a été réalisée en utilisant les cadres capables successifs des segments de l'entité. En effet, utiliser le cadre capable total de l'entité aurait été plus imprécis.

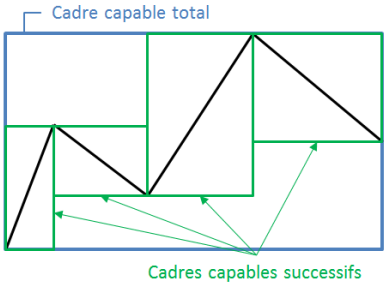


Figure 29 Comparaison de la précision des cadres capables

Pour réaliser cela, nous avons d'abord récupéré et enregistré dans une matrice les coordonnées des différents sommets successifs de l'entité. Nous avons alors utilisé une boucle pour segmenter successivement les différents segments de l'entité. A chaque itération (segment), les coordonnées des deux sommets du segment sont reprises depuis la matrice créée puisqu'elles correspondent aux limites de son cadre capable. La segmentation est alors effectuée comme précédemment : les coordonnées de chaque point du nuage sont testées par rapport aux coordonnées limites de ce cadre pour déterminer si le point y est contenu. Ainsi, itération après itération (segment après segment), la segmentation de l'entité est réalisée.

Le principe de test reste le même mais l'enregistrement des points est particulier. A la première itération (premier segment), les points vérifiant les conditions sont segmentés et ajoutés à la liste *'segmented'* et ceux ne la vérifiant pas sont ajoutés à la liste *'remains'*. A la fin de cette itération, les points de *'remains'* sont utilisés pour former le nuage testé à la deuxième itération. Les points segmentés durant cette 2ème itération sont ajoutés à la suite de la liste *'segmented'* de la première itération. Les points non segmentés sont de nouveaux utilisés pour reformer un nuage pour l'itération suivante, et ainsi de suite. On s'assure par là, que les points ne soient segmentés qu'une seule fois. Ainsi, après la dernière itération, nous avons donc deux nuages résultats : *'segmented'* avec l'ensemble des points segmentés lors des itérations, et *'remains'* avec les points restants.

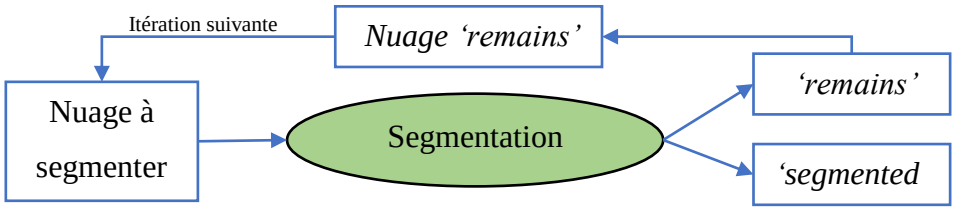


Figure 30 Schéma de la segmentation par itération et enregistrement des résultats

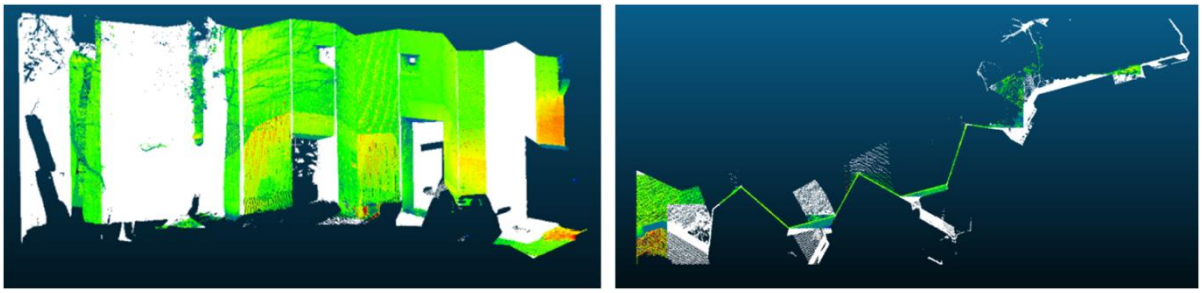


Figure 31 Résultat de la généralisation de la méthode de segmentation à tous les segments d'une entité. En couleurs, les points segmentés, et en blanc, ceux restants (Vue du haut à droite)

V.4.1.3. Segmentation des entités du calque « Mur »

Enfin, cette technique a été généralisée à la segmentation des toutes les entités du même calque. Pour ce faire, nous avons utilisé une boucle parcourant successivement les différentes entités du calque et les segmentant selon la méthode expliquée ci-dessus (cf. V.4.1.2). De la même manière, nous n'utilisons qu'une seule liste *'segmented'* pour enregistrer les points segmentés pour les différentes entités. Elle est transformée en nuage résultat une fois toutes les segmentations réalisées. Les points non-segmentés sont ici aussi réutilisés d'une segmentation d'entité à l'autre de sorte qu'aucun point ne soit segmenté plusieurs fois. Malgré les deux murs dessinés sur le plan, un seul de ces murs était représenté dans le nuage de test utilisé. Le résultat est donc identique au précédent puisque seule une entité a pu être segmentée.

V.4.2. **Zones tampons orientées**

La technique de segmentation par cadre capable possède plusieurs défauts qui seront discutés par après (cf. VI.1.4). La deuxième technique se base sur le même principe : on teste si les points du nuage sont compris dans une certaine zone délimitée, ici une zone tampon calculée autour de l'entité. Comme expliqué dans les prétraitements, la zone est calculée en utilisant la librairie Shapely. Les opérations d'importation / extraction d'informations restent identiques mais on utilise à partir d'ici les vraies données de travail : celles du 2ème étage du bâtiment B5a (ULiège, Sart-Tilman). Seule petite modification : nous ne reprenons parmi les calques que ceux commençant par « PCS_ » (cf. V.2.3) pour créer le dictionnaire. Nous travaillons toujours à un niveau de généralisation intermédiaire : la segmentation des entités d'un seul calque.

Comme avec les cadres capables, la méthode commence par la récupération des entités du calque traité. Nous avons traité le calque « Murs » pour débuter : nous avons donc repris les entités enregistrées dans le dictionnaire sous une clé (nom de calque) incluant le terme « Wall ». La clé « PCS_Wall » correspondant au nom de calque créé lors des prétraitements est donc reconnue et les entités associées récupérées.

Chacune des entités récupérées est alors transformée en une polyligne Shapely via les coordonnées de ses sommets. Ces polylignes Shapely sont des objets plans 2D définis dans Shapely et propres à cette librairie. Il s'agit d'une classe d'objets (au sens orienté-objet) de la librairie possédant donc des caractéristiques propres ainsi que des méthodes associées. Parmi ces méthodes qui peuvent leur être appliquées, nous retrouverons le calcul de la zone tampon.

Les murs ont été transformés en polylignes car ils correspondent, sur le plan, à des lignes simples ne formant pas de polygone (cf. V.2.3). Une fois les entités transformées, nous avons alors calculé la zone tampon de chaque polyligne pour une valeur seuil (*s*) de 0,025m (arbitraire). Nous avions ainsi déterminé l'ensemble des zones à tester.

Le test des points se fait en utilisant une autre méthode de la librairie Shapely : "objectA.contains(objectB)". Cette méthode est appliquée, dans notre cas, à une surface (la zone tampon) et permet de déterminer si un autre élément (dans notre cas, un point) y est contenu. Le résultat est donc binaire : vrai ou faux (Gillies, 2018). Toutefois, pour pouvoir utiliser cette méthode avec nos points, ceux-ci ont dû être transformés en point Shapely. Comme les polylignes Shapely, ces points Shapely correspondent à une classe d'objets définie par la librairie et possédant des caractéristiques propres et des méthodes associées. Nous avons donc créé un point Shapely pour chaque point de notre copie du nuage.

Nous avons pu alors réaliser notre segmentation. Pour chaque zone tampon (entité), chaque point Shapely est passé en revue. La méthode "contains" est utilisée pour savoir si le point est contenu dans le tampon. Si la méthode renvoie « vrai » alors le point est segmenté. Comme précédemment les points segmentés sont ajoutés à une seule liste maintenant nommée *'PC_Mur'*. Il s'agit bien ici des points du nuage et non des points Shapely. Ces derniers ne sont utilisés que pour le test et ne constituent pas un résultat de la segmentation. Les points non-segmentés sont, comme auparavant, ajoutés à une liste *'remains'* pour être réutilisés dans la segmentation suivante. Le nuage à segmenter ayant ainsi changé, les points Shapely calculés ne correspondaient plus à ceux de ce nouveau nuage. Entre chaque entité traitée (chaque segmentation), les points Shapely devaient donc être recalculés pour ce nuage. Nous avons ainsi deux nuages résultats : *'PC_Mur'* et *'remains'*.

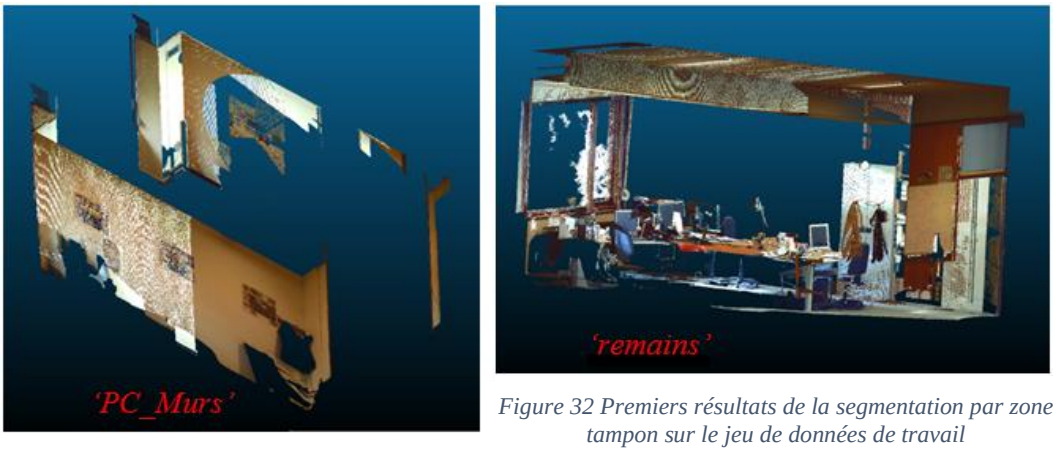


Figure 32 Premiers résultats de la segmentation par zone tampon sur le jeu de données de travail

V.5. Optimisation des temps de traitement

Si les résultats obtenus grâce à cette deuxième technique sont plus précis et plus exacts, le temps de traitement est relativement long : quasiment 12 minutes pour segmenter le calque 'PCS_Wall'. Or, cette méthode devra être appliquée aux autres calques ce qui allongera encore ce temps. La durée des traitements devait donc être réduite pour faciliter la suite des tests et pour améliorer l'efficacité de l'algorithme. Une première opération à réaliser est de travailler sur les données : les échantillonner ou les indexer. Ensuite, il est aussi possible de travailler directement sur l'algorithme en améliorant ses différentes parties.

V.5.1. **Indexation & Echantillonnage**

Ces deux possibilités de réduction des temps de traitement peuvent être mises en œuvre avant le développement des méthodes de segmentation et de l'algorithme. Elles sont présentées ici pour être rassemblées avec les autres techniques permettant d'améliorer la rapidité des traitements. Elles seront appliquées aux données avant de commencer à segmenter.

Le principe de l'indexation a été rappelé au sein de la méthodologie. Deux types d'index spatiaux ont été considérés dans notre cas : l'octree et le KD tree. (Illustrations en annexes)

- L'octree se caractérise par une division régulière de l'espace à la manière d'un quadtree (cf. IV.5). L'espace et les sous-espaces sont récursivement divisés en huit volumes égaux en se basant sur des plans orthogonaux aux axes du système de coordonnées. La division est donc régulière mais peut nécessiter un nombre important de niveaux de précision (profondeur de l'index) et produire un grand nombre de sous-espaces vides. Cela peut engendrer des temps de parcours importants. (Duncan et al., 2001 ; Saftly et al., 2014) (cf. IX.6)

- Le KD tree utilise lui aussi des plans orthogonaux aux axes du système de coordonnées. Il subdivise l'espace récursivement en deux. Pour ce type d'indexation, l'orientation et la localisation des plans de division peuvent varier. Vu qu'ici l'espace et les sous-espaces sont divisés en deux, la profondeur de l'index sera souvent plus importante que celle de l'octree. Le nombre d'opérations réalisées et de sous-espaces parcourus lors d'une recherche reste en moyenne identique dans les deux cas, le KD tree demandant moins de tests par niveau. Le KD tree produira moins de sous-espaces que l'octree. (Duncan et al., 2001 ; Saftly et al., 2014) (cf. IX.6)

Les deux types d'indexation sont d'une efficacité assez semblable et nous avons essayé de les mettre en œuvre pour les tester. Pour ce faire, nous avons employé quelques librairies proposant des méthodes pour appliquer ces indexations : Pyoints, Boxtree, Open3D, etc. Toutefois, plusieurs problèmes se sont présentés. Pour commencer, certaines librairies n'ont pu être installées dans Anaconda notamment à cause de problèmes de dépendances. Ensuite, pour celles qui ont pu être installées, le manque de documentation rendait leur utilisation compliquée. Enfin, pour celle que nous avons pu installer et utiliser, le manque de méthodes associées à l'indexation et à son résultat ne nous a pas permis de les utiliser comme nous l'entendions. Tous ces problèmes ont fait que l'indexation n'a pas pu être mise en œuvre. Cette limitation sera discutée par après (cf. VI.3)

Nous avons aussi testé et mis en œuvre un échantillonnage des données. Cela va nous permettre d'accélérer les traitements en réduisant le nombre de points du nuage utilisé (cf. IV.5) Cette méthode engendre donc une perte d'informations puisque seuls les points repris dans ce nouveau nuage seront segmentés dans nos différentes classes. Pour déterminer le degré d'échantillonnage (pourcentage de points restants), nous avons testé différentes valeurs de divisions de l'échantillon : 1 point sur 5 (20%), 8 (12.5%), 10 (10%), 12 (8.3%), 15 (6.66%) et 20 (5%). Lors des améliorations présentées ci-après, nous avons, à chaque fois, chronométré le traitement pour chacun de ces échantillonnages. Cela nous a permis de suivre la diminution du temps de traitement selon les améliorations effectuées, mais aussi selon les différents degrés d'échantillonnage utilisés.

V.5.2. **Optimisation de l'algorithme de segmentation**

Les améliorations présentées ici sont basées sur les résultats des chronométrages partiels (étape par étape) et totaux de la segmentation. Les temps de traitement totaux obtenus après ces différentes améliorations effectuées sur notre méthode de segmentation sont repris dans un tableau récapitulatif. Chaque amélioration ou « version » de l'algorithme correspond à une ligne du tableau. Nous avons aussi réalisé un second tableau reprenant la différence relative lors du passage d'une version à la suivante (en % de l'ancienne). Par exemple, si on passe de 50 secondes à 40 secondes, on aura une différence relative de -20%.

$$\text{Différence relative entre } Vi \text{ et } Vi+1 = \frac{((Vi+1)-Vi)}{Vi} * 100$$

Temps de traitement des différentes versions de la méthode de segmentation par zone tampon, appliquée au nuage Office_1								
Temps de traitement en secondes (s)		Degrés d'échantillonnage						
		1	1/5	1/8	1/10	1/12	1/15	1/20
VERSIONS	V1	215	46	30	22	19	16	12
	V2	734	140	89	60	52	42	31
	V3	349	68	42	35	29	23	18
	V4	273	52	33	26	22	18	14
	V5	Pas de modification par rapport à V4						
	V6	/	103	63	51	42	34	25
	V7	/	108	66	48	40	32	24
	V8 - finale	/	141	87	68	60	49	36

CODE COULEURS
Tests avec une entité
Tests avec toutes les entités d'un calque
Tests avec toutes les entités de tous les calques

Tableau 4 Temps de traitement selon les versions de l'algorithme et le degré d'échantillonnage du nuage (Office_1), code couleurs associé.

Augmentations relatives des temps de traitement d'une version de la méthode de segmentation par zone tampon à la suivante (pour le nuage Office_1)									
Augmentations relatives du temps de traitement (%)		Degrés d'échantillonnage							Moyenne (%)
		1	1/5	1/8	1/10	1/12	1/15	1/20	
VERSIONS	V1-V2	241.3953	204.3	196.7	172.73	173.68	162.5	158.33	187.09
	V2-V3	-52.4523	-51.43	-52.81	-41.67	-44.23	-45.24	-41.94	-47.11
	V3-V4	-21.7765	-23.53	-21.43	-25.71	-24.14	-21.74	-22.22	-22.94
	V4-V6	/	98.08	90.91	96.154	90.909	88.89	78.571	90.58
	V6-V7	/	4.854	4.762	-5.882	-4.762	-5.882	-4	-1.82
	V7-V8	/	30.56	31.82	41.667	50	53.13	50	42.86

Tableau 5 Augmentations relatives des temps de traitement selon les versions de l'algorithme et le degré d'échantillonnage du nuage (Office_1)

Ces résultats seront analysés dans la discussion (cf. VI.1.6.2).

La première version (V1) de la méthode correspond à celle présentée au point 4.2 de ce chapitre, appliquée à une entité. La seconde version (V2) correspond à cette même technique mais appliquée à toutes les entités du calque. La version trois (V3) correspond à la première amélioration réalisée et que nous allons détailler maintenant.

Grâce aux chronométrages partiels des différentes parties de la méthode, nous nous sommes rendus compte que les parties jusqu'à l'extraction de l'information ne correspondaient qu'à 3% du temps total. La création des points Shapely prenait, elle, 56% du temps de traitement total. La segmentation, c'est-à-dire le test des points et leur ajout dans les listes résultats, représentait 41% du temps total.

La version 3 (V3) de la méthode va réduire le temps octroyé à la création des points Shapely. Dans ce but, nous avons fait en sorte de ne pas devoir recréer ces points après chaque segmentation. A la place, les points sont créés une fois pour le nuage intial (après son échantillonnage). Durant la segmentation, lorsqu'un point ne vérifie pas les conditions pour être segmenté, il est ajouté à une liste utilisée pour reformer le nuage à segmenter par après. Pour éviter de recréer les points Shapely, ceux-ci sont aussi ajoutés à une liste lorsque le point n'est pas segmenté. L'ajout se faisant avant de passer au point suivant, les points du nuage et Shapely sont à la même position dans leur liste respective. On obtient donc, après la segmentation, trois listes résultats : une avec les points segmentés, une avec les points du nuage non-segmentés et une avec les points Shapely correspondant aux points de la liste « non-segmentés ». Ces deux dernières listes seront utilisées pour la segmentation suivante. Nous avons ainsi réduit le temps de création des points Shapely à environ 20% du temps

La version 4 (V4) de la méthode va accélérer la partie consacrée à la segmentation. La solution employée est de ne parcourir qu'une seule fois les points du nuage. Pour cela, nous avons inversé une partie du processus. En effet, plutôt que de tester tous les points pour chaque zone tampon, nous avons fait en sorte de tester chaque zone tampon lors de l'analyse d'un point. Cela nous a permis de réaliser toute la segmentation d'un calque en une seule boucle sur les points. Cela signifie aussi que la réécriture du nuage et des points Shapely restants ne se fait qu'une fois par calque, lorsque la segmentation est terminée. Après cette réduction, le temps total a été diminué mais les parts relatives de la création des points et de la segmentation restent sensiblement identiques à leur proportion précédente (20-25% et 65 à 70%, respectivement).

Pour réduire encore plus le temps de traitement, nous avons analysé et chronométré les deux parties de la segmentation : le test des points et l'enregistrement dans la liste. Nous avons constaté qu'il y avait un ordre de grandeur de différence entre le temps de calcul de la condition (98% en moyenne du temps nécessaire à la partie segmentation) et celui de l'enregistrement. Nous n'avons pas trouvé de solution plus rapide pour calculer cette condition.

```
condition:  0.000658159488466481 secondes
enreg. :  8.980733809949015e-06 secondes
False
condition:  9.06626455616788e-05 secondes
enreg. :  1.7106160612456733e-06 secondes
False
condition:  8.382018177144346e-05 secondes
enreg. :  3.4212318951176712e-06 secondes
False
```

Figure 33 Exemples de résultats du chronométrage du calcul de la condition et de l'enregistrement lors de la segmentation. Les « False » sont les résultats des tests

Enfin, la 5ème version de la méthode de segmentation n'a pas pour but d'améliorer la rapidité des opérations mais plutôt d'améliorer la visualisation des résultats. Comme expliqué dans la méthodologie (cf. IV.8), il y a plusieurs manières d'enregistrer les résultats. Avec cette version (V5), nous combinons les deux techniques citées dans la méthodologie : un fichier par classe et un tag de classification par classe. Les points du nuage possèdent déjà un attribut « classification » que nous modifions simplement lors de la segmentation. Le nuage résultant de la segmentation d'une classe est toujours enregistré dans un fichier « .las » indépendant. A la fin de l'algorithme, nous rassemblerons tous les sous-nuages et les enregistrerons en un nuage total contenant le tag de classification pour chaque point. Ce nuage pourra alors être visualisé en appliquant une échelle ordonnée de couleurs basée sur les tags de classification. Ce nuage nous sera aussi utile lors de la validation. Cette opération n'impacte pas le temps de traitement.

CLASSES	Tag de classification
« Remains »	1
« Sol-Plafond »	2
« Murs »	3
« Poutres »	4
« Meubles encastrés »	5
« Portes »	6
« Fenêtres »	7

Tableau 6 Classes et leur tag de classification associé

V.6. Généralisation de la segmentation à toutes les classes (V6)

Dans cette partie, nous allons appliquer la méthode de segmentation par zones tampons développée jusqu'ici aux autres classes découlant des calques (la classe « Sol-Plafond » n'est pas encore traitée). Nous réalisons ces segmentations selon un certain ordre inspiré notamment de celui établi lors de la prise en mains : murs, poutres, meubles encastrés, portes et enfin fenêtres. Cet ordre pourra être modifié par la suite, notamment lorsque nous aurons développé la segmentation du sol et du plafond et affiné celle des autres classes.

Pour chaque segmentation, chaque classe, nous avons commencé par adapter la récupération des entités au calque adéquat. Nous avons passé en revue les clés présentes dans le dictionnaire et avons repris les entités lorsque le nom de cette clé comprenait le terme associé à notre classe : 'Beam' pour « Poutres », 'Door' pour « Portes », etc.

Nous avons ensuite adapté le calcul de la zone tampon en fonction du type d'entité à segmenter dans la classe. Pour rappel, lorsqu'une entité est transformée en polyligne, la zone tampon ne prend en compte que les contours tandis que pour un polygone, l'intérieur de l'entité est compris dans la zone tampon (cf. V.2.3).

- Pour le mur, nous avons utilisé des polylignes comme expliqué lors du développement de l'algorithme ci-dessus (cf. V.4.2)

- Les poutres possèdent par contre une face horizontale que l'on doit reprendre, ces entités ont donc été transformées en polygones

- Une polyligne aurait pu suffire pour les meubles encastrés car ils ne dépassent pas du mur et sont tous fermés. Toutefois, dans une réflexion plus générale, ces meubles peuvent dépasser du mur et leur intérieur peut être visible s'ils sont ouverts. La polyligne n'aurait alors pas été suffisante, ils ont donc été transformés en polygones

- Les portes et les fenêtres, comme les meubles encastrés, sont mobiles et ont plusieurs configurations possibles. Elles ont aussi été redessinées de manière à ce que l'intérieur soit repris : nous les avons donc transformées en polygones.

Ainsi, toutes les entités sont transformées en polygones si ce n'est celles du calque mur. L'influence des différences de configuration, du type des objets, de leur variabilité, etc. sera analysée dans la discussion (cf. VI.1.7 et Fig. 49).

Nous avons aussi adapté le tag de classification à la classe segmentée. C'est aussi dans cette étape que nous avons paramétré l'enregistrement du nuage résultat de chaque classe et l'enregistrement du nuage total.

Une fois tout cela mis en place et l'algorithme lancé, nous avons obtenu des premiers résultats de segmentation totale. Les temps de traitement ont été calculés pour les différents degrés d'échantillonnage et sont repris dans le tableau sous « V6 » (Version 6).

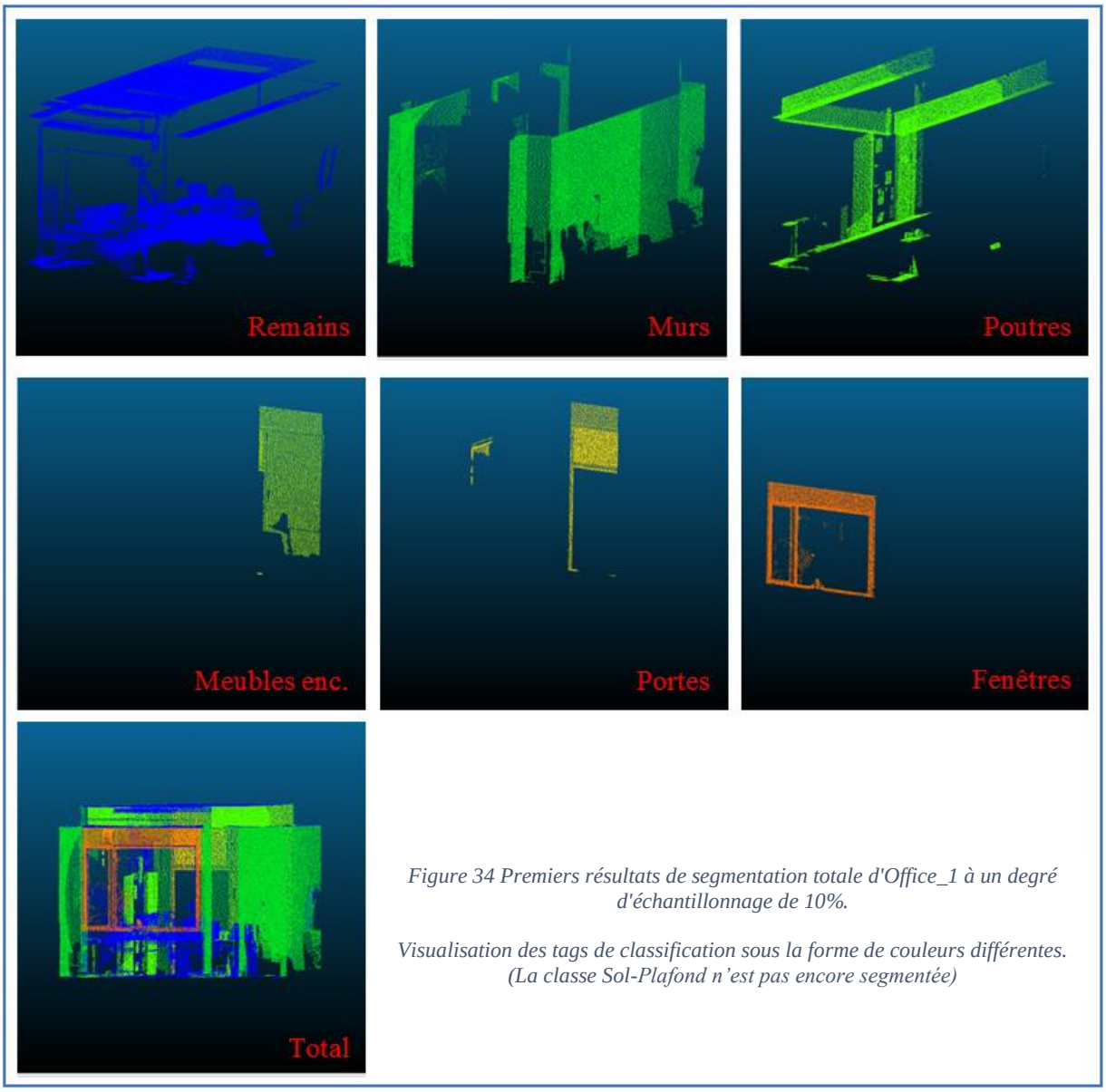


Figure 34 Premiers résultats de segmentation totale d'Office_1 à un degré d'échantillonnage de 10%.

Visualisation des tags de classification sous la forme de couleurs différentes. (La classe Sol-Plafond n'est pas encore segmentée)

V.7. Amélioration de la segmentation par classe

Après avoir obtenu ce premier résultat, nous avons passé en revue les différentes classes de notre classification et tenté d'améliorer les résultats individuels de chaque segmentation. Notamment, nous avons pris en compte les hauteurs dans certaines de ces segmentations. Pour commencer, nous avons développé une segmentation pour la classe « Sol-Plafond ». Le principe de cette segmentation a été présenté dans la méthodologie (cf. IV.7) : il repose sur la détermination de pics de fréquence dans les coordonnées Z des points du nuage. Grâce à ces pics, des hauteurs limites pourront être déterminées.

V.7.1. **Segmentation de la classe « Sol-Plafond » (V7)**

Cette étape correspond à la version 7 de notre algorithme (cf. Tab. 4 et 5). Nous avons commencé par construire un histogramme de fréquences pour les hauteurs des points du nuage, par classe de 10cm. Parmi ces classes, nous avons repris, comme classe de plafond, celle de fréquences la plus importante située dans le dernier quart du nuage (au-delà de 75% de la hauteur du nuage). Similairement, nous avons repris la classe de fréquences la plus importante située dans le premier quart du nuage (en deça de 25% de la hauteur du nuage) comme classe de sol. Cela nous permet d'éviter de reprendre des classes de pic ne correspondant pas au sol ou au plafond (interférences).

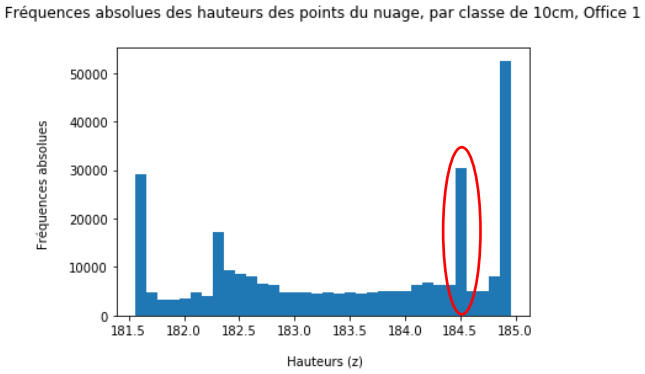


Figure 35 Exemple d'un histogramme construit pour le nuage "Office_1". Le pic entouré en rouge correspond à une interférence potentielle

NB : Dans les histogrammes présentés, si les données sont absolues, c'est qu'elles n'ont pu être normalisées à cause d'une division en classes absolues et non relatives (classes de 10cm et non 10 classes de 10% par exemple)

Une fois ces deux classes déterminées, nous devions définir pour chacune d'elles une valeur limite de segmentation. Plutôt que d'utiliser, pour ces valeurs, la borne supérieure de la classe de sol et la borne inférieure de la classe plafond, nous avons préféré augmenter la précision de discrimination. Nous avons ainsi appliqué une gaussienne aux données de chacune des deux classes. Avec cette gaussienne, nous avons pu déterminer un intervalle de confiance en calculant la moyenne (μ) et l'écart-type (σ) de ces données. Nous avons testé différents niveaux de confiance et finalement utilisé un intervalle à 95% de confiance : [$\mu \pm 1{,}96\,\sigma$].

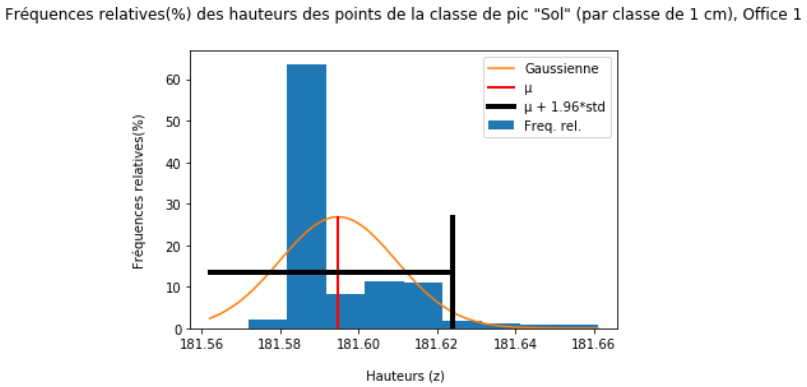


Figure 36 Données de la classe de pic "sol" pour le nuage "Office_1" et statistiques associées

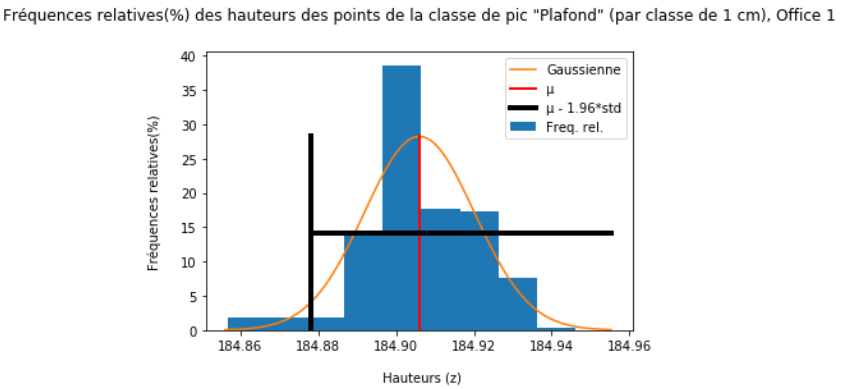


Figure 37 Données de la classe de pic "plafond" pour le nuage "Office_1" et statistiques associées

Nous avons alors segmenté les points selon la condition basée sur nos deux intervalles. Les points appartiennent à la classe « Sol-Plafond » s'ils sont à une hauteur inférieure à la borne supérieure de l'intervalle calculé pour la classe de fréquences « sol », ou s'ils sont à une hauteur supérieure à la borne inférieure de l'intervalle calculé sur la classe de fréquences « plafond ». Comme pour les autres segmentations, les points segmentés sont ajoutés à une liste *'PC_Sol_Plafond'* et les points non-segmentés à une liste *'remains'* utilisée dans la segmentation suivante.

La segmentation présentée ici se fait après la segmentation des poutres de sorte qu'aucune partie des poutres ne soit segmentée en même temps. En effet, dans le cas d'un faux plafond, les poutres peuvent être situées dans le même plan que ce plafond et être segmentées avec celui-ci. La segmentation des poutres se fait donc en premier lieu et nécessite l'utilisation de points Shapely. Or, étant donné que ces points Shapely ont été créés, ils doivent être pris en compte dans la segmentation de la classe « Sol-Plafond », bien qu'ils n'y soient pas utilisés. S'ils ne sont pas pris en compte, la liste des points non-segmentés réutilisée pour la segmentation suivante ne correspondra plus à la liste de points Shapely. Comme précédemment, ils sont simplement ajoutés à une liste lorsqu'un point correspondant du nuage n'est pas segmenté.

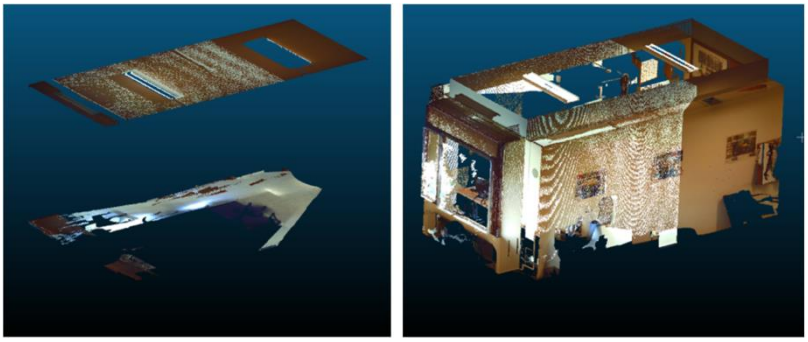


Figure 38 Résultat de la segmentation de la classe "Sol-Plafond" par rapport au reste du nuage (précédemment non-segmentée)

V.7.2. **Affinage des segmentations classe par classe (V8)**

V.7.2.1. Classe « Murs »

La segmentation des murs ne nécessite pas de modifier ou d'ajouter une partie à notre méthode initiale. Nous avons simplement déterminé une épaisseur pertinente pour nos zones tampons. Le plan du bureau « Office_1 » et le nuage associé (coupe de 30cm à 1,3m dans le nuage total « Office_1 ») ont été affichés dans le logiciel AutoCAD. Cela nous a permis de visualiser (en vue du haut) les extensions potentielles des zones tampons, ainsi que les éléments problématiques liés aux murs (affiches, meubles touchant le mur, occlusions, etc.). Nous avons alors testé quelques épaisseurs de zone tampon et retenu quelques-unes de ces valeurs : 15mm, 20mm, 25mm.

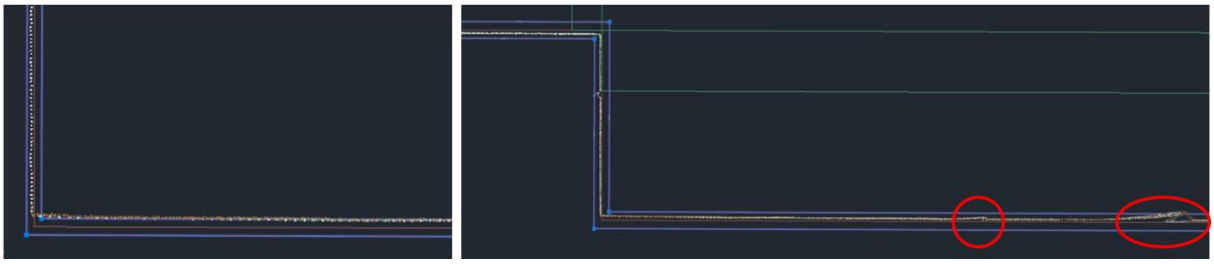


Figure 39 Exemple de tests d'épaisseur de la zone tampon par rapport aux murs de la coupe du nuage (15mm à gauche, 25mm à droite). Certains problèmes présents dans le nuage sont aussi visibles (en rouge) : une affiche (à gauche) et une chaise touchant le mur (à droite).

Pour déterminer quelle épaisseur utiliser au final, nous avons réalisé des tests de segmentations pour chacune des valeurs et visualisé les résultats dans CloudCompare. Avec une épaisseur de 15mm, il manquait certains morceaux de murs mais les éléments problématiques étaient aussi moins présents. A 20mm, nous avions récupéré certaines parties de murs mais aussi des éléments non-« Murs ». A 25mm, nous avions récupéré la totalité des murs mais aussi, de fait, plus d'éléments problématiques. Nous avons finalement choisi une épaisseur de 25mm (identique à la valeur que nous avions donnée par défaut). Il est, en effet, préférable de segmenter totalement les murs plutôt que d'essayer de réduire de petites erreurs qui seront présentes quelle que soit l'épaisseur choisie. Les affiches, tableaux et autres éléments fins accrochés aux murs n'ont donc pas pu être séparés des murs via notre segmentation. L'épaisseur déterminée ici a aussi été utilisée pour les autres segmentations. Cette limitation sera discutée par la suite (cf. VI.3)

Pour éviter que des parties de poutres ne soient pas réprises dans cette classe « Murs », nous avons décidé de segmenter les murs après les poutres.

V.7.2.2. Classe « Poutres »

Nous avons, pour cette classe, introduit une condition supplémentaire sur la hauteur des points (à vérifier en plus de la zone tampon). En effet, les poutres ne débutent qu'à une certaine hauteur : leurs points doivent être distingués des points situés dans la zone tampon de l'élément du plan, mais sous cette hauteur. Cette condition consiste donc en une simple hauteur seuil sous laquelle les points ne sont pas segmentés. La condition totale pour cette segmentation est donc un volume 3D : l'extension plane correspond à la zone tampon, tandis que l'extension verticale est, elle, limitée inférieurement par la valeur seuil de hauteur.

Nous avons ainsi dû déterminer une hauteur seuil produisant un résultat correct. Cette hauteur correspond à celle de la face horizontale inférieure des poutres. Nous posons l'hypothèse que cette hauteur est identique pour toutes les poutres. Nous avons utilisé la 1ère segmentation réalisée avec notre méthode et les statistiques de fréquence associées pour la calculer. La technique utilisée est similaire à celle de la segmentation du sol et du plafond.

Après avoir réalisé la première segmentation par notre méthode, nous avons construit un histogramme de fréquences (classe de 10cm) des hauteurs pour les points segmentés. Nous avons ensuite récupéré la classe de fréquences correspondant à la surface horizontale inférieure des poutres. En travaillant sur le résultat de cette première segmentation, nous évitons l'influence du reste des points, et notamment du plafond, dans notre histogramme. Il est toutefois probable que des points de sol, de tables et autres surfaces horizontales soient présents dans le nuage segmenté et engendrent des pics de fréquences interférents. Pour éviter ces interférences, nous avons repris la classe de fréquences la plus importante située dans le quart supérieur du nuage (comme pour le plafond). Cette classe de fréquences correspond normalement à celle recherchée.

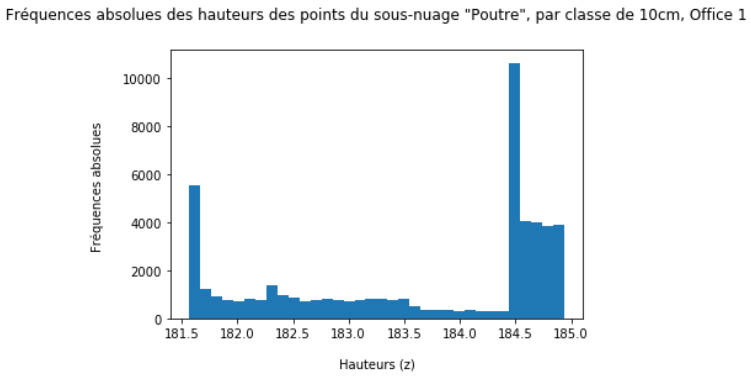


Figure 40 Exemple de l'histogramme construit sur les points du nuage segmenté par notre méthode. On constate que le sol engendre un pic interférent.

Comme pour la classe « Sol-Plafond », nous avons appliqué une gaussienne aux données de cette classe et calculé un intervalle de confiance à 95%. La borne inférieure de cet intervalle sera utilisée comme hauteur seuil dans la condition de segmentation. Les points déjà segmentés ont tous été analysés et ceux situés au-delà de cette hauteur ont été resegmentés.

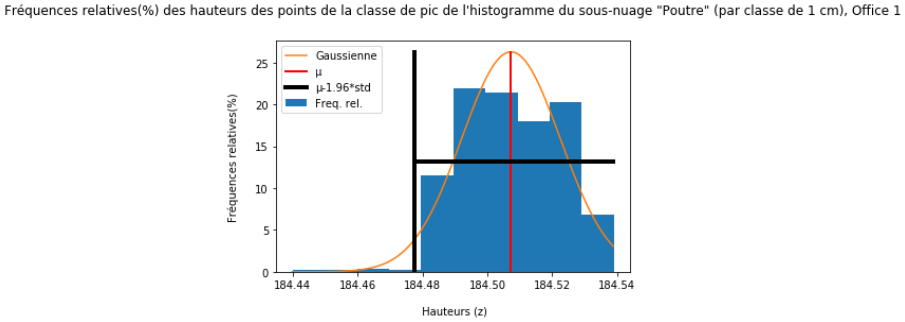


Figure 41 Histogramme des données issues de la classe de pic « Poutres » (produit par la première segmentation) et statistiques associées

Pour ce qui est de l'enregistrement des points, nous avons, à la suite de la 1ère segmentation, deux fois deux listes : deux listes pour les points segmentés, une contenant les points du nuage et une contenant les points Shapely correspondants, et deux listes similaires pour les points non segmentés (points du nuage et points Shapely correspondants). Nous réutilisons les points segmentés et leurs points Shapely dans la deuxième segmentation. Les points segmentés lors de cette 2ème segmentation sont ajoutés à une nouvelle liste qui deviendra le nuage résultat de la classe « Poutres ». Les points Shapely ne sont, cette fois, pas conservés. Les points non-segmentés lors de la deuxième segmentation et leurs points Shapely associés sont ajoutés à la suite des listes « non-segmentés » créées lors de la première segmentation. Ces deux dernières listes seront utilisées comme listes à segmenter par la segmentation suivante.

Le tag de classification n'est ajouté que lors de la deuxième segmentation pour ne pas l'attribuer à des points qui ne seraient pas resegmentés après la seconde segmentation.

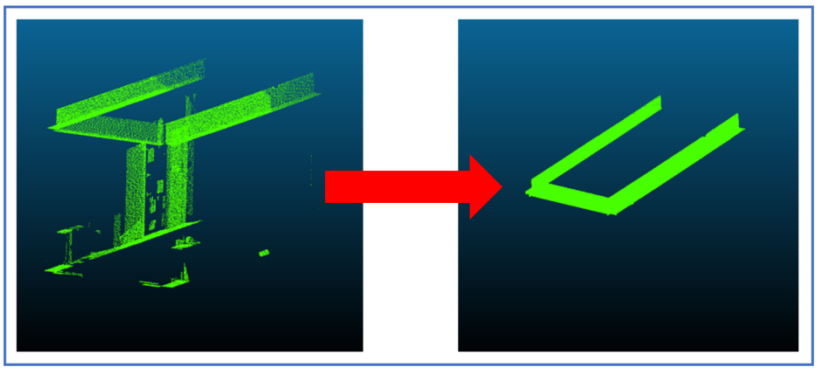


Figure 42 Comparaison entre la segmentation des poutres avant et après la mise en oeuvre de l'amélioration détaillée ci-dessus, pour le nuage « Office_1 ».

NB : le nuage 'avant' est échantillonné (à 10%), le nuage 'après' ne l'est pas.

V.7.2.3. Classe « Fenêtres »

Pour la classe « Fenêtres », nous n'avons pas développé de seconde segmentation pour prendre en compte les hauteurs. En effet, le nuage ne représente que les premières surfaces qui sont rencontrées par les rayons. Or la plupart du temps, une fenêtre est renfoncée dans un mur et donc située plus loin que l'interface de ce mur, comme on peut le voir dans la coupe ci-contre. La zone jaune correspond à la partie segmentée et la ligne rouge correspond à l'interface du mur scannée. L'espace entre les deux correspond au début de l'embrasure de la fenêtre et n'est pas segmenté. On remarque donc qu'à cause des occlusions engendrées par le mur et d'autres objets, aucun point n'est situé sous la fenêtre ou au-dessus de celle-ci. L'intégration des hauteurs dans la condition n'est pas nécessaire puisqu'elle n'engendrerait pas de modification. Il existe toutefois des zones qui ne sont pas segmentées que ce soit pour la fenêtre ou pour le mur.

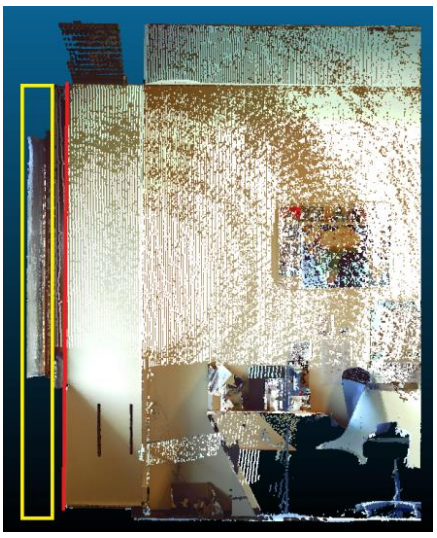


Figure 43 Coupe de profil d'une partie du nuage "Office_1"

V.7.2.4. Classes « Portes » et « Meubles encastrés »

Les problèmes de segmentation des portes et meubles encastrés sont similaires et seront donc traités simultanément. Tout d'abord, nous avons noté qu'il existait un décalage entre le plan et le nuage au niveau de la porte engendrant une mauvaise segmentation de celle-ci. Ce cas particulier illustre un problème qui peut se produire dans d'autres segmentations et qui dépend directement de la justesse du plan, du nuage et du géoréférencement.

Le but de cet affinage sera d'intégrer une condition de hauteur dans la segmentation afin de resegmenter le pan de mur situé au-dessus de ces éléments et repris dans leur classe. Dans ce cas-ci, les fréquences des hauteurs n'étaient pas assez discriminantes pour nous aider : la hauteur à laquelle débutait le mur n'était pas distinguable dans l'histogramme. Nous n'avons pas pu nous baser sur les vrais éléments (portes et meubles encastrés) à cause de leur mobilité et de leur variabilité (ouvert/fermé, simple/double porte, angle d'ouverture, etc.). Utiliser les fréquences selon l'axe X pour distinguer la transition entre la porte ou le meuble et le mur ne donnait pas de résultat satisfaisant non plus (transition non distinguable).

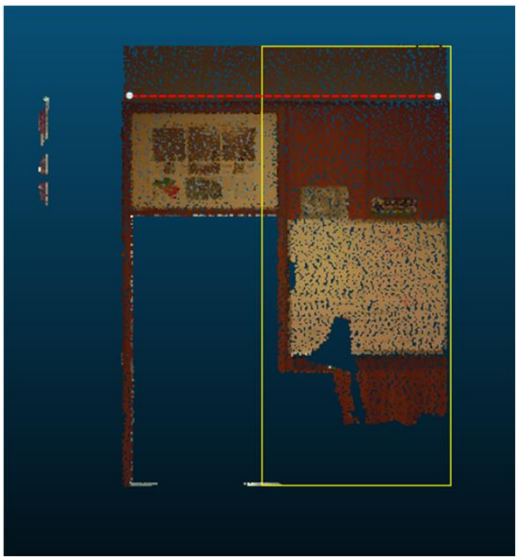


Figure 44 Illustration du problème de segmentation entre la porte et le meuble ainsi que du problème lié au pan de mur. En rouge, la limite entre les éléments et le pan de mur. En jaune, l'extension du sous-nuage « Meubles encastrés »

La solution que nous avons envisagée repose sur une constatation qui sera utilisée en tant qu'hypothèse de notre développement. Cette constatation/hypothèse est que les portes (et armoires) d'un bâtiment atteignent souvent la même hauteur. C'est notamment le cas dans nos données : les portes et les meubles s'arrêtent à une hauteur correspondant à la hauteur maximale des murs. Nous avons alors envisagé d'utiliser cette valeur maximale des murs comme valeur discriminante pour resegmenter nos portes et meubles encastrés. Cette technique ne fonctionnera cependant que si tous les éléments (portes et meubles) sont de même hauteur et que cette hauteur correspond bien à la hauteur maximale des murs.

- Si la hauteur de l'élément est supérieure à la hauteur maximale des murs (cas extrêmement rare, plus probable avec d'autres types de valeurs seuils), alors on segmentera une partie de l'élément avec le pan de mur ce qui nous fera perdre des points au départ bien segmentés.

- Si la hauteur de l'élément est inférieure à la hauteur maximale des murs, alors le pan de mur sera partiellement retiré, ou ne sera pas du tout resegmenté (si la hauteur maximale du mur correspond à la hauteur maximale du pan).

Ainsi, tant que la hauteur maximale du mur est égale ou supérieure à la hauteur des portes et meubles encastrés, la segmentation sera soit améliorée (partiellement ou totalement) soit elle ne sera pas modifiée. Le problème serait d'avoir une hauteur de mur plus grande que les éléments : la segmentation serait alors détériorée (peu probable).

Nous avons donc utilisé cette hauteur maximale des murs comme condition dans notre segmentation. Le mode opératoire est identique à la resegmentation des poutres. Nous avons analysé les points segmentés lors de la première segmentation par rapport à notre condition supplémentaire. Ainsi, seuls les points de coordonnée Z inférieure à la hauteur maximale des murs ont été resegmentés (ajoutés à la liste résultat de cette segmentation). Les points analysés ne respectant pas cette condition ont été rassemblés en une liste regroupée par la suite avec les résultats de la segmentation de « Murs ». Ainsi, les points étaient segmentés soit dans la classe « Portes » ou « Meubles encastrés », soit dans la classe « Murs ».

Contrairement aux poutres, il n'était donc pas nécessaire de conserver les points Shapely pour les points conservés après la 1ère segmentation. En effet, les points étant obligatoirement segmentés dans une classe, ils ne seront pas disponibles pour une autre segmentation. Les points Shapely correspondant aux points non segmentés lors de la première segmentation ont eux été conservés pour les segmentations suivantes. Ici aussi, le tag de classification n'est ajouté qu'après la deuxième segmentation.

Nous avons testé cette nouvelle segmentation sur le nuage « Office_1 » mais le résultat n'a pas été concluant. En effet, la hauteur maximale de « Murs » était identique à la hauteur maximale du pan de mur situé au-dessus de la porte et du meuble. L'essai n'ayant pas été concluant, nous avons vérifié si la méthode fonctionnait réellement en la testant sur un des nuages de validation. De ce fait, même si l'utilisation est minime, nous avons employé un nuage de validation dans notre paramétrage. Cela biaise quelque peu l'utilisation de ce nuage en tant que nuage de validation. Ce problème sera discuté par après, dans la section appropriée (cf. VI.2). Le résultat était concluant pour ce second nuage.

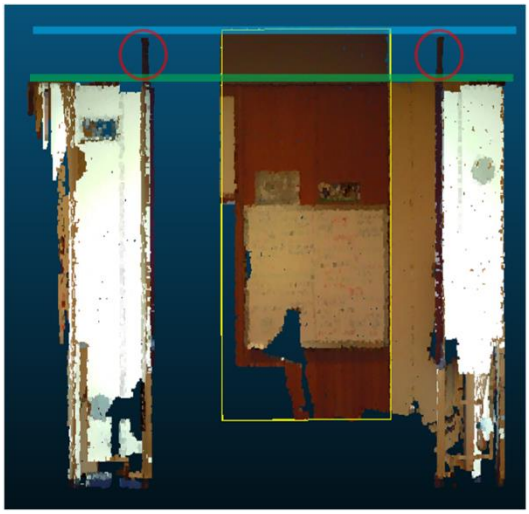


Figure 45 llustration du problème de resegmentation du meuble encastré dans le nuage « Office_1 ».

En vert : la hauteur maximale des murs attendue

En bleu : la hauteur maximale des murs obtenue

Entourés en rouge : les éléments de murs posant problème

Nous avons néanmoins essayé de régler le problème présent dans le nuage « Office_1 » en essayant de déterminer la hauteur de transition de différentes manières. Nous avons notamment tenté de nous appuyer sur les fréquences des hauteurs et sur des statistiques associées (bornes d'une classe de fréquences pic, intervalle de confiance en appliquant une gaussienne aux données des sous-nuages, ...). Ces essais ayant tous été infructueux, nous n'avons pas trouvé de moyen de résoudre ce problème.

Cette resegmentation des portes et meubles encastrés conclut l'affinage de notre méthode de segmentation. Le code obtenu après ces améliorations correspond à la version finale de l'algorithme développé ici (V8). Les résultats obtenus correspondent à des listes (nuages) reprenant chacune les points propres à la classe associée.

V.8. Enregistrement des résultats

Les différents résultats obtenus grâce aux diverses versions de notre code ont été enregistrés sous la forme de listes reprenant les points segmentés pour la classe associée. Ces listes devaient être transformées en fichiers interprétables et visualisables sous la forme de nuages de points. Dans un premier temps, pour le jeu de données de test notamment, ces listes ont été enregistrées sous format « .txt ». Toutefois, ce format n'était pas adapté en ce qui concerne la visualisation des données sur CloudCompare : il ne permettait pas de conserver les couleurs des points et était plus lent à charger qu'un fichier « .las ».

Les nuages en entrée étant au format « .las », nous avons fait en sorte d'enregistrer les résultats sous ce même format. Pour ce faire, nous avons codé une fonction Python créant un fichier « .las » en reprenant le même en-tête que dans le fichier du nuage initial. En créant le fichier de cette manière, la version du format utilisée est identique pour les deux fichiers et ils ont donc les mêmes attributs. Lors de l'extraction des informations depuis le nuage, nous avons repris les informations de chaque point depuis différentes listes. Ces listes, issues de l'importation du nuage, conservaient les attributs de tous les points (une liste *'X'*, une liste *'Y'*, etc.). Pour enregistrer les points, le principe est simplement inversé : on va ajouter à la suite de chaque liste (conçue lors de la création du fichier) la valeur de l'attribut correspondant de notre point : on enregistre l'abscisse du point dans la liste *'X'*. Les informations du point doivent donc être toujours complètes pour que toutes les listes soient complétées et que les écarts soient évités. Les informations ayant toutes été conservées durant la segmentation, l'enregistrement se fait sans problème.

Les listes résultant des différentes segmentations sont donc chacune enregistrées dans un fichier individuel nommé d'après la classe qu'il représente. Les points non-segmentés après la dernière segmentation se voient d'abord attribués un tag de classification correspondant à la classe « Remains » avant d'être enregistrés dans leur fichier. Enfin, les listes résultats sont rassemblées en une seule liste pour enregistrer un nuage total. Dans celui-ci, les points ne sont classifiés qu'au travers de leur attribut de classification.

Pour des facilités de manipulation, toutes les listes mentionnées sont, en fait, converties en matrices Numpy lors de l'étape d'enregistrement, avant de lancer la fonction Python. Cette fonction Python d'enregistrement demande d'ailleurs comme paramètre une matrice Numpy et non une liste. Elle prend aussi comme paramètre une chaine de caractères qui servira de nom au fichier créé.

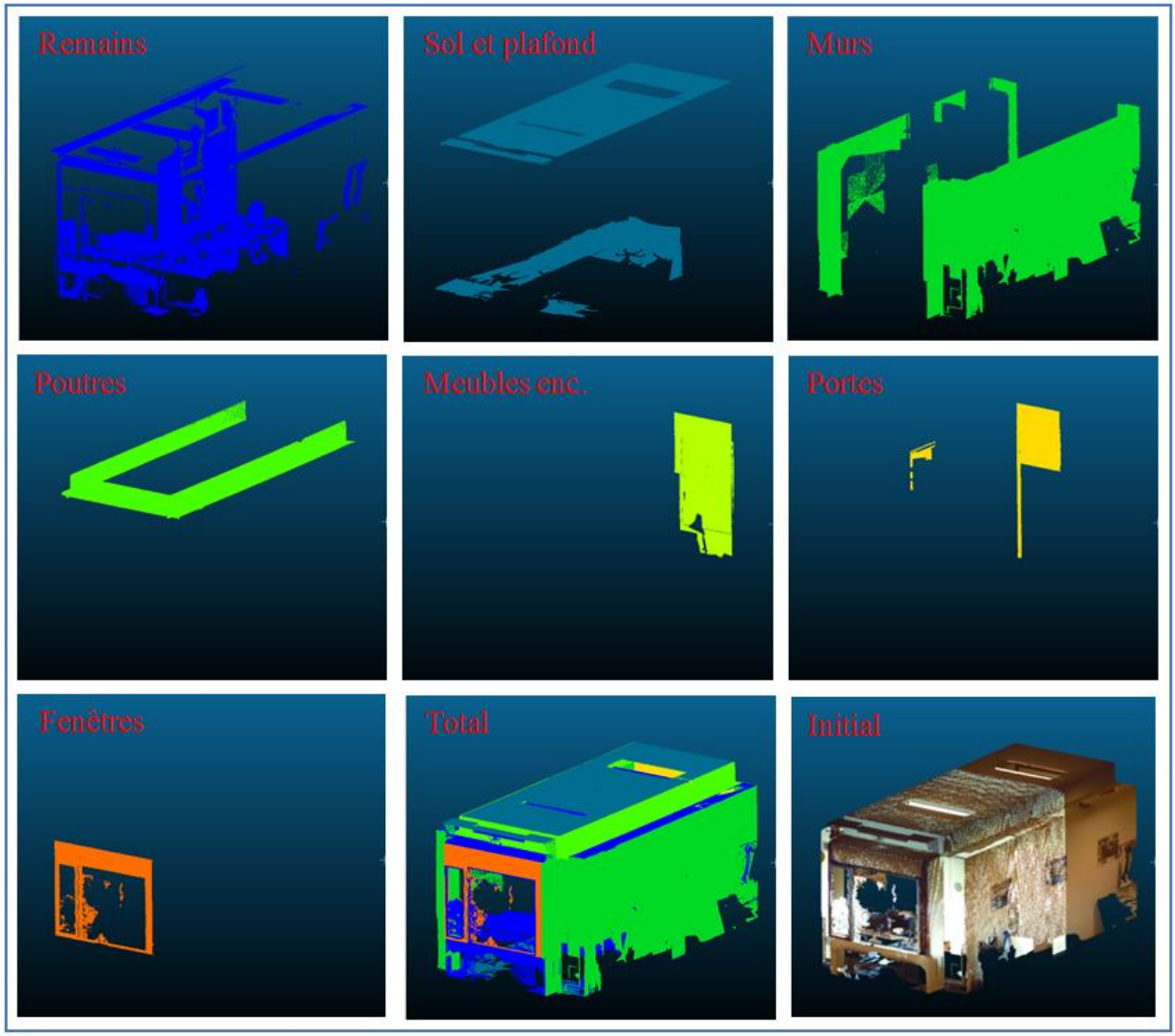


Figure 46 Résultats de la segmentation finale (V8) du nuage "Office_1" (non échantillonné)

V.9. Validations et évaluations statistiques

Enfin, ayant nos résultats, nous avons ensuite validé notre méthode sur les nuages prévus à cet effet. Nous avons ainsi obtenu des résultats pour « Office_2, 3 et 4 » (cf. IX.7). Nous avons dû aussi utiliser différentes métriques pour pouvoir évaluer la justesse de notre méthode, comme expliqué dans la méthodologie.

Nous avons donc créé un autre code afin de calculer ces métriques (cf. IX.2). Pour chaque nuage, nous avons ainsi déterminé : la matrice de confusion comparant notre résultat (sous la forme du nuage total créé en fin d'enregistrement) et la vérité terrain associée (créée lors des prétraitements), l'exactitude globale, les taux d'omission et de confusion, le coefficient Kappa de Cohen général et les Kappa de Cohen par classe.

Pour pouvoir calculer les matrices de confusion, nous devions comparer deux nuages de points identiques, ce qui signifie que nous ne pouvions pas utiliser des résultats échantillonnés. Nous avons donc segmenté entièrement les différents nuages à analyser (environ 13 minutes par nuage). De ce fait, nous avions le même nombre de points dans notre résultat et dans notre vérité terrain. Les points de cette vérité terrain, bien que segmentés en différents nuages selon les classes, ne possédaient pas de tags de classification. Par conséquent, lors de l'importation de ces différents nuages de vérité terrain, nous avons attribué à leurs points le tag de classification approprié. Ils ont ensuite été rassemblés pour former la vérité terrain totale.

Un autre problème à résoudre concernait la différence au niveau de l'ordre d'enregistrement des points dans le résultat et la vérité terrain (due à des traitements différents). A cause de cette différence, la comparaison des deux listes de points dans la matrice de confusion n'était pas possible. Nous avons donc trié les points des deux nuages dans le même ordre, en fonction de leurs coordonnées XYZ. Pour éviter que d'autres informations ne soient utilisées dans le tri, nous avons recréé une matrice Numpy contenant uniquement les coordonnées des points. Nous n'avons pas directement trié ces matrices : nous avons déterminé un index de tri pour chacune d'elles (une matrice par nuage). Cet index est une liste des positions des points dans leur matrice de départ. Ces positions sont ordonnées de manière croissante selon les coordonnées du point associé. Par exemple, la liste de points [c, a, b, d] triés dans l'ordre alphabétique (chez nous l'ordre de coordonnées croissant) donnerait l'index : [1, 2, 0, 3].

Le calcul de la matrice ne nécessitant que les tags de classification, nous avons créé une liste par nuage reprenant les tags de classification des points, ordonnés selon l'index déterminé. Ainsi, nous avons obtenu deux listes de même longueur et triées de la même manière, prêtes à être comparées.

Dès lors, nous avons pu utiliser ces listes pour calculer nos matrices de confusion via une fonction de la librairie scikit-learn (Scikit-learn, 2019). En utilisant les résultats de ces matrices, nous avons pu calculer les autres statistiques mentionnées ci-dessus. Les équations relatives à cette étape sont présentées dans la partie méthodologie (cf. IV.9). Les résultats de cette étape sont affichés et analysés dans la discussion (cf. VI.2).

Matrice de confusion pour le nuage "Office_1"

		Classes du Résultat obtenu							
		Remains	Sol-Plafond	Murs	Poutres	Meubles encastrés	Portes	Fenêtres	TOTAUX
Classes de la Vérité Terrain	Remains	1349654	101819	44090	20875	0	301	3914	1520653
	Sol-Plafond	69973	1348280	37568	44486	763	670	0	1501740
	Murs	44754	43855	1484104	16437	26454	29169	0	1644773
	Poutres	17400	14968	17892	445100	0	3	0	495363
	Meubles encastrés	18	190	2069	40	136975	4	0	139296
	Portes	0	188	1981	29	18103	110718	0	131019
	Fenêtres	97638	54	36181	0	0	0	276179	410052
	TOTAUX	1579437	1509354	1623885	526967	182295	140865	280093	5842896

Tableau 7 Matrice de confusion pour le nuage "Office_1".

Statistiques globales	
Exactitude Globale	88.16%
Cohen's Kappa	84.68%

Office_1		Statistiques par classe		
		Omission	Commission	Kappa de Cohen
Classes	Remains	11.25%	14.55%	84.59%
	Sol-Plafond	10.22%	10.67%	86.22%
	Murs	9.77%	8.61%	86.47%
	Poutres	10.15%	15.54%	88.85%
	Meubles Enc.	1.67%	24.86%	98.28%
	Portes	15.50%	21.40%	84.12%
	Fenêtres	32.65%	1.40%	65.71%

Tableau 8 Résultats statistiques de l'évaluation de notre segmentation du nuage "Office_1"

CHAPITRE VI. VALIDATION ET DISCUSSION

Dans ce chapitre, nous allons analyser et discuter les différents résultats obtenus au cours de notre travail. Nous commencerons par examiner les résultats intermédiaires avant de détailler les résultats finaux et leur validation. Nous aborderons enfin les limitations de notre méthode.

VI.1. Résultats intermédiaires

VI.1.1. **Géoréférencement**

Malgré le référencement que nous avons réalisé, il existe de légers décalages entre le plan et le nuage. Ces écarts peuvent être dus à des erreurs de précision et d'exactitude dans le plan ou le nuage. Un exemple de problème qui sera observé par la suite réside au niveau des portes. Il existe, en effet, un décalage entre le plan et le nuage pour celles-ci, de sorte qu'une partie des points les représentant se situent dans une zone du plan associée à un autre élément (murs, armoires, …) (cf. Fig 44). Les murs ne sont pas non plus tous exactement alignés avec les lignes les représentant dans le plan. Ces petits décalages (de l'ordre de quelques millimètres) sont pris en compte dans le calcul des zones tampons. Notons aussi que le géoréférencement ne s'est fait que via deux points : le manque de redondance n'a donc pas permis d'ajuster le résultat par moindres carrés. Enfin, la correction d'échelle de projection du système du nuage a été réalisée sur base des informations fournies par le logiciel, et vérifiée en comparant des coordonnées réelles et dans le plan, pour plusieurs points. Toutefois, il n'est pas exclu que cette correction n'ait pas été totale, laissant donc des erreurs résiduelles.

VI.1.2. **Prétraitements**

Le choix de la classification est subjectif mais a été basé et justifié par une analyse construite sur différentes sources. Ce choix était, comme spécifié précédemment, limité par les calques trouvés dans le plan. La création des vérités terrains a, quant à elle, été largement facilitée par l'utilisation des données de M. Kharroubi. Toutefois, la segmentation et classification manuelle étant une opération subjective, la modification de ces données était nécessaire pour que les résultats correspondent à notre vision. Cette subjectivité signifie aussi que ces vérités terrains ne constituent pas des vérités absolues, que ce soit au niveau des découpages spatiaux ou des classifications (le choix des classes est aussi subjectif). Nous comparons donc nos résultats finaux non pas à des résultats absolus, mais à une vérité relative issue d'une autre méthode et admise comme juste.

Au sujet de la mise en forme du plan, son implémentation constitue une limite relativement importante de notre méthode et sera discutée ci-dessous (cf. VI.3). Pour ce qui est des résultats, nous pouvons remarquer que les éléments tracés ne distingueront pas le bruit ou les occlusions, la zone tampon associée ne constituant qu'une délimitation spatiale. Les formes tracées pour les éléments mobiles (portes, armoires, fenêtres) pourront ne pas correspondre au nuage comme nous le verrons par la suite. En effet, la difficulté, ici, est de tracer une forme assez précise pour fournir un bon résultat mais aussi assez générale pour prendre en compte les différents types d'objets.

VI.1.3. **Extraction d'informations**

Cette étape ne nécessite pas vraiment d'analyse : les informations sont extraites et structurées de la manière la plus efficace et la plus claire possible afin de faciliter la suite des opérations.

VI.1.4. **Segmentation par cadre capable**

L'utilisation d'un cadre capable présente différents avantages : elle est relativement aisée à mettre en œuvre, le test est simple et elle fournit des premiers résultats relativement exacts dans la plupart des cas. En effet, lors de la segmentation du segment de l'élément mur (cf. V.4.1.1), l'entièreté du segment est récupérée. Toutefois, le résultat n'est pas suffisamment précis : après la généralisation, on peut voir que certaines parties (relativement importantes) du sol ou d'autres éléments sont récupérées avec les segments de murs. De plus, cette généralisation complexifie et alourdit nettement les traitements, chaque segment devant être traité individuellement. Un autre défaut à prendre en compte est que, pour des segments parallèles à l'axe X ou Y, ce cadre capable sera réduit au segment lui-même. Cela engendre une condition pour le test des points bien trop discriminante. A l'inverse, plus le segment tend vers un angle de 45° par rapport à ces axes, moins la condition sera discriminante. Ainsi, selon l'orientation du segment par rapport aux axes X et Y, le cadre de segmentation sera plus ou moins large et la précision de segmentation plus ou moins réduite (cf. Fig. 47).

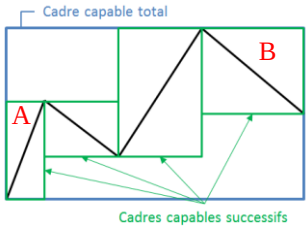


Figure 47 Le cadre capable de A est plus discriminant que celui de B

L'exactitude, elle, ne variera pas avec la taille du cadre mais sera hautement dépendante des informations extraites du plan : les coordonnées issues du plan doivent concorder à celles du nuage. Ainsi, les problèmes de segmentation du second résultat de cette méthode (cf. Fig. 31) sont notamment dus à la précision variable, mais aussi à des erreurs de représentation du mur sur le plan de ces données de test (mal dessiné).

VI.1.5. Segmentation par zones tampons orientées

En ce qui concerne l'implémentation de l'utilisation de zones tampons pour la segmentation du nuage, le principe du test des points reste relativement simple notamment grâce aux fonctions explicites de la librairie Shapely. De plus, le fait de prendre les éléments du plan entièrement et non segment par segment constitue une amélioration certaine en termes de rapidité et de simplicité. Cependant, il faut rappeler que l'utilisation de la librairie Shapely a impliqué la mise en forme du plan. En outre, l'analyse des noms de calques (généralisée par la suite) permettra de ne considérer que les éléments pertinents du plan, et de rendre ainsi la méthode plus efficace.

En termes de précision et d'exactitude, les zones tampons définies ici sont plus précises que les cadres capables utilisés précédemment. En effet, la zone se base sur la forme de l'élément du plan plutôt que de définir un simple cadre autour. Ainsi, le contour de l'élément est suivi et on ne reprend que les points situés à une certaine distance (seuil unique) de celui-ci. En utilisant la même valeur seuil pour toutes les zones tampons d'un calque, on s'assure une précision identique pour tous ses éléments. En outre, cette valeur étant modulable, elle peut s'adapter aux besoins des différentes classes. Si le calcul de la zone est plus compliqué ici, la méthode fournie par Shapely nous simplifie nettement la tâche. L'exactitude de cette deuxième technique dépend ici aussi des informations extraites du plan. Avec les cadres capables, ces erreurs pouvaient être contrebalancées de manière brute et quelque peu aléatoire grâce à une extension suffisante du cadre. Ici, le tampon de l'élément permet de prendre en compte les décalages plan-nuage mais de manière plus fine, contrôlée et précise. L'exactitude, si les informations sont bonnes, est donc relativement identique mais sera, ici, mieux contrôlée en cas de problèmes (cf. Fig. 48).

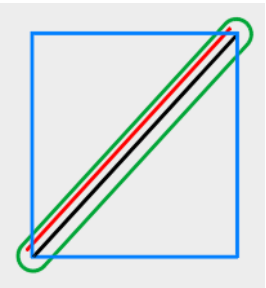


Figure 48 Différences de la prise en compte d'erreurs. (Rouge : erreur ; noir : élément bon ; vert : zone tampon ; bleu : cadre capable)

VI.1.6. Optimisation des temps de traitement

VI.1.6.1. Indexation & Echantillonnage

L'indexation n'a pas pu être mise en œuvre malgré différents essais (cf. V.5.1) : l'algorithme n'en a donc pas profité. Il s'agit d'une limitation qui sera discutée ci-dessous.

Différents degrés d'échantillonnages ont été testés et utilisés par la suite. Le problème lié à l'utilisation d'un échantillonnage est qu'on perd de l'information. Le but est donc de trouver un équilibre entre un gain de temps significatif et une perte de points minimale. Pour faire le lien avec l'état de l'art, on constate ici que le volume de données et le temps de traitement sont de réels problèmes dans le traitement des nuages. Dans notre cas, l'échantillonnage était utilisé afin de gagner du temps lors du paramétrage de l'algorithme. Ainsi, les essais de paramétrage du code étaient d'abord réalisés avec un échantillonnage à 5% (un point sur 20) permettant d'obtenir les résultats plus rapidement. Une fois le code paramétré, les autres degrés d'échantillonnage étaient alors testés afin d'évaluer les équilibres gain de temps/perte d'informations. Nous avons constaté qu'au-delà de 10% d'échantillonnage, la perte d'informations devenait trop importante (visualisation des résultats réduite). Les résultats liés au gain de temps sont discutés dans le point suivant.

VI.1.6.2. Optimisation de l'algorithme et chronométrage

Temps de traitement des différentes versions de la méthode de segmentation par zone tampon, appliquée au nuage Office_1

Temps de traitement en secondes (s)		Degrés d'échantillonnage						
		1	1/5	1/8	1/10	1/12	1/15	1/20
VERSIONS	V1	215	46	30	22	19	16	12
	V2	734	140	89	60	52	42	31
	V3	349	68	42	35	29	23	18
	V4	273	52	33	26	22	18	14
	V5	Pas de modification par rapport à V4						
	V6	/	103	63	51	42	34	25
	V7	/	108	66	48	40	32	24
	V8 - finale	/	141	87	68	60	49	36

Augmentations relatives des temps de traitement d'une version de la méthode de segmentation par zone tampon à la suivante (pour le nuage Office_1)

Augmentations relatives du temps de traitement (%)		Degrés d'échantillonnage						
		1	1/5	1/8	1/10	1/12	1/15	1/20
VERSIONS	V1-V2	241.3953	204.3	196.7	172.73	173.68	162.5	158.33
	V2-V3	-52.4523	-51.43	-52.81	-41.67	-44.23	-45.24	-41.94
	V3-V4	-21.7765	-23.53	-21.43	-25.71	-24.14	-21.74	-22.22
	V4-V6	/	98.08	90.91	96.154	90.909	88.89	78.571
	V6-V7	/	4.854	4.762	-5.882	-4.762	-5.882	-4
	V7-V8	/	30.56	31.82	41.667	50	53.13	50

Tableau 9 Analyse de l'évolution des temps de traitement au cours des versions du logiciel et selon l'échantillonnage (cf. Tab. 4 et 5)

De manière globale, les différentes améliorations de cette phase d'optimisation (V2 à V5) ont relativement bien fonctionné. Tout d'abord, l'augmentation relative importante du temps de traitement entre la version 1 et 2 (+180%) s'explique simplement par la généralisation qui a eu lieu en V2. On passe de la segmentation d'une entité, à la segmentation de toutes les entités du calque. Dès les premières réductions (V3 – réduction liée à la création des points Shapely), on réduit les temps de traitement de moitié (Tableau de droite, V2-V3), soit une amélioration significative. La phase suivante (Tableau de droite, V3-V4) correspond à l'unification des tests pour chaque zone tampon en un test général unique. Il permet de réduire de 20 à 25% les temps de traitement. Ces deux étapes ont donc été cruciales pour la rapidité et l'efficacité de l'algorithme.

CODE COULEURS
Tests avec une entité
Tests avec toutes les entités d'un calque
Tests avec toutes les entités de tous les calques

Tableau 10 Code couleurs associé au tableau des temps de traitement

En effet, la version suivante va doubler les temps de traitement précédents. Cette version correspond à la généralisation des segmentations aux différentes classes (V4-V6) : on passe d'une segmentation à cinq. On peut remarquer que cette augmentation est proportionnellement moindre que celle de la généralisation précédente (V1-V2). De plus, bien qu'on multiplie par cinq le nombre de segmentations, le temps lui n'est que doublé. Il y a deux facteurs qui interviennent ici. Tout d'abord, certains calques contiennent plus d'éléments que d'autres, ce qui signifie plus de zones tampons à tester (rappelons que, dans la segmentation, c'est bien le test des points qui est chronophage). Le deuxième facteur correspond à la gestion de nos points segmentés. En effet, au fur et à mesure de nos segmentations, les points sont retirés du nuage. Le nombre de points à tester est donc réduit d'une segmentation à la suivante, ce qui accélère ces opérations. Ce dernier facteur explique aussi pourquoi l'introduction de la segmentation du sol et du plafond (V7) n'engendre en moyenne peu ou pas de variations du temps de traitement (Tableau de droite, V6-V7). L'affinage des segmentations (Tableau de droite, V7-V8) engendre une augmentation de 40% en moyenne : l'amélioration de la précision et de l'exactitude des résultats se fait au détriment de notre temps de traitement. Remarquons que, grâce aux améliorations réalisées au niveau du temps, nous réalisons la segmentation totale et affinée du nuage en un temps identique à la segmentation d'un calque avant ces améliorations.

En ce qui concerne les degrés d'échantillonnages, on voit logiquement une corrélation linéaire avec le temps de traitement : multiplier le degré d'échantillonnage par un facteur divise le temps de traitement par ce même facteur. Cette corrélation devrait logiquement s'estomper lorsque la part de temps relative aux étapes dépendantes des points se rapproche de la part de temps relative aux étapes qui en sont indépendantes (ici, celles relatives au plan). Or, ces dernières ne représentent ici qu'une part très mince du temps de traitement. Au vu des considérations sur la perte de données et sur le gain de temps, un échantillonnage entre 10 et 20% (1/10 et 1/5) semble être un bon compromis. Toutefois, pour des résultats totaux, il est préférable de ne pas échantillonner le nuage (pas de perte). Nous avons dû, dans notre validation, calculer de tels résultats : le temps nécessaire était d'environ 800 secondes.

Ce temps de traitement reste assez bon par rapport au temps d'une segmentation manuelle d'un nuage entier qui est de plusieurs heures. Bien sûr, il ne faut pas oublier de prendre en compte le temps nécessaire aux prétraitements. Malgré ce temps supplémentaire, si on compare notre méthode à la méthode manuelle, toutes proportions gardées, notre méthode permettra de segmenter un nuage de points plus rapidement.

VI.1.7. Généralisation et amélioration des segmentations

Reprenons et analysons l'affinage des segmentations des différentes classes, pour le nuage « Office_1 » (nuage de paramétrage). Les résultats visuels sont disponibles dans les résultats ainsi qu'en annexes (cf. Fig. 46 et 68). Commençons par la classe « Sol-plafond »

Pour cette classe, il n'était pas possible de se baser sur le plan. Toutefois, la méthode statistique semble être relativement efficace avec un indice Kappa de 86% (concordance forte entre les classes des deux méthodes (cf. Fig. 10)). Les taux de commission et d'omission sont quasi identiques. Il y a 10% de points qui devraient être attribués à cette classe et qui ne le sont pas (omission) et 10% de points qui y sont classés alors qu'ils ne devraient pas (commission). Dans les deux cas, les classes principalement concernées sont « Remains », « Murs » et « Poutres » qui correspondent aux différentes interfaces avec le sol ou le plafond. Il s'agit, de manière générale, d'une différence entre les deux méthodes dans la gestion de l'interface avec ces éléments. Cela découle notamment d'un ordre de segmentation différent et de différences dans les hauteurs de segmentation du sol et plafond. Par exemple, « Poutres » est segmenté, ici, en premier lieu alors que dans la vérité terrain, c'est « Sol-Plafond ». Nous reprenons donc des points du plafond dans notre classe « Poutres » (omission pour notre classe « Sol-Plafond »). Cependant, la vérité « Sol-plafond » va aussi reprendre des points de « Poutres ».

Matrice de confusion pour le nuage "Office_1"

Classes de la Vérité Terrain	Classes du Résultat obtenu							
	Remains	Sol-Plafond	Murs	Poutres	Meubles encastrés	Portes	Fenêtres	TOTAUX
Remains	1349654	101819	44090	20875	0	301	3914	1520653
Sol-Plafond	69973	1348280	37568	44486	763	670	0	1501740
Murs	44754	43855	1484104	16437	26454	29169	0	1644773
Poutres	17400	14968	17892	445100	0	3	0	495363
Meubles encastrés	18	190	2069	40	136975	4	0	139296
Portes	0	188	1981	29	18103	110718	0	131019
Fenêtres	97638	54	36181	0	0	0	276179	410052
TOTAUX	1579437	1509354	1623885	526967	182295	140865	280093	5842896

Statistiques globales	
Exactitude Globale	88.16%
Cohen's Kappa	84.68%

Office_1		Statistiques par classe		
		Omission	Commission	Kappa de Cohen
Classes	Remains	11.25%	14.55%	84.59%
	Sol-Plafond	10.22%	10.67%	86.22%
	Murs	9.77%	8.61%	86.47%
	Poutres	10.15%	15.54%	88.85%
	Meubles Enc.	1.67%	24.86%	98.28%
	Portes	15.50%	21.40%	84.12%
	Fenêtres	32.65%	1.40%	65.71%

Tableau 11 Résultats de l'évaluation statistique de "Office_1"

Il est dès lors compliqué d'analyser les matrices de confusion puisque nos vérités terrains peuvent aussi contenir des erreurs influençant ces statistiques : certains points peuvent être mieux classés dans notre méthode (Ex. : Poutres). Il s'agit d'une limitation de la validation.

Pour « Murs », l'utilisation de polylignes et d'un seuil de 0,025m semble être efficace : le Kappa indique une concordance forte (86.5%). Cette classe présente des omissions avec toutes les autres classes si ce n'est « Fenêtres ». En effet, les murs représentent l'interface liant entre eux les différents éléments structurants de la scène. L'interface avec le sol et le plafond a déjà été évoquée. Le problème est quasi le même pour les poutres : elles sont segmentées avant et reprennent un peu du mur là où ils se rencontrent. Pour les autres classes, les omissions et commissions viennent de la gestion de la zone tampon. En effet, plus la valeur seuil du tampon est importante, plus on sera sûr de reprendre tous les points de la classe de vérité terrain « Murs » (l'omission diminue) mais plus on reprendra des points provenant d'autres classes (la commission augmente). Ce constat s'inverse lorsqu'on diminue la valeur seuil. Il faut donc trouver un équilibre entre les deux, ce qui est plus ou moins bien réalisé ici puisque les taux d'omission et de commission sont assez proches. Il faut noter qu'une zone tampon s'étend dans toutes les directions, et donc au-delà de l'extension en longueur de l'élément aussi. Cela explique certains problèmes de commission. Une solution pourrait être de modifier la zone tampon pour qu'elle s'arrête à cette extension.

Pour les poutres, l'indice de Kappa atteint presque 90% ce qui prouve que la segmentation est efficace, notamment grâce à l'affinage. Les classes intervenant dans l'omission et la commission sont celles citées et analysées précédemment : « Remains », « Murs » et « Sol-Plafond ». Si les taux d'omission/commission sont proportionnellement plus importants ici, c'est que la somme totale de points dans la classe « Poutres » (utilisée comme dénominateur dans le calcul de ces taux) est inférieure à celle de « Murs » ou « Sol-Plafond ».

Supposant que les hauteurs des poutres sont identiques, il a été envisagé de réaliser en quelque sorte une segmentation d'un deuxième niveau de plafond à cette hauteur (extraction de la zone entre la base inférieure des poutres et le plafond). Toutefois, cette méthode a été jugée trop spécifique à notre cas d'étude (overfitting) et n'a pas été implémentée. Cette méthode aurait notamment permis de segmenter le pan de mur situé au-dessus des portes et meubles encastrés, ainsi que de récupérer certains éléments considérés comme restes : lampes, détecteur de fumée, etc. Notons aussi que l'hypothèse d'uniformité de hauteur des poutres est aussi appliquée dans notre méthode, et bien assimilée dans notre cas au vu de nos résultats. En outre, pour ce qui est de la détermination de hauteurs seuils ici pour « Poutres » et dans la classe « Sol-Plafond », l'utilisation d'une distribution normale pour décrire les données de la classe de pic fonctionne bien. Néanmoins, cette technique pourrait être améliorée en utilisant une distribution plus adaptée aux données.

Pour les portes et meubles encastrés, la méthode est bien, voire très bien adaptée : 98% et 84% de concordance selon l'indice Kappa (respectivement). Sans trop de surprises, la classe avec laquelle ces deux classes présentent un fort taux de commission est la classe « Murs ». Comme expliqué dans les résultats, l'affinage de la segmentation n'a pas fonctionné ici : le pan de mur situé au-dessus de la porte et du meuble n'a pas été resegmenté. Ainsi, un certain nombre de points normalement attribués à « Murs » se retrouvent dans « Portes » ou « Meubles encastrés », expliquant le taux de commission important. Dans les taux d'omission, nous retrouvons des points « Murs » principalement, ce qui s'explique par l'extension de la zone tampon de ceux-ci, mentionnée précédemment. Nous remarquons aussi un fort taux d'omission de « Portes » pour « Meubles encastrés ». Il s'agit ici de l'erreur du plan engendrant un décalage entre l'élément « Portes » du plan et sa représentation dans le nuage : une partie de la porte est comprise dans la zone tampon du meuble encastré adjacent (cf Figure 46). Si notre méthode de segmentation semble donner de bons résultats, il faut toutefois les relativiser en rappelant que l'affinage de ces segmentations est ajusté à notre cas et ne fonctionnerait peut-être pas aussi bien en d'autres circonstances (Overfitting).

Nous pouvons, ici, faire la distinction entre des éléments « fixes » tels que les murs, plafond, poutres ou « modulables » tels que les portes, fenêtres et meubles. En effet, ces derniers sont plus compliqués à segmenter de par leur configuration variable : plus ou moins ouverts, plus ou moins profonds, vitrés ou non, encastrés totalement ou dépassant, montant jusqu'au plafond ou avec un pan de mur au-dessus, etc. Il est donc difficile de déterminer une méthode de segmentation reprenant tous ces états. Il est par contre plus envisageable de fixer des règles de levé : fenêtres et meubles toujours fermés, portes ouvertes à 90°, etc. Nous avons notamment le cas, dans notre nuage total initial, de portes ouvertes au maximum et qui ne sont, dès lors, pas reprises dans la zone de l'élément du plan correspondant.

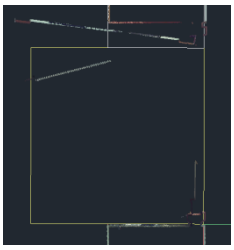


Figure 49 Problème de porte trop ouverte (hors de la zone jaune)

Enfin, la classe « Fenêtres » ne présente que 65% de concordance (Modéré, cf. Fig. 10) et un fort taux d'omission lié à « Remains » et « Murs ». Cela peut s'expliquer simplement par le fait qu'une partie du chambranle de la fenêtre n'est pas segmenté (cf. Fig. 43). En effet, cette partie de points non segmentés sépare partiellement la fenêtre des murs. Cette zone explique aussi l'absence d'omission de « Murs » vers « Fenêtres ». Il apparait donc que dans ce cas-ci, notre segmentation a été trop discriminante et mériterait certaines améliorations.

« Remains » ne correspond pas à une classe d'objets bien définis. Elle ne sera donc pas analysée ici, si ce n'est via les autres classes. Pour conclure avec cette analyse des résultats finaux, nous pouvons observer que notre méthode a permis d'obtenir 88% de points bien classés et un Kappa général de 85%. Bien que ces résultats ne soient pas parfaits, ils restent forts selon la grille utilisée par McHugh (2012) (cf. Fig. 10). Cela est toutefois à mettre en perspective avec le fait que notre vérité terrain ne constitue pas une référence absolue.

VI.2. Validation et statistiques

Nous avons validé l'algorithme en l'appliquant aux nuages prévus à cet effet et nous allons analyser les résultats obtenus. Rappelons que cette validation présente deux limites : l'hypothèse que nos vérités terrains sont des vérités absolues et le fait que les données de validation proviennent de la même source que les données de paramétrage. L'utilisation du nuage « Office_3 » dans le cadre du paramétrage ne sera pas considérée comme une limite pour la validation. En effet, il a uniquement servi de validation visuelle du test d'affinage des segmentations de « Poutres » et « Meubles encastrés ». Aucune donnée de ce nuage n'a été extraite ni utilisée pour paramétrer l'algorithme. Par cette implication minimale, nous jugeons que ce nuage peut donc être employé dans la validation.

VI.2.1. « Office_2 »

Matrice de confusion pour le nuage "Office_2"

		Classes du Résultat obtenu							
		Remains	Sol-Plafond	Murs	Poutres	Meubles encastrés	Portes	Fenêtres	TOTAUX
Classes de la Vérité Terrain	Remains	982280	246225	493915	24415	195	0	0	1747030
	Sol-Plafond	513260	1682453	3153	57622	151	28	0	2256667
	Murs	188402	326388	1360624	43337	0	0	0	1918751
	Poutres	39414	38188	748	97838	0	0	0	176188
	Meubles encastrés	0	855	6992	7	80743	0	0	88597
	Portes	0	506	0	0	16848	51008	0	68362
	Fenêtres	8570	17329	14093	0	0	0	172584	212576
	TOTAUX	1731926	2311944	1879525	223219	97937	51036	172584	6468171

Statistiques globales	
Exactitude Globale	68.45%
Cohen's Kappa	55.86%

Office_2		Statistiques par classe		
		Omission	Commission	Kappa de Cohen
Classes	Remains	43.77%	43.28%	40.22%
	Sol-Plafond	25.45%	27.23%	60.40%
	Murs	29.09%	27.61%	59.00%
	Poutres	44.47%	56.17%	53.94%
	Meubles Enc.	8.87%	17.56%	91.00%
	Portes	25.39%	0.06%	74.41%
	Fenêtres	18.81%	0.00%	80.67%

Tableau 12 Résultat de l'évaluation statistique de "Office_2"

Pour ce nuage « Office_2 », il apparait que les résultats sont moins bons qu'« Office_1 » : l'exactitude globale n'est que de 68,5% et la concordance de 56% (faible, cf. Fig. 10).

En observant les classes « Portes » et « Meubles encastrés », nous remarquons qu'elles présentent une configuration similaire à celle du premier nuage. Nous retrouvons le problème associé à la représentation des portes

dans le plan, engendrant la classification de points « Portes » dans « Meubles encastrés » (omission pour « Portes »). On retrouve aussi le problème entre « Murs » et « Meubles encastrés » dû à la segmentation de points du second dans le résultat du premier (omission pour les meubles encastrés). La classe « Fenêtres », quant à elle, est relativement bien segmentée dans ce cas-ci, avec une concordance de l'ordre de 81%. Toutefois, on constate qu'il n'y a aucune commission mais bien de l'omission : notre méthode est donc trop discriminante. Ces points non classés dans « Fenêtres » sont principalement ajoutés soit à « Murs », soit à « Sol-Plafond » et augmentent donc les commissions de ces classes. Il s'agit ici aussi d'un problème lié à la non-segmentation d'une partie du chambranle de la fenêtre (extension du tampon insuffisante dans le sens de la longueur de la pièce).

Les autres segmentations sont assez problématiques et engendrent des statistiques plus faibles. Une grande partie de ces problèmes s'explique par la présence d'un faux plafond légèrement incliné (cf. Fig. 75) qui engendre un certain nombre d'erreurs. Une partie du plafond, trop basse par rapport à la hauteur seuil déterminée, ne sera pas segmentée (associée à « Remains »). A cela s'ajoutent des erreurs déjà présentes dans « Office_1 ». Les problèmes entre « Sol-Plafond » et « Murs » consistent surtout en une commission importante due à la segmentation dans « Sol-Plafond » de nombreux points « Murs ». Cela est sûrement lié à l'inclinaison du plafond engendrant une mauvaise détermination de hauteur seuil pour la segmentation. De nombreux points de « Murs » sont enregistrés dans « Remains » et inversément. Cela s'explique certainement par une gestion différente des éléments situés à l'interface entre ces classes : étagères, posters, armoires, … Enfin, les poutres se trouvent à même hauteur que le faux plafond et sont donc coplanaires à celui-ci. C'est pour ce cas de figure que la segmentation des poutres est réalisée en premier lieu : cela évite qu'elles soient segmentées avec le plafond. Toutefois, la segmentation n'a pas été assez discriminante (trop large) : des points du plafond et des murs ont été segmentés avec les poutres. Cela, ajouté au nombre relativement faible de points segmentés, expliquent le taux de commission important (plus de 50%) de « Poutres ». L'omission importante, elle, est principalement due à la classe « Remains » et donc à l'inclinaison du faux plafond.

Ainsi on voit que, pour une configuration différente (faux plafond) et un défaut d'inclinaison, les résultats sont affaiblis. On retrouve aussi des problèmes identiques à ceux de « Office_1 ». Notons que la segmentation du pan de mur surplombant la porte et le meuble n'a pas été nécessaire ici, la hauteur de ces éléments correspondant à celle du faux plafond.

VI.2.2. « Office_3 »

Matrice de confusion pour le nuage "Office_3"

Classes de la Vérité Terrain	Classes du Résultat obtenu: Remains	Sol-Plafond	Murs	Poutres	Meubles encastrés	Portes	Fenêtres	TOTAUX
Remains	1571784	57663	218491	23116	0	3207	0	1874261
Sol-Plafond	10196	1724218	1315	36982	2	758	0	1773471
Murs	55274	16589	1238405	5799	1149	6990	0	1324206
Poutres	5948	2449	2111	564884	0	1024	0	576416
Meubles encastrés	30684	475	4392	168	176321	0	0	212040
Portes	46601	9440	52838	950	25481	488459	0	623769
Fenêtres	13520	0	7091	0	0	0	193200	213811
TOTAUX	1734007	1810834	1524643	631899	202953	500438	193200	6597974

Statistiques globales	
Exactitude Globale	90.29%
Cohen's Kappa	87.67%

Office_3		Statistiques par classe: Omission	Commission	Kappa de Cohen
Classes	Remains	16.14%	9.36%	78.11%
	Sol-Plafond	2.78%	4.78%	96.17%
	Murs	6.48%	18.77%	91.57%
	Poutres	2.00%	10.61%	97.79%
	Meubles Enc.	16.85%	13.12%	82.62%
	Portes	21.69%	2.39%	76.53%
	Fenêtres	9.64%	0.00%	90.07%

Tableau 13 Résultat de l'évaluation statistique de "Office_3"

Pour ce 3ème nuage, nous avons obtenu des résultats légèrement meilleurs que pour « Office_1 ». L'exactitude globale est de 90% et le Kappa global de 88%, ce qui correspond à une concordance forte entre les résultats des deux méthodes.

L'indice Kappa de la classe « Sol-Plafond » est très élevé (96%) ce qui prouve une bonne concordance pour cette classe. Les taux d'omission et de commission sont, eux, très bas. Les classes liées à ces taux restent les mêmes que précédemment. Quelques points de « Remains » et de « Murs » sont incorrectement repris dans cette classe (parce que la segmentation de « Sol-Plafond » a lieu en 1er dans la référence mais en 2nd dans notre méthode). Les points omis sont, eux, classés en « Poutres » (segmentées en 1er lieu) et un peu en « Remains ». L'explication reste la même : la gestion des interfaces entre classes diffère un peu selon la méthode.

« Murs » est peu influencée par « Poutres » mais bien par « Remains ». En effet, ce bureau comprend des éléments muraux qu'il est difficile de distinguer du mur avec notre technique. Par exemple, les 3 cadres situés sur un des murs en longueur ne seront pas repris en tant que reste (cf. Fig. 64 et 71). Une partie des deux portes supplémentaires du nuage est reprise dans « Murs » à cause de l'extension latérale des zones tampons des murs, ainsi que des décalages existant pour les portes entre plan et nuage. Cela explique la commission importante de « Murs » pour « Portes » (et l'omission inverse). L'omission des points « Murs » au profit de « Portes » et « Meubles encastrés » provient certainement de la segmentation du pan de mur.

Les segmentations des poutres présentent ici une concordance très élevée : 98%. Les omissions et commissions de « Remains » par rapport aux poutres proviennent certainement de la gestion légèrement différente de la séparation entre les poutres et les conduits.

On retrouve ici aussi le problème des points repris dans « Meubles encastrés » alors qu'ils appartiennent à « Portes » (omission de « Portes ») et qui est dû au décalage entre le plan et le nuage. Le même problème se présente entre les portes et les murs, comme précisé ci-dessus. La segmentation du mur situé au-dessus de la porte et du meuble s'est, dans ce cas-ci, très bien déroulée. L'embrasure des portes n'a pas été totalement segmentée et on en retrouve des points dans « Remains » (omission de « Portes » vers cette classe) (cf. Fig. 71, Remains). Enfin, on note ici un exemple de problème associé à la modularité des meubles : la porte supérieure de l'armoire étant ouverte, elle n'est pas reprise dans la zone tampon de cet élément et n'est donc pas segmentée (attribuée à « Remains ») (cf. Fig. 72). Ces différentes petites erreurs et le plus faible nombre de points que ces classes contiennent expliquent que leur Kappa soit légèrement inférieur à celui des classes précédentes.

Concernant la fenêtre, le constat est le même qu'à la validation précédente : l'indice Kappa est élevé (90%) et il n'y a aucune commission. Cela suggère que notre méthode a été trop discriminante pour la fenêtre. Les omissions présentes sont liées aux murs et au reste.

Globalement, cette seconde validation nous offre de bons résultats. Toutefois, la configuration de la scène est proche de celle de « Office_1 » et éprouve donc peu la méthode.

VI.2.3. « Office_4 »

Matrice de confusion pour le nuage "Office_4"

Classes de la Vérité Terrain \ Classes du Résultat obtenu	Remains	Sol-Plafond	Murs	Poutres	Meubles encastrés	Portes	Fenêtres	TOTAUX
Remains	2047246	40491	190037	26450	0	0	0	2304224
Sol-Plafond	5273	1985931	1443	37882	31	70	0	2030630
Murs	297350	12132	1099222	4762	1707	1219	0	1416392
Poutres	4779	717	830	617512	0	0	0	623838
Meubles encastrés	35570	29	2410	117	154908	0	0	193034
Portes	20	10	6297	245	22551	160018	0	189141
Fenêtres	13559	0	11742	0	0	0	189351	214652
TOTAUX	2403797	2039310	1311981	686968	179197	161307	189351	6971911

Statistiques globales	
Exactitude Globale	89.71%
Cohen's Kappa	86.30%

Office_4		Statistiques par classe		
		Omission	Commission	Kappa de Cohen
Classes	Remains	11.15%	14.83%	82.98%
	Sol-Plafond	2.20%	2.62%	96.89%
	Murs	22.39%	16.22%	72.42%
	Poutres	1.01%	10.11%	98.88%
	Meubles Enc.	19.75%	13.55%	79.73%
	Portes	15.40%	0.80%	84.24%
	Fenêtres	11.79%	0.00%	87.88%

Tableau 14 Résultat de l'évaluation statistique de "Office_4"

Les résultats statistiques de cette dernière validation sont assez proches de ceux de la précédente. L'exactitude et le Kappa globaux sont élevés : respectivement 90 et 86%.

La classe « Sol-Plafond » atteint ici aussi un Kappa très élevé de 97%. On remarque que les taux d'omission et de commission pour cette classe sont très bas. Les points omis ou commis s'expliquent comme auparavant par l'ordre des segmentations

(« Poutres ») et l'extension de la zone tampon reprenant des points hors de la classe. Pour ce qui est de la classe « Poutres », on peut observer un Kappa quasiment parfait (99%). La plupart des omissions et commissions ne sont pas significatives. Si on s'intéresse toutefois à la commission de « Poutres » par rapport à « Remains », celle-ci peut s'expliquer par la gestion de la séparation entre les poutres et les conduits (non-segmentés). Les commissions de « Poutres » par rapport à « Sol-Plafond » et « Murs » sont dues à l'ordre de segmentation.

La classe « Murs » est intéressante dans cette validation : on retrouve dans ce bureau plusieurs tableaux et affiches qui seront difficilement pris en compte par notre méthode alors qu'ils le sont dans la vérité terrain. Ces différents éléments situés à l'interface entre le mur et la partie non-classée de ce nuage expliquent la commission importante associée à cette classe. L'omission importante de « Murs » au profit de « Remains » peut s'expliquer par une partie de pan de mur qui n'a pas été correctement segmentée (cf. Fig. 65 et 73). La cause de ce problème de segmentation est une inclinaison de la représentation du mur dans le nuage par rapport à sa représentation dans le plan. Cette erreur peut résulter de plusieurs causes : le mur n'est réellement pas droit, il s'agit d'un problème de géoréférencement, etc. Les autres omissions et commissions associées à cette classe sont peu significatives et correspondent à des causes déjà détaillées auparavant (extension latérale de la zone tampon).

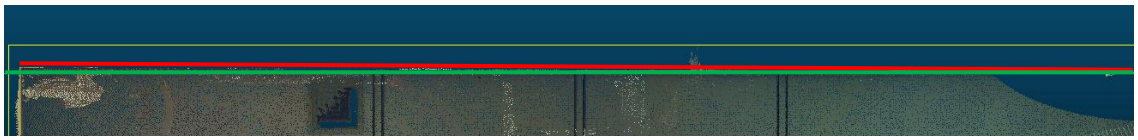


Figure 50 Problème d'inclinaison d'un mur d'Office_4. La ligne verte est une ligne parallèle à l'axe X et provenant du second coin du mur. La ligne rouge représente le plan vertical du mur. (Vue du haut)

La porte est ici particulièrement bien segmentée mais présente toujours le même problème avec « Meubles encastrés » (décalage entre plan et nuage). Le meuble encastré présente lui une segmentation moins bonne que dans le nuage précédent, caractérisée par une importante omission au profit de « Remains ». En effet, une des portes de l'armoire est suffisamment entrouverte pour dépasser hors de la zone tampon associée à cet élément : certains points ne sont donc pas segmentés mais repris comme reste. Ici aussi la segmentation du pan de mur situé au-dessus de ces deux éléments est réussie. Enfin, pour la classe « Fenêtres », on retrouve le même cas de figure que pour les autres nuages de validation.

Ainsi, cette dernière validation fournit de bons résultats, malgré un bureau un peu plus bruité et un léger problème d'inclinaison du mur (moins pénalisant qu'une inclinaison du plafond).

VI.3. Limitations

Comme tout algorithme, la méthode développée ici présente certaines limitations qui ont été mises en évidence au fil de ce texte. Nous allons les détailler ici.

Tout d'abord, nos prétraitements sont relativement lourds, chronophages et nécessitent une connaissance a priori de la librairie Shapely. De plus, ces traitements n'ont pas été automatisés. Cependant, comme spécifié précédemment, ils sont indispensables au bon fonctionnement de l'algorithme : les plans utilisés doivent correspondre à un format bien spécifique. Ces prétraitements sont donc assez limitants et leur impact se fait ressentir sur la rapidité et l'efficacité de l'algorithme. S'ils ne sont pas ou mal réalisés, ils impacteront le fonctionnement même de l'algorithme et donc les résultats.

Un des principes limitant (mais inhérent) de notre méthode est que celle-ci repose principalement sur l'utilisation des données extraites du plan associé au nuage. Ce plan étant en 2D, il ne fournit aucune information concernant les hauteurs alors que ces informations sont déterminantes pour certaines classes telles que « Poutres » ou « Sol-Plafond ». Les données 3D doivent donc être extraites depuis le nuage.

L'indexation spatiale n'a pas pu être mise en place pour des raisons citées précédemment (cf. V.5.1). Le but aurait été d'appliquer l'indexation au nuage et de récupérer, par exemple, les coordonnées limites des sous-espaces créés pour une phase de pré-segmentation. En effet, avant de tester l'ensemble des points du nuage, nous aurions d'abord testé les intersections entre les zones tampons traitées et les limites de tous les sous-espaces indexés. Cela nous aurait permis de déterminer un index (cf. Fig. 8) des différents sous-espaces contenant des points susceptibles de tomber au sein de nos zones tampons. Nous aurions ainsi réduit considérablement, à chaque segmentation, le nombre de points à analyser et, par conséquent, accéléré cette opération. La non-indexation du nuage constitue donc une limitation à l'efficacité de notre méthode.

Les segmentations affinées de notre méthode sont limitées par différents choix. Tout d'abord, la même valeur seuil a été utilisée pour les zones tampons des éléments de classes différentes. Or, ceux-ci peuvent nécessiter des zones tampons plus ou moins larges. Si cela a été en partie pris en compte lors de la transformation des éléments du plan en éléments Shapely, cette transformation n'a modifié que l'extension interne de la zone tampon (intérieur de la zone considérée ou non).

Ainsi, le manque d'adaptation de l'extension externe de la zone tampon engendre des erreurs limitant l'exactitude de nos résultats. En outre, l'affinage de la segmentation de « Portes » et « Meubles encastrés » reste spécifique à notre cas d'étude. En effet, utiliser la hauteur maximale des murs comme valeur seuil pour la resegmentation de ces éléments ne fonctionnera pas nécessairement dans d'autres cas. Ce problème limite la reproductibilité de notre méthode. Les résultats possibles selon les diverses configurations des lieux ont été détaillés précédemment (cf. V.7.2.4) : la technique employée ne posera problème que si la hauteur des murs représentés est inférieure à celle des portes et meubles encastrés (extrêmement rare). Dans les autres cas, nous n'obtiendrons qu'une resegmentation partielle ou aucune modification. Ce constat atténue quelque peu l'impact de cette limitation. Nous avons aussi utilisé une distribution normale pour décrire les données lors de la détermination des hauteurs des poutres et du plafond, mais d'autres distributions pourraient être plus adaptées. L'absence d'affinage de la segmentation de « Fenêtres » engendre aussi des erreurs. Ces deux problèmes constituent des limitations pour la précision et l'exactitude des résultats.

En outre, nous utilisons comme références dans nos matrices de confusion des vérités terrains réalisées manuellement. Or, leur réalisation manuelle peut entrainer un certain nombre de petites erreurs. Ces petites erreurs engendreront des différences avec les résultats de notre méthode, différences reprises dans la matrice de confusion. Ainsi, l'analyse de ces matrices est complexifiée : il faut réussir à déterminer si les problèmes mis en évidence proviennent d'erreurs dans notre méthode ou dans la vérité terrain. Le résultat d'exactitude globale obtenu n'est donc pas parfaitement représentatif de l'exactitude de notre méthode mais sera peut-être légèrement sous-estimé. Par exemple, la classe « Sol-Plafond » de la vérité terrain, parce qu'elle est segmentée en premier lieu, va reprendre quelques points appartenant réellement à « Poutres ». Ces points seront, dans notre cas, correctement attribués à « Poutres » mais seront considérés comme omis de « Sol-Plafond ». Ce problème s'applique aussi aux statistiques Kappa qui étudient la concordance entre les résultats des méthodes. L'exactitude réelle d'une méthode ne pourra en être déduite que si l'autre méthode comparée présente des résultats parfaits. On pourrait donc plutôt parler d'exactitude relative.

Pour terminer, le fait que nos données (nuages) de validation proviennent de la même source que nos données (nuage) de paramétrage va limiter la pertinence de cette validation. En effet, les nuages provenant d'un même environnement, ils se ressemblent et éprouvent donc peu la méthode. L'idéal aurait été de valider la méthode sur des nuages issus d'une autre source.

CHAPITRE VII. CONCLUSION ET PERSPECTIVES

Pour conclure, nous allons reprendre les points et résultats importants présentés dans notre travail. Nous allons répondre à nos hypothèses et présenter des perspectives d'amélioration et de développements supplémentaires.

VII.1. Conclusion

Nous partons de constatations simples découlant de l'analyse de la littérature. Tout d'abord, l'utilisation des nuages de points 3D est en expansion depuis quelques années déjà. La segmentation-classification manuelle de ces nuages de points est fastidieuse, subjective et nécessite certaines connaissances a priori. Cette étape de traitement fait donc l'objet de nombreuses recherches pour être automatisée. Les plans 2D au format CAD constituent une source d'informations intéressante et utile pour la segmentation et la classification de nuages de points. L'analyse de la littérature nous permet de passer en revue différentes techniques de segmentation sémantique déjà mises au point. De cette analyse, il ressort que peu de recherches intègrent l'utilisation de plans CAD 2D dans leur méthode de segmentation.

Nous formulons alors une question-problème : « Est-il possible de développer une méthode automatique et efficace (en termes de rapidité, précision et exactitude) de segmentation et classification d'un nuage de points 3D d'intérieur de bâtiment, à partir d'informations sémantiques et spatiales extraites du plan CAD associé à cet environnement ? »

Nous en tirons deux hypothèses :

- « Il est effectivement possible d'extraire de l'information sémantique et spatiale depuis le plan d'intérieur au format CAD, et d'utiliser ces informations dans une opération de segmentation sémantique d'un nuage de points d'intérieur représentant le même environnement que le plan ».
- « Cette méthode de segmentation sémantique est plus rapide qu'une méthode manuelle et va fournir des résultats aux moins aussi précis et exacts que cette méthode manuelle ».

Grâce à ces hypothèses, nous avons mis au point une méthodologie reproductible composée de neuf grandes étapes. Après un certain nombre de prétraitements des données, nous développons une première méthode de segmentation sémantique et nous obtenons nos premiers résultats. Ces résultats sont améliorés avec une deuxième méthode plus précise et plus exacte, conçue en se basant sur l'utilisation de zones tampons. Pour des questions de rapidité et d'efficacité, un certain nombre d'améliorations sont mises en place et optimisent les temps de traitement. Par la suite, les segmentations de chaque classe sont, elles aussi, affinées afin d'obtenir des résultats toujours plus exacts. Cet affinage consiste principalement à l'introduction de la troisième dimension dans nos conditions de segmentation. Ainsi, avec cette méthode affinée, nous pouvons produire nos résultats finaux. Ce développement confirme notre première hypothèse puisque nous avons pu extraire de l'information d'un plan CAD 2D et l'utiliser dans la segmentation sémantique du nuage de points représentant le même environnement intérieur que ce plan.

La méthode subit ensuite une phase de validation : elle est appliquée à trois nuages, différents de celui utilisé pour son paramétrage. Les résultats pour ces trois nuages sont alors comparés à une vérité terrain au travers de plusieurs statistiques : matrice de confusion, exactitude globale, Kappa Index of Agreement (KIA, utilisant le Kappa de Cohen), …

Dans la phase d'analyse nous mettons, tout d'abord, en évidence les forces et faiblesses associées aux différentes étapes de notre méthodologie. Nous analysons notamment l'évolution du temps de traitement au cours du développement. Il en ressort que les optimisations des temps de traitement ont été efficaces et nécessaires. Nous en déduisons aussi que, malgré des prétraitements assez longs, la méthode est plus rapide qu'une segmentation manuelle. Cela répond en partie à notre seconde hypothèse.

Nous analysons ensuite les statistiques obtenues pour les résultats de segmentation des nuages de validation. Il apparait que quelques problèmes sont récurrents : décalage entre les représentations de portes dans le nuage et le plan, gestions différentes de l'ordre de segmentation, portes d'armoire ouvertes et non segmentées, segmentation de la classe « Fenêtres » trop discriminante, … L'affinage des segmentations est efficace dans la plupart des cas pour « Poutres » ainsi que pour « Portes » et « Meubles encastrés ». Toutefois, pour ces deux dernières, l'affinage est trop ajusté à notre cas d'étude (« Overfitting », difficilement reproductible) et donc moins pertinent. Il apparait aussi que des problèmes d'inclinaison des murs ou du plafond dans le nuage peuvent engendrer des soucis de segmentation.

Enfin, il ressort que notre méthode n'est que peu ou pas influencée par la présence de bruit ou d'occlusions. Inversément, elle ne permet pas non plus de distinguer ces problèmes. De manière générale, les segmentations de validation présentent des exactitudes globales et des indices de concordance (KIA) assez élevés (85% à 90%) lors de leur comparaison avec la méthode manuelle. Néanmoins, certains problèmes (inclinaison) peuvent faire diminuer ces résultats (cf. VI.2.1). L'exactitude et la précision de notre méthode se rapprochent donc de celles de la méthode manuelle mais ne les égalent pas. Nous ne pouvons donc pas confirmer la seconde partie de notre deuxième hypothèse.

Ces différents résultats sont à mettre en perspective avec les limitations associées à notre méthode : les vérités terrains ne sont pas des vérités absolues, les prétraitements ne sont pas automatisés, etc.

VII.2. Perspectives

Tout d'abord, plusieurs améliorations peuvent être envisagées, notamment, pour résoudre les limitations de notre méthode : mettre en œuvre l'indexation spatiale, utiliser d'autres nuages de validation, réaliser une comparaison avec une autre méthode, etc.

Pour ce qui est des perspectives de développement et de travaux futurs, il serait envisageable d'étendre l'échelle d'application de la méthode à un étage ou un bâtiment entier. Tout comme le travail réalisé par Djemâ (2018), notre méthode pourrait aussi servir de référence pour l'élaboration de méthodes similaires de segmentation sémantique (basées sur l'utilisation de plans CAD 2D) appliquées dans un contexte extérieur (autres que la segmentation des arbres, déjà abordée par Djemâ (2018)).

Il serait aussi envisageable d'utiliser notre méthode comme traitement facilitant les segmentations intégrées dans des travaux plus importants. Elle pourrait, par exemple, servir de base à une segmentation des éléments structurants d'un bâtiment, nécessaire à la création du modèle 3D de celui-ci (BIM, utilisé dans de nombreux domaines (cf. II.1.3)).

Enfin, notre méthode pourrait être combinée avec (ou intégrer) une méthode similaire d'extraction d'informations sur les hauteurs. Par exemple, des plans de coupe (sections), au format CAD, du bâtiment pourraient être envisagés comme sources d'informations. Le principe d'utilisation de ces plans serait semblable à notre méthode et apporterait de l'information externe sur les hauteurs.

CHAPITRE VIII. REFERENCES BIBLIOGRAPHIQUES

- Anaconda Documentation (2019). *Anaconda Navigator (version 1.9.6)*. Anaconda. https://docs.anaconda.com/anaconda/navigator/. Consulté le 20 février 2019.
- Anagnostopoulos, I., Pătrăucean, V., Brilakis, I., & Vela, P. (2016). Detection of Walls, Floors, and Ceilings in Point Cloud Data. *In* Perdomo-Rivera, J. L., Gonzáles-Quevedo, A., López del Puerto, C., Maldonado-Fortunet, F., & Molina-Bas, O. I. (Eds.), *Construction Research Congress 2016*, 2302–2311). https://doi.org/10.1061/9780784479827.229
- Armeni, I., Sener, O., Zamir, A. R., Jiang, H., Brilakis, I., Fischer, M., & Savarese, S. (2016). 3D Semantic Parsing of Large-Scale Indoor Spaces. *In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR*), *June 26–July 1, 2016, Las Vegas, USA.* IEEE Computer Society 2016, 1534–1543.
- Autodesk Help (2016). *ADETRANSFORM (Transform command).* Autodesk. https://knowledge.autodesk.com/support/autocad-map-3d/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/MAP3D-Use/files/GUID-AB726381-7DD2-49D0-9902-C1300C636A7C-htm.html. Consulté le 13 juin 2019.
- Autodesk Help (2016). *Calques.* Autodesk. https://knowledge.autodesk.com/fr/support/autocad/learn-explore/caas/CloudHelp/cloudhelp/2018/FRA/AutoCAD-Core/files/GUID-FA005756-B8F5-4A78-988F-31335A68D77C-htm.html. Consulté le 9 août 2019.
- Autodesk Help (2016). *Importation et exportation de fichiers DXF.* Autodesk. https://knowledge.autodesk.com/fr/support/autocad/learn-explore/caas/CloudHelp/cloudhelp/2018/FRA/AutoCAD-Core/files/GUID-D4242737-58BB-47A5-9B0E-1E3DE7E7D647-htm.html. Consulté le 8 août 2019.
- Autodesk Help (2017*). AutoCAD 2017 help*. Autodesk. https://help.autodesk.com/view/ACD/2017/ENU/.
- Billen, R. (2016). Conversion d'un système local de coordonnées à un système global. *In Topographie : instruments et méthodes.* Notes de cours de Bachelier en sciences géographiques, orientation générale, Liège, Université de Liège, inédit, 13 p.
- Brown, G., Butler, H. (2014). *Laspy : Documentation Version 1.2.5*. Laspy. https://laspy.readthedocs.io/en/latest/. Consulté le 13 mars 2019.
- Budroni, A., & Boehm, J. (2010). Automated 3D Reconstruction of Interiors from Point Clouds. *International Journal of Architectural Computing*, *8*(1), 55–73. https://doi.org/10.1260/1478-0771.8.1.55
- Chen, S., Laefer, D. F., & Mangina, E. (2016). State of Technology Review of Civilian UAVs. *Recent Patents on Engineering*, *10*(3), 160–174. https://doi.org/http://dx.doi.org/10.2174/1872212110666160712230039
- CloudCompare (2016). *CloudCompare User Manuel Version 2.6.1*. CloudCompare. http://www.cloudcompare.org/doc/qCC/CloudCompare%20v2.6.1%20-%20User%20manual.pdf Consulté le 16 Novembre 2018
- Cornet, Y. (2016). Validation des classifications. *In Télédétection avancée*. Notes de cours de Bachelier en sciences géographiques, orientation générale, Liège, Université de Liège, inédit, 7 p.
- Coughlan, J. M., & Yuille, A. L. (1999). Manhattan World: compass direction from a single image by Bayesian inference. *In Proceedings of the International Conference on Computer Vision, September 20-25, 1999, Kerkyra, Greece*. IEEE Computer Society 1999, 941–947 (vol. 2). https://doi.org/10.1109/iccv.1999.790349

- Djemâ, N., (2018). *Segmentation d'un nuage de points obtenu par scanner laser sur base d'un levé topographique classique*. Mémoire de master en sciences géographiques orientation géomatique et géométrologie, Liège, Université de Liège, inédit, 64 p. http://hdl.handle.net/2268.2/4317
- Donnay, J.-P. (2018). Implémentation du SIG. *In S.I.G.* Notes de cours de master en sciences géographiques, orientation géomatique et géométrologie, à finalité, Liège, Université de Liège, inédit, 46 p.
- Duncan, C. A., Goodrich, M. T., & Kobourov, S. (2001). Balanced Aspect Ratio Trees : Combining the Advantages of k -d Trees and Octrees. *Journal of Algorithms*, *38*, 303–333. https://doi.org/10.1006?jagm.2000.1135
- Engelmann, F., Kontogianni, T., Hermans, A., & Leibe, B. (2017). Exploring Spatial Context for 3D Semantic Segmentation of Point Clouds. *In 2017 IEEE International Conference on Computer Vision Workshops, ICCV Workshops 2017, October 22-29, 2017, Venice, Italy*. IEEE Computer Society 2017, 716–724. https://doi.org/10.1109/ICCVW.2017.90
- Esri (2018). *Stockage de données lidar (ArcMap 10.7).* Esri. http://desktop.arcgis.com/fr/arcmap/latest/manage-data/las-dataset/storing-lidar-data.htm Consulté le 8 août 2019.
- Fischler, M. A., & Bolles, R. C. (1981). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, *24*(6), 381–395.
- Geier, D. (2014). *Advanced Octrees 1: preliminaries, insertion strategies and maximum tree depth*. WordPress. https://geidav.wordpress.com/2014/07/18/advanced-octrees-1-preliminaries-insertion-strategies-and-max-tree-depth/. Consulté le 8 août 2019.
- Gillies, S. (2018). *The Shapely User Manual (version 1.6.4)*. Shapely. https://shapely.readthedocs.io/en/stable/manual.html. Consulté le 10 juin 2019.
- Gimenez, L., Hippolyte, J., Robert, S., Suard, F., & Zreik, K. (2015). Review : reconstruction of 3D building information models from 2D scanned plans. *Journal of Building Engineering*, *2*, 24–35. https://doi.org/10.1016/j.jobe.2015.04.002
- Grilli, E., Menna, F., & Remondino, F. (2017). A REVIEW OF POINT CLOUDS SEGMENTATION AND CLASSIFICATION ALGORITHMS. *In* Aguilera, D., Georgopoulos, A., Kersten, T., Remondino, F., & Stathopoulou, E. (Eds.), *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, March 1-3, 2017, Nafplio, Greece*, *Vol. XLII-2/W3*, 339–344. https://doi.org/10.5194/isprs-archives-XLII-2-W3-339-2017
- Günther, M., Wiemann, T., Albrecht, S., & Hertzberg, J. (2017). Model-based furniture recognition for building semantic object maps. *Artificial Intelligence*, *247*, 336–351. https://doi.org/10.1016/j.artint.2014.12.007
- Hackel, T., Wegner, J. D., & Schindler, K. (2016). FAST SEMANTIC SEGMENTATION OF 3D POINT CLOUDS WITH STRONGLY VARYING DENSITY. *In* Halounova, L., Šafář, V., Gong, J., Hanzl, V., Wu, H., Vyas, A., … Faltýnová, M. (Eds.), *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, July 12-19, 2016, Prague, Czech Republic*, *Vol. III–3*,177–184. https://doi.org/10.5194/isprsannals-III-3-177-2016
- Harris Geospatial Solutions (2018). *Calculate Confusion Matrices*. Harris Geospatial Solutions. https://www.harrisgeospatial.com/docs/CalculatingConfusionMatrices.html. Consulté le 15 juillet 2019.
- Hermans, A., Floros, G., & Leibe, B. (2014). Dense 3D semantic mapping of indoor scenes from RGB-D images. *In Proceedings of the 2014 IEEE International Conference on Robotics and Automation, May 31-June 5, 2014, Honk Kong, China*. IEEE 2014, 2631–2638. https://doi.org/10.1109/ICRA.2014.6907236
- Hong, S., Jung, J., Kim, S., Cho, H., Lee, J., & Heo, J. (2015). Semi-automated approach to indoor mapping for 3D as-built building information modeling. *Computers, Environment and Urban Systems*, *51*, 34–46. https://doi.org/10.1016/j.compenvurbsys.2015.01.005

- Huang, H. C., Lo, S. M., Zhi, G. S., & Yuen, R. K. K. (2008). Graph theory-based approach for automatic recognition of CAD data. *Engineering Applications of Artificial Intelligence*, *21*, 1073–1079. https://doi.org/10.1016/j.engappai.2007.12.001
- Hunter, J., Dale, D., Firing, E., Droettboom, M., & The Matplotlib development team (2019). *Matplotlib User's Guide Version 3.1.0*. The Matplotlib development team. https://matplotlib.org/3.1.0/users/index.html. Consulté le 15 juillet 2019.
- Kim, C., Son, H., & Kim, C. (2013). Fully automated registration of 3D data to a 3D CAD model for project progress monitoring. *Automation in Construction*, *35*, 587–594. https://doi.org/10.1016/j.autcon.2013.01.005
- Landrieu, L., & Simonovsky, M. (2018). Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs. *In Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2018), June 18-22, 2018, Salt Lake City, USA.* IEEE Computer Society 2018, 4558–4567.
- Lewis, R., & Séquin, C. (1998). Generation of 3D building models from 2D architectural plans. *Computer-Aided Design*, *30*(10), 765–779.
- Macher, H., Landes, T., & Grussenmeyer, P. (2017). From Point Clouds to Building Information Models: 3D Semi-Automatic Reconstruction of Indoors of Existing Buildings. *Applied Sciences*, *7*(10), 30 p. https://doi.org/10.3390/app7101030
- McHugh, M. L. (2012). Interrater reliability: the kappa statistic. *Biochemia Medica 2012(Zagreb)*, *22*(3), 276–282.
- Moizi, M. (2018). *Dxfgrabber Version 1.0.0.* Dxfgrabber. https://dxfgrabber.readthedocs.io/en/latest/. Consulté le 13 avril 2019.
- Nan, L., Xie, K., & Sharf, A. (2012). A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics*, *31*(6), 10. https://doi.org/10.1145/2366145.2366156
- Nguyen, A., & Le, B. (2013). 3D point cloud segmentation : A survey. *In Proceedings of the 2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM), November 12-15, 2013, Manila, Philippines*. IEEE 2013, 225–230. https://doi.org/10.1109/RAM.2013.6758588
- Nguyen, C. H. P., & Choi, Y. (2018). Comparison of point cloud data and 3D CAD data for on-site dimensional inspection of industrial plant piping systems. *Automation in Construction*, *91*, 44–52. https://doi.org/10.1016/j.autcon.2018.03.008
- Nurunnabi, A., Belton, D., & West, G. (2016). Robust Segmentation for Large Volumes of Laser Scanning Three-Dimensional Point Cloud Data. *IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING*, *54*(8), 4790–4805. https://doi.org/10.1109/TGRS.2016.2551546
- Obadia, S. (2019). *La numérisation au bon secours de Notre-Dame*. Les cahiers techniques du bâtiment. https://www.cahiers-techniques-batiment.fr/article/la-numerisation-au-bon-secours-de-notre-dame.40380. Consulté le 12 août 2019.
- Okorn, B., Xiong, X., Akinci, B., & Huber, D. (2010). Toward Automated Modeling of Floor Plans. *In Proceedings of the 5th International Symposium on 3D Data Processing, Visualization and Transmission, May 17-20, 2010, Paris, France, Vol.2*, 8 p.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesney, E. (2011). Scikit-learn : Machine Learning in Python. *Journal Of Machine Learning Research*, *12*, 2825–2830.
- Poux, F., & Billen, R. (2019). Voxel-based 3D Point Cloud Semantic Segmentation : Unsupervised Geometric and Relationship Featuring vs Deep Learning Methods. *ISPRS International Journal of Geo-Information*, *8*(5), 34 p. https://doi.org/10.3390/ijgi8050213
- Poux, F., Hallot, P., Neuville, R., & Billen, R. (2016). SMART POINT CLOUD : DEFINITION AND REMAINING CHALLENGES. *In* Dimopoulou, E., & van Oosterom, P. (Eds.), *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume IV-2/W1, 2016 11th 3D Geoinfo Conference, October 20–21, 2016, Athens, Greece*, 119–127. https://doi.org/10.5194/isprs-annals-IV-2-W1-119-2016

- Poux, F., Neuville, R., & Billen, R. (2017). Point cloud classification of tesserae from terrestrial laser data combined with dense image matching for archaeological information extraction. *In* Hayes, J., Ouimet, C., Quintero, M. S., Fai, S., & Smith, L. (Eds.), *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume IV-2/W2, 2017 26th International CIPA Symposium, August 28– September 01, 2017, Ottawa, Canada*, 203–211. https://doi.org/10.5194/isprs-annals-IV-2-W2-203-2017
- Poux, F., Neuville, R., Hallot, P., & Billen, R. (2017). Model Semantically Rich Point Cloud Data. *In* Kalantari, M., & Rajabifard, A. (Eds.), *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume IV-4/W5, 2017 12th 3D Geoinfo Conference, October 26–27 , 2017, Melbourne, Australia*, 107–115. https://doi.org/https://doi.org/10.5194/isprs-annals-IV-4-W5-107-2017
- Poux, F., Neuville, R., Nys, G. A., & Billen, R. (2018). 3D point cloud semantic modelling: Integrated framework for indoor spaces and furniture. *Remote Sensing*, *10*, 35 p. https://doi.org/10.3390/rs10091412
- Previtali, M., Barazzetti, L., Brumana, R., Cuca, B., Oreni, D., Roncoroni, F., & Scaioni, M. (2013). Automatic Façade Segmentation for Thermal Retrofit. *In* Boehm, J., Remondino, F., Kersten, T., Fuse, T., & Gonzalez-Aguilera, D. (Eds.), *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XL-5/W1, 3D-ARCH 2013 - 3D Virtual Reconstruction and Visualization of Complex Architectures, February 25 – 26, 2013, Trento, Italy*, 197–204. https://doi.org/10.5194/isprsarchives-xl-5-w1-197-2013
- Previtali, M., Barazzetti, L., Brumana, R., & Scaioni, M. (2014). Towards automatic indoor reconstruction of cluttered building rooms from point clouds. *In* Remondino, F., & Menna, F. (Eds.), *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume II-5, 2014 ISPRS Technical Commission V Symposium, June 23 – 25, 2014, Riva del Garda, Italy*, 281–288. https://doi.org/10.5194/isprsannals-II-5-281-2014
- Qi, C. R., Su, H., Kaichun, M., & Guibas, L. J. (2017). PointNet : Deep Learning on Point Sets for 3D Classification and Segmentation. *In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, July 21-26, 2017, Honolulu, USA*. IEEE Computer Society 2017, 77–85. https://doi.org/10.1109/CVPR.2017.16
- Rajala, M., & Penttilä, H. (2006). Testing 3D Building Modelling Framework in Building Renovation. *In Communicating Space(s):24th eCAADe Conference Proceedings, 2006, University of Thessaly, Volos, Greece*, 268–275.
- Saftly, W., Baes, M., & Camps, P. (2014). Hierarchical octree and k -d tree grids for 3D radiative transfer simulations. *Astronomy & Astrophysics*, *561*, 9 p. https://doi.org/10.1051/0004-6361/201322593
- Schnabel, R., Wahl, R., & Klein, R. (2007). Efficient RANSAC for Point-Cloud Shape Detection. *COMPUTER GRAPHICS Forum*, *26*(2), 214–226.
- Scikit-learn (2019). *Confusion matrix*. Scikit-learn. https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html# Consulté le 17 juillet 2019
- Scikit-learn (2019). *Documentation of scikit-learn 0.21.3*. Scikit-learn. https://scikit-learn.org/stable/documentation.html Consulté le 17 juillet 2019
- Silberman, N., & Fergus, R. (2011). Indoor scene segmentation using a structured light sensor. *In Proceedings of the IEEE International Conference on Computer Vision Workshops, ICCV 2011 Workshops, November 6-13, 2011, Barcelona, Spain*. IEEE Computer Society 2011, 601–608. https://doi.org/10.1109/ICCVW.2011.6130298
- Spyder Docs (2018). *Spyder : The Scientific Python Development Environment — Documentation (Version 3.3.4)*. Spyder. https://docs.spyder-ide.org/index.html. Consulté le 20 février 2019.

- Tang, P., Huber, D., Akinci, B., Lipman, R., & Lytle, A. (2010). Automatic reconstruction of as-built building information models from laser-scanned point clouds : A review of related techniques. *Automation in Construction*, *19*, 829–843. https://doi.org/10.1016/j.autcon.2010.06.007
- The SciPy community (2019). *Numpy User Guide Version 1.16.4*. SciPy.org. https://docs.scipy.org/doc/numpy/user/. Consulté le 9 avril 2019.
- The SciPy community (2019). *SciPy Version 1.3.0*. SciPy.org. https://docs.scipy.org/doc/scipy/reference/.
- Valero, E., Adán, A., & Cerrada, C. (2012). Automatic method for building indoor boundary models from dense point clouds collected by laser scanners. *Sensors*, *12*, 16099–16115. https://doi.org/10.3390/s121216099
- Xiong, X., Adan, A., Akinci, B., & Huber, D. (2013). Automatic creation of semantically rich 3D building models from laser scanner data. *Automation in Construction*, *31*, 325–337. https://doi.org/10.1016/j.autcon.2012.10.006
- Xiong, X., & Huber, D. (2010). Using Context to Create Semantic 3D Models of Indoor Environments. *In Proceedings of the 2010 British Machine Vision Conference, BMVC 2010, August 31 - September 3, 2010, Aberystwyth, UK*. British Machine Vision Association 2010, 1–11. https://doi.org/10.5244/C.24.45
- Ziai, A. (2017). *Inter-rater agreement Kappas.* Towards Data Science. https://towardsdatascience.com/inter-rater-agreement-kappas-69cd8b91ff75. Consulté le 15 juillet 2019.

CHAPITRE IX. ANNEXES

IX.1. Code de segmentation

```python
# -*- coding: utf-8 -*-
"""
Created on Mon May 20 14:51:36 2019

@author: LESECQUE Florian
"""
""" ###########   Bibliothèques & directory file   ########### """
import os
os.chdir(r'C:\Users\User\Desktop\GEOGRAPHIE Unif\Master 2\MEMOIRE\5 CODE_test\TEST_Office_4')
import numpy as np
import time
import laspy as lp
import dxfgrabber as dg
from shapely.geometry import LineString, Point, Polygon
#import matplotlib as mpl


#%%
""" ###########   FONCTIONS   ########### """

""" Fonction pour écrire des résultats sous .las """

def laswrite (numpyPC, filename): #exemple of filename "Lasfile.las"
    t = lp.file.File(filename, mode="w", header=cloud.header)
    try:
        if (len(numpyPC)>0):
            t.x= numpyPC[:,0] # numpyPC = numpy array a enregistrer,
            t.y= numpyPC[:,1] #les colonnes [:,1] correspondent aux attributs à spécifier
            t.z= numpyPC[:,2]
            t.intensity= numpyPC[:,3]
            t.flag_byte= numpyPC[:,4]
            t.raw_classification= numpyPC[:,13]
            t.scan_angle_rank= numpyPC[:,6]
            t.user_data=numpyPC[:,7]
            t.pt_src_id= numpyPC[:,8]
            t.gps_time= numpyPC[:,9]
            t.Red= numpyPC[:,10]
            t.Green= numpyPC[:,11]
            t.Blue= numpyPC[:,12]
            return "Las file saved!"
        else:
            return "Numpy point cloud empty!"
    finally: t.close()

""" Fonction pour créer les points shapely """
def shppoint (remains):
    points_shapely=list()
    i=0
    while (i<len(remains)): # Crée un point shapely via les XYZ du point
        points_shapely.append(Point(remains[i][0],remains[i][1],remains[i][2]))
        i+=1
    return points_shapely

""" Fonction pour testet les points pour toute les entités d'un calque  """
def testpoint(buffer,point):
    boolean=False
    for B in buffer:
        if (B.contains(point)): # Si le buffer contient le point
            boolean=True
            return boolean


#%%
""" ########### Importation du Nuage & échantillonnage ########### """

### A) Importation
start=time.clock()
print("1) Loading Cloud")
tm=time.clock()
cloud= lp.file.File('Office_4.las')

### B) Attribut du .las
x=cloud.x
y=cloud.y
z=cloud.z
I=cloud.intensity
f=cloud.flag_byte
rc= cloud.raw_classification
sar=cloud.scan_angle_rank
ud=cloud.user_data
ID=cloud.pt_src_id
GPS=cloud.gps_time
R=cloud.Red
G=cloud.green
B=cloud.blue

### C) Creation array numpy avec les attributs du .las --> + rapide que "numpy.asarray" !
npcloud =(np.array([x,y,z,I,f,rc,sar,ud,ID,GPS,R,G,B])).T
print(time.clock()-tm, "secondes") # ici :3secondes VS np.asarray(cLoud) : 2min30
print("")


### D) Echantillonage ###
print("2) Sampling cloud : ")
tm=time.clock()
sampcloud = npcloud[0::1] #Nuage échantillonné
print(time.clock()-tm, "secondes")
print("")


#%%
""" ########### Importation du DXF & Extraction des informations ###########  """

### A) Importation
print("3) Load dxf")
dxf=dg.readfile("Office_4.dxf")
```

```

### B) Extraction des informations
print("4) Extraction Layer & Entities")
tm=time.clock()
Layers= dxf.layers #table des calques du dessin
Entities= dxf.entities #table des entités du dessin

### C) Création des listes de noms, d'entités et du dictionnaire ###
layername=list() #noms
layer_ent=list() #entités
dico_layer={} #dictionnaire
i=0
for layers in Layers: #Remplissage des 2 listes (noms et entités) et du dictionnaire
    if layers.name[0:4]=='PCS_': #on ne reprend que les calques mis en forme
        layername.append(layers.name)
        layer_ent.append([entity for entity in Entities if entity.layer==layername[i]])
        dico_layer[layername[i]]=layer_ent[i] #dictionnaire
        i+=1
print(time.clock()-tm, "secondes")
print("")



#%%
""" ###########  PREPARATION SEGMENTATION ########### """

""" Récupération du nuage entier + Creation des points shapely """

print("5) Création de remains :")
tm=time.clock()
remains=list() #nuage utilisé pour la segmentation (réduit au fur et à mesure des segmentations)
for row in sampcloud:
    remains.append(row)
remains=np.asarray(remains)
print(time.clock()-tm, "secondes")
print("")


""" ########### POINTS SHAPELY ########### """

print("6) Création points shapely :") #Les points sont créés une fois et puis conservés d'une segmentation à l'autre
tm=time.clock()
shp_point= shppoint(remains) #fonction définie au début du code
print(time.clock()-tm, "secondes")
print("Temps pré-segmentation : ", time.clock()-start, "secondes")
print("")



#%%
""" ########### PARTIE POUTRE ########### """

""" 1ère Segmentation """

### A) Récupère les entités
print("7) POUTRES")
for cle in dico_layer:
    if ('Beam' in cle):entite=dico_layer[cle]

### B)  Détermine les Zones Tampons
print("Buffer creation :")
tm=time.clock()
polygon= list() #polygone car on veut l'intérieur de la zone tampon
buff=list()
i=0
for Entite in entite: # Pour chaque entité, on crée un polygone et détermine sa zone tampon
    polygon.append(Polygon(Entite.points))
    buff.append(polygon[i].buffer(0.025))
    i+=1
print(time.clock()-tm, "secondes")

### C) Crée les listes temporaires de résultats
PC_Poutre_temp =list() #liste pour les points vérifiant la condition
PC_Poutre_pt_shp=list() #liste pour les points shapely correspondant
temp=list() #liste pour les points ne vérifiant pas la condition
pointtemp=list() #liste pour les points shapely correspondant à ceux de temp
print("temps total :", time.clock()-start, "secondes")
print("")


### D) Première segmentation en 2 listes
print("Première Segmentation :")
tm=time.clock()
i=0
for p in shp_point:
    boolean=testpoint(buff,p) # utilise la fonction définie en début de code
    if boolean == True:
        PC_Poutre_temp.append(remains[i])
        PC_Poutre_pt_shp.append(p)
    else:
        temp.append(remains[i]) # liste pour récréer, avec les points non segmentés, un nuage 'remains' réutilisé par la suite
        pointtemp.append(p)  #liste temporaire pour conserver les points shapely correspondant à ceux de temps et ne pas les recréer
    i+=1

print(time.clock()-tm, "secondes")
print("")

PC_Poutre_temp=np.asarray(PC_Poutre_temp)


""" 2ème Segmentation """

### E) Création de statistiques sur les z des points de PC_Poutre_temp et extraction de la classe correspondant au début des Poutres
zmin = np.amin(PC_Poutre_temp[:,2])
zmax = np.amax(PC_Poutre_temp[:,2])
Zrange = (zmax-zmin) #Récupère le range en Z pour spécifier les classes dans l'histogramme
freq, bins =np.histogram(PC_Poutre_temp[:,2], bins=(int(round(Zrange/0.1)))) #bins=nombre de colonnes dans l'histo (ici, colonnes de 10cm)
#mpl.pyplot.hist(PC_Poutre_temp[:,2], bins=(int(round(Zrange/0.1))))

frequency=list()#Liste des fréquences par classe de 10cm en z : tuple (borne inférieure de la classe, fréquence de la classe)
i=0
for frq in freq:
    frequency.append([bins[i],freq[i]])
    i+=1
frequency=np.asarray(frequency)
sorted_freq = np.flip(frequency[frequency[:,1].argsort()]) # Tri par fréquence
```

```

### F) Recherche de la borne inférieure de la classe de pic
B_value=max(PC_Poutre_temp[:,2]) #B_value pour "beam"_value --> valeur de base = valeur maximale dans le nuage
i=0
for z in sorted_freq: #Recherche la classe de pic correspondant à la base de la poutre, z[0]=bornes z[1]=fréquence
    if np.all([z[1]>(zmax- (0.25*Zrange)), z[0]>(0.1*sum(sorted_freq[:,0])) ]): #  le pic est gardé s'il est dans le 1/4 supérieur des z et d'une fréque
        B_value=z[1] #La première valeur obtenue sera la plus fréquente de ce quart grâce au tri
        break
    i+=1

### G) Extraction des points de la classe de pic déterminée
beam=list()
i=0
for z in PC_Poutre_temp[:,2]: #Pour chaque point segmenté précédemment
    if np.all([B_value<z, z<=(B_value+0.1)]): #si le point est dans la classe pic trouvée
        beam.append(z) # Récupère uniquement le z
    i+=1
beam=np.asarray(beam)

### H) Statistiques sur les données de la classe  pic et détermination de la borne inférieure de l'intervalle
if (len(beam)>0): #Si on a des points pré-segmentés
    #mpl.pyplot.hist(beam)
    μ = np.mean(beam) #moyenne des z de la classe de pic
    std = np.std(beam) #écart-type des z de la classe de pic
    BI = μ - 1.96*std # On prend la borne inférieur d'un intervalle à 95% (1.96*std)
else:
    BI=max(PC_Poutre_temp[:,2])

### I) Segmentation et classification
PC_Poutre =list() # Liste résultat
print("Deuxième Segmentation :")
tm=time.clock()
i=0
for z in PC_Poutre_temp[:,2]:
    if (BI<z): # on test si les points tombent dans l'intervalle
        PC_Poutre.append(np.hstack((PC_Poutre_temp[i],4))) # On ajoute le tag à la fin car avant les points pourraient être rejetés
        # 4 = tag de classification pour Poutre
    else:
        temp.append(PC_Poutre_temp[i])
        pointtemp.append(PC_Poutre_pt_shp[i])
    i+=1

### J) Enregistrement des points non segmentés pour la suite des segmentations
remains=np.asarray(temp)
shp_point=pointtemp
print("temps de segmentation: ", time.clock()-tm, "secondes")
print("temps total POUTRE :", time.clock()-start, "seconds")
print("")



#%%
""" ########## PARTIE SOL et PLAFOND ########## """
print("8) SOL et PLAFOND ")
print("Première segmentation :")
tm=time.clock()

### A) Création de l'histogramme (col = 0.1m) des z du nuage et extraction des classes pic correspondant au sol et plafond
zmin = np.amin(remains[:,2])
zmax = np.amax(remains[:,2])
Zrange = (zmax-zmin) #Récupère l'extension en Z pour spécifier les classes dans l'histogramme
freq, bins =np.histogram(remains[:,2], bins=(int(round(Zrange/0.1)))) #bins = nombre de colonnes dans l'histo (colonnes de 10cm)
#mpl.pyplot.hist(remains[:,2], bins=(int(round(Zrange/0.1))))


frequency=list()#Liste des fréquences par classe de 10cm en z
i=0
for frq in freq:
    frequency.append([bins[i],freq[i]])
    i+=1
frequency=np.asarray(frequency)
sorted_freq = np.flip(frequency[frequency[:,1].argsort()]) # Tri par fréquence

for z in sorted_freq[:,1]: #Recherche la borne inférieure de la classe de pic sol (dans le premier quart des z)
    if (z< (zmin+ (0.25*Zrange))):
        F_value=z
        break
for z in sorted_freq[:,1]: #Recherche la borne inférieure de la classe de pic plafond (dans le dernier quart des z)
    if (z> (zmax- (0.25*Zrange))):
        C_value=z
        break

### B) Première segmentation (Large)
ceiling=list()
floor=list()
i=0
for z in remains[:,2]:
    if np.all([F_value<=z, z<=(F_value+0.1)]):
        floor.append(z) # on ne récupère que les z
    elif np.all([C_value<=z, z<=(C_value+0.1)]):
        ceiling.append(z)
    i+=1
ceiling=np.asarray(ceiling)
floor=np.asarray(floor)
print("temps de segmentation: ", time.clock()-tm, "secondes")
print("")


### C) Statistiques sur les résultats obtenus
#mpl.pyplot.hist(ceiling[:,2]) #Gaussienne
ceiling_μ = np.mean(ceiling) #moyenne de la classe pic plafond
ceiling_std = np.std(ceiling) #écart-type de la classe pic plafond
ceiling_int = ceiling_μ -1.96*ceiling_std # On prend un intervalle à 95% (1.96*std)
# On reprendra les points de ceiling depuis la borne inférieure de l'intervalle jusqu'au maximum

#mpl.pyplot.hist(floor[:,2])  # Gaussienne
floor_μ = np.mean(floor) #moyenne de la classe  pic sol
floor_std = np.std(floor) #écart-type de la classe pic sol (Gaussienne)
floor_int = floor_μ + 1.96*floor_std # On prend un intervalle à 95% (1.96*std)
# On reprendra les points de floor depuis le minimum jusqu'à la borne supérieure de l'intervalle

#mpl.pyplot.hist(floor, label='Données')


### D) Crée la liste des résultats
PC_Sol_Plafond =list()
#PC_Sol =list()
#PC_Plafond =list()

```

```
### E) Segmentation et classification
print("Segmentation affinée :")
tm=time.clock()
temp=list()
pointtemp=list()
i=0
for z in remains[:,2]:
    if (z<=floor_int): # condition pour les points de sol
        PC_Sol_Plafond.append(np.hstack((remains[i],2)))
        # 2 = tag de classification pour Sol-plafond
        #PC_Sol.append(np.hstack((floor[i],2)))
    else:
        temp.append(remains[i])
        pointtemp.append(shp_point[i])
    i+=1
remains=np.asarray(temp)
shp_point=pointtemp

temp=list()
pointtemp=list()
i=0
for z in remains[:,2]:
    if (ceiling_int<=z): # condition pour les points de plafond
        PC_Sol_Plafond.append(np.hstack((remains[i],2)))
        #PC_Plafond.append(np.hstack((ceiling[i],8)))
    else:
        temp.append(remains[i])
        pointtemp.append(shp_point[i])
    i+=1

### F) Enregistrement des points non segmentés pour la suite des segmentations
remains=np.asarray(temp)
shp_point=pointtemp
print("temps de segmentation: ", time.clock()-tm, "secondes")
print("temps total SOL-PLAFOND :", time.clock()-start, "secondes")
print("")


#%%
""" ########## PARTIE MURS ########## """

### A) Récupère les entités
print("9) MURS")
for cle in dico_layer:
    if ('Wall' in cle):entite=dico_layer[cle]

### B)  Détermine les zones tampons
print("Buffer creation :")
tm=time.clock()
polyline= list() # polyline car on travaille avec des lignes simpless (et non refermées)
buff=list()
i=0
for Entite in entite:
    polyline.append(LineString(Entite.points))
    buff.append(polyline[i].buffer(0.025))
    i+=1
print(time.clock()-tm, "secondes")

### C) Crée des listes résultast et temporaires pour remains et shp_point
PC_Mur =list() # points de la classe "Murs"
temp=list() # points non segmentés
pointtemp=list() # points shapely non segmentés
print("temps total :", time.clock()-start, "secondes")
print("")


### D) Segmentation et classification
print("Segmentation :")
tm=time.clock()
i=0
for p in shp_point:
    boolean=testpoint(buff,p)
    if boolean == True:
        PC_Mur.append(np.hstack((remains[i],3)))
        # 3 = tag de classification pour Mur
    else:
        temp.append(remains[i])
        pointtemp.append(p)
    i+=1

### E) Enregistrement des points non segmentés pour le suite des segmentations
remains=np.asarray(temp)
shp_point=pointtemp
print("temps de segmentation: ", time.clock()-tm, "secondes")
print("temps total :", time.clock()-start, "secondes")
print("")

### F) Récupère la valeur z maximale du mur pour la segmentation des Meubles encastrés et des Portes
npMur=np.asarray(PC_Mur)
#mpl.pyplot.hist(npMur[:,2], bins=(int(round(Zrange/0.1))), rwidth=0.80)
Zmax_mur = np.amax(npMur[:,2])


#%%
""" ########## PARTIE MEUBLES ENCASTRES ########## """
### A) Récupère les entités
print("10) MEUBLES ENCASTRES")
for cle in dico_layer:
    if ('Embedded' in cle):entite=dico_layer[cle]

### B)  Détermine les zones tampons
print("Buffer creation :")
tm=time.clock()
polygon= list()
buff=list()
i=0
for Entite in entite:
    polygon.append(Polygon(Entite.points))
    buff.append(polygon[i].buffer(0.025))
    i+=1


### C) Crée la liste résultat et les listes temporaires pour remains et shp_point
PC_ME_temp = list()
temp=list()
pointtemp=list()
print(time.clock()-tm, "secondes")
print("temps total :", time.clock()-start, "secondes")
print("")

```

```
### D) Première Segmentation
print("Première Segmentation :")
tm=time.clock()
i=0
for p in shp_point:
    boolean=testpoint(buff,p)
    if boolean == True:
        PC_ME_temp.append(remains[i])
    else:
        temp.append(remains[i])
        pointtemp.append(p)
    i+=1
PC_ME_temp=np.asarray(PC_ME_temp)
remains=np.asarray(temp)
shp_point=pointtemp
print("temps de segmentation :",time.clock()-tm, "secondes")
print("")

### E) Deuxième segmentation via Zmax_mur (cf. fin partie mur). Le reste est rajouté non pas à remains mais à PC_Mur
print("Deuxième Segmentation :")
PC_Meuble_Encastre =list()
i=0
for z in PC_ME_temp[:,2]:
    if (z <= Zmax_mur):
        PC_Meuble_Encastre.append(np.hstack((PC_ME_temp[i],5)))
        # 5 = tag de classification pour Meubles_Encastrés
    else:
        PC_Mur.append(np.hstack((PC_ME_temp[i],3))) # Si le point n'appartient pas à Meubles encastrés, il appartient à Mur.
    i+=1

### F) Enregistrement des points non segmentés pour le suite des segmentations
remains=np.asarray(temp)
shp_point=pointtemp
print("temps de segmentation :",time.clock()-tm, "secondes")
print("temps total :", time.clock()-start, "secondes")
print("")


#%%
""" ########## PARTIE PORTES ########## """
### A) Récupère les entités
print("11) PORTES")
for cle in dico_layer:
    if ('Door' in cle):entite=dico_layer[cle]

### B)  Détermine les zones tampons
print("Buffer creation :")
tm=time.clock()
polygon= list()
buff=list()
i=0
for Entite in entite:
    polygon.append(Polygon(Entite.points))
    buff.append(polygon[i].buffer(0.025))
    i+=1


### C) Crée la liste résultat et les listes temporaires pour remains et shp_point
PC_Porte_temp =list()
temp=list()
pointtemp=list()
print(time.clock()-tm, "secondes")
print("temps total :", time.clock()-start, "secondes")
print("")

### D) Première Segmentation
print("Première Segmentation :")
tm=time.clock()
i=0
for p in shp_point:
    boolean=testpoint(buff,p)
    if boolean == True:
        PC_Porte_temp.append(remains[i])
        # 6 = tag de classification pour Porte
    else:
        temp.append(remains[i])
        pointtemp.append(p)
    i+=1
PC_Porte_temp=np.asarray(PC_Porte_temp)
print("temps de segmentation :",time.clock()-tm, "secondes")
print("")


### E) Deuxième segmentation via Zmax_mur (cf fin partie mur). Le reste est rajouté non pas à remains mais à PC_Mur
print("Deuxième Segmentation :")
PC_Porte =list()
i=0
for z in PC_Porte_temp[:,2]:
    if (z <= Zmax_mur):
        PC_Porte.append(np.hstack((PC_Porte_temp[i],6)))
        # 6 = tag de classification pour Porte
    else:
        PC_Mur.append(np.hstack((PC_Porte_temp[i],3))) # Si le point n'appartient pas à Porte, il appartient à Mur.
    i+=1


### F) Enregistrement des points non segmentés pour le suite des segmentation
remains=np.asarray(temp)
shp_point=pointtemp
print("temps de segmentation: ", time.clock()-tm, "secondes")
print("temps total :", time.clock()-start, "secondes")
print("")


#%%
""" ########## PARTIE FENETRES ########## """
### A) Récupère les entités
print("12) FENETRES")
for cle in dico_layer:
    if ('Window' in cle):entite=dico_layer[cle]

### B)  Détermine les zones tampons
print("Buffer creation :")
tm=time.clock()
polygon= list()
buff=list()
i=0
for Entite in entite:
    polygon.append(Polygon(Entite.points))
    buff.append(polygon[i].buffer(0.025))
```

```
    i+=1

### C) Crée la liste résultat et les listes temporaires pour remains et shp_point
PC_Fenetre =list()
temp=list()
pointtemp=list()
print(time.clock()-tm, "secondes")
print("temps total :", time.clock()-start, "secondes")
print("")

### D) Segmentation et classification
print("Segmentation :")
tm=time.clock()
i=0
for p in shp_point:
    boolean=testpoint(buff,p)
    if boolean == True:
        PC_Fenetre.append(np.hstack((remains[i],7)))
        # 7 = tag de classification pour Porte
    else:
        temp.append(remains[i])
        pointtemp.append(p)
    i+=1

### E) Enregistrement des points non segmentés
remains=np.asarray(temp)
shp_point=pointtemp
print("temps de segmentation: ", time.clock()-tm, "secondes")
print("temps total :", time.clock()-start, "secondes")
print("")


#%%
""" ########## Ecriture des résultats sous .las ########## """
print("13) Enregistrement des résultats:")
tm=time.clock()
PC_total = list() #Pour créer un nuage total classifié

### Utilisation de la fonction implémentée au début du code ( ATTENTION à la conversion liste->array numpy )
laswrite(np.asarray(PC_Sol_Plafond),"T_04_PC_Sol_Plafond.las")
PC_total.extend(PC_Sol_Plafond)
#laswrite(np.asarray(PC_Sol),"PC_Sol.las")
#laswrite(np.asarray(PC_Plafond),"PC_Plafond.las")
laswrite(np.asarray(PC_Mur),"T_04_PC_Mur.las")
PC_total.extend(PC_Mur)
laswrite(np.asarray(PC_Poutre),"T_04_PC_Poutre.las")
PC_total.extend(PC_Poutre)
laswrite(np.asarray(PC_Meuble_Encastre),"T_04_PC_Meuble_Encastre.las")
PC_total.extend(PC_Meuble_Encastre)
laswrite(np.asarray(PC_Porte),"T_04_PC_Porte.las")
PC_total.extend(PC_Porte)
laswrite(np.asarray(PC_Fenetre),"T_O4_PC_Fenetre.las")
PC_total.extend(PC_Fenetre)

remains=np.c_[remains, np.ones(len(remains))] #ajoute le tag de classification pour les points non classés = 1
laswrite(remains,"T_04_remains.las")
PC_total.extend(remains)

laswrite(np.asarray(PC_total),"T_O4_PC_total.las")

```

Figure 51 Code de segmentation final

IX.2. Code de calcul des matrices de confusion et statistiques liées

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jul 17 00:25:34 2019

@author: User
"""
import os
os.chdir(r'C:\Users\User\Desktop\GEOGRAPHIE Unif\Master 2\MEMOIRE\5 CODE_test\Valid_04')
import numpy as np
#import time
import laspy as lp
from sklearn.metrics import confusion_matrix, cohen_kappa_score
#import matplotlib.pyplot as plt
#import matplotlib as mpl

#%%
""" ###########   FONCTIONS   ########### """

""" Fonction pour écrire des résultats sous .las """

def laswrite (numpyPC, filename): #exemple de filename "lasfile.las"
    t = lp.file.File(filename, mode="w", header=cloud_ok.header)
    try:
        if (len(numpyPC)>0):
            t.x= numpyPC[:,0] # numpyPC = numpy array à enregistrer,
            t.y= numpyPC[:,1] #les colonnes "[:,1]" correspondent aux attributs à spécifier
            t.z= numpyPC[:,2]
            t.intensity= numpyPC[:,3]
            t.flag_byte= numpyPC[:,4]
            t.raw_classification= numpyPC[:,5]
            t.scan_angle_rank= numpyPC[:,6]
            t.user_data=numpyPC[:,7]
            t.pt_src_id= numpyPC[:,8]
            t.gps_time= numpyPC[:,9]
            t.Red= numpyPC[:,10]
            t.Green= numpyPC[:,11]
            t.Blue= numpyPC[:,12]
            return "Las file saved!"
        else:
            return "Numpy point cloud empty!"
    finally: t.close()

def to_npcloud (cloud):
    x=cloud.x
    y=cloud.y
    z=cloud.z
    I=cloud.intensity
    f=cloud.flag_byte
    rc= cloud.raw_classification
    sar=cloud.scan_angle_rank
    ud=cloud.user_data
    ID=cloud.pt_src_id
    GPS=cloud.gps_time
    R=cloud.Red
    G=cloud.green
    B=cloud.blue

    return (np.array([x,y,z,I,f,rc,sar,ud,ID,GPS,R,G,B])).T

def assign_tag(cloud, tag):
    i=0
    while i<len(cloud):
        cloud[i,5]=tag
        i+=1
    return ("Tous les points du nuage ont le tag de classification %s " % (tag))

#%%
""" ########### Importation des nuages & tags de classification ########### """

### A) Importation
cloud_ok= lp.file.File('T_O4_PC_total.las')
PC_total=to_npcloud(cloud_ok)

cloud= lp.file.File('O4_remains.las')
remains=to_npcloud(cloud)
assign_tag(remains,1)

cloud= lp.file.File('O4_Sol-Plafond.las')
SP=to_npcloud(cloud)
assign_tag(SP,2)

cloud= lp.file.File('O4_Wall.las')
Mur=to_npcloud(cloud)
assign_tag(Mur,3)

cloud= lp.file.File('O4_Beam.las')
Poutre=to_npcloud(cloud)
assign_tag(Poutre,4)

cloud= lp.file.File('O4_Embedded.las')
ME=to_npcloud(cloud)
assign_tag(ME,5)

cloud= lp.file.File('O4_Door.las')
Porte=to_npcloud(cloud)
assign_tag(Porte,6)

cloud= lp.file.File('O4_Window.las')
Fenetre=to_npcloud(cloud)
assign_tag(Fenetre,7)

"""  ########### Création de la vérité terrain totale ###########  """

print('A :Création de la vérité terrain totale')
print(" ")
#cloud= lp.file.File('O3_Verite.las')
#Verite=to_npcloud(cloud)
Verite=np.vstack((remains, SP, Mur, Poutre, ME,Porte,Fenetre ))
laswrite(Verite,"O4_Verite.las")


""" ########### Création de listes XYZ et de la liste des tags de classification ########### """
print('B: Création de listes XYZ et de la liste des tags de classification')
print(" ")
PC_total_XYZ=list()
PC_total_t=list()
for i in PC_total: # Multiplication des coordonnées afin de les considérer comme des entiers ce qui sera plus facile pour le tri
    j=(i[0]*1000000,i[1]*1000000,i[2]*1000000) #1=*10000 ; 2=*100000; 3=*1000000; 4=*10000000
    k=i[5]
    PC_total_XYZ.append(j) # liste des coordonnées XYZ des points
    PC_total_t.append(k) # liste des tags des points
```

```

Verite_XYZ=list()
Verite_t=list()
for i in Verite:
    j=(i[0]*1000000,i[1]*1000000,i[2]*1000000)
    k=i[5]
    Verite_XYZ.append(j) # Liste des coordonnées XYZ des points
    Verite_t.append(k) # Liste des tags des points

""" ########## Création de matrices (arrays) structurés ########## """
print('C: Création de matrices (arrays) structurés')
print(" ")
dtp=np.dtype({'names': ['x', 'y', 'z'], 'formats': [int,int,int]})
# On spécifie le nom des colonnes et le type de données présentes dans celles-ci (nécessaire pour le tri)
# On converti aussi nos listes en matrices
PC_total_XYZ=np.array(PC_total_XYZ, dtype=dtp )
PC_total_t=np.array(PC_total_t)
Verite_XYZ=np.array(Verite_XYZ, dtype=dtp)
Verite_t=np.array(Verite_t)


""" ########## Index de classification ########## """
print('D: Index de classification')
print(" ")
sortIndex_PC=np.argsort(PC_total_XYZ, order=['x','y','z']) # Trie en fonction de l'ordre fournis via les noms de colonnes
sortIndex_Ve=np.argsort(Verite_XYZ, order=['x','y','z'])


""" ########## Classification des matrices de tags ########## """
print('E: Classification des matrices de tags')
print(" ")
PC_total_t_sort=list()
test = list()
for i in sortIndex_PC: # Crée une matrice reprenant les tags de classification des points du nuages, trié selon l'index déterminé
    PC_total_t_sort.append(PC_total_t[i])
    test.append(PC_total_XYZ[i])

Verite_t_sort=list() #495363
test2=list()
for i in sortIndex_Ve: # Crée une matrice reprenant les tags de classification des points du nuages, trié selon l'index déterminé
    Verite_t_sort.append(Verite_t[i])
    test2.append(Verite_XYZ[i])

PC_total_t_sort=np.asarray(PC_total_t_sort) # Conversion des listes en matrices
Verite_t_sort=np.asarray(Verite_t_sort)


""" ########## Matrice de confusion ########## """
print('F: Matrice de confusion')
print(" ")
labels=[1,2,3,4,5,6,7]
classes=['Remains', 'Sol-Plafond', 'Murs','Poutres', 'Meubles Enc.', 'Portes', 'Fenêtres' ]

Matrice3 = confusion_matrix(Verite_t_sort, PC_total_t_sort ) # Calcul de la matrice de confusion

## Taux commission
Commission=list()
P_utilisateur=list() # Précision utilisateur (non utilisée)
i=0
while (i<len(Matrice3)):
    Commission.append((sum(Matrice3[:,i]) - Matrice3[i][i])/sum(Matrice3[:,i]))
    P_utilisateur.append(1-Commission[i])
    i+=1

## Taux Omission
Omission=list()
P_producteur=list() # Précision producteur (non utilisée)
i=0
while (i<len(Matrice3)):
    Omission.append((sum(Matrice3[i]) - Matrice3[i][i])/sum(Matrice3[i]))
    P_producteur.append(1-Omission[i])
    i+=1

## Exactitude Globale
n=len(PC_total) # Taille du nuage
i=0
sum_diag=0
while (i<len(Matrice3)):
    sum_diag+=Matrice3[i][i]
    i+=1
Global_exactitude= sum_diag/n

""" ########## KAPPA Statistiques ########## """
print('G: KAPPA Statistiques')
print(" ")
## Kappa Total
Kappa=cohen_kappa_score(Verite_t_sort, PC_total_t_sort)

## Kappa par classe
Kappa_cond=list()
i=0
while (i<len(Matrice3)):
    D=Matrice3[i][i] #Diagonale
    TC=sum(Matrice3[:,i]) #Total Colonne
    TL=sum(Matrice3[i]) #Total Ligne
    K= ((n*D) - (TC*TL)) / ((n*TL) - (TC*TL))
    Kappa_cond.append(K)
    i+=1

""" ########## Resumé des statistiques ########## """
print('H: Resumé des statistiques')
print(" ")
## Statistiques générale
resume=list()
resume.append(("Exactitude Globale", Global_exactitude))
resume.append(("Cohen's Kappa", Kappa))
resume=np.asarray(resume)


## Statistiques par classe
resume_classe = list()
i=0
while (i<len(Matrice3)):
    resume_classe.append((classes[i],round(Omission[i],5),round(Commission[i],5),round(Kappa_cond[i],5)))
    # On pourrait ajouter " round(P_producteur[i],5), round(P_utilisateur[i],5), "
    i+=1
resume_classe=np.asarray(resume_classe)


```

Figure 52 Code de création des matrices et de calcul des statistiques

IX.3. Résultats du géoréférencement

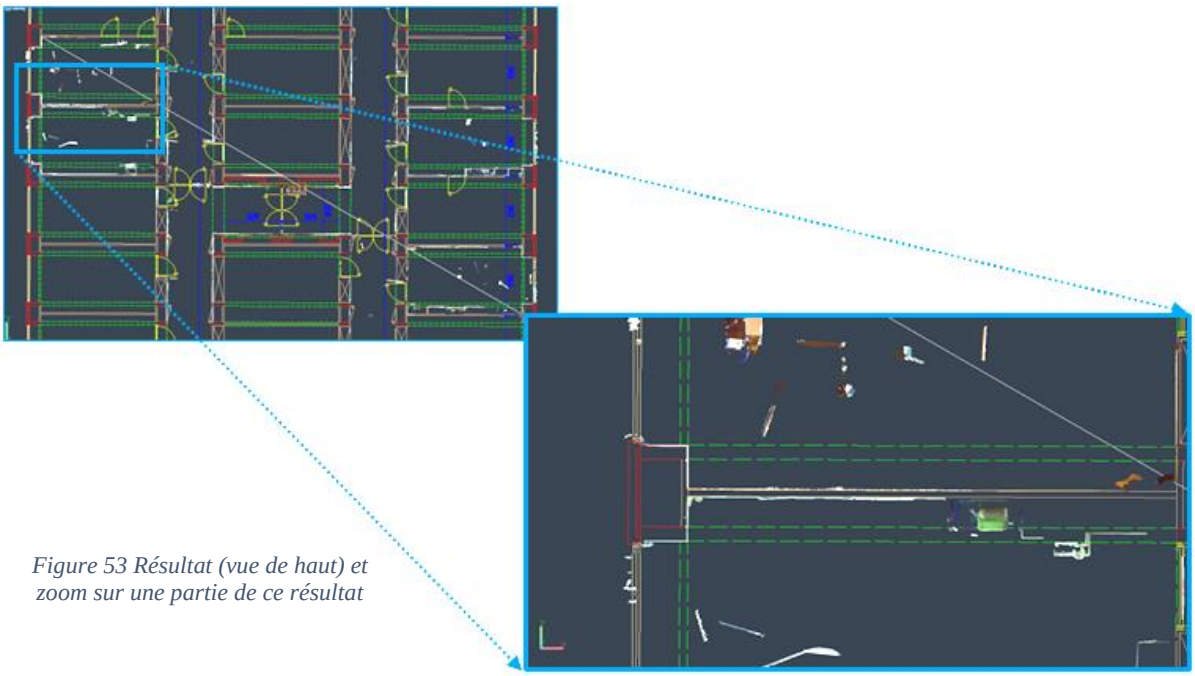


Figure 53 Résultat (vue de haut) et zoom sur une partie de ce résultat

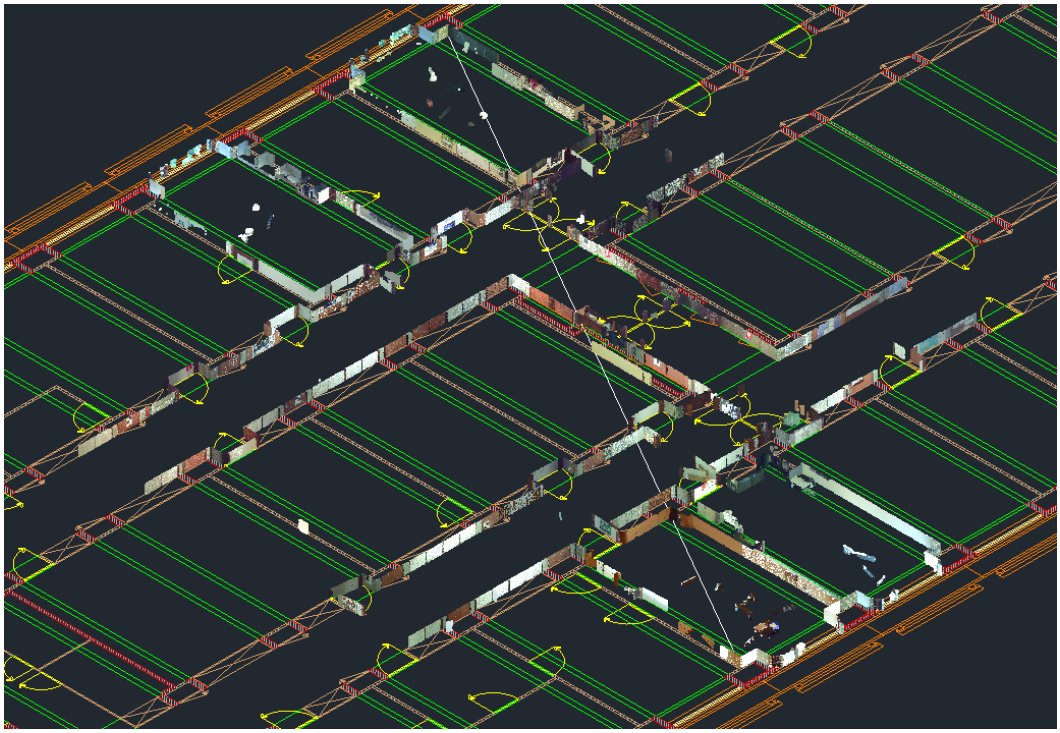


Figure 54 Résultat total du géoréférencement en vue 3D

IX.4. Nuages de base et plans mis en forme

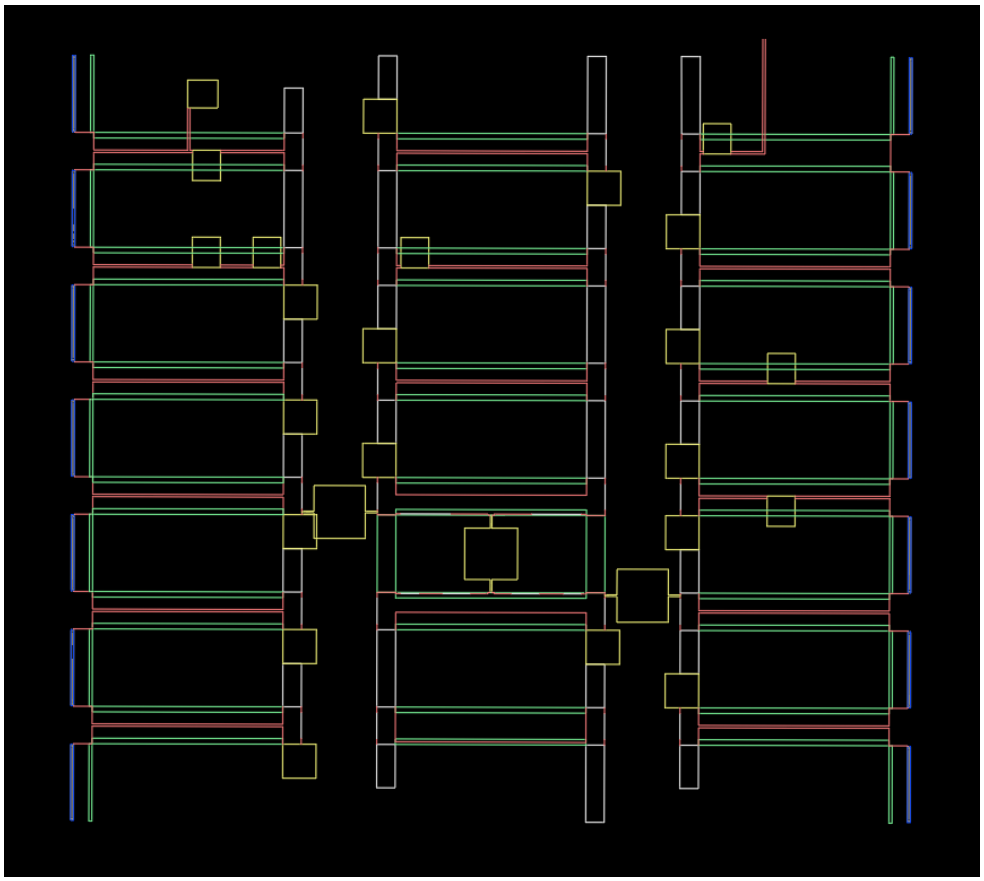


Figure 55 Résultat général de la mise en forme de l'ensemble du plan

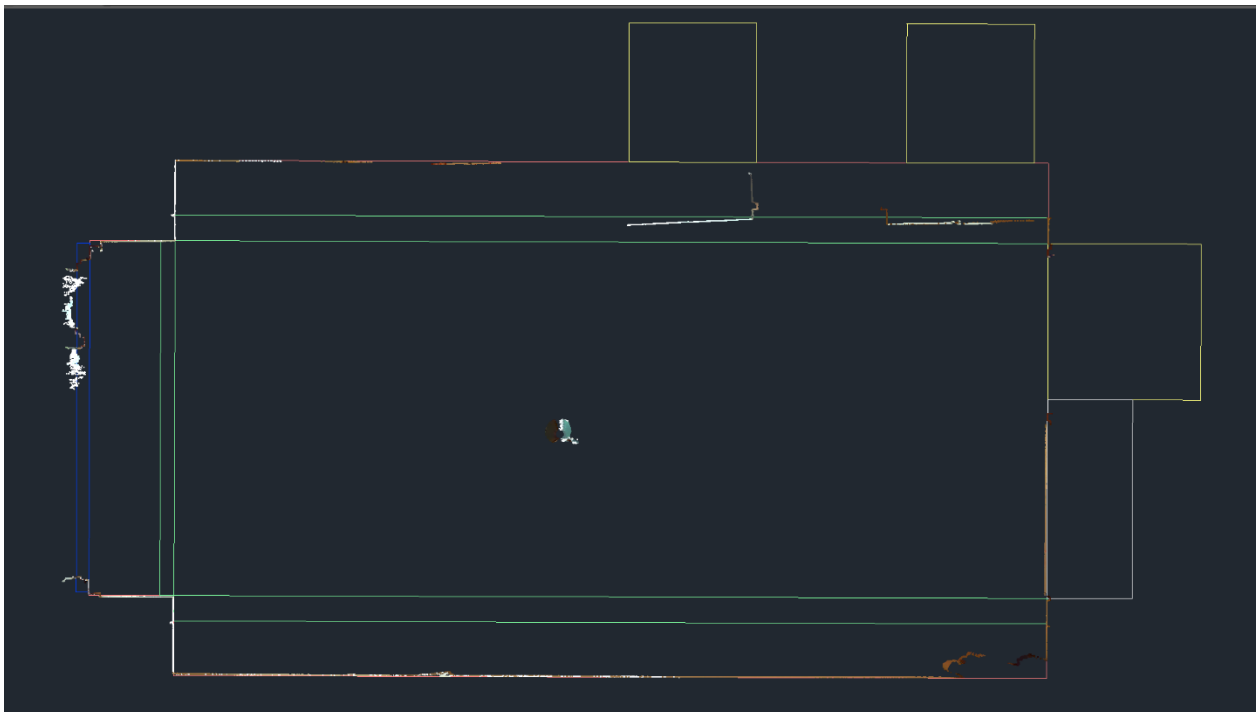


Figure 56 Sous-plan du bureau "Office_1", extrait depuis le résultat général de la mise en forme du plan

La coupe à 1,3m du nuage « Office_1 » est ajoutée par-dessus pour une meilleure visualisation

Figure 57 Nuage "Office_2", vue totale (à gauche) et vue tronquée d'une partie (à droite).

Figure 58 Plan mis en forme associé à "Office_2"

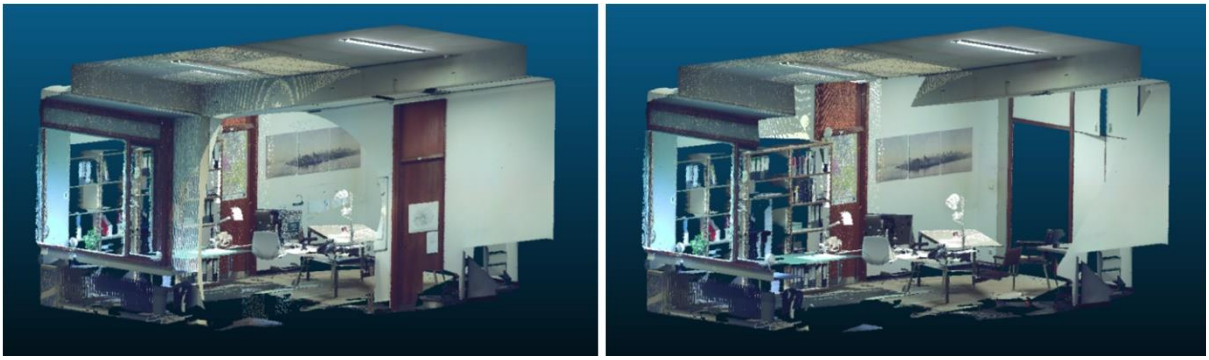


Figure 59 Nuage "Office_3", vue totale (à gauche) et vue tronquée d'une partie (à droite).

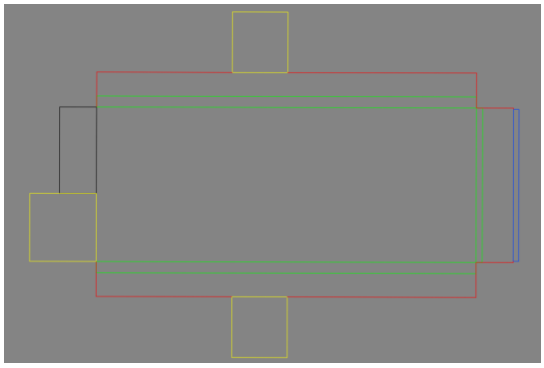


Figure 60 Plan mis en forme associé à "Office_3"

Figure 61 Nuage "Office_4", vue totale (à gauche) et vue tronquée d'une partie (à droite).

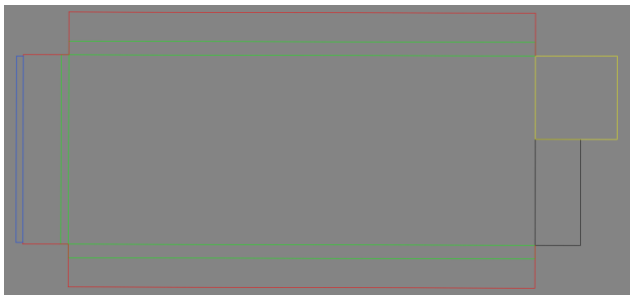


Figure 62 Plan mis en forme associé à "Office_4"

IX.5. Vérités Terrains

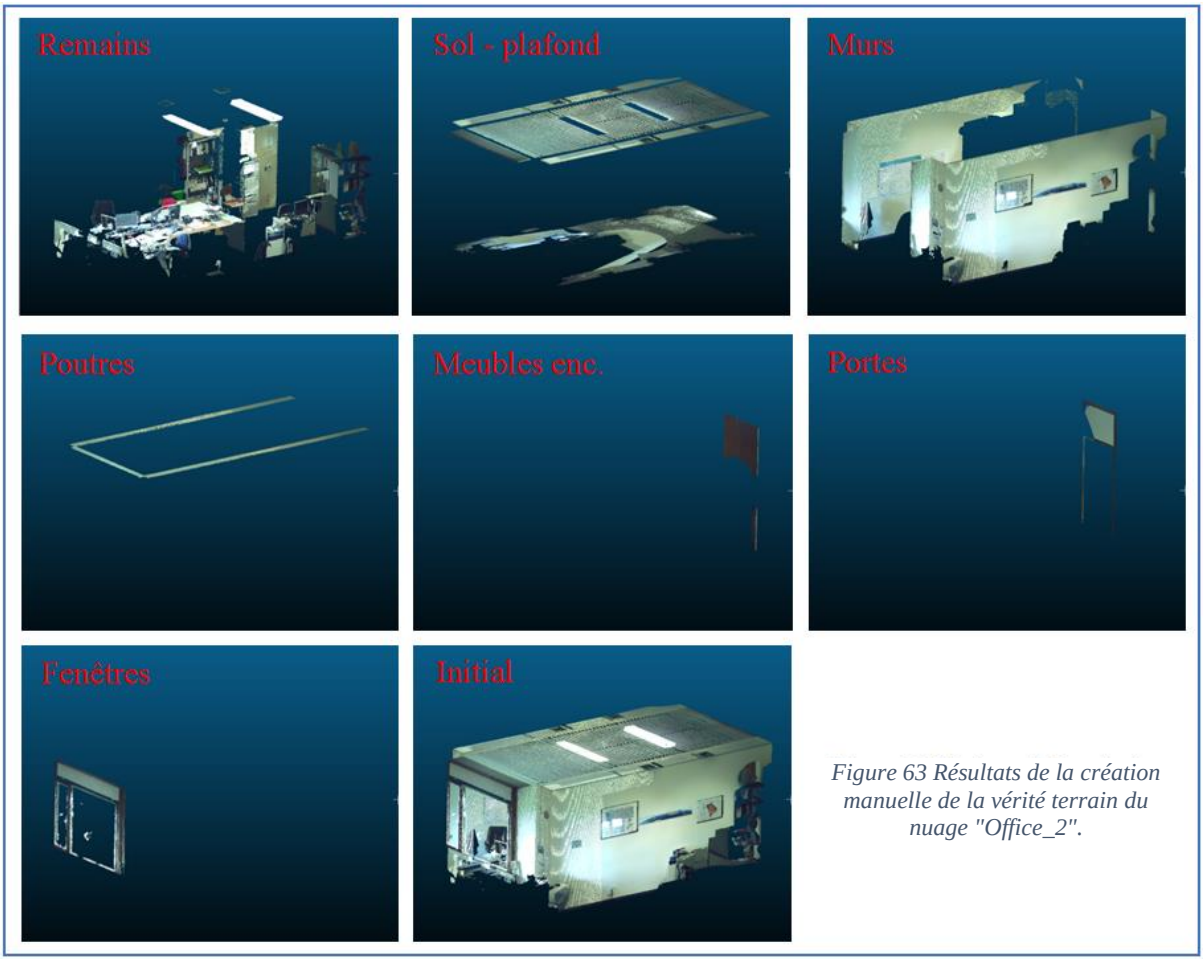


Figure 63 Résultats de la création manuelle de la vérité terrain du nuage "Office_2".

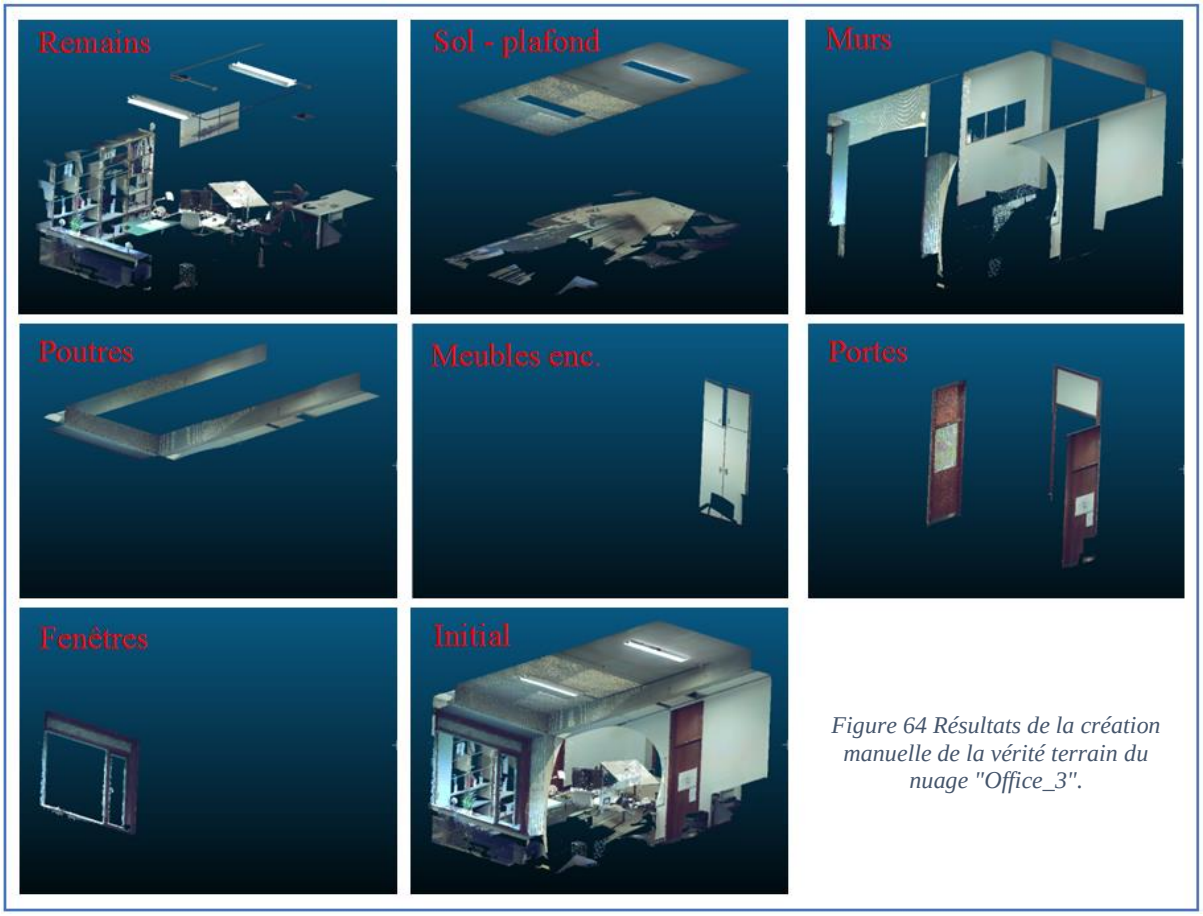


Figure 64 Résultats de la création manuelle de la vérité terrain du nuage "Office_3".

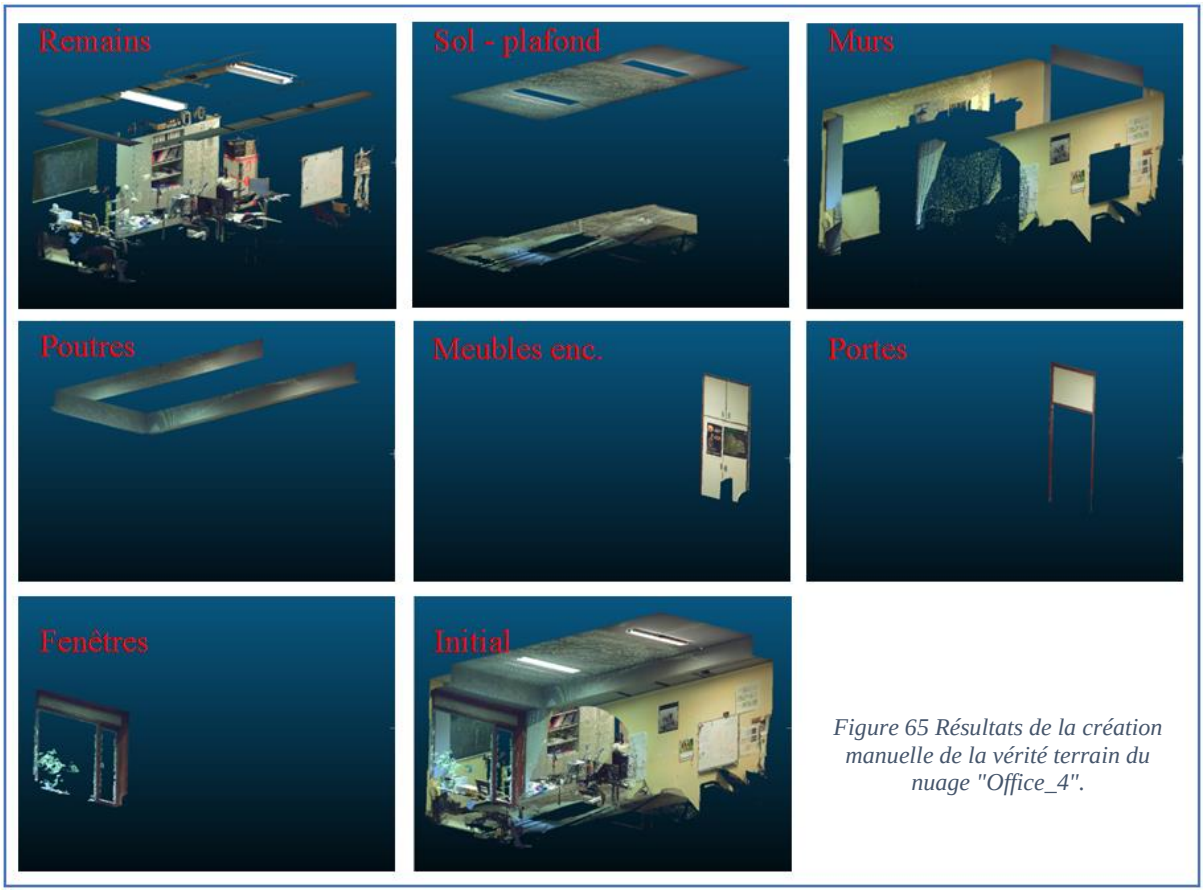


Figure 65 Résultats de la création manuelle de la vérité terrain du nuage "Office_4".

IX.6. Indexation: Octree et KD tree

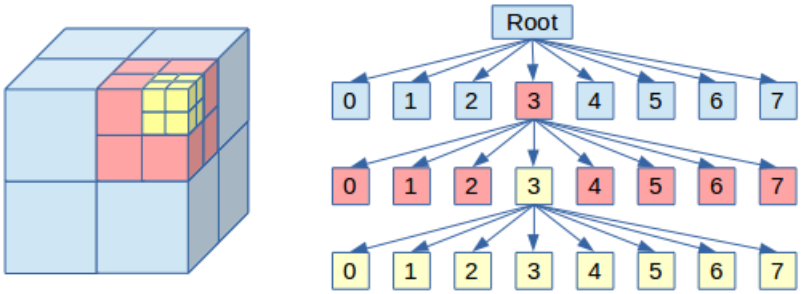


Figure 66 Indexation par Octree et construction de l'arbre associé (Geier, 2014)

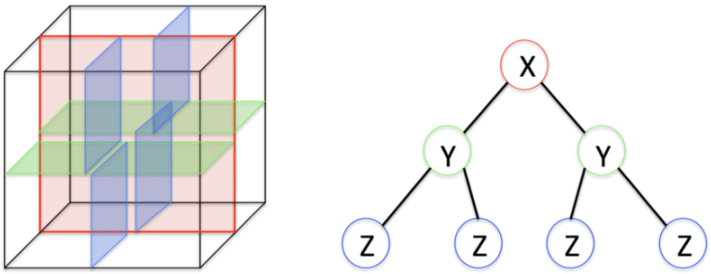


Figure 67 Indexation par KD tree et construction de l'arbre associé (Chen et al., 2016)

IX.7. Résultats finaux, validation et analyse

«Office_1»:

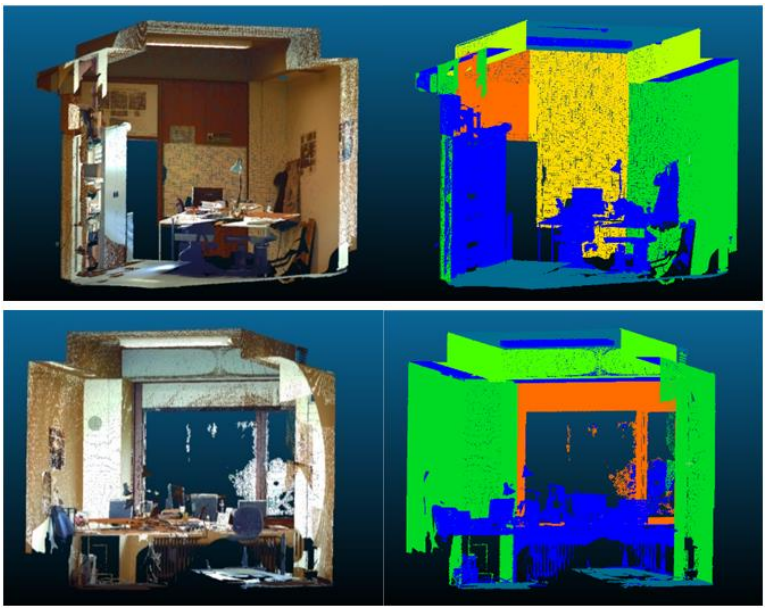


Figure 68 Résultat de la segmentation finale de "Office_1", divisé en deux (Vue intérieure).

« Office_2 »

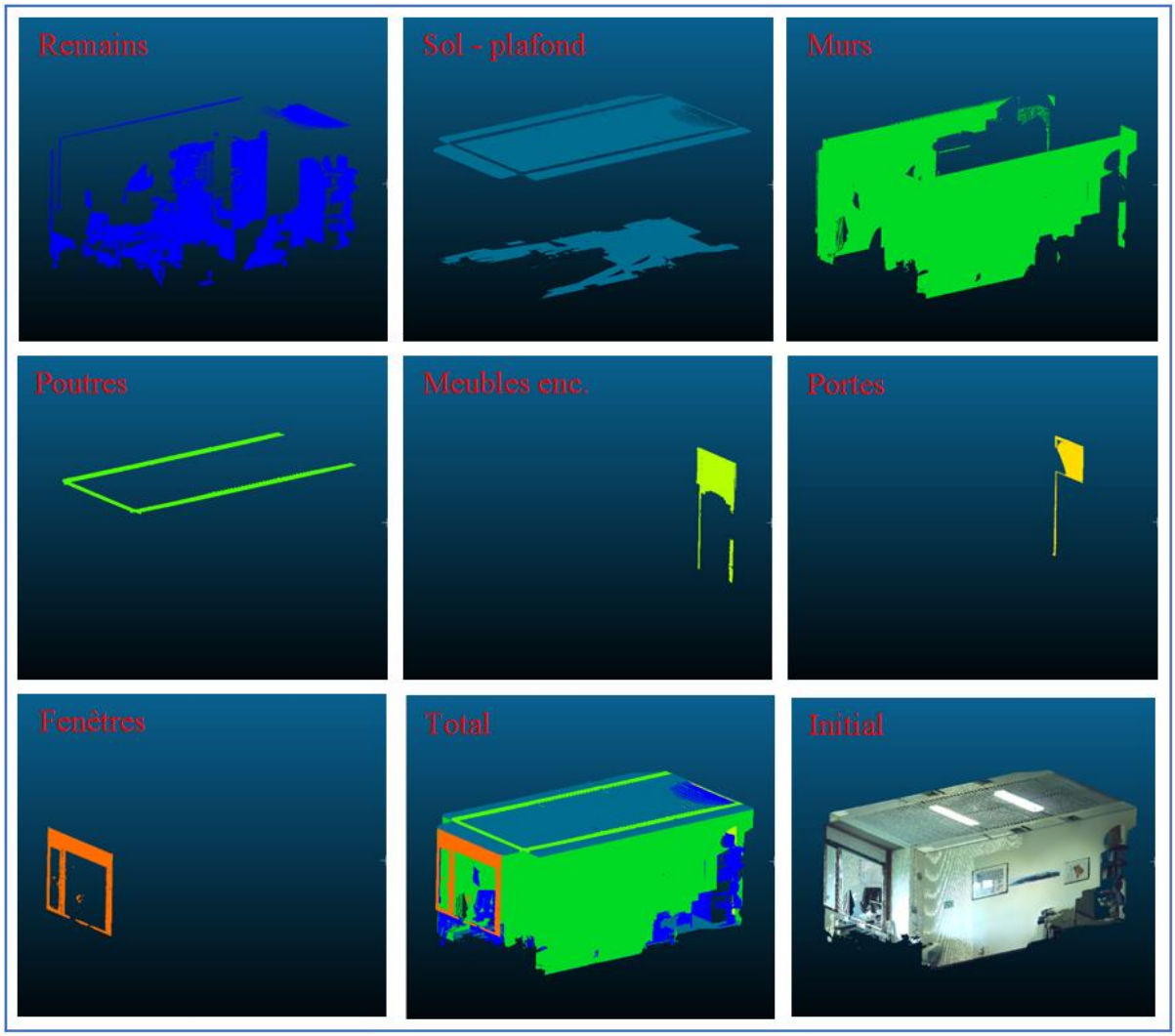


Figure 69 Résultats de la segmentation finale (V8) du nuage "Office_2" (non échantillonné).

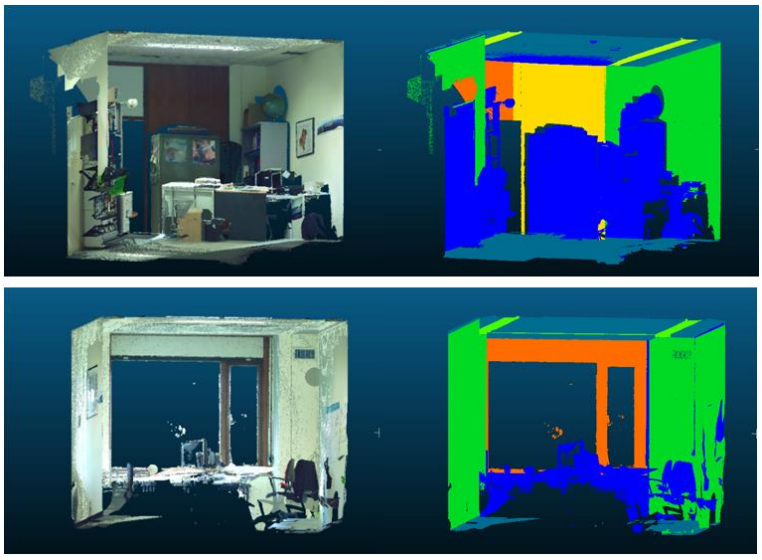


Figure 70 Résultat de la segmentation finale de "Office_2", divisé en deux (Vue intérieure).

« Office_3 » :

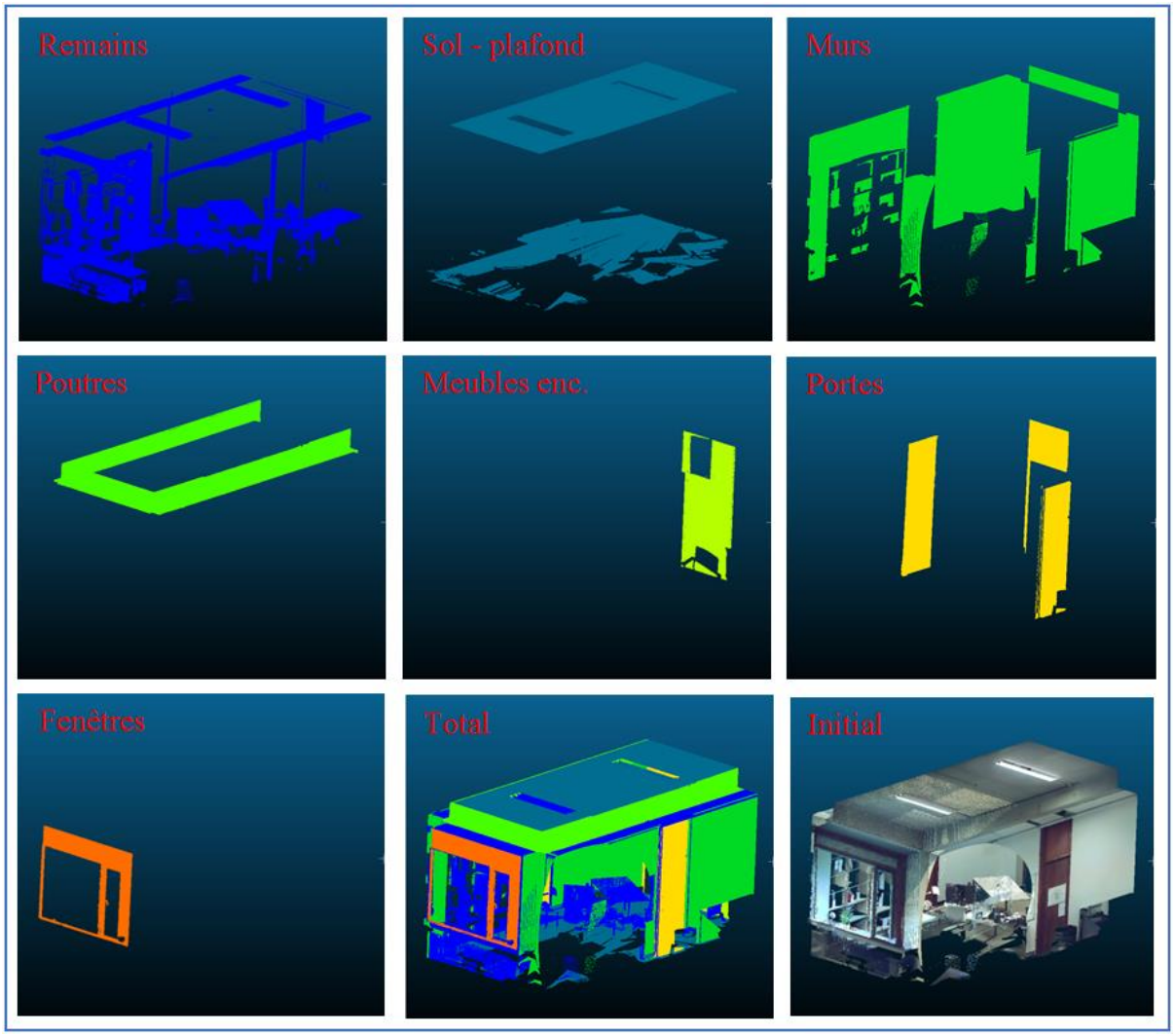


Figure 71 Résultats de la segmentation finale (V8) du nuage "Office_3" (non échantillonné).

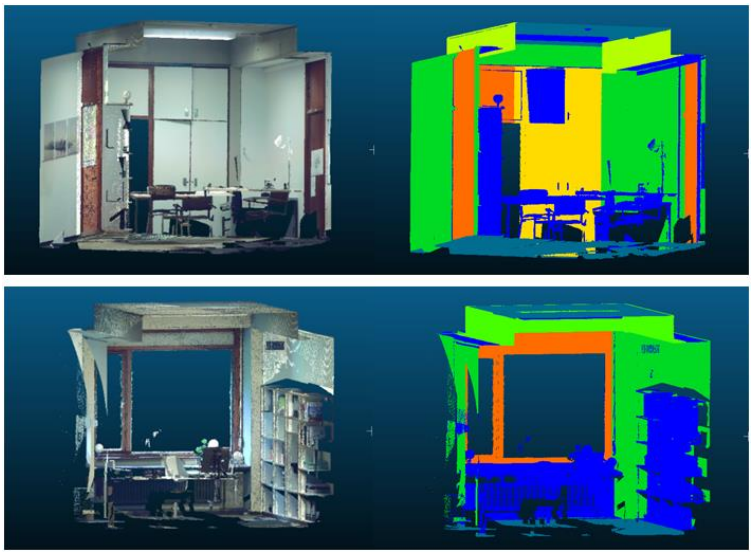


Figure 72 Résultat de la segmentation finale de "Office_3", divisé en deux (Vue intérieure).

« Office_4 » :

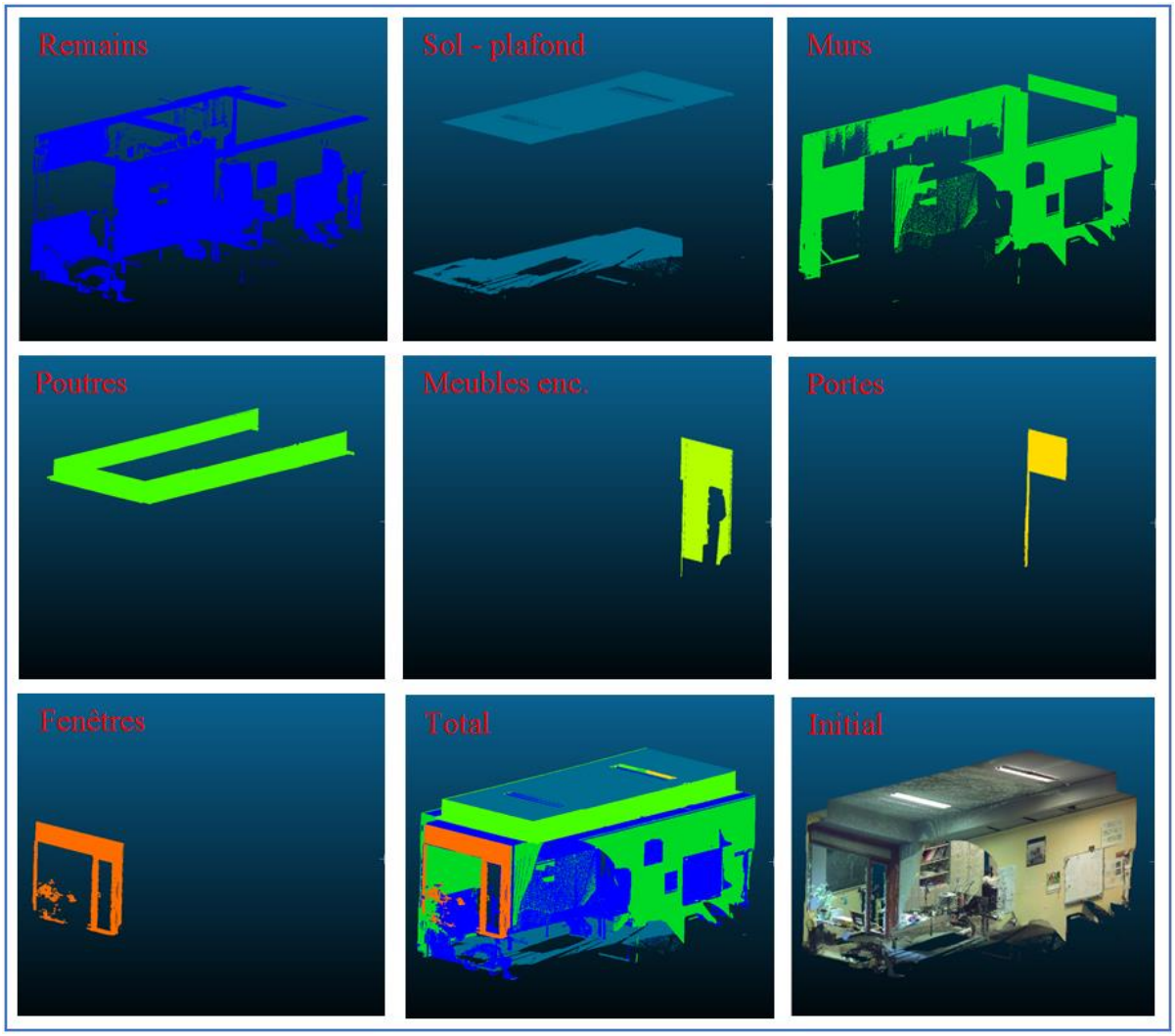


Figure 73 Résultats de la segmentation finale (V8) du nuage "Office_4" (non échantillonné).

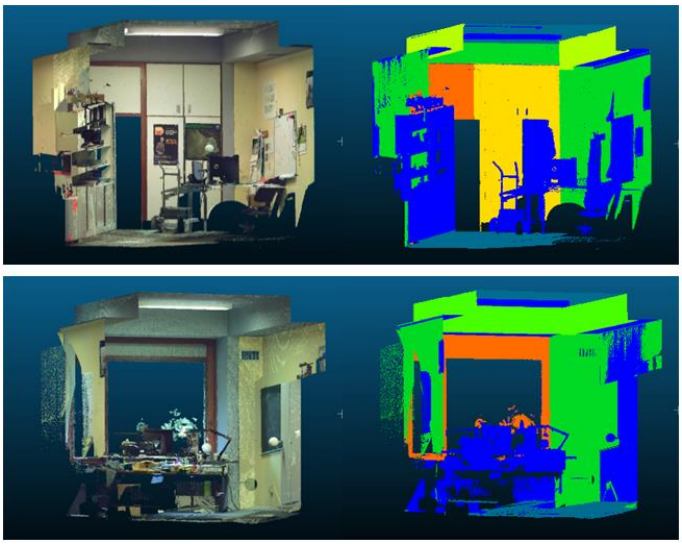


Figure 74 Résultat de la segmentation finale de "Office_4", divisé en deux (Vue intérieure).

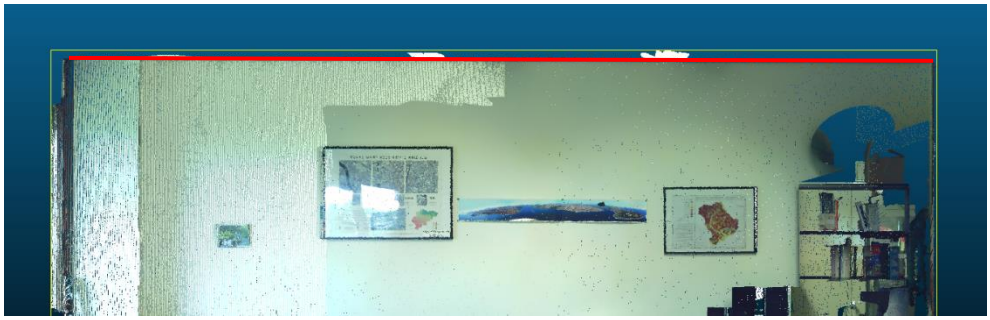


Figure 75 Problème d'inclinaison du faux plafond dans le nuage "Office_2"