Extraction d'éléments géométriques dans un nuage de points LiDAR terrestre : application aux relevés de façades

Mounir Ait Mansour

▶ **To cite this version:**

Mounir Ait Mansour. Extraction d'éléments géométriques dans un nuage de points LiDAR terrestre : application aux relevés de façades. Sciences de l'ingénieur [physics]. 2014. dumas-01164570

HAL Id: dumas-01164570

https://dumas.ccsd.cnrs.fr/dumas-01164570

Submitted on 17 Jun 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

le cnam
École supérieure des géomètres et topographes esgt

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

ÉCOLE SUPÉRIEURE DES GÉOMÈTRES ET TOPOGRAPHES

MÉMOIRE

présenté en vue d'obtenir

le DIPLÔME D'INGÉNIEUR CNAM

Spécialité : Géomètre et Topographe

par

Mounir AÏT-MANSOUR

Extraction d'éléments géométriques dans un nuage de points LiDAR terrestre. Application aux relevés de façades.

Soutenu le mercredi 17 septembre 2014

JURY

PRÉSIDENTE : **Mme Marie FOURNIER**

MEMBRES : **M. Ange-Lucien GUIDICELLI, maître de stage**
M. Ghyslain FERRE, professeur référent
M. Jean-Michel FOLLIN
M. Jean-Marie SEITE
M. Vincent HABCHI

Remerciements

J'adresse mes remerciements aux personnes qui m'ont aidé dans la réalisation de ce travail de fin d'étude.

M. Ghyslain Ferré qui, en tant que professeur référent, m'a donné quelques conseils avisés et m'a guidé dans mes travaux.

Les associés de l'entreprise GeoSystem Surveying, Ange-Lucien Guidicelli, David Verdier et Nicolas Vivier pour avoir donné de votre temps pour ce projet et tout simplement pour ces cinq mois passés dans votre société.

J'adresse aussi mes remerciements à ma famille qui m'a toujours poussé à continuer mes études.

Table des matières

Introduction

0.1 Contexte de l'étude

Les premiers prototypes de Lidar terrestres (acronyme anglais *« light detection and ranging »* pour « télédétection[1] par laser »), autrement appelé scanner laser terrestre, sont apparus dans les années 80 (T. Landes P. G., 2011). Cet instrument permet de numériser un environnement en trois dimensions sous forme d'un ensemble de points appelé nuage de points. A l'époque, il s'agissait d'une technologie peu répandue et développée dans des laboratoires de recherche.

Mais depuis le début des années 2000, le scanner laser terrestre connaît un essor très important. Plusieurs sociétés dans le monde le commercialisent pour différents secteurs tels que l'architecture et le bâtiment (maquette en trois dimensions), l'archéologie et la conservation du patrimoine (numérisation et conservation d'édifice historique), l'industrie automobile (contrôle qualité, maquette automobile) ainsi que la topométrie (auscultation d'ouvrage, récolement).

Sa précision millimétrique et sa rapidité d'acquisition en font un outil désormais incontournable, notamment pour le géomètre-topographe. L'utilisation de scanner laser terrestre dans les cabinets de géomètres s'est réellement démocratisée depuis ces cinq dernières années. Cet instrument est une véritable révolution dans la profession : c'est l'un des premiers outils de relevé qui permet d'alléger la tâche de tri et de sélection de l'information sur le terrain.

C'est dans ce critère d'exhaustivité de l'information terrain que réside la plus-value de l'utilisation d'un scanner laser terrestre par rapport à une station totale : elle permet une maîtrise des coûts par sa rapidité et une maîtrise des risques par la quantité d'information terrain acquise (pas de retours sur le terrain pour complément).

0.2 Problématique

Toutefois, cette plus-value sur le terrain est ternie par un post-traitement au bureau assez lourd. Il n'est pas rare d'avoir plusieurs dizaines de millions de points dans un nuage pouvant correspondre à des dizaines de giga-octets de données. Il faut alors investir dans des ordinateurs suffisamment puissants pour travailler sur ces données. À cela s'ajoute une tâche de sélection de l'information (auparavant réalisée sur le terrain avec un instrument conventionnel) parfois délicate à l'écran pour un non-initié. Il apparait clairement qu'il faille trouver dans le futur des procédés pour automatiser une partie du traitement des nuages de points.

C'est au sein de l'entreprise G2S GeoSystem Surveying que cette tentative d'automatisation du traitement de nuage de points a été réalisée. C'est un bureau d'étude de géomètres créé en août 2013 sous la forme d'une coopérative. Les trois associés-fondateurs sont portés vers les nouvelles technologies et utilisent depuis plus de quatre ans des instruments de type scanner laser terrestre, ce qui leur donne une certaine expérience dans ce secteur émergent.

[1] La télédétection est une méthode qui permet d'obtenir des informations sur des objets en recueillant et en analysant des données sans contact direct entre l'instrument utilisé et l'objet analysé (définition de l'Agence Spatiale Européenne)

C'est lors de stages précédents que m'est apparu ce déséquilibre dans le cadre de relevés de façades. Il y avait une nette différence entre un relevé généralement rapide et efficace et un traitement des données plutôt chronophage. Ce travail de fin d'étude proposera une solution pour réduire ce déséquilibre en automatisant une partie de la chaîne de production d'un livrable 2D ou 3D d'une façade.

0.3 Présentation du travail

Ainsi, après avoir fait un bref descriptif de ce qui a été réalisé dans le domaine de l'extraction d'éléments géométriques dans un nuage de points (**Partie 1**), l'étude portera sur le développement d'un outil qui essaiera de reconnaître automatiquement des éléments géométriques dans un nuage de points. L'outil tentera de les modéliser en trois dimensions (**Partie 2**) afin d'être exploitable dans un logiciel de dessin assisté par ordinateur, permettant d'alléger le travail du dessinateur dans une optique de réduction des coûts du traitement des données. Après cette phase de développement, la chaîne de traitement sera testée (**Partie 3**) à partir d'un échantillon de relevés de façades.

Partie 1 Etat de l'art

Cette première partie sera consacrée à l'état de l'art en matière d'extraction d'éléments géométriques dans un nuage de points. Elle sera divisée en trois points :

- *« Acquisition des données et prétraitement »* : il y sera décrit le fonctionnement théorique de l'appareil ainsi que le procédé de relevé sur le terrain. Nous verrons par ailleurs que les données obtenues doivent suivre un certain nombre de prétraitement avant de pouvoir être exploitées.
- *« Segmentation automatique du nuage »* : on y fera une synthèse des différentes techniques pour subdiviser un nuage de points en sous-ensemble cohérent suivant des critères prédéfinis.
- *« Rendu 3D et modélisation »* : il s'agira d'une synthèse de différentes méthodes pour transformer un environnement discret (i.e. le nuage de points) en un environnement continu (exemple des objets maillés).

Chacun de ces points décrira les aspects théoriques des procédés étudiés puis exposera un cas pratique à travers des logiciels libres.

1.1 Acquisition des données et prétraitement

1.1.1 Le scanner laser terrestre : principe de fonctionnement

Le nom de « scanner laser terrestre » (SLT) est composé des trois termes que sont «scanner», «laser» et «terrestre». Ces trois termes correspondent à trois fonctions, trois principes du fonctionnement de l'appareil.

1.1.1.1 Le terme « laser »

Le SLT utilise le principe physique du laser (acronyme anglais «light amplification by stimulated emission of radiation» pour «amplification de la lumière par émission stimulée de rayonnement»). Le laser est une source de lumière dont le faisceau émis est très fin et peu divergent (de l'ordre du milli radian). Il est utilisé depuis quelques décennies par les géomètres dans la mesure de distances (utilisation du tachéomètre et du distancemètre). Dans un SLT, cette mesure de distance s'effectue essentiellement par deux méthodes :

- Par mesure du temps de vol : L'appareil émet un faisceau laser pulsé (Shan & Toth, 2008) et en mesure le temps aller-retour pour en déduire une distance. Cette dernière est retrouvée à partir de la formule $D = \frac{c_0}{2n} * \Delta t$ où D est la distance du SLT à l'objet que l'on souhaite mesurer, c_0 la vitesse de la lumière dans le vide, n l'indice de réfraction du milieu (il s'agit en règle générale de l'air) et Δt l'intervalle de temps mesuré entre l'émission du rayon et sa réception.

  Équation 1 Mesure par temps de vol

- Par différence de phase : L'appareil émet un faisceau laser continu modulé en amplitude[2] (Boulaassal, 2010) et mesure le déphasage[3] $\Delta\varphi$ pour déterminer la distance D à l'objet. Soit f la fréquence du signal modulé et λ sa longueur d'onde[4] associée. On retrouve la distance aller-retour par la somme des *N* répétitions de λ et de la longueur du déphasage : $2D = N\lambda + \frac{\Delta\varphi * \lambda}{2\pi} \leftrightarrow D = \frac{c}{f}\left(\frac{N}{2} + \frac{\Delta\varphi}{\pi}\right)$.

  Le problème consiste à retrouver le nombre entier *N* de répétition de λ appelé ambiguïté sur la distance. Des méthodes existent pour résoudre cette ambiguïté (fréquences proches, emboîtées).

  Équation 2 Mesure par différence de phase

Ces deux méthodes de mesure de distance ont des différences en termes de précision et de portée de la mesure. Pour (Dujardin, 2013), la mesure par temps de vol est adaptée pour les mesures de longues distances (> 100m), à l'opposé de la mesure par différence de phase qui est plutôt adaptée pour les portées plus courtes (80 m maximum, cependant le problème semble résolu dans le dernier modèle de Scanner 3D Faro Focus X330[5]). A contrario, la mesure par temps de vol est moins précise et plus sensible au bruit.

1.1.1.2 Le terme « scanner »

Selon le (Larousse, 2014), un scanner est un « capteur analogique ou numérique pourvu d'un dispositif de balayage pour l'obtention d'images de la Terre ». Le dispositif de balayage est un élément clé pour définir le SLT. Ainsi, (T. Landes P. G., 2011) définit la notion de balayage laser terrestre comme étant une « technique d'acquisition rapide et automatique de données tridimensionnelles utilisant la lumière laser pour mesurer directement, sans contact avec l'objet et selon une trame régulière, les coordonnées 3D des points sur des surfaces depuis une position terrestre ».

Ce balayage tridimensionnel est effectué par deux mécanismes de mouvement :

- un premier par rotation verticale d'un miroir permettant de balayer l'environnement verticalement. Le champ de vision dans cette direction est limité par la structure basse de l'instrument ;
- un second par rotation de l'appareil sur un plan horizontal permettant de balayer l'environnement panoramique (rotation sur 180° et balayage vertical avant-arrière).

Il en découle ainsi des mesures d'angles horizontaux et verticaux. Elles sont alors couplées aux mesures de distances par laser afin d'obtenir la numérisation de l'environnement sous forme d'un nuage de points. Selon que le SLT est équipé d'un système de mesure à temps de vol ou à différence de phase, les caractéristiques du balayage ne seront pas identiques :

- dans un SLT à temps de vol, le processus de relevé est « ralenti » par le fait qu'il faille attendre le signal retour du laser pulsé avant de changer d'angle et de passer à la mesure suivante.

[2] La modulation d'amplitude consiste à faire varier l'amplitude d'un signal de fréquence élevée (appelé porteuse) en fonction d'un signal de basse fréquence (contenant l'information)

[3] Différence de phase entre le signal modulé allé et retour

[4] La longueur d'onde correspond à la distance parcourue par une période

[5] Conférence Faro du 16 mai 2014 à Tremblay en France

- dans un SLT à différence de phase, le signal laser continu permet un balayage plus fluide de l'environnement. Toutefois, la mesure en continue augmente l'apparition de « bruit » et d'artéfact dans le nuage de points (Dujardin, 2013).

1.1.1.3 Le terme « terrestre »

Les SLT englobent les systèmes fixes (scanner laser statique i.e. posé sur un trépied) et les systèmes dynamiques (scanner laser mobile i.e. embarqué dans un véhicule). Ils se différencient des SLA (scanner laser aéroporté) par leurs fonctions d'instrument de levé au sol (topographie à petite échelle, corps de rue, façade d'un bâtiment etc.).

Les relevés par SLT fixe et dynamique produisent tous les deux le même type de données : le nuage de points. Cependant, la construction du nuage de points est différente :

- dans un SLT dynamique, la position de l'appareil est connue en temps réel car il est rattaché à une centrale inertielle ainsi qu'un GNSS différentiel[6] (Goulette, 2009). Cela permet de déterminer les coordonnées des points du nuage directement ;
- dans un SLT fixe, l'appareil est placé à différents endroits afin d'avoir un relevé complet de la scène. Il faut alors procéder à un assemblage de ces différents nuages « locaux ». C'est ce qu'on appelle l'étape de consolidation, étape qui sera traitée dans la partie 1.1.3.

Dans la suite du mémoire, on se concentrera sur l'utilisation d'un SLT fixe à différence de phase, le Faro Focus 3D S (voir annexe 1 *« Fiche technique du Faro Focus 3D »*).

1.1.2 Le nuage de points

Comme expliqué dans l'introduction, le SLT permet de numériser un environnement en trois dimensions sous forme d'un nuage de points. Selon (T. Landes P. G., 2011), le nuage de points est « un ensemble tramé de points 3D représentant la surface relevée par l'instrument ».

1.1.2.1 Caractéristiques

Le nuage de points est caractérisé par :

- La géométrie des points : ils peuvent être réguliers ou irréguliers, selon que les points sont placés selon une grille de tramage ou non ;
- La valeur scalaire des points : il s'agit essentiellement de l'intensité (albédo du point relevé) ou de la colorimétrie (couleur du point issue de la projection d'une photographie).

1.1.2.2 Formats de fichier des données

Les formats constructeurs :

Les constructeurs ainsi que les logiciels propriétaires ont chacun créé leurs propres formats de fichier : *.rcp pour Autodesk, *.fls pour Faro, *.pts pour Leica etc. L'utilisation de ces formats de fichiers rend difficile

[6] Méthode de positionnement centimétrique par satellite

l'échange et le traitement des données sur plusieurs logiciels. Le passage d'un format à un autre peut engendrer des imprécisions, des erreurs voir des pertes de données.

Les formats d'échanges E57 et LAS :

Il vaut mieux alors se tourner vers des formats ouverts tel que l'E57 ou le LAS. Ce sont des formats standards supportés par la plupart des logiciels, y compris les logiciels constructeurs. La structure du fichier E57 est décrite dans la *Figure 1* par l'utilisation du logiciel CloudCompare. Selon (Huber, 2011), l'E57 est un format plus général que le LAS car ce dernier est utilisé en premier lieu dans le traitement de données Lidar aéroporté.

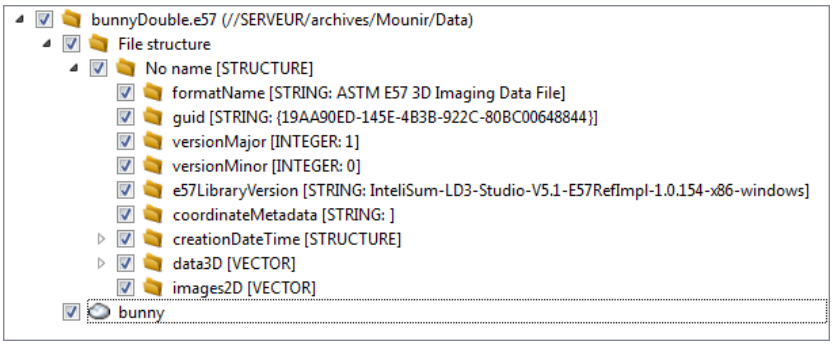


Figure 1 Structure du fichier E57 lu par CloudCompare

Un format libre, le PCD :

La Point Cloud Library[7] (PCL) est une bibliothèque C++ distribuée sous licence BSD. Elle est développée par des chercheurs de différentes universités dans le monde et soutenue par d'importants acteurs tels que Trimble et Leica Geosystems (Zygmunt, 2013). Elle a pour but de faciliter les recherches dans le domaine du traitement de nuage de points 3D en fournissant des outils implémentés.

Cette bibliothèque propose son propre format de fichier, le format PCD (*Point Cloud Data*). Il est constitué d'un entête (qui décrit les propriétés du nuage de points) et de données ASCII :

`VERSION .7`	version du fichier pcd
`FIELDS x y z rgb`	format du point (ici coordonnées x y z + couleur)
`SIZE 4 4 4 4`	taille des données en octet
`TYPE F F F F`	type des données (F pour nombre flottant)
`COUNT 1 1 1 1`	
`WIDTH 213`	largeur du nuage (ou nombre de points)
`HEIGHT 1`	hauteur du nuage (ou 1 pour nuage non régulier)
`VIEWPOINT 0 0 0 1 0 0 0`	point de vue du nuage (translation / quaternion)
`POINTS 213`	nombre total de points du nuage
`DATA ascii`	type de données stocké (ASCII)
`0.93773 0.33763 0 4.2108e+06`	
`0.90805 0.35641 0 4.2108e+06`	
`0.81915 0.32 0 4.2108e+06`	
`0.97192 0.278 0 4.2108e+06`	
`0.944 0.29474 0 4.2108e+06`	données (ici x y z)

Figure 2 Exemple d'un fichier PCD

[7] www.pointclouds.org

1.1.3 Consolidation et prétraitement du nuage sur Scene 5.2 de Faro

1.1.3.1 Définition de la consolidation

Par relevé scanner laser terrestre fixe (SLT fixe), un environnement a peu de chance d'être balayé en une unique station. En général, on positionne le scanner en plusieurs endroits pour bénéficier de plusieurs points de vue et ainsi recouvrir l'ensemble de la scène. On limite ainsi les masques et les obstacles.

Après l'acquisition des données, nous obtenons plusieurs nuages de points indépendants pour chaque station. Il faut alors lier ces différents points de vue, ces différents nuages afin de reconstituer la scène numérisée dans un système de coordonnées unique. Cette reconstitution est réalisée à partir d'objets communs (sphères et cibles) et/ou à partir de zones du nuage communes à plusieurs stations. Cette première étape est appelée la consolidation du nuage.

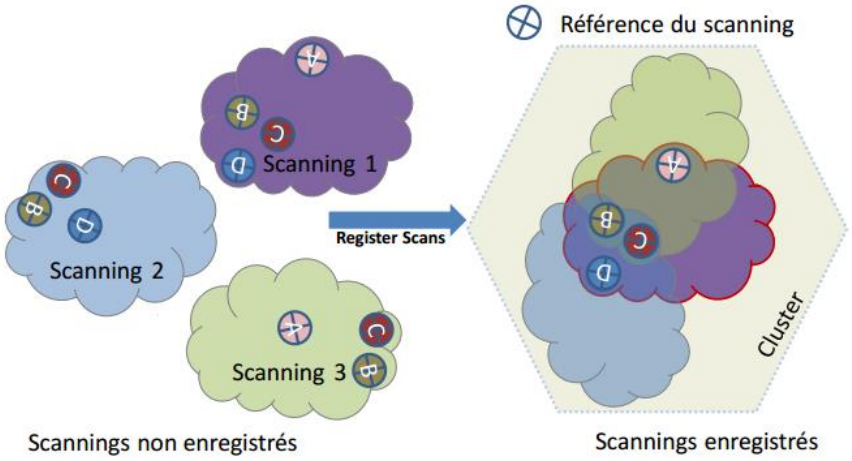


Figure 3 Schéma de la consolidation (Faro, 2014)

La consolidation est basée sur une transformation d'Helmert 3D à six paramètres que sont les trois translations et les trois rotations (matrice R des rotations) selon les axes x,y et z :

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix} + R * \begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix}$$

Équation 3 Transformation d'Helmert à 6 paramètres

Cette équation est ensuite linéarisée afin de procéder à un ajustement itératif par moindres carrés. Plusieurs algorithmes sont rassemblés et décrits dans (Shawa, 2006) : consolidations dépendante des entités géométriques (i.e. sphères) ; consolidation basée sur RANSAC[8] et consolidation ICP[9]. Cette dernière est la plus répandue dans les logiciels de consolidation de nuage (Shawa, 2006).

[8] RANdom SAmple Consensus : méthode itérative pour estimer les paramètres d'un modèle mathématique à partir d'un ensemble de données observées.

[9] Iterative Closest Point : méthode de minimisation itérative de la distance entre deux nuages de points.

1.1.3.2 Le prétraitement sur Scene 5.2 de Faro

Scene de Faro est le logiciel constructeur du Faro Focus 3D S. Il offre un large éventail d'outils pour traiter les nuages de points, pour ne citer que la consolidation, la sélection et l'édition de points.

Edition de points :

Le logiciel permet d'éditer le nuage de points en offrant une large gamme d'options : sélection de points, suppression de fantômes[10], allégement du nuage par échantillonnage etc.

La consolidation :

La consolidation de nuages est appelée dans Scene l'enregistrement (de l'anglais « registration »). Il permet de raccorder différentes stations à partir d'objets d'attaches appelés références. Il peut s'agir de références « naturelles » (points de coin, plans, lignes etc.) ou de références « artificiels » (cibles et sphères posés sur le terrain et identifiés lors du traitement). Leurs déterminations peuvent se faire de manière manuelle ou automatique (Figure 4).

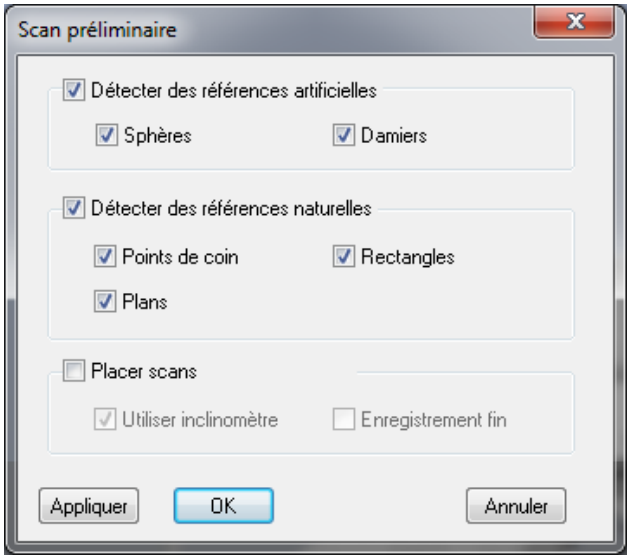


Figure 4 Consolidation (Scene 5.2)

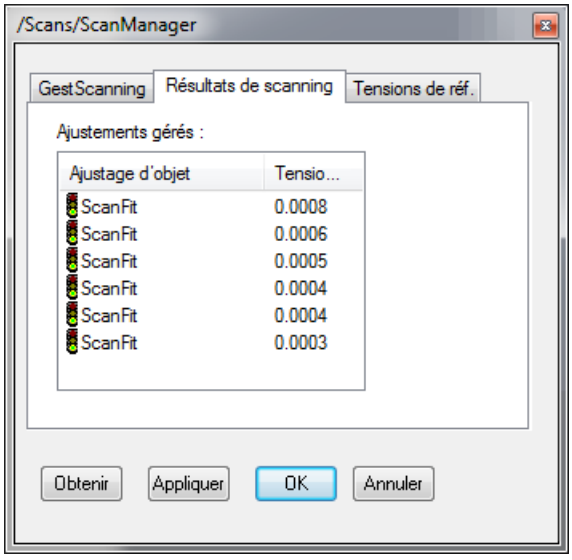


Figure 5 Résultat Consolidation (Scene 5.2)

Le résultat de la consolidation est affiché à partir de la fenêtre ScanManager (Figure 5). Les tensions correspondent aux résidus pour chaque station après ajustement[11]. L'outil implémenté dans Scene est très efficace. La solution automatique peut parfois aboutir à des résultats aberrants qui peuvent toutefois être corrigé par une intervention de l'opérateur (forcer les références par noms, créer des clusters[12] cohérents etc.). La consolidation est très efficace sur ce logiciel. Il n'est donc pas intéressant de développer une autre solution pour la consolidation de nuage.

[10] Les fantômes sont des ensembles de points dans le nuage issus du balayage de passant, de voitures en mouvement etc. donnant des formes étranges

[11] Après optimisation par le logiciel de l'assemblage initial

[12] Un cluster est un sous-ensemble du nuage de points

1.2 Segmentation automatique du nuage

La plupart des traitements de nuage de points 3D s'inspirent des traitements numériques d'images 2D (Vosselman, 2004). Ainsi, (Pu & Vosselman, 2006) définissent la segmentation dans un nuage de points 3D comme étant un « processus de labélisation de chaque points du nuage de points, de façon à ce que ces points, appartenant à telle ou telle surface ou région, aient le même label ». Cette définition, proche de la segmentation en traitement des images 2D, peut être complétée avec une définition plus générale. Selon (T. Landes P. G., 2011b) : « la segmentation d'un nuage de points est une subdivision de l'ensemble des points 3D en sous-ensembles homogènes, suivant des critères prédéfinis ».

C'est donc une étape clé dans la reconstruction de l'environnement 3D car elle identifie chaque sous-ensemble comme entité indépendante. Par exemple, dans le cas d'une façade, deux plans différents seront identifiés par segmentation et traités indépendamment pour la modélisation.

Il existe deux grandes familles de segmentation (Vosselman, 2004) que sont la segmentation par reconnaissance de formes géométriques et la segmentation par agrégation de points.

1.2.1 La segmentation par reconnaissance de formes géométriques

Ce type de segmentation estime les paramètres mathématiques d'un objet (plan, cylindre etc.) à partir d'un ensemble de points cohérents. Dans la littérature, deux algorithmes sont généralement utilisés : la transformée de Hough et l'algorithme RANSAC.

1.2.1.1 La transformée de Hough

Proposé par (Hough, 1962), la transformée de Hough a été énormément utilisée pour la détection de formes dans le traitement d'images numériques et la vision par ordinateur[13]. Pour cela, l'algorithme détecte des formes paramétrées de l'espace euclidien (i.e. de l'espace cartésien de l'image 2D) à partir d'un espace paramétré appelé « espace de Hough ».

Par exemple, dans le cas d'une prise de vue photographique d'une façade, nous souhaiterions acquérir les droites principales qui caractérisent cette façade (Figure 6 Photographie d'une façade et analyse par transformée de Hough (ESGT)). Pour un ensemble de points $M_i(x_i, y_i)$ définis dans l'espace cartésien de l'image (O, x, y), il existe un ensemble de droites y_i passant par ces points. Ces droites d'équations $y_i = ax_i + b$ ont pour paramètres a et b qui varient en fonction de la droite choisie.

Le principe de l'algorithme est de représenter dans un espace paramétré (O', a, b) l'ensemble des droites par leurs couplets (a, b). Ainsi deux points passant par la même droite $y = ax + b$ dans l'espace image correspondront dans l'espace paramétré à l'intersection en (a, b) de deux droites.

[13] Science qui étudie la perception et la compréhension d'un environnement par une machine

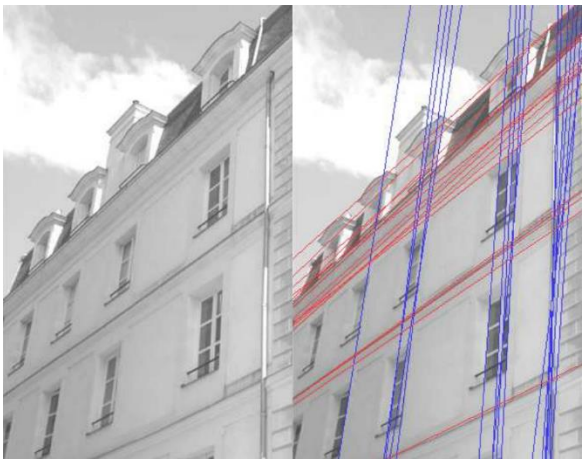


Figure 6 Photographie d'une façade et analyse par transformée de Hough (ESGT)

Vosselman décrit cette technique pour la détection de cylindres, de sphères et de plans (Vosselman, 2004). Pour lui, le traitement des plans peut notamment être accéléré par l'utilisation et la détermination des vecteurs normaux aux plans.

Une façade est composée en majeur parti de plans, c'est pourquoi ce travail reposera essentiellement sur la détection de cette primitive géométrique. Nous verrons aussi dans les parties suivantes que la plupart des algorithmes travaillent sur ce critère de normalité au plan.

1.2.1.2 L'algorithme RANSAC

L'algorithme RANSAC (pour RANdom SAmple Consensus) a été proposé pour la première fois par (Fischler & Bolles, 1981). Tout comme la transformée de Hough, RANSAC a été utilisé dans un premier temps en traitement des images 2D pour la détection de formes paramétrées. Dans le traitement de nuage de points 3D, il est essentiellement utilisé dans la consolidation et la segmentation.

Le principe est le suivant : l'algorithme sélectionne en premier lieu et de manière aléatoire un ensemble de points. Cet ensemble est utilisé pour déterminer une première primitive géométrique dont sa validité sera testée sur l'ensemble des points du nuage. La notion d'outliers et d'inliers[14] (Burk, 2009) est souvent reprise pour décrire cet algorithme.

Une fois la primitive trouvée au bout de n itérations, elle est ajustée en fonction de l'ensemble des points validant ce modèle. Le nombre d'itération de la boucle est estimé par (Fischler & Bolles, 1981) avec p la probabilité de trouver une primitive correcte, w le pourcentage de points corrects et s le nombre minimal pour décrire une primitive (pour un plan, il faut trois points minimum).

$$n = \frac{\log(1-p)}{\log(1-w^s)}$$

Équation 4 Itération de la boucle RANSAC

[14] Termes anglais pour points « aberrants » et points « validés » par la primitive

(Schnabel & Wahl, 2007) ont développé un algorithme de segmentation et de modélisation à partir de RANSAC pour détecter plans, sphères, cylindres, cônes et tores. Selon eux, la principale difficulté d'utilisation de RANSAC est que l'estimation de la primitive est régie par la probabilité des candidats choisis, autrement dit sur une sélection aléatoire des candidats. C'est pourquoi d'autres algorithmes basés sur le « SAmple Consensus » (SAC) sont aussi utilisés. (Dujardin, 2013) utilise par exemple l'algorithme MSAC pour la détection de cylindres (différence dans la sélection des candidats). (Schnabel & Wahl, 2007) précisent toutefois qu'il n'y a pas de gains de performances mais simplement une différence d'initialisation de l'algorithme.

Dans le cas d'un relevé par SLT, (Boulaassal, 2010) estime dans son état de l'art que « RANSAC produit également des résultats prometteurs ». Il ajoute toutefois « que la multitude de plans proches les uns aux autres » dans le cas d'un relevé de façade peut compliquer la segmentation. Il illustre son propos (Figure 7) en comparant des plans détectés par RANSAC supposés parallèles.

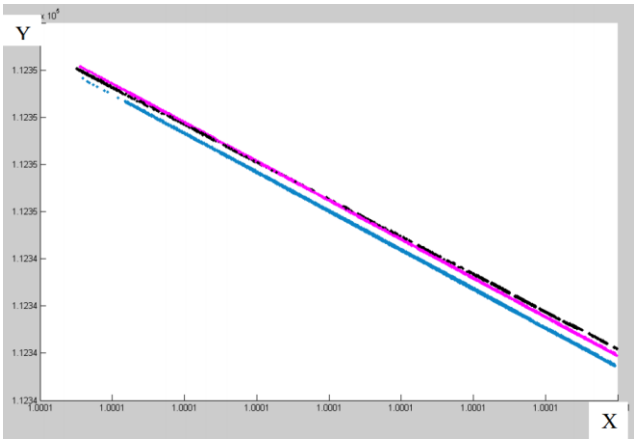


Figure 7 Profils de plans verticaux projetés (Boulaassal, 2010)

Pour résoudre ce problème, il intègre des contraintes « d'ordre géométrique et la fixation d'un ordre de détection des plans selon leur orientation ». Il ajoute que pour son travail, il combine la reconnaissance de formes géométriques et l'agrégation de points afin d'obtenir une segmentation optimale. Cette combinaison des deux algorithmes est aussi utilisée par (Ait El Kadi, Tahiri, Simonetto, & Sebari, 2013) pour la segmentation de façade.

1.2.2 La segmentation par agrégation de points : croissance de surface

Cette méthode est aussi issue du traitement numérique des images 2D. La segmentation par agrégation de points est une étude de critères locaux du nuage de points autour de graines. Ces critères, une fois définis, sont testés au voisinage de la graine. Si les points voisins répondent aux critères, alors ils sont acceptés dans ladite surface segmentée. Le processus s'arrête lorsque les voisins ne répondent plus aux critères. La segmentation par croissance de surface (encore appelé croissance de région) est une forme de segmentation par agrégation de points.

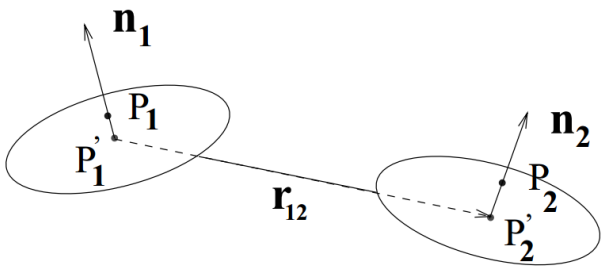


Figure 8 Deux surfaces graines et leurs caractéristiques (Stamos & Allen, 2002)

Selon (Stamos & Allen, 2002), la surface graine est définie par un plan ajusté autour de n points voisins. Ce plan local permet alors de tester les points candidats. Ils sont validés selon les critères suivants :

- Critère de proximité : seuls les n points voisins peuvent être ajoutés à la surface graine. Le voisinage est défini par un rayon r fixe ou par un nombre de points n ;
- Critère de planéité local : une fois le plan local ajusté, la distance orthogonale de chaque point au plan est testé. Cette distance doit être inférieure à un certain seuil ;
- Critère de normalité : les normales aux différents plans voisins sont testées. Leurs différences angulaires ne doivent pas dépasser un certain seuil, sinon la croissance s'arrête.

La croissance de surface est utilisée par (Pu & Vosselman, 2006) car elle est décrite comme étant plus appropriée pour la segmentation de façades.

Figure 9 Segmentation d'une façade (Pu & Vosselman, 2006)

Cependant cette segmentation est souvent sujette aux problèmes de sur-segmentation et de sous-segmentation : les critères locaux (plan local, normale, courbure) sont sensibles au bruit et aux variations locales des plans.

Les différentes méthodes de segmentation sont appliquées en fonction de plusieurs facteurs tels que l'origine des données (les méthodes de segmentation ne seront pas forcément identiques dans le cas d'un relevé par

scanner laser terrestre et d'un relevé par scanner laser aéroporté par exemple) ou encore l'environnement (relever une scène industrielle ou une façade ne répond pas aux mêmes critères etc…).

De plus, chaque méthode a ses inconvénients et ses atouts : RANSAC est robuste au bruit et aux artefacts mais a l'inconvénient de privilégier le plus grand plan de la scène au détriment des plans plus petits ; la transformée de Hough obtient de bon résultat dans la détection d'objets paramétrés mais demande un temps de calcul très important. Enfin, la croissance de région est sensible au bruit et dépend fortement des critères locaux pour croitre mais peut inversement détecter des détails assez fins.

1.2.3 CloudCompare : un outil libre pour la segmentation

Présentation du logiciel :

CloudCompare a été développé par (Girardeau-Montaut, 2006) dans le cadre d'une thèse issue de la collaboration de Telecom ParisTech et EDF. Comme l'indique son nom, le logiciel avait pour objectif initial la détection et la visualisation de différences entre des nuages de points. Aujourd'hui, la plateforme a énormément évolué pour devenir un projet indépendant, libre et gratuit de référence dans le traitement de nuage de points.

Plugin RANSAC :

Un des plugins proposé par le logiciel est la détection de primitives géométriques par l'algorithme RANSAC basé sur les travaux de (Schnabel & Wahl, 2007). Dans le cadre d'une segmentation de façade, l'outil est utilisé pour détecter des plans :

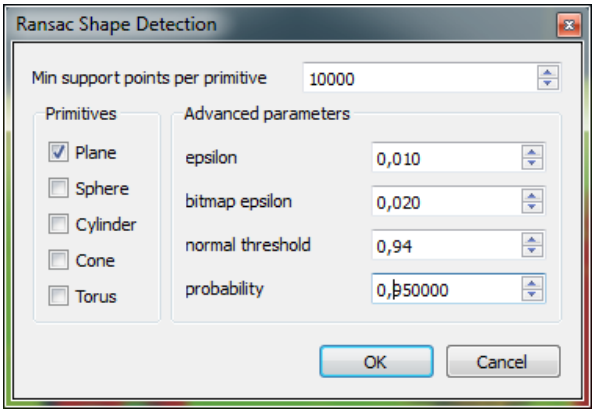


Figure 10 Boite de dialogue du plugin RANSAC de CloudCompare (Schnabel & Wahl, 2007)

« Min support points per primitive » correspond au minimum de points nécessaires pour former une primitive géométrique ; « *normal threshold »* correspond au seuil angulaire entre deux normales pour différencier deux plans voisins (la valeur à saisir est le cosinus de cet angle) ; *« epsilon »* représente la distance seuil de part et d'autre du plan, *« bitmap epsilon »* représente le pas (i.e. résolution spatiale) entre points du nuage, *« probability »* la probabilité issue de la formule de Fischler & Bolles (partie 1.2.1.2).

Résultat sur une façade parisienne :

Après plusieurs essais et différents ajustements des paramètres, le résultat le plus concluant est le suivant :

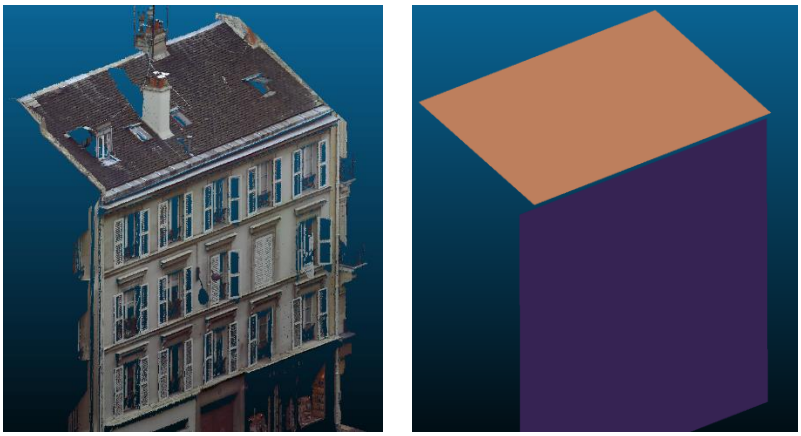


Figure 11 Détection des plans principaux sur une façade parisienne

Les résultats peuvent être validés à partir de l'outil de comparaison nuage/objet *« compute cloud/mesh distance »*. Les plans ont été largement estimés à partir des points colorisés en vert :

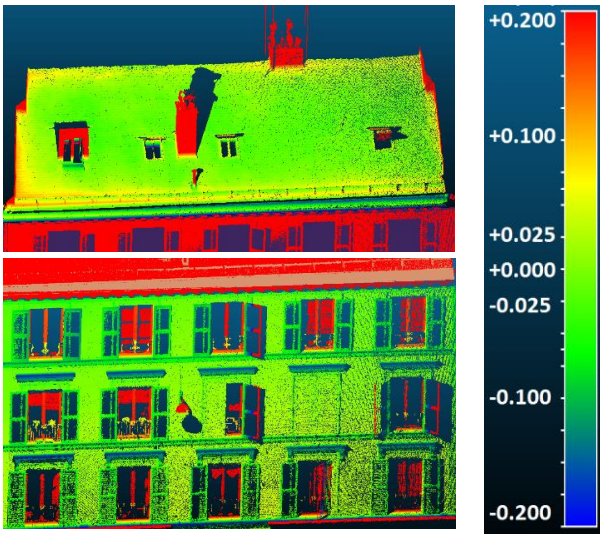


Figure 12 Utilisation de l'outil de comparaison nuage/objet dans CloudCompare

Le plugin RANSAC de CloudCompare atteste des travaux de (Schnabel & Wahl, 2007) et permet de mieux appréhender cet algorithme. La détection efficace des plans principaux par RANSAC va dans le sens des travaux de (Ait El Kadi, Tahiri, Simonetto, & Sebari, 2013) et de (Boulaassal, 2010).

1.3 Rendu 3D et modélisation

1.3.1 Généralités

Comme nous l'avons vu dans la partie 1.2, la segmentation est cruciale car elle permet de regrouper les points du nuage en fonction de critères géométriques prédéfinis. Pour obtenir un rendu exploitable, la segmentation doit être complétée par une étape dite de modélisation, permettant d'obtenir un véritable modèle 3D de la scène relevée. Cette modélisation consiste à construire l'objet en trois dimensions.

D'après (T. Landes P. G., 2011b), il existe trois sortes de modèles, qu'on peut différencier en fonction des données de départ et du rendu souhaité :

- le modèle « reconstruit » est issue d'une reconstruction d'un environnement dont les données terrains sont incomplètes. Ce modèle est en général effectué par les archéologues et les architectes pour reconstituer des vestiges historiques ;
- le modèle « tel que saisi » fourni la modélisation de l'existant tel qu'il est au moment du relevé. Comme son nom l'indique, cette modélisation est issue des données mesurées. Elle dépend donc en grande partie de la précision et de la qualité des mesures ;
- le modèle « tel que construit » permet d'obtenir un modèle réaliste de l'objet tel qu'il fut à sa construction. Cette modélisation est issue de données terrains complétées par d'autres informations (plan d'architecte, vue éclatée etc.).

1.3.2 Types de modélisations

Le travail du géomètre est de représenter le « tel que saisi » et, dans un relevé par SLT, les rendus finaux qu'il propose sont essentiellement des maquettes 3D (Amara, 2014)[15]. Dans la littérature, le processus automatique permettant de passer du nuage de points à cette maquette 3D est effectué de deux manières que sont la modélisation par maillage et la modélisation géométrique.

1.3.2.1 Modélisation par maillage

Le maillage consiste à relier les points du nuage entre eux afin de construire une surface décrivant les objets d'une scène. Le modèle maillé est alors constitué de sommets issus directement des points du nuage, d'arrêtes et de faces reliant lesdits sommets.

L'avantage de la modélisation par maillage est qu'elle suit au plus près le nuage de points et donc l'existant. Dans le cas d'une triangulation de Delaunay, l'algorithme créé des facettes triangulaires pour relier les points du nuage.

[15] Les rendus possibles sont aussi des plans en deux dimensions et des rendus bruts (nuage de points)

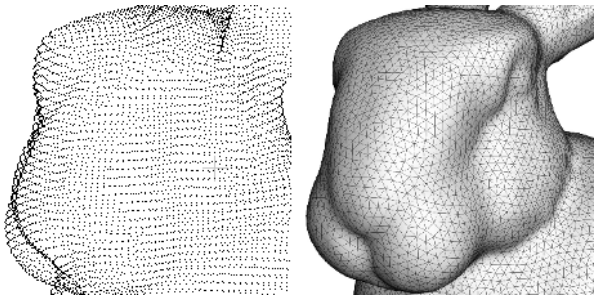


Figure 13 Triangulation de Delaunay - Lapin de Stanford (Creatis, 2014)

Cette méthode est très efficace pour représenter des modèles complexes (cathédrales, statues etc.) car elle interpole directement les facettes à partir des points du nuage. Elle a cependant l'inconvénient de produire un fichier final lourd, dépendant fortement de la qualité du nuage de départ.

Par exemple, dans le cas de la représentation du nu d'un mur, il faudrait autant de triangles que nécessaire pour relier tous les points du nuage constituant ce mur, alors que théoriquement, il suffirait de deux triangles pour le modéliser.

Ainsi, il est parfois plus adapté d'utiliser une autre technique qu'est celle de la modélisation géométrique.

1.3.2.2 Modélisation géométrique

D'après (Boulaassal, 2010) et (T. Landes P. G., 2011b), la modélisation géométrique est « basée sur la construction des éléments identifiables et descriptibles d'un point de vue mathématique dans le nuage de points ». Autrement dit, il s'agit de construire le modèle à l'aide de primitives géométriques. Dans notre cas, il s'agira de construire des plans qui correspondent au mieux au nuage de points.

Cette modélisation a l'avantage de simplifier l'objet à sa plus pure représentation mathématique. Revenons sur le cas d'un nu de mur. La modélisation géométrique permet de trouver le modèle qui correspond au mieux à ce sous ensemble de points constituant ce mur. Il en résulte un plan défini par quelques paramètres (contrairement au maillage qui donnerait un nombre important de triangles pour définir ce même mur).

Cette modélisation peut être très efficace si elle est précédée d'une segmentation qui identifiera les principales primitives.

1.3.3 Meshlab : un outil libre pour la modélisation

Développé au laboratoire italien de l'ISTI[16], Meshlab est un logiciel libre regroupant un ensemble d'outils dans la modélisation tel que le maillage de points, qui va ici nous intéresser.

Le logiciel regroupe près d'une trentaine de maillages différents. Il faut être prudent quant à leur utilisation car les outils proposés demandent de solides connaissances mathématiques, les rendant difficiles d'accès pour les non-initiés. Deux méthodes de reconstruction seront présentées : la triangulation de Delaunay et la reconstruction de Poisson.

1.3.3.1 Triangulation de Delaunay

La définition mathématique est la suivante (Pillet, 2014) : « une triangulation T d'un ensemble P de points du plan est dite de Delaunay si l'intérieur de tout triangle T ne contient aucun point de P ».

Le principe de cette méthode est de générer des faces triangulaires les plus petites et les plus équilatérales possibles afin de relier les points du nuage. La triangulation obtenue est unique : il n'existe qu'un seul résultat possible d'une triangulation de Delaunay. En trois dimensions, la triangulation d'un ensemble de points P est dite de Delaunay si aucun point P ne se situe à l'intérieur de la sphère circonscrite aux trois sommets dudit triangle.

L'outil *Delaunay Triangulation* est situé dans la catégorie *Filters, Remeshing, Simplification & Reconstruction*. La Figure 14 est issue d'un relevé du bureau de l'entreprise.

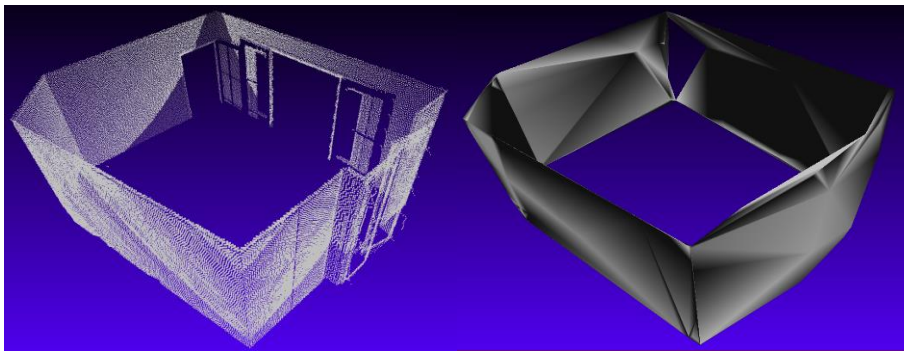


Figure 14 Triangulation de Delaunay Meshlab - Pièce d'intérieur

1.3.3.2 Reconstruction de Poisson

Cette méthode a été introduite par (Kazhdan, Bolitho, & Hoppe, 2006) pour la reconstruction de surfaces à partir d'un nuage de points. Contrairement à la triangulation de Delaunay, cet algorithme génère une surface lissée à partir d'une détermination des normales locales en chaque point :

- Les normales sont déterminées. Il s'agit d'un ensemble de vecteurs $\vec{V_i}$;
- On détermine $grad\ \vec{V_i}$ afin d'obtenir la fonction indicatrice qui nous permet d'identifier les points décrivant la surface ;

[16] Pour *« Istituto di Scienza e Tecnologie dell'Informazione »* (Institut des Sciences et des Technologies de l'Information)

- Finalement, la surface à reconstruire est extraite.

Quelques paramètres sont à entrer pour ajuster au mieux la reconstruction de Poisson :

- La profondeur *« depth »* correspondant à la définition de l'arbre[17] utilisé. L'arbre utilisé par défaut est l'octree ;
- Le nombre d'échantillon par nœud *« samples per node »* définissant le nombre minimum de points que doit contenir un nœud du K-D tree. Ce paramètre doit être ajusté en fonction du bruit de la mesure (1.0 - 5.0 dans le cas d'un nuage faiblement bruité, 15.0 - 20.0 pour un nuage fortement bruité) (Poux, 2013).

1.4 Conclusion de l'état de l'art

Cette première partie fut consacrée à la synthèse des travaux préexistants dans le domaine de l'extraction d'éléments géométriques dans les nuages de points. Il ressort dans cette étude bibliographique un procédé relativement commun dans le traitement des nuages de points. Ce procédé est résumé dans la figure suivante. Elle servira de base au développement d'une solution pour le traitement des relevés de façades.

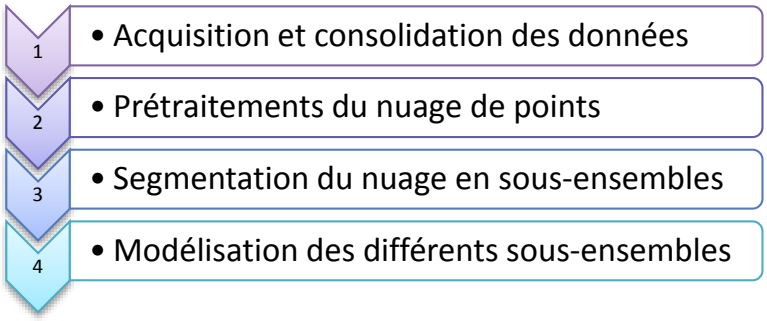


Figure 15 Processus de traitement

[17] L'arbre (en anglais « tree ») est un procédé permettant de structurer les données du nuage en le décomposant en cube. Chaque cube (ou nœud) est décomposé en x parties, x représentant le niveau du « tree ».

Partie 2 Développement du programme

Cette seconde partie est consacrée au développement d'une solution pour traiter les relevés par scanner laser terrestre de façades de bâtiments. Cette solution s'inspire en grande partie de l'étude bibliographique précédente, notamment dans la succession des étapes de traitement.

Cette seconde partie comporte trois points :

- *« Prérequis »* : il y sera expliqué quels ont été les critères déterminants pour développer l'applicatif. Par la suite, nous verrons comment le nuage est allégé et nettoyé de tout artéfact afin d'optimiser les étapes de traitements qui suivent.
- *« Segmentation en clusters »* : il y est décrit le procédé pour isoler et identifier les sous-ensembles du nuage. Les critères de segmentation y seront explicités.
- *« Modélisation des plans principaux de la façade »* : il y sera détaillé le processus de modélisation des sous-ensembles plans.

2.1 Prérequis

2.1.1 Choix de développement : la Point Cloud Library

2.1.1.1 Environnements de développement

Les premiers programmeurs travaillaient en langage machine (ensemble d'instructions correspondant à de longues écritures « binaires » composées de 0 et de 1). Par la suite, des langages de programmation ont été créés afin de faciliter l'écriture de programme. Leur « traduction » par la machine se fait de deux façons :

- le langage interprété : l'interpréteur (un logiciel) lit le programme et l'exécute ligne par ligne. L'avantage de ce type de langage est qu'il nous dédouane de toute tâche inhérente aux problèmes de gestion de mémoire, de communication avec le système etc., permettant de se concentrer aisément sur l'algorithmique. Le principal inconvénient de ce type de langage est que l'optimisation du programme est limitée par la rapidité de l'interpréteur. Matlab, Scilab et Python sont des langages interprétés.
- le langage compilé : le compilateur va transformer le code écrit en un objet exécuté en langage machine (i.e. directement exécuté par le système sans autre intermédiaire). Ce type de langage a le principal avantage d'améliorer significativement l'exécution du programme. Cependant, il a l'inconvénient d'être beaucoup plus difficile à maitriser puisque demandant des connaissances plus poussées en informatique. C++ et C sont des langages compilés.

Dans la littérature scientifique consultée, le choix d'un langage pour développer un applicatif ou un algorithme est assez partagé entre Matlab, Scilab et Python d'un côté et C++/C de l'autre. Cette question du choix du langage était extrêmement importante car elle conditionnait l'organisation même du travail à effectuer.

2.1.1.2 Un langage indépendant : le C++

Le choix du langage de programmation pour développer l'applicatif a été influencé par :

- son prix : choisir un langage libre ne demandant aucune licence propriétaire à acquérir ;
- sa rapidité : choisir un langage adapté à l'utilisation de données extrêmement volumineuses en trois dimensions que sont celles des nuages de points ;
- sa popularité : un langage partagé par le plus grand nombre et si possible par une communauté ayant un intérêt particulier pour le traitement des nuages de points.

Le choix s'est alors tourné vers le C++. Initié par Bjarne Stroustrup dans les années 1980 dans le cadre de ses travaux au laboratoire de recherche Bell, le C++ reste après plus d'une trentaine d'année un langage informatique de référence[18]. Il est notamment utilisé par la communauté de la Point Cloud Library qu'on présentera au paragraphe suivant. Ce langage permet de créer des programmes s'exécutant rapidement sur une variété d'environnements informatiques (Liberty & Cadenhead, 2011), et c'est un langage qui n'appartient à personne et est donc libre d'utilisation.

Ce choix du langage de développement a été fait bien avant le début du stage ingénieur, afin de ne pas trop me focaliser sur l'aspect prise en main de l'environnement de travail / apprentissage du langage de programmation.

2.1.1.3 Une communauté dynamique : la PCL

Avec l'avènement des scanners laser de nouvelle génération (léger, autonome, fiable), l'utilisation des données d'acquisitions en trois dimensions est de plus en plus démocratisé : industrie (scanner à bras mécanique haute précision) ; jeux vidéo, perception par ordinateur (Kinect de Microsoft) ; bâtiment (scanner laser terrestre) ; cartographie (scanner laser aéroporté). Leurs prix sont de plus en plus compétitifs et de plus en plus de personnes pourront acquérir des données 3D en temps réel et à moindre coût.

Il est alors souhaitable que des outils de traitement, des algorithmes ainsi que des normes soient partagés par le plus grand nombre. C'est exactement le rôle que joue la bibliothèque logicielle C++ Point Cloud Library (PCL). Elle vise à fournir une collection d'algorithmes et d'outils pour traiter les données nuages de points. Comme défini dans la partie 1.1.2.2, la PCL est open-source et sous licence BSD et est par conséquent libre d'utilisation par tout le monde.

Figure 16 Logo de la Point Cloud Library

[18] www.tiobe.com

Initié à l'origine par (Rusu & Cousins, 2011), le projet réunit des chercheurs, des universités, des entreprises du monde entier[19] et est devenu rapidement une référence dans le domaine de la 3D. La communauté de développement regroupe désormais près de 200 chercheurs et ingénieurs[20] et la bibliothèque est régulièrement mise à jour.

Malgré un temps de préparation conséquent avant de commencer le stage ingénieur (apprentissage du langage C++, utilisation de l'environnement de travail), l'installation de la PCL peut s'avérer être un véritable parcours du combattant pour un non initié ou pour une personne utilisant des langages interprétés. Vous pouvez consulter un petit tutoriel en annexe 2 « *Bien démarrer avec la Point Cloud Library* » si vous souhaitez découvrir et installer cette plateforme riche et dynamique.

2.1.2 Prétraitement du nuage de points

Le prétraitement des données est une étape à ne pas négliger. En amont de la chaîne de production, elle peut avoir des conséquences directes sur les traitements *a posteriori*.

Tout d'abord la consolidation du nuage de points s'effectuera sur le logiciel constructeur du Scanner Faro Focus 3D S. Les données seront par la suite exportées au format E57 afin d'être mis au format PCD par le logiciel CloudCompare. Le cheminement pour passer d'un format de fichier à un autre peut sembler laborieux. C'est cependant le moyen le plus simple pour accéder au format PCD qui on le rappelle, est supporté par la bibliothèque Point Cloud Library.

Une fois que les données sont disponibles au format PCD, les prétraitements effectués par le logiciel développé serviront à alléger et nettoyer le nuage de points.

2.1.2.1 Consolidation et export du nuage de points

La consolidation et l'export du nuage de points présentés dans cette partie ont été effectués sur un petit chantier à Fontenay-le-Fleury. Il s'agissait à l'origine d'un plan masse pour le compte d'un architecte. L'exhaustivité du relevé par scanner laser terrestre m'a permis de récupérer uniquement les stations concernées par le relevé de façade.

Figure 17 Nuage de points - Fontenay-le-Fleury (Scene 5.2)

[19] http://www.pointclouds.org/about

[20] http://dev.pointclouds.org/projects/pcl

Consolidation par références naturelles

Définies dans la partie 1.1.3.2, les références naturelles correspondent aux points, lignes et plans détectés automatiquement par le logiciel. Elles sont utilisées pour trouver une corrélation entre différentes stations recouvrant une même zone d'étude. Cette corrélation permet de consolider l'ensemble du levé.

La consolidation automatique s'effectue à partir de l'outil *« scan préliminaire »* qui regroupe un ensemble de prétraitement avant création du nuage de points.

Après plusieurs tentatives d'ajustement, la consolidation par références naturelles fonctionne dans assez peu de cas, le cas le plus favorable étant la détection de points de coins. Il faut cependant ne pas omettre d'utiliser aussi l'inclinomètre afin d'aider le traitement pour la consolidation. En effet, pendant la durée d'une station, l'inclinomètre prend constamment des mesures angulaires afin d'ajuster un plan horizontal bien défini, et ce pour chaque station.

Consolidation par références artificielles

Les références artificielles correspondent aux sphères et cibles équipant le scanner laser terrestre. Elles permettent de réaliser une consolidation beaucoup plus efficace, aussi bien en termes de rapidité de traitement que de précision. Il faut cependant être vigilant à placer suffisamment de références afin d'avoir une redondance des observations. Si les références ne sont pas suffisantes, le logiciel peut ajuster l'ensemble de manière incorrecte, tout en attestant que le résultat de l'ajustement est précis (Figure 18).

Figure 18 Ajustement erroné – Fontenay-le-Fleury (Scene 5.2)

Conclusion sur la consolidation du nuage

La consolidation du nuage est une étape essentielle dans le prétraitement du nuage. Dès lors, il est important de vérifier la précision de l'assemblage du nuage. Cependant, dans ce mémoire, il n'est pas indispensable de s'attarder sur cette question et ce pour deux raisons :

- la première raison est que, une fois l'assemblage réalisé, le logiciel Scene propose d'ajuster les tensions, c'est-à-dire de faire « coller » au mieux les deux nuages en peaufinant l'assemblage ;

- la seconde raison est la problématique même des relevés de façades. Traditionnellement, les géomètres fournissent ces plans à une certaine échelle (même si, avec l'avènement des rendus numériques, la notion d'échelle devient de plus en plus caduque) que sont celles du 50^{e} et du 100^{e}. Ainsi, pour 0,25mm (acuité visuelle moyenne) sur la lecture d'un plan, on peut atteindre au maximum une précision de report de 1,3cm au 50^{e} et 2,5cm au 100^{e}. Il faut donc s'assurer de garantir ces précisions pour la consolidation du nuage (sachant que pour un relevé de façade, on a rarement plus de 10 stations, les imprécisions étant alors très limitées).

L'ensemble des méthodes utilisés sur ce chantier est résumé dans le tableau suivant :

Consolidation	Type d'observations	Nombre d'obs.	Précision globale (m)
Références Naturelles	Points de Coins	15	0,019
	Points de Coins + Inclinomètre	17	0,012
Références Artificielles	Sphères + Inclinomètre	4	0,006

Tableau 1 Résultat de la consolidation - Fontenay-le-Fleury

L'utilisateur peut consolider son nuage suivant différentes méthodes, chacune ayant leurs avantages et leurs inconvénients. La précision globale pourra être fortement réduite dans tous les cas en procédant à un ajustement des tensions.

Export du nuage de points

Une fois le nuage de points consolidé et créé dans Scene, il faut l'exporter au format PCD afin qu'il puisse être lu par le logiciel de traitement développé dans ce mémoire. Or le format PCD n'est pas reconnu par Scene. Il faut donc préalablement exporter les données dans un format intermédiaire compatible. On pourra choisir les formats de fichiers définis dans la partie 1.1.2.2 (E57 et LAS) ou encore les formats de type ASCII (PTS et XYZ).

Ces formats de données interchangeables peuvent être aisément convertis au format PCD, soit par le logiciel CloudCompare, soit en entrant à la main les variables d'entête décrites dans la partie 1.1.2.2 pour les formats de type ASCII. L'annexe *3* « *Obtenir un fichier au format PCD avec CloudCompare* » explique en détails la procédure.

2.1.2.2 Sous-échantillonnage du nuage à partir d'une grille de voxels

Les pixels (de l'anglais *« picture element »*) sont les plus petits éléments décrivant une image 2D. Ils sont carrés, uniformes et triés par lignes et par colonnes. De même, les voxels (pour *« volumetric pixel »*) divisent l'espace 3D en cellules cubiques.

Explication :

Le premier prétraitement effectué par le logiciel est de sous-échantillonner (autrement dit, dé-densifier le nuage de point à un certain seuil pour ne pas perdre trop d'information). Le sous-échantillonnage d'un nuage de points par grille de voxels repose sur le principe suivant : l'espace est découpé par une grille de voxels v_i. Chaque voxel de dimension $n * n * n$ (n étant la longueur d'un côté du voxel) contient un certain nombre k

de points du nuage. Ces points $p_{i,k}$ sont moyennés à l'intérieur de chaque voxel afin d'obtenir un unique point p'_i.

$$p'_i = \frac{\sum_1^k p_{i,k}}{k}$$

Équation 5 Estimation du centroïde i de k points

Ainsi, chaque voxel ne contiendra plus qu'un unique point p'_i. Plus la dimension *n* du voxel sera grande, plus le nuage sera sous-échantillonné et moins il contiendra de points. La technique de sous-échantillonnage choisie (i.e. par grille de voxel) est efficace (Chin-Feng & Don-Lin, 2001) en terme de temps de calcul. Le traitement de la grille s'effectue tranche par tranche, donnant à l'algorithme une certaine efficacité. Toutefois il ne permet pas la compression des données telles que l'agrégation de voxels similaires de type « tree ».

Implémentation dans la PCL :

Cette méthode est implémentée dans la PCL. Il suffit de faire référence au header[21] en question et appeler la fonction et ses paramètres :

```
#include <pcl/filters/voxel_grid.h>
// J'incorpore à mon code source le header voxel_grid.h

int main ()

{
  pcl::VoxelGrid<pcl::PCLPointCloud2> Grille;
  // J'appelle la fonction VoxelGrid à travers l'alias Grille
  Grille.setInputCloud (nuageE);
  // Je vais chercher mon nuage de points en entrée
  Grille.setLeafSize (0.04f, 0.04f, 0.04f);
  // Je précise la dimension de mon voxel en mètre. Ici n = 4cm
  Grille.filter (*nuageS);
  // J'applique le sous-échantillonnage et j'obtiens en sortie un nuage
  return (0);
```

Équation 6 Sous-échantillonnage par grille de voxels / C++

Cette première étape permet d'alléger considérablement le nuage afin d'optimiser les traitements postérieurs. La notion d'échelle de plans (exposée dans la partie précédente 2.1.2.1) peut aider à choisir une juste valeur pour le pas *n* de sous-échantillonnage.

Fichiers	Nombres de points	Taille du fichier (mo)
Façade initiale (ascii)	2 112 629	103,5
Façade sous-éch. (n = 4cm)	255 769	7,4
Gain :	**x7**	**x13**

Table 1 Résultat du sous-échantillonnage d'une façade parisienne

[21] Le header ou « entête » en français est un fichier C++ contenant les attributs et les prototypes d'une classe. Ce header peut être importé dans le code source du programme pour faire appel à une fonction par exemple.

2.1.2.3 Nettoyage du nuage par test statistique

Le second prétraitement effectué par le logiciel est d'exclure du nuage les points issus d'artefacts divers : réfraction du signal sur une arrête de bâtiment, végétation, objets en mouvements laissant une « poussière » de mesures derrières eux (voitures, passants etc.). Ce prétraitement permet de diminuer le temps des traitements *a posteriori* et d'éviter des erreurs factuelles telles que des recherches de voisins sur des points aberrants par exemple.

Explication :

Ce nettoyage du nuage repose sur un test statistique (Rusu & Al., 2008) qui élimine les points clairsemés dans des zones d'études locales. Pour chaque point étudié, une distance moyenne est déterminée entre le point et ses voisins. En supposant que l'ensemble des distances moyennes du nuage suivent une distribution gaussienne pour une moyenne μ_k et un écart-type σ_k, alors tous les points dont la distance moyenne est en dehors d'un certain intervalle peuvent être considérés comme des valeurs clairsemées aberrantes (en anglais on parle de *« sparse outlier »*).

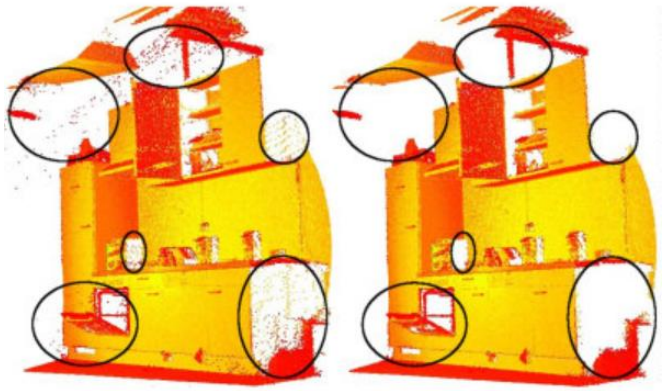


Figure 19 Suppression des points aberrants clairsemés (Rusu & Al., 2008)

Algorithme (Rusu & Al., 2008) :

Soit P^k les k-voisinages en chaque point du nuage ;

Pour chaque point $p_q \in P$, on calcule une distance moyenne $\bar{d}$ aux k-plus proches voisins ;

Ensuite, on détermine une distribution des distances $\bar{d}_i$ avec une moyenne μ_k et un écart-type σ_k estimé ;

Enfin on garde tous les points du nuage P^* dont la distance $\bar{d}$ est suffisamment proche de μ_k.

Ainsi :

$$P^* = \left\{ p_q^* \in P \mid (\mu_k - \alpha * \sigma_k) \leq \bar{d}^* \leq (\mu_k + \alpha * \sigma_k) \right\}$$

Équation 7 Nettoyage du nuage par test statistique (Rusu & Al., 2008)

Où α est le facteur de restriction de l'intervalle autour de la moyenne.

Implémentation dans la PCL :

Ce traitement est implémenté dans la PCL. Il faut alors appeler le header et utiliser les paramètres de la méthode :

```
#include <pcl/filters/statistical_outlier_removal.h>
// J'incorpore à mon code source le header statistical_outlier_removal.h

int main ()
{
    // [...]

    pcl::StatisticalOutlierRemoval<pcl::PointXYZ> Filtre;
    // J'appelle la fonction à travers l'alias Filtre
    Filtre.setInputCloud (nuageE);
    // Je vais chercher mon nuage de points en entrée
    Filtre.setMeanK (50);
    // Je spécifie le nombre de k-plus proche voisins pour la détermination de d barre
    Filtre.setStddevMulThresh (1.0);
    // Je précise le facteur alpha pour l'intervalle
    Filtre.filter (*nuageS);

    // [...]
 }
```

Équation 8 Nettoyage du nuage par test statistique / C++

Le sous-échantillonnage et le nettoyage du nuage sont les deux principales étapes de prétraitement du nuage. Elles sont importantes et ce pour les raisons suivantes :

- Le sous-échantillonnage permet d'alléger considérablement la taille du fichier et les temps de calculs à posteriori. Il permet entre autre d'uniformiser la résolution spatiale du nuage de points ;
- Le nettoyage du nuage permet de limiter les artéfacts et diminue les risques de mauvais traitement du nuage. Les points exclus du nuage ne seront plus pris en compte dans les estimations globales (RANSAC) et locales (croissance de surface).

2.2 Segmentation en clusters

Comme décrit dans l'état de l'art, la segmentation du nuage de points est une étape cruciale dans la chaîne de traitement. Elle doit être la plus précise et la plus correcte possible. Nous utiliserons ici deux méthodes de segmentation du nuage. La première est basée sur une reconnaissance de formes géométriques et la seconde sur une agrégation de points.

2.2.1 Recherche des plans principaux par RANSAC

RANSAC est très présent dans la littérature (partie 1.2.1.2) et est largement utilisé pour la segmentation de nuage de points. Il est décrit comme étant un estimateur robuste de primitives géométriques car il est capable d'estimer correctement les paramètres d'une forme géométrique à partir d'un ensemble de points malgré une forte présence de points aberrants. Autrement dit, dans le cas d'une détection de plans, l'algorithme va estimer de manière rigoureuse les coefficients définissant un plan à partir des données bruitées du scanner laser terrestre.

2.2.1.1 Algorithme proposé[22] (Rusu & Cousins, 2011)

Radu B. Rusu nous propose un procédé de traitement des plans par RANSAC dans le *header* sac_model_plane.h à la ligne 124. Il peut être résumé de la manière suivante :

$RANSAC\ (\ nuage\ de\ départ\ , largeur\ seuil\ )$
$\quad Le\ Plan = \emptyset$
$\quad Plan\ déterminé = \emptyset$
$\quad InlierMaximum = 0$
$\quad Pour\ n\ itérations$
$\quad\quad Selection\ trois\ points\ (\ nuage\ de\ départ\ )$
$\quad\quad Equation\ du\ plan\ (\ trois\ points\ )$
$\quad\quad Chercher\ Inliers\ (\ plan\ calculé, largeur\ seuil\ )$
$\quad\quad Si\ NombreInlier > InlierMaximum$
$\quad\quad\quad InlierMaximum = NombreInlier$
$\quad\quad\quad Plan\ déterminé = plan\ calculé$
$\quad\quad Fin\ Si$
$\quad Fin\ Pour$
$\quad Réajustement\ du\ Plan\ par\ SVD$[23]
$\quad Le\ Plan$
$Fin\ RANSAC$

Équation 9 Algorithme RANSAC utilisé dans la PCL

Pour détecter plusieurs plans, on peut réitérer le processus tant qu'une certaine proportion du nuage ne sera pas extraite :

$Les\ Plans = \emptyset$
$Tant\ que\ l'on\ identifie\ des\ plans$
$\quad RANSAC\ (\ nuage\ de\ départ\ , largeur\ seuil\ )$
$\quad Les\ Plans = Les\ Plans + Le\ Plan$
$\quad nuage\ de\ départ = nuage\ de\ départ - Inliers\ trouvés$
$Fin\ Tant\ que$

L'algorithme va donc sélectionner aléatoirement trois points dans le nuage de départ afin de déterminer l'équation d'un plan calculé. Ce plan calculé est ensuite épaissi d'une certaine largeur seuil afin d'accepter des points voisins du plan calculé appelés *inliers*.

Le nombre d'*inliers* trouvé est stocké et testé : si le nombre d'*inliers* est supérieur au nombre d'*inliers* maximum trouvé à l'itération n-1, alors le nombre d'*inliers* devient une valeur maximale et le plan calculé devient un plan déterminé.

Ce processus est répété n fois afin d'identifier le plan ayant le plus d'*inliers* possible, autrement dit le plan le plus grand dans le nuage (voir 1.2.1.2). Une fois que ce plan déterminé est trouvé, il est stocké en mémoire

[22] Voir l'API PCL : http://docs.pointclouds.org/trunk Module Sample_Consensus > Modèle SACMODEL_PLANE

[23] SVD : algorithme d'ajustement basé sur la décomposition en valeurs singulières d'une matrice

en tant que *cluster* du nuage. Il est ensuite retiré du nuage de départ afin de l'isoler entièrement du traitement à l'itération n+1.

Enfin, le processus complet est répété tant que l'on arrive encore à identifier des plans dans le nuage de points. Cette condition peut paraitre étrange au premier abord, mais nous allons voir dans la partie suivante qu'elle permet de gagner un temps considérable dans le calcul si elle est utilisée à bon escient.

L'algorithme fournit alors une liste de plans indépendants les uns des autres. Chacun de ces plans sont ajustés par moindres carrés afin d'en affiner leurs coefficients[24].

2.2.1.2 Description des paramètres utilisés

Nous avons vu précédemment que l'algorithme RANSAC utilisé dans la PCL utilisait un certain nombre de paramètres tout au long du traitement : la largeur seuil autour du plan calculé, le nombre d'itération pour trouver les plans etc. Nous allons voir à quoi correspondent ces paramètres.

Largeur seuil :

Elle correspond à la limite d'acceptation des points par le modèle plan. L'algorithme calcule la distance des points au plan :

$$l = \frac{|a * x + b * y + c * z - d|}{\sqrt{a^2 + b^2 + c^2}}$$

Équation 10 Largeur seuil au plan

Nombre d'itération n :

Il correspond au nombre d'itération défini par Fischler & Bolles (partie 1.2.1.2). Dans le cas de la détection d'un plan, on a :

$$n = \frac{\log(1 - p)}{\log(1 - w^3)}$$

Avec : p la probabilité que les trois points sélectionnés soient des inliers ; w le pourcentage d'*inliers* dans le nuage de points. Il est facile de déterminer p si l'on souhaite maximiser les chances d'obtenir directement des *inliers* (on admet que $p = 0.99$). Il est cependant plus compliqué d'estimer le pourcentage d'*inliers* dans le nuage (autrement dit, l'importance du bruit de la mesure dans le nuage de points).

Proportion d'inliers w (%)	0,20	0,30	0,40	0,50	0,60	0,70	0,80
Probabilité p (%)	0,99	0,99	0,99	0,99	0,99	0,99	0,99
Itérations n	573	168	70	34	19	11	6

Tableau 2 Nombre d'itérations n pour RANSAC

L'algorithme implémenté dans la PCL propose d'entrer sa propre valeur d'itération. Cette valeur sera fixée permettant ainsi de s'affranchir de l'estimation du pourcentage d'*inliers* dans le nuage.

[24] $P = \{a * x + b * y + c * z + d = 0 \mid a, b, c, d \in R \text{ les paramètres du plan } P\}$

Tant que l'on identifie des plans :

RANSAC est utilisé pour déterminer les plus grands plans dans un nuage donné. Pour trouver un certain nombre de « grands plans » dans un nuage, on itère le processus jusqu'à ce qu'une certaine proportion du nuage soit traité. Autrement dit, l'algorithme continue sa détection de plans tant que X pourcent du nuage reste à traiter.

Dans le cadre d'un relevé de façade, on peut supposer qu'il y a au moins 50% des points du nuage qui définissent le plan principal de la façade et un pan de toit. Autrement dit, la boucle « tant que » s'arrête dès que 50% du nuage est traité.

Ces paramètres sont très importants et influent directement sur la qualité de la segmentation. Nous allons voir comment ils sont utilisés dans le programme et comment utiliser les fonctions de la PCL.

2.2.1.3 Implémentation dans la PCL

RANSAC est implémenté dans la PCL. Il faut alors appeler le header concerné et initialiser les paramètres (voir annexe 4 *« Ajustement des paramètres »*) :

```
#include <pcl/segmentation/sac_segmentation.h>
// Opérations mathématiques pour la segmentation
#include <pcl/sample_consensus/method_types.h>
// Définition de la méthode
#include <pcl/sample_consensus/model_types.h>
// Définition du modèle
#include <pcl/filters/extract_indices.h>
// Fonction permettant d'extraire des données

[...]
largeurRANSAC=0.05;
// Paramètre l correspondant à la largeur seuil en mètre
pourcentage=0.50;
// pourcentage de points qui ne seront pas traités

pcl::SACSegmentation<pcl::PointXYZ> Segmentation;
// J'appelle la fonction Segmentation
Segmentation.setModelType (pcl::SACMODEL_PLANE);
// Le modèle à chercher est un plan
Segmentation.setMethodType (pcl::SAC_RANSAC);
// J'utilise la méthode RANSAC
Segmentation.setMaxIterations (1000);
// La boucle RANSAC sera limité à 1000 itérations
Segmentation.setDistanceThreshold (largeurRANSAC);
// La largeur seuil définie d'après l'utilisateur
Segmentation.setOptimizeCoefficients(true);
// Booléen pour utiliser l'ajustement SVD du plan
pcl::ExtractIndices<pcl::PointXYZ> Extraire;
// J'appelle la fonction Extraire

while (nuage->points.size () > pourcentage * nr_points)
// tant qu'il me reste plus de 50% du nuage à traiter
{
    Segmentation.setInputCloud (nuage);
    Segmentation.segment (*inliers, *coefficients);
    // Détection d'un plan
    Extraire.setInputCloud (nuage);
    Extraire.setIndices (inliers);
    Extraire.setNegative (false);
    Extraire.filter (*plan);
    // J'extrait mes inliers dans un cluster "plan"

    Extraire.setNegative (true);
    // J'extrait le reste des points non traité
    Extraire.filter (*outliers);
    nuage.swap (outliers);
    // Le reste des points remplace le nuage initial
    // pour la nouvelle itération
    i++;
}
```

Équation 11 Détection des "grands plans" avec RANSAC / C++

Ce procédé est adapté pour détecter les plus grands plans du nuage. Dans le cas du relevé d'une façade, il s'agit essentiellement du nu extérieur principal de la façade et du pan de toit. Il reste alors quelques éléments à détecter tels que les corniches et les volets. Pour cela, on procède différemment. Ces éléments seront détectés par croissance de surface.

2.2.2 Recherche des objets plans secondaires par croissance de surface

Décrite dans la partie 1.2.2, l'agrégation de points par croissance de surface se base sur des critères locaux de normalités et de courbures. Ce traitement est donc plus adapté aux objets plus petits sur la façade tels que les corniches et les volets.

2.2.2.1 Algorithme proposé (Rabbani & Al., 2006)

L'algorithme repris dans la PCL est issu des travaux de Rabbani. La croissance de surface cherche dans le nuage à rassembler un ensemble de points P dans une même région R. Ces rassemblements s'effectuent autour d'échantillons appelés graines. Ces régions grandissent tant que les points candidats décrivent une même surface lisse et s'arrêtent dès qu'un contour est détecté (autrement dit dès que la courbure et l'orientation de la normale locale varient en dehors du seuil).

Le traitement commence par sélectionner une graine dont la courbure locale est relativement faible. Il vérifie ensuite pour chaque point candidat l'appartenance ou non à la région de ladite graine. Le critère est basé sur la différence d'orientation entre deux normales locales.

Si le seuil angulaire entre deux normales locales est validé, la courbure locale est aussi testée. Elle doit être inférieure à un certain seuil. Les différents paramètres utilisés sont définis dans le paragraphe suivant.

2.2.2.2 Description des paramètres utilisés[25] (Urshakov, 2014)

S. Urshakov a implémenté dans la PCL le code C++ issu des travaux de Rabbani sur la segmentation par agrégation de points.

k-voisins :

Nombres de voisins définissant la surface locale autour de la graine. Ce nombre dépend du pas de sous-échantillonnage du nuage (voir l'annexe 4 *« Ajustement des paramètres »*).

Orientation des normales locales :

Ce paramètre correspond au seuil angulaire entre deux normales locales pour valider l'appartenance des points voisins à la région. Le seuil angulaire est définie par :

$$\cos^{-1}(\vec{n}, \vec{n}_i) \leq \theta_{seuil}$$

Équation 12 Seuil angulaire entre deux normales locales

Le seuil angulaire dépend du bruit de la mesure et du nombre de k-voisins (voir l'annexe 4 *« Ajustement des paramètres »*).

Seuil de courbure locale (Rabbani & Al., 2006) :

Trouver une normale locale revient à estimer le meilleur plan tangent à une surface donnée (i.e. une surface locale décrite par les points locaux du nuage) :

[25] Voir l'API PCL : http://docs.pointclouds.org/trunk Classes > Class Index > RegionGrowing

$Soit\ g\ un\ point\ du\ nuage\ correspondant$ à $la\ graine$

$Soit\ p_i\ les\ points\ voisins$ à $g\ et\ \vec{n}\ la\ normale\ au\ point\ g\ décrivant\ le\ plan\ local\ P$

$Alors\ la\ distance\ entre\ p_i\ et\ P\ est\ d_i = (p_i - g) * \vec{n}$

Le problème est alors résolu par moindres carrés. La solution pour $\vec{n}$ est donnée par les valeurs propres et les vecteurs propres de la matrice de covariance $C \in \mathcal{M}_{3,3}$:

$$g = \bar{p} = \frac{1}{k} * \sum_{i=1}^{k} p_i$$

$$C = \frac{1}{k} * \sum_{i=1}^{k} (p_i - \bar{p}) * (p_i - \bar{p})^T \ , C * \vec{v}_j = \lambda_j * \vec{v}_j \quad , j\epsilon\{0,1,2\}$$

Équation 13 Matrice de covariance pour l'estimation d'un plan tangent local

Avec $\lambda_j \in \mathbb{R}$ les valeurs propres de *C* et $\vec{v}_j \in \mathbb{R}^3$ les vecteurs propres de *C*.

La courbure locale est directement approximée à partir des valeurs propres. Si $\lambda_0 = \min(\lambda_j)$, la courbure locale peut être estimée par :

$$\sigma = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}$$

Il s'agit du ratio entre la valeur propre minimale et la somme des valeurs propres de *C*. Plus σ est petit, plus les points de l'ensemble *P* sont proches du plan tangent à la surface. Cette valeur dépend fortement du nombre de voisins *k* et du bruit de la mesure (voir l'annexe 4 *« Ajustement des paramètres »*).

2.2.2.3 Export en clusters de nuage

Une fois la segmentation faite, le nuage est divisé en sous-ensembles appelés clusters. Chaque cluster représente un élément plan du nuage indépendant du reste du nuage. Par ailleurs, chacun de ces clusters comporte une information dudit plan ajusté. Cette information est stockée sous forme d'un vecteur $\overrightarrow{n'}\ (a, b, c, d) \in \mathbb{R}^4$ où a, b, c, d représentent les coefficients du plan. Ce vecteur conservé en mémoire est utilisé par la suite dans la modélisation des différents objets plans.

2.3 Modélisation des plans principaux de la façade

La modélisation consiste à reconstruire un ensemble discret (le nuage de points) en un objet continu en trois dimensions. Dans l'état de l'art, nous avons vu que cette reconstruction pouvait se faire par maillage ou par reconstruction géométrique (partie 1.3.2).

Dans cette partie, nous allons tenter d'utiliser les deux méthodes de reconstruction en même temps afin de :

- simplifier l'objet à sa plus pure représentation mathématique à travers la reconstruction géométrique. Pour cela, nous procéderons à une projection des points sur ledit plan ;
- faciliter la reconstruction 3D par la mise en œuvre d'un maillage qui consistera à relier ces points projetés entre eux afin de construire la surface.

2.3.1 Projection des points sur le plan

Les coefficients du plan segmenté par RANSAC sont utilisés pour la reconstruction géométrique du plan. Pour cela, l'ensemble des points segmentés sont projetés sur ce plan.

2.3.1.1 Récupération de la normale unitaire du plan

La première étape est de récupérer le vecteur normal unitaire au plan détecté par méthode RANSAC. Soit P un plan défini par :

$$P = \{a * x + b * y + c * z + d = 0 \mid a, b, c, d \in \mathbb{R}, (x, y, z) \in \mathbb{R}^3\}$$

Équation 14 Equation cartésienne d'un plan

Les facteurs réels a, b, c, d sont appelés coefficients du plan. Par définition, ils constituent les composantes du vecteur normal au plan $\overrightarrow{n} = \vec{u} \wedge \vec{v} = [a\ b\ c]^T$, où $\vec{u}$ et $\vec{v}$ sont deux vecteurs directeurs du plan. Il faut ensuite normaliser le vecteur afin de pouvoir l'utiliser aisément dans les calculs qui suivront. Le vecteur normal unitaire devient $\overrightarrow{n_u} = \left[\frac{a}{\|\vec{n}\|} \quad \frac{b}{\|\vec{n}\|} \quad \frac{c}{\|\vec{n}\|}\right]^T$.

2.3.1.2 Projection des points sur le plan[26]

La projection s'effectue en deux étapes :

- La distance de projection AH (définie comme étant la distance la plus courte du point au plan $d(A, P) = \min(AM)$) est calculée pour chaque point $p_i(x_i, y_1, z_i)$: $AH = \overrightarrow{n_u} * p_i$;
- Les nouvelles coordonnées des points p_i sont déterminées en retirant la longueur *AH* sur chaque composante : $p_{projeté} = p_i - \overrightarrow{n_u} * AH_i$.

La projection des points permet d'obtenir un plan physiquement délimité et non plus infini.

2.3.2 Reconstruction par triangulation rapide

2.3.2.1 Triangulation rapide des points[27]

Cette méthode de reconstruction crée un maillage de triangles désordonnés pour relier les points entre eux. Contrairement à la triangulation de Delaunay, cette reconstruction n'est pas unique et est beaucoup moins gourmande en ressource.

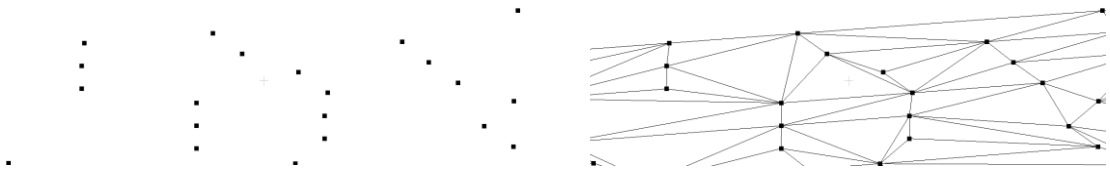


Figure 20 Nuage projeté et faces générées par triangulation

[26] Voir l'API pcl/sample-consensus/sac_model_plane.hpp

[27] Voir l'API pcl/surface/gp3.hpp

2.3.2.2 Format du fichier modélisé

Chaque plan modélisé est sauvegardé dans un fichier au format PLY (pour « *Polygon File Format* ») qui décrit les objets modélisés comme un ensemble de polygones.

```
ply
format ascii 1.0
element vertex 158116
property float x
property float y
property float z
property uchar red
property uchar green
property uchar blue
element face 304939
property list uchar int vertex_indices
end_header
107.583 151.86 98.1594 27 22 26
109.83 158.229 98.154 121 121 100
109.836 158.244 98.154 112 109 94
109.841 158.26 98.155 124 121 108
109.953 158.576 98.1535 125 123 110
***
3 574 927 1287
3 271 574 1287
3 573 271 1287
3 269 573 1287
3 76 269 1287
3 926 76 1287
3 925 926 1287
3 925 924 1287
3 572 926 925
```

Nombre de sommets

Structure des points : x, y, z et r, g, b

Nombre de polygones

Extrait de la liste des points

Extrait de la liste des polygones de types triangles (3) :
3 sommets, sommet 1, sommet 2, sommet 3
Exemple avec le triangle 1 :
3 sommets, sommet n°574, n°927, n°1287

2.4 Conclusion du développement du programme

Le développement du programme s'est articulé autour des différentes étapes dégagées dans l'état de l'art. Il s'est concentré sur l'aspect algorithmique du traitement de nuages de points. Vous pouvez consulter l'annexe 5 *« Algorigramme du procédé de traitement »* afin d'avoir un aperçu plus général des étapes de traitement. La partie suivante sera orientée sur la validation des résultats et l'amélioration de la prise en main du programme.

Partie 3 Amélioration et validation du programme

Concevoir une application peut s'avérer être une tâche ardue pour quelqu'un qui n'est pas développeur de formation. Ainsi, lorsque l'on développe un logiciel informatique en *« standalone »*, c'est-à-dire fonctionnant de manière autonome (sans être le greffon ou l'extension d'un programme préexistant), il faut être particulièrement vigilant quant aux tests du logiciel afin de rendre son utilisation la plus efficiente possible.

Le développement du logiciel pour le traitement de nuage de points et l'extraction d'éléments géométriques dans le cas d'un relevé de façade devait ainsi communiquer avec le système et l'utilisateur de la manière la plus efficace possible et traiter les données en utilisant des procédés exacts issus de la littérature.

Le logiciel sera étudié sur la façon dont il communique avec son environnement. A partir de cette étude, le logiciel devra être optimisé dans son traitement des données, afin de l'utiliser sur un échantillon de relevés de façades pour avoir un premier aperçu des résultats.

3.1 Interface avec l'utilisateur et le système

Pour reprendre ce qui a été dit précédemment, il est assez compliqué pour un non-spécialiste du domaine de l'informatique de produire un logiciel conciliant simplicité d'utilisation et exactitude des traitements. Souvent, l'algorithmique est traité de manière approfondie (soit en développant ses propres calculs, soit en se basant sur des travaux préexistants) au détriment de l'interface avec l'utilisateur et le système. Nous verrons ici comment l'interface a été conçue afin de rendre l'utilisation du logiciel la plus agréable possible.

3.1.1 Structure logique du programme

Le programme est divisé en quatre parties que sont : un dossier appelé *« data »* contenant les données ; un ensemble de bibliothèques dynamiques nécessaires au bon fonctionnement des traitements ; un exécutable (accompagné du code source) où se trouve le cœur du programme ; et un fichier de configuration des paramètres qui sera décrit dans le paragraphe 3.1.2.

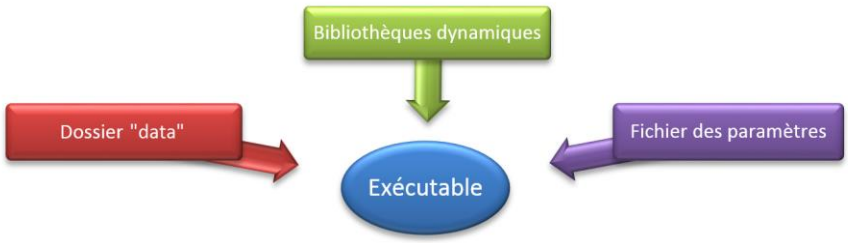


Figure 21 Structure logique du programme

3.1.1.1 Le dossier *« data »*

Le dossier *« data »* contient l'ensemble des données en entrée et en sortie utilisé par l'exécutable. Selon que l'utilisateur souhaite utiliser un traitement précis (par exemple le sous-échantillonnage d'un nuage) ou un ensemble de traitements (par exemple un nettoyage par test statistique du nuage suivi d'une détection des plus grands plans par la méthode RANSAC), le dossier sera utilisé en conséquence.

Ce dossier fait partie intégrante du programme. Il ne doit pas être supprimé ou renommé. Si c'est le cas, il suffit de le recréer sous sa dénomination d'origine.

3.1.1.2 Les bibliothèques dynamiques

Les bibliothèques dynamiques (en anglais *« Dynamic Link Library »*) sont l'ensemble des fichiers *.dll utilisé par le logiciel durant son exécution. Ces fichiers *.dll sont issus de la compilation de la bibliothèque Point Cloud Library (voir annexe 2 *« Bien démarrer avec la Point Cloud Library »*) et correspondent à différentes boîtes à outils de la PCL. La lecture des données, la segmentation, la visualisation et la modélisation sont autant de boîtes à outils mis à dispositions dans ces fichiers *.dll. Ils sont donc *de facto* indispensables à l'exécution du logiciel.

3.1.1.3 L'exécutable

L'exécutable est le cœur du logiciel. Il s'agit d'un fichier *.exe correspondant au code source C++ compilé en code binaire s'exécutant directement par la machine (revoir le paragraphe 2.1.1.1). Il regroupe l'ensemble des traitements décrits dans la Partie 2 *« Développement du programme »* que sont :

- Le sous-échantillonnage en grille de voxels ;
- Le nettoyage par test statistique ;
- La segmentation des plus grands plans par RANSAC ;
- La modélisation par triangulation des plans RANSAC ;
- La segmentation des détails par croissance de surface.

Ces différents modules sont indépendants les uns des autres. Si l'utilisateur ne souhaite utiliser qu'une partie de la chaîne de traitement, il le peut. Cette sélection est disponible à travers le fichier des paramètres de configuration.

3.1.2 Fichier des paramètres de configuration

Le fichier des paramètres de configuration *_config.ini est l'interface principale pour contrôler le comportement de l'exécutable, autrement dit du logiciel. Ce fichier des paramètres sera lu par l'exécutable afin d'initialiser les paramètres utilisés tout au long du traitement. Ces paramètres peuvent être :

- des variables utilisées directement dans les modules (par exemple le seuil angulaire entre deux normales est utilisé par l'algorithme de croissance de région) ;
- ou des variables changeant le comportement du programme (par exemple l'exécution ou non du sous-échantillonnage du nuage).

Existant depuis 1985[28], le format de fichier INI est conçu pour stocker sous forme d'un fichier texte un ensemble de paramètres regroupés en sections. Chacune de ces sections est écrite entre crochets. Chaque paramètre est indiquée par $parametre = valeur$. Ce fichier est présenté de manière plus complète dans l'annexe 6 *« Tutoriel du logiciel »*.

```
;fichier de configuration du logiciel. Conservez une copie de ce fichier

;données et processus de traitement
[donnees]
;format: forme du fichier souhaité en sortie 1:binaire, 2:ascii
format=1

;sous-echantillonnage par grille de voxels
[sousech]
;taille en mètre du voxel
voxel=0.03
```

Figure 22 Extrait du fichier de configuration *_config.ini

3.1.3 Lecture, écriture et affichage des données

3.1.3.1 Le nom du projet et la lecture des données

Une fois l'exécutable lancé, le programme demande de saisir **le nom du projet**. Ce projet correspond au préfixe des différents fichiers utilisés dans le programme. Ainsi, si le nom du projet saisi par l'utilisateur est **facadenord**, alors le programme initialisera ses paramètres d'après le fichier de configuration intitulé **facadenord**_config.ini du dossier *« config »* et lira en entrée le nuage **facadenord**.pcd du dossier *« data »*. D'ailleurs, tous les fichiers en sortie contiendront pour racine commune **facadenord**. Autrement dit, il faut vérifier que la racine des noms des fichiers *_config.ini et *.pcd sont concordants avec le nom du projet. La saisie d'un nom de projet permet de traiter *n* cas en même temps en lançant *n* fois le programme.

3.1.3.2 L'écriture des données

L'écriture des données, que ce soit pour les nuages de points ou les modèles 3D triangulés se font dans le dossier *« data »* :

- les nuages sont enregistrés au format *.pcd. Ce format de fichier peut être de la forme ASCII (lisible par l'être humain) ou binaire (lisible directement par la machine) selon que le paramètre *format* de la section *[donnees]* est 0 ou 1 ;
- les modèles 3D sont enregistrés au format *.ply. Pour obtenir un fichier au format *.dxf (utilisation pour AutoCAD), il faut utiliser CloudCompare qui est compatible avec la plupart des formats de données 3D ;
- Par ailleurs, le programme fournit un certain nombre d'informations lors du traitement (rappel des paramètres, temps de traitement, résultat etc.).

[28] Créé par Microsoft pour ses systèmes d'exploitation Windows

3.2 Optimisation du programme

Le développement du logiciel s'est effectué en deux temps :

- Il s'est d'abord focalisé sur l'aspect algorithmique, c'est-à-dire sur le traitement des données et les résultats des fichiers en sortie. Le traitement des données a été vu dans les sous-parties 2.2 *« Segmentation en clusters »* et 2.3 *« Modélisation des plans principaux de la façade »*. Les résultats des fichiers en sortie seront vus à travers les échantillons relevés dans la dernière sous-partie 3.3 ;
- Par la suite, le travail fut orienté sur l'amélioration de la prise en main du logiciel (partie précédente 3.1 *« Interface avec l'utilisateur et le système »*) et sur son optimisation.

C'est ce dernier point, l'optimisation, qui sera présenté ici. Pour ce faire, Nous montrerons comment les paramètres ont été ajustés afin de proposer une configuration par défaut du logiciel, puis nous verrons quelles sont les données que nous pouvons visualiser pendant la chaîne de traitement. Enfin, nous étudierons un cas d'optimisation de temps de calculs : celui de la détermination des normales locales.

3.2.1 Ajustement des paramètres

Une fois l'aspect théorique des paramètres étudié (sous-parties 2.2 et 2.3), il a fallu les tester à titre expérimental afin d'en connaître leurs portés. Cette partie se veut totalement illustrative en termes d'essais. C'est pourquoi les tests ont été effectués sur une simple table. Ce n'est pas la seule raison : de par son faible poids, cet échantillon-test permet notamment de gagner du temps de calculs et de tester plus rapidement les paramètres.

Cette table a été scannée à partir d'un dispositif Kinect de Microsoft. Le nuage comporte 460 400 points pour un poids de 5,38mo au format PCD Binaire.

3.2.1.1 Prétraitement

Sous-échantillonnage :

Différentes valeurs de densités (i.e. du pas de sous-échantillonnage) ont été testées afin de recueillir quelques informations.

pas d (mm)	d.f	points pt	pt.f	poids pi (ko)	pi.f	temps t (ms)	t.f	commentaire
2,5	100	460 400	100	5 517	100	-	-	nuage d'origine
5,0	25	141 525	31	1 474	27	183	100	
10,0	6	41 049	9	475	9	173	95	
20,0	2	11 598	3	138	3	147	80	informations perdues

Tableau 3 Différents pas de sous-échantillonnage - Table

La surface du pas (mm²), le nombre de points et le poids du fichier diminuent de la même façon. Il faut être prudent car un pas x fois plus grand correspond à une surface x^2 fois plus grande. Il faut aussi ne pas trop sous-échantillonner le nuage, synonyme de perte d'informations. Le temps de calcul diminue sensiblement en fonction du pas de sous-échantillonnage.

Nettoyage statistique :

Ce traitement teste la densité des points dans l'espace et exclu les points dits « clairsemés ». Pour ce faire, l'algorithme compare la distance moyenne locale $\bar{d}$ d'un point p_q du nuage à celle de l'ensemble des points du nuage (μ_k, σ_k) (cf. Équation 7). L'utilisateur saisie en entrée le nombre de k-plus proches voisins (k-ppv) pour la détermination de $\bar{d}$ et α_{sigma} le facteur de restriction autour de la moyenne.

sigma	k-ppv	points pt	pt.f	temps t (ms)	t.f	commentaire
1,0	200	436 491	95	11 343	100	pertes d'informations
1,0	100	447 407	97	4 526	40	
1,0	50	451 410	98	2 181	19	résultat optimal

Le nombre de *k-ppv* influe sur la suppression des points clairsemés : plus *k-ppv* est grand, plus la suppression des points est importante. Il influe aussi sur le temps de calcul : plus *k-ppv* est petit, plus le temps de calcul est faible.

sigma	k-ppv	points pt	pt.f	temps t (ms)	t.f	commentaire
0,5	100	411 032	89	4 506	40	trop restrictif
1,0	100	447 407	97	4 526	40	correct
2,0	100	456 409	99	4 536	40	optimal

Tableau 4a et 4b Différents sigma et k-ppv pour le nettoyage par test statistique - Table

La valeur α_{sigma} influe directement sur la suppression des points : la distance moyenne locale de p_q doit appartenir à l'intervalle $[\mu_k \pm \alpha_{sigma} * \sigma_k]$. Plus α_{sigma} est élevée, plus l'intervalle est grand et moins il y aura de suppressions de points.

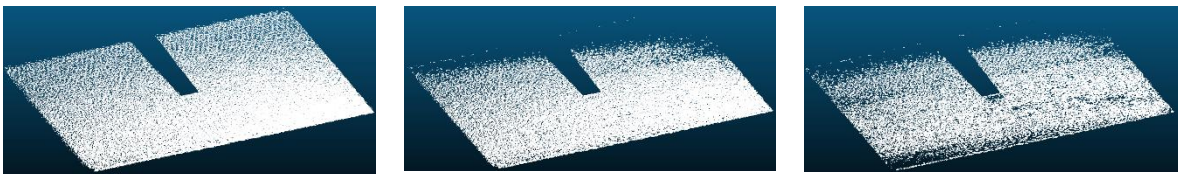
Les résultats visuels sont disponibles en annexe 7 *« Résultats complémentaires »*.

3.2.1.2 Segmentation

Segmentation RANSAC :

RANSAC permet de détecter le plus grand plan dans un nuage donné : pour chaque i-plan déterminé à partir de trois points échantillons, l'algorithme va compter le nombre de points valides et choisir le plan où il y a le plus de points valides. Ces derniers sont testés en fonction de leurs distances au plan. Si elles sont inférieures à un certain seuil l, alors les points sont validés.

Le seuil doit être ajusté de façon à conserver le maximum de points sur le plan détecté tout en évitant les points aberrants (i.e. les points qui sont proches du plan mais qui ne le décrivent pas).



$l = 0{,}8\ mm$ $l = 0{,}4\ mm$ $l = 0{,}1\ mm$

Figure 23 Différents seuil *l* pour RANSAC - Table

Plus le seuil l est faible et plus le critère de sélection des points est contraint. L'algorithme prendra alors du temps à déterminer le plan. Plus le seuil est large et plus il y aura de points validés par le modèle en raison d'une meilleure prise en compte des imprécisions de mesures, de la surface etc.

seuil (cm)	équation du plan	points pt	t (ms)	t.f	similarité (°)
0,8	0,002933	110 611	93	3	-
	0,865816				
	0,500353				
	0,493717				
0,4	0,001926	106 776	160	6	± 0,073
	0,865886				
	0,500238				
	0,493559				
0,1	0,002375	68 908	2 734	100	± 0,141
	0,866780				
	0,498684				
	0,491584				

Tableau 5 Différents seuil l pour RANSAC - Table

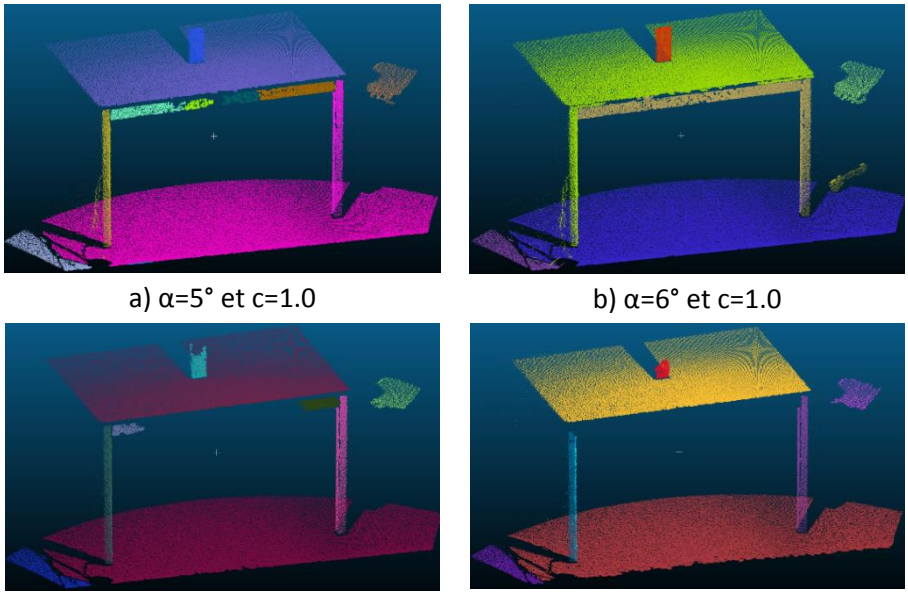
Aussi, il est important de noter qu'en fonction du seuil, le plan détecté sera sensiblement différent. On peut le voir en comparant la similarité cosinus des normales aux plans définie par :

$$s\left(\overrightarrow{n_1}, \overrightarrow{n_2}\right) = \cos^{-1} \frac{\overrightarrow{n_1} * \overrightarrow{n_2}}{\|\overrightarrow{n_1}\| * \|\overrightarrow{n_2}\|}$$

Équation 15 Similarité cosinus entre deux vecteurs

Segmentation par croissance de surface :

L'agrégation de points par croissance de surface se base sur des critères locaux de normalités et de courbures. De fait, le traitement est extrêmement sensible aux imperfections locales et peut donner très rapidement des résultats de sur-segmentation ou de sous-segmentation.



a) α=5° et c=1.0 b) α=6° et c=1.0

c) α=5° et c=0.1 d) α=10° et c=0.05

Figure 24 Différents cas de croissance de surface - Table

Les cas a) et b) correspondent respectivement à une sur-segmentation et une sous-segmentation. Le critère de courbure locale semble trop sous-estimé. Les cas c) et d) correspondent à deux segmentations relativement correctes. Ces deux derniers ont deux approches différentes : c) est plus focalisé sur l'angle des normales tandis que d) est plus axé sur la courbure locale.

3.2.1.3 Modélisation

La modélisation par triangulation rapide implémentée dans le programme permet de représenter l'objet plan en 3D de manière simple. Seule la dimension des triangles est à renseigner par l'utilisateur.

3.2.2 Visualisation du nuage

Visualisation Toolkit (VTK) est un logiciel libre de visualisation et de traitement de données 2D/3D. Il se compose d'une bibliothèque C++ qui peut être utilisé à partir de différents interpréteurs (Java, Python) et sur différents systèmes (Windows, Mac, Linux). VTK est une dépendance de PCL pour les traitements de reconstructions de surface, l'export de données 3D et la visualisation.

Cette dernière est proposée en option -dans le logiciel- pour la segmentation par croissance de région étant donné qu'il s'agit du traitement le plus difficile à ajuster.

3.2.3 Un cas d'optimisation de temps de calculs : les normales locales

3.2.3.1 Première approche

La similarité des normales locales est un des critères de segmentation par croissance de surface. Il est donc nécessaire en premier lieu de générer en chaque point du nuage un vecteur normal du plan local des k-plus proche voisins.

PCL fournit dans son module *features* deux procédés de détermination des normales locales :

- La première méthode est séquentiel (*SISD : Single Instruction on Single Data*). Les normales locales sont calculées les unes après les autres. Nous le retrouvons dans la classe *PCL::NormalEstimation*[29] ;
- La seconde méthode est parallèle (*SIMD : Single Instruction on Multiple Data*). Les normales locales sont calculées en simultanées, fonction du nombre de tâches ouvertes. Nous le retrouvons dans la classe *PCL::NormalEstimationOMP*[30].

Dans ce second cas, PCL utilise la bibliothèque OpenMP qui est spécialisée dans les traitements parallèles multitâches.

[29] Voir API normal_3d.h

[30] Voir API normal_3d_omp.h

3.2.3.2 Fonctionnement

PCL::NormalEstimationOMP fonctionne de la manière suivante :

$norm = \emptyset$

$NormalEstimation\ (\uparrow norm)$

$\#ordre\ au\ compilateur : la\ variable\ \uparrow norm\ sera\ partagée\ entre\ tous\ les\ threads$

$Pour\ i : 0 \rightarrow n_{pts}$

$Calcul\_k_{ppv}\ (\downarrow pt_i, \downarrow k, \uparrow ppv_i)$

$Calcul\_Normal\ (\downarrow pt_i, \downarrow ppv_i, \uparrow plan_i)$

$Coefficients\ (\downarrow plan_i, \uparrow norm_i)$

$Fin\ pour$

$Fin\ NormalEstimation$

OpenMP permet de gérer de manière plus aisé le multitâche. Dans le cas de la génération de vecteurs normaux, le gain de temps est estimé de 2 à 4 fois plus rapide que lors d'un traitement séquentiel.

3.3 Echantillon de relevés

Après avoir vu quelle était la portée des paramètres en entrée et comment était conçue la chaîne de traitement, quelques essais ont été faits sur des relevés de façades à Paris. La capitale s'est métamorphosée au fil des siècles et acquiert un paysage urbain relativement singulier (Carbonnier, 2009). Ces changements se traduisent notamment à travers les différents styles de façades que l'on retrouve dans la ville.

Les échantillons de relevés tentent de retracer une partie des différentes façades que l'on peut retrouver dans la capitale. Ces échantillons-tests sont autant de cas susceptibles d'être traités par le logiciel. Par ailleurs, les tests ont été effectués avec un ordinateur portable 64bits équipé d'un processeur Intel Core 2 Duo @ 2.26GHz et de 6Go de mémoire RAM.

3.3.1 Façade du milieu du XIX^{ème} siècle, rue de Lévis

L'enchaînement des défaites Napoléoniennes entre 1814 et 1815 laisse place à un retour de la souveraineté monarchique : la Restauration. Les constructions luxueuses se font rares et l'habitat populaire va connaître un essor important (Larbodière, 2000). Le bâtiment rue de Lévis est caractéristique de cette époque : construction sur quatre étages, immeuble très simple et ouvertures denses.

Figure 25 Rue de Lévis - Photographie

Le nuage suivant sera testé :

Contenu	Points	Poids (mo)	Format
Façade Principale	2 112 629	49	binaire RGB

Aucun prétraitement n'est effectué sur ce nuage de points du fait de sa légèreté (résolution spatiale du scanner de 6mm/10m) et d'un nettoyage manuel correct.

Figure 26 Rue de Lévis - Nuage

3.3.1.1 Détection du nu principal

La détection du nu principal du mur de la façade s'est déroulée en deux temps : le premier calcul a été fait avec la largeur seuil par défaut $l = 2cm$. Après un premier résultat correct, un second calcul a été fait avec $l = 3cm$ afin d'avoir les quelques points du mur manquants.

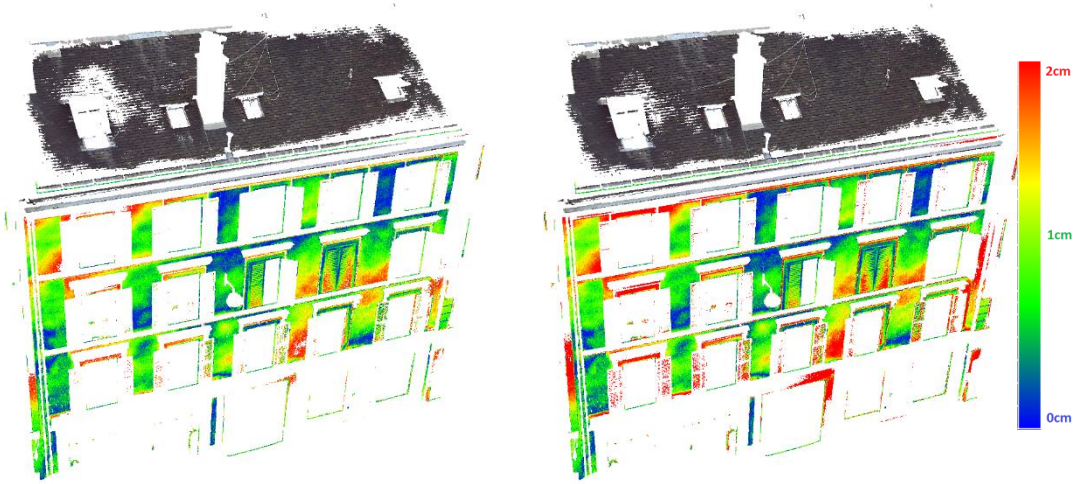


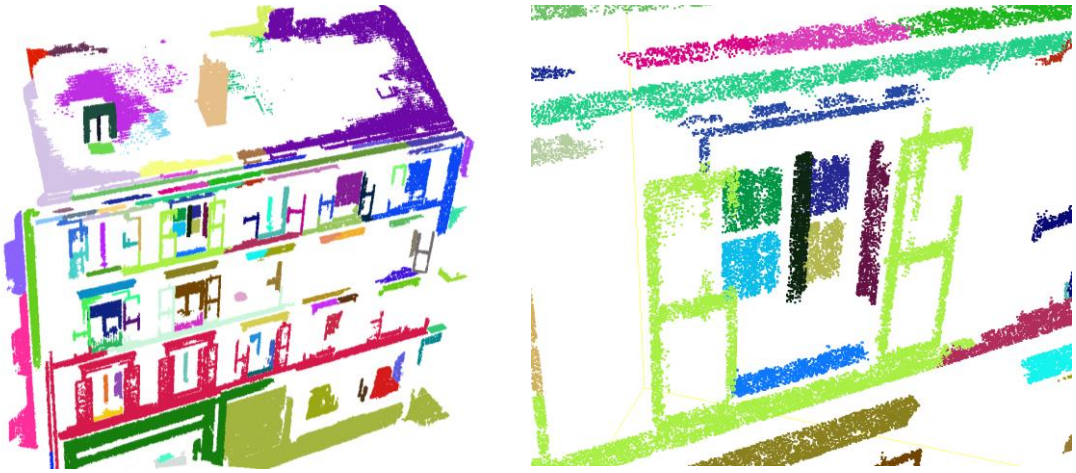
Figure 27 Rue de Lévis - Plan principal

Les deux plans détectés sont relativement similaires au vu des résultats de la figure 27. Les écarts de distances des points du nuage par rapport aux plans calculés (CloudCompare) montrent un léger saillant du mur sur l'angle haut gauche de la façade.

La modélisation des plans est traitée en *annexe 7 - façade 1*.

3.3.1.2 Détection des détails de la façade

La détection des détails est effectuée par croissance de surface avec $\alpha = 10°$ et $c = 0{,}5$. Le résultat est un peu trop juste : certains objets de la façade sont sous-segmentés tandis que d'autres sont sur-segmentés.



vue générale | vue détaillée

Figure 28 Rue de Lévis - Croissance de surface

La vue détaillée nous montre un premier cas de sur-segmentation. En haut de l'image, nous pouvons voir que la corniche de l'immeuble est segmentée en trois surfaces (violet, rose, et vert foncé). Le second cas visible est celui d'une sous-segmentation. La surface verte claire décrivant les volets continue de croitre sur la corniche basse de l'immeuble.

3.3.2 Façade Post-Haussmannienne, rue de la Tombe-Issoire

Lorsque Napoléon III arriva au pouvoir en 1852, il souhaita faire de Paris une ville européenne majeure et décida d'entreprendre de grands travaux afin de métamorphoser la capitale. L'année qui suit, il nomme Haussmann préfet de Seine afin de diriger ces grands changements. Paris se transforme alors en un énorme chantier et la ville accueille des immeubles d'un nouveau style architectural qu'on nommera plus tard le Haussmannien. Ce style influencera pendant longtemps l'architecture parisienne, notamment à travers ce qu'on appelle le Post-Haussmannien. L'immeuble rue de la Tombe-Issoire, construit en 1912, appartient à ce style architectural : façade en pierre de taille, rez-de-chaussée et entresol striés, transition avec l'étage supérieur par un balcon filant et signature de l'immeuble par l'architecte.

Figure 29 Rue de la Tombe-Issoire - Photographie

Le relevé de cette façade est différent des autres relevés. Les paramètres du Faro Focus 3D ont été changés : la résolution spatiale a été augmentée (3 mm/10m), le bruit a été réduit en augmentant le facteur de mesures d'un même point (facteur 2 à 4 en fonction de la station), cela afin de voir l'influence de ces facteurs sur le traitement des données. Le nuage de points est alourdi en conséquence.

Contenu	Points	Poids (mo)	Format
Façades d'angle	33 995 152	665	binaire RGB

3.3.2.1 Extraction des éléments sans prétraitement

Le prétraitement a été exclu de la chaîne de traitement afin de voir si le programme est capable d'extraire correctement des éléments du nuage de points.

Ransac à 40% du nuage traité					Croissance de surface, kppv = 50				
seuil l (cm)	nb. plans	simi. p1 (°)	simi. cm/10m	temps t1 (h)	normales α (°)	courbure c	clusters	% pts. segmentés	temps t2 (h)
1,5	8	-	-	1h14	5,0	0,1	42	5,4%	36h02

Le premier constat est que, sans prétraitement, l'extraction d'éléments géométriques devient très chronophage[31]. De plus, les résultats sont peu concluants :

- les grands plans détectés sont incomplets. Ceci peut s'expliquer par les déformations du nu de la façade et par une résolution spatiale r très variable. Ainsi, pour une position du SLT à 5m du mur, la résolution spatiale sera de $r = 1{,}5mm$ au pied de ce mur et de $r = 4{,}5mm$ à son sommet à 15m de hauteur. De plus, la résolution spatiale est décuplée si une zone est scannée en plusieurs endroits. Dès lors que l'algorithme RANSAC compte le nombre de points pour valider un plan (Équation 9), ce dernier apportera plus d'importance à une zone où la résolution spatiale est petite ;
- la croissance de surface est très limitée. Etant par définition sensible au bruit de la mesure, nous avons volontairement augmenté le nombre de mesures d'un même point afin d'en réduire les effets.

[31] D'autant plus que le traitement n'utilise qu'un seul cœur de processeur les 9/10e du temps. Cela permet aussi de voir qu'il reste encore des améliorations à porter au niveau de l'optimisation du logiciel.

Malgré cela, la segmentation n'a pas fonctionnée. A noter que la croissance de surface dépend de la recherche des k_{ppv}. La croissance de surface est donc elle aussi sensible aux variations de r.

Lecture :

en couleur issue de la photographie : le plan RANSAC détecté à la première itération ;

en noir : le plan RANSAC détecté à l'itération suivante. L'algorithme a pris en compte les zones « oubliées » par le premier traitement ;

en couleur : la segmentation par croissance de surface.

Figure 30 Rue de la Tombe-Issoire - Résultat sans prétraitement

Le même travail a été réalisé avec un prétraitement léger. L'analyse est disponible en *annexe 7 – façade 2*.

3.3.2.2 Extraction des éléments après prétraitement

Après avoir vu précédemment l'importance de prétraiter le nuage de points (et après plusieurs tentatives cf. *annexe 7 – façade 2*), les paramètres suivants ont été utilisés afin d'extraire les éléments de la façade :

Prétraitement				Ransac à **25%** du nuage traité				Croissance de surface, **kppv = 60**					
ss-éch. d (cm)	points nuage	t. stat. sigma	t0 (mn)	seuil l (cm)	plans	simi. cm/10m	t1 (mn)	norm. α (°)	courb. c	clust.	surf. fausses	% pts. Seg.	t2 (mn)
2,0	4 231 110	-	0,3	**5,0**	2	± 3,1	2,7	**7,0**	**1,5**	185	10%	45%	51,0

Tableau 6 rue de la Tombe-Issoire - Résultat

Le seuil l a été élargi afin d'avoir un maximum de points candidats pour les deux seuls plans ajustés (25% du nuage traité). Le reste du nuage est traité par croissance de surface. L'angle $\alpha = 7{,}0°$ a été ajusté afin d'obtenir des résultats optimaux. Pour éviter une sur-segmentation, le paramètre k_{ppv} est passé de 50 à 60 afin que les normales locales soient moins sensibles au bruit. Enfin, le seuil de courbure a été augmenté afin de ne pas avoir de cas de sur-segmentation sur les oriels incurvés.

A - Plans majeurs

B – Autres segments plans

C - Vue sur l'angle de l'immeuble

Figure 31 Rue de la Tombe-Issoire - Résultat avec prétraitement

Lecture de la Figure 31 : les plans extraits par RANSAC sont relativement complets (A) tandis que les surfaces de détails sont encore légèrement sous-segmentées (B, C).

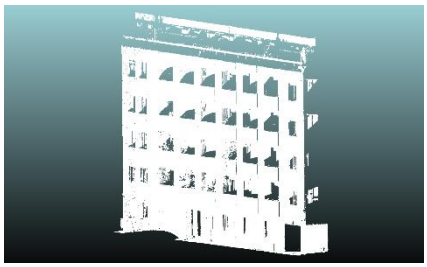
3.3.3 Façade récente, rue du Père Corentin

A partir de 1977, Paris se dote d'un Plan d'Occupation des Sols (POS) et d'un Schéma Directeur d'Aménagement et d'Urbanisme (SDAU). Le style « quantitatif » des trente glorieuses est remplacé dans les années 70 par une architecture moins surdimensionnée (Larbodière, 2000). Le bâtiment rue du Père Corentin a été construit en 1979. La façade est caractérisée par des ouvertures en biais contrastant avec une certaine monotonie du nu principal de la façade.

Figure 32 Rue du Père Corentin - Photographie

Les deux nuages suivants seront testés :

Contenu	Points	Poids (mo)	Format
Façade Principale	4 313 181	66	binaire sans couleur
Façade + Façades voisines	4 999 068	76	


Nuage 1 : façade principale


Nuage 2 : ensemble de façades

Figure 33 Rue du Père Corentin - Nuages

3.3.3.1 Façade principale

La façade principale correspond à un nuage nettoyé de toutes façades voisines. La chaîne de traitement est partielle : seuls les prétraitements et la détection RANSAC ont été effectués.

Prétraitement				Ransac à 40% du nuage traité			
sous-éch. *l* (cm)	points nuage	test-stat. sigma	temps t0 (s)	seuil (cm)	plans détectés	points segmentés	temps t1 (s)
1,0	3 262 606	1,0	36	2,0	1	1 378 370	131

Tableau 7 Rue du Père Corentin - Résultat RANSAC

Le traitement s'est déroulé en 2'47''. Les paramètres par défauts ont été utilisés. Le nu du mur principal a été détecté correctement. Il servira de comparaison avec l'autre plan détecté dans le nuage regroupant l'ensemble des façades voisines.

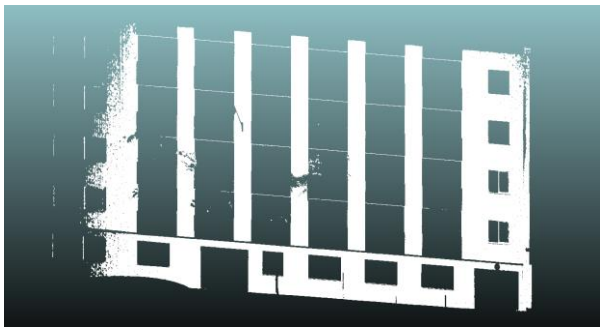


Figure 34 Rue du Père Corentin - Plan principal

3.3.3.2 Façade principale et façades voisines

Ce nuage permet de tester l'influence des façades voisines sur la segmentation du nu principal de la façade. En effet, nous allons voir que la segmentation RANSAC est sensible à l'ensemble des points proches à $\pm l$ du plan détecté. Nous avons aussi procédé à la détection des détails par croissance de surface. Le traitement s'est déroulé en 8'56''.

Ransac à 40% du nuage traité						Croissance de surface, kppv = 50					
seuil (cm)	plans détectés	points segmentés	similarité plan 1 (°)	similarité à 10m (cm)	temps t1 (s)	normales α (°)	courbure c	surfaces détectées	surfaces aberrantes	pts surf. segmentés	temps t2 (s)
2,0	2	1 800 064	± 0,067	± 1,2	282	14,0	0,100	53	oui	1 763 479	254

Tableau 8 Rue du Père Corentin - Résultat façades voisines

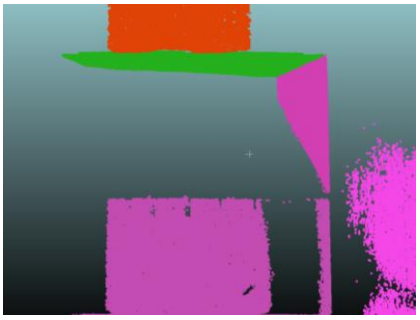
Cette fois-ci, deux plans ont été détectés par RANSAC (le premier plan détecté a moins de *« poids »* dans le nuage ce qui entraîne la détection d'un second plan pour atteindre les 40% du nuage traité).

Détection des plans principaux :

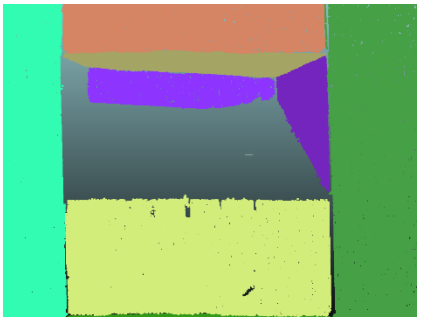
Le résultat est un peu juste pour la détection RANSAC. Le plan détecté est sensiblement différent de celui détecté dans le paragraphe 3.3.3.1. En comparant la similarité cosinus de ces deux plans, nous retrouvons une différence angulaire entre les normales de 0,067° soit 1,2 cm à 10m. Nous pouvons aussi observer les écarts en utilisant l'outil de comparaison nuage/nuage de CloudCompare (*annexe 7 - façade 3*). Il est donc nécessaire de bien isoler la façade sur laquelle nous souhaitons travailler afin d'éviter des interférences des plans voisins dans le calcul du plan principal.

Détection des éléments plans secondaires :

La segmentation est assez décevante et les éléments plans sont clairement sous-segmentés. L'angle entre deux normales α est trop élevé. Le traitement a été relancé avec un angle $\alpha = 5°$. Les surfaces détectées (au nombre de 109) sont de meilleures qualités, même si subsistent quelques défauts. Nous pouvons voir sur la figure suivante un exemple d'un balcon segmenté :



segmentation pour $\alpha = 10°$



segmentation pour $\alpha = 5°$

Figure 35 Rue du Père Corentin - Croissance de surface

3.3.4 Conclusion sur la validation du programme

Le développement du programme a porté sur :

- la chaîne de traitement observée dans la littérature (Figure 15 *Processus de traitement*) ;
- les algorithmes implémentés dans la Point Cloud Library (Partie 2 *Développement du programme*) ;
- l'amélioration du programme afin d'essayer de le rendre le plus pertinent possible.

Les échantillons de relevés ont permis de donner un premier aperçu des résultats obtenus :

- la segmentation des plans principaux par RANSAC est efficace dans la plupart des cas ;
- la segmentation des détails par croissance de surface est sensible au bruit et aux paramètres d'entrées. Il faut y consacrer un temps important pour ajuster les bonnes valeurs des paramètres ;
- la modélisation par triangulation rapide est globalement peu satisfaisante, notamment sur les résultats décrits en annexe 7 - façade 1.

Conclusion générale et perspectives

Conclusion

L'étude réalisée de mars à août 2014 dans le cabinet G2S a été concrétisée dans ce mémoire ingénieur. Elle avait pour objectif de proposer un applicatif pour automatiser la production de rendu 2D ou 3D de façades d'immeubles.

Pour se faire, cette étude s'est appuyée sur une recherche bibliographique approfondie des différents processus et méthodes existants dans le traitement des nuages de points. Le processus de traitement communément utilisé consistait à prétraiter, segmenter et modéliser les données.

Cependant, les méthodes pour arriver à ce cheminement différaient selon les points de vue et la nature des données à traiter. Des choix ont ainsi dû être faits afin de proposer un outil utile et accessible. Il a notamment été choisi de baser les travaux sur une bibliothèque *Open Source*, la *Point Cloud Library*, qui donne accès à un certain nombre d'outils performants qui ont été très utiles dans ce projet. Il a donc fallu utiliser cet outil innovant tout en conservant l'enchaînement logique observé dans la bibliographie.

La chaîne de traitement s'est caractérisée avant tout par le prétraitement des données. Il en est ressorti de cette étape un avantage considérable qui permet d'alléger -par sous-échantillonnage- et de nettoyer -par suppression des points clairsemés- le nuage de points. Le prétraitement est toutefois efficace si l'utilisateur a suffisamment isolé sa région d'intérêt du reste des données non utilisées.

Par la suite, la segmentation a permis d'extraire un certain nombre de plans majeurs de la façade (qui constituent en général le nu principal et le pan de toit) ainsi qu'un certain nombre de segments plans (détails de la façade tel que les corniches). Tandis que les éléments majeurs sont détectés par reconnaissance de formes géométriques, les détails sont quant à eux traités par agrégation de points.

Par ailleurs, le rendu a été conçu dans l'optique d'offrir à l'utilisateur un certain nombre d'éléments. Une partie de ce rendu est disponible sous forme de nuages de points. Il s'agit des différents éléments majeurs extraits du nuage et des différents détails différenciés par couleur. Mais une autre partie du rendu est disponible sous forme de modélisation 3D plane plus ou moins efficace. Il semblerait à première vue que la triangulation rapide n'est pas adaptée pour la modélisation de surfaces planes.

Perspectives

Le traitement de données LiDAR terrestre représente un réel défi pour l'avenir de la profession. Le choix d'utiliser un environnement de développement libre, dynamique et dont les normes sont partagées par le plus grand nombre était crucial afin de rendre l'applicatif aussi pérenne que possible. Ce travail de fin d'étude d'ingénieur se veut être un point de départ modeste de ce qui pourrait être développé en utilisant une bibliothèque spécialisée dans le traitement de nuages de points 3D.

Plus généralement, ce travail m'a permis de renforcer mes connaissances autour de cette technologie très prometteuse qu'est le Scanner Laser tout en découvrant un langage de programmation orienté dans le domaine de la 3D. Il me permettra plus tard d'être plus à même de collaborer avec des développeurs de métier.

Bibliographie

Ait El Kadi, K., Tahiri, D., Simonetto, E., & Sebari, I. (2013). Modélisation 3D des façades de bâtiments des anciennes Médina. *Revue Marocaine des Sciences Agronomiques et Vétérinaires*, 5-11.

Amara, F. (2014). Implanter le BIM dans l'entreprise. *Conférence Innovative Building.* Paris, Porte de Versailles.

Boulaassal, H. (2010). *Segmentation et modélisation géométriques de façades de bâtiments à partir de relevés laser terrestres.* Université de Strasbourg.

Burk, R. (2009). *Segmentation et classification de points 3D issus d'un scanner laser mobile.* Marne-la-Vallée: ENSG.

Chin-Feng, L., & Don-Lin, Y. (2001). A Marching Voxels Method for Surface Rendering of Volume Data. Dans *Feng Chia University* (pp. 306-313). Taiwan: IEEE.

Creatis. (2014, 06 10). *Creatis*. Récupéré sur Insa Lyon: http://www.creatis.insa-lyon.fr/site/

Dujardin, M. (2013). *Le scanner laser 3D : reconnaissance de formes et modélisation de déformations.* Le Mans: CNAM ESGT.

Faro. (2014). *Scene 5.2 Manuel de l'utilisateur.* Lake Mary: Société Faro.

Fischler, M. A., & Bolles, R. C. (1981). Random Sample Consensus : A paradigm for Model fitting with application to Image Analysis and Automated Cartography. *Communications of the ACM vol 24*, 381-395.

Girardeau-Montaut, D. (2006). *Détection de changement sur des données géométriques tridimensionnelles.* Paris: Doctoral dissertation, Télécom ParisTech.

Goulette, F. (2009). Relevés laser urbains par systèmes mobiles de cartographie. *XYZ n°119, Association Française de Topographie*, pp. 21-25.

Hough, P. V. (1962). Method and means for recognizing complex patterns. *US patent, 3(069), 654*.

Huber, D. (2011). *The ASTM E57 File Format for 3D Imaging Data Exchange.* Pittsburgh: Carnegie Mellon University.

Kazhdan, M., Bolitho, M., & Hoppe, H. (2006). *Poisson surface reconstruction.* Baltimore: Proceedings of the fourth Eurographics symposium on Geometry processing.

Larbodière, J.-M. (2000). *Reconnaître Les Façades - Du Moyen Age À Nos Jours, À Paris.* Paris: Charles Massin.

Larousse. (2014, 04 15). *Définition du scanner*. Récupéré sur Dictionnaire Larousse: http://www.larousse.fr/dictionnaires/francais/scanner/71306

Liberty, J., & Cadenhead, R. (2011). *Sams Teach Yourself C++ in 24 hours.* Indianapolis: Pearson Education.

Penasa, L. (2012). *Laserscanner Cyclostratigraphy : A new approach to refine numerical astronomical solutions and the geological time scale.* Padova, Italy: Dipartimento di Geoscienze, Università degli Studi di Padova.

Pillet, M. (2014, 06 10). *Triangulation de Delaunay*. Récupéré sur ENS Cachan: http://perso.univ-rennes1.fr/basile.pillet/publications/DeloneTrig.pdf

Poux, F. (2013). *Vers de nouvelles perspectives lasergrammétriques: optimisation et automatisation de la chaîne de production de modèles 3D.* Le Mans: CNAM ESGT.

Pu, & Vosselman. (2006). *Automatic extraction of building features from terrestrial laser scanning.* Netherlands: ITC.

Rabbani, T., & Al. (2006). *Segmentation of point clouds using smoothness constraint.* Netherlands: International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences.

Rusu, R. B., & Al. (2008). *Towards 3D Point Cloud Based Object Maps for Household Environments.* Munich: Robotics and Autonomous Systems Cloud Based Object Maps for Household Environments.

Rusu, R., & Cousins, S. (2011). *3D is here : Point Cloud Library.* Menlo Park USA: Willow Garage.

Schnabel, R., & Wahl, R. (2007, 06 01). Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum Vol. 26, No. 2*, pp. 214-226.

Shan, J., & Toth, C. K. (2008). *Topographic laser ranging and scanning: principles and processing.* CRC Press.

Shawa, M. A. (2006). *Consolidation des nuages de points en lasergrammétrie.* Nancy: ENSA - Université Henri Poincaré.

Stamos, I., & Allen, P. K. (2002). Geometry and texture recovery of scenes of large scale. *Computer Vision and Image Understanding vol. 88 n°2*, 94-118.

T. Landes, P. G. (2011). Les principes fondamentaux de la lasergrammétrie terrestre 1 : systèmes et caractéristiques. *XYZ n°128, Association Française de Topographie*, pp. 37-49.

T. Landes, P. G. (2011b). Les principes fondamentaux de la lasergrammétrie terrestre 2 : acquisition, traitement des données et applications. *XYZ n°129, Association Française de Topographie*, pp. 25-38.

Urshakov, S. (2014, 07 03). *Code Implementation.* Récupéré sur pointclouds.org/blog/trcs/velizhev

Vosselman, G. S. (2004). *Recognising strucutre in laser scanner point clouds.* Netherlands: ITC.

Zygmunt, M. (2013). *The testing of PCL : An open-source library.* Krakowie, Pologne: GLL, Uniwersytet Rolniczy.

Figures

Tableaux

Equations

Annexes

1- Fiche technique du Faro Focus 3D

2- Bien démarrer avec la Point Cloud Library

3- Obtenir un fichier au format PCD avec CloudCompare

4- Ajustement des paramètres

5- Algorigramme du procédé de traitement

6- Tutoriel du logiciel

7- Résultats complémentaires

8- Annexes numériques

1 Fiche technique du Faro Focus 3D

Caractéristiques de performances Focus^{3D} S

Unité de mesure de distance
Intervalle d'ambiguïté : 153,49 m
Portée Focus^{3D} S 120[1] : 0,60 m - 120 m en intérieur ou extérieur par éclairement faible et incidence normale sur une surface réfléchissante à 90 %
Portée Focus^{3D} S 20 : 0,60 m - 20 m à une incidence normale sur des surfaces mates réfléchissantes >10 %
Taux de scan (points /s) : 122 000 / 244 000 / 488 000 / 976 000
Incertitude de mesure[2] : ±2 mm

Bruit[3]	@10 m	@10 m - compression du bruit[4]	@25 m	@25 m - compression du bruit[4]
@ 90 % réfl.	0,6 mm	0,3 mm	0,95 mm	0,5 mm
@ 10 % réfl.	1,2 mm	0,6 mm	2,20 mm	1,1 mm

Unité couleur
Résolution : couleur jusqu'à 70 mégapixels
Couleur dynamique : adaptation automatique de la luminosité

Déflecteur
Champ de vision (vertical/horizontal) : 300° / 360°
Résolution (verticale/horizontale) : 0,009° (40 960 points 3D sur 360°) / 0,009° (40 960 points 3D sur 360°)
Vitesse max. de rotation du miroir : 5 820 rpm ou 97 Hz

Laser (émetteur optique)
Performance du laser (cw Ø) 20 mW (classe de laser 3R)
Longueur d'onde : 905 nm
Divergence du rayon : 0,19 mrad (0,011°)
Diamètre du rayon (à la sortie) : 3,0 mm, circulaire

Gestion des données et commande
Mémorisation des données : SD, SDHC™, SDXC™ ; carte de 32 GB fournie avec l'appareil
Commande du scanner : par écran tactile et WLAN
Nouvel accès WLAN : la commande du scanner à distance, la visualisation et le téléchargement des numérisations sont possibles sur des terminaux mobiles équipés de Flash®.

Multi-Capteurs
Compensateur bi-axial : Nivelle chaque numérisation ; précision 0,015° (plage de mesure ±5°)
Capteur de hauteur : Un baromètre électronique permet de calculer la hauteur relative par rapport à une valeur de référence et de l'attribuer aux numérisations.
Boussole : La boussole électronique fournit aux numérisations des données d'orientation par rapport aux points cardinaux. Une fonction de calibrage est disponible.

[1] Dépend de la lumière ambiante qui peut être une source de bruit. Une lumière vive comme celle du soleil peut réduire la portée effective du scanner. Par éclairage faible, la portée peut dépasser 120 m par incidence normale sur des surfaces hautement réfléchissantes. [2] L'incertitude de mesure se définit comme une erreur systématique de mesure à 10 m et 25 m, un sigma. [3] Le bruit est défini comme l'écart standard des valeurs sur le meilleur plan d'ajustement. [4] Un algorithme de compression du bruit peut être activé pour calculer la moyenne des points moyens dans des sets de 4 ou 16, comprimant ainsi le bruit des données brutes par un facteur de 2 à 4. Informations susceptibles d'être modifiées sans indication préalable.

Général

Alimentation électrique : 19 V (alimentation externe), 14,4 V (batterie interne)
Consommation électrique : respectivement 40 W et 80 W (pendant la charge de la batterie)
Autonomie de la batterie : jusqu'à 5 heures
Température : 5° - 40°C
Humidité : sans condensation

Connecteur du câble : situé dans le support du scanner
Poids : 5,0 kg
Dimensions : 240 x 200 x 100 mm
Maintenance / Calibrage : une fois par an
Sans erreur de parallaxe : Oui

2 Bien démarrer avec la Point Cloud Library

Avec la démocratisation des scanners laser de nouvelle génération, de plus en plus de personnes de tout bord acquièrent des données 3D en temps réel et à moindre coût. Il est alors souhaitable que des outils de traitements, des algorithmes ainsi que des normes soient partagés par le plus grand nombre. C'est exactement le rôle que joue la bibliothèque logicielle C++ Point Cloud Library (PCL). Elle vise à fournir une collection d'algorithmes et d'outils pour traiter les nuages de points 3D.

Cette annexe a pour but de retracer de manière la plus succincte possible l'ensemble des étapes quant à l'installation et l'utilisation de la PCL. Elle résulte du travail d'un non-spécialiste en informatique pour un usage précis qu'est celui du traitement des nuages de points. Cette annexe n'est donc ni exhaustive ni « informatiquement correct ». Elle ne constitue pas non plus un procédé universel pour utiliser la PCL.

Nous verrons tout d'abord une méthode d'installation (parmi d'autres) sur Windows de la PCL[32], puis nous étudierons une utilisation concrète d'un projet de développement.

Installation de la PCL 1.7.2 sur les systèmes Windows

Comme nous l'avons écrit précédemment, Point Cloud Library est une bibliothèque C++. Pour être installée, deux choix se présentent :

- soit installer une version précompilée et dans ce cas il suffit de se rendre sur leur site internet et d'installer « en quelques clics » la version la plus adaptée à son système (architecture du processeur et système, etc.). L'avantage est que ce type d'installation est très facile. L'inconvénient est que vous ne bénéficiez pas des dernières innovations de la bibliothèque (les versions précompilées sont une sorte de photographie à un instant t de la bibliothèque). La version disponible sur le site web de PCL est la version 1.6.0 ;
- soit installer la dernière version à partir du code source. L'installation est beaucoup plus ardue mais la bibliothèque aura le mérite d'être à jour en termes de développement (il n'y a par exemple pas de code implémenté pour la croissance de surfaces dans la version 1.6.0).

C'est l'installation à partir du code source qui va nous intéresser. La version actuelle de la PCL est la version 1.7.2.

[32] Sur Linux, le principe est le même. CMake y est disponible en ligne de commande et en interface graphique. Pour l'IDE et le compilateur, plusieurs choix sont disponibles dont G++ et Code::Blocks. De plus, des ressources sont disponibles sur le site internet de la PCL pour compiler à partir du code source sur Linux.

Configuration nécessaire

Pour installer et utiliser PCL, vous avez besoin d'un système de construction logicielle multiplateforme. Dans cette annexe, nous utiliserons le système libre *CMake 2.8.12.2*. Il s'agit d'un logiciel qui permet de transformer -à partir d'une liste d'instruction- un code source en un projet IDE.

Vous aurez donc aussi besoin d'un IDE (Environnement de Développement Intégré) pour finir d'installer et utiliser PCL. L'utilisation de *Microsoft Visual Studio C++ 2010 Pro.* est fortement conseillée. Beaucoup de problèmes subsistent pour PCL sur les versions *Visual 2012, 2013* et *Express*. Vous pouvez aussi utiliser *Code::Blocks* qui est un IDE libre. On obtient donc le processus d'installation (obtention d'une bibliothèque compilée) et d'utilisation (obtention de l'exécutable) suivant :

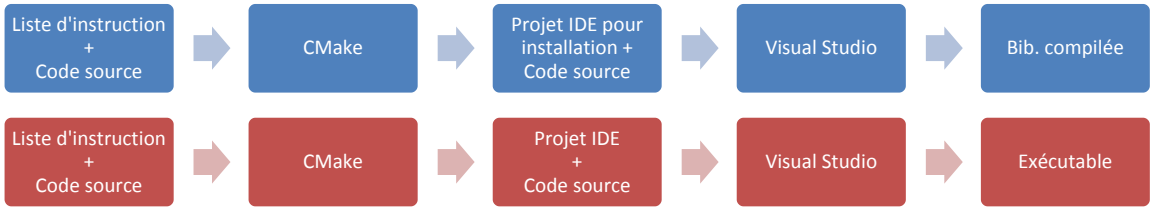


Figure 36 de CMake à Visual Studio : installation et utilisation de la PCL

Installation des dépendances[33]

La bibliothèque nécessite l'installation préalable d'autres bibliothèques appelées dépendances. Voici un tableau qui énumère les différentes dépendances en fonction de leurs importances :

Dépendances	Importance	Disponibilité	Version	Commentaire	Lien
Boost	Obligatoire	Précompilée toute version	1.55.0	Pointeurs intelligents	www.boost.org
Eigen	Obligatoire	Native en fichiers *.h	3.0.5	Calculs matriciels	wew.pointclouds.org
Flann	Obligatoire	à compiler	1.8.4	Optimisation des Kd-tree	www.cs.ubc.ca/research/flann
VTK	Obligatoire	à compiler	5.10.1	Visualisation des données 3D	www.vtk.org
Qhull	Obligatoire	à compiler	2012.1	Reconstruction 3D	www.qhull.org
Windows SDK	Obligatoire	Installeur	7.0 et 7.1	Indispensable MàJ pour VS2010	dans google "windows sdk iso"
Python	Conseillé	Installeur	2.7	Utilisé par VTK	www.python.org
OpenNI 1	Conseillé	Précompilée VS2010 Pro. 64	1.5.4	Semble utile pour le module I/O	wew.pointclouds.org
OpenNI 2	Conseillé	Précompilée toute version	2.2.0	Semble utile pour le module I/O	structure.io/openni
OpenMP	Conseillé	Supporté VS2010 Pro. 64	-	Multithreading CPU	http://openmp.org
CUDA	Facultatif	Installeur	6.0	Multithreading GPU	developer.nvidia.com
Intel PC SDK	Facultatif	Installeur	2013	-	software.intel.com
Qt	Facultatif	à compiler avec Jom	5.3	GUI, ⚠ complication avec VTK	qt-project.org
Autres	Facultatif	-	-	-	-

Tableau 9 Liste des dépendances de la PCL

[33] Voir http://pointclouds.org/documentation/tutorials/compiling_pcl_dependencies_windows.php

Windows SDK 7.0 et 7.1 pour les systèmes Windows :

Windows SDK n'est pas en soi une dépendance mais un kit de développement à jour pour les systèmes d'exploitation Windows. Il faut l'installer pour bénéficier d'une compilation sans erreur dans Visual Studio C++.

Boost :

Des versions précompilées sont disponibles sur le site internet de Boost (catégorie *Windows Binaries*). Il suffit de choisir la version en adéquation avec l'architecture du système (x86 ou amd64) et la version du compilateur de l'IDE.

Eigen et OpenNI 2 :

Eigen n'a pas besoin d'être compilée. Il faut utiliser l'installeur disponible sur le site internet de la PCL. OpenNI 2 possède des versions précompilées sur son site internet.

Flann :

1- Télécharger le zip source sur le site internet de Flann, extraire le fichier zip.
2- A l'intérieur du dossier *flann-1.8.4-src*, créer un dossier */build*.
3- **Ouvrir CMake** avec les droits administrateurs**.**
4- Entrez le chemin d'accès vers le code source de la bibliothèque (i.e. *flann-1.8.4-src*) dans *« Where is the source code ».*
5- Entrez le chemin d'accès vers où le projet IDE sera construit (i.e. *flann-1.8.4-src/build*) dans *« Where to build the binaries »*.
6- Cliquez sur le bouton ***« Configure »***.
7- Choisissez la version de l'IDE et du compilateur (dans mon cas il s'agit de Visual Studio 10 Win64. Cliquez ensuite sur *« Finish »*. CMake prépare un premier « jet » du projet.
8- Une liste d'éléments sur fond rouge apparaît dans **la première fenêtre**. Il s'agit des paramètres lus par CMake dans le fichier racine *flann-1.8.4-src/CMakelists.txt*.
   La couleur rouge veut dire qu'il faut en contrôler le contenu. Vous pouvez les regrouper par catégories en cochant *« grouped »* et *« advanced »*. Vous pourrez alors différencier les dépendances des éléments à générer pour le projet.
9- *HDF5, Ghostscript* et *Gtest* ne sont pas nécessaires pour notre compilation. Il n'est pas nécessaire de contrôler *Ungrouped entries*.
10- Dans la catégorie *BUILD*, décocher les modules *MATLAB_BINDINGS* et *PYTHON_BINDINGS*. Nous n'avons besoin de générer que le module *C_BINDINGS*.
11- Recliquez sur le bouton ***« Configure »***. Tous les éléments qui étaient rouges deviennent blancs. Le logiciel considère que vous les avez contrôlés. Si des éléments rouges apparaissent, c'est qu'ils ont été nouvellement créés en fonction de vos changements à l'étape précédente.
12- Désormais, il faut observer le contenu de la **deuxième fenêtre**. Il s'agit d'une boîte de dialogue qui décrit les étapes de préparation du projet par CMake. Le logiciel fait figurer en noir les étapes majeurs du traitement et en rouge les étapes qui posent problèmes. On peut voir dans le cas de Flann que le logiciel n'a pas trouvé *HDF5* et qu'il ne pourra pas installer de tests (*« hdf5 library not found, some*

tests will not be run »). Certains messages d'erreurs, comme celui-ci, ne seront pas importants, d'autres oui.

13- Le plus important pour pouvoir transformer le code source en un fichier projet pour l'IDE est de voir tout en bas ***« Configuration done »***.

14- Cliquer sur ***« Generate »***. Le projet est alors créé dans le dossier */build*.

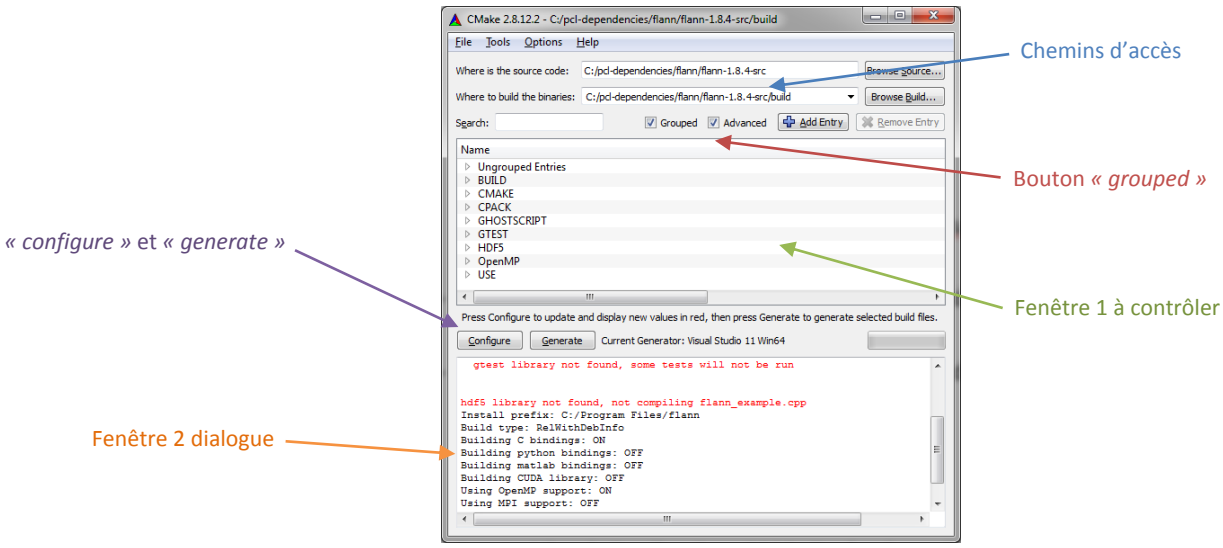
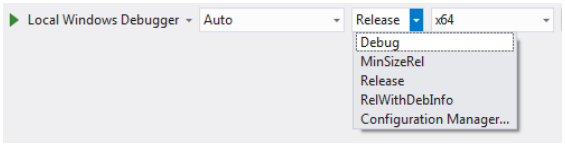


Figure 37 Fenêtre principale de CMake

15- Ouvrez **Visual Studio C++ 2010 Professionnel** (ou votre IDE préféré) avec les droits administrateurs. Ouvrez le projet *flann.sln* dans le dossier *build*.

16- Vérifiez que le panneau *« Solution Explorer »* est ouvert (en général, ce panneau est à gauche. Si absent, faire CTRL+ALT+L).

17- Vérifiez que vous êtes en mode **Debug**. Dans la barre des tâches principales, sélectionnez dans *Solution Configurations* le mode Debug.



18- Installez la bibliothèque en mode **Debug**. Dans *Solution Explorer*, sélectionnez le sous-projet *ALL_BUILD* puis cliquez droit et faites *« Build »* (ou *« générer »* en français). Une fois le traitement terminé, refaite la même chose pour le sous-projet *INSTALL*.

19- Installez la bibliothèque en mode **Release**. Réitérez le processus décrit au point 18.

20- Flann est installé !

Qhull :

Son installation suit le même principe que celle de Flann.

VTK :

Son installation suit le même principe que celle de Flann. Toutefois, quelques notas sont ajoutés ici propres à l'installation de VTK.

Dans CMake :

1- Les alertes de type *« Policy CMP0022 is not set »* dans la fenêtre 2 de dialogue peuvent être résolues en éditant le fichier *CMakeLists.txt*. Il faut modifier la première boucle *FOREACH* (celle concernant *CMAKE_POLICY(SET ${policy}* ***NEW***)) en ajoutant dans la liste des policy **CMP0022**.
2- Dans la catégorie *BUILD*, décochez *BUILD_TESTING* et cochez *BUILD_SHARED_LIBS*.
3- Dans la catégorie *VTK*, il vaut mieux ne pas utiliser *VTK_USE_QT* dans le sens où cela complique réellement l'installation.

Dans Visual Studio C++ :

4- Il se peut que vous rencontriez des alertes du compilateur dans Visual Studio C++. Il faut vérifier que les exceptions contenues dans le fichier *vtkDataArrayTemplate.h* concordent avec les alertes du compilateur.

Installation de la PCL

Il faut suivre le même cheminement que pour l'installation de Flann. Voici cependant quelques notas :

Dans CMake :

1- Dans la catégorie *PCL*, vérifiez que *PCL_SHARED_LIBS* est bien coché.
2- Vérifiez les chemins d'accès pour *QHULL* :

QHULL	
QHULL_INCLUDE_DIR	C:/Program Files/qhull/include
QHULL_LIBRARY	C:/Program Files/qhull/lib/qhullstatic.lib
QHULL_LIBRARY_DEBUG	C:/Program Files/qhull/lib/qhullstatic_d.lib

3- Vérifiez les chemins d'accès pour *OPENNI* et *OPENNI2* :

OPENNI2	
OPENNI2_INCLUDE_DIRS	C:/Program Files/OpenNI2/Include
OPENNI2_LIBRARY	C:/Program Files/OpenNI2/Lib/OpenNI2.lib

4- Vérifiez les chemins d'accès pour *FLANN* :

FLANN	
FLANN_INCLUDE_DIR	C:/Program Files/flann/include
FLANN_LIBRARY	C:/Program Files/flann/lib/flann_cpp_s.lib
FLANN_LIBRARY_DEBUG	C:/Program Files/flann/lib//flann_cpp_s.lib

5- Pour *Boost*, il est fort probable que CMake ne retrouve pas les chemins d'accès à la bibliothèque. Dans la catégorie *Boost*, entrez le chemin d'accès pour *Boost_INCLUDE_DIR* vers *C:/Program Files/boost*. Ensuite, cliquez sur le bouton 

et saisissez :

   Name: *Boost_LIBRARY_DIR*, Type: *FILEPATH*, Value: *C:/Program Files/boost/lib*, puis validez.

   Si vous n'avez pas installé *Boost* dans son répertoire par défaut, changez le chemin racine *C:/Program Files* par votre chemin. Cliquez sur le bouton *« Configure »* pour vérifier que CMake a bien réussi à lire les chemins de Boost.

Boost	
Boost_CHRONO_LIBRARY...	C:/Program Files/boost/lib/libboost_chrono-vc110-mt-gd-1_55.lib
Boost_CHRONO_LIBRARY...	C:/Program Files/boost/lib/libboost_chrono-vc110-mt-1_55.lib
Boost_DATE_TIME_LIBRA...	C:/Program Files/boost/lib/libboost_date_time-vc110-mt-gd-1_55.lib
Boost_DATE_TIME_LIBRA...	C:/Program Files/boost/lib/libboost_date_time-vc110-mt-1_55.lib
Boost_FILESYSTEM_LIBRA...	C:/Program Files/boost/lib/libboost_filesystem-vc110-mt-gd-1_55.lib
Boost_FILESYSTEM_LIBRA...	C:/Program Files/boost/lib/libboost_filesystem-vc110-mt-1_55.lib
Boost_INCLUDE_DIR	C:/Program Files/boost
Boost_IOSTREAMS_LIBRA...	C:/Program Files/boost/lib/libboost_iostreams-vc110-mt-gd-1_55.lib
Boost_IOSTREAMS_LIBRA...	C:/Program Files/boost/lib/libboost_iostreams-vc110-mt-1_55.lib
Boost_LIBRARY_DIR	C:/Program Files/boost/lib

6- Dans la catégorie *WITH*, décochez tout ce qui n'est pas nécessaire à l'installation de la PCL. Cochez *WITH_CUDA* uniquement si vous souhaitez l'utiliser (voir le site internet de Nvidia) :

WITH			
WITH_CUDA	☑	WITH_PCAP	☐
WITH_DOCS	☐	WITH_PNG	☐
WITH_FZAPI	☐	WITH_PXCAPI	☐
WITH_LIBUSB	☐	WITH_QHULL	☑
WITH_OPENNI	☑	WITH_QT	☐
WITH_OPENNI2	☑	WITH_TUTORIALS	☐
		WITH_VTK	☑

7- Dans *BUILD*, décochez tout ce qui ne serait pas utilisé par votre projet[34] :

BUILD					
BUILD_2d	☑	BUILD_global_tests	☐	BUILD_sample_consensus	☑
BUILD_CUDA	☐	BUILD_io	☑	BUILD_search	☑
BUILD_GPU	☐	BUILD_kdtree	☑	BUILD_segmentation	☑
BUILD_all_in_one_installer	☐	BUILD_keypoints	☑	BUILD_simulation	☐
BUILD_apps	☐	BUILD_ml	☑	BUILD_stereo	☐
BUILD_common	☑	BUILD_octree	☑	BUILD_surface	☑
BUILD_examples	☐	BUILD_outofcore	☑	BUILD_tools	☑
BUILD_features	☑	BUILD_people	☐	BUILD_tracking	☐
BUILD_filters	☑	BUILD_recognition	☐	BUILD_visualization	☑
BUILD_geometry	☑	BUILD_registration	☑		

8- Vérifiez les chemins d'accès pour *EIGEN_INCLUDE_DIR* et *VTK_DIR* dans *Ungrouped Entries* :

Ungrouped Entries	
CPACK_SOURCE_ZIP	☑
EIGEN_INCLUDE_DIR	C:/Program Files/Eigen/include
VTK_DIR	C:/Program Files/VTK/lib/vtk-5.8

9- Relancez *« Configure »*, normalement la plupart des alertes ont disparu. Vérifiez une dernière fois la fenêtre de dialogue pour voir si les modules générés sont conformes à vos choix.

Dans Visual Studio C++ 2010 :

Reprenez le processus d'installation décrit pour Flann. Voici cependant quelques notas :

- Si vous rencontrez des problèmes en utilisant CUDA, il se peut que vous deviez copier dans *C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\bin* le contenu du dossier *x86_amd64* vers *amd64* (pensez à conserver une sauvegarde de ce dernier). Puis dans *amd64*, créer un fichier *vcvars64.bat* et éditer son contenu en écrivant *CALL setenv /x64*.
- Si vous utilisez Intel PC SDK, il faut régénérer les **.lib* en utilisant le *sample* disponible dans *C:/Program Files (x86)/Intel/PCSDK/sample/common* pour le rendre compatible avec PCL. Ouvrez un des projets **.sln* disponibles dans le dossier :

[34] Voir la présentation des modules de la PCL dans http://pointclouds.org/documentation

- Cliquez droit sur le projet puis sur propriété. Dans *Config properties > C++ > Code Generation*. Puis dans *Runtime Library* changer pour *Muti-threaded DLL (MD)* en mode Release et pour *Muti-threaded Debug DLL (MDd)* en mode Debug.
- Ensuite régénérez le projet en debug et en release.
- Retournez dans votre projet PCL en mode Debug. Cliquez droit sur le sous-projet *pcl_io*. Changez dans *Linker > Input > Dependencies* le fichier *libpxcutils.lib* en *libpxcutils_d.lib*.

Si tout s'est bien passé, vous devriez avoir installé la PCL.

Création de son projet et utilisation de PCL 1.7.2

La bibliothèque installée est désormais constituée d'un dossier *« include »* composé des fichiers d'entêtes (**.h*, **.hpp*) décrivant un ensemble de fonctions et de classes de la PCL utilisables pour le compilateur, d'un dossier *« lib »* contenant les fichiers de codes compilés (** .lib*) pour l'éditeur de liens, et d'un dossier *« bin »* contenant les fichiers pour l'exécution du programme (**.dll*). La bibliothèque sera utilisée par votre projet IDE.

Pour le projet, nous allons recréer le cheminement de production que nous avons vu en Figure 36.

Générer le projet à partir de CMake

- Créer un dossier contenant votre fichier code source *main.cpp*, un dossier vide *build* et un fichier *CMakelists.txt* contenant les instructions CMake suivantes :

```
cmake_minimum_required(VERSION 2.6 FATAL_ERROR)
project(nom_projet)
find_package(PCL 1.3 REQUIRED COMPONENTS common io)
include_directories(${PCL_INCLUDE_DIRS})
link_directories(${PCL_LIBRARY_DIRS})
add_definitions(${PCL_DEFINITIONS})
add_executable(nom_projet main.cpp)
target_link_libraries(nom_projet ${PCL_COMMON_LIBRARIES} ${PCL_IO_LIBRARIES})
```

- Ouvrez CMake et refaite le processus habituel vu dans la partie sur l'installation de Flann.
- Vérifiez que les chemins d'accès aux dépendances sont bien interprétés par CMake.
- Démarrer Visual Studio C++. Vérifiez si les points suivants sont bien renseignés (i.e. vérifier qu'ils font bien tous références aux dépendances) :
  - *Propreties > C++ > General > Additional Include Directories*
  - *Properties > Linker > General > Additional Library Directories*
  - *Properties > Linker > Input > Additional Dependencies*

  Vérifiez notamment que figure l'ensemble des modules **.lib* de PCL dans Additional Dependencies.

Votre projet est désormais utilisable et vous pouvez commencer à écrire votre programme.

3 Obtenir un fichier au format PCD avec CloudCompare

Introduction :

Le PCD (*Point Cloud Data*) est le format de fichier des nuages de points spécifique à la Point Cloud Library (partie 1.1.2.2). Son usage est assez peu répandu en dehors de l'usage de cette bibliothèque de développement, rendant l'accessibilité à ce format de fichier un peu difficile.

Cependant, deux choses sont à noter :

- Le format PCD suit à peu de chose près la logique des autres formats de fichiers. Les métadonnées sont stockées en entête du fichier et font figurer des informations telles que la taille du fichier en nombre de points, l'information portée par les points (couleurs, normales locales etc.), la nature des données (stockées sous forme de chaîne de caractères ASCII ou sous forme binaire), etc.
- Le format PCD est libre de droit et ouvert : la structure du fichier est claire et disponible en ouvrant simplement le fichier avec un éditeur de texte.

Pour ces deux raisons, il est possible de passer d'un format de nuage de points à un autre. C'est ce que fait (entre autre) le logiciel CloudCompare (partie 1.2.3) à travers le plugin qPCL développé par Luca Penasa (Penasa, 2012).

Procédure :

	Exportez vos travaux sur votre logiciel de traitement de données brutes (Cyclone pour Leica, Scene pour Faro etc.) dans un des formats lus par CloudCompare. Le format E57 est un des formats utilisés par la plupart des constructeurs.
	Importez ledit fichier dans CloudCompare. Si vous avez plusieurs stations identifiées (i.e. plusieurs sous-ensembles importés), unifiez ces stations en un unique nuage en les sélectionnant et en utilisant l'outil *Merge Multiple Clouds*. 
	L'exportez à nouveau au format PCD en utilisant le plugin qPCL de CloudCompare (*Save to PCD file*).
	A Noter que si vous souhaitez récupérer un nuage PCD dans un autre format de nuage, vous avez la possibilité d'importer des nuages au format PCD.

4 Ajustement des paramètres

Introduction :

Les différents paramètres rencontrés lors du développement du logiciel (paramètres issus de la Point Cloud Library) peuvent paraître dans un premier temps abstrait. Il est important d'essayer de comprendre leurs portés et de connaître les facteurs pouvant influencer ces paramètres. Dans cette annexe, nous allons voir que les caractéristiques techniques intrinsèques au Faro Focus 3D S (*annexe 1 « Fiche technique du Faro Focus 3D »*) et divers facteurs propres à la scène relevée peuvent influencer ces paramètres. Ces derniers sont définis dans la partie *2.2 « Segmentation en clusters »*.

Nota : Cette annexe complète la partie 3.2.1 *« Ajustement des paramètres »*, cette dernière se voulant plus illustrative que cette annexe.

Les caractéristiques techniques propres au Faro Focus 3D S (fiche technique) :

- L'incertitude de mesure de distance $\delta_i = \pm 2mm$: selon Faro, elle se définit comme « une erreur systématique de mesure à 10m et 25m, un sigma ». Autrement dit - si on suit cette définition - la mesure de distance est entachée d'une erreur systématique « inconnue » de $\pm 2mm$ pour une portée de 10m et de 25m. Cette plage de normalité est donnée au niveau de confiance 1 $sigma$, i.e. à 68%. δ_i sera utilisé par la suite afin de connaître son influence sur les paramètres du logiciel.
- Le bruit de la mesure $\delta_{bruit} = \pm 0{,}6mm/10m$: selon Faro, le bruit de la mesure correspond à « l'écart standard des valeurs sur le meilleur plan d'ajustement ». Cette valeur ne pourra pas être étudiée car on n'en connait pas son procédé d'ajustement.

Autres facteurs propres à la scène relevée :

Le propos du logiciel est d'extraire des éléments géométriques plans issus de relevés par SLT de façades de bâtiments. Cette extraction se fait à travers l'étape dite de segmentation du nuage de points qui utilise l'algorithme RANSAC (pour l'extraction des grands plans du nuage) et le procédé par Croissance de surfaces (pour l'extraction des détails plans). Quels peuvent être les problèmes rencontrés pour segmenter des façades ? Nous allons voir ici une brève énumération de ce qui peut poser problème dans la segmentation d'une façade :

- Le mur principal du bâtiment ne suit pas parfaitement l'équation d'un plan : dans ce cas, on rencontrera quelques difficultés à appliquer RANSAC. Il faudra alors élargir suffisamment la distance seuil autour du plan afin d'avoir l'ensemble des points du nu principal de la façade. Il faut cependant être prudent à ne pas trop élargir ce seuil de recherche sous peine d'avoir des points qui n'ont plus rien avoir avec le nu du mur recherché (débuts de corniches, ouvrants etc.).

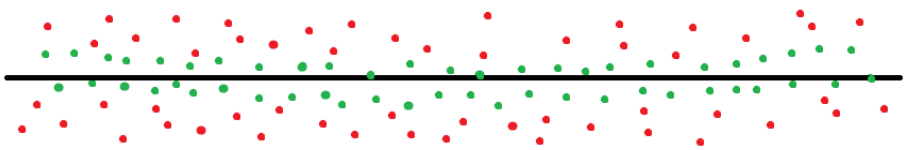


Figure 38 Schéma d'ajustement d'un plan par RANSAC – vue aérienne

Sur la figure ci-contre, les points verts semblent épousés une courbe légère en « U ». Le plan noir a tout de même pu être estimé et ajusté à partir des points verts, les *« inliers »* qui se situent à l'intérieur de la distance seuil autour du plan.

- La rugosité des éléments plans (revêtements etc.), couplée à l'incertitude δ_i accentue le phénomène de bruit de la mesure. Les critères de planéité locale et *a fortiori* les traitements par croissance de région sont très sensibles à ces facteurs.

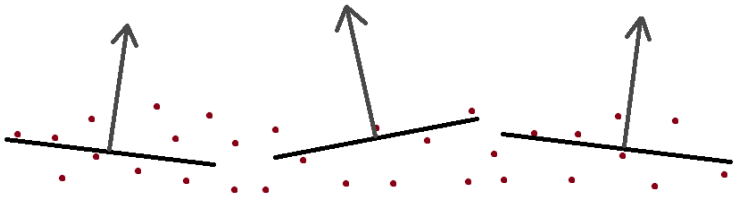


Figure 39 Schéma des plans locaux sur une surface bruitée

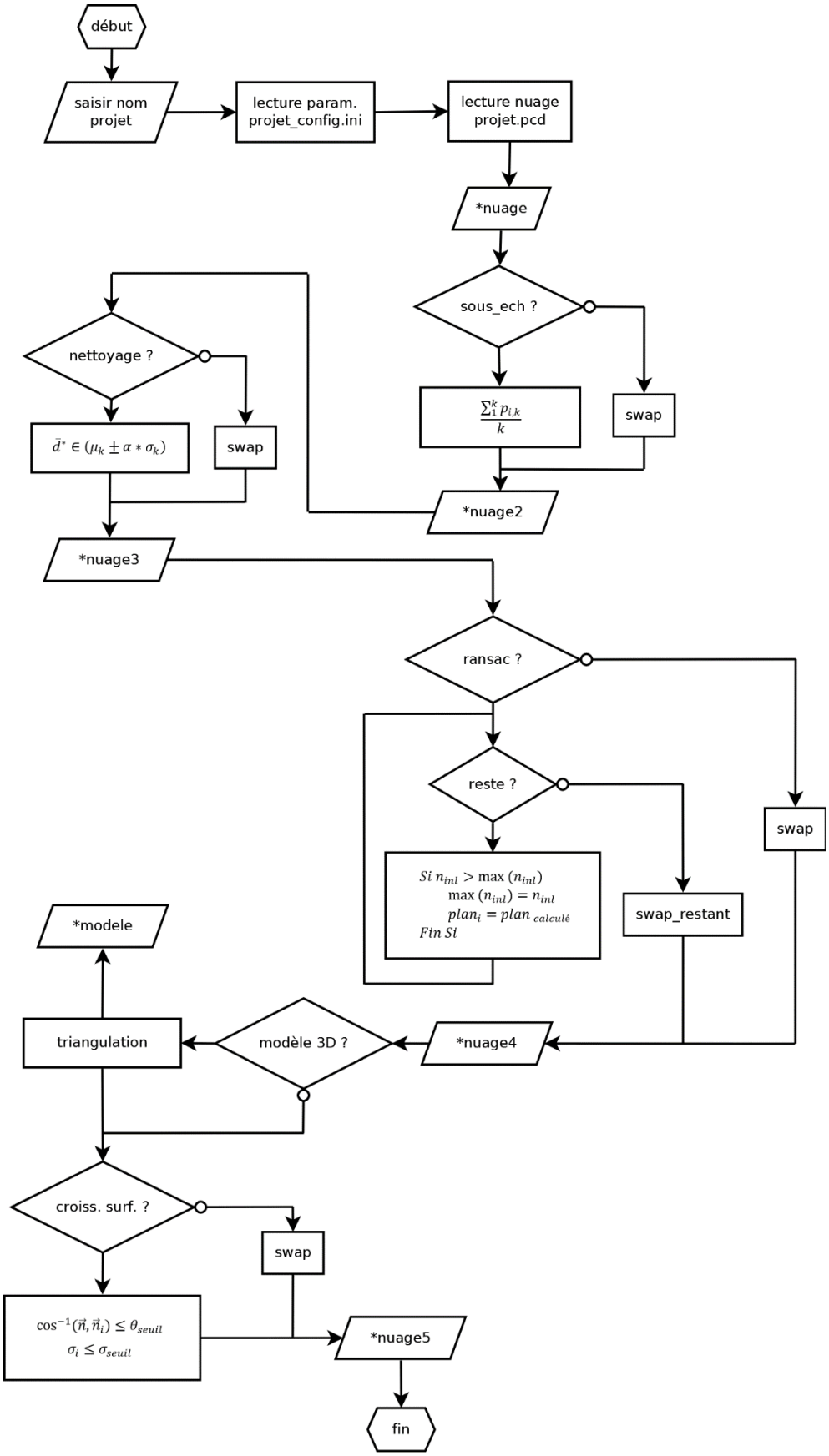
Ajustement des paramètres pour RANSAC :

- Largeur seuil l : elle dépend essentiellement du défaut de planéité global du nu du mur. $l = 2cm$ semble un bon compromis pour détecter le nu principal de la façade sans avoir d'éléments parasites. Les déformations excessives du mur ne seront pas pris en compte de par la robustesse de l'algorithme ;
- Nombre d'itération n : selon la formule de Fischler & Bolles, on considérera que le nuage dans lequel nous souhaitons détecter le plan est fortement bruité. On entendra par là qu'une certaine proportion du nuage ne décrit pas de plans. Le nombre maximal d'itération sera fixé à 600.

Ajustement des paramètres pour la croissance de régions :

- k-voisins : c'est le nombre de voisins pour l'estimation des plans locaux (i.e. des normales aux plans locaux). Il faut saisir un nombre suffisamment grand pour éviter d'avoir des plans locaux trop petits et donc trop sensibles à l'incertitude δ_i. Cependant, si le nombre de k-voisins est trop important, alors les normales tendront vers la même direction et il sera impossible de déceler des orientations différentes. Le nombre par défaut est fixé à 50.
- Orientation des normales locales : elle correspond à l'angle entre deux normales locales. Cette valeur dépend du pas de sous-échantillonnage du nuage et de l'incertitude de la mesure. Plus le nuage sera sous-échantillonné, plus les k-voisins représenterons une surface importante, moins la normale locale sera influencée par l'incertitude δ_i. *A contrario*, plus l'incertitude δ_i est grande et plus la direction des normales locales seront changeantes d'une surface locale à une autre, et plus l'angle devra être « relâché » pour que la surface puisse croître. Il semblerait à première vue qu'un angle compris entre 5 et 10° donne de bons résultats. Il faudra cependant à l'utilisateur plusieurs essais pour obtenir une segmentation correcte par croissance de surfaces.
- Seuil de courbure locale : cette valeur correspond à la tendance qu'ont les points à former une surface locale courbée. Cette valeur a été mise par défaut à 0.1. Plus cette valeur est petite, moins les surfaces courbées sont tolérées.

5 Algorigramme du procédé de traitement



6 Tutoriel du logiciel

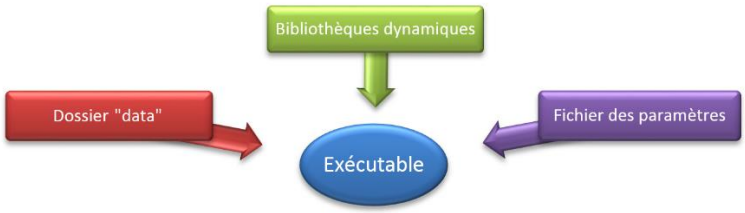
Les modules :

Le logiciel a été conçu afin de découvrir une bibliothèque C++ orientée dans le traitement de nuage de points, la Point Cloud Library (Rusu & Cousins, 2011), et de donner un exemple de ce qui peut être fait dans le traitement de nuages de points issus de relevés de façades, c'est-à-dire :

- Sous-échantillonner un nuage de points afin de l'alléger ;
- Nettoyer les points aberrants ;
- Segmenter le nuage afin d'en extraire les éléments plans (nu du mur, toit etc.) ;
- Modéliser les grands plans en trois dimensions.

Principe de fonctionnement :

Afin de revoir comment fonctionne le logiciel, reprenons la Figure 21 :



- Le dossier *« data »* contient l'ensemble des fichiers en entrée et en sortie de programme. Le logiciel ne peut lire en entrée que le format **.pcd* et produit en sortie des ensembles du nuage au format **.pcd*. Dans le cas d'une modélisation, le logiciel crée un fichier au format **.ply*.
- Le dossier *« config »* contient le fichier **_config.ini* où figure l'ensemble des paramètres utilisés par le programme. Ce fichier correspond à l'interface principale de saisie des paramètres par l'utilisateur.
- L'exécutable est le cœur du programme. Il demande à l'utilisateur de saisir le nom du projet. Ce nom de projet correspond à la racine * des différents fichiers précédents.
  Pour exemple, si le nom de projet est **batiment**, alors l'exécutable lira les paramètres dans le fichier \config**batiment**_config.ini et le nuage de points dans le fichier \data**batiment**.pcd. Le projet doit toujours être conforme aux noms de fichiers en entrée.

  La visualisation des données peut être faite en utilisant le logiciel CloudCompare (annexe 3 *« Obtenir un fichier au format PCD avec CloudCompare »*).

Le fichier ini :

Données permet de gérer les formats de fichier en sortie et le processus de traitement.

Nettoyage retire des points qui sont isolés du reste du nuage. Ce traitement repose sur un test statistique en calculant une distance moyenne autour des k-voisins du point.

Croissance de région détecte les éléments plans restants. Les régions grandissent par agrégation de points en testant les surfaces locales par différence angulaire entre les normales et par la valeur de la courbure locale.
Il est possible de visualiser le résultat à partir du paramètre *viewer*.

```
[donnees]
format=1
sous_echantillonnage=1
nettoyage_statistique=0
segmentation_ransac=0
modelisation_ransac=0
croissance_region=1

[sousech]
voxel=0.004

[nettoyage]
sigma=1.0
kppvnet=50

[sacseg]
inlierdist=0.008
pourcent=60.0

[mesh]
rayon=0.015

[region]
viewer=true
kppvnor=50
anglenor=10.0
courbure=0.05
```

Sous-échantillonnage permet d'alléger un nuage de points en le sous-échantillonnant avec un certain pas défini par une grille de voxels.

Segmentation RANSAC détecte les grands plans du nuage de manière itérative. Il faut spécifier une épaisseur du plan permettant d'accepter ou non des points autour d'un plan candidat.

Modélisation procède à une triangulation des points pour chaque plan issu de *Seg. RANSAC*. Il faut spécifier une distance limite pour rechercher les points formants un triangle.

Figure 40 Fichier de configuration complet

L'exécutable :

```
*************************************************
*                                               *
*      Extraction d'elements geometriques       *
*          dans un nuage de points 3D           *
*      Application aux releves de facades       *
*                                               *
*************************************************

>
nom du projet (identique au nom du nuage sans l'extension) : orleans

***** Chargement des donnees
> Lecture du fichier de configuration
> Initialisation des nuages
  Chargement data\orleans.pcd [fait, 7675 ms, 2762588 points]

***** Sous-echantillonnage du nuage de points
  Taille du voxel (m) : 0.05
> Calcul [fait, 1092 ms, 164520 points]
  Sauvegarde [data\orleans_echantillonne.pcd]

***** Nettoyage du nuage de points
> Donnees pour le nettoyage par test statistique
  Intervalle a 1 sigma
  k-ppv pour distance moyenne locale : 50
> Calcul [fait, 1372 ms : 149061 points]
> Sauvegarde [data\orleans_nettoye.pcd]

***** Segmentation RANSAC - detection des plus grands plans
> Donnees pour la segmentation RANSAC
  Largeur seuil autour du plan (m) : 0.03
  Pourcentage du nuage traite : 40
> Calcul
> Plan ajuste 1
  Segmentation : [0.942917 -0.33298 -0.00559851 -50.3403 coefficients]
  [Fait, 3385 ms, points : 58179]
  Sauvegarde [data\orleans_ransac_1.pcd]
  Points restant a traiter, i=0 : 90882
> Plan ajuste 2
  Segmentation : [-0.942989 0.332792 0.00464492 50.4069 coefficients]
  [Fait, 2979 ms, points : 35298]
  Sauvegarde [data\orleans_ransac_2.pcd]
  Points restant a traiter, i=1 : 55584
> Sauvegarde des points restants : [data\orleans_ransac_nontraite.pcd]


  Le traitement est termine

  Appuyez sur entree pour fermer le programme_
```

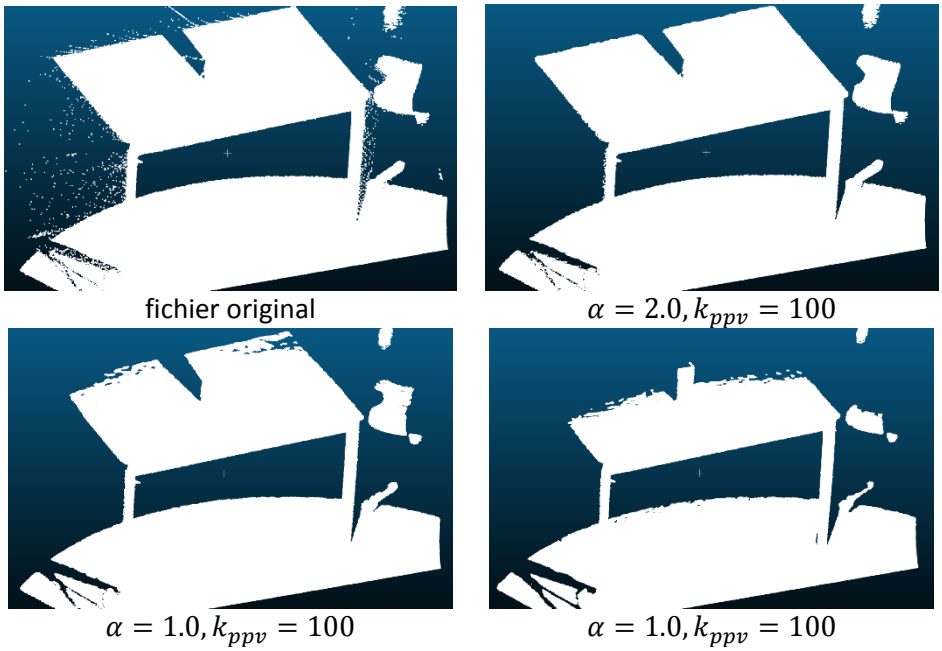
- Saisie du nom du projet correspondant à la racine des fichiers *.pcd et *_config.ini ;
- Temps de traitement pour chaque étape ;
- Rappel des paramètres ;
- Données mathématiques lors du traitement (coefficient d'équation des plans, nombre de points dans le nuage etc.) ;
- Sauvegarde des nuages pour chaque étape.

7 Résultats complémentaires

Nota : un tableau récapitulant tous les paramètres et les analyses complémentaires est disponible à la fin de cette annexe dans la partie *« synthèse des résultats »*. Par ailleurs, les résultats sont aussi fournis au format numérique dans le CD accompagnant ce mémoire (annexe 8).

Ajustement des paramètres sur une table

Nettoyage du nuage par test statistique :



fichier original | $\alpha = 2.0, k_{ppv} = 100$

$\alpha = 1.0, k_{ppv} = 100$ | $\alpha = 1.0, k_{ppv} = 100$

Figure 41 Essai sur une table - Annexe - Nettoyage du nuage

Façade 1 : Rue de Lévis

Modélisation du nu principal :

vue d'ensemble | volet modélisé | volet maillé

Figure 42 Rue de Lévis - Annexe - Modélisation

La modélisation 3D par triangulation rapide est globalement peu satisfaisante. Les raisons peuvent être les suivantes :

- une mauvaise interprétation des paramètres en entrée, notamment le paramètre interne du programme (dont l'utilisateur n'a pas accès) *k-ppv* ;
- la triangulation n'est pas adaptée pour modéliser des éléments plans. Cela se voit notamment sur Meshlab. Les faces des triangles sont aléatoires orientées (selon que la normale est positive ou négative). Il aurait peut-être fallu utiliser d'autres procédés plus appliqués au cas des objets plans (enveloppe concave par exemple) ;
- le nuage n'est pas suffisamment régulier pour que la triangulation puisse être efficace.

Façade 2 : Rue de la Tombe-Issoire

Résultat avec prétraitement léger :

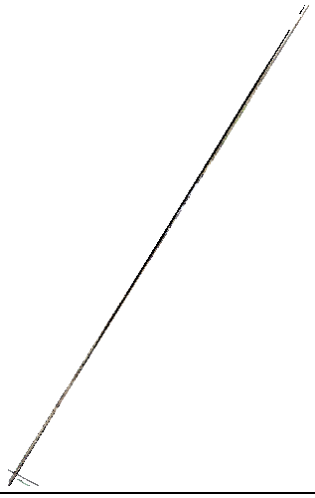
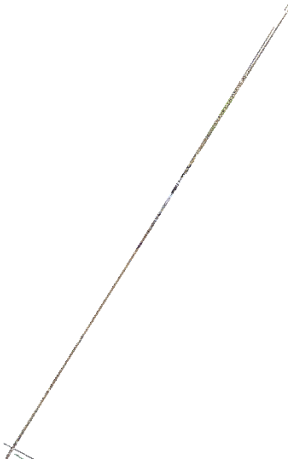
Prétraitement				(1) Ransac à 40% du nuage traité					(2) Croissance de surface, kppv = 50				
ss-éch. d (cm)	points nuage	t. stat. sigma	temps t0 (mn)	seuil l (cm)	plans	simi. p1 (°)	simi. cm/10m	temps t1 (h)	norm. α (°)	courb. c	clust.	% pts. Seg.	tps. t2 (h)
1	12 314 472	2,0	1,2	1,5	13	0,087	± 1,523	0h46	5	0,1	65	9,6%	10h17

Pour un pas de seulement $d = 1cm$ de sous-échantillonnage, le nuage est allégé de 70% et le traitement ne dure plus que 11h01. Pour les mêmes paramètres (1) et (2), le résultat est sensiblement différent selon que le nuage fut prétraité ou non. L'algorithme RANSAC a détecté 5 plans de plus. Le plan à la première itération est légèrement différent de celui rencontré dans la partie 3.3.2.1 (tâche noire sur la vue de face, Figure 43). Les plans semblent toutefois correctement segmentés (vue en plan et vue coupée).

Vue de face | Vue en plan | Vue coupée

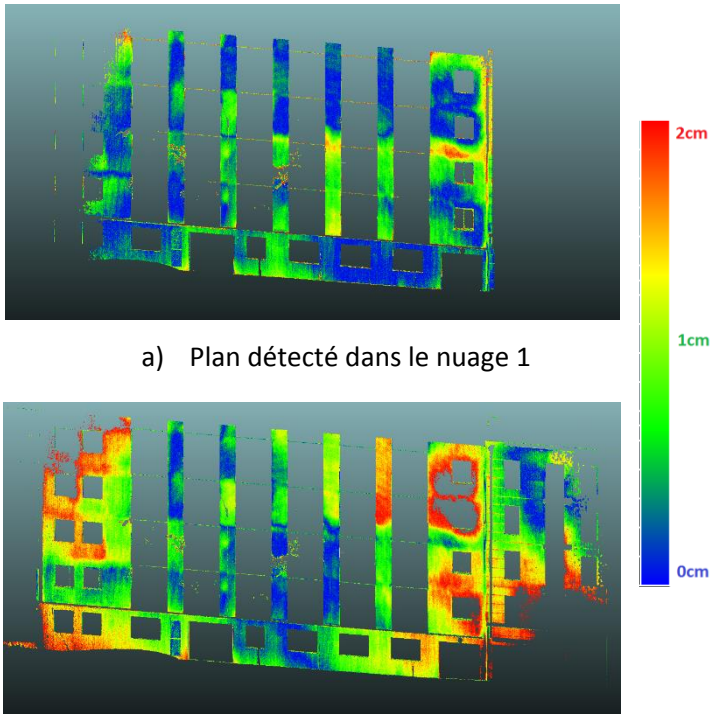
Figure 43 Rue de la Tombe-Issoire - Annexe - Prétraitement léger

Autres résultats avec prétraitement :

	Vue de face	Vue en plan	Vue coupée
Paramètre : Ransac : *l=5cm, tx*=60% Commentaire : Sur-segmentation Ransac : le taux de traitement *tx* du nuage est trop élevé (trop de plans détectés aux itérations Ransac). En noir : les plans détectés aux itérations suivantes. Des plans proches du premier plan sont détectés. Ils sont caractérisés par une légère déviation angulaire sur les vues en plan et coupée.			
Paramètre : Ransac : *l=3cm, tx*=40% Croissance de surface : *α=10°, c=1.0, k=50* Commentaire : La détection des plans est incorrecte (moins précise que pour *l=1.5cm*) et incomplète. En noir : les parties non détectés à l'itération 1. Les surfaces des détails plans sont sous-segmentées (surfaces vertes et turquoises).			
Paramètre : Ransac : *l=3cm, tx*=40% Croissance de surface : *α=7.5°, c=1.0, k=50* Commentaire : Les surfaces des détails plans sont un peu mieux segmentées. Quelques-unes d'entre elle demeurent sous-segmentées (turquoises).			

Façade 3 : Rue du Père Corentin

Détection du plan principal :



a) Plan détecté dans le nuage 1

b) Plan détecté dans le nuage 2

Figure 44 Rue du Père Corentin - Annexe - Comparaison RANSAC

Lecture : différences entre les points d'origines et le plan principal issu du nuage 1 (a, façade seule) et du nuage 2 (b, façade avec façades voisines). Utilisation de l'outil comparaison nuage/nuage de CloudCompare.

Commentaire : la promiscuité des plans des différentes façades peut engendrer des imprécisions quant à la détection du plan principal de la façade. Il n'est donc pas inutile de prétraiter manuellement le nuage en isolant la façade sur laquelle nous souhaitons travailler afin de détecter le plus exactement possible le nu principal de la façade.

Synthèse des différents essais sur les échantillons de façades

		Prétraitement				1 - Ransac à 40% du nuage traité					2 - Croissance de surface, kppv = 50						Résultat			
	nom du traitement	ss-éch. d (cm)	points nuage	t. stat. sigma	temps t0 (mn)	seuil l (cm)	plans	simi. p1 (°)	simi. cm/10m	temps t1 (mn)	norm. α (°)	courb. c	clust.	surfaces fausses	% pts. Seg.	tps. t2 (mn)	tps. T (h)	c.q. mod. 3D	RAM/ mo.file (ratio)	commentaires
Rue du Père Corent.	Corentin Faç. Ppale.	1,0	3,3E+06	1,0	0,6	2,0	1	± 0,067	**± 1,2**	2,2	-	-	-	-			**0,05**	mauv.	9,4	plan 1 sensiblement différent par influence des façades voisines
	Corentin Façs. α=14°	1,0	4,3E+06	1,0	0,7	2,0	2			4,7	14,0	0,100	53	90%	71%	4	**0,16**	-	13,7	surfaces sous-segmentées seuil α trop large
	Corentin Façs. α=5°	1,0	4,3E+06	1,0	0,7	2,0	2			4,7	5,0	0,100	109	10%	84%	19	**0,41**	-	12,2	segmentation correcte
Rue de Levis	Levis	-	2,1E+06	-	0,0	3,0	2	± 0,056	± 1,0	1,7	10,0	0,500	114	40%	100%	11	**0,21**	-	8,1	surfaces sous et sur-segmentées
	Levis l=2cm	-	2,1E+06	-	0,0	2,0	2			2,1	-	-	-	-		0	**0,03**	correct	7,5	plans ransac incomplets (défauts) seuil l trop petit
	Levis l=3cm	-	2,1E+06	-	0,0	**3,0**	2	± 0,056	**± 1,0**	1,7	-	-	-	-		0	**0,03**	correct	8,2	plan 1 sensiblement différent
Rue de la Tombe Essoire	Haus. 0 prétrait.	-	3,4E+07	-	0,2	1,5	8	± 0,087	± 1,5	73,5	5,0	0,100	42	-	5%	2 162	**37,27**	mauv.	2,7	temps conséquent sans prétrait. plans ransac incomplets échec de la seg. par agrégation
	Haus. (défaut)	1,0	1,2E+07	2,0	1,2	1,5	13			45,6	5,0	0,100	65	-	10%	617	**11,06**	mauv.	-	prétraitement insuffisant (temps) plans ransac ok mais sur-seg. échec de la seg. par agrégation
	H. sac 4cm@35%	1,5	6,8E+06	-	0,3	**4,0**	4	± 0,135	**± 2,4**	8,2	**7,5**	**1,000**	85	-	14%	113	**2,03**	-	-	prétraitement correct quelques imprecisions pour Ransac seg. par agrégation peu efficace
	5cm@60% surf. K=30	2,0	4,2E+06	2,0	0,5	5,0	11	± 0,169	± 2,9	10,5	6,0	1,000	96	-	12%	23	**0,56**	-	-	prétraitement efficace Ransac imprecis 5cm @ 60% échec de la seg. par agrégation
	Haus. 3cm@40% α=7,5°	1,5	6,8E+06	-	0,3	3,0	7	± 0,135	± 2,4	15,9	7,5	1,000	171	30%	36%	257	**4,55**	-	-	Ransac correct, 40% trop élevé seg. par agrég. Nuage légèrement sous-segmenté
	H. c. surf. α=10°	1,5	6,8E+06	-	0,7	3,0	7	± 0,135	± 2,4	15,9	**10,0**	1,000	156	80%	54%	198	**3,58**	-	-	surf. par agrégation largement sous-segmentée
	Définitif @25% K=60	2,0	4,2E+06	-	0,3	**5,0**	2	± 0,178	**± 3,1**	2,7	7,0	1,500	185	10%	45%	51	**0,90**	-	-	seuil l large pour max. de candidats seg. agrégation correcte

8 Annexes numériques

Des compléments d'informations techniques sont disponibles dans le CD-ROM fourni avec ce mémoire (consultation à l'ESGT) :

- Code source du programme ;
- Installeur du programme ;
- Résultats des différents traitements fournis au format BIN de CloudCompare.

Nota : code source disponible en pièce jointe du PDF

Extraction d'éléments géométriques dans un nuage de points LiDAR terrestre Application aux relevés de façades

Mounir AÏT-MANSOUR, Septembre 2014

SCOP GeoSystem Surveying
3, rue Marie-Davy, Paris

Ecole Supérieure des Géomètres et Topographes
1, boulevard Pythagore, le Mans

Introduction

Le LiDAR terrestre, autrement appelé scanner laser terrestre (SLT) permet de numériser un environnement en trois dimensions sous forme d'un ensemble de points appelé nuage de points. Depuis quelques années, le SLT connaît un essor très important dans les bureaux d'études topographique. Sa précision millimétrique et sa rapidité d'acquisition en font un outil désormais incontournable.

Toutefois, cette plus-value est ternie par un post-traitement des données assez lourd. Il n'est pas rare d'avoir plusieurs dizaines de millions de points dans un nuage à traiter. Il apparait clairement qu'il faille trouver dans le futur des procédés pour automatiser une partie du traitement des nuages de points.

Cette étude proposera d'automatiser une partie de l'extraction d'information issue de relevés par lasergrammétrie de façades d'immeubles. Le travail s'appuiera sur une étude bibliographique approfondie des différents processus et méthodes existants dans le traitement des nuages de points. Il utilisera notamment les algorithmes partagés par la communauté libre C++ Point Cloud Library (Rusu & Cousins, 2011) pour développer un applicatif. Ce dernier sera testé à partir d'un échantillon de relevés de façades pour valider l'étude.

Figure 1 Logo de la Point Cloud Library

Prétraitement

Sous-échantillonnage

Ce prétraitement permet de dé-densifier le nuage de point. Pour cela, l'espace est découpé par une grille de voxels v_i de dimension n^3(n étant la longueur d'un côté du voxel) (Chin-Feng & Don-Lin, 2001). Chaque v_i contient un nombre k_i de points. Ces points $p_{i,k}$ sont moyennés à l'intérieur de chaque voxel afin d'obtenir un unique point p'_i.

$$p'_i = \frac{\sum_1^k p_{i,k}}{k}$$

Équation 1 Estimation du centroïde i de k points

Suppression des points clairsemés

Le second prétraitement effectué par le logiciel est de supprimer du nuage les points clairsemés. Ce nettoyage du nuage repose sur un test statistique par comparaison de la distance $\bar{d}$ aux k plus proches voisins d'un point p_q à l'ensemble des distances moyennes locales du nuage supposé suivre une distribution gaussienne (μ_k, σ_k). Ainsi, tous les points dont la distance moyenne est en dehors d'un certain intervalle peuvent être considérés comme des valeurs aberrantes.

$$\{ p_q^* \in P \mid (\mu_k - \alpha * \sigma_k) \leq \bar{d}^* \leq (\mu_k + \alpha * \sigma_k) \}$$

Équation 2 Nettoyage du nuage par test statistique (Rusu & Al., 2008)

Le prétraitement des données est une étape à ne pas négliger. En amont de la chaîne de production, elle peut avoir des conséquences directes sur les traitements *a posteriori*.

Segmentation

Selon (T. Landes P. G., 2011b) : « la segmentation d'un nuage de points est une subdivision de l'ensemble des points 3D en sous-ensembles homogènes, suivant des critères prédéfinis ». Il existe deux grandes familles de segmentation (Vosselman, 2004) que sont la segmentation par reconnaissance de formes géométriques (ex. RANSAC) et la segmentation par agrégation de points (ex. Croissance de surface).

RANSAC

L'algorithme RANSAC (pour RANdom SAmple Consensus) (Fischler & Bolles, 1981) est utilisé dans la segmentation pour extraire des primitives géométriques et notamment des plans dans le cas d'un relevé de façade.

Le principe est le suivant : à chaque itération, trois points sont sélectionnés aléatoirement dans le nuage afin de déterminer un plan $\mathcal{F}$. Ce plan calculé est épaissi d'une certaine largeur l afin d'accepter les points voisins appelés *inliers*. Le nombre d'*inliers* In trouvé est stocké et testé : si il est supérieur au nombre d'*inliers* maximum trouvé aux itérations précédentes, alors le nombre d'*inliers* devient une valeur maximale et le plan calculé devient un plan déterminé. Ce processus est répété i fois afin d'identifier le plan ayant le plus d'*inliers* possible, autrement dit le plan le plus grand dans le nuage $\mathcal{M}$.

Ainsi, pour détecter n plans $\mathcal{F}_n$ sur une façade, ce processus est réitéré n fois. En utilisant la bibliothèque PCL, nous pouvons faire :

$\mathcal{F}_n = \emptyset$
Tant que l'on identifie des plans
  $pcl::SACSegmentation\ (\mathcal{M}, l)$
  $\mathcal{F}_n = \mathcal{F}_n + \mathcal{F}$
  $\mathcal{M} = \mathcal{M} - In$
Fin Tant que

Algorithme 1 Détection de n plans avec RANSAC

RANSAC a l'avantage de détecter rapidement et de manière robuste les plans majeurs de la façade. Il n'est cependant pas optimisé pour extraire les éléments de détails. Il faut donc combiner RANSAC avec un traitement par agrégation de points afin d'obtenir une segmentation optimale.

Croissance de surface

L'agrégation de points par croissance de surface est basée sur des critères locaux de normalités et de courbures.

L'algorithme -repris dans la PCL- est issu des travaux de Rabbani (Rabbani & Al., 2006). Des régions grandissent autour de points-échantillons appelés graines tant que les points candidats décrivent une même surface (autrement dit tant que la courbure et l'orientation des normales locales varient en dessous d'un certain seuil). Si ce seuil est dépassé, la surface cesse de grandir et est extraite du nuage. Normalité θ et courbure σ sont définis par :

$$\cos^{-1}(\vec{n}, \vec{n}_i) \leq \theta_{seuil}$$

Équation 3 Seuil angulaire entre deux normales

$$\sigma = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}$$

Équation 4 Courbure locale σ

La courbure locale σ est directement approximée à partir des valeurs propres issues de la matrice de covariance de l'estimation du plan tangent local. Contrairement à RANSAC, la croissance de surface est assez sensible au bruit et dépend fortement des critères locaux. Elle est donc plus adaptée à la segmentation des détails de la façade.

Plus généralement, la segmentation est une pré-étape clé dans la reconstruction de l'environnement 3D car elle identifie chaque sous-ensemble comme entité indépendante.

Modélisation

La modélisation est la dernière étape du traitement des données. Elle permet de construire les objets de la scène en trois dimensions. Deux méthodes combinées sont utilisées : la modélisation par maillage et la modélisation géométrique.

Reconstruction géométrique

Cette méthode consiste à simplifier l'objet à sa plus pure représentation mathématique. Pour cela, nous procédons à une projection des points sur le plan détecté. La première étape est de récupérer le vecteur normal unitaire :

$$\overrightarrow{n_u} = \left[\frac{a}{\|\vec{n}\|} \quad \frac{b}{\|\vec{n}\|} \quad \frac{c}{\|\vec{n}\|}\right]^T$$

Équation 5 Vecteur normal unitaire au plan

Le vecteur $\overrightarrow{n_u}$ est utilisé pour calculer la distance de projection $AH = \overrightarrow{n_u} * p_i$ pour chaque point p_i. Les nouvelles coordonnées des points projetés sont déterminées en retirant la longueur AH sur chaque composante : $p_{projeté} = p_i - \overrightarrow{n_u} * AH_i$.

La projection des points permet d'obtenir un plan physiquement délimité.

Maillage des points

Cette méthode de reconstruction crée un maillage de triangles désordonnés pour relier les points entre eux. Contrairement à une triangulation de Delaunay, cette méthode est beaucoup moins gourmande en temps de calculs.

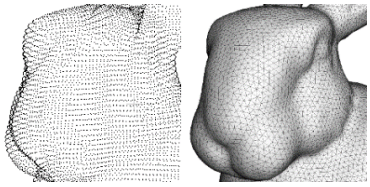


Figure 2 Lapin de Stanford (Creatis, 2014)

Echantillon-test

Plan majeur avec RANSAC

La segmentation des plans majeurs par RANSAC est efficace dans la plupart des cas. Elle permet de détecter le nu principal de la façade et le pan de toit si ce dernier est présent dans le nuage de points.

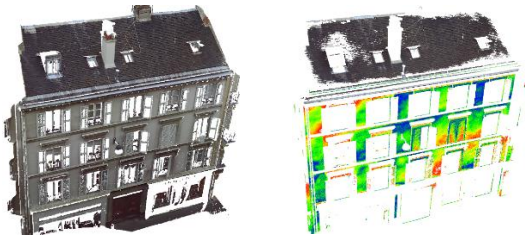


Figure 3 Rue de Lévis - Plan principal détecté

Une comparaison nuage/nuage avec CloudCompare permet de visualiser les différences entre le nuage de départ et les points projetés sur le plan détecté (Fig. 3).

Modélisation du plan majeur

La modélisation 3D par triangulation rapide est globalement peu satisfaisante car peu adaptée pour modéliser des éléments plans : le fichier est peu optimisé (poids lourd) et des données manquantes apparaissent.

Figure 4 Modélisation d'un volet

Extraction des détails

La détection des détails est effectuée par croissance de surface. Il est nécessaire d'ajuster les paramètres afin d'éviter une sous ou sur segmentation. Le résultat est fourni sous forme d'un nuage coloré.

Figure 5 Rue de la Tombe-Issoire - Croissance de surface

Conclusion

Résultat

La segmentation des plans principaux par RANSAC est efficace tandis que la segmentation des détails par croissance de surface est assez sensible au bruit et demande un temps important pour ajuster les paramètres. La modélisation par triangulation rapide est globalement peu satisfaisante car peu adaptée pour modéliser des éléments plans.

Perspective

Plus généralement, le traitement de données LiDAR terrestre représente un réel défi pour l'avenir de la profession. Le choix d'utiliser un environnement de développement libre dont les normes sont partagées par le plus grand nombre permet de pérenniser ce travail de fin d'étude afin d'être un point de départ modeste de ce qui pourrait être développé en utilisant une bibliothèque spécialisée dans le traitement de nuages de points 3D.

Extraction d'éléments géométriques dans un nuage de points LiDAR terrestre. Application aux relevés de façades.

Mémoire d'Ingénieur ESGT, Le Mans, 2014

Résumé

Ces dernières années, les scanners laser acquièrent des données 3D de plus en plus efficacement. Cependant, le traitement des données demeure relativement lourd rendant son automatisation nécessaire afin de rééquilibrer temps d'acquisition et temps de traitement. Ces scanners laser, dont le LiDAR terrestre, permettent de numériser l'environnement sous forme de nuage de points. Ils sont utilisés dans différents secteurs tels que l'industrie, le bâtiment ou encore la robotique.

Il est alors souhaitable que des normes et des méthodes de traitements automatisés des données soient partagées par le plus grand nombre. C'est exactement le rôle que joue la bibliothèque logicielle C++ Point Cloud Library (PCL).

Cette étude propose d'automatiser une partie de la chaîne de traitement de relevés de façades d'immeubles à travers le développement d'un applicatif. Pour cela, le travail s'est appuyé sur la PCL et sur une recherche bibliographique approfondie des différents processus et méthodes existants dans le traitement des nuages de points.

Mots clés : scanner laser terrestre, lasergrammétrie, nuage de points, segmentation, extraction de formes, reconstruction, point cloud library, façade d'immeuble

Abstract

Recently, laser scanners get data more and more efficiently. However, data processing is still heavy and it is necessary to automate it to balance acquisition's time and processing's time. These laser scanners (of which the terrestrial LiDAR) allow to scan a scene as a point cloud. They are used in various sectors such as industry, building or robotic.

Thus, it is desirable that standards and methods of data processing can be shared by everyone. This is exactly the role of C ++ Point Cloud Library (PCL).

This study proposes to automate some of the processing line of building facades surveys by developing an application. The work was based on the PCL library and on various existing processes and methods in the point cloud processing stemming from the thorough literature review.

Key words: terrestrial laser scanning, LiDAR, point cloud, segmentation, feature extraction, reconstruction, point cloud library, building façade