Optimisation du traitement de nuage de points pour la production de plan de façade au sein d'un cabinet de géomètre-expert

Maxime Marchand

le cnam
École supérieure des géomètres et topographes esgt

CONSERVATOIRE NATIONAL DES ARTS ET MÉTIERS

ÉCOLE SUPÉRIEURE DES GÉOMÈTRES ET TOPOGRAPHES

MÉMOIRE

présenté en vue d'obtenir

le DIPLÔME D'INGÉNIEUR CNAM

SPÉCIALITÉ : Géomètre et Topographe

par

Maxime MARCHAND

Optimisation du traitement de nuage de points pour la production de plan de façade au sein d'un cabinet de géomètre-expert

Soutenu le 4 juillet 2018

JURY

PRÉSIDENT :	**M. Laurent MOREL**	**Président du jury**
MEMBRES :	**M. Arnaud RASS**	**Maître de stage**
	M. Franck FERRET	**Maître de stage**
	M. Mathieu BONNEFOND	**Examinateur**
	Mme Elisabeth SIMONETTO	**Professeur référent**

Remerciements

Je tiens tout d'abord à remercier Monsieur Daniel Legrand de m'avoir accueilli au sein de son entreprise. Mes remerciements vont également à Messieurs Arnaud Rass et Franck Ferret, mes maîtres de stage, pour avoir su m'orienter et m'avoir permis d'effectuer ce stage dans d'excellentes conditions.

Je remercie Madame Elisabeth Simonetto, ma professeur référente, pour son aide et ses remarques constructives concernant le plan de travail.

Un grand merci à Monsieur Nicolas Landrivon de la société Technodigit pour m'avoir accordé une licence étudiante du logiciel 3DReshaper le temps de mes recherches.

Je souhaite particulièrement remercier Monsieur Jean Charles Herbin pour sa relecture attentive et ses conseils avisés sur la forme de ce mémoire.

Enfin, je remercie toutes les personnes qui ont pu m'aider dans la réalisation de ce travail de fin d'études.

Glossaire

3D : abréviation de « trois dimensions ». Il s'agit d'un système tridimensionnel permettant de se situer dans l'espace. Dans ce mémoire, nous allons parler de « point 3D », qui est un point connu en coordonnées X, Y et Z.

BIM : acronyme de « Building Information Modeling », le BIM représente l'ensemble des méthodes de travail mises en place autour de la maquette numérique d'un bâtiment. L'objectif étant de mettre en place un même langage de travail pour l'ensemble des acteurs intervenant sur le bâtiment.

CAO : acronyme de « Conception assistée par ordinateur », elle permet la modélisation d'objets en deux ou trois dimensions.

Grand Paris : nom donné au projet de transformation de l'agglomération parisienne visant à corriger le déséquilibre concernant les réseaux de transport en commun entre Paris et sa banlieue.

LASER : acronyme de « Light amplification by stimulated emission of radiation », il s'agit d'un rayonnement lumineux émis par un appareil.

LIDAR : acronyme de « Laser (ou Light) detection and ranging », il s'agit d'une technique d'acquisition de données, basée sur l'analyse du faisceau lumineux renvoyé par l'objet.

Livrable : dans ce mémoire, nous appellerons « livrable » tout résultat d'un travail qui traduit l'achèvement d'un projet et qui peut être fourni au commanditaire.

MNT : acronyme de « Modèle Numérique de Terrain », il s'agit dans un espace en trois dimensions, de tous les points appartenant au terrain naturel.

Projection : « On appelle projection d'un point sur un plan le pied de la perpendiculaire abaissée de ce point sur le plan. »[1]

Scan : dans ce mémoire, nous appellerons « scan », l'ensemble des points acquis depuis une position précise.

.dwg : acronyme de « DraWinG », il s'agit du format natif des dessins provenant du logiciel AutoCAD. Il s'agit d'un des formats les plus fréquemment utilisés en matière de dessin CAO.

1 C. Briot dans « Éléments de géométrie (Application). ». Édition de 1856 – page 1.

Table des matières

Introduction

A l'heure de l'essor des grands projets de construction tel que le Grand Paris ou du BIM (Building Information Modeling), la place occupée par les systèmes d'acquisitions tridimensionnelles est extrêmement importante. Cela est principalement dû au développement de la technologie LIDAR dont le rapport quantité de données par la vitesse d'acquisition reste jusqu'à aujourd'hui inégalé. C'est dans ce cadre, où chaque composant d'un bâtiment tend à être référencé et modélisé, que de nombreux géomètres ont su se positionner en faisant valoir leur savoir faire dans le domaine de l'acquisition en trois dimensions de l'espace. En effet, ce procédé permet de produire, à partir d'un unique élément qu'est le nuage de points, de multiples produits livrables (se référer à la figure 3 page 8).

Si la littérature est d'accord sur le fait que de nos jours la phase d'acquisition de données 3D sur le terrain est maîtrisée, un adjectif revient sans cesse pour qualifier la phase de traitement des données : chronophage. En effet, de nombreux facteurs tels que la quantité importante de mesures, le bruit dans les données ou encore les zones d'ombres rendent le travail d'interprétation et de traitement long et fastidieux. C'est dans la perspective d'améliorer cette étape que de nombreux scientifiques travaillent depuis plusieurs années sur le développement de méthodes de traitement automatique, fiables et efficaces. Parmi ces recherches nous pouvons citer, entre autres, la détection de points bruités, l'extraction automatique du sol, la reconnaissance de formes géométriques ou d'objets urbains...

Ce mémoire a pour but d'identifier différentes méthodes de traitements et processus pouvant être mis en place au sein d'un cabinet de géomètre-expert afin de réduire le temps de traitement et d'automatiser certaines actions. Nous nous intéresserons particulièrement à la création de plan de façades. Ce type de plan d'élévation est important lors de projets de construction ou de rénovation et répond à des normes spécifiques que nous évoquerons en détail dans la partie I.2.1. C'est au sein de la SELAS Daniel Legrand que ce travail a été réalisé. Il s'agit d'un des plus anciens cabinets de géomètres parisiens qui compte aujourd'hui plus de 80 collaborateurs. Disposant d'un département de traitement 3D, la société se présente comme « un des premiers cabinets à investir dans le BIM » et est en recherche constante d'innovation. Cette démarche s'inscrit ainsi dans la méthode de gestion de la qualité appelée roue de Deming.

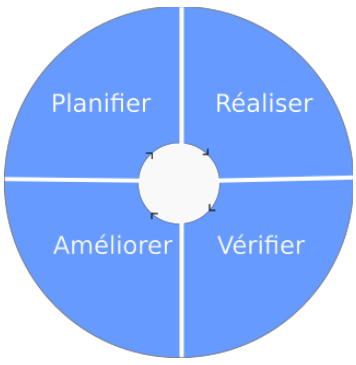


Illustration 1: Représentation de la roue de Deming

Lors de la réalisation d'un plan de façade, deux méthodes d'acquisitions sont répandues. La première est le relevé de points ponctuels par le biais de tachéomètres équipés d'une visée laser. Les

points mesurés sont la plupart du temps des points d'encadrements d'ouvertures, situés sur des arêtes ou des arcs. Ils sont tous issus de la décision de l'opérateur qui manipule l'appareil et chacun représente une information sur le bâtiment qui sera exploitée lors de la phase de dessin. Une façade relativement simple, comportant un niveau de détail moyen, peut contenir quelques centaines de points mesurés. La deuxième méthode est celle qui va nous intéresser dans ce travail. Elle consiste à relever la façade du bâtiment à l'aide d'un scanner laser. Dans ce cas, l'opérateur ne choisit pas les points à mesurer, puisque tous les points de l'environnement sont acquis par l'appareil en respectant une certaine densité. Le résultat obtenu est un nuage de points pouvant contenir plusieurs millions d'entités.

Dans le cadre de l'élaboration d'un plan « à la main », c'est-à-dire où le dessinateur va tracer à l'aide d'un logiciel de CAO toutes les lignes caractéristiques de la façade en s'appuyant sur le nuage de points situé en arrière plan, plus de 99% de points contenus dans le nuage seront inutiles. En revanche, cette information va se révéler vitale lors des procédés de segmentation, de détection, d'extraction...

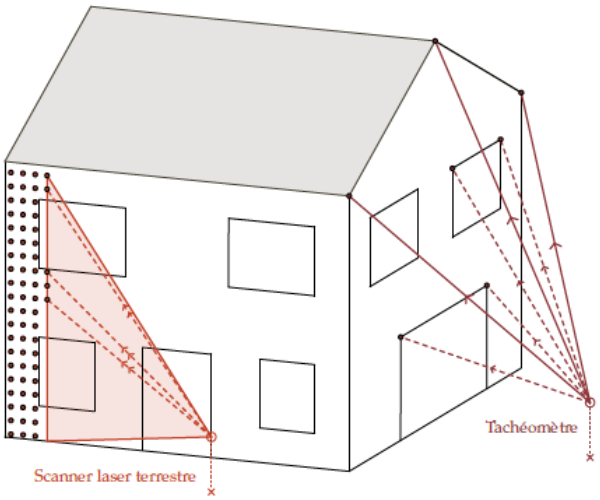


Illustration 2: Acquisition de façades au scanner laser et au tachéomètre sans réflecteur (Hélène Macher, 2017)

En effet, nous pouvons analyser l'organisation des points en trois dimensions et tenter une représentation du nuage par des primitives géométriques. L'intensité du laser réfléchi par une surface renferme également une information sur le point acquis et peut dans certains cas, permettre des prétraitements automatiques du nuage. A la suite de l'étape de consolidation du nuage[2], les données obtenues sont une multitude de points unifiés au sein d'un même ensemble. La décomposition de cet ensemble en primitives géométriques a fait l'objet de nombreuses études de la part de la communauté scientifique et est présentée comme une étape préliminaire fiable dans le cadre des traitements automatiques.

Pour un cabinet de géomètre, automatiser certains traitements par le biais d'algorithmes peut être très efficace. En matière de gain de temps, cela serait avantageux pour la réalisation d'opérations communes à de nombreux dossiers. Une homogénéité dans la précision des résultats pourra également être observée entre les différents dossiers traités car le même traitement mathématique

2 Étape non développée dans ce mémoire.

sera appliqué tandis que lorsqu'un opérateur effectue le traitement manuellement, des erreurs de pointé sont susceptibles d'être produites.

Ce travail de fin d'étude est structuré en trois parties. Dans la première partie, nous présenterons le contexte de ce mémoire, qui est composé d'une part des méthodes employées à ce jour par l'entreprise pour l'acquisition des données et pour la création de livrables, et d'autre part d'un résumé des recherches scientifiques pouvant être utilisé pour nos recherches. Nous introduirons également en fin de partie la présentation des données sur lesquelles se sont effectués les différents tests. Dans la seconde partie, nous mettrons en avant les étapes mises en place pour réaliser l'extraction de contours remarquables sur différents types de façades, afin de faciliter la création de plan 2D sous le logiciel de CAO Autocad. Nous mettrons en application ces méthodes sur nos données et nous évaluerons les résultats obtenus.

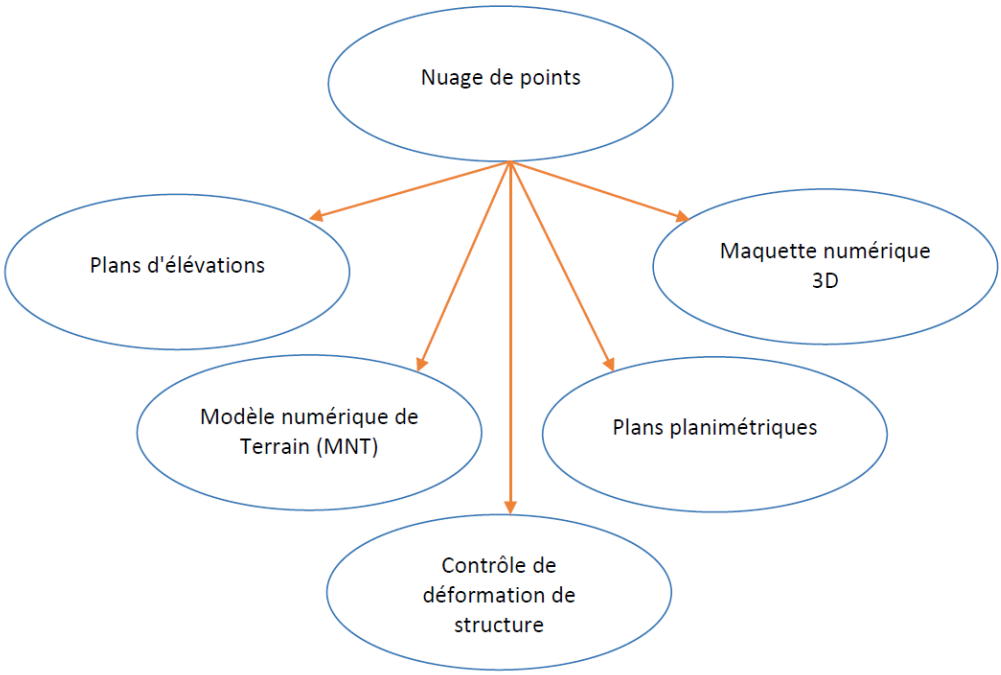


Illustration 3: Exemples de produits réalisables à partir d'un nuage de points

I. Contexte

I.1. Problématique et objectifs

La SELAS Daniel Legrand a bien conscience des enjeux et opportunités apportés par les lasers scanners terrestres et a su s'adapter à cet outil relativement récent. Si elle reconnaît la performance des appareils sur le terrain, la phase de traitement a mis en évidence certaines problématiques, aussi bien en ce qui concerne le temps de traitement que la capacité de stockage des nuages de points. Autre problématique liée aux logiciels de traitements, le prix, souvent très élevé, peut freiner l'investissement d'une entreprise dans des solutions logiciels si elle n'est pas sûre qu'elle pourra exploiter les outils du programme au maximum pour valoriser ses données.

Dans son histoire, Paris à connu de nombreuses tendances architecturales, chacune étant associée à une temporalité bien précise. C'est un mélange de ces genres que l'on retrouve aujourd'hui en levant la tête dans les rues de la capitale. A chaque époque son architecture et à chaque architecture son niveau de complexité. Ainsi, la représentation d'une façade haussmannienne (1850 - 1900) va demander souvent plus de travail que celle de l'époque de la Restauration (1815 - 1830). Même observation entre le style Art déco en 1920 avec le style des années Trente. Si le temps requis pour le dessin de la façade dépend du niveau de complexité de sa structure, il en est de même pour le traitement automatique du nuage de points. En effet, l'apparition de détails tels que des bandeaux horizontaux pour séparer les étages, des moulures, des statues limitent les possibilités d'automatisation.

L'objectif concernant la production de plan de façade sera dans un premier temps la détection, l'extraction et la vectorisation de contours d'éléments structurels de divers bâtiments et leur importation dans le logiciel de CAO Autocad. Pour ce faire, une étape de segmentation du nuage doit être réalisée. C'est pourquoi nous nous intéresserons à un type d'algorithme en particulier et nous étudierons sa déclinaison sous plusieurs supports. En effet, des outils de segmentation existent actuellement sur le marché et nous nous intéresserons au rapport qualité / prix de chacun. Nous développerons nous-même certains outils adaptés à notre problématique. Si nous travaillerons sur un nombre limité de façades, l'objectif de ce mémoire est de proposer une méthodologie qui soit transposable pour tous les édifices mesurés par l'entreprise.

A l'heure de la multiplication des logiciels et plates-formes en ligne de traitement, il est important de se poser la question du format des fichiers de nuage de points. En effet, il existe de nombreux formats de lecture, et certains ne peuvent pas être traités par tous les logiciels que nous utiliserons. C'est pourquoi nous essayerons de proposer les formats à privilégier pour nos traitements.

Une des fonctions de ce mémoire est également de réaliser une première approche des méthodes de traitement automatique appliquées à des nuages de points en trois dimensions. Un conclusion sur la pertinence de cette nouvelle façon de traiter les données sera présentée à la fin de cette étude et plusieurs perspectives d'application seront envisagées.

I.2. Processus d'acquisition et de traitement des données au sein du cabinet

I.2.1. Types de rendus client

Nous présenterons ici les documents qui peuvent être produits par le cabinet à partir d'un relevé au scanner laser. Il s'agit pour la plupart de levés d'architecture sur du bâti existant pour des travaux de rénovation ou d'agrandissement. Les documents produits sont les suivants[3]:

→ Plans des intérieurs

→ Plans des façades

→ Plans des toitures

→ Plans de coupes, avec ou sans projection de l'arrière-plan

→ Maquettes numériques en trois dimensions

→ Nuages de points 3D et visualisation des images panoramiques avec Leica Truview ou Autodesk Recap

Ces documents sont des rendus de base à partir desquels nous allons pouvoir créer d'autres produits tels que :

→ Plans de division et plans de vente

→ Calculs de superficies

→ Plans de profils en long et en travers

→ Création de Modèle Numérique de Terrain (MNT) et calculs de cubatures

Souvent, lors d'interventions sur des chantiers de constructions ou de rénovations, le cabinet va livrer le nuage de points assemblés en tant que produit intermédiaire. Cela a pour but de permettre au client de commencer à se projeter et de vérifier si les grandes contraintes du projet peuvent être respectées. Bien qu'il ne constitue pas une finalité en soi, ce rendu est intéressant dans la mesure où il peut être livré peu de temps après l'intervention sur le terrain.

Lorsque le client demande une maquette numérique, le cabinet va également fournir une visualisation panoramique du nuage de points 3D colorisé. Le Truview permet de réaliser directement des mesures et des annotations sur des photos-réalistes du site considéré.

Dans notre cas, nous allons traiter des plans de façades. En réalité, il s'agit d'un abus de langage car nous réalisons davantage des plans d'élévations. Un plan de façade, est une pièce obligatoire à fournir pour l'obtention d'un permis de construire. Les éléments qu'il contient sont très précisément décrits par la notice CERFA n° 51434 05. Cette dernière stipule que le plan doit :

→ Contenir une échelle

→ Présentation de l'aspect général des façades et des toitures

→ La répartition des matériaux et leurs aspects

→ Les portes, les fenêtres, les cheminées...

3 Ces informations sont consultables sur le site internet du cabinet : www.dlegrand.com

Il n'est pas précisé que le plan doive contenir l'élévation de certains points caractéristiques et puisqu'il s'agit d'un document visant à obtenir un permis de construire, cela signifie que ledit édifice n'est qu'un projet et que par conséquent le géomètre ne peut intervenir sur le terrain. Le seul cas où il est question de mesurer un bâtiment existant lors d'un plan de façade est encadré par l'article R*431-10 a) du code de l'urbanisme qui dispose que « lorsque le projet a pour effet de modifier les façades ou les toitures d'un bâtiment existant, ce plan fait apparaître l'état initial et l'état futur ».

Le cabinet intervient principalement pour la création de plans d'élévations sur des bâtiments déjà construits en vue de leurs rénovations ou modifications. Les clients ont, dans ce cas, besoin de connaître l'élévation de certains points afin de projeter correctement leurs projets. Un exemple de plan d'élévation est présenté en Annexe 1. Il s'agit du modèle qualité appliqué au sein du cabinet, à l'imitation duquel chaque plan doit tendre dans sa forme.

I.2.2. Méthodologie terrain

Il est rare qu'un chantier ne nécessite qu'une seule station. Il faut dans la plupart des cas acquérir les données depuis plusieurs endroits. Les différents scans doivent alors être assemblés dans un système de projection cartographique lors de la phase de traitement. Mais il existe différentes méthodes de géoréférencement et il faut savoir laquelle va être utilisée puisque chacune requiert des opérations spécifiques sur le terrain. Landes *et al* (2011) indiquent qu'il existe deux approches différentes pour obtenir un nuage de points géoréférencé : une méthode de géoréférencement direct et une méthode de géoréférencement indirect.

La méthode de géoréférencement direct nécessite de positionner le scanner directement sur une station connue en coordonnées. De cette manière chaque point de chaque scan est directement calculé dans le système de projection choisi. L'approche par géoréférencement indirect comporte deux sous-méthodes : la première utilise une consolidation basée sur les cibles tandis que la deuxième utilise une consolidation basée sur les points caractéristiques des nuages. Dans chacun des cas, les cibles et les points caractéristiques sont mesurés à l'aide d'un levé topographique. L'ensemble de ces étapes est résumé dans l'illustration 4.

Les relevés effectués dans le cadre de ce mémoire reposent sur la méthode de géoréférencement indirect avec une consolidation basée sur les cibles. Il est difficile de classer ces différentes méthodes pour en privilégier une plutôt qu'une autre. En effet, chacune présente des avantages et des inconvénients. Par exemple, en géoréférencement direct et indirect (avec consolidation par cibles) , la présence de zones de recouvrements n'est pas obligatoire. A l'inverse, une consolidation basée sur les points caractéristiques présente l'avantage d'un gain de temps sur le terrain, puisqu'il n'est pas nécessaire de scanner chacune des cibles.

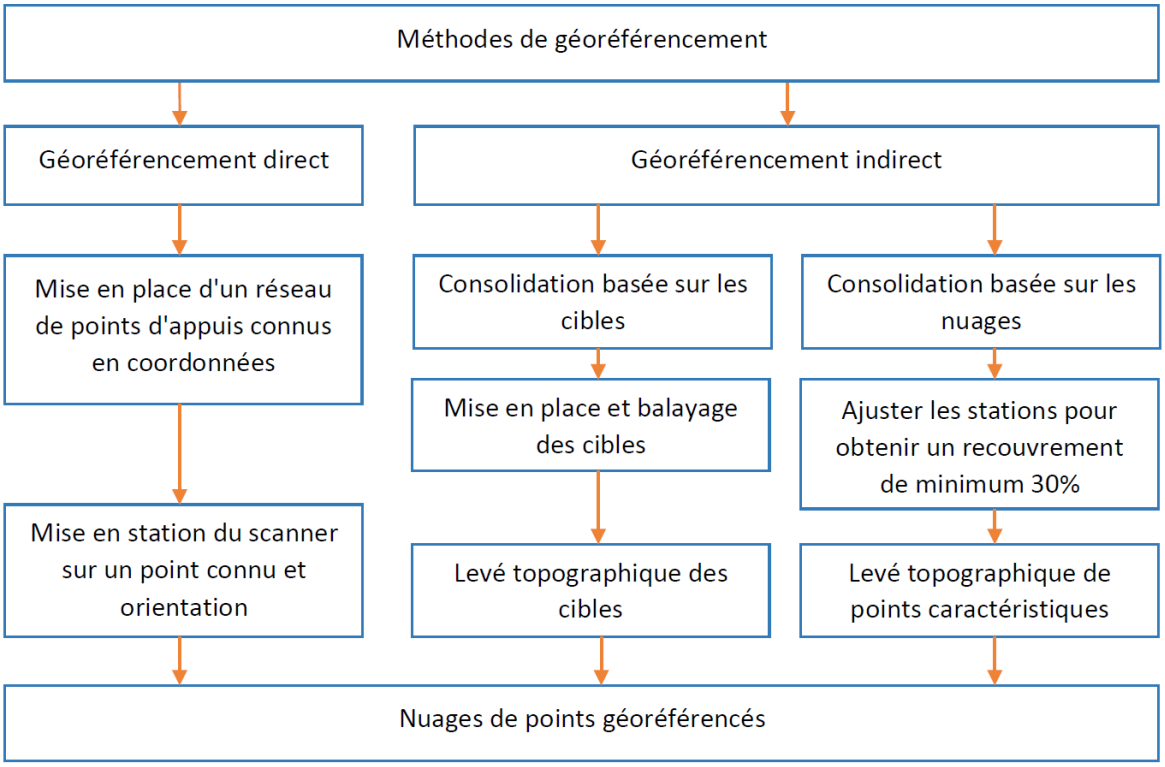


Illustration 4: Description schématique des différentes méthodes de géoréférencement (inspiré de Landes et al. (2011) et Lerma Garcia et al. (2008))

Lorsque l'on a recours aux cibles, il y a des conditions à respecter pour un résultat optimal lors de la consolidation. Une question sujette à débat est le nombre minimum de cibles qui doivent être mesurées par une station. Barber *et al* (2003) énonce qu'un minimum de 4 cibles doivent être visé par le scanner lorsque ce dernier n'est pas positionné sur un point de coordonnées connues. Garcia *et al* (2008) soutiennent ce choix et l'expliquent par le fait que la sur-détermination des relations géométriques entre les données permet de minimiser les erreurs en effectuant une optimisation par moindres carrés.

Il peut arriver dans certains cas que l'on effectue une consolidation basée sur les nuages. Dans ce cas, il faut s'assurer d'un recouvrement d'au minimum 30% entre les deux nuages. C'est généralement la méthode ICP (Iterative Closed Point) qui est choisie pour réaliser la consolidation. Cette méthode analyse les distances entre les points des deux nuages et estime la meilleure transformation possible, c'est-à-dire la transformation comportant la plus faible erreur (Lerma Garcia *et al*, 2008). Il y a alors un nuage dit « modèle » qui va être considéré comme fixe et un nuage dit « observé » qui subit les transformations pour s'aligner avec le nuage modèle (Boulaassal, 2010).

Les deux détenteurs du brevet de cette méthode, Paul J. Besl et Neil D. McKay, ont mis au point la formule suivante (1.1) qui permet de calculer les paramètres de translation et de rotation à appliquer au nuage observé.

Soit: → f la fonction à minimiser
→ R la matrice de rotation

→ T le vecteur de translation
→ N_p le nombre de points du nuage le moins dense situés dans la zone de recouvrement
→ x_i un point du nuage de référence
→ p_i son point homologue dans le nuage à consolider

Alors on peut écrire : $f(R,T)=\frac{1}{N_p}\times\sum_{i=1}^{i=N_p}(x_i-R\times p_i-T)^2$

Ce système est ensuite linéarisé avant d'être résolu par la méthode des moindres carrés. On ré-applique cette méthode jusqu'à ce que les distances obtenues soient inférieures à un certain palier (Landes *et al*, 2011)(Besl *et al*, 1992).

La figure 5 décrit la méthodologie terrain employée au sein de la SELAS Daniel Legrand pour une consolidation basée sur les cibles.

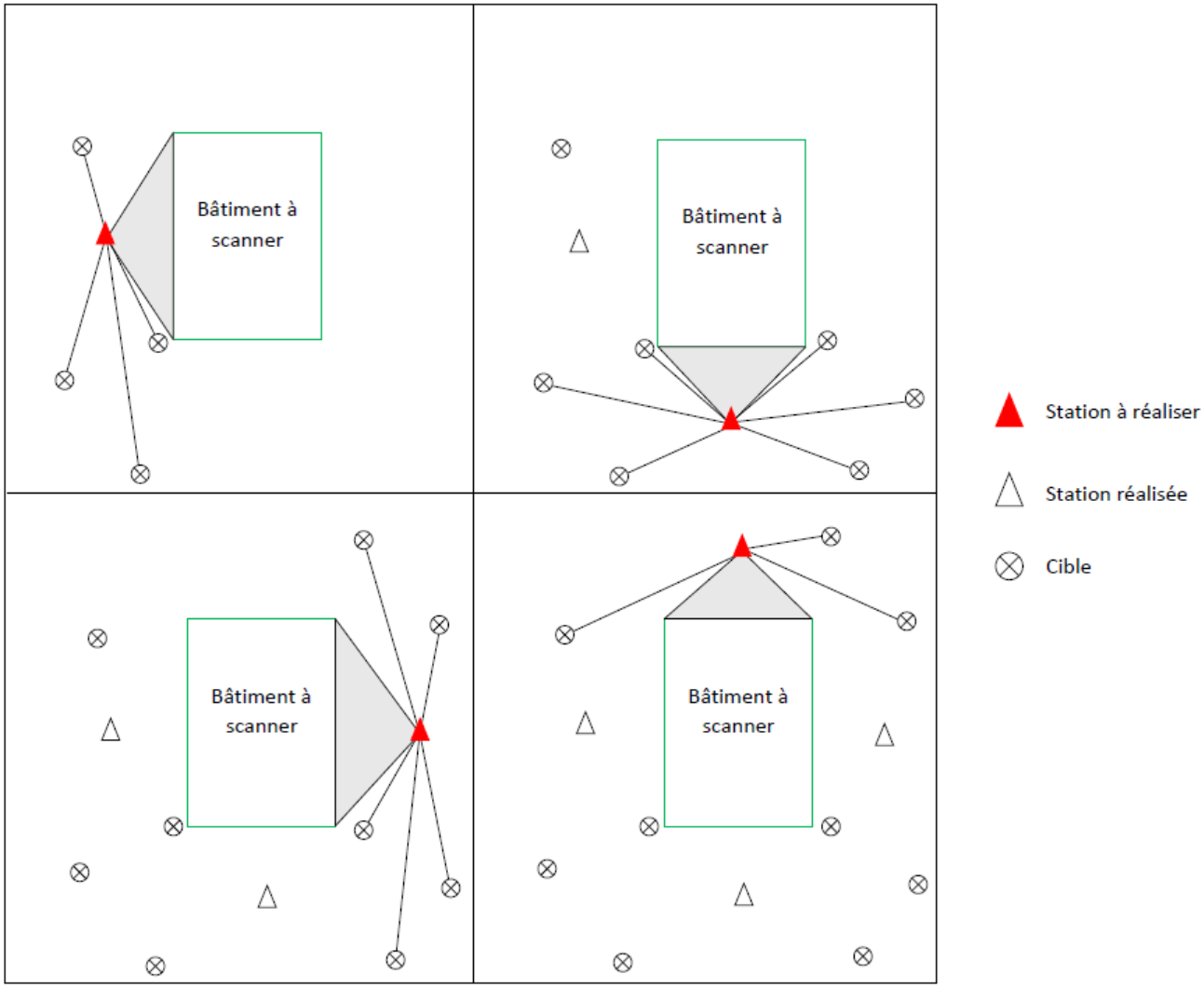


Illustration 5: Explication schématique de la méthode de levé scanner par consolidation cible – cible utilisée pour ce mémoire

I.2.3. Appareils et logiciels utilisés

La SELAS Daniel Legrand dispose de moyens importants aussi bien en ce qui concerne les appareils d'acquisition que les logiciels de traitements et de dessins. Pour mener à bien ses chantiers, la société dispose de 3 scanners laser terrestre Leica : 2 scanners P30 et 1 scanner P40. Sur le terrain, les scans sont reliés par des cibles circulaires Leica noire et blanche (référence GZT21). Pour la réalisation de polygonale, nous utilisons les modèles suivants de station totale : Leica Viva TS11, Leica Viva TS15 et Leica Viva TS16.

Au niveau du traitement des nuages de points, les logiciels utilisés au sein du cabinet sont les programmes Leica REGISTER et Leica PUBLISHER qui sont réunis dans le logiciel Cyclone. Le premier est utilisé pour la consolidation des nuages de points tandis que le second permet « la visualisation et l'annotation de nuages de points 3D sur des interfaces internet »[4]. Le logiciel Autodesk RECAP est également utilisé pour la conversion des données au format Autocad. La création des maquettes numériques est effectuée sous le logiciel Autodesk Revit.

Des plugins viennent s'ajouter aux logiciels existants afin d'améliorer la phase de traitement. Le principal outil concerné est Leica Cloudworx, qui offre une meilleure visualisation et manipulation du nuage de points sous Autocad et sous Revit. Ponctuellement, nous pourrons avoir recours au module 3D du logiciel COVADIS édité par la société Geomedia.

Dans le cadre du mémoire, nous avons décidé de nous ouvrir à de nouveaux logiciels offrant des outils de traitement intéressants. Au premier plan, le logiciel 3DReshaper propose de nombreuses fonctions intuitives parmi lesquelles un système de nettoyage et de segmentation automatique du nuage. Avoir un nuage de point le plus « propre » possible, c'est à dire éliminer le maximum d'artefacts, constitue une étape primordiale pour la détection de formes. Le logiciel propose également différentes méthodes de maillages, à partir desquelles nous allons pouvoir effectuer nos traitements de reconnaissance de contours. L'avantage majeur du logiciel est que la grande majorité des fonctions est encadrée par diverses options dont le paramétrage est laissé à l'utilisateur. De plus, l'export de données directement dans Autocad est réalisable grâce à l'outil Cloudworx, également présent sur 3DReshaper.

4 https://leica-geosystems.com

I.2.4. Chaîne de traitement des données au sein du cabinet

Une fois les données acquises par le scanner, nous procédons à leur import dans le logiciel Cyclone afin de procéder à l'assemblage des nuages de points. Il s'agit là de l'opération commune à tous les types de dossiers issus d'un relevé 3D. Cette étape, précédemment abordée, consiste à identifier des points homologues entre différents nuages. Ces points sont souvent représentés par des cibles, mais il peut également s'agir de point caractéristiques (Landes *et al*, 2011). Au cabinet, la majorité des consolidations est effectuée à l'aide de cibles plates. Le logiciel va calculer et extraire le centre de chaque cible en s'appuyant sur la géométrie et l'intensité du laser renvoyé. Il va ensuite effectuer une transformation des nuages à l'aide de rotations et de translations.

Pour la création de plan de façade, il faut alors exporter le nuage dans un format pris en charge par Autocad. A partir de là, pour obtenir un plan .dwg nous dessinons directement par transparence sur le nuage de points dans le logiciel de CAO. Le cabinet disposant d'un nombre important de dessinateurs, il est important de développer une charte graphique à respecter dans le but que tous les documents produits présentent les mêmes teintes de couleurs et les mêmes représentations graphiques. Un modèle qualité (cf. Annexe 1) a été créé et chaque plan doit tendre au même résultat. De même, chaque élément doit être rangé dans le calque qui lui correspond. La figure 6 présente une synthèse du processus de traitement des données au sein de la SELAS Daniel Legrand depuis la consolidation des nuages de points jusqu'à la création des différents produits.

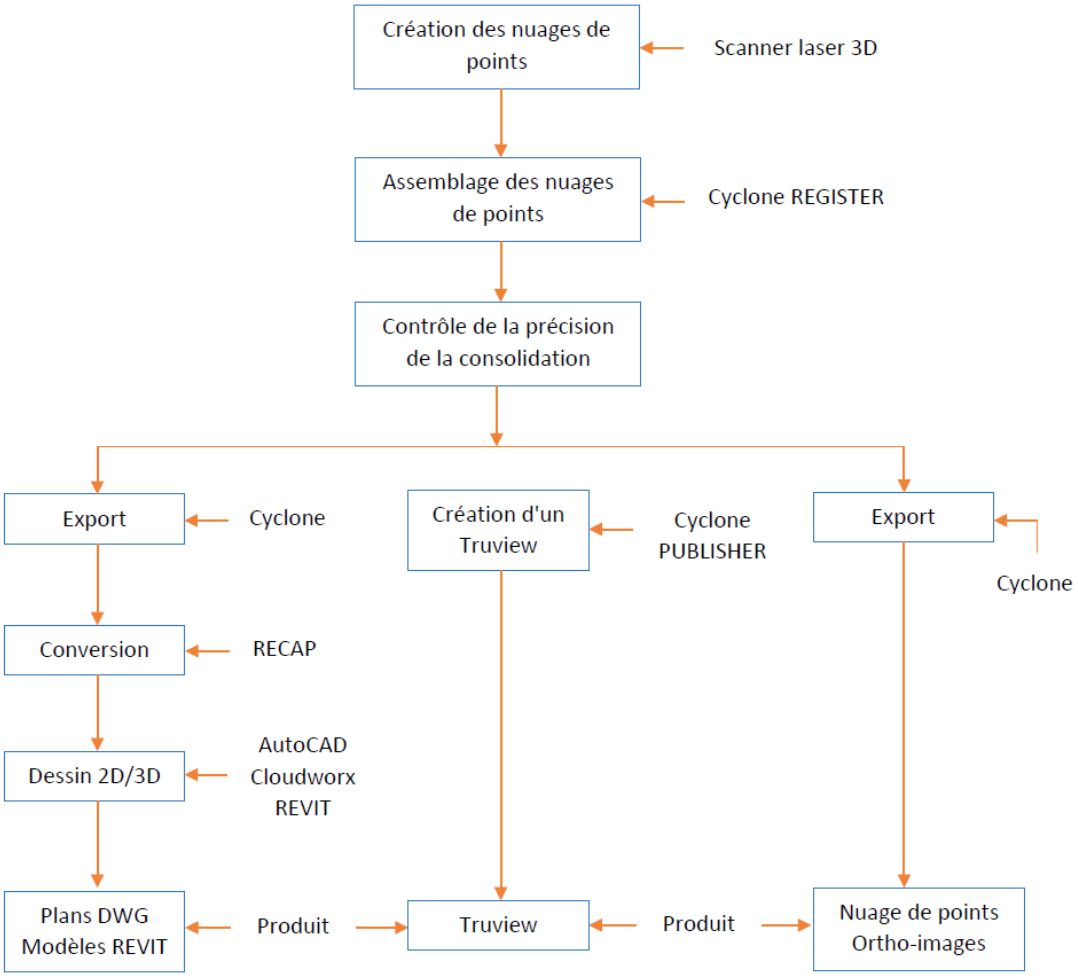


Illustration 6: Synthèse du processus de traitement des données au sein du cabinet

I.3. État de l'art

I.3.1. Segmentation des données

Cet état de l'art s'appuie sur celui de Boulaassal (2010).

I.3.1.1. Introduction

La segmentation constitue une étape incontournable lorsque l'on souhaite effectuer des traitements et des analyses sur un nuage de points. Landes *et al*. (2011) donnent une définition précise de ce processus : « La segmentation d'un nuage de points est un partage/subdivision de l'ensemble des points 3D en sous-ensembles (sous-nuage de points) homogènes, suivant des critères prédéfinis ». Il s'agit donc d'une étape visant à séparer différents éléments d'une scène afin de pouvoir les traiter individuellement. Les « critères prédéfinis » évoqués dans la définition sont des critères d'homogénéités qui dépendent de l'utilisateur. Il peut s'agir de l'intensité du laser renvoyé, mais également de la courbure ou de la planéité décrite par un ensemble de points (Boulaassal, 2010).

Il existe aujourd'hui un grand nombre de techniques de segmentation de données 3D. La majorité sont issues d'algorithmes de segmentation d'images (2D) qui ont été adaptés de façon à prendre en compte la position spatiale des points. Après une étude approfondie de la littérature, Boulaassal (2010) constate que l'ensemble des procédés de segmentation peut être classé dans deux grandes familles :

1) La segmentation basée sur le principe de fusion
2) La segmentation basée sur le principe de reconnaissance automatique de formes géométriques

Pour chaque catégorie, nous allons développer deux des méthodes les plus utilisées. Ainsi, nous expliquerons le principe de la segmentation par croissance de surface et par division-fusion pour illustrer la première famille, et nous décrirons le principe de segmentation par transformée de Hough et par l'algorithme RANSAC pour illustrer la seconde.

I.3.1.2. Segmentation par croissance de surface

Cette méthode repose sur le même principe mathématique que le principe de croissance de régions développé pour le traitement de données 2D. Ce procédé comporte deux étapes :

→ sélection d'un point de départ, également appelé point graine

→ accroissement de la surface autour du point graine

Cet accroissement est effectué uniquement si les conditions définies par l'utilisateur sont réunies. Concernant la segmentation de bâtiments, les critères d'homogénéités sont multiples. Vosselman *et al.* (2004) en énoncent trois principaux :

→ La proximité des points

→ L'appartenance à un plan local

→ Des vecteurs normaux proches

Pour modéliser en trois dimensions des façades de bâtiments d'anciennes Médina, Ait El Kadi *et al* (2014) ont utilisé un algorithme de croissance de région qui repose sur le critère de la similarité de l'intensité laser.

Cela signifie que pour qu'un point soit ajouté à la surface, il faut dans un premier temps que sa distance soit inférieure à une certaine valeur. Il faut également que ce point appartienne au plan local. Pour vérifier cette condition, un plan est créé à partir des points présents dans la région et nous calculons la distance orthogonale entre le point à ajouter et ce plan. Si elle est en-dessous d'un certain seuil, le point peut être ajouté à la surface. Enfin, nous calculons le vecteur normal du point à ajouter et nous le comparons au vecteur normal du point le plus proche appartenant à la surface. Si l'écart angulaire entre ces deux vecteurs est inférieur à un certain seuil, le point testé peut définitivement être ajouté à la surface.

La figure 7 illustre ces explications. Concernant les notations, P_1 et P_2 représentent deux points du nuages. Nous supposerons que P_1 appartient à la surface en cours de construction et que nous testons P_2. Dans ce cas, n_1 et n_2 correspondent aux normales en ces points, P_1' et P_2' correspondent aux point P_1 et P_2 projetés sur leur surface locale et r_{12} est la distance entre P_1' et P_2'.

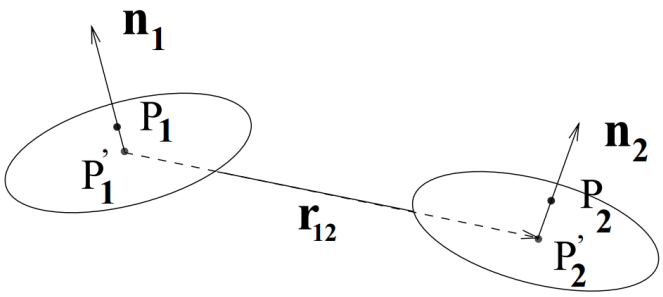


Illustration 7: Schéma explicatif de la mesure de condition de coplanarité (Boulaassal, 2010)(Stamos, 2001)

Les conditions mathématiques à respecter sont par conséquent (Boulaassal, 2010)(Stamos, 2001) :

$$\alpha = \arccos(\boldsymbol{n1.n2}) \leq t_{seuil}$$

$$d = max\ (|\boldsymbol{r}_{(12)}.\boldsymbol{n}_1|, |\boldsymbol{r}_{(12)}.\boldsymbol{n}_2|) \leq d_{seuil}$$

La partie qui diffère entre les multiples algorithmes existants est également la plus délicate. Il s'agit de la phase d'initialisation, c'est-à-dire du choix du point source autour duquel la surface va être construite. Ce choix peut être effectué de différentes manières. L'une d'elle est entièrement manuelle, c'est l'utilisateur qui va désigner le point source. Cette façon de faire à l'avantage d'être fiable, mais est chronophage. Dans son mémoire, Boucher (2016) utilise une sélection semi-aléatoire des points germes. Afin d'être certaine de la qualité de ses points, elle va effectuer une pré-analyse basée sur l'histogramme du nuage de points selon l'intensité laser afin de sélectionner un point source situé sur une pierre et non sur un joint.

Si les résultats d'une segmentation par croissance de régions dans la littérature semblent satisfaisants, des inconvénients doivent être soulignés. Un des plus importants est l'influence du bruit qui va venir fausser les résultats de l'algorithme en créant des zones de sous-segmentation ou de sur-segmentation comme illustré dans la figure 8.

Illustration 8: Exemple de sur-segmentation (Boulaassal, 2010)

I.3.1.3. Segmentation par division fusion

Cette méthode de segmentation repose sur la création d'une structure de données particulière appelée octree. Cette structure, souvent utilisée pour de l'indexation spatiale, est un cube que l'on va diviser successivement en sous-cube. De cette manière, un cube de base peut donner 8 sous-cubes, et chaque sous-cube peut à son tour se diviser en 8 « sous-sous-cube ». Chaque palier de division est un niveau. La figure 9 illustre ce principe pour un octree de niveau 2.

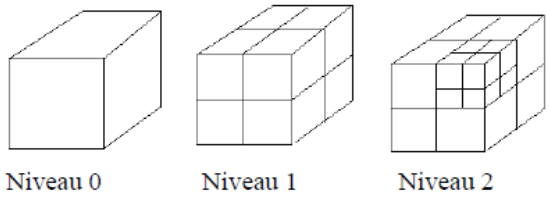


Illustration 9: Deux niveaux de subdivisions selon la structure octree (Boulaassal, 2010)

Le principe de l'algorithme est de tester un critère d'homogénéité dans chaque cube. Si ce critère n'est pas respecté, le cube se divise en 8 et on regarde si la condition est validée pour chacun des cubes. Ce processus s'effectue jusqu'à ce que tous les cubes remplissent les critères ou jusqu'à obtenir des cubes vides. Avec cette méthode, il est fréquent d'utiliser un critère de coplanarité entre les points présents dans un cube. Le résultat de la procédure de division est donc l'obtention de segments plans.

L'algorithme va calculer le meilleur plan possible à partir des points dans chaque sous cube en utilisant une estimation des moindres carrés, ce qui revient à minimiser la somme des carrés des distances entre les points et le plan extrait. Mathématiquement, un point M de coordonnées (x,y,z) appartient au plan P de paramètres (a,b,c,d) si et seulement si :

$$ax+by+cy+d=0$$

alors la distance *d* entre le point M et le meilleur plan P calculé par l'algorithme suit la relation :

$$d=\frac{|ax+by+cz+d|}{\sqrt{a^2+b^2+c^2}}$$

Une deuxième phase suit la division : la fusion. Pour cela, l'algorithme va tester la proximité ainsi que la coplanarité des segments plans voisins. Si les conditions sont respectées, les deux segments plans seront fusionnés. Ainsi, deux segments plans vont être fusionnés si l'écart entre leurs normales est inférieur à un seuil établi. Il faut également prendre en compte la position relative des sous-cubes les uns par rapport aux autres. En effet, deux segments plans peuvent avoir des normales présentant un écart angulaire inférieur au seuil, mais se situer l'un au dessus de l'autre. Dans ce cas, la fusion serait incohérente. La figure 10 illustre différents cas dans lesquelles il faut procéder à une fusion ou non.

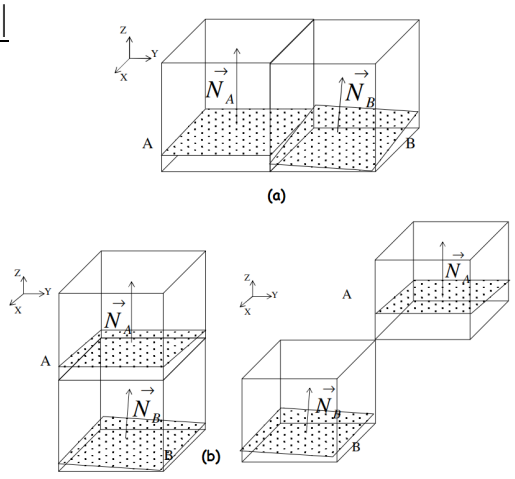


Illustration 10: (a) Exemple de deux voisins pouvant être fusionnés. (b) Exemple de deux voisins ne pouvant pas être fusionnés.(Wang et Tsang, 2004)

I.3.1.4. Segmentation par transformée de Hough

La transformée de Hough est une méthode de reconnaissance de lignes dans des photographies mise au point par Paul V. C. Hough en 1962. Elle a depuis été adaptée, notamment pour la détection de plans dans un nuage de points 3D. Cette méthode a elle aussi recours à un accumulateur. Ce dernier va servir à stocker les données (θ,φ,ρ) issues d'un point du nuage. Ainsi, un point du nuage va générer une surface dans l'accumulateur. Cette surface suit la relation suivante :

$$\cos(\varphi)\times\cos(\theta)\times x+\cos(\varphi)\times\sin(\theta)\times y+\sin(\varphi)\times z+\rho=0$$

De cette façon, l'algorithme va traiter plusieurs points du nuages et donc créer différentes surfaces dans l'accumulateur. L'intersection de ces surfaces peut supposer la présence d'un plan. Pour détecter le plan le plus probable, un vote a lieu basé sur la distance entre le point et le plan défini par (θ,φ,ρ). Si la distance est inférieure au seuil pré-établi, on incrémente de 1 le nombre de votes en faveur de cette surface. A l'issue de cette étape, la surface ayant le plus de voix est choisie (Simonetto, 2018 ; Borrmann *et al*., 2011). On peut observer la surface générée par un point dans un accumulateur sur la figure 11.a. Chaque flèche relie un point du nuage à sa surface correspondante dans l'accumulateur.

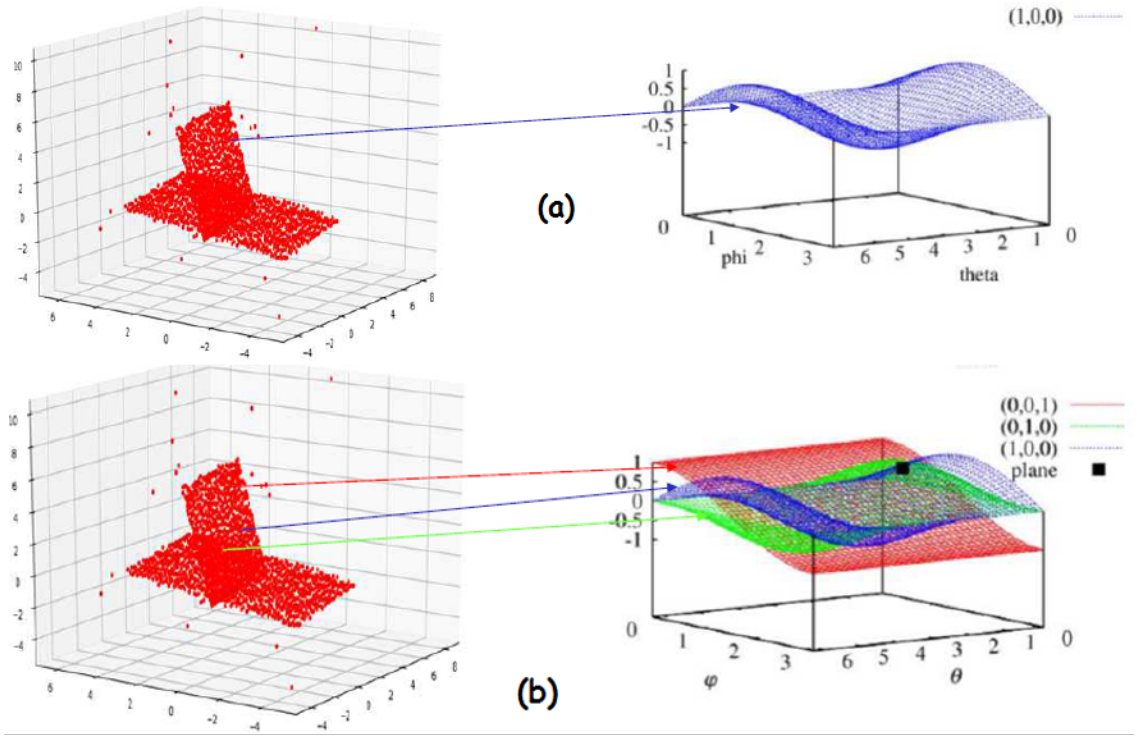


Illustration 11: (a) Représentation dans l'accumulateur de la surface correspondant à un point du nuage. (b) Représentation dans l'accumulateur de 3 surfaces issues de 3 points du nuage et leur intersection (Simonetto, 2018 inspiré de Borrmann et al., 2011)

Les résultats obtenus à partir de cette méthode sont bons, mais elle possède deux inconvénients. La première est qu'elle nécessite un temps de calcul relativement long, qui est très fortement corrélé au nombre de points présents dans le nuage. De plus, la représentation de l'accumulateur pose également des difficultés (Borrmann *et al*., 2011).

I.3.1.5. Segmentation par algorithme RANSAC

L'algorithme RANSAC[5] est probablement l'algorithme le plus présent dans la littérature quand il est question d'effectuer des interpolations de données comportant un nombre non négligeable de points bruités. En effet, le problème soulevé par l'acquisition de données via le scanner laser reste la colossale quantité de données mesurées parmi lesquelles un nombre important de points faux. De plus, le temps de calcul est moins contraignant que pour la transformée de Hough.

Il s'agit d'un algorithme basé sur les probabilités. En effet, la fiabilité du résultat dépend directement du nombre d'itération. Concrètement, à chaque itération, l'algorithme construit la forme géométrique désirée à partir du minimum de points possibles choisis au hasard. Par exemple, pour construire une droite, il faudra toujours au minimum deux points. La distance entre la droite créée et les autres points du nuage est alors mesurée. Cette étape est répétée *k* fois. Plus la valeur de *k* est élevée, plus la probabilité d'obtenir le meilleur résultat possible augmente. Une fois l'étape d'itération terminée, la forme géométrique qui approxime le plus de points est extraite et l'algorithme continue avec les points restants. La figure 12 présente le principe de l'algorithme RANSAC pour la détection de droites.

Pseudo code pour la détection d'une droite à partir de l'algorithme RANSAC

Données en entrées :
- Nuage de points
- Nombre d'itérations à réaliser → k
- Écart maximal entre un point retenu et le modèle → t

Données en sorties :
- Meilleure droite détectée
- Ensemble de points justes (inliers)
- Ensemble de points faux (outliers)

Initialisation
```
Itérateur = 0
Meilleur_modèle = aucun
Meilleur_points = aucun
Points_sélectionnés = aucun
Modèle = aucun
point[i] = ième point du nuage de points

Tant que Itérateur < k :
    Points_sélectionnés ← 2 points choisis au hasard
    Modèle ← Droite correspondant aux deux points extraits
    Pour chaque point[i] non présent dans Points_sélectionnés :
        Si la distance entre le point[i] et le modèle < t :
            Points_sélectionnés ← point[i]
    Si Meilleur_points < Points_sélectionnés :
        Meilleur_points ← Points_sélectionnés
        Meilleur_modèle ← modèle
    Incrémentation de l'itérateur
```

Illustration 12: Pseudo-code de l'algorithme RANSAC pour la détection de droite

5 RANdom SAmple Consensus.

La figure 13 représente le résultat de l'application du pseudo code de la figure 12 dans Python. Pour tester la robustesse de l'algorithme face au bruit, nous avons volontairement créé un modèle contenant 700 points justes (inliers) et 100 points faux (outliers). Comme nous pouvons l'observer, la droite moyennant l'ensemble des points n'est pas du tout représentative des inliers. En revanche, la droite issu de l'algorithme RANSAC exclu correctement les outliers situés à une distance supérieure de 20 unités[6] du modèle.

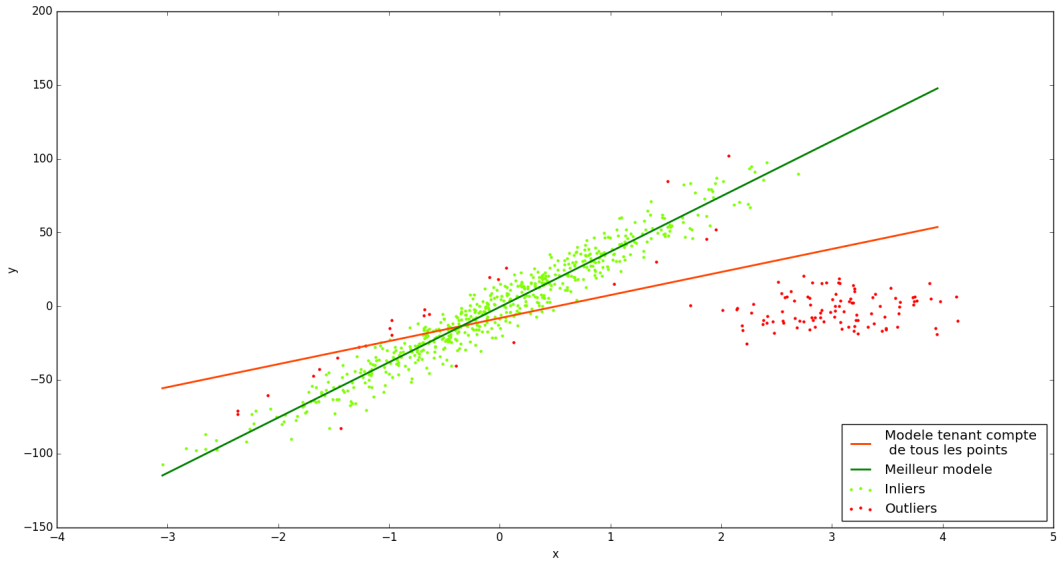


Illustration 13: Principe de l'algorithme RANSAC pour la détection d'une droite dans un espace en deux dimensions (python)

Cet algorithme peut par la suite être étendu pour traiter des données en trois dimensions. Nous allons en particulier parler de la méthode proposée par Schnabel *et al.* (2007), puisqu'il s'agit de celle que nous utiliserons dans la partie II.2.1. pour proposer une segmentation efficace de nos nuages de points. Ce travail permet la détection de formes simples au sein d'un nuage de points tels que des plans, des cylindres, des sphères, des cônes ou bien des tores. Cette idée de décomposition de l'espace en formes géométriques simple était déjà énoncée par le peintre Paul Cézanne en 1904 en ces termes : « Tout dans la nature se modèle sur la sphère, le cône et le cylindre, il faut apprendre à peindre sur ces figures simples, on pourra ensuite faire tout ce qu'on voudra »[7]. L'algorithme va permettre en outre, de traiter des millions de points en un temps « raisonnable », ce qui est un avantage considérable pour le traitement des données au sein d'un cabinet de géomètre.

Les auteurs présentent leur travail comme étant un « algorithme RANSAC à haute performance » dans le sens où ils se sont attachés à trouver le meilleur moyen de sélectionner un point tout en veillant à utiliser un temps de calcul relativement faible. Pour ce faire, la méthode se base sur le calcul de la surface normale de chaque point du nuage. En effet, dans le cadre de la détection des formes géométriques citées précédemment il faut, selon la forme, déterminer entre 3 et 7 paramètres. Chaque point fixe un paramètre de la forme et l'orientation de la normale en fixe deux autres. De cette manière, une forme peut être estimée à partir d'un ou deux échantillons seulement. Nous allons nous focaliser sur la détection de plans.

6 Meilleure valeur déduite après plusieurs tests.
7 *Lettre à Emile Bernard, 1904.*

I.3.2. Maillage

L'étape du maillage est une étape de modélisation consistant à transformer le nuage de points en modèle surfacique. Concrètement, cela consiste à relier les points entre eux en respectant certaines conditions. La méthode utilisée pour la création de maillage repose la plupart du temps sur la triangulation de Delaunay où chaque triangle va créer une facette. Ainsi, la surface va être composée d'une multitude de triangles, caractérisés par leurs sommets et leurs cotés. Plus le nuage de points sera dense, plus la précision du maillage sera fidèle au bâtiment mesuré. Mais la densité du nuage va également poser le problème de la taille des fichiers. En effet, chaque triangle va contenir comme information ses sommets. La quantité d'informations à stocker devient rapidement très importante ce qui peut impacter de manière non négligeable le temps de traitement.

Les points sont reliés principalement en fonction de leur éloignement, qui est défini par le diagramme de Voronoï. Ce dernier décompose l'espace en régions organisées autour de chaque point de telle sorte qu'il s'agisse du point le plus proche en n'importe quel endroit de la région. On procède ensuite à la triangulation en elle même en respectant le procédé suivant : « deux points sont reliés par une arrête s'ils sont dans des régions de Voronoï ayant une arête commune. » (Aurenhammer *et al.,* 2000). La figure 14 illustre la création d'un diagramme de Voronoï et d'une triangulation de Delaunay d'un ensemble de points. Cette illustration a été réalisée en deux dimensions sous le logiciel de dessin AutoCAD. La réalisation d'une triangulation sur des données en 3D est plus délicate. En effet, il faut avoir recours à une méthode permettant de ne pas créer des formes indésirables puisqu'il existe de nombreuses façons de relier un ensemble de points 3D proches.

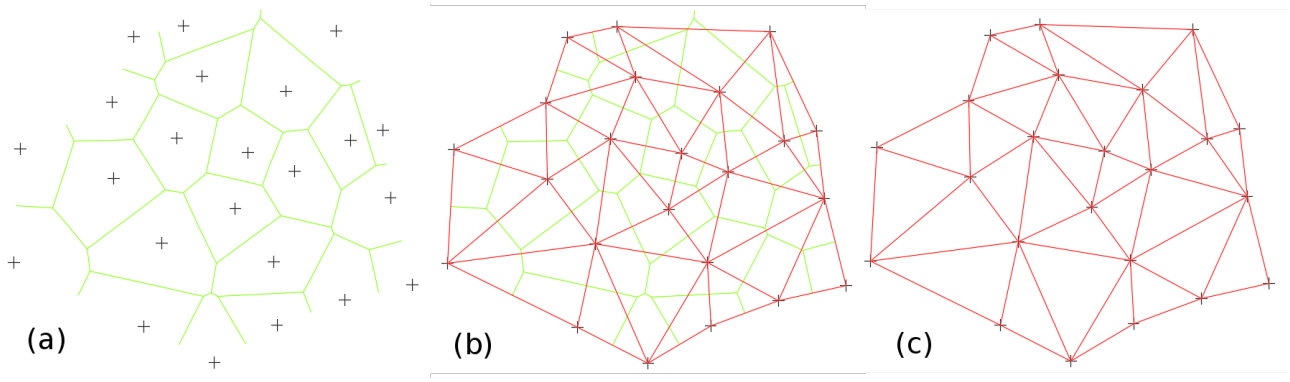


Illustration 14: (a) Diagramme de Voronoï d'un semis de points. (b) Superposition du diagramme de Voronoï et de la triangulation de Delaunay. (c) Triangulation de Delaunay

Comme nous l'avons énoncé ci-dessus, le maillage est un procédé de modélisation. Il ne s'agit pas de l'unique procédé. Il existe également la modélisation géométrique, qui consiste à modéliser le nuage de points par des primitives géométriques simples. Si le modèle obtenu est beaucoup plus léger et présente un meilleur rendu, il ne permet pas de modéliser des formes complexes. Or dans le cas de l'élaboration de plans de façades, le géomètre intervient uniquement sur des bâtiments construits, présentant des formes variées. Dans ce cas de figure, nous souhaitons représenter le modèle « tel que saisi ». Il peut être intéressant de combiner ces deux approches afin d'obtenir un modèle hybride, utilisant des primitives géométriques pour les éléments architecturaux simples, et le maillage pour les éléments architecturaux complexes.

I.3.3. Extraction de formes relatives à la création de plans de façade

La finalité du travail est de pouvoir créer un plan de façade comportant le maximum d'informations générées de façon semi-automatique. Concrètement, cela revient à représenter les contours et les lignes remarquables des éléments architecturaux par des lignes en deux dimensions. Dans sa thèse, H. Boulaassal définit un contours comme étant « l'ensemble des points du nuage constituant les arêtes extérieures ou intérieures d'un élément architectural situé sur la façade d'un bâtiment ». La figure 15 présente le plan de façade d'un bâtiment relevé au scanner laser. Nous pouvons observer que les éléments principaux du bâti sont représentés tels que la toiture, les portes et les fenêtres. Mais sont également présents les parements d'ouvertures, les appuis de fenêtre, les corniches et les ornements. Si ces éléments viennent enrichir le plan et lui apportent une finition esthétique, leur détection et leur vectorisation est délicate et nécessite souvent une correction manuelle.

Illustration 15: Exemple d'éléments présents sur un plan de façade.

Différentes méthodes sont présentées dans la littérature quant à la détection et la vectorisation des contours. Concernant l'extraction des éléments principaux tels que le toit ou les ouvertures, la majorité des méthodes s'appuient sur un modèle surfacique. Parmi ces méthodes, nous pouvons citer entre autres l'algorithme Alpha – Shape (Edelsbrunner *et al*, 1994) et l'algorithme de comparaison des longueurs des triangles du maillage (Boulaassal, 2010). Nous détaillerons le fonctionnement de ces méthodes par la suite. Certaines méthodes se basent directement sur le nuage de points. C'est le cas de la méthode présentée par C. Briese (2006) ou de l'algorithme écrit par Weber *et al*. (2010). Les lignes à extraire pour la création de plans de façade sont décrites par C. Briese (2006) et illustrées dans la figure 16 . Les lignes de rupture de pente sont détectées dans le nuage à partir de l'intersection de deux plans. Pour les lignes courbes, l'auteur détermine la direction principale de la courbature à partir d'une quadrique[8]. En s'appuyant sur cette direction, nous pouvons estimer un ensemble de points de contours. La méthode employée pour détecter les bords abrupts est similaire à celle utilisée pour les lignes de rupture si ce n'est qu'il n'y a pas d'intersection de plan. Dans ce cas, on détermine deux plans et on réalise un calcul de résidus sur tous les points dans le but de les affecter à l'un des deux plans. Les contours de chaque surface sont ensuite déterminés.

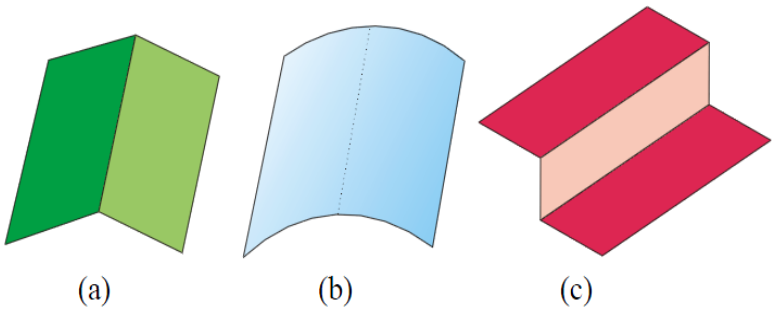


Illustration 16: (a) Ligne de rupture. (b) Ligne en arc. (c) Ligne de bord (C. Briese, 2006)

8 Surface algébrique de degré 2.

L'algorithme mis au point par Weber *et al.* (2010) est un procédé itératif, permettant la détection automatique des points de contours d'objets 3D. Dans un premier temps, l'ensemble des données est structurée, c'est-à-dire qu'une analyse du voisinage de chaque point est réalisée pour déterminer si le point se situe sur un contour. Pour cela, les auteurs ont recours à une application de Gauss, qui définit une correspondance entre chaque point du voisinage avec la sphère unité qui est centrée sur le point testé. Ce dernier va être triangulé avec les autres points présents dans la sphère et le vecteur normal au triangle va être calculé par la relation :

$$\vec{n}_{ij} = \vec{pp}_i \times \vec{pp}_j$$

où p est le point testé et (p_i, p_j) des points contenus dans la sphère unité. Cette démarche est illustrée par la figure 17.

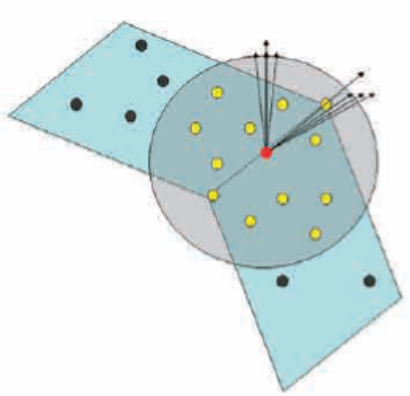


Illustration 17: Calcul des vecteurs normaux utilisés pour l'identification de points caractéristiques dans un voisinage local. (Weber et al., 2010)

L'algorithme alpha-shape va permettre de détecter les points de contours d'un nuage de points. L'objectif est d'abord de rechercher l'enveloppe convexe du nuage. Cette dernière est définie comme étant la surface minimale contenant tous les points du nuage et dont l'angle entre deux arêtes successives n'est pas supérieur à 180°. On parlera d'enveloppe concave lorsque tous les angles sont acceptés. L'application de la méthode nécessite de réaliser dans un premier temps une triangulation de Delaunay de notre nuage de points. Ces triangles vont ensuite être comparés à une valeur α qui dépend de la longueur des triangles. Si le rayon du cercle circonscrit d'un triangle est supérieur à α, alors ce triangle est supprimé. Reprenons notre échantillon de points utilisé pour illustrer la triangulation de Delaunay à la figure 14. La figure 18.a représente l'enveloppe convexe de ces points tandis que la figure 18.b représente une possibilité d'enveloppe concave. Il est important de souligner que la création de l'enveloppe convexe est unique, tandis que de nombreuses possibilités sont disponibles pour créer une enveloppe concave.

Dans son article *Introduction to Alpha Shape* (2000), K. Fischer vulgarise la méthode en utilisant une comparaison avec un pot de crème glacée. Cette comparaison a été traduite par Martin Laloux (2014) en ces termes: « *K. Fischer suggère d'assimiler le concept des formes alpha à un espace 3D rempli d'une importante masse de crème glacée contenant des pépites de chocolat. Avec une cuillère à glace nous creusons et retirons toute la crème glacée que nous pouvons atteindre sans toucher aux pépites (et ce y compris depuis l'intérieur). Plus le rayon de la cuillère diminue, plus cette cuillère peut atteindre des endroits étroits et plus de crème glacée sera enlevée. Avec un rayon infiniment petit, il ne restera que les pépites. A l'inverse, une cuillère de rayon trop grand nous empêchera de la bouger, laissant la masse intacte* ».

La figure 18.c présente deux cercles circonscrits (en bleu) et leur comparaison relative à un cercle de rayon α. Dans le cas du haut, le rayon du cercle circonscrit est inférieur à α, donc le triangle est conservé. Dans le cas du bas, le rayon du cercle circonscrit est supérieur à α, donc le triangle est supprimé. Nous pouvons observer le résultat sur l'ensemble des données à la figure 18.d. Nous en déduisons que dans cet exemple, la valeur α est trop petite, ce qui exclut des données. La principale difficulté de cette méthode est de réussir à déterminer α de façon optimale.

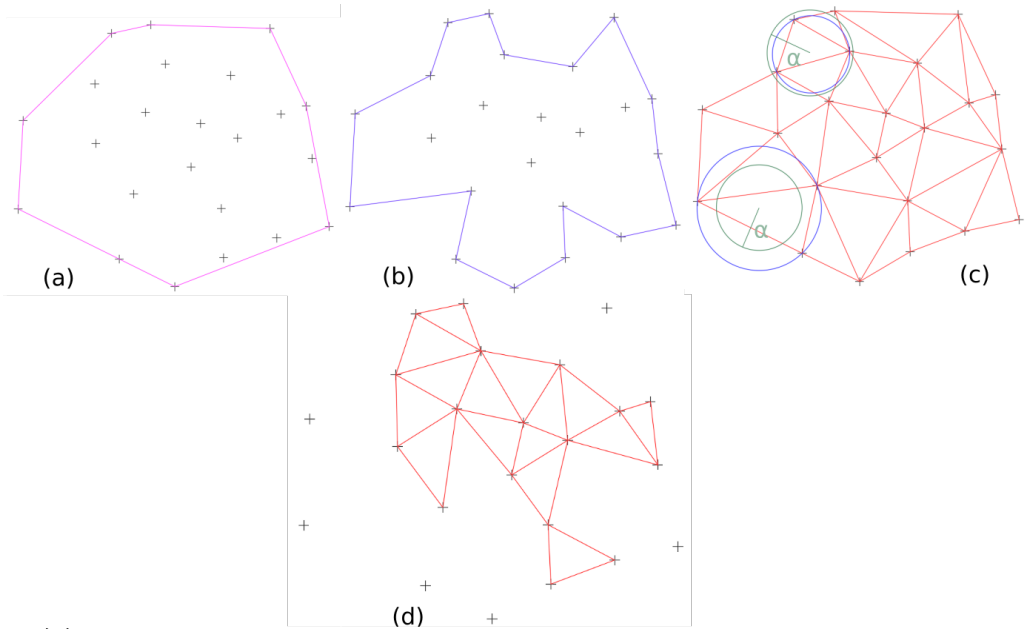


Illustration 18: (a) Enveloppe convexe d'un semis de points. (b) Exemple d'enveloppe concave du même semis. (c) Triangulation de Delaunay du semis et représentation des cercles de rayon α. (d) Résultat de l'algorithme Alpha-Shape sur le semis de points.

Un méthode régulièrement utilisée pour l'extraction de points de contour est l'analyse des longueurs des cotés des triangles. Nous pouvons distinguer deux phases d'approche. La première se base sur une étude du voisinage des triangles. En effet, deux triangles sont voisins s'ils ont une arête en commun. Cela implique qu'une arête est située sur un contour uniquement si elle n'appartient qu'à un seul triangle. Ainsi, en ne sélectionnant que les points des extrémités des arêtes appartenant à un unique triangle nous pouvons extraire les points de contours. Cette méthode est particulièrement efficace pour le traitement de points en 2D. Cependant, nous traitons ici des points en 3D, et bien que la projection des points sur un plan moyen soit possible, les relations de voisinage ne sont pas toujours conservées lorsque nous revenons à la 3D (Boulaassal, 2010). Cette méthode doit être utilisée avec précaution et nécessite une segmentation efficace en amont.

Pour remédier à ce problème, H. Boulaassal a recours à une autre solution, toujours basée sur l'analyse des cotés des triangles. Il va s'intéresser aux longueurs des cotés et énonce que « les points de contours sont les extrémités des plus longues arêtes des triangles de Delaunay ». Une fois la triangulation effectuée, toutes les arêtes sont extraites et ordonnées par longueur. L'utilisateur choisit alors un seuil au delà duquel seul les points des extrémités des arêtes correspondantes seront sélectionnées. Cette méthode a fourni des résultats très satisfaisants. Un exemple des étapes et du résultat de traitement est présenté à la figure 19 .

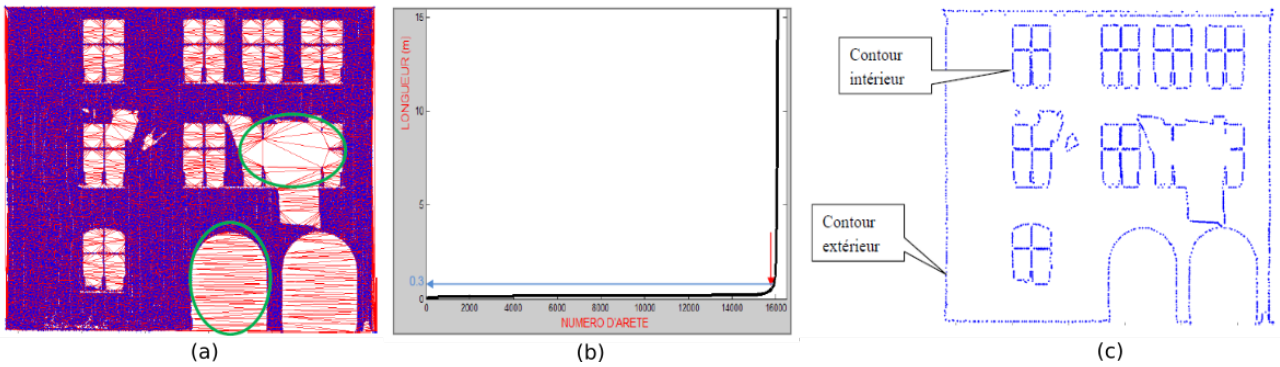


Illustration 19: (a) Façade triangulée montrant les arêtes longues (encerclées en vert). (b) Graphique présentant les longueurs des arêtes des triangles de Delaunay, rangées dans l'ordre croissant. La flèche rouge indique le changement de pente ; la flèche bleue le seuil choisi. (c) Exemple de résultat de détection de contours intérieurs et extérieurs d'une façade (Boulaassal, 2010).

Toutes les méthodes présentées ci-dessus produisent le même type de résultat : des points de contours. Ces données doivent subir une dernière étape afin d'être pleinement exploitables dans la création de plan. Cette étape est la vectorisation. Le plus souvent, les points sont vectorisés en lignes ou en arcs. Il existe différentes méthodes de vectorisation. La plus simple d'entre elles consiste à relier les points entre eux, générant ainsi des lignes comportant autant de sommets qu'il existe de points. Le rendu d'une telle méthode va d'une part ne pas être esthétique visuellement, d'autre part s'éloigner de la réalité du bâtiment. Un autre moyen consiste à simplifier les droites générées en réduisant le nombre de sommets. Cela est réalisé par le biais de méthodes d'estimation de meilleure droite, comme c'est le cas avec l'algorithme RANSAC, dont le fonctionnement est décrit page 20.

I.4. Présentation des données

I.4.1. Présentation du scanner d'acquisition

Le scanner utilisé pour l'acquisition de toutes les données qui vont être présentées par la suite est le Leica ScanStation P40. Ce scanner dispose d'une portée maximale de 270 mètres associée à une vitesse d'acquisition pouvant aller jusqu'à 1 million de points par seconde. La mesure de la distance repose sur la technique du temps de vol, qui détermine la distance qui sépare un point du scanner en utilisant la mesure du temps de parcours réalisé par le laser.

I.4.2. Présentation du nuage n°1

Le premier nuage de points est extrait d'un levé d'un magasin du 1er arrondissement de Paris. Il s'agit d'un dossier déjà complété par le cabinet et nous allons nous servir du nuage de points pour notre étude. Le levé a permis de fournir au client le plan des intérieurs, le plan des façades, des coupes, mais également la maquette numérique du bâtiment. Sur l'ensemble des points, nous avons sélectionné uniquement ceux d'une façade. La bâtiment a été construit en 1867 par l'architecte Henri Blondel et est composé d'un parement de pierres parfaitement géométriques, de grilles d'aération, d'allèges en briques et de murs pleins. Le nuage extrait est composé d'environ 19 millions de points.

I.4.3. Présentation du nuage n°1bis

Il s'agit d'un petit nuage d'environ 43 000 points que nous avons extrait du nuage 1 afin de vérifier le bon fonctionnement de nos différents tests tout en bénéficiant d'un temps de calcul amoindri. Ce nuage représente quatre pierres de parement de façade. Il est composé de quatre plans distincts : deux plans verticaux perpendiculaires à la façade, un plan vertical parallèle au plan de façade contenant les pierres de parement et un plan vertical parallèle au plan de façade contenant les joints entre les pierres.

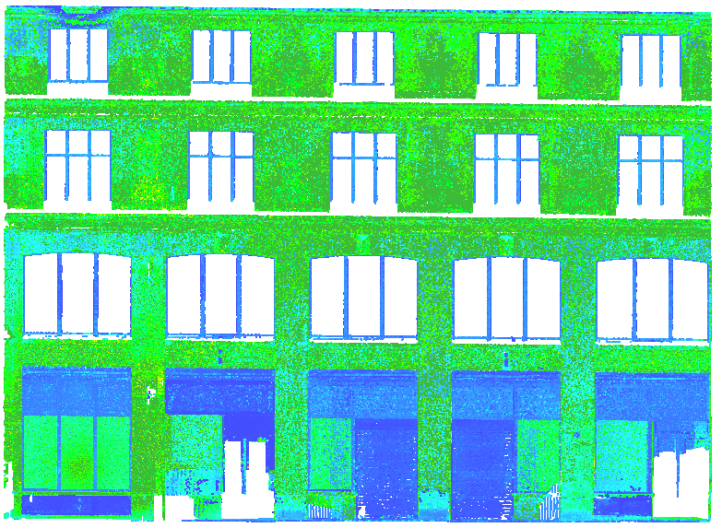


Illustration 21: Nuage de points n°1 colorisés par la valeur de l'intensité de retour de l'impulsion laser

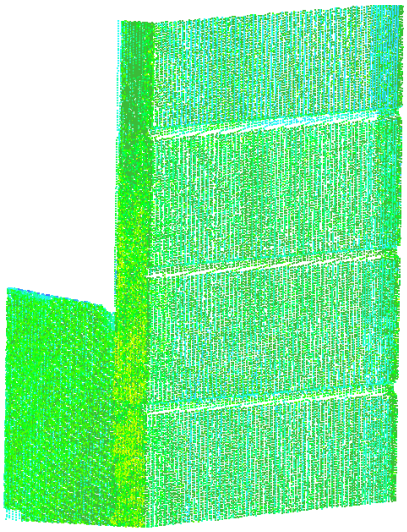


Illustration 20: Nuage de points n°1bis colorisés par la valeur de l'intensité de retour de l'impulsion laser

I.4.4. Présentation du nuage n°2

Le deuxième nuage de points a été acquis sur un immeuble du 15e arrondissement de Paris. Ce bâtiment ne présente pas beaucoup de relief. Au niveau des étages, sa façade est composée d'un mur plein en béton tandis qu'au rez-de-chaussé, la façade des magasins est composée de plaquettes de parement en pierre. Ce bâtiment s'inscrit dans le style Restauration. Le nombre d'éléments plans est nettement réduit par rapport au nuage n°1. Comme nous pouvons l'observer sur l'illustration 22, de nombreux masques sont présents, principalement au niveaux du rez-de-chaussé. Cela s'explique par le fait que la façade a été acquise uniquement par une station qui était décalée sur le côté gauche. L'objectif du traitement de ce nuage serait de détecter correctement le plan principal de la façade, et de mettre en place une méthode rapide et fiable pour la détection des encadrements et des appuis de fenêtres. D'autre part, comme nous pouvons le constater, certaines fenêtres ont également été mal acquises. Il faudra alors trouver un moyen de les représenter correctement, en s'appuyant par exemple sur les fenêtres correctement dessinées. Le nuage extrait est composé d'environ 600 000 points.

Illustration 22: Nuage de points n°2 colorisés par les couleurs réelles, capturées par numérisation grâce à la caméra intégrée au scanner.

II. Détection de contours

Dans cette partie nous allons implémenter différentes méthodes permettant d'extraire les contours d'un maximum d'éléments architecturaux. Ces méthodes seront ensuite évaluées en fonction de critères généraux, tels que la précision du rendu ou le temps d'exécution du programme, avant d'être comparées entre elles. Toutes les démarches qui vont être développées suivent le même schéma principal, décrit par la figure 23 .

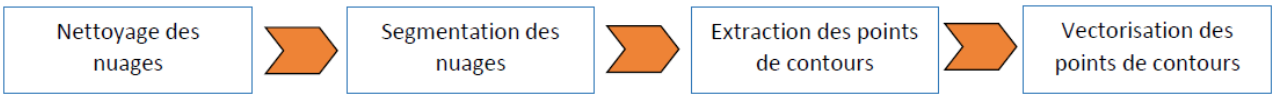


Illustration 23: Chaîne de traitement global pour obtenir les contours des éléments principaux de la façade.

II.1. Nettoyage des nuages

La phase de nettoyage est inévitable à partir du moment où nous souhaitons appliquer des algorithmes de segmentation automatiques. En effet, le scanner ne va malheureusement jamais acquérir seulement les points « utiles ». Du bruit va venir s'ajouter aux données. Il peut être de natures diverses et variées comme par exemple : une réflexion du laser sur des vitres, un piéton, une voiture ou bien encore l'acquisition de points à l'intérieur d'un bâtiment en passant par une fenêtre. Le diamètre du faisceau laser est également susceptible de générer du bruit. En effet, il est possible que le faisceau touche deux éléments de surface différents et qu'il renvoie par conséquent un signal faux. Ce phénomène se produit notamment aux niveaux des arêtes des éléments. La figure 24 illustre ce problème rencontré lors du traitement du nuage n°1 et 1bis. Le mur du rez-de-chaussée est composé d'un parement en pierres qui sont séparées par un joint. Lors du balayage, le scanner a acquis des points sur les pierres et sur les joints, mais également des points situés entre les deux, qui ne correspondent à aucun élément physique.

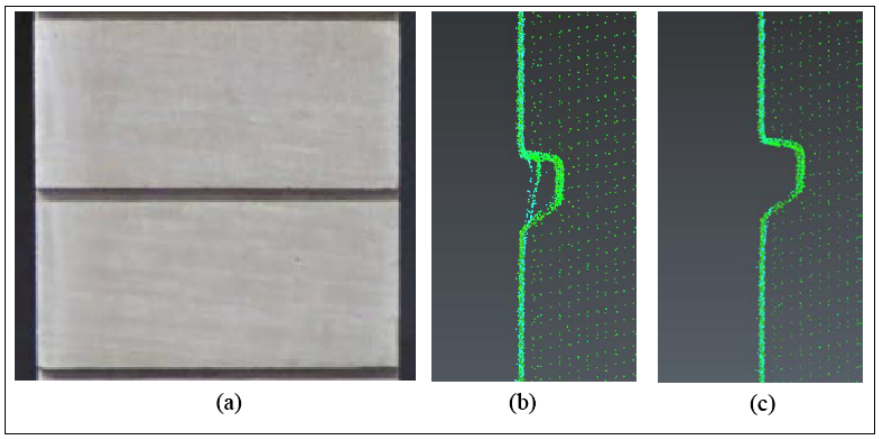


Illustration 24: (a) Photographie de 2 pierres de parement (Vue de façe). (b) Extrait du nuage de points n°1bis représentant le bruit généré au niveau du joint (Vue de profil). (c) Extrait du nuage n°1bis après l'opération de nettoyage manuelle (Vue de profil).

Il est délicat d'avoir recours à une méthode automatique pour supprimer ce bruit susceptible de fausser nos traitements. Des méthodes existent mais elles sont généralement efficaces uniquement pour supprimer le bruit qui est situé relativement loin de nos données principales. Le logiciel 3DReshaper propose de multiples solutions pour filtrer les points d'un nuage. Ces fonctions sont disponibles dans le menu « Nuage » puis « Filtrer / Exploser Nuage(s) ». La figure 25 représente l'interface permettant l'utilisation de ces outils. La première commande « Réduire le bruit » va sélectionner les points qui sont éparpillés dans la scène Cette sélection repose sur le calcul de la densité moyenne du nuage et propose de supprimer les points situés dans les zones les moins denses. Si l'outil laisse à l'utilisateur le choix de l'intensité du nettoyage, il est davantage fiable lorsque les données principales ont une densité constante. De plus, il ne supprimera jamais totalement le bruit, il ne fera que le diminuer.

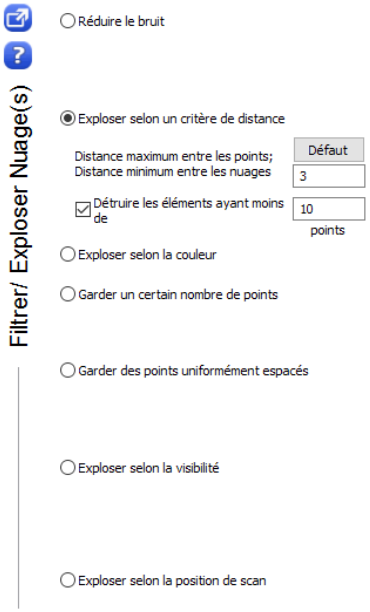


Illustration 25: Boite de dialogue de l'outil « Filtrer / Exploser Nuage(s) » du logiciel 3DReshaper

Le deuxième outil proposé est plus intéressant, car il divise le nuage à partir d'un critère de distance. Si la distance entre deux points est supérieure à la distance paramétrée par l'utilisateur, alors chacun des points est associé à un nuage distinct. Cet outil est accompagné d'une fonction permettant de détruire les nouveaux nuages créés quand ils sont composés de moins de *n* points. Cela est très efficace pour supprimer des petits groupes de points qui sont éparpillés dans la scène, mais ne réduit pas le bruit qui est attenant à la façade. Le troisième outil présente également un grand intérêt, puisqu'il permet de diviser le nuage en fonction de la couleur des points. Cela peut être utile pour nettoyer les bruits provoqués par un reflet. En effet, il est possible de coloriser des points selon une direction grâce à l'outil « Colorer selon une direction » présent dans le menu « Mesure ». Cette méthode a été utilisée en prétraitement du nuage n°1 et est particulièrement efficace pour supprimer le bruit qui est attenant à la façade. La figure 26 montre le résultat obtenu avec cette méthode pour le nettoyage de la façade que nous souhaitons traiter. Nous avons dans un premier temps colorisé les points en fonction de leur éloignement à la façade, puis effectué une division selon la couleur. Cette opération nous a permis de supprimer rapidement les façades qui ne nous intéressaient pas, le bruit généré par les passants et les voitures, les reflets provoqués par les fenêtres, mais également les lampadaires

qui sont attenants à la façade. Mais peu importe les outils utilisés, le nettoyage devra toujours se conclure par une phase manuelle. Il s'agit de l'unique moyen pour supprimer les points bruités présentés à la figure 24.b.

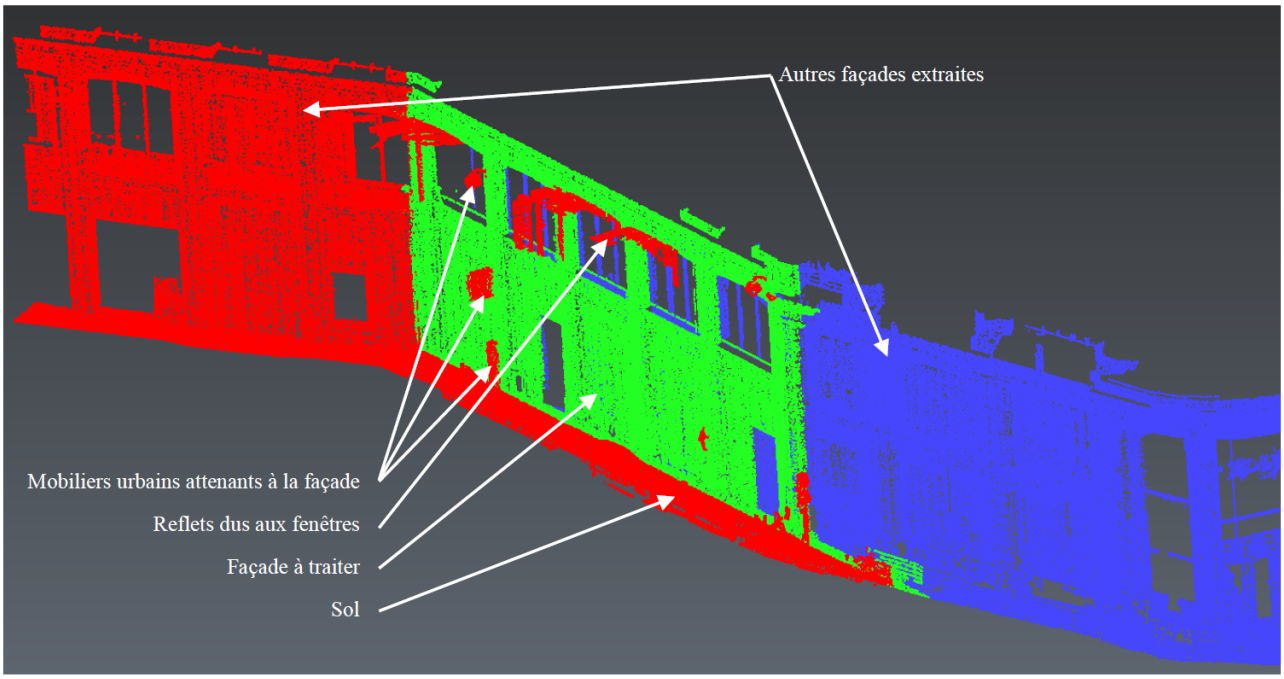


Illustration 26: Segmentation des différentes façades composant le nuage n°1

Nous soulignerons que les travaux d'Hernandez et al (2009) peuvent constituer une première approche pour extraire une façade des différents objets urbains présents sur la voirie.

II.2. Segmentation des nuages

Comme énoncé dans la partie I.3.1., séparer les différents composants d'une scène est nécessaire si l'on souhaite effectuer des analyses et des opérations sur un ensemble de points précis. Les différentes méthodes de segmentation présentées dans l'état de l'art sont aujourd'hui bien connues, mais leurs applications à des données en 3 dimensions au sein de cabinets de géomètres ne sont pas, ou peu, employées. L'objectif de cette partie est d'appliquer certaines de ces méthodes à nos données, en jouant sur les différents paramètres existants, afin de proposer une ou plusieurs solutions en fonction des cas rencontrés. Nous appliquerons dans un premier temps une segmentation géométrique par le biais d'algorithmes RANSAC adaptés à notre problématique. Puis nous traiterons les résultats obtenus avec la méthode de croissance de surface dont le fonctionnement a été expliqué page 16.

II.2.1. Détection des plans de façade par RANSAC adaptés

Nous avons fait le choix d'avoir recours à la méthode de détection proposée par l'algorithme RANSAC à cause de sa robustesse face aux bruits. En effet, si les points bruités sont à une distance supérieure au seuil choisi, ils ne seront pas retenu par l'algorithme et ne viendront pas perturber les traitements ultérieurs. D'autre part, les éléments architecturaux composants une façade sont majoritairement constitués de plans. Leurs détections précise nous permettrait par la suite d'extraire puis de vectoriser leurs contours. Nous avons mis au point et testé différentes versions afin de sélectionner celle produisant les meilleurs résultats pour notre application au sein du cabinet.

II.2.1.1. Algorithme RANSAC de CloudCompare

II.2.1.1.1. Méthode

CloudCompare est un logiciel de traitement des nuages de points en trois dimensions. Sa première fonction était de comparer deux nuages de points entre eux ou un écart entre un nuage et un modèle surfacique par le biais d'une carte de chaleur. Le logiciel est sous licence GPL, ce qui signifie qu'il s'agit d'un logiciel libre que l'utilisateur peut exécuter pour n'importe quel usage, peut étudier et modifier le fonctionnement interne d'un programme et peut même redistribuer des copies du logiciel[9]. Grâce à cette licence, le logiciel a pu s'enrichir au fur et à mesure de diverses fonctionnalités comme par exemple le recalage entre deux nuages, des calculs de densité ou bien encore des outils de segmentations automatiques.

C'est notamment cette dernière catégorie que nous allons exploiter pour traiter nos données. En effet, le logiciel dispose d'un plugin de détection de formes par l'algorithme RANSAC selon l'approche proposée par Schnabel *et al.* (2007). Cette méthode a l'avantage de proposer un temps de calcul relativement faible grâce à l'estimation en amont des normales en chaque point du nuage. La figure 27 présente la boîte de dialogue du plugin. Dans le premier champ, l'utilisateur renseigne le nombre minimum de points qui doivent appartenir à une surface pour que celle-ci soit détectée. Ce paramètre permet d'imposer à l'algorithme une condition de fin. Le deuxième champ correspond à la distance seuil séparant un point du plan estimé. Il s'agit du paramètre commun à tous les algorithmes RANSAC. Il est très important, puisqu'il va jouer sur la qualité de la segmentation. Une valeur trop faible va provoquer une sur-segmentation, ce qui va compliquer la phase d'extraction de contours. A l'inverse, une valeur trop élevée de ce paramètre va entraîner une sous-segmentation rendant imprécis tout traitement futur.

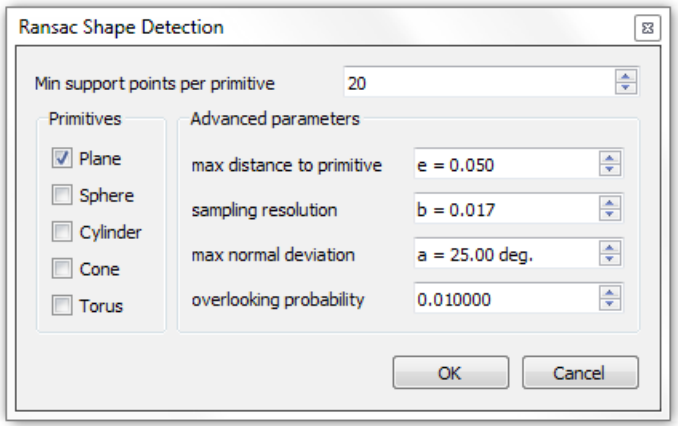


Illustration 27: Boîte de dialogue du plugin RANSAC du logiciel CloudCompare

Le paramètre de résolution spatial permet à l'utilisateur de rentrer la distance entre deux points au-delà de laquelle nous considérons qu'ils appartiennent à deux éléments architecturaux distincts. Il s'agit de la même fonction que celle présente dans le logiciel de traitement 3DReshaper (cf. page 29) qui est directement intégré dans l'algorithme. Cela permet de résoudre facilement le problème de la connectivité surfacique entre deux segments plans présenté par Boulaassal (2010)[10]. Le champ « max normal deviation » correspond à l'écart angulaire maximum entre les normales de deux points pour qu'ils appartiennent au même plan. Enfin le dernier champ permet de définir la

9 www.gnu.org – Définition du logiciel libre.

10 « Segmentation et modélisation géométriques de façades de bâtiments à partir de relevés laser terrestres » - page 94.

probabilité de choisir les meilleurs points possibles pour le calcul du plan. Cette version de l'algorithme permet également d'extraire d'autres formes que des plans. Ainsi, des points composant des sphères, des cylindres, des cônes ou encore des tores pourront être détectés. Pour pouvoir effectuer le traitement, l'utilisateur devra au préalable importer son nuage. CloudCompare prend en charge un nombre important de formats de nuages de points. Dans notre étude, nous travaillerons principalement avec des données aux formats .txt.

II.2.1.1.2. Application et analyse des résultats

Avant de lancer l'algorithme, une étude préliminaire du nuage doit être effectuée pour réaliser un paramétrage optimal. En effet, nous devons clairement définir ce que nous aimerions extraire afin de rentrer les paramètres correspondants. Cette étape est extrêmement importante, puisqu'elle va directement influencer la qualité et la pertinence de la segmentation, mais également l'importance du temps de calcul.

Nous allons traiter dans un premier temps le nuage n°1bis. Dans ces données, nous souhaitons extraire les pierres de parement afin de pouvoir faciliter et automatiser leurs créations dans un plan d'élévation. Pour ce faire, nous devons distinguer les pierres des joints. Nous constatons que ces derniers sont situés dans un renfoncement de 2 cm par rapport aux pierres. Pour nous assurer d'une bonne détection, il faut donc que le paramètre « max distance to primitive » soit inférieur à la moitié de cette distance. En effet, si pendant la phase de tirage aléatoire des points, certains sont choisis entre les deux plans à détecter, l'épaisseur du plan généré va englober les points de pierres et les points de joints.

De même, l'algorithme propose de diviser les nuages dont la distance est supérieure à un certain seuil donné. Dans le cas du nuage n°1bis, nous aimerions séparer chaque pierre des autres. Avec l'algorithme RANSAC classique, toutes les pierres seraient extraites dans un même nuage, puisqu'elles appartiennent toutes à un même plan mathématique. L'espacement vertical entre chaque pierre est de 3,5 cm. Il faut veiller à ne pas renseigner une valeur trop faible, car cela augmenterait les risques d'obtenir une sur-segmentation. La figure 28 présente différents résultats obtenus sur le nuage n°1bis pour de multiples paramètres d'entrée. Sur la figure (a) nous observons une sur-segmentation due à une valeur trop faible de la distance maximale entre deux entités. La figure (b) illustre le cas où la distance seuil est trop importante, ce qui ne permet pas de distinguer les points de joints des points de pierres. Il s'agit d'un cas de sous-segmentation. L'illustration (c) est le résultat d'un paramètre d'épaisseur de plan de bonne qualité, mais de distance maximale entre deux segments plans trop importante. Enfin, dans la figure (d), tous les plans ont été extraits et chaque pierre a été séparée afin de pouvoir être traitée séparément.

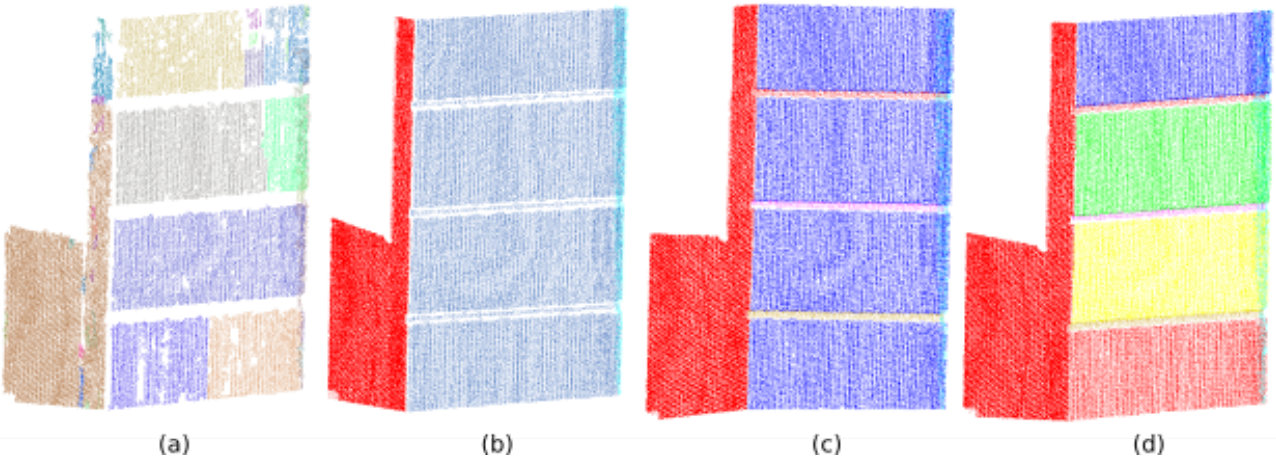


Illustration 28: (a) Sur-segmentation du nuage n°1bis. (b) Sous-segmentation : les joints sont extraits comme étant des pierres de parement. (c) Sous-segmentation : les différentes pierres de parement ne sont pas dissociées. (d) Résultat optimal

Le tableau 1 permet d'évaluer les résultats obtenus après le traitement du nuage n°1bis par le plugin RANSAC du logiciel CloudCompare. Le nuage évalué est celui présenté à la figure 28.c. Le tableau a été organisé sous forme de matrice de confusion, où chaque colonne représente le nombre de points détectés par l'algorithme pour chaque entité architecturale et chaque ligne référence le nombre de points extraits manuellement. Par exemple, la colonne 1 indique que l'algorithme a classifié dans la catégorie « Pierres de parement » 39 919 points. Les lignes nous permettent de vérifier à quelle classe appartiennent ces points dans la réalité. Ainsi, parmi ces points, 38 330 appartiennent bien à la classe « Pierres de parement », 551 appartiennent en réalité à la classe « Joint », 636 appartiennent à la classe « Côté Latéral 1 » et 402 appartiennent à la classe « Côté Latéral 2 ».

(a)

	Pierres de parement	Joint	Côté Latéral 1	Côté Latéral 2	Total
Pierres de parement	38 330	0	0	0	38 330
Joint	551	2 024	0	0	2 575
Côté Latéral 1	636	0	29 386	0	30 022
Côté Latéral 2	402	0	0	2 774	3 176
Total	39 919	2024	29386	2774	74 103
Précision de la classe	96%	100%	100%	100%	
Sensibilité	100%	79%	98%	87%	
Sensibilité Globale	98%				

(b)

	Dénomination des éléments détectés par l'algorithme
	Dénomination des éléments détectés manuellement
	Nombre de points correctement détectés dans une classe
	Nombre de points mal détectés dans une classe

Tableau 1: (a) Résultat du plugin RANSAC du logiciel CloudCompare appliqué au nuage n°1bis sous forme de matrice de confusion. (b) Légende des couleurs du tableau.

Pour évaluer plus précisément ces résultats, trois critères sont calculés. Le premier est la précision de la classe. Il correspond à la qualité de la classe détectée, exprimé comme étant le rapport entre le nombre de points correctement classés par l'algorithme et le nombre de points total classés dans la même catégorie (correct + incorrect). Nous remarquons que pour chaque classe, la précision est supérieure à 96%. Cela signifie que quasiment la totalité des points détectés appartiennent à la bonne catégorie. Une faible erreur survient à la colonne 1 où l'on remarque que quelques points appartenant à la catégorie « Joint », « Côté 1 » et « Côté 2 » ont été détectés en tant que « Pierres de parement ». Cette erreur est due à l'épaisseur du plan et à l'ordre de détection, mais reste négligeable.

Le deuxième critère est la sensibilité. Cela correspond au nombre de résultats positifs obtenu avec

l'algorithme par rapport au nombre de résultats positifs total pour une catégorie. Il permet d'apprécier le pourcentage de points qui ont été mal (ou pas) détectés. La classe « Joint » est la moins bien détectée avec 79%, mais ce chiffre est satisfaisant compte tenu du niveau de détails. De plus, les joints ne figurant pas sur le plan de façade, l'optimisation de leur détection n'est pas notre priorité. Enfin le troisième paramètre est la sensibilité globale de notre segmentation. Elle est calculée en divisant le nombre total de points correctement détectés par l'algorithme par le nombre total de points détectés manuellement. Il est important de souligner qu'il ne s'agit pas d'une moyenne des sensibilités. Pour notre exemple, cela signifie que sur l'ensemble des points du nuage, 98% ont été extraits correctement. Le temps de traitement pour ce nuage a été très bref : moins de 5 secondes. Ce résultat est donc très satisfaisant.

Nous effectuons à présent la même analyse sur le nuage n°1 afin de vérifier si la qualité des résultats obtenus sur l'échantillon peut s'étendre à la façade entière. Les pierres de parement constituent toujours le niveau de détails le plus faible à détecter, c'est pourquoi nous utilisons les mêmes paramètres que précédemment. Le résultat obtenu est surprenant et difficilement explicable. En effet, même si nous renseignons une distance maximale entre les points et le meilleur plan de 1 cm, les nuages obtenus après la segmentation contiennent des points dans la distance au plan peut atteindre 7 cm. L'illustration xx montre ce résultat. Les points verts et rouges représentent un nuage obtenu suite à la segmentation sous le logiciel CloudCompare. Nous avons colorisé les points en fonction de leur distance au plan grâce à l'outil « Comparer/Inspecter » de 3DReshaper. Un point est colorisé en rouge lorsque sa distance au plan est supérieure à 1cm. Nous pouvons voir que le résultat est inexploitable.

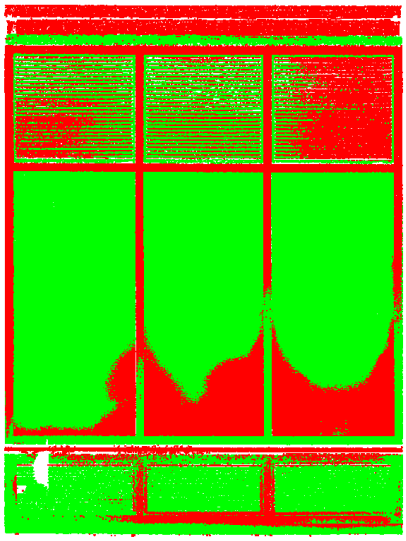


Illustration 29: Exemple de résultat obtenu lors de la segmentation du nuage n°1 par le plug-in RANSAC du logiciel CloudCompare.

La façade du nuage n°2 est beaucoup plus simple que celle du n°1, c'est pourquoi nous avons pu effectuer une segmentation manuelle du nuage afin de pouvoir évaluer quantitativement le résultat de la segmentation automatique. La partie qui nous intéresse sur cet échantillon est principalement la partie supérieure. En effet, la partie inférieure n'a pas été acquise de façon optimale, et de nombreux masques sont présents rendant difficile le traitement automatique. Pour simplifier la lecture de la matrice de confusion et faciliter l'analyse du résultat, nous avons classifié les points du nuages en 4 catégories distinctes : « Façade principale », « Appuis de fenêtres », « Murs » (autre que la façade principale) et « Autres ». Cette dernière catégorie regroupe tous les petits éléments épars dans la scène.

Pour ce traitement, nous avons choisi de paramétrer une distance maximale point – plan de 1 cm. En effet, la distance entre la façade principale et les appuis de fenêtre est d'environ 9 cm. Ainsi, nous ne risquons pas de sous-segmentation sur ces éléments. L'écart angulaire entre la normale de 2 points a été fixée à 25 degrés afin de bien séparer deux plans qui se touchent, mais dont la direction est sensiblement différente. Ce paramètre est en partie responsable du fait que la précision de chaque classe est de 100%. Le tableau 2 présente la matrice de confusion du résultat de la segmentation du nuage n°2.

(a)

	Façade	Appui de fenêtre	Murs	Autres	Non détécté	Total
Façade	399 640	0	0	0	8 125	407 765
Appui de fenêtre	0	3 626	0	0	3 898	7 524
Murs	0	0	70 349	0	3 335	73 684
Autres	0	0	0	82 524	13 226	95 750
Total	399 640	3 626	70 349	82 524	28 584	584 723
Précision de la classe	100%	100%	100%	100%		
Sensibilité	98%	48%	95%	86%		
Sensibilité Globale	95%					

(b)

	Dénomination des éléments détectés par l'algorithme
	Dénomination des éléments détectés manuellement
	Nombre de points correctement détectés dans une classe
	Nombre de points mal détectés dans une classe
	Nombre de points non détectés dans une classe

Tableau 2: (a) Résultat du plugin RANSAC du logiciel CloudCompare appliqué au nuage n°2 sous forme de matrice de confusion. (b) Légende des couleurs du tableau.

Pour chaque classe, la précision est de 100%, ce qui signifie que tous les points détectés appartiennent uniquement à un seul élément architectural. Ce résultat met en évidence la robustesse de l'algorithme RANSAC face au bruit. En revanche, nous avons un niveau de sensibilité amoindri comparé au nuage n°1bis. Cela est principalement dû au fait que la qualité d'acquisition est moins bonne. Ce sont notamment les appuis de fenêtres qui ont été mal extraits, avec une sensibilité de 46%. La précision et la qualité de la représentation des contours vont être fortement impactés, puisque plus de la moitié des points situés sur les appuis n'ont pas été extraits. Néanmoins, la sensibilité globale est satisfaisante, surtout si nous prenons en considération la qualité du nuage sur lequel le traitement a été appliqué.

La raison pour laquelle le plug-in produit des bons résultats sur la façade n°2 et des résultats inexploitables sur la façade n°1 est la différence de précision requise. En effet, dans le premier nuage, nous souhaitons faire la différence entre les pierres de parement et les joints qui sont espacé d'environ 2 cm. Dans le second nuage, nous souhaitons faire la différence entre la façade principale et les appuis de fenêtres qui sont séparés d'environ 9 cm. Ce que nous n'arrivons pas à expliquer, c'est pourquoi des points dont la distance au plan est supérieure au seuil imposé sont détectés. Par conséquent, la segmentation avec le plug-in RANSAC de CloudCompare est à éviter si d'autres choix sont possibles car nous ne maîtrisons pas le traitement.

II.2.1.2. Développement d'une approche RANSAC sous JavaScript

II.2.1.2.1. Méthode

Nous avons très largement travaillé avec le logiciel 3DReshaper pendant la réalisation de ce mémoire. Ce logiciel de traitement de nuages de points a la particularité de permettre à l'utilisateur d'écrire ses propres scripts en langage JavaScript. Cela nous permet de bénéficier des fonctions du logiciel en les intégrant dans nos propres routines. Traiter le nuage de points directement sous ce type de logiciel nous donne également l'avantage et le confort de pouvoir directement visualiser les résultats grâce à l'interface visuelle.

La méthode mise en place suit un schéma « classique » de l'algorithme RANSAC, mais nous offre une plus grande liberté dans la gestion et le traitement des différents éléments. Nous procédons dans un premier temps au chargement du nuage, puis l'utilisateur est invité à déclarer les variables. A partir de ces dernières, le programme va calculer le nombre d'itérations à réaliser. L'illustration 30 présente la boîte de dialogue pour la déclaration des variables. La phase suivante est la création d'un plan défini par 3 points choisis de façon aléatoire par le script. La distance entre ce plan et chaque point du nuage est alors calculée. Si cette distance est inférieure à la distance seuil définie par l'utilisateur, alors le point est catégorisé en tant qu'inlier. Le plan comptabilisant le maximum d'inliers à la fin des *n* itérations est retenu et les points sont ajoutés à la scène. L'algorithme réitère ce processus sur les points restants tant que le nombre de points qui n'ont pas été affectés à un plan est supérieur à x. Cette valeur est également définie par l'utilisateur lors de la déclaration des variables. Le script est disponible en annexe Annexe 9.

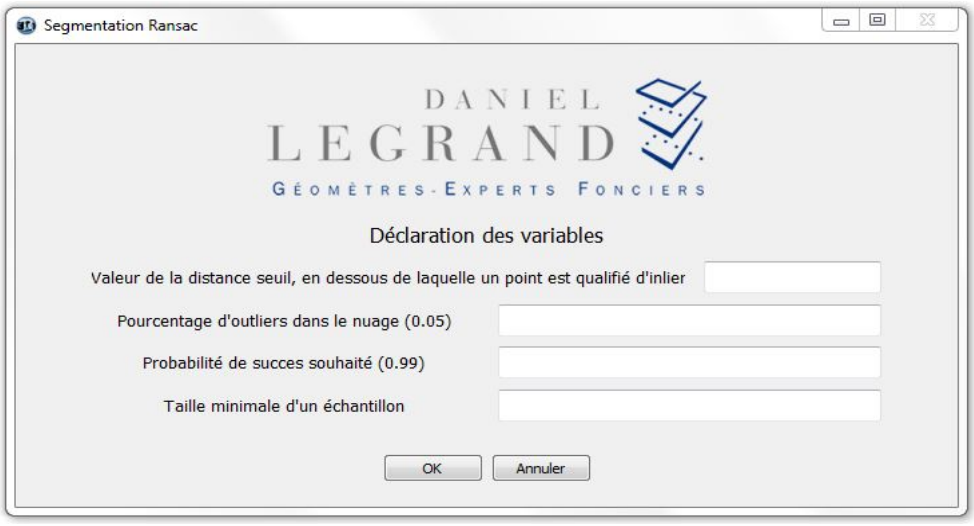


Illustration 30: Boîte de dialogue de la segmentation d'un nuage selon l'algorithme RANSAC développé sous le logiciel 3DRESHAPER.

II.2.1.2.2. Application et analyse des résultats

Comme dans la partie précédente, nous appliquons d'abord le programme au nuage n°1bis. Nous allons utiliser les même valeurs pour les variables. Pour rappel, ces valeurs sont :

- Distance seuil → 1 cm
- Minimum de points par plan → 50

- Pourcentage d'outliers dans le nuage → 5%
- Probabilité de succès souhaité → 99%

Le script effectue avec ces paramètres 58 itérations, et le calcul pour le traitement des 74 103 points est réalisé en 4 minutes et 17 secondes. Nous insérons les résultats obtenus dans la matrice de confusion (Tableau 3).

	Pierres de parement	Joint	Côté Latéral 1	Côté Latéral 2	Total
Pierres de parement	38 330	0	0	0	38 330
Joint	549	2 026	0	0	2 575
Côté Latéral 1	869	0	29 153	0	30 022
Côté Latéral 2	293	0	0	2 883	3 176
Total	40 041	2 026	29 153	2 883	74 103
Précision de la classe	96%	100%	100%	100%	
Sensibilité	100%	79%	97%	91%	
Sensibilité Globale	98%				

Tableau 3: Résultat de l'approche RANSAC développé sous 3DRESHAPER et appliqué au nuage n°1bis sous forme de matrice de confusion.

En analysant le tableau 3, nous nous apercevons rapidement que tous les points de pierres de parement ont été correctement détectés, puisque la sensibilité de la classe est de 100%. La précision de la classe est de la même grandeur que celle obtenue avec l'algorithme de CloudCompare. Le résultat est donc validé en matière de qualité de segmentation. Mais un paramètre est préoccupant avec cette méthode : la gestion de la mémoire. En effet, si nous analysons la courbe d'attribution de la mémoire vive pendant la durée des calculs, nous remarquons qu'entre chaque détection de plan, la mémoire n'est pas réinitialisée. Lors de chaque itération, la distance entre chaque point du nuage et le plan testé est calculée et stockée dans la mémoire. Multiplié par le nombre de plan à détecter et par le nombre d'itérations réalisées pour chaque plan, la quantité de données stockées est très importante. La figure 31 présente ce phénomène. Avant le lancement du calcul, la mémoire utilisée est de 30%. La première pente correspond à la recherche du premier plan et nous constatons que la mémoire croît de façon linéaire, au fur et à mesure que les distances sont calculées et stockées. Cette phase de croissance est suivie d'une période constante, correspondant à l'étape d'ajout des meilleurs inliers dans la scène. S'il apparaît que cette étape est consommatrice de temps, le pourcentage de RAM utilisé n'augmente pas. Ces périodes de croissance et de stase se répètent à chaque recherche d'un nouveau plan et nous pouvons constater que la période est de plus en plus courte. Cela s'explique par le fait que le nombre de points à traiter est de plus en plus faible au fur et à mesure que les plans sont extraits.

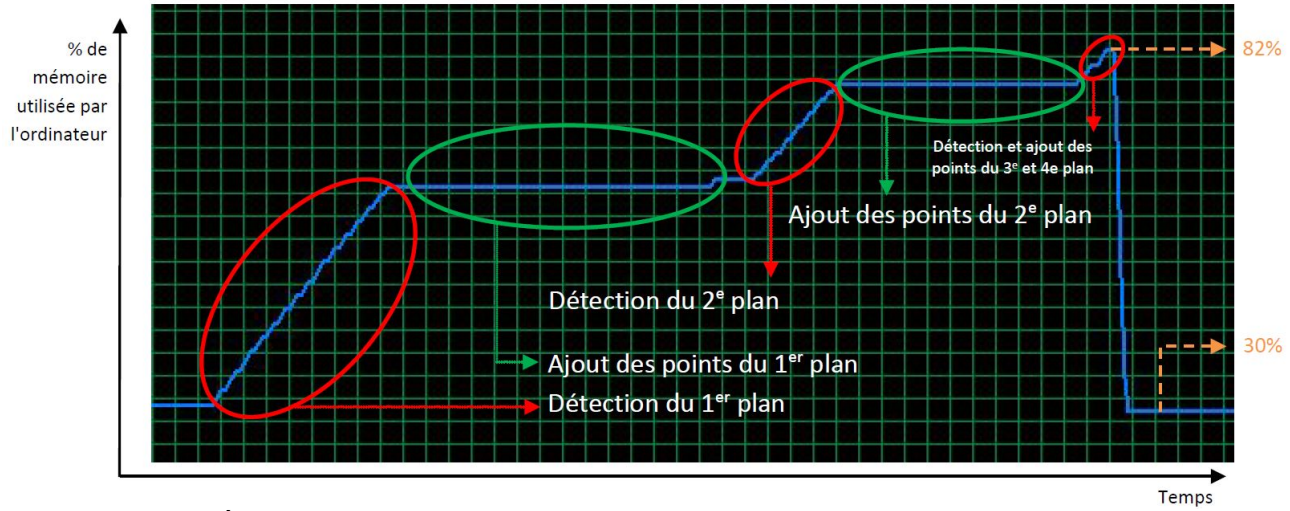


Illustration 31: Évolution de la mémoire de l'ordinateur durant l'exécution de l'approche RANSAC appliquée au nuage n°1bis sous le logiciel 3DRESHAPER.

A la fin du script, la mémoire vive réquisitionnée est de 82% et se réinitialise à 30% lors de l'arrêt du programme. Cela ne pose pas de problème pour des nuages de petite taille, comme c'est le cas pour le nuage n°1bis. En revanche, ce genre de traitement n'est pas possible pour le nuage n°1 qui comporte plus de 19 millions de points. En effet, dans ce cas la mémoire atteint 100% d'utilisation, provoquant un bug du logiciel (et plus généralement de l'ordinateur). Dans le langage JavaScript, la mémoire est gérée par un « Garbage collector », ce qui se traduit en français par « Ramasse miettes » ou « Récupérateur de mémoire ». Globalement, cet outil analyse les variables qui ne sont plus utilisées par le programme et récupère la mémoire qui leur était attribuée en supprimant leur chemin d'accès. Dans notre cas, aucune variable n'est supprimée et l'utilisation de la mémoire n'est pas contrôlée. Dans le milieu informatique, ce phénomène est appelé « fuite de mémoire ». Malgré un temps important alloué à la résolution de ce problème, la connaissance et les recherches effectuées sur ce sujet ne nous ont pas permis de stopper cette fuite de mémoire.

Au moment de l'écriture de ce mémoire, les bons résultats de cette technique sont amoindris par la gestion de la mémoire. Néanmoins, il nous a semblé intéressant de présenter ce résultat, car cette solution de traitement offre l'avantage majeur de pouvoir utiliser le très large panel d'outils mis à disposition par le logiciel. De plus, le problème de mémoire rencontré doit pouvoir être résolu par une personne familière des langages informatiques utilisant un récupérateur de mémoire.

II.2.1.3. Développement d'une approche RANSAC sous Python

II.2.1.3.1. Méthode

Nous avons décidé d'écrire un algorithme RANSAC spécifiquement adapté à nos besoins, c'est-à-dire capable de détecter correctement les surfaces représentées sur un plan de façade. Si nous raisonnons uniquement en termes de plans, une façade est composée de 4 types de plans différents. Soit P un plan d'équation cartésienne ax + by + cz + d = 0. On distingue alors (Boulaassal, 2010) :

→ Les plans horizontaux perpendiculaires au plan principal, avec a et b proches de 0.

→ Les plans verticaux perpendiculaires au plan principal avec b et c proches de 0.

→ Les plans inclinés avec c différents de 0 et 1.

→ Les plans verticaux parallèles au plan principal avec a et c proches de 0.

Ces paramètres supposent que l'axe x a été préalablement orienté parallèlement au plan de la façade et que l'axe z représente l'élévation de la façade. Il n'y a que les deux dernières catégories qui nous intéressent dans notre étude. L'algorithme que nous avons utilisé précédemment dans le logiciel CloudCompare ne distinguait pas ces différents plans. La détection des plans superflus augmente le temps de calcul et risque de supprimer des points appartenant à des plans parallèles à la façade. De plus, il faudra les supprimer, puisqu'ils vont dégrader la qualité esthétique du rendu final.

Nous choisissons d'appliquer un ordre de détection dans les types de plans afin de favoriser l'attribution d'un point au plan le plus pertinent. Nous décidons de détecter en premier les plans inclinés. En effet, comme le montre la figure 32, l'impact sur la qualité de la détection est amoindrie lorsque nous traitons dans un premier temps les points situés sur les plans inclinés. Une fois que ces points sont extraits, nous mettons en place une étape de croissance de surface pour séparer ces différents plans entre eux. L'objectif est de conserver les plans les plus importants et de renvoyer les points restants dans le nuage de points initial afin qu'ils ne soient pas exclus des traitements suivants. Pour ce faire, nous sélectionnons un premier point germe et nous l'ajoutons à une nouvelle région. Nous calculons ensuite la distance euclidienne entre ce point et tous les autres. Ceux qui présentent une distance inférieure au seuil prédéfini par l'utilisateur sont ajoutés à la région et nous réitérons cette étape pour chaque nouveau point. Une fois que tous les points ont été attribués à une région, nous supprimons les plus petites. Les régions restantes sont alors exportées au format texte.

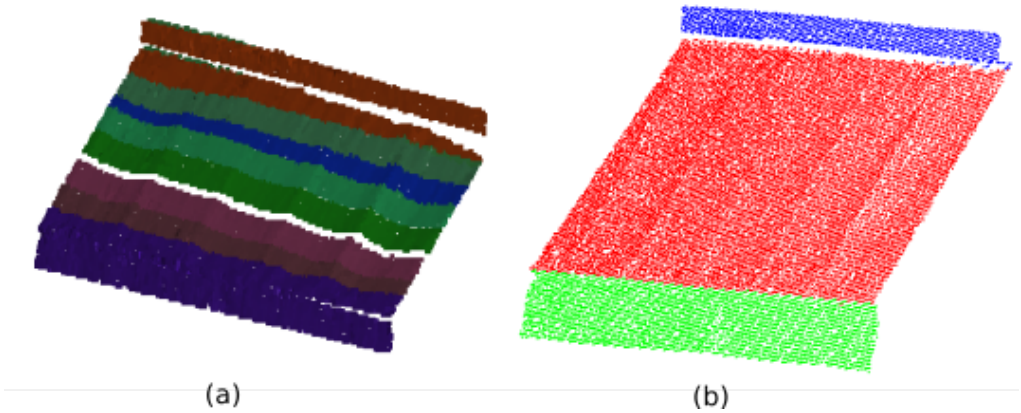


Illustration 32: (a) Résultat de l'algorithme sans priorité de détection des plans inclinés. (b) Résultat de l'algorithme avec priorité de détection des plans inclinés.

L'algorithme se poursuit par la détection des plans verticaux parallèles au plan principal. La méthode employée reprend celle expliquée dans l'état de l'art (page 20). A chaque itération, l'algorithme va extraire trois points choisis aléatoirement et calculer le plan correspondant. Après cette étape, nous rajoutons un test pour vérifier que le plan extrait est bien parallèle au plan de façade. Si la condition est vérifiée, la distance euclidienne entre chaque point du nuage et le plan est calculée. Si cette distance est inférieure au seuil prédéfini, les points sont classifiés en tant qu'inliers. A la fin de cette étape, le plan comptabilisant le plus d'inliers est retenu. Nous réitérons l'ensemble de ces étapes tant que le nombre de points affectés à un plan est supérieur à un certain seuil.

II.2.1.3.2. Application et analyse des résultats

Une fois encore, nous testons en premier notre algorithme sur le nuage n°1bis. Nous avons choisi les paramètres suivants :

Détection d'un point sur un plan incliné

Soit le vecteur normal $\vec{N}$ de chaque point P du nuage tel que $\vec{N}$ = (Nx Ny Nz). Après des études de cas, nous sommes convenus qu'un point appartient à un plan incliné si et seulement si $0{,}2 < Nz < 0{,}8$. Cela signifie que seuls les plans dont l'angle avec la verticale est compris entre 18° et 72° seront considérés comme plans inclinés.

Distance minimum entre deux plans inclinés distincts

Pour la croissance de surface, nous choisissons une distance seuil de 2 cm entre les segments plans détectés.

Taille minimum des segments plans inclinés

Nous décidons que seuls les plans inclinés contenant plus de 1000 points seront conservés.

Distance maximale entre un plan et un point pour que ce dernier soit qualifié d'inlier

Une distance de 5 mm est choisie pour le traitement du nuage n°1bis.

Nombre d'itérations à réaliser

Cette partie est la plus délicate car elle va avoir un impact direct sur la qualité de la segmentation mais également sur le temps de calcul. Le nombre d'itérations peut être calculé par la formule mathématique suivante (rappels issus de Simonetto, 2018):

→ Soit P un nuage de n points et N le nombre d'itérations à effectuer.

→ Soit ε le pourcentage d'outiliers dans les données. Alors $1-\varepsilon$ représente le pourcentage d'inliers dans les données.

→ Soit s la taille minimale d'un échantillon (s = 2 pour une droite, s = 3 pour un plan). Alors $(1-\varepsilon)^s$ représente la probabilité de n'avoir que des inliers dans un échantillon et intuitivement, $1-(1-\varepsilon)^s$ représente la probabilité d'avoir un échantillon contenant au moins un outlier.

→ Soit p la probabilité d'avoir au moins un échantillon composé uniquement d'inliers après N tirages. Alors $1-p$ est la probabilité de n'avoir aucun échantillon composé uniquement d'inliers après N tirages.

On remarque par conséquent la relation suivante : $1-p=(1-(1-\varepsilon)^s)^N$. Nous en déduisons donc que $N=\dfrac{\log(1-p)}{\log(1-(1-\varepsilon)^s)}$.

Si les valeurs de s et de p sont faciles à déterminer, ce n'est pas le cas de ε qui varie pour chaque nuage et qui n'est pas facile à estimer sans y consacrer un minimum de temps. Dans le cas du nuage de points n°1bis, nous avons réalisé dans un premier une segmentation manuelle. Grâce à cela, nous pouvons quantifier précisément le pourcentage d'inliers et d'outliers. Avec le Tableau 3, nous apercevons que le nuage contient 74 103 points dont 38 330 appartiennent à des pierres de parement. Nous avons donc 52% d'inliers et 48% d'outliers dans nos données. En appliquant ce résultat, le nombre d'itérations à réaliser serait de **30**.

Or, ce nombre est à appliquer uniquement dans le cas de l'utilisation « classique » de l'algorithme

RANSAC, ce qui n'est pas notre cas ici, car nous avons mis en place un algorithme adapté pour le traitement de nos données de façade. En effet, nous avons rajouté une condition sur l'orientation du plan détecté aléatoirement. Seul les plans orientés parallèlement à la façade ou les plans inclinés sont traités. Sur l'échantillon 1bis, cela revient à dire que les points situés sur les côtés 1 et 2 (cf. Tableau 3) sont ignorés. De ce fait, seul les points situés sur les joints sont des outliers. Cela signifie que nos données comportent 94% d'inliers et 6% d'outliers. En appliquant ces valeurs, le nombre d'itérations à réaliser serait de **3**.

Les différents tests réalisés avec cette valeur sur ce nuage ont produit des bons résultats, mais des légères erreurs ont pu être aperçues à force de répéter le traitement. Nous pensons que cela arrive dans le cas où le nombre d'itérations est très faible. En effet, avec cette méthode la détection des plans repose sur un modèle statistique et la probabilité que le meilleur plan ne soit pas détecté au bout de 3 itérations n'est pas nulle. C'est pourquoi nous décidons d'augmenter légèrement le nombre d'itérations à réaliser. Pour cela, nous avons créé une fonction que nous avons appelée « fonction de sécurité ». Ce paramètre va venir multiplier le nombre d'itérations à réaliser. L'objectif est d'augmenter le nombre d'itérations lorsque ce dernier est relativement faible et de ne pas le modifier lorsqu'il devient plus important.

Après plusieurs essais, nous apercevons que la fonction de sécurité f_s définie tel que $f_s(N)=\frac{1+\ln(N)}{\ln(N)}$ nous permet d'obtenir des résultats intéressants pour un faible surcoût en termes de temps de calcul. Nous choisissons d'appliquer cette fonction uniquement lorsque le nombre d'itérations est compris entre 2 et 10. En effet, f_s est strictement décroissante sur l'intervalle $[2;10]$ à valeur dans $[1{,}4;2{,}4]$ ce qui nous permet d'augmenter suffisamment le nombre de tests. Dans le cas de figure du nuage n°1bis, le nombre d'itérations passe ainsi de 3 à 6. L'illustration 33 permet de se rendre compte de la nécessité de l'ajout d'une fonction de sécurité.

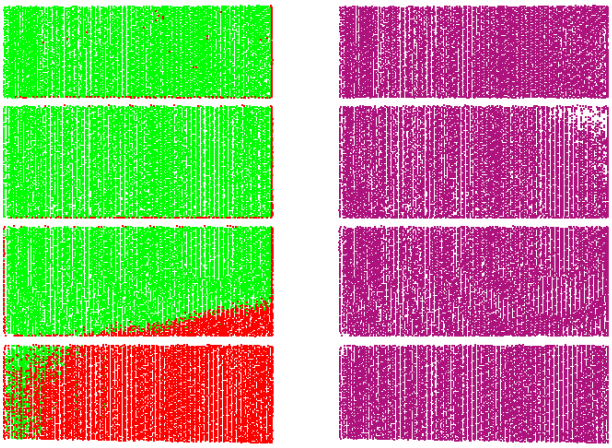


Illustration 33:
A gauche : Résultat de l'algorithme RANSAC python sans la fonction de sécurité (3 itérations)
A droite : Résultat de l'algorithme RANSAC python avec la fonction de sécurité (6 itérations)

Les points ayant servi au calcul du plan dans le premier cas sont bien des inliers. Cependant, le plan calculé n'est pas obligatoirement le meilleur. Dans l'illustration précédente, le plan vert contient exclusivement des inliers, mais pas tous les inliers.

Détection d'un plan parallèle au plan de façade

Soit P un plan extrait du nuage de points, défini par les facteurs (a, b, c, d) tel que $ax+by+cz+d=0$. Après des études de cas, nous sommes convenus qu'un plan pouvait être considéré comme parallèle au plan de façade si et seulement si $a < 0{,}02$ *et* $c < 0{,}02$. Cela signifie que seuls les plans dont l'angle avec la verticale est inférieur à 1,8° seront considérés comme plans parallèles au plan de façade.

Taille minimum des segments plans inclinés

Nous décidons que seuls les plans contenant plus de 3000 points seront conservés.

Nous allons proposer par la suite des traitements sur différents supports. En effet, le cabinet envisage de se doter d'un logiciel de traitement de points, c'est pourquoi une solution sera proposée directement à partir de ce logiciel. Une solution alternative sera également proposée dans le cas où l'investissement dans le logiciel ne se ferait pas.

L'Annexe 4 (respectivement l'Annexe 5) présente les résultats des algorithmes RANSAC + croissance de régions appliqués au nuage n°1 (respectivement au nuage n°2).

II.3. Détection des éléments architecturaux pleins et réguliers sur 3DReshaper

II.3.1. Détection des contours par enveloppe convexe

Cette méthode est particulièrement efficace pour détecter et représenter les contours des entités pleines. Nous qualifions d'éléments pleins tous les éléments constitués d'un seul bloc et ne comportant aucune ouverture. Par exemple une façade entière n'est pas un élément plein puisqu'elle contient des portes, des fenêtres... En revanche, une pierre de parement est un élément plein. Si cette détection est peu efficace pour représenter fidèlement les éléments comportant des ouvertures ou trop de détails, c'est parce qu'elle repose sur le calcul de l'enveloppe convexe du nuage. La définition de ce procédé a été donnée page 25.

Le logiciel 3DReshaper dispose directement d'une fonction « Enveloppe Convexe » disponible dans le menu « Polyligne ». Mais cette opération doit être réalisée manuellement pour chaque nuage segmenté et cela peut se reveler très long. Nous avons écrit un script en langage javascript afin d'automatiser cette action. L'utilisateur doit afficher à l'écran uniquement les nuages qu'il souhaite traiter. Le programme va dans un premier temps extraire pour chaque segment plan la direction de la normale puis calculer l'enveloppe convexe 2D en fonction de cette direction. A ce stade, les polylignes comportent un grand nombre d'arêtes et de sommets. C'est pourquoi nous ajoutons une étape de rééchantillonage des polylignes. Si nous prenons l'exemple des pierres de parement du nuage n°1bis, les contours doivent être composés de 4 arêtes et de 4 sommets à angle droit. Nous pouvons imposer ces conditions grâce à l'outil « Rééchantilloner Polyligne(s) » présent dans le menu « Polyligne ». Ces fonctions ont été regroupées dans un même script de façon à automatiser le traitement. Le script est présent en Annexe 10. L'illustration 34 présente la boîte de dialogue du programme.

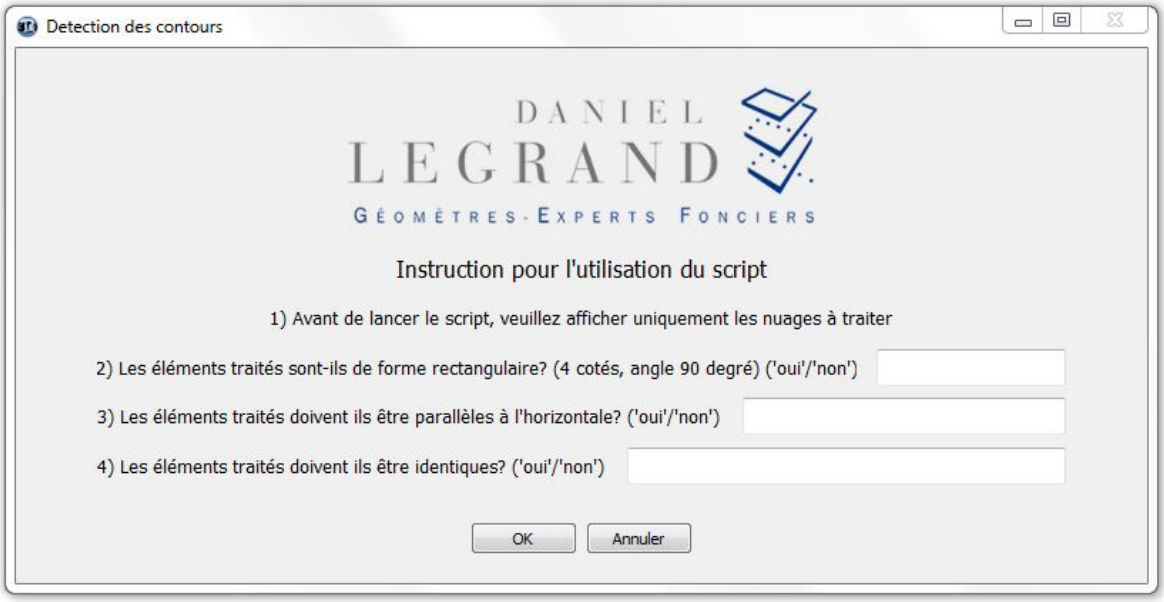


Illustration 34: Boîte de dialogue de l'algorithme de détection des contours des éléments architecturaux « pleins » - développé sous le logiciel 3DRESHAPER

Les contours obtenus ne sont toujours pas « parfaits » et doivent être améliorés. En effet, leur qualité dépend directement de la qualité de la segmentation et du nuage de points. Ainsi, il est fort probable que les contours générés ne respectent pas quelques caractéristiques inhérentes à l'architecture du bâtiment. Chaque élément doit respecter un certain nombre de conditions géométriques. Par exemple, toutes les pierres de parement doivent être de la même taille, doivent être parallèles entre elles et au sol, doivent être alignées entre elles selon le même axe vertical et/ou horizontal... L'illustration 35 illustre les conditions géométriques que doivent respecter les contours des pierres. C'est pourquoi nous avons écrit 4 programmes permettant de repositionner rapidement et précisément les contours. Ces programmes sont composés par :

→ 1 script de rotation
→ 2 scripts d'alignement horizontal et vertical
→ 1 script de clonage de contours

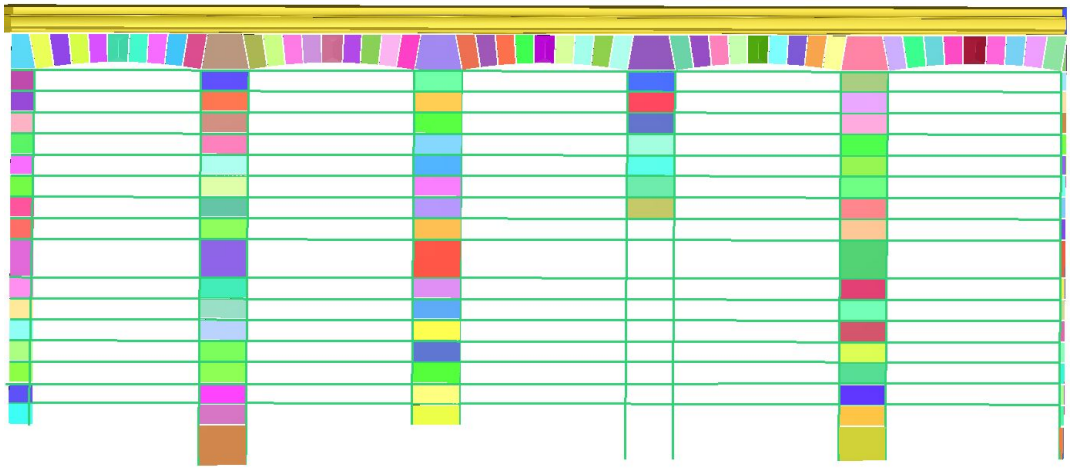


Illustration 35: Représentation de la façade n°1 sous forme de modèle numérique 3D. Chaque pierre de parement est alignée verticalement avec toutes les autres pierres situées sur la même colonne. Chaque pierre de parement est alignée horizontalement avec toutes les autres pierres situées sur la même ligne.

II.3.2. Le script de rotation

Il s'agit ici de forcer l'alignement d'un contour afin qu'il soit parfaitement parallèle à l'horizontale. Pour ce faire, chaque polyligne sélectionnée est traité une par une. On extrait dans un premier temps le barycentre ainsi que les coordonnées de chaque sommet. Le principe est de calculer l'angle entre l'horizontale et le côté à aligner. Pour faciliter cette opération, et augmenter la vitesse du traitement, nous partons du postulat que nous travaillons avec des formes géométriques simples : rectangles, losanges, triangles... et que le côté à aligner est celui dont l'écart angulaire est le plus faible. L'algorithme est écrit de telle sorte que le point 1 est toujours celui dont la valeur de Z est la plus faible. Nous rencontrons alors deux cas, qui sont illustrés par la figure 36. Dans le cas n°1, l'angle est obtenu par la formule $\alpha = \arctan\left(\frac{Z_2 - Z_1}{X_2 - X_1}\right)$ et par la formule $\alpha = \arctan\left(\frac{Z_2 - Z_1}{X_1 - X_2}\right)$ dans le cas n°2.

Une fois l'angle obtenu, nous calculons la matrice de rotation que nous appliquons aux sommets de la polyligne. Le centre de la rotation est le barycentre de la polyligne. La matrice de rotation d'angle alpha dans un espace en deux dimensions est caractérisée par l'expression suivante :
$R = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$. Mais l'application de cette matrice en l'état réalise une rotation d'angle alpha autour de l'origine du repère. Or nous souhaitons réaliser une rotation autour du barycentre. Pour cela, nous effectuons d'abord une translation de la polyligne, du barycentre jusqu'à l'origine du repère, puis nous appliquons la matrice de rotation R. Une fois la rotation réalisée, nous effectuons la translation inverse pour repositionner la forme correctement.

Ainsi, si B est le barycentre de la polyligne de coordonnées (x_b, y_b), si T est une matrice de translation et R une matrice de rotation, la matrice finale M à appliquer à la figure est de la forme : $M = T(x_b, y_b) * R(\alpha) * T(-x_b, -y_b)$. Le script du programme est disponible en Annexe 11.

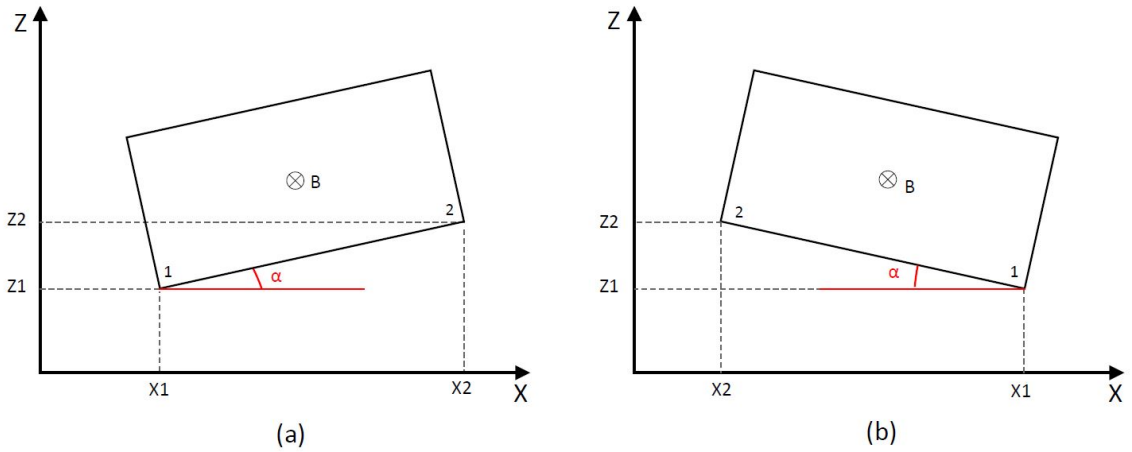


Illustration 36: (a) Cas n°1 : L'angle entre le côté à aligner et l'horizontale est positif dans le sens trigonométrique. (b) Cas n°2 : L'angle entre le côté à aligner et l'horizontale est positif dans le sens horaire. Dans les deux cas, la rotation est appliquée au point B (barycentre de la forme).

II.3.3. Le script d'alignement horizontal et vertical

L'objectif de ces fonctions est de forcer l'alignement de certains éléments avec un objet de référence choisi par l'utilisateur. Comme pour la fonction Rotation détaillée précédemment, nous nous plaçons dans le cas de figure où nous traitons des quadrilatères. L'utilisateur va alors pouvoir choisir selon quel côté de la forme de référence il souhaite réaliser l'alignement. L'illustration 37 présente les cas de figures rencontrés lors de l'alignement vertical de 2 polylignes (en vert et rouge) avec une polyligne de référence (en bleu). Le dx entre les deux objets est alors calculé par $dx = X_{ref} - X_{obj}$. Le vecteur de translation est alors calculé. Si $dx > 0$, la translation est appliquée au barycentre selon le vecteur (1,0,0). Si $dx < 0$, la translation est appliquée au barycentre selon le vecteur (-1,0,0). L'alignement horizontal repose sur le même raisonnement, mais avec le calcul de dz. Les 2 scripts sont disponibles en Annexe 12 et Annexe 13.

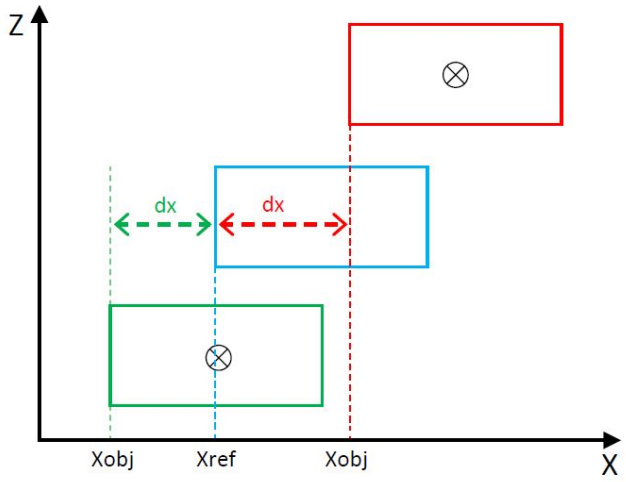


Illustration 37: Le côté gauche des formes verte et rouge est à aligner sur le côté gauche de la forme bleue.

Il est possible de contrôler visuellement la qualité des contours obtenus en les superposant sur le nuage. Cela nous permet de contrôler rapidement si des aberrations ont été commises. Il est difficile de développer des outils automatiques de contrôle entre le contour généré et le nuage de points. Il est plus aisé de quantifier l'écart entre un contour généré automatiquement et un contour généré manuellement. Mais en faisant cela, nous allons à l'opposé de l'objectif du traitement automatique. C'est pourquoi il est préférable avec cette méthode de se limiter à un contrôle visuel de chaque contour.

II.4. Extraction des points de contours

II.4.1. Triangulation adaptée

Dans cette partie, notre objectif va être d'extraire les points de contours des nuages obtenus à la suite de la segmentation du nuage principal suivant l'approche RANSAC. Le processus mis en place par Boulaassal (2010), repris par Bidino (2013) et expliqué page 25 de ce mémoire, repose sur la triangulation des points des segments plans et sur l'analyse de la longueur des triangles générés. Une autre méthode consiste à classifier les arêtes des triangles comme « frontières » ou non. On appelle « arête frontière », un côté n'appartenant qu'à un seul triangle (Illustration 38). Dans cette dernière méthode, les extrémités des arêtes frontières sont considérées comme étant les points de contours. Pour pouvoir appliquer cette méthode, les points doivent auparavant être projetés sur un plan afin que l'on puisse travailler en deux dimensions. Cette façon de procéder n'a pas été retenue par les auteurs précités, car ces derniers souhaitaient un résultat en trois dimensions. Si le passage d'un système à trois dimensions vers un système à deux dimensions est relativement simple, la réciproque est fausse, car il est difficile de conserver les relations de voisinage (Boulaassal, 2010).

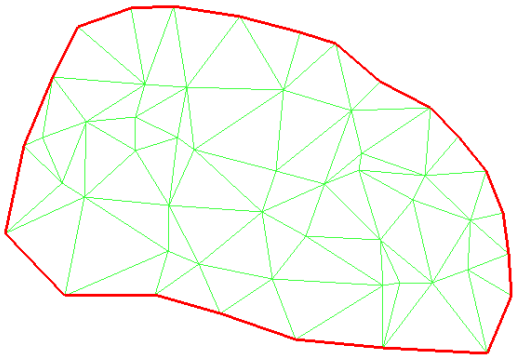


Illustration 38: Triangulation de Delaunay d'un semis de points.
Les arêtes rouges sont des arêtes frontières.

Dans notre cas, nous ne souhaitons pas revenir en trois dimensions après la détection et l'extraction des points de contours. En effet, lorsque nous travaillons en élévation, comme c'est le cas pour un plan de façade, le seul plan mathématique qui nous importe est le plan Oxz. C'est pourquoi nous allons recourir à la méthode reposant sur la détection des arêtes frontières. Cela nous permet d'automatiser un peu plus le processus, car nous n'avons pas à choisir de valeur seuil pour la longueur minimale des côtés des triangles extraits, comme c'est le cas à la figure 19.b.

Dans un premier temps, nous devons projeter chaque point du nuage sur le plan Oxz. Pour ce faire, nous calculons les cordonnées du point projeté de la façon suivante : Soit P le plan mathématique d'équation $ax+by+cz+d=0$ et $\vec{n}$ un vecteur normal du plan tel que $\vec{n}=\begin{bmatrix} a \\ b \\ c \end{bmatrix}$. Si nous appelons $M(x_M, y_M, z_M)$ le point du nuage et $H(x_H, y_H, z_H)$ le projeté orthogonal de M sur le plan P, alors il existe un réel k tel que $\overrightarrow{PM}=k\vec{n}$, puisque les deux vecteurs sont colinéaires. Nous obtenons donc un système de trois équations : $\begin{cases} x_H = x_M - ka \\ y_H = y_M - kb \\ z_H = z_M - kc \end{cases}$. Puisque le point H est également un point du plan P, nous pouvons écrire que $ax_H+by_H+cz_H+d=0$. Si nous substituons les coordonnées de H par l'expression des trois équations nous obtenons la relation suivante $a(x_M-ka)+b(y_M-kb)+c(z_M-kc)+d=0$ qui nous permet de calculer la valeur de k. A partir de cette étape, nous pouvons calculer les coordonnées de H.

Une fois les points projetés, nous appliquons une triangulation de Delaunay à l'ensemble du semis, tel que décrit dans la partie I.3.2. Cependant, une condition doit être ajoutée lors de cette étape. En effet, la triangulation de Delaunay telle que décrite précédemment va trianguler tous les points du semis. Or nous souhaitons que les ouvertures comme les portes et les fenêtres ne soient pas maillées. Nous rajoutons donc une condition de longueur maximale sur les côtés d'un triangle. Si un des côtés du triangle est supérieur à la valeur maximale, alors ce triangle est supprimé du maillage. La figure 39 illustre le résultat de la triangulation sur une pierre de parement du nuage n°1bis comportant 9 966 points. Cette étape a généré 19 606 triangles.

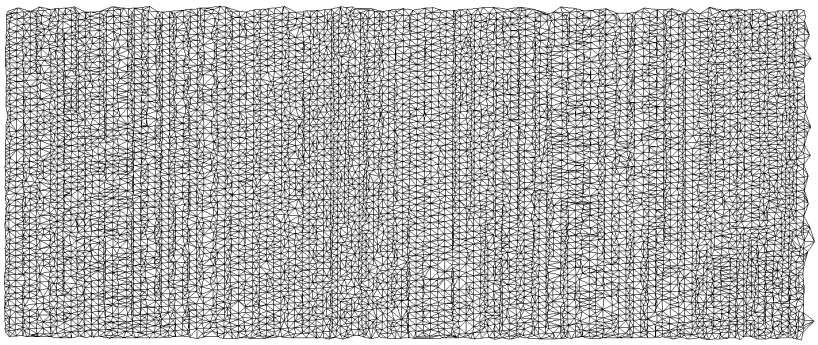


Illustration 39: Triangulation de Delaunay d'une pierre de parement du nuage n°1bis

Si la qualité du résultat est satisfaisante, le nombre de triangles créés l'est beaucoup moins. En effet, si nous considérons que tous les points sont maintenant en deux dimensions, une forme géométrique tel qu'une pierre de parement doit pouvoir être décrite avec un nombre de triangles beaucoup moins important. Cela va notamment influencer le temps de traitement de la détection des points de contours. Le principe est de réaliser une simplification en réduisant le nombre de sommets à l'intérieur du maillage. Un test est effectué après chaque suppression de sommet pour valider ou non l'opération car une suppression hasardeuse et mal contrôlée peut entraîner la déformation du maillage (H. Hoppe, 1996). Ce test consiste à mesurer l'écart entre le nouveau maillage et celui à simplifier. Si l'écart est inférieur à un seuil prédéfini, la suppression du sommet i est validée et l'algorithme va tester le sommet i+1. La figure 40 illustre ce principe appliqué à la pierre de parement de l'illustration 39. Nous avons réalisé cette simplification sous le logiciel 3DRESHAPER en paramétrant un écart maximum entre les deux maillages de 1 mm. A l'issu de la modification, le

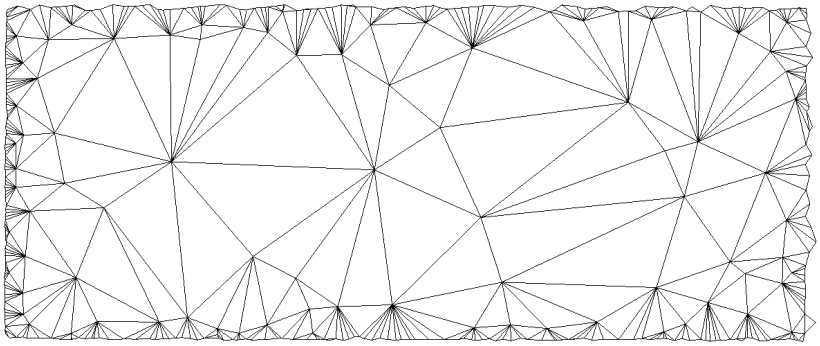


Illustration 40: Triangulation simplifiée d'une pierre de parement du nuage n°1 bis

maillage est constitué de 405 sommets et de 484 triangles. Nous avons ainsi pu réduire le nombre de triangles de 97,5%, ce qui est très important pour la vitesse des traitements à suivre.

II.4.2. Détection des arêtes frontières

Une fois que nous possédons une triangulation précise et allégée du segment plan, nous allons pouvoir détecter les points de contours à l'aide d'un script python. Pour pouvoir être correctement lu par l'algorithme, un maillage de *n* triangles doit être enregistré au format texte de la manière suivante :

$$
\begin{array}{ccc}
X_{T1S1} & Y_{T1S1} & Z_{T1S1} \\
X_{T1S2} & Y_{T1S2} & Z_{T1S2} \\
X_{T1S3} & Y_{T1S3} & Z_{T1S3} \\
X_{T2S1} & Y_{T2S1} & Z_{T2S1} \\
X_{T2S2} & Y_{T2S2} & Z_{T2S2} \\
X_{T2S3} & Y_{T2S3} & Z_{T2S3} \\
\ldots & \ldots & \ldots \\
X_{TnS3} & Y_{TnS3} & Z_{TnS3}
\end{array}
$$

où T1S1 signifie Triangle 1 Sommet 1 et T2S3 signifie Triangle 2 Sommet 3. Si l'utilisateur a réalisé la triangulation sur un logiciel de modélisation 3D tel que 3DRESHAPER, Meshlab... le maillage doit être exporté au format STL ASCII dont l'extension est .stl. Le programme a été écrit pour gérer ce type d'extension, couramment utilisé dans la modélisation d'objet 3D. La structure du format de chaque triangle est présentée par la figure 41. Nous nous intéressons uniquement aux sommets des triangles, dont les coordonnées sont présentes aux lignes débutant par le mot « vertex ».

```
facet normal 0 1 0
  outer loop
    vertex -19.1214866648443 1.96448099500815 4.19675778967101
    vertex -19.1028385172613 1.96448099500815 4.19498777013976
    vertex -19.1284008036383 1.96448099500815 4.12079810813773
  endloop
```

Illustration 41: Description d'un triangle au format STL ASCII

Le programme python va directement récupérer les coordonnées des sommets de chaque triangle et les classer selon la même forme que la matrice présentée ci-dessus. Ensuite, une matrice de n lignes et 7 colonnes est créée où n = 3*nombre_de_triangles. Cette matrice décrit chaque côté de chaque triangle de la façon suivante :

$$\begin{bmatrix} X_{Point1} & Y_{Point1} & Z_{Point1} & X_{Point2} & Y_{Point2} & Z_{Point2} & Distance_{Point1-Point2} \\ X_{Point1} & Y_{Point1} & Z_{Point1} & X_{Point3} & Y_{Point3} & Z_{Point3} & Distance_{Point1-Point3} \\ X_{Point2} & Y_{Point2} & Z_{Point2} & X_{Point3} & Y_{Point3} & Z_{Point3} & Distance_{Point2-Point3} \end{bmatrix}$$

Lorsque la matrice est créée, nous la parcourons et étudions la dernière colonne contenant la distance entre deux sommets. Si cette distance est présente plusieurs fois dans la matrice, cela signifie que ce côté appartient à 2 triangles. Dans le cas contraire, les coordonnées des deux extrémités de ce côté sont rangées dans un tableau, sous réserve de ne pas y être déjà. La figure 42 nous permet d'analyser le résultat obtenu sur la façade du nuage n°2. Nous remarquons que tous les points de contours ont été correctement détectés. Le traitement est surtout pertinent pour la classe « Façade » (représentée par les points noirs) et « Appui de fenêtre » (représentée par les points rouges).

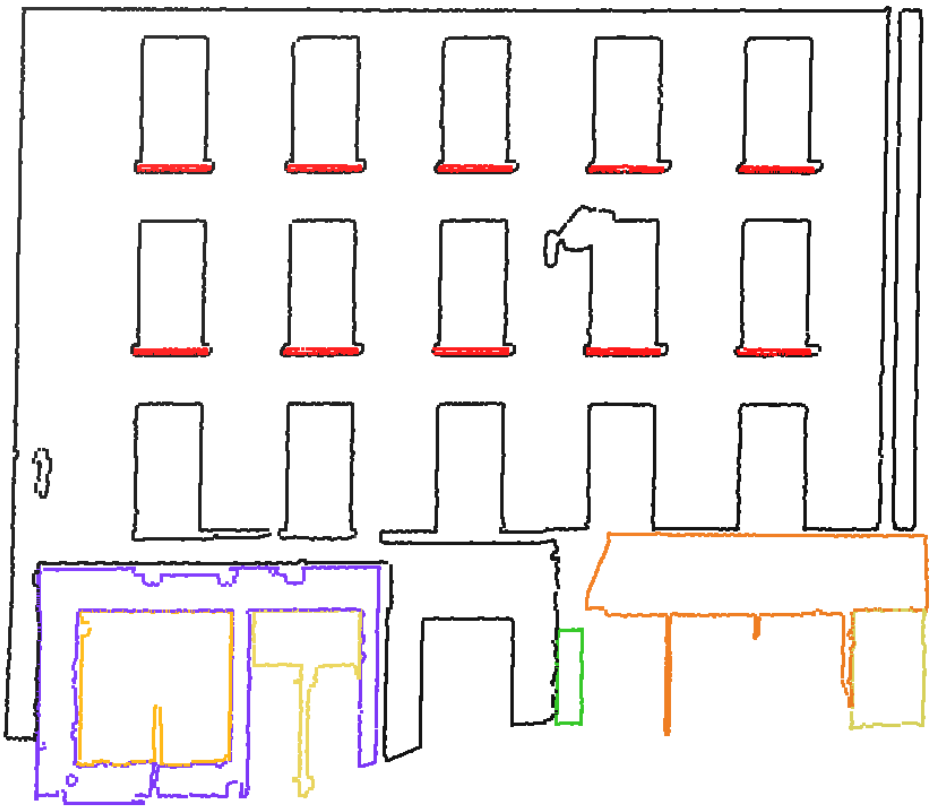


Illustration 42: Résultat de l'application de l'algorithme de détection de contour appliqué au nuage de la façade n°2

II.4.3. Analyse des résultats

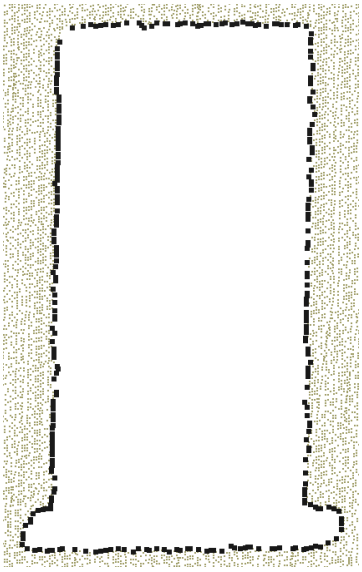


Illustration 43: Points de contours d'une fenêtre de la façade n°2

La figure 43 met en avant la qualité de détection des contours d'une fenêtre issue de la façade n°2. Nous pouvons voir que les coins, même arrondis, sont correctement extraits, mais pas de façon optimale car certains points mériteraient d'être extraits en tant que points de contour. C'est le cas par exemple du coin arrondi situé en haut à gauche de la figure 43. Cette légère défaillance, déjà rencontrée par Boulaassal (2010), est due à la qualité du maillage. Pour la corriger, il faudrait diminuer la valeur maximale que peut prendre un côté de triangle. Mais une valeur trop faible risquerait d'entraîner l'apparition de trou dans la surface du maillage.

II.5. Vectorisation des points de contours

II.5.1. Détection et classification des droites de contours

Nous connaissons à présent la façon d'obtenir l'ensemble des points de contours d'un segment plan. Pour obtenir un modèle exploitable, il nous faut modéliser les lignes de contours. Nous ne pouvons pas nous contenter de relier les points entre eux, les uns à la suite des autres. Cela générerait un modèle déformé qui n'aurait aucun lien avec la réalité du bâtiment. Si nous souhaitons respecter cette condition, nous devons trouver un moyen de générer des contours à géométrie simple tout en prenant en compte les points de contours. C'est pourquoi nous allons avoir recours une fois de plus à l'approche RANSAC, mais nous l'utiliserons cette fois-ci dans un espace en deux dimensions. Ce procédé a déjà été expliqué dans la partie I.3.1.5. L'algorithme va alors pouvoir détecter les ensembles de points permettant de générer les meilleures droites présentes à l'intérieur du semis.

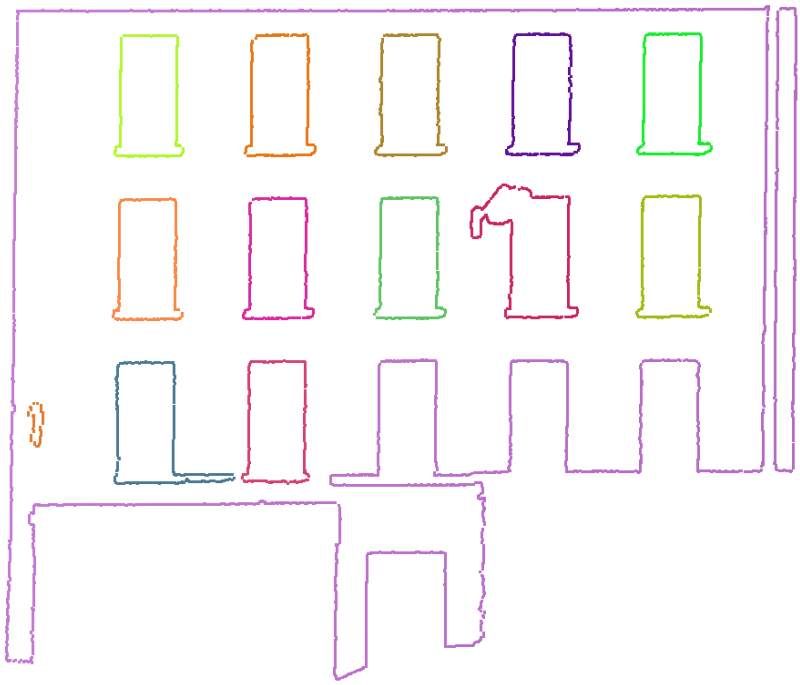


Illustration 44: Résultat de l'algorithme de croissance de région appliqué à l'ensemble des points de contours composant la façade du nuage n°2 - Distance seuil de 10 centimètres.

Si nous appliquons l'algorithme directement à l'ensemble des points de contours, les arêtes des fenêtres

superposées vont certainement être détectées comme appartenant à la même droite et il sera plus délicat de calculer des intersections. Pour améliorer cette phase de traitement, nous avons mis en place deux étapes supplémentaires. La première étape consiste à segmenter l'ensemble des points en fonction des éléments géométriques qui leur sont propres. Nous allons ainsi segmenter le semis en appliquant un algorithme de croissance de région. Ce dernier va prendre un point quelconque du nuage et va calculer la distance le séparant de tous les autres points. Les points dont la distance est inférieure à un seuil prédéfini sont ajoutés à la région. Ce procédé est ensuite réitéré pour chaque nouveau point ajouté. Pour ce faire, l'ensemble des points est d'abord rangé dans un tableau de la forme : $T = \begin{bmatrix} X_1 & Y_1 & Z_1 & False_1 & False_2 \\ X_2 & Y_2 & Z_2 & False_1 & False_2 \\ \dots & \dots & \dots & \dots & \dots \\ X_n & Y_n & Z_n & False_1 & False_2 \end{bmatrix}$ où $False_1$ représente une variable booléenne dont la valeur est négative au début du traitement et devient positive lorsque le point est ajouté à une région. De façon similaire, $False_2$ représente une variable booléenne négative qui devient positive lorsque la distance entre ce point et tous les points n'appartenant pas à une région est calculée. La région grossit jusqu'à ce que tous les points qu'elle contient aient servi de base pour le calcul de la distance seuil. Autrement dit, la région grossit jusqu'à ce qu'il n'y ait plus de variable booléenne négative dans la dernière colonne. La figure 44 représente le résultat de ce traitement obtenu sur les points de contours du nuage n°2 pour une distance seuil de 10 centimètres.

Maintenant que chaque élément architectural est correctement décomposé, nous allons pouvoir appliquer notre algorithme de détection des droites. Ce dernier va également classifier les droites en trois catégories : droites horizontales, droites verticales et droites inclinées. Cette dernière partie va nous permettre de détecter les arcs présents au sein du nuage. En effet, un arc sera décomposé par l'algorithme RANSAC en une succession de petit segment droit. C'est pourquoi nous appliquerons les règles suivantes pour détecter les points appartenant à des arcs. Des points sont classifiés en tant qu'arc si et seulement si :

→ Ils appartiennent à une droite inclinée.

→ Ils sont voisins d'une autre droite inclinée.

→ Le semis extrait comporte un nombre de points relativement faible.

L'illustration 45 représente le résultat de l'algorithme appliqué au semis de points obtenu à l'étape précédente et présenté à la figure 44. Le zoom sur l'ombre laser générée par un lampadaire nous permet d'observer la façon dont les courbes sont détectées avec RANSAC. En effet, nous constatons que les ensembles de points extraits à cet endroit respectent bien les trois conditions énoncées ci-dessus.

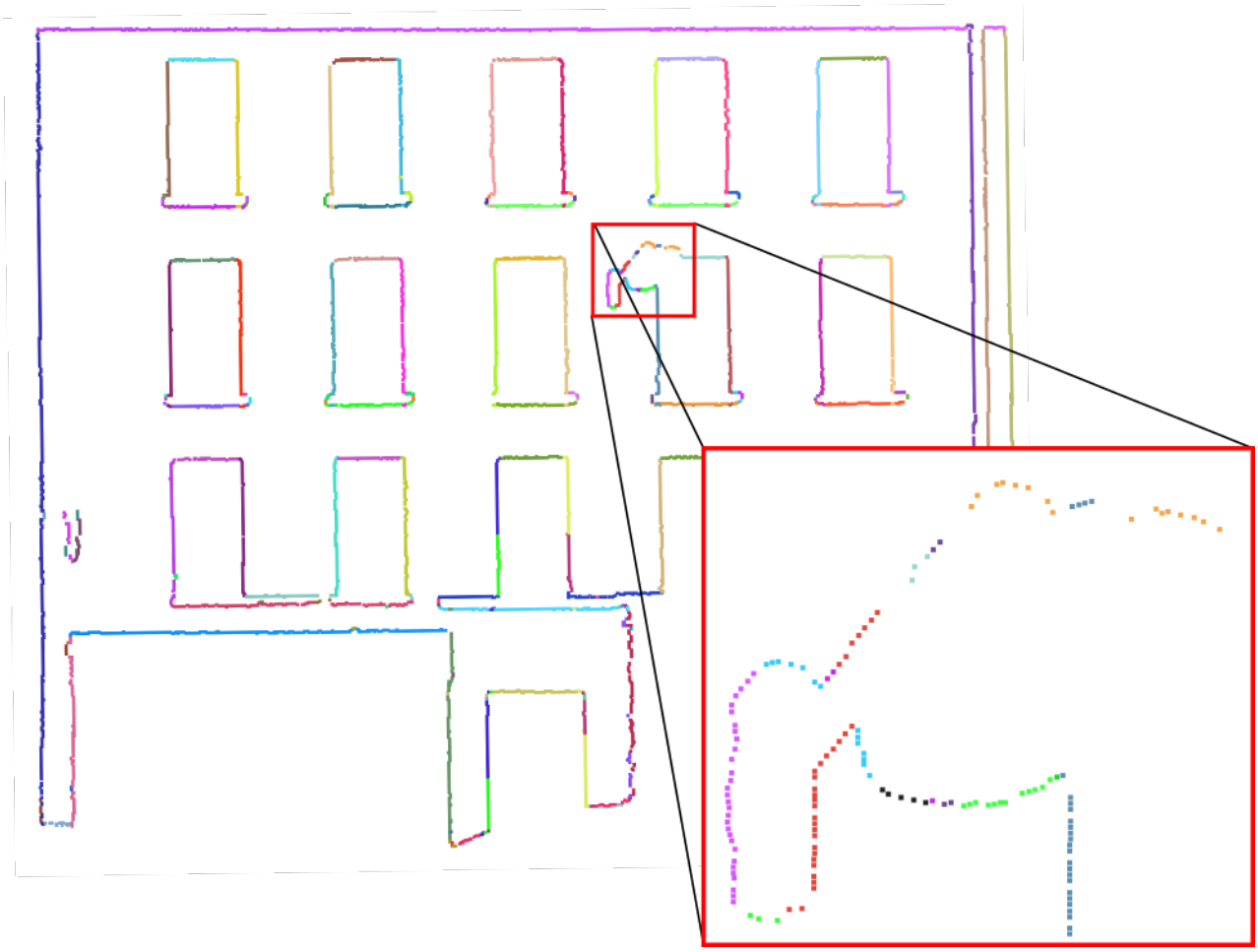


Illustration 45: Résultat de l'algorithme RANSAC appliqué à chacune des régions détectées précédemment. Agrandissement du lampadaire (encadré rouge). La classification n'a pas été effectuée.

Avant de pouvoir les classifier, nous allons devoir calculer les paramètres de chacune des droites. La droite qui a servi de calcul pour l'approche RANSAC n'est pas satisfaisante parce que ses paramètres sont calculés à partir de deux points. Nous allons utiliser le principe des moindres carrés ordinaire. Cette méthode consiste à calculer la droite dont la somme des carrés des écarts avec les points est minimale. Mathématiquement, cela signifie que si nous nous situons dans le plan Oxy et que les paramètres de la droite à calculer sont le coefficient directeur a et l'ordonnée à l'origine b tel que $y = ax+b$, alors $a=\frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$ et $b=\bar{y} - a\bar{x}$ où x_i et y_i sont les coordonnées de chaque point du semis, $\bar{x}$ (respectivement $\bar{y}$) représente la moyenne des x (respectivement des y) des points.

Pour chaque segment droit, nous obtenons donc la droite s'ajustant au mieux aux données. Nous allons maintenant pouvoir classifier toutes les droites en tant qu'élément horizontal, vertical ou incliné. Pour ce faire, nous allons calculer l'angle entre la droite et l'axe vertical. Théoriquement, une droite verticale va générer un écart angulaire de 0 degré tandis qu'une droite horizontale va retourner un écart angulaire de 90 degrés. En pratique, les droites détectées ne sont pas parfaitement orientées. C'est pourquoi, nous tolérons un écart de 5 degrés. Ainsi, si l'écart angulaire θ entre la droite et l'axe vertical est inférieur à 5 degrés, alors la droite sera considérée comme verticale. De la

même façon, si θ est compris entre 85 et 95 degrés, la droite sera considérée comme horizontale. Dans les autres cas, elle sera classifiée en tant que 'Droite inclinée'.

Dans ce paragraphe nous allons résumer l'ensemble des étapes qui ont précédé. Nous avons dans un premier temps segmenté notre nuage de points en une multitude de plans en utilisant l'algorithme RANSAC en trois dimensions. Pour chacun des plans, nous avons détecté les points de contours en calculant une triangulation allégée. Ces points de contours ont ensuite été traités par un algorithme de croissance de région afin de séparer les différents éléments architecturaux. Sur chacun des éléments, nous avons ensuite classifié les droites en fonction de leur orientation en appliquant l'algorithme RANSAC en deux dimensions.

II.5.2. Intersection des primitives géométriques

Comme dans Boulassaal (2010), nous devons donc à présent calculer l'intersection des droites. Cette opération est délicate, car nous ne devons pas calculer toutes les intersections possibles, mais uniquement celles où les conditions suivantes sont respectées :

→ Les deux droites appartiennent au même élément architectural

→ La distance entre les deux extrémités des droites à intersecter est inférieure à un seuil prédéfini

En pratique, nous écrivons dans une matrice les coordonnées de chaque extrémité. Nous calculons ensuite pour chaque extrémité de chaque droite la distance la séparant des autres extrémités. Nous supposons alors que l'extrémité dont la distance est la plus faible correspond à une droite à intersecter. La figure 46 illustre ce raisonnement dans le cas de la vectorisation des points de contours d'une fenêtre. Si nous prenons le point n°2 en exemple, et que nous calculons la distance le séparant de tous les autres, nous nous apercevons facilement que le point le plus proche est le n°3 et que les deux droites dont sont issus les points 2 et 3 doivent être reliées. Cette intersection est symbolisée par le point vert.

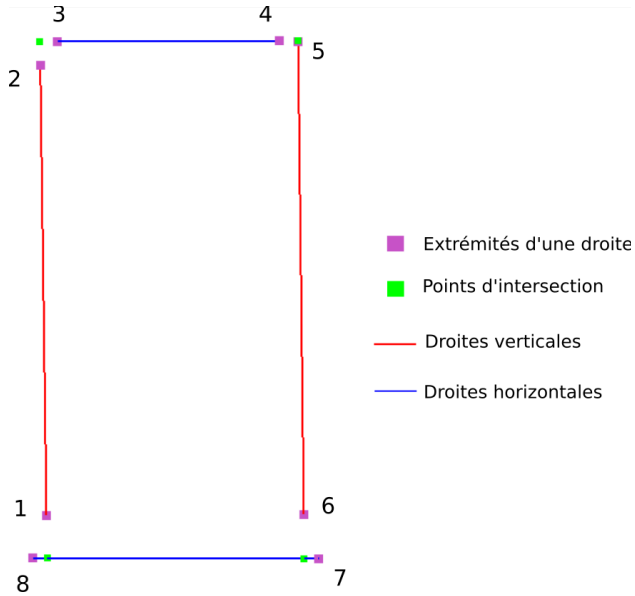


Illustration 46: Explication de la méthode d'intersection appliquée à une fenêtre simple

Nous obtenons finalement la modélisation des éléments architecturaux par le biais de primitives géométriques. La détection d'éléments similaires, déjà évoquée page 43, soulève une nouvelle fois la problématique de la finition esthétique du plan. En effet, dans la majorité des cas, les fenêtres présentes sur une façade sont similaires. La représentation automatique de ces éléments peut entraîner des écarts de l'ordre de quelques millimètres à quelques centimètres ainsi qu'un défaut d'alignement. C'est pourquoi nous devons une fois encore modifier les contours que nous venons d'obtenir. Nous utilisons pour cela les scripts développés dans la partie II.3. en langage JavaScript pour le logiciel 3DRESHAPER. Bien que cela n'ait pas pu être réalisé durant le temps imparti pour ce mémoire, les programmes développés en JavaScript peuvent rapidement être adaptés en Python au cas où nous ne souhaiterions pas dépendre de l'interface du logiciel 3DRESHAPER.

II.5.3. Analyse des résultats

Afin de pouvoir évaluer les résultats obtenus, nous avons dessiné manuellement le plan de façade du nuage de points n°2. Pour ce faire, nous avons suivi la méthode employée actuellement par le cabinet et décrite à la partie I.2.4. (Chaîne de traitement des données au sein du cabinet). Le plan de façade est consultable en Annexe 3. Pour réaliser l'évaluation, nous avons superposé le modèle généré automatiquement sur celui créé manuellement et nous avons attribué des numéros à des points caractéristiques de la façade (Illustration 47). Si P1 est un point caractéristique de la façade réalisée à la main, alors P1' correspond au même point de contrôle, mais sur le modèle obtenu automatiquement. Nous affichons alors les écarts mesurés entre les points homologues dans le tableau présent en Annexe 7 et Annexe 8.

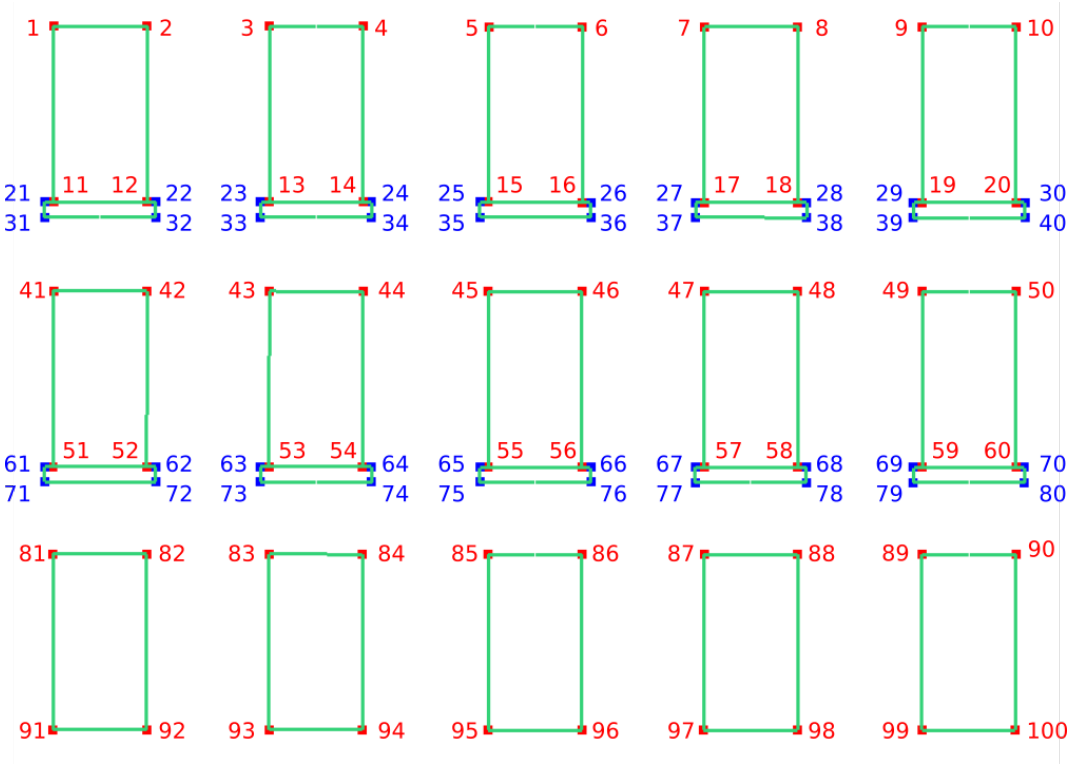


Illustration 47: Répartition des points de contrôle sur les fenêtres et les appuis de la façade n°2

Les tableaux de l'Annexe 7 présentent les écarts obtenus entre les points de contrôle situés sur les encadrements des fenêtres. Le résultat est très satisfaisant pour les points situés sur le haut des fenêtres (points : 1 → 10 // 41 → 50 // 81 → 90), puisque l'écart moyen est de 1,3 centimètre avec un écart type de 2,6 millimètres. Cela signifie que les valeurs sont bien centrées autour de la moyenne. En revanche, pour les points situés sur le bas des fenêtres (points : 11 → 20 // 51 → 60 // 91 → 100) la moyenne est de 4,1 centimètres. Mais nous remarquons que l'écart type pour cette catégories est presque d'un centimètre, ce qui signifie que les valeurs obtenues sont beaucoup plus dispersées. En regardant le détail du tableau, nous remarquons que la moyenne est tirée vers le haut par les mauvais résultats de la détection des points 91 → 100. La faible qualité de l'extraction des points bas est due à la présence d'ombres lasers causées par les appuis de fenêtre. Inversement, nous voyons que lorsque les parties sont correctement acquises, les résultats sont tout à fait satisfaisants.

Les tableaux de l'Annexe 8 présentent les écarts obtenus entre les points de contrôle situés sur les appuis des fenêtres. La moyenne des écarts, de l'ordre de 3 centimètres, peut paraître assez élevée, mais cela est encore une fois dû aux ombres lasers. Il faut souligner qu'une imprécision porte également sur le modèle réalisé manuellement. En tenant compte de ces paramètres, le résultat obtenu sur les appuis de fenêtres est satisfaisant.

Nous avons également appliqué l'ensemble des algorithmes présentés précédemment sur le nuage n°1. Le résultat obtenu est consultable en Annexe 6. Visuellement, la représentation nous semble bonne et nous remarquons que de nombreux éléments ont pu être détectés. Toutefois, des erreurs apparaissent, par exemple sur les pierres de parement qui sont organisées en bandeau horizontal. En effet, le côté inférieur de ces pierres est en réalité composé d'un arc et à l'heure actuelle, nos programmes traitent mal ce cas de figure. En superposant ce modèle avec celui réalisé manuellement (cf. Annexe 2), nous observons que les écarts sur les pierres de parement organisées en bandeau vertical sont d'environ 1 centimètre, ce qui est très satisfaisant. Globalement, la qualité des résultats est nettement supérieure que ceux obtenus après le traitement de la façade n°2. Cela s'explique par l’absence d'ombre laser dans le nuage n°1.

Conclusion et ouverture

A travers cette étude, une première approche des traitements automatiques au sein d'un cabinet de géomètre a pu être réalisée. Elle a notamment permis de constater, par le biais d'un traitement appliqué à des façades de bâtiments, les possibilités ainsi que les limites d'une telle application. Notre objectif initial était, à partir d'un nuage de points d'une façade, de détecter les contours des éléments remarquables et de générer automatiquement leur représentation par un modèle vectoriel en deux dimensions afin de faciliter une partie de la création de plan d'élévation dans un logiciel de CAO.

Pour cela nous avons d'abord segmenté le nuage de points en utilisant le principe de l'algorithme RANSAC. La primitive géométrique évaluée est le plan, parce que les surfaces planes sont les plus présentes sur une façade. Cet algorithme a été préféré à d'autres (présentés dans la partie I.3.1.) à cause d'une part, de sa robustesse face au bruit, et d'autre part pour les adaptations que nous pouvons intégrer à l'algorithme afin de ne pas effectuer de calculs inutiles et ainsi, de réduire le temps de calcul. Plusieurs versions d'algorithmes ont été présentées dans la partie II.2.1. pour les comparer en matière de rapidité, d'options et de résultats. C'est finalement la méthode que nous avons nous-même implémentée qui présente les résultats les plus intéressant. Cette méthode a été spécialement adaptée pour le traitement d'une façade. En effet, seuls les plans inclinés et parallèles à la direction de la façade sont pris en compte pour le calcul des meilleurs inliers. Cette façon de faire permet de réduire considérablement le temps de calcul.

La méthode mise en place pour détecter les points de contours repose sur le principe d'une triangulation en deux dimensions sur laquelle nous allons effectuer une analyse sur les côtés des triangles, analyse qui va nous permettre d'extraire les arêtes frontières. Les extrémités de ces dernières sont alors classifiées comme points de contour. Si les résultats obtenus par ce système sont tout à fait satisfaisants, ils ont également permis de mettre en évidence les limites d'un tel traitement. En effet, les masques présents lors de la phase d'acquisition vont provoquer des ombres lasers dont les contours vont être détectés par nos traitements. Lorsque ces ombres affectent des éléments répétitifs, nous avons mis en place un outil de duplication permettant de contourner ce problème. En revanche, aucune solution n'a pour le moment été trouvée dans le cas où l'acquisition d'un élément non répétitif est affectée par un masque. Cela met en évidence les limites des traitements automatiques dans les nuages de points, surtout dans le cas des acquisitions réalisées en extérieurs où le nombre d'éléments pouvant générer des ombres peut être important.

Nous nous sommes ensuite appuyé sur les points de contours pour générer des modèles vectoriels en deux dimensions. Pour cela, une seconde utilisation de la méthode RANSAC a été mise en œuvre. Cette fois-ci, nous ne l'avons pas appliquée pour détecter des plans (3 dimensions), mais pour détecter des droites (2 dimensions). Nous avons enfin utilisé un algorithme de calcul des intersections des droites précédemment extraites afin d'obtenir un modèle vectoriel fiable et simple. Les résultats obtenus sont bons pour des droites mais doivent encore être améliorés pour la représentation des arcs.

La démarche de ce traitement a été présentée devant les responsables de la SELAS Daniel Legrand et accueilli favorablement. A la suite de cela, deux nouveaux objectifs ont été formulés. Il s'agit dans un premier temps, de former les techniciens à cette nouvelle méthode et de mettre en place une

série de protocoles à appliquer pour obtenir des résultats exploitables. Dans un deuxième temps, il a été décidé de poursuivre ce travail de développement et de l'appliquer à d'autres types de livrable. En ce sens, nous avons déjà réalisé des tests prometteurs pour la génération automatique d'un plan d'intérieur à partir d'un nuage de points. Il en est de même pour la création de plans de coupe. D'une manière générale, nous constatons qu'il est possible d'étendre cette façon de faire à l'ensemble des livrables 2D dont le relevé a été réalisé par un scanner laser.

Néanmoins, des limites existent et des précautions sont à prendre. Au niveau du temps de calcul, la phase de segmentation peut se révéler extrêmement longue pour un résultat décevant si les paramètres d'entrée sont mal maîtrisés. De même, le traitement se présente sous la forme d'une multitude de programmes python et javascript dont l'utilisation n'est pas forcément instinctive pour toute personne étrangère à ces langages. C'est pourquoi un projet d'intégration de ces algorithmes dans une application intuitive à été mise à l'étude. Cette application devra notamment comporter une interface graphique permettant à l'utilisateur de visualiser directement le résultat des différents traitements. De plus, une nouvelle approche basée sur l'utilisation de voxels (Aijazi *et al*, 2012) pourra être envisagée dans l'optique de réduire significativement le temps de calcul.

Enfin, la validation des traitements a été effectuée à partir de plans réalisés manuellement mais aucune solution numérique n'a été trouvée pour valider ou non la création automatique des contours. Cela soulève la question de la qualité des données, surtout si nous prenons en compte qu'un géomètre à le devoir de garantir la précision de ses rendus. En attendant la mise en place de protocoles de contrôle, il est vivement recommandé à tous les utilisateurs de cette méthode d'être vigilants et de vérifier la qualité des résultats en les superposant au nuage de points initial.

Bibliographie

A. Aijazi, P. Checchin, L. Trassoudaine, 2012. Segmentation et classification de points 3D obtenus à partir de relevés laser terrestres : une approche par super voxels [en ligne]. In 18e édition sur la Reconnaissance des Formes et l'Intelligence Artificielle. 8 pages.

K. Ait El Kadi, D. Tahiri, E. Simonetto, I. Sebari, 2014. Modélisation 3D des façades de bâtiments des anciennes Médina [en ligne]. Revue marocaine des sciences agronomiques et vétérinaires, Volume 1, pages 5-11.

S. Bidino, 2013. Étude de l'extraction automatique de coupe à partir de nuages de points [en ligne]. Sciences de l'ingénieur. Institut National des Sciences Appliquées. 70 pages.

D. Borrmann, J. Elseberg, K. Lingemann, 2011. The 3D Hough Transform for Plane Detection in Point Clouds : A review and a New Accumulator Design [en ligne]. 3D Research, Volume 2, page 1-13.

P. Boucher, 2016. Reconnaissance automatique des contours de pierres sur nuages de points 3D acquis pas laser scanner terrestre [en ligne]. Sciences de l'ingénieur. École Supérieure des Géomètres et Topographes. 66 pages.

H. Boulaassal, T. Landes, P. Grussenmeyer, 2009. Automatic extraction of planar clusters and their contours on building façades recorded by terrestrial laser scanner. International journal of architectural computing [en ligne], vol. 7, Issue 1, pages 1-20.

H. Boulaassal, 2010. Segmentation et modélisation géométriques de façades de bâtiments à partir de relevés laser terrestres [en ligne]. Sciences de l'ingénieur. Institut National des Sciences Appliquées. 212 pages.

C. Briese, 2006. Structure line modelling based on terrestrial laserscanner data [en ligne]. ISPRS Symposium, Dresden, Commission V – Image Engineering and Vision Metrology.

C. Briot, C. Vacquant, édition de 1856. Éléments de géométrie (Application). Page 1.

H. Edelsbrunner, E.P. Mücke, 1994. Three-dimensional alpha shapes. ACM Trans. Graphics 13, pages 43-72.

M. A. Fischler and R. C. Bolles, 1981. Random sample consensus : A paradigm for model fitting with applications to image analysis and automated cartography [en ligne]. Communications of the ACM, Volume 24, pages 381-395.

J. Hernandez, B. Marcotegui, 2009. Point cloud segmentation towards urban ground modeling [en ligne]. IEEE Joint Urban Remote Sensing Event. 5 pages.

M. Laloux, 2015. Sur la création des enveloppes concaves et les divers moyens d'un parvenir [en ligne]. Disponible sur : www.portailsig.org. Consulté le 28/03/18.

T. Landes, H. Boulaassal, P. Grussenmeyer, 2011. Les principes fondamentaux de la lasergrammétrie terrestre : acquisition, traitement des données et applications [en ligne]. Revue XYZ, Volume 129, pages 25-38.

H. Macher, 2017. Du nuage de points à la maquette numérique de bâtiment : reconstruction 3D semi-automatique de bâtiments existants [en ligne]. Sciences de l'ingénieur. Institut National des Sciences Appliquées. 166 pages.

R. Schnabel, R. Wahl, R. Klein, 2007. Efficient RANSAC for point-cloud shape detection [en ligne]. Computer Graphics Forum, Volume 26, pages 214-226.

E. Simonetto, 2018. Traitement des nuages de points. École Supérieure des Géomètres et Topographes. 66 diapositives.

G. Vosselman, B.G.H. Gorte, G. Sithole, T. Rabbani, 2004. Recognising structure in laser scanner point clouds. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences [en ligne], Volume 46, pages 33- 38.

M. Wang, Y. Tseng, 2004. LIDAR data segmentation and classification based on octree structure [en ligne], In 20th International Society for Photogrammetry and Remote Sensing (ISPRS) Congress, 12-23 July 2004, Istanbul, Turquie. Édité par O. Altan. ISPRS Archives, Volume 35, pages 308-313.

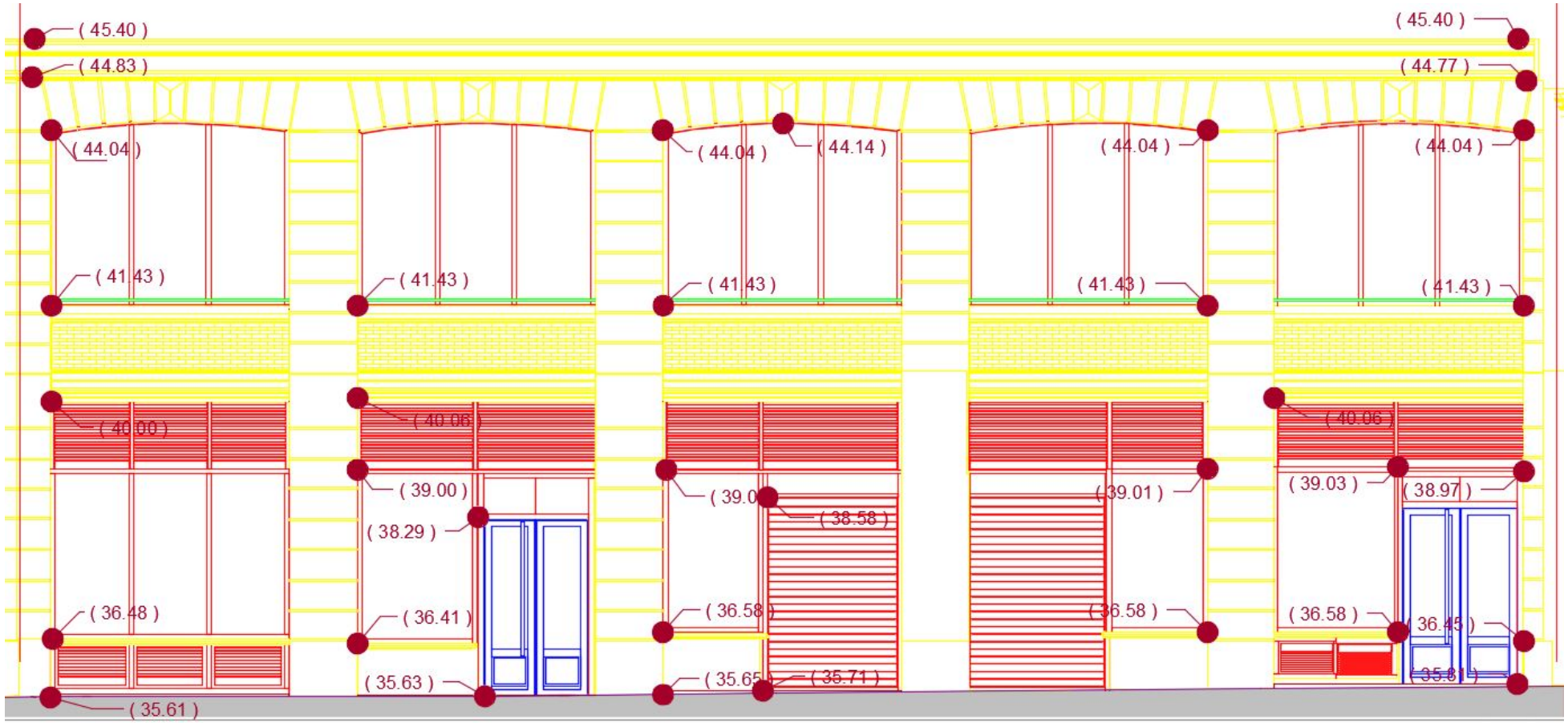
C. Weber, S. Hahmann, H. Hagen, 2010. Sharp feature detection in point clouds [en ligne]. Shape Modeling International Conference, pages 175 – 186.

Table des annexes

Note : Les scripts ont été ajoutés en annexes pour mettre en avant la partie programmation, qui a représenté une part conséquente du temps de travail. Néanmoins, la présentation n'a pas été optimisée pour la lecture et des erreurs dans les codes peuvent être présentes. C'est pourquoi, nous conseillons aux lecteurs qui voudraient s'inspirer de ces algorithmes, de se référer en priorité aux pseudo-codes présents dans le corps du mémoire.

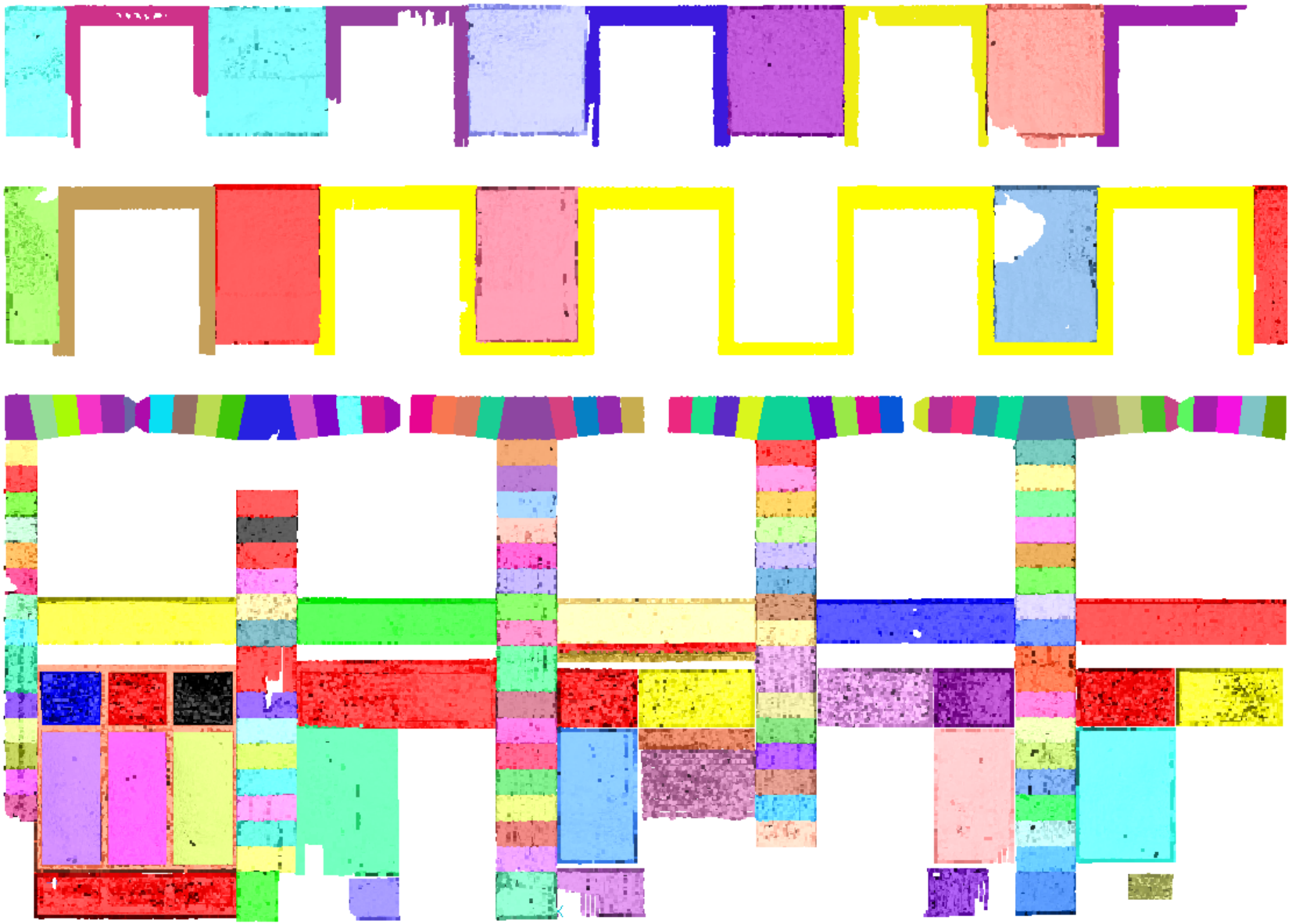
Annexe 1
Modèle qualité d'un plan de façade au sein de la SELAS Daniel Legrand

Annexe 2
Plan d'élévation de la façade n°1 réalisé manuellement

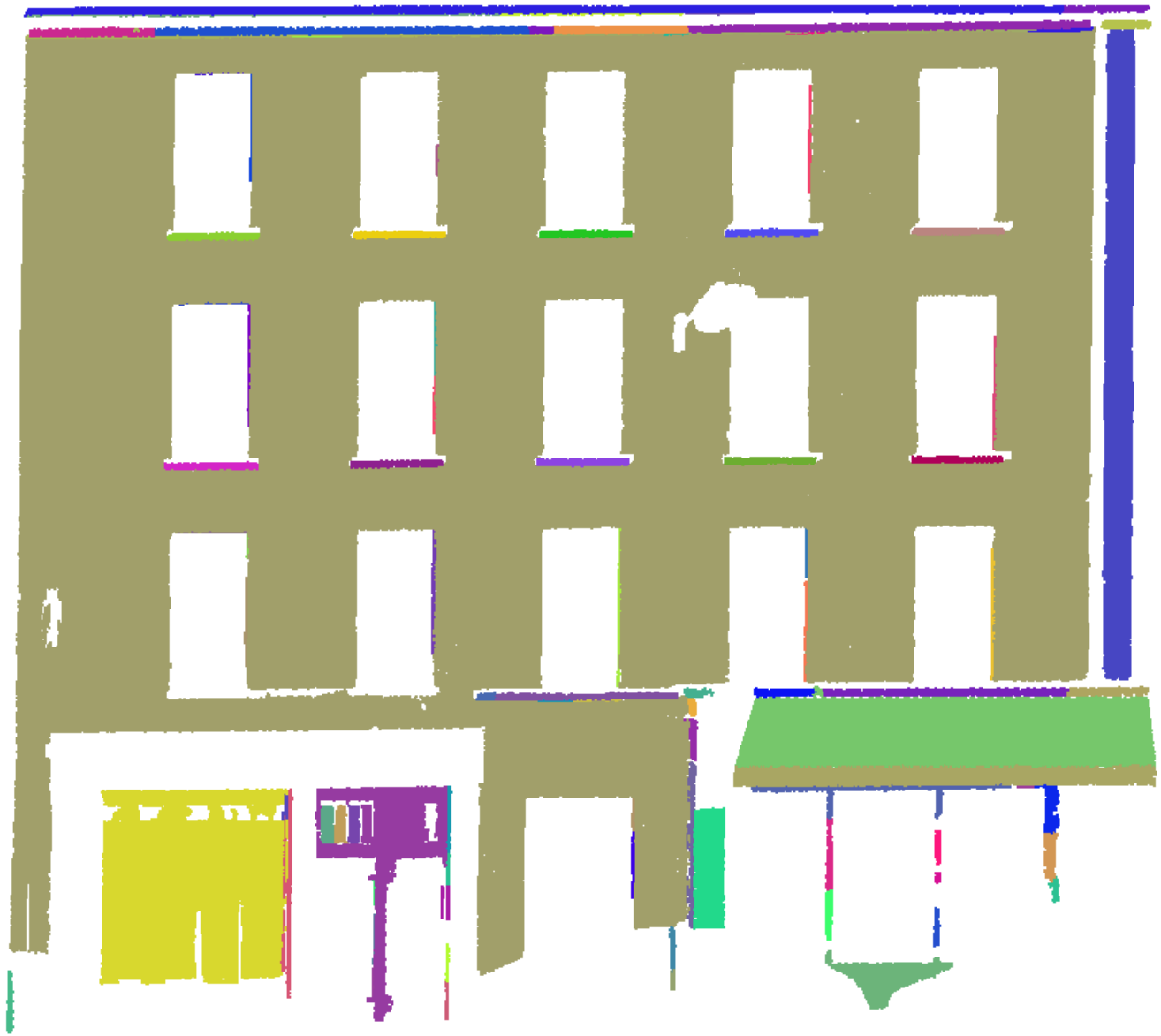


Annexe 3
Plan d'élévation de la façade n°2 réalisé manuellement

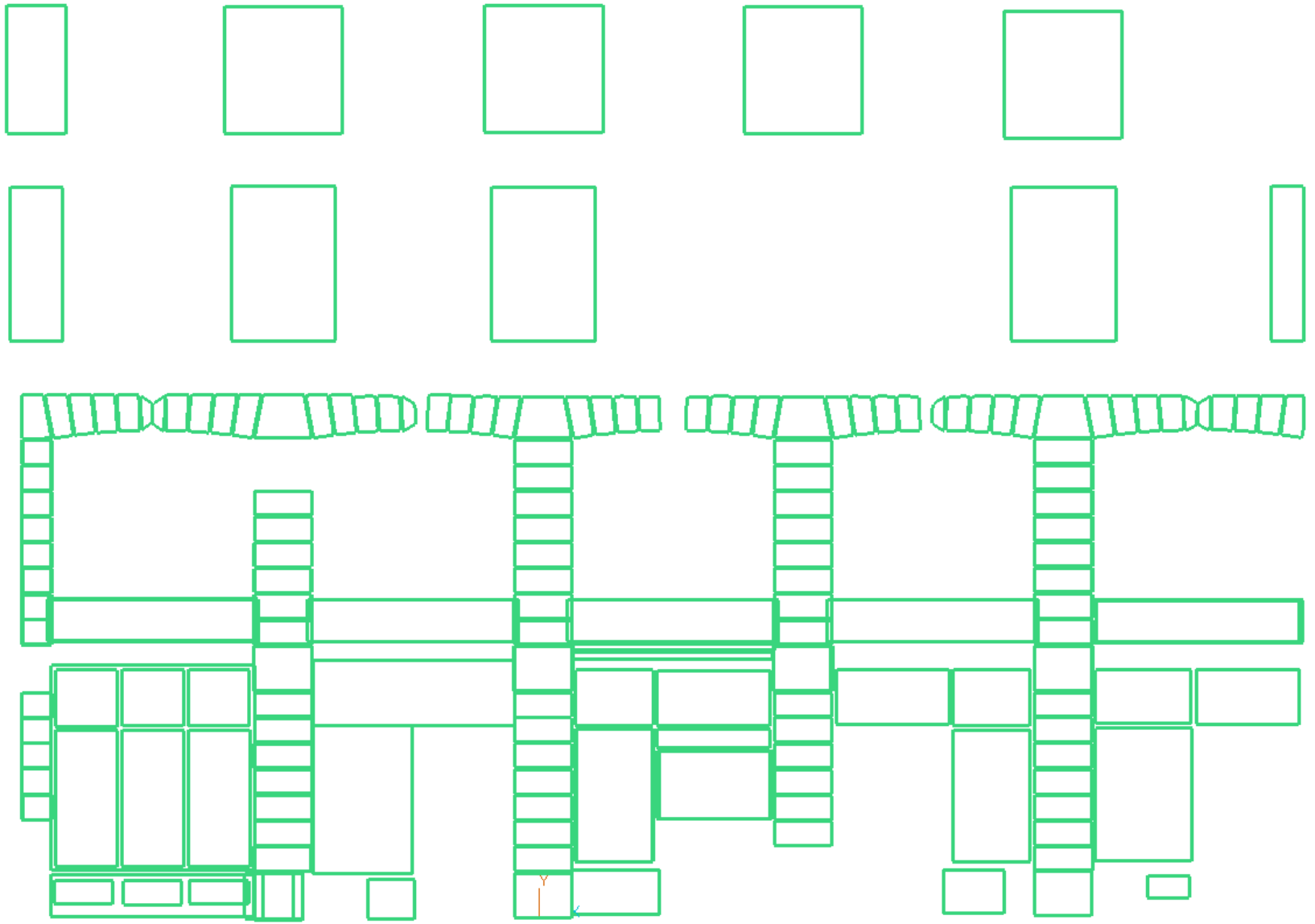
Annexe 4
Résultat de la segmentation RANSAC + croissance de région sur le nuage n°1



Annexe 5
Résultat de la segmentation RANSAC + croissance de région sur le nuage n°2



Annexe 6
Résultat de l'algorithme de détection des contours appliqué au nuage n°1



Annexe 7
Tableaux n°1 et n°2 des écarts entre les modèles

Points haut des fenêtres

Points sur contour	Points sur contour	ΔX (cm)	ΔZ (cm)	ΔD (cm)
P1	P1'	0,37	0,94	1,01
P2	P2'	0,54	1	1,14
P3	P3'	1,30	0,94	1,60
P4	P4'	1,82	0,99	2,07
P5	P5'	0,61	0,93	1,11
P6	P6'	0,78	0,98	1,25
P7	P7'	1,29	0,91	1,58
P8	P8'	1,12	0,97	1,48
P9	P9'	0,57	0,91	1,07
P10	P10'	0,40	0,97	1,05
P41	P41'	0,38	1,01	1,08
P42	P42'	0,56	1,06	1,20
P43	P43'	1,29	0,99	1,63
P44	P44'	1,11	1,05	1,53
P45	P45'	0,57	0,98	1,13
P46	P46'	0,74	1,04	1,28
P47	P47'	1,28	0,97	1,61
P48	P48'	1,11	1,03	1,51
P49	P49'	0,57	0,96	1,12
P50	P50'	0,39	1,02	1,09
P81	P81'	0,40	0,96	1,04
P82	P82'	0,57	0,9	1,07
P83	P83'	1,27	0,98	1,60
P84	P84'	1,10	0,92	1,43
P85	P85'	0,58	0,99	1,15
P86	P86'	0,75	0,93	1,19
P87	P87'	1,27	0,98	1,60
P88	P88'	1,09	0,94	1,44
P89	P89'	0,54	1,01	1,15
P90	P90'	0,37	0,95	1,02

Moyenne = 1,31
Ecart type = 0,26

Points bas des fenêtres

Points sur contour	Points sur contour	ΔX (cm)	ΔZ (cm)	ΔD (cm)
P11	P11'	0,90	3,31	3,43
P12	P12'	1,49	3,38	3,69
P13	P13'	0,77	3,31	3,40
P14	P14'	0,18	3,4	3,40
P15	P15'	1,14	3,32	3,51
P16	P16'	1,72	3,4	3,81
P17	P17'	0,77	3,34	3,43
P18	P18'	0,18	3,42	3,42
P19	P19'	0,04	3,35	3,35
P20	P20'	0,54	3,42	3,46
P51	P51'	0,91	3,25	3,37
P52	P52'	1,50	3,33	3,65
P53	P53'	0,77	3,26	3,35
P54	P54'	0,17	3,34	3,34
P55	P55'	1,09	3,27	3,45
P56	P56'	1,68	3,35	3,75
P57	P57'	0,75	3,28	3,36
P58	P58'	0,17	3,36	3,36
P59	P59'	0,03	3,3	3,30
P60	P60'	0,55	3,37	3,41
P91	P91'	0,93	5,22	5,30
P92	P92'	1,52	5,3	5,51
P93	P93'	0,74	5,23	5,28
P94	P94'	0,15	5,31	5,31
P95	P95'	1,11	5,24	5,36
P96	P96'	1,70	5,32	5,59
P97	P97'	0,74	5,26	5,31
P98	P98'	0,15	5,33	5,33
P99	P99'	0,02	5,27	5,27
P100	P100'	0,57	5,34	5,37

Moyenne = 4,10
Ecart type = 0,92

Annexe 8
Tableaux n°3 et n°4 des écarts entre les modèles

Points haut des appuis

Points sur contour	Points sur contour	ΔX (cm)	ΔZ (cm)	ΔD (cm)
P21	P21'	1,48	3,3	3,62
P22	P22'	1,35	3,39	3,65
P23	P23'	0,19	3,31	3,32
P24	P24'	0,32	3,4	3,42
P25	P25'	1,72	3,32	3,74
P26	P26'	1,58	3,41	3,76
P27	P27'	0,19	3,33	3,34
P28	P28'	0,32	3,42	3,43
P29	P29'	0,53	3,34	3,38
P30	P30'	0,40	3,43	3,45
P61	P61'	1,49	3,24	3,57
P62	P62'	1,36	3,33	3,60
P63	P63'	0,18	3,25	3,25
P64	P64'	0,31	3,35	3,36
P65	P65'	1,73	3,26	3,69
P66	P66'	1,60	3,36	3,72
P67	P67'	0,17	3,28	3,28
P68	P68'	0,31	3,37	3,38
P69	P69'	0,55	3,29	3,34
P70	P70'	0,42	3,38	3,41

Moyenne = 3,49
Ecart type = 0,17

Points bas de appuis

Points sur contour	Points sur contour	ΔX (cm)	ΔZ (cm)	ΔD (cm)
P31	P31'	1,48	2,88	3,24
P32	P32'	1,35	2,97	3,26
P33	P33'	0,19	2,89	2,90
P34	P34'	0,32	2,98	3,00
P35	P35'	1,72	2,9	3,37
P36	P36'	1,59	2,99	3,39
P37	P37'	0,19	2,91	2,92
P38	P38'	0,32	3,01	3,03
P39	P39'	0,53	2,92	2,97
P40	P40'	0,40	3,02	3,05
P71	P71'	1,49	2,83	3,20
P72	P72'	1,36	2,92	3,22
P73	P73'	0,18	2,84	2,85
P74	P74'	0,31	2,93	2,95
P75	P75'	1,73	2,85	3,33
P76	P76'	1,60	2,94	3,35
P77	P77'	0,17	2,86	2,87
P78	P78'	0,30	2,95	2,97
P79	P79'	0,55	2,87	2,92
P80	P80'	0,42	2,97	3,00

Moyenne = 3,09
Ecart type = 0,19

Annexe 9
Script de l'approche RANSAC écrit en JavaScript sous le logiciel 3DRESHAPER

```
//-------------------------------------------------------------//
//-------------------- Partie Fonctions ----------------------//
//-------------------------------------------------------------//

//----- Sélection de 3 points au hasard -----
function getRandomInt(max)
 {
	return Math.floor(Math.random() * Math.floor(max));
}

function getrandomPt(tableau)
{
	var i=getRandomInt(tableau.length);
	var P = SPoint.New(tableau[i][1],tableau[i][2],tableau[i][3]);

	return P
}

//Création d'une table contenant les points du nuage
function CreateTabOfPoints(nuage,tableau)
{
	var it =nuage.GetIterator();
	var i =0
	while(it.IsValid())
		{
		var currentPt = it.GetPt();
		var X=currentPt.GetX();
		var Y=currentPt.GetY();
		var Z=currentPt.GetZ();
		tableau.splice(tableau.length,0,[i,X,Y,Z]);
		i++
		it.GetNext();
		}
	return tableau
}

//Calcul des distances au plan

function DtoPlan(plan,TabPoints)
{
	for (var i=0;i<TabPoints.length;i++)
	{
		var p = SPoint.New(TabPoints[i][1],TabPoints[i][2],TabPoints[i][3]);
		var d = plan.Distance(p);
		TabPoints[i][4]= Math.abs(d);
	}
	return TabPoints;
}

//Répartition des points dans les tableaux d'inliers et d'outliers en fonction d'une distance seuil
function ValidationD (tableau, seuil,Inlier,Outlier)
{
	var j=0;
	var k=0;
	for (var i=0;i<tableau.length;i++)
	{
		if(tableau[i][4]< seuil)
		{
			var x =  tableau[i][1]
			var y = tableau[i][2]
```

```
                        var z =tableau[i][3]
                        var distance = tableau[i][4]
                        Inlier.push([j,x,y,z,distance])
                        j++;
                }
                else
                {
                        var x =  tableau[i][1]
                        var y = tableau[i][2]
                        var z =tableau[i][3]
                        var distance = tableau[i][4]
                        Outlier.push([k,x,y,z,distance])
                        k++;
                }
        }
        return [Inlier, Outlier];
}

//Insère les points du plan détectés dans la scène
function InsererNuage(MeilleurIn)
{
        var InsererInliers = new Array();
        for (var l = 0 ; l < MeilleurIn.length ; l++)
        {
                InsererInliers[l] = SPoint.New(MeilleurIn[l][1],MeilleurIn[l]           [2],MeilleurIn[l][3]);
        }

        var InsInlier = SCloud.Extract(InsererInliers);
        var aa = InsInlier.Cloud;
        aa.AddToDoc();
        return aa;
}

//-------------------------------------------------------------//
//-------------------- Partie Programme -------------------//
//-------------------------------------------------------------//

// Récupération du nuage
var result = SCloud.All(1);
if (result.length!=1)
{
        throw new Error( "Veuillez n'afficher qu'un seul nuage a l'ecran" );
}
var cloudToTreat = result[0];

//---------------------------- Affichage de la boite de dialogue ----------------------------------------------------
var myDialog = SDialog.New('Segmentation Ransac');
var imagePath =("C:/Users/MMA/Desktop/Scripts/Logo.png")
myDialog.AddImage(imagePath,400,400);
myDialog.AddLine("D\351claration des variables",false, {"align":"center","size" : 12});
myDialog.AddLine("Valeur de la distance seuil, en dessous de laquelle un point est qualifi\351 d'inlier",true,
{"align":"center","size" : 10});
myDialog.AddLine("Pourcentage d'outliers dans le nuage (0.05)",true, {"align":"center","size" : 10});
myDialog.AddLine("Probabilit\351 de succes souhait\351 (0.99)",true, {"align":"center","size" : 10});
myDialog.AddLine("Taille minimale d'un \351chantillon",true, {"align":"center","size" : 10});
var result = myDialog.Execute();
if (result.ErrorCode !=0)
{
        print('Vous avez annul\351 la commande')
        break
}

//---------------------------------------- Récupération des réponses de l'utilisateur----------------------------------------------
var dist = result.InputTbl[0]
var w = result.InputTbl[1]
var p = result.InputTbl[2]
var s = result.InputTbl[3]
```

```
var MaxInliers = 0

// Calcul du nombre d'itérations a réaliser
var N = (Math.log(1-p))/(Math.log(1-Math.pow(1-w,s)))
N = Math.round(N)
print("L'algorithme effectuera " + N +" iterations.")

//Rangement des points dans un tableau
	var t = new Array();
	var tab = CreateTabOfPoints(cloudToTreat,t);

for (var iteration=0 ; iteration < N ; iteration++)
{
	print("boucle 1")
	//Création du plan
	var plane = SPlane.New(getrandomPt(tab),getrandomPt(tab),getrandomPt(tab));

	//Calcul de la distance de chaque point au plan
	var d = DtoPlan(plane,tab)

	//Répartition des points dans les tableaux d'inliers et d'outliers
	var Inlier = new Array();
	var Outlier = new Array();
	var resultat = ValidationD(d,dist,Inlier,Outlier)
	var In = resultat[0]
	var Out = resultat[1]

	// Test si meilleur ensemble de points
	if (In.length>MaxInliers)
	{
		var MeilleurIn = In
		var MeilleurOut = Out
		var MeilleurPlan = plane
		MaxInliers = In.length
	}
}

// Insertion des inliers
InsererNuage(MeilleurIn);

//Désactive la visibilité du nuage initial
cloudToTreat.SetVisibility(false);

while (MeilleurOut.length>1000)
{
	var tab = MeilleurOut;
	var MaxInliers = 0
	for (var iteration=0 ; iteration < N ; iteration++)
	{
		var plane = SPlane.New(getrandomPt(tab),getrandomPt(tab),getrandomPt(tab));
		var d = DtoPlan(plane,tab);
		var Inlier = new Array();
		var Outlier = new Array();
		var resultat = ValidationD(d,dist,Inlier,Outlier);
		var In = resultat[0];
		var Out = resultat[1];
		if (In.length>MaxInliers)
		{
			var MeilleurIn = In;
			var MeilleurOut = Out;
			var MeilleurPlan = plane;
			MaxInliers = In.length;
		}
	}
	InsererNuage(MeilleurIn);
}
```

Annexe 10
Script de la création automatique des contours des formes architecturales « pleines » écrit en langage JavaScript sous le logiciel 3DRESHAPER

```
//---------------------------- Affichage de la boite de dialogue ----------------------------------------------------
var myDialog = SDialog.New('Detection des contours');
var imagePath =("C:/Users/MMA/Desktop/Scripts/Logo.png")
myDialog.AddImage(imagePath,400,400);
myDialog.AddLine("Instruction pour l'utilisation du script",false, {"align":"center","size" : 12});
myDialog.AddLine("1) Avant de lancer le script, veuillez afficher uniquement les nuages \340 traiter",false, {"align":"center","size" : 10});
myDialog.AddLine("2) Les \351l\351ments trait\351s sont-ils de forme rectangulaire? (4 cot\351s, angle 90 degr\351) ('oui'/'non')",true, {"align":"center","size" : 10});
myDialog.AddLine("3) Les \351l\351ments trait\351s doivent ils \352tre parall\350les \340 l'horizontale? ('oui'/'non')",true, {"align":"center","size" : 10});
myDialog.AddLine("4) Les \351l\351ments trait\351s doivent ils \352tre identiques? ('oui'/'non')",true, {"align":"center","size" : 10});
var result = myDialog.Execute();
if (result.ErrorCode !=0)
{
	print('Vous avez annul\351 la commande')
	break
}

//---------------------------------------- Récupération des réponses de l'utilisateur----------------------------------------------
var reponse1 = result.InputTbl[0]
var reponse2 = result.InputTbl[1]
var reponse3 = result.InputTbl[2]


//------------------------------------------------- Chargement des nuages ------------------------------------------------
var nuages = SCloud.All(1)
for ( var i =0; i<nuages.length;++i)
{
	var cloud = nuages[i];

	//-------------------------------- Création du meilleur plan a partir des points du nuage ---------------------------------------
	plan = cloud.BestPlane();
	if ( plan.ErrorCode != 0 )
	{
		throw new Error('Le calcul du meilleur plan a echou\351.');
	}
	var result = plan.Plane;

	//------------------------------ Récupération des composantes de la normale du plan ----------------------------------
	var n = result.GetNormal();
	var x = n.GetX();
	var y = n.GetY();
	var z = n.GetZ();

	//------------------------------ Création de l'enveloppe convexe du nuage ------------------------------------------------
	cont = cloud.GetConvexContour(SVector.New(x,y,z));
	var contour = cont.Multi;

	//---------------------------------------------- Rééchantillonage ----------------------------------------
	if (reponse1 == 'oui')
	{
		contour.Reduce(4,0.03,1);
		contour.RightAngle(45);
		if(contour.GetNumber()>5)
		{
			contour.Reduce(4,0.1,1);
			contour.RightAngle(45);
		}
```

```
		}

		//------------------------------------- Ajout du contour crée ---------------------------------------------------
		contour.AddToDoc();

}
//------------------------------------------- Alignement -----------------------------------------
if (reponse2 == 'oui')
{
		Include('C:/Users/MMA/Desktop/Scripts/DetectionContoursPleins/RotationAll.js');
}

//------------------------------------------- Dupliquer les contours similaires ------------------------------------------
if (reponse3 == 'oui')
{
		//--------------------------- Affichage de la boite de dialogue ------------------------------------------------
		var myDialog = SDialog.New('Choix du meilleur motif');
		myDialog.AddLine("Analysez visuellement les r\351sultats obtenus et s\351l\351ctionnez le contour \340 dupliquer",false,
{"align":"center","size" : 12});
		var result = myDialog.Execute();
		if (result.ErrorCode !=0)
		{
			print('Vous avez annul\351 la commande')
			break
		}

		select = SMultiline.FromClick();
		best = select.Multi;
		best.RemoveFromDoc();
		centre = best.GetCentroidLinear();
		xc = centre.GetX() ; yc = centre.GetY() ; zc = centre.GetZ() ;
		var ligne = SMultiline.All(1);
		for (var i=0; i<ligne.length;++i)
		{
			bestcopy = SMultiline.New(best)
			c = ligne[i].GetCentroidLinear()
			x = c.GetX() ; y = c.GetY() ; z = c.GetZ() ;
			var vecteur = SVector.New(x-xc,y-yc,z-zc)
			bestcopy.Translate(vecteur)
			bestcopy.AddToDoc();
			ligne[i].RemoveFromDoc();
		}
		best.AddToDoc();
}

//------------------------------------------- Désactive l'affichage du nuage ---------------------------------------------
//cloud.SetVisibility(false);
```

Annexe 11
Script de l'outil « Rotation » pour aligner des polylignes avec l'horizontale écrit en langage JavaScript sous le logiciel 3DRESHAPER

```
//------------------------------ Récupération des polylignes dans la scène ------------------------------------
var ligne = SMultiline.FromSel()
for (var i=0; i<ligne.length;++i)
{
	centre = ligne[i].GetCentroidLinear()
	point1 = ligne[i].GetPoint(0)
	point2 = ligne[i].GetPoint(1)
	point3 =  ligne[i].GetPoint(2)
	point4 =  ligne[i].GetPoint(3)

//------------------------------ Classement des sommets dans un tableau ------------------------------------
	var tabx=new Array([point1.GetZ(),point1 ],[point2.GetZ(),point2],
					[point3.GetZ(),point3],[point4.GetZ(),point4]);

	tabx.sort()
	p1=tabx[0][1]
	p2=tabx[1][1]


	if (p1.GetX()<p2.GetX())
	{
		angle = Math.atan((p2.GetZ()-p1.GetZ())/(p2.GetX()-p1.GetX()))
		angle = (angle*180)/Math.PI
		print(angle)
		if(angle!=0)
		{	//------- Création de la matrice de rotation dans le cas n°1 ------
			var rotateMatrix = SMatrix.New(centre,
							SVector.New(0,1,0),
							angle,
							SMatrix.DEGREE);

			ligne[i].ApplyTransformation(rotateMatrix)
		}
	}
	else
	{
		angle = Math.atan((p2.GetZ()-p1.GetZ())/(p1.GetX()-p2.GetX()))
		angle = (angle*180)/Math.PI
		print(angle)
		if(angle!=0)
		{	//------- Création de la matrice de rotation dans le cas n°1 ------
			var rotateMatrix = SMatrix.New(centre,
							SVector.New(0,-1,0),
							angle,
							SMatrix.DEGREE);

			ligne[i].ApplyTransformation(rotateMatrix)
		}
	}
}
```

Annexe 12
Script de l'outil « Alignement Vertical » pour aligner verticalement des polylignes entre elles. Ce script est écrit en langage JavaScript sous le logiciel 3DRESHAPER

```
//---------------------------- Affichage de la boite de dialogue ----------------------------------------------------
var myDialog = SDialog.New('Alignement Vertical');
var imagePath =("C:/Users/MMA/Desktop/Scripts/Logo.png")
myDialog.AddImage(imagePath,400,400);
myDialog.AddLine("Instruction pour l'utilisation du script",false, {"align":"center","size" : 12});
myDialog.AddLine("1) Avant de lancer le script, veuillez ne s\351l\351ctionner que les contours \340 traiter",false,
{"align":"center","size" : 10});
myDialog.AddLine("2) Apr\350s avoir cliqu\351 sur 'OK', veuillez s\351l\351ctionner le contour modele",false,
{"align":"center","size" : 10});
myDialog.AddLine("3) Souhaitez vous r\351aliser un alignement \340 gauche ou \340 droite ? ('g'/'d')",true, {"align":"center","size" :
10});
var result = myDialog.Execute();
if (result.ErrorCode !=0)
{
	print('Vous avez annul\351 la commande')
	break
}

var reponse1 = result.InputTbl[0]


//------------------------------ Fonction pour classement croissant --------------
function compareC(x,y)
{
	return x - y;
}

//------------------------------ Fonction pour classement décroissant --------------
function compareD(x,y)
{
	return y - x;
}

var modele = SMultiline.FromClick();
var geomodele = modele.Multi;
point1 = geomodele.GetPoint(0)
point2 = geomodele.GetPoint(1)
point3 =  geomodele.GetPoint(2)
point4 =  geomodele.GetPoint(3)

var polys = SMultiline.FromSel();

if (reponse1 == "g")
{
	var tabx=new Array(point1.GetX(),point2.GetX(),
                                                        point3.GetX(),point4.GetX());
	tabx.sort(compareC)
	for (var i=0; i<polys.length;++i)
	{
		var ligne = polys[i];
		centreligne = ligne.GetCentroidLinear();
		p1 = ligne.GetPoint(0)
		p2 = ligne.GetPoint(1)
		p3 =  ligne.GetPoint(2)
		p4 =  ligne.GetPoint(3)
		var tabx2=new Array([p1.GetX() ],[p2.GetX()],
                                                                [p3.GetX()],[p4.GetX()]);
		tabx2.sort(compareC)
		var dx = Math.abs(tabx2[0] - tabx[0])
```

```
                if(tabx[0]<tabx2[0])
                {       //------- Création du vecteur de translation dans le cas n°1 ------
                        var vecteur = SVector.New(-dx,0,0);
                        var Matrix = SMatrix.New(vecteur);
                        ligne.ApplyTransformation(Matrix);
                }
                else
                {       //------- Création du vecteur de translation dans le cas n°2 ------
                        var vecteur = SVector.New(dx,0,0);
                        var Matrix = SMatrix.New(vecteur);
                        ligne.ApplyTransformation(Matrix);
                }

        }
}

if (reponse1 == "d")
{

        var tabx=new Array(point1.GetX(),point2.GetX(),
                        point3.GetX(),point4.GetX());
        tabx.sort(compareD)
        for (var i=0; i<polys.length;++i)
        {
                var ligne = polys[i];
                centreligne = ligne.GetCentroidLinear();
                p1 = ligne.GetPoint(0)
                p2 = ligne.GetPoint(1)
                p3 =  ligne.GetPoint(2)
                p4 =  ligne.GetPoint(3)
                var tabx2=new Array(p1.GetX() ,p2.GetX(),
                                p3.GetX(),p4.GetX());
                tabx2.sort(compareD)
                var dx = Math.abs(tabx2[0] - tabx[0])
                if(tabx[0]<tabx2[0])
                {       //------- Création du vecteur de translation dans le cas n°1 ------
                        var vecteur = SVector.New(-dx,0,0);
                }
                else
                {       //------- Création du vecteur de translation dans le cas n°2 ------
                        var vecteur = SVector.New(dx,0,0);
                }
                //------- Création de la matrice selon le vecteur de translation ------
                var Matrix = SMatrix.New(vecteur);
                ligne.ApplyTransformation(Matrix);
        }
}
```

Annexe 13
Script de l'outil « Alignement Horizontal » pour aligner verticalement des polylignes entre elles. Ce script est écrit en langage JavaScript sous le logiciel 3DRESHAPER

```
//---------------------------- Affichage de la boite de dialogue ----------------
var myDialog = SDialog.New('Alignement Horizontal');
var imagePath =("C:/Users/MMA/Desktop/Scripts/Logo.png")
myDialog.AddImage(imagePath,400,400);
myDialog.AddLine("Instruction pour l'utilisation du script",false, {"align":"center","size" : 12});
myDialog.AddLine("1) Avant de lancer le script, veuillez ne s\351l\351ctionner que les contours \340 traiter",false,
{"align":"center","size" : 10});
myDialog.AddLine("2) Apr\350s avoir cliqu\351 sur 'OK', veuillez s\351l\351ctionner le contour modele",false,
{"align":"center","size" : 10});
myDialog.AddLine("3) Souhaitez vous r\351aliser un alignement \340 bas ou \340 haut ? ('b'/'h')",true, {"align":"center","size" :
10});
var result = myDialog.Execute();
if (result.ErrorCode !=0)
{
	print('Vous avez annul\351 la commande')
	break
}

var reponse1 = result.InputTbl[0]

//----------------------------- Fonction pour classement croissant --------------
function compareC(x,y)
{
	return x - y;
}

//---------------------------- Fonction pour classement décroissant --------------
function compareD(x,y)
{
	return y - x;
}

var modele = SMultiline.FromClick();
var geomodele = modele.Multi;
point1 = geomodele.GetPoint(0)
point2 = geomodele.GetPoint(1)
point3 =  geomodele.GetPoint(2)
point4 =  geomodele.GetPoint(3)

var polys = SMultiline.FromSel();

if (reponse1 == "b")
{
	var tabz=new Array(point1.GetZ(),point2.GetZ(),
                                                point3.GetZ(),point4.GetZ());
	tabz.sort(compareC)
	for (var i=0; i<polys.length;++i)
	{
		var ligne = polys[i];
		centreligne = ligne.GetCentroidLinear();
		p1 = ligne.GetPoint(0)
		p2 = ligne.GetPoint(1)
		p3 =  ligne.GetPoint(2)
		p4 =  ligne.GetPoint(3)
		var tabz2=new Array(p1.GetZ(),p2.GetZ(),
                                p3.GetZ(),p4.GetZ());
		tabz2.sort(compareC)
		var dz = Math.abs(tabz2[0] - tabz[0])
		if(tabz[0]<tabz2[0])
		{		//------- Création du vecteur de translation dans le cas n°1 ------
```

```
                        var vecteur = SVector.New(0,0,-dz);
                        var Matrix = SMatrix.New(vecteur);
                        ligne.ApplyTransformation(Matrix);
                }
                else
                {       //------- Création du vecteur de translation dans le cas n°2 ------
                        var vecteur = SVector.New(0,0,dz);
                        var Matrix = SMatrix.New(vecteur);
                        ligne.ApplyTransformation(Matrix);
                }

        }
}

if (reponse1 == "h")
{

        var tabz=new Array(point1.GetZ(),point2.GetZ(),
                        point3.GetZ(),point4.GetZ());
        tabz.sort(compareD)
        for (var i=0; i<polys.length;++i)
        {
                var ligne = polys[i];
                centreligne = ligne.GetCentroidLinear();
                p1 = ligne.GetPoint(0)
                p2 = ligne.GetPoint(1)
                p3 =  ligne.GetPoint(2)
                p4 =  ligne.GetPoint(3)
                var tabz2=new Array(p1.GetZ() ,p2.GetZ(),
                                p3.GetZ(),p4.GetZ());
                tabz2.sort(compareD)
                var dz = Math.abs(tabz2[0] - tabz[0])
                if(tabz[0]<tabz2[0])
                {       //------- Création du vecteur de translation dans le cas n°1 ------
                        var vecteur = SVector.New(0,0,-dz);
                }
                else
                {       //------- Création du vecteur de translation dans le cas n°2 ------
                        var vecteur = SVector.New(0,0,dz);
                }
                var Matrix = SMatrix.New(vecteur);
                ligne.ApplyTransformation(Matrix);
        }
}
```

Liste des figures

Liste des tableaux

Optimisation du traitement de nuage de points pour la production de plan de façade au sein d'un cabinet de géomètre-expert.

Mémoire d'Ingénieur C.N.A.M., Le Mans 2018

RESUME

La présence des scanners lasers au sein des cabinets de géomètres-experts s'est généralisée au cours des dernières années. Le nuage de points obtenu est actuellement très peu utilisé à sa juste valeur et l'information qu'il contient est sous-exploitée. L'objectif de ce mémoire est de proposer des solutions et des méthodes de traitements, automatisées ou semi-automatisées, permettant de simplifier la création de plans d'élévation.

Pour cela, deux algorithmes ont été particulièrement utilisés durant cette étude : l'algorithme RANSAC et l'algorithme de Croissance de région. Ces méthodes itératives, appliquées à une importante quantités de données, sont consommatrices de temps. C'est pourquoi des versions alternatives, adaptées à notre cas d'étude, ont été proposées. Le nuage de points a ainsi été segmenté en éléments partageant des caractéristiques communes, puis une méthode de détection et de vectorisation des points de contours a été implémentée afin de produire un modèle vectoriel en deux dimensions.

Mots clés : 3D, nuage de points, LIDAR, façade, segmentation, RANSAC, Croissance de région, automatisation.

ABSTRACT

The use of laser scanners in surveying firm has become widespread in recent years. The point cloud obtained is curently very little used at its fair value and the information contained is underutilized. The purpose of this paper is to propose solutions and methods of automated or semi-automated processes to simplify the creation of elevation plans.

In order to achieve this, two algorithms were particularly used during this study : the RANSAC algorithm and the region growing algorithm. These iterative methods, applied to a large amounts of data, are time consuming. This is why alternative versions, adapted to our case study, have been proposed. The point cloud has been segmented into elements sharing common characteristics, then a method of edge points detection and vectorization was implemented to produce a two-dimensional vector model.

Key words : 3D, point cloud, LIDAR, facade, segmentation, RANSAC, Region growing, automating.