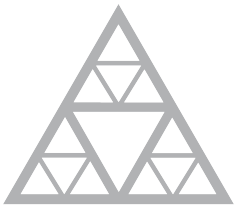

École des Ponts
ParisTech

Reconstruction de bâtiments en 3D à partir de nuages de points LIDAR

Mickaël Bergem

25 juin 2014

Maillages et applications

Table des matières

Introduction

Dans le cadre du cours de Maillages et Applications de l'École des Ponts ParisTech, j'ai choisi le projet "modélisation de bâtiments en 3D sous forme d'un maillage dense à partir d'un nuage de points Lidar". J'ai du développer un algorithme qui, à partir d'un nuage de points Lidar, produit un maillage dense de l'environnement urbain duquel le nuage est issu.

Contrairement aux autres groupes, j'ai travaillé seul (nous étions un nombre impair) sur ce projet orienté recherche, et j'ai bénéficié des conseils de Florent Lafarge pour me guider sur la méthode à adopter.

J'ai utilisé les fonctions de traitements des maillages de la librairie CGAL, et développé un algorithme en C++.

1 La modélisation numérique de milieux urbains

La modélisation numérique de milieux urbains correspond à la génération d'une représentation virtuelle d'un bâtiment ou d'une ville entière, représentation qui peut intégrer de nombreux aspects comme la géométrie (forme des bâtiments), la sémantique (différencier un arbre d'un trottoir et d'un toit), l'utilisation de textures pour améliorer le réalisme visuel du modèle.

FIGURE 1 – Exemple de LIDAR aéroporté

Ce domaine a été appréhendé d'un grand nombre d'approches, d'une part par les différents aspects évoqués ci-dessus, eux-mêmes déclinés sur un certains nombre de critères : quelle précision pour la reconstruction géométrique (à la rue près, au banc près, au buisson près ?), quelle distinction sémantique (combien de classes différentes ?), quelles hypothèses sont faites (les murs sont forcément verticaux, un arbre ne peut pas dépasser 10m de hauteur) ? D'autre part, les méthodes peuvent être automatisées ou comporter une partie de travail manuel, par exemple dans le choix de paramètres, ou dans la détection des erreurs. Certains travaux, comme ceux de Poullis & You [1], montrent la possibilité de reconstruire des villes entières à partir de nuages de points acquis par LIDAR aéroporté (Figure 2).

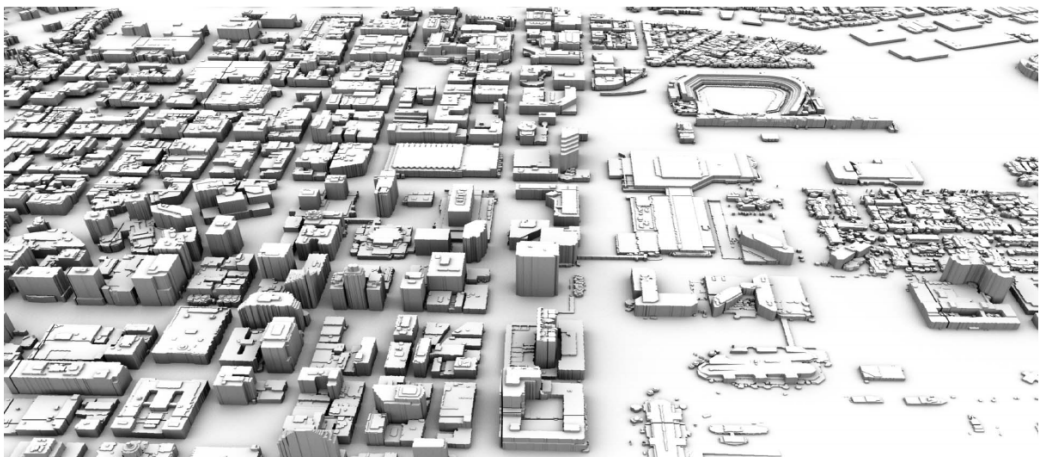

FIGURE 2 – Reconstruction 3D de la ville de Baltimore

2 L'algorithme de reconstruction

2.1 La méthode

Pour rappel, l'objectif de l'algorithme est de produire un maillage dense à partir d'un nuage de points LIDAR, fourni au format PLY.

Le fonctionnement de l'algorithme peut se décomposer en quatre phases distinctes :

1. Le chargement du nuage de point LIDAR en mémoire
2. L'échantillonage dans le plan (X,Y) des points
3. La conversion en matrice d'élévation
4. La simplification du maillage

2.2 Acquisition du nuage de points

Le nuage de points est fourni dans un fichier PLY au format ASCII (il existe une version binaire, plus légère mais plus difficilement lisible), qui contient une série de lignes d'en-têtes, puis le nuage à raison d'un point par ligne, dont les trois coordonnes sont séparées par un espace.

Ne trouvant pas de module existant pour charger ce nuage de point en mémoire (voire directement dans CGAL), j'ai choisi d'écrire mon propre module de lecture de fichier PLY. Celui-ci renvoie donc pour un chemin vers un fichier PLY, un `Vector <Point> nuage` ainsi que d'autres informations relatives au nuage (taille, nombre de points).

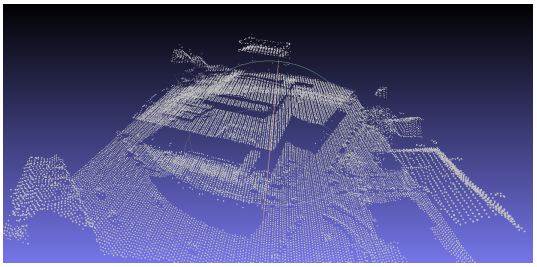

FIGURE 3 – Le nuage de points ouvert avec Meshlab

2.3 Échantillonnage

À partir du nuage de point fourni précédemment, nous allons chercher à calculer la matrice d'élévation, qui en chaque couple de coordonnées (X,Y) associe l'altitude mesurée. Cette matrice serait échantillonnée sur une grille

discrète de taille N*N. Les coordonnées seront donc à valeur dans ⟦0,N-1⟧. Le nuage initial étant à coordonnées réelles, il faut dans un premier temps aligner tous les points sur la grille (sur les noeuds de la grille, cf figure 2.3)

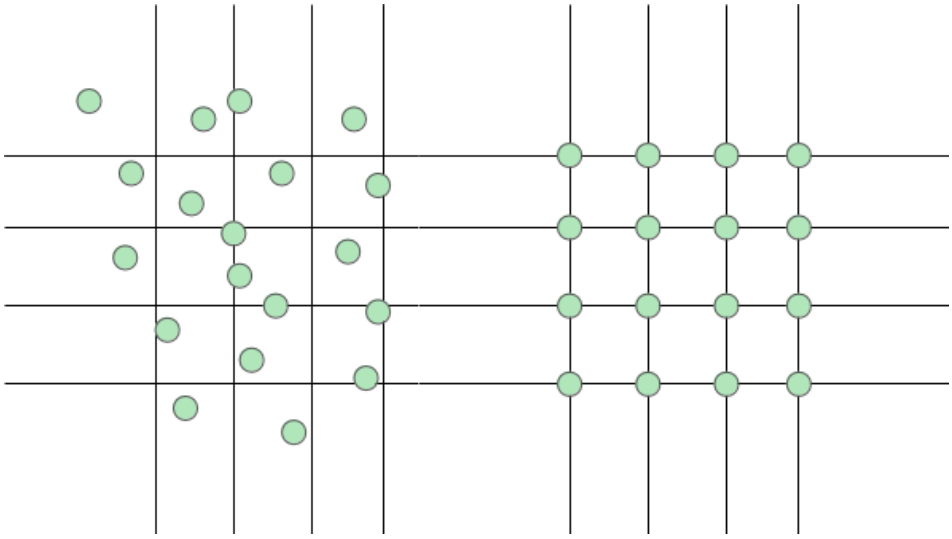

FIGURE 4 – Alignement des points dans le plan (X,Y), sur la grille

Chaque point est donc translaté dans le plan (X,Y) vers le noeud le plus proche, tout en conservant sa coordonnée Z.

On se retrouve à la fin de l'échantillonage avec un nuage de point (toujours un `Vector <Point> nuage` non structuré) mais dont les points sont à coordonnées X et Y discrètes.

2.4 Matrice d'élévation

On va maintenant parcourir chaque point du nuage, et l'ajouter sur une matrice grille, qui est donc de type `Vector <Point> nuage` : nous obtenons un ensemble structuré qui à chaque noeud de la matrice associe le(s) point(s) associé(s).

Nous parcourons alors cette première matrice grille pour générer la matrice d'élévation (qui à un noeud associe une unique élévation). L'algorithme

suivant est utilisé :

```
Data: Matrice grille
Result: Matrice d'élévation
foreach noeud in matrice_grille do
    if nombre de points présents sur ce noeud > 0 then
        matrice_elevation[noeud]=moyenne_z(matrice_grille[noeud]);
    else
        if nombre de points présents sur les 4 noeuds voisins > 0 then
            matrice_elevation[noeud]=moyenne_z(matrice_grille[noeud]);
        else
            matrice_elevation[noeud]=altitude_minimale;
        end
    end
end
```

Algorithm 1: Calcul de la matrice d'élévation à partir de la grille

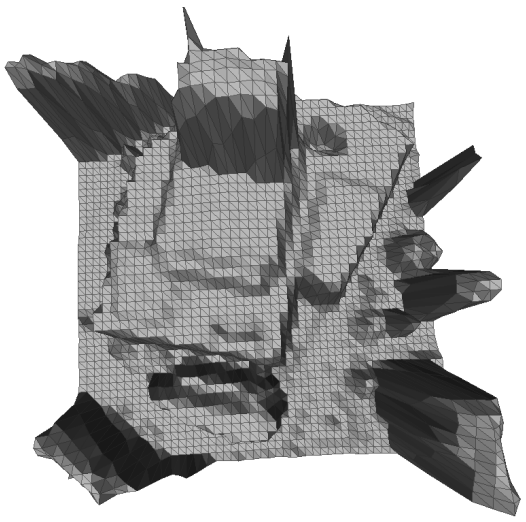

FIGURE 5 – La matrice d'élévation

Nous obtenons donc une carte d'élévation (discrète), ce qui est cohérent avec la notion de "vue du ciel" où tout le plan (X,Y) est visible et associé à une seule coordonnée Z. La figure 5 montre un exemple de carte d'élévation.

2.5 Conversion en Polyhedron_3D

Puisque nous allons par la suite utiliser des fonctions de CGAL, il faut convertir cette carte d'élévation en `Polyhedron_3D`, qui est une structure

CGAL pour décrire des polyèdres tridimensionnels (comme son nom l'indique).

Pour réussir cette conversion, j'ai du utiliser un *incremental builder*, et l'adapter (non sans mal) à mes besoins pour qu'il puisse lire la structure de données de la matrice d'élévation.

Le code est relativement complexe lorsque l'on n'a pas l'habitude d'utiliser CGAL, mais l'explication du code en lui-même est hors-sujet dans ce rapport.

2.6 Simplification du maillage

C'est avec la fonction `edge_collapse` de CGAL que j'ai pu réduire le nombre de triangles dans le maillage, en préservant la forme générale du maillage.

Pour décrire rapidement le fonctionnement de cette fonction, elle remplace un edge par un vertex, en supprimant et recréant les edges adéquats, et diminuant ainsi le nombre d'arêtes et de triangles.

Cette fonction prend en paramètre un prédicat d'arrêt (qui décidera lorsqu'il faudra arrêter la simplification) et il faut donc décider d'un nombre d'arêtes minimum à conserver.

3 Implémentation et résultats

3.1 Gestion du projet

Pour travailler sur ce projet, j'ai choisi de créer un dépôt sur le gestionnaire de code source GitHub, disponible à l'adresse https://github.com/MickaelBergem/LidarMaillage, publiquement.

La gestion des tickets m'a permis d'ordonner et de prioriser les différentes tâches, et le versionnement du code m'a permis de retrouver beaucoup plus rapidement les régressions.

3.2 Fonctionnement

L'algorithme se présente sous la forme d'un petit exécutable en ligne de commande, programmé en C++, qui prend en paramètres :

— le chemin vers le fichier PLY à lire
— la taille de la grille (en nombre de noeuds sur un côté)
— le nombre limite d'arêtes à conserver

Il génère deux fichiers `pre-export.off` (avant simplification) et `final-export.off` (après simplification) qui peuvent être visualisés par Meshlab.

Le lecteur est invité à lire le code source du programme (fonction `main` notamment) pour obtenir le détail de l'implémentation, qui est relativement compréhensible grâce aux commentaires.

3.3 Résultats

En faisant varier le nombre limite d'arrêtes à laisser sur le maillage (et donc la simplification du maillage), on obtient le résultat suivant :

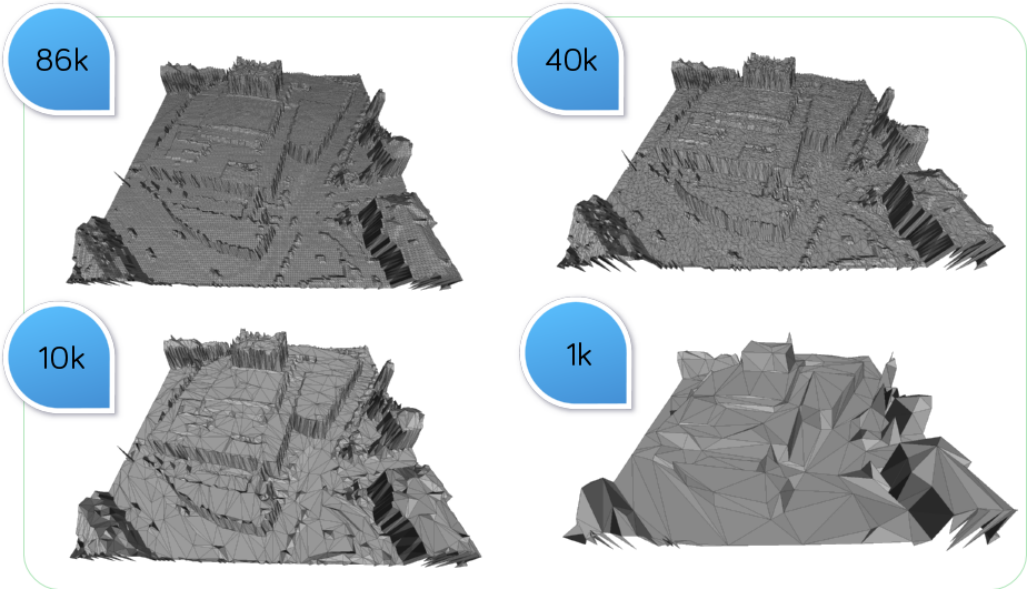

FIGURE 6 – Différents seuils de réduction

La question posée par la figure 6 est la suivante : comment déterminer le nombre d'arêtes limite ? Dans notre exemple, on peut s'apercevoir (de manière empirique et visuelle) que la limite se situe entre 10000 et 1000 arrêtes, pour un modèle original à 86000 arêtes.

Il faudrait alors introduire, avec plus de temps, un indicateur basé sur le rapport nombre de triangles / erreur de modélisation.

On peut également obtenir des résultats similaires avec d'autres nuages de points, comme le montrent les figures 7 8 et 9.

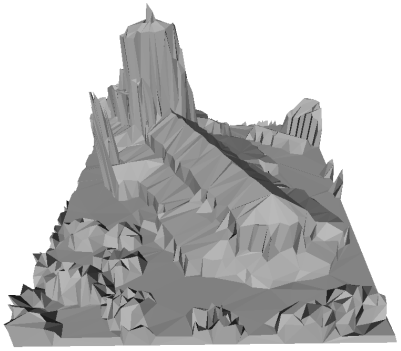

FIGURE 7 – Modèle B5 avec 4000 arêtes

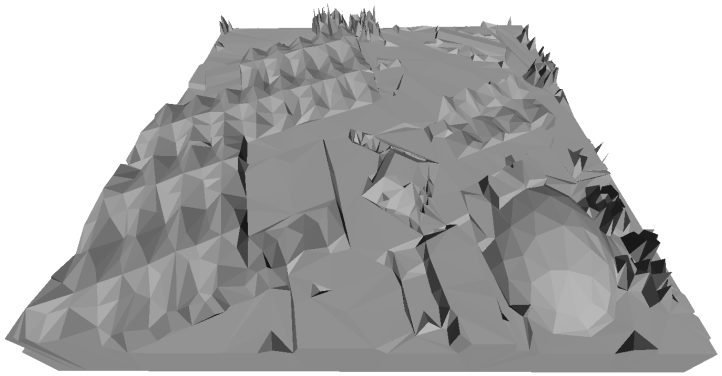

FIGURE 8 – Modèle B8 avec 4000 arêtes

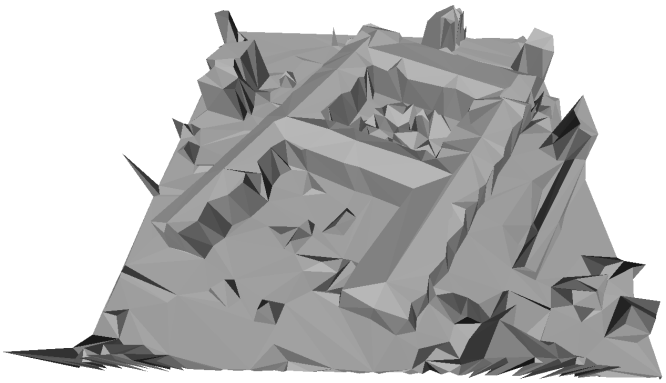

FIGURE 9 – Modèle B9 avec 2000 arêtes

3.4 Performances

Nous pouvons également constater sans surprise que :

— diminuer le nombre limite d'arêtes demande plus de temps (il faut passer plus de temps à simplifier le maillage)
— utiliser une grille deux fois plus dense demande beaucoup plus de temps (il y a quatre fois plus de noeud, donc six fois plus d'arêtes)
— le nombre initial de points dans le maillage n'est pas très significatif par rapport aux autres facteurs

Points dans le nuage	Taille de la grille	Nombre cible d'arêtes	Temps
86021	170*170	10000	16s
86021	170*170	40000	7.9s
58499	120*120	10000	6.4s
58499	240*240	10000	38s
22300	149*149	10000	13s

Conclusion et perspectives

Pour finir, j'ai donc réalisé un programme simple et rapide, qui transforme des nuages non structurés stockés dans des fichiers PLY en des maillages denses au format OFF, en permettant de jouer sur la qualité du maillage et sa densité.

La taille de la grille est à définir manuellement, bien qu'une alerte s'affiche en cas de sur-échantillonnage, de même que le nombre d'arêtes limite.

Le programme étant simple et sa taille étant limitée par le fait que j'étais seul sur ce projet, on pourrait imaginer un grand nombre d'améliorations :

- "nettoyage" automatique des outliers, par exemple en supprimant les points isolés (ceux dont les k plus proches voisins n'appartiennent pas à une sphère de taille fixée), directement dans le nuage initial
- lisser les surfaces après maillage, pour éviter les effets escalier sur des façades censées être lisses
- ajouter une fonction de classification des objets : détection des plans, classification basée sur l'altitude, groupements, angle entre les plans... ou encore une classification sémantique basée sur une grammaire de construction
- l'ajout de textures à partir de photos aériennes ou au sol
- la superposition de plusieurs nuages pour permettre la reconstruction de villes entières

Ce projet a été pour moi l'occasion de découvrir les problématiques et les applications des méthodes de maillages de surfaces, et de prendre en main la librairie CGAL.

Références

[1] Charalambos Poullis, and Suya You, *Automatic reconstruction of cities from remote sensor data.* CVPR, 2009.