Modélisation de toitures à partir de nuages de points 3D

Dobrina Boltcheva, Chouaib Fellah, Damien Liehn, Clément Poull

Modélisation de toitures à partir de nuages de points 3D

Dobrina Boltcheva [1] [2], Chouaib Fellah [3], Damien Liehn [2], Clément Poull [2]

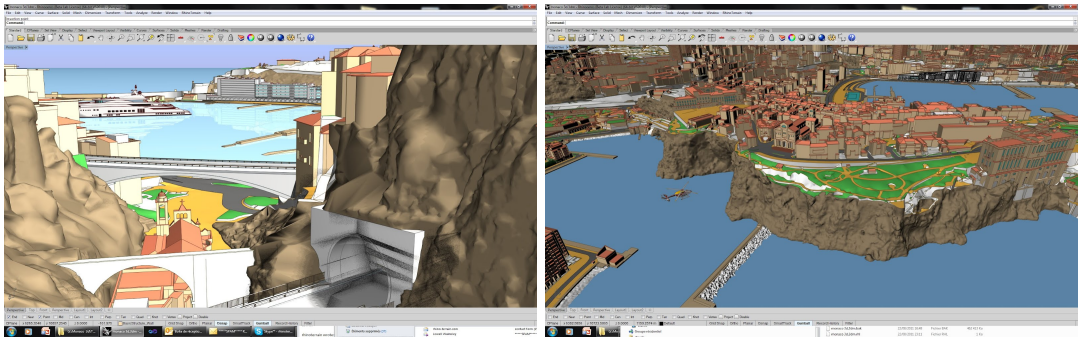
[1] LORIA-INRIA Grand Est
[2] Université de Lorraine
[3] ICUBE, Université de Strasbourg

Sommaire

1 Introduction

La numérisation d'objets réels est de plus en plus utilisée dans des domaines tels que l'urbanisme, l'architecture ou l'aménagement d'espaces publics. Les outils d'acquisition tels que les LiDAR (acronyme anglais de Light Detection And Ranging) ou la photogrammétrie permettent de produire des représentations numériques de villes entières sous forme de nuages de points 3D échantillonnant les surfaces des objets de l'environnement.

Aujourd'hui, le processus de création d'une maquette numérique urbaine à partir de tels relevés est long, fastidieux et essentiellement manuel. Dans ce processus de rétro-conception, l'opérateur humain trace *à la main* les éléments constitutifs de la maquette de sorte à coller au mieux au nuage de points 3D. Par exemple, la figure ci-dessous représente une maquette numérique de la ville de Monaco, générée avec le logiciel RhinoCity à partir de données LIDAR et de photogrammétrie. Des outils d'assistance semi-automatiques émergent sur le marché, mais ils ne répondent pas suffisamment au besoin, tant en termes de précision que d'efficacité.



Notre équipe de recherche travaille en collaboration avec la société RhinoTerrain qui est spécialisée dans la géomatique et l'imagerie 3D. Elle développe une nouvelle génération d'outils 3D de "Géo-Modélisation" dédiés à la production structurée de données géoréférencées et standardisées. Un des axes R&D de la société porte sur l'optimisation du processus de réalisation d'une maquette 3D à partir des relevés pouvant provenir de sources différentes (points 3D lidar, images satellites, photos, etc.). L'objectif est de réduire les coûts liés à la constitution des maquettes, d'accroître sa compétitivité et d'accélérer l'adhésion des industriels à la méthodologie.

Les outils d'acquisition tels que les LiDAR (détection et télémétrie par ondes lumineuses) permettent de produire des représentations numériques de villes entières sous forme de nuages de points 3D échantillonnant les surfaces des objets de l'environnement. C'est une technique de mesure à distance fondée sur l'analyse des propriétés d'un faisceau de lumière renvoyé vers son émetteur et qui produit des mesures sous forme de points 3D x, y, z d'une grande precision (voir Figure 1).

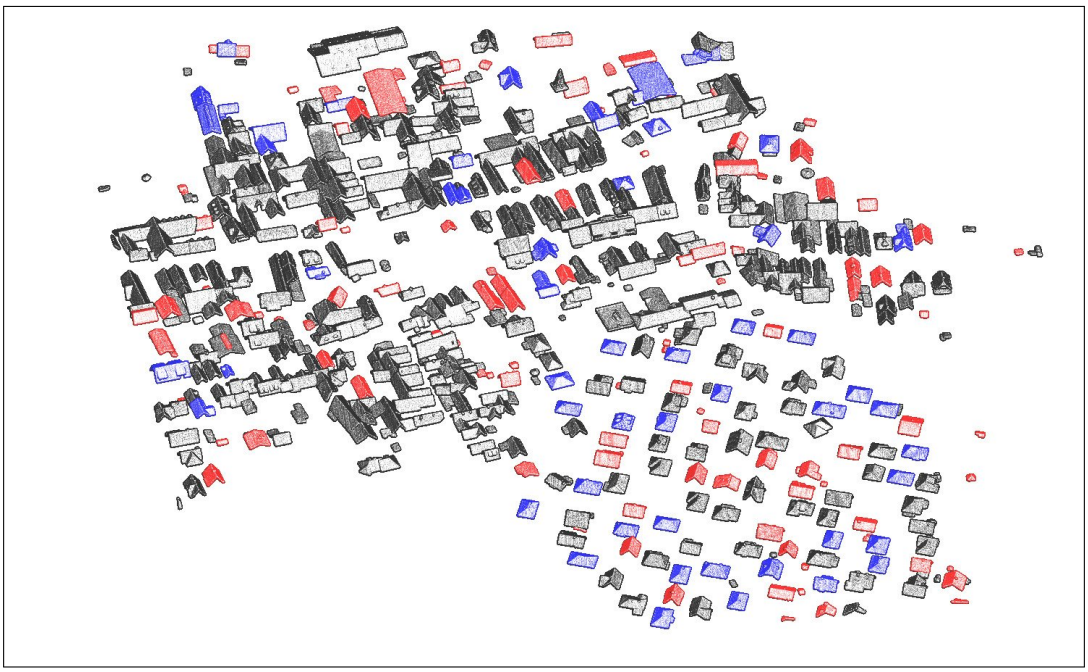


FIGURE 1 – Nuage de points 3D de 445 toitures d'un lotissement. Il y a 128 toits à deux pans (en rouge), 52 à quatre pans (en bleu) et des toitures complexes (en noir).

Le pipeline de traitement présenté dans ce rapport de recherche prend en entrée un nuages de points 3D de toitures et construit automatiquement un ensemble de polygones correspondant aux toits (voir Figure 2).

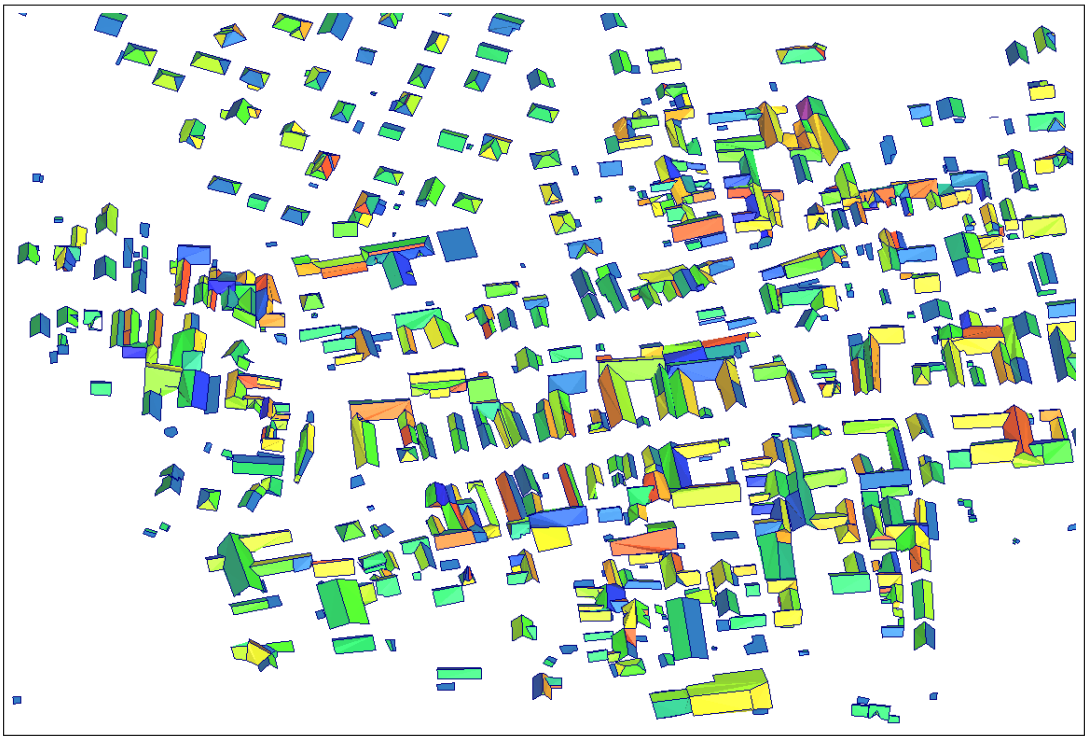


FIGURE 2 – Polygones construits automatiquement par notre méthode à partir des nuages de points de la figure 1.

2 État de l'art : modélisation de bâtiments à partir de données LiDAR

Le processus de modélisation de bâtiments à partir de nuages de points LiDAR comporte cinq étapes principales, à savoir l'acquisition des données, la trajectographie, le prétraitement, la segmentation et la modélisation (voir Fig.3). La modélisation constitue le sujet principal de cette étude. Nous nous intéressons, en particulier, à la reconstruction de polygones correspondant aux toitures à partir d'un nuage de points 3D préalablement classifiés dans des données LIDAR brutes.

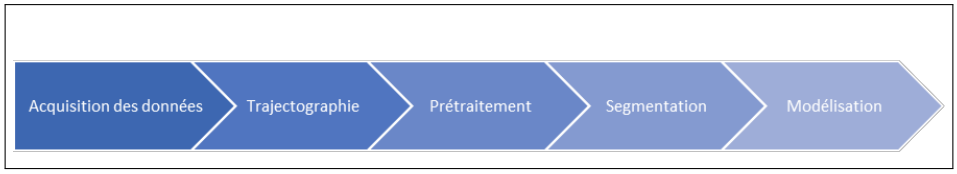


FIGURE 3 – Processus global de modélisation de bâtiments pour maquette numérique urbaine à partir de données LiDAR.

La modélisation géométrique des toitures constitue un défi majeur pour les efforts de recherche actuels à cause de la quantité et de la qualité des nuages de points 3D qui sont souvent corrompus par du bruit.

De nombreuses approches pour la reconstruction de modèles de villes 3D à partir de ces données ont été développées au cours de ces dernières années [13]. Les différentes techniques proposées peuvent être regroupées en deux familles, à savoir des approches paramétriques et des approches génériques (ou non-paramétriques).

2.1 Méthodes paramétriques

Les approches paramétriques sont aussi appelées des *Approches Basées Modèle* utilisent des modèles prédéfinis dans une bibliothèque. Les méthodes associées se basent sur un algorithme qui calcule dans le nuage de points un certain nombre de paramètres, puis compare ces paramètres à l'ensemble des modèles prédéfinis dans le catalogue. L'ordinateur va alors déterminer quel modèle correspond le mieux au nuage en question.

Les avantages de la modélisation paramétrique viennent du fait de leur vitesse de calcul, le fait qu'elles fournissent des modèles géométriques sans déformations visuelles. Néanmoins les modèles ne sont fiables que si la densité de points est relativement élevée et homogène. L'inconvénient majeur de la modélisation paramétrique est de ne proposer qu'un nombre réduit de modèles de toitures/bâtiments. Ce nombre réduit est destiné à conserver des temps de calculs les plus faibles possibles. Ainsi, pour la modélisation de bâtiments complexes, ces algorithmes réalisent des combinaisons de modèles simples. Cependant, l'hypothèse de base selon laquelle un bâtiment complexe (bloc de bâtiments) peut être décomposé en un ensemble de bâtiments basiques simples, n'est pas toujours vérifiée. Enfin, les catalogues des méthodes paramétriques peuvent rapidement devenir très volumineux si une grande zone urbaine avec un fort nombre de formes de bâtiments différentes doit être reconstruite.

Nguatem [18] proposent une approche adaptant automatiquement des formes de toitures paramétrées au nuage de points 3D à l'aide d'un échantillonnage séquentiel. La méthode représentée sur la figure 10 ci-dessous est la suivante : La donnée d'entrée est un nuage de points 3D non structuré, contenant un bâtiment quadrilatéral et son empreinte au sol. En commençant au point le plus haut au-dessus de l'empreinte et en descendant par séquence, l'algorithme estime les configurations de toits possibles à plusieurs niveaux de hauteur. A chaque séquence, la forme de la toiture proposée est réévaluée afin de parvenir à la plus «

juste » configuration pour chaque niveau de hauteur. Le résultat est un ensemble de surfaces polygonales connectées définissant le type de toit choisi. Cette méthode est adaptée à la modélisation de toits simples comme les toits à pignon, à croupe, à pinacle et les toits mansardes. Cependant, la forme quadrilatérale imposée est la limite majeure de la méthode.

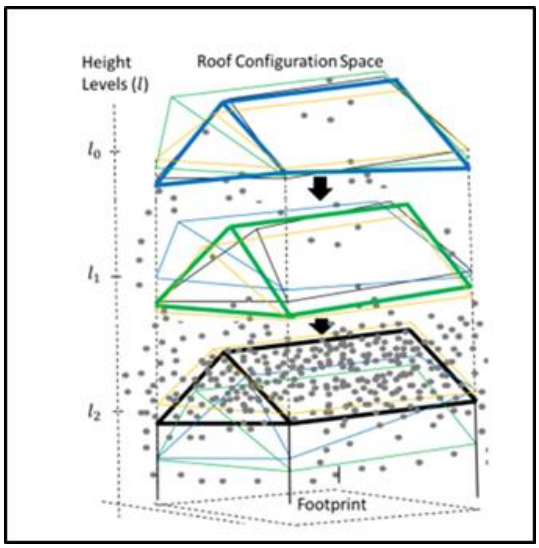


FIGURE 4 – Illustration de la méthode proposé dans [18]. A chaque hauteur de recherche, la configuration la plus probable est indiquée en gras.

D'après Maas et Vosselman [16], un bâtiment simple peut être décrit par un ensemble de paramètres (voir Fig.5) qui sont calculés en utilisant les équations des moments statiques d'ordre zéro, d'ordre un et d'ordre deux des points du nuage. Une fois ces paramètres obtenus, on vérifie la correspondance entre le modèle proposé grâce à ses paramètres et le nuage en question. Cette approche traite directement le nuage de points brut. Cependant, le nombre restreint de paramètres géométriques présents au sein des équations génériques proposées par cette technique limite les modèles établis à deux formes de bâtiments : les maisons avec un toit à deux pans inclinés et les maisons avec un toit plat.

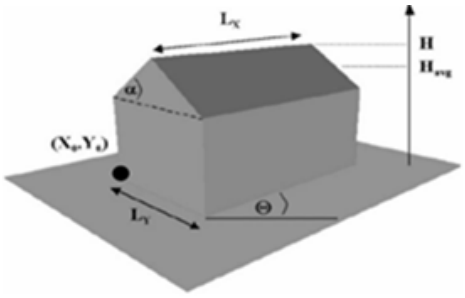


FIGURE 5 – Illustration des paramètres géométriques utilisés par la méthode de [16].

Verma [23] propose une approche de modélisation basée sur l'utilisation de graphes de topologie de toits (RTG). Cette méthode se compose de trois étapes :

— La segmentation des points de toitures : Tous les points représentant des surfaces localement non planes sont détectés puis éliminés. Les points restants sont regroupés en fonction de leurs proximités spatiales. On différencie ensuite les points de toitures de ceux du terrain. Pour cela, l'hypothèse

suivante est faite : le plus grand groupe de points représente le terrain. Cette composante est donc éliminée.

— L'inférence de la topologie : L'étape d'inférence topologique consiste à créer un graphe topologique des toitures. Ce graphe est une représentation de la structure du toit de telle sorte que chaque surface plane du toit est représentée par un sommet. Deux sommets du graphe topologique sont reliés si, et seulement si, les surfaces planes du toit correspondant aux deux sommets partagent une arrête de toit (voir Fig.6). Les liaisons entre chaque sommet sont caractérisées en fonction de la position relative des plans de toitures entre eux.

— L'ajustement paramétrique : Cette étape consiste à comparer les liaisons du graphe de topologie avec celles des modèles de toitures présents dans une bibliothèque. L'algorithme va alors choisir le modèle de toiture correspondant le mieux au graphe puis affiner sa position de manière à minimiser l'erreur entre le modèle de toit sélectionné et les points du nuage.
Les approches exploitant la RTG aboutissent à de bons résultats de reconstruction, cependant plus le nombre de sous-graphes prédéfinis nécessaires est grand, plus le temps de calcul engagé augmente.

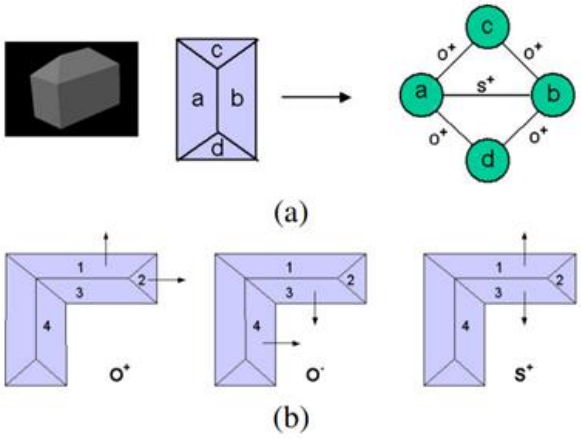


FIGURE 6 – Illustration de la méthode de modélisation par les graphiques de topologie de toits utilisée par [23].

Kada et McKinley [12] proposent une approche basée sur la décomposition cellulaire. Cette méthode nécessite les contours 2D des toitures en complément du nuage de points LiDAR. Chaque contour de bâtiment est décomposé en cellule non chevauchante (exemple en Fig.7), de préférence de forme rhomboïde ou trapézoïdale. Les cellules sont ensuite divisées en huit sections (Fig.7) afin de déterminer efficacement le modèle correspondant le mieux au nuage. Les vecteurs normaux de tous les points d'une section sont ensuite étudiés pour affecter une orientation globale à chacune d'elle (zones de couleur dans la figure 14). Le modèle retenu sera celui ayant le plus de similitude avec l'un des modèles de la bibliothèque.

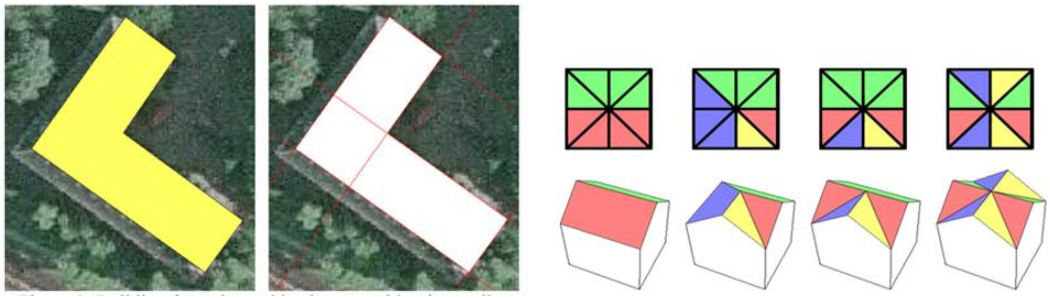


FIGURE 7 – Illustration de la décomposition en cellule et des modèles avec leur division respective en sections, utilisés dans [12].

Lafarge [14] utilise un Modèle Numérique d'Elévation (MNE) et propose également une approche basée sur la décomposition cellulaire. Cependant contrairement à Kada et McKinley [12] le choix du gabarit de forme approprié n'est pas basé sur l'orientation principale d'une autre subdivision, mais il utilise un modèle probabiliste de Gibbs pour contrôler l'assemblage de blocs et l'ajustement aux données. L'objectif est de parvenir à une configuration maximisant la cohérence entre la réalisation et le MNE ; ainsi que la cohérence des assemblages entre les cellules, à l'aide de connaissances « à priori » de l'agencement des modèles entre eux. La valeur U de l'énergie de Gibbs associée à la densité permet, à priori, de favoriser l'assemblage d'objets de manière proportionnelle à la qualité de l'assemblage.

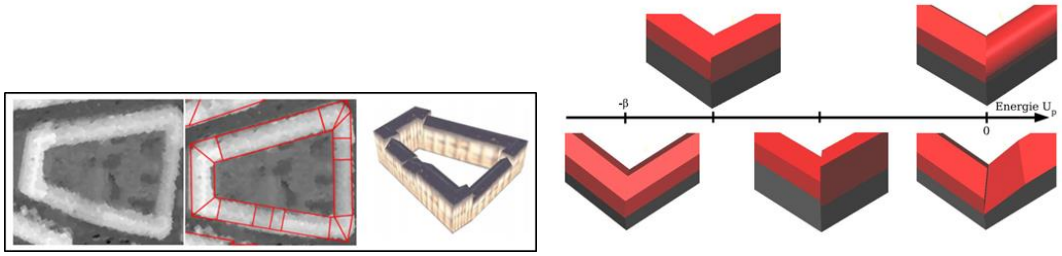


FIGURE 8 – Illustration de la décomposition en cellule par utilisés dans [14] et principe de l'assemblage d'objets à l'aide de l'énergie de Gibbs.

2.2 Méthodes non-paramétriques

La modélisation non-paramétrique est basée sur le nuage de points, sans référence à des modèles prédéfinis ni à d'autres sources d'information. Plusieurs méthodes ont été développées dans ce sens avec comme avantage essentiel de pouvoir construire n'importe quelle forme de bâtiment. Car contrairement aux approches de reconstruction paramétrique, elles ne se limitent pas à un ensemble prédéfini de formes de bâtiments et ne possèdent donc pas les restrictions associées à un catalogue de formes. Elles ressemblent alors de très près aux données d'entrée. En revanche, le modèle 3D d'un bâtiment complexe pourra contenir beaucoup de déformations visuelles si un ajustement des formes extraites n'est pas réalisé. La sémantique et les contraintes géométriques sont importantes puisqu'elles améliorent le rendu des modèles de bâtiments 3D.

En général, ce type de modélisation se divise en deux parties. Tout d'abord une segmentation du nuage de points à l'aide de modèles mathématiques est réalisée. Elle permet d'extraire des formes géométriques (plans, arêtes...) représentant la structure des bâtiments et plus particulièrement des toitures. Puis les

modèles de bâtiments 3D sont construits sur la base des éléments précédemment extraits.

2.2.1 Méthodes d'extraction de plans

L'un des principaux défis pour une reconstruction réussie de modèles de bâtiments tridimensionnels à partir de nuages de points lidar aériens consiste à obtenir une reconnaissance et une segmentation de haute qualité des primitives planaires des toits. Cependant, la détection des plans des toitures est une tâche complexe car les données de nuages de points 3D ne contiennent aucune information de connexion et ne fournissent aucune caractéristique sémantique des surfaces analysées sous-jacentes. De plus, la densité irrégulière des nuages de points, le bruit d'acquisition, les profils de toit complexes et l'occlusion ajoutent à la difficulté.

De nombreuses méthodes d'extraction de primitives planaires à partir de nuages de points 3D ont été développées telles que l'analyse en composantes principales (PCA en anglais) [28] , l'approche par "RANdom Simple Consensus" (RANSAC) [21, 15], l'approche par transformée de Hough [1] et l'approche par croissance de surfaces (ou de régions) [4].

Les trois dernières approches sont des concurrents majeurs dont nous exposons brièvement les principes ci-bas.

2.2.2 Le paradigme de RANSAC

Le paradigme de RANSAC (RANdom SAmple Consensus) est une méthode fréquemment appliquée pour la segmentation de forme géométrique dans un nuage de points. Il a été créé et utilisé par Fischler et Bolles [8] afin de détecter des entités géométriques en 2D mais a été réadapté par la suite à la 3D.

L'algorithme RANSAC est une méthode itérative pour estimer les paramètres d'un certain modèle (droite ou plan, par exemple) à partir d'un ensemble de données observées.

Pour la reconstruction d'un plan, le principe de l'algorithme est le suivant :

1. Sélectionner au hasard 3 points dans les données, ce qui définira un plan $\mathscr{P}$
2. Trouver les distances des points restants du plan $\mathscr{P}$. Les points dont la distance est inférieure à une distance t sont appelés *inliers* et appartiennent au plan $\mathscr{P}$. On sauvegarde les trois points ainsi que nombre d'*inliers*, cet enregistrement s'appelle "*best_model*".
3. répéter 1. et 2. k fois ou jusqu'à ce qu'aucun plan avec un nombre de points supérieur à d ne puisse être trouvé. A chaque fois, si le nombre d'*inliers* est supérieur à celui du *best_model*, remplacer *best_model* géré précédemment par le nouveau. Au final, les paramètres du modèle de plan sont déterminés à partir du *best_model* final.

Cet algorithme est fréquemment utilisé dans la littérature pour l'extraction de plans représentant les toitures dans un milieu urbain.

Tarsha-Kurdi [22] propose d'améliorer cet algorithme en prenant en compte non seulement le nombre de points, mais aussi la distribution des points autour du plan par le biais de l'écart-type des distances entre point et plan moyen au moment de la validation.

Une étude de la qualité des facette de toitures extraites à l'aide de l'algorithme de RANSAC à partir de données LIDAR peut être trouvée dans [27].

2.2.3 Transformée de Hough

La transformée de Hough [11] est un procédé initialement utilisé dans le traitement numérique d'image pour l'extraction de formes géométriques simples qui peuvent répondre à un modèle mathématique para-

métrable telles que des droites et des cercles, par exemple.

Le principe de la transformée de Hough appliquée au cas d'une droite 2D consiste à traduire chaque point de l'espace euclidien $(O\ x\ y)$, par une sinusoïde (suite à une transformation en coordonnées polaires) dans l'espace des paramètres $(O'\ \theta\ \phi)$ dit « espace de Hough ». Une droite passant par un certain nombre de points sera visualisée dans l'espace de Hough par un regroupement d'intersections (voir Fig.9).

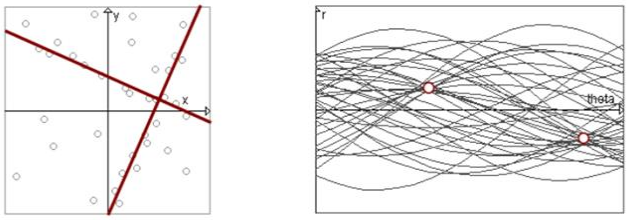


FIGURE 9 – Principe de la transformée de Hough, cas de la recherche de lignes 2D.

Cette technique est transposable en 3D et permet de détecter des plans dans des nuages de points 3D. Pour cela, il suffit tout simplement d'incorporer la composante z, au principe vu précédemment et d'utiliser l'équation suivante d'un plan :

$$cos\theta.sin\phi.x + sin\phi.sin\theta.y + cos\phi.z \ = \ \rho \ (1)$$

Où θ est l'azimut (angle du vecteur normal sur le plan xy), ϕ est l'angle d'élévation et ρ la longueur du segment normal au plan et passant par l'origine. Les paramètres θ, et ρ sont les paramètres de la normale du plan passant par l'origine. Ainsi, un plan peut être présenté comme un point de l'espace paramétrique de Hough. L'intersection de 3 surfaces sinusoïdales dans l'espace paramétrique de Hough correspond aux coordonnées polaires (i.e., θ, et ρ) du plan défini par les trois points.
En mettant en oeuvre l'équation (1) à chaque point de l'ensemble de points 3D en utilisant un pas d'angle prédéfini de θ et dans une plage d'angle, de nombreuses intersections dans l'espace paramétré de Hough auront lieu. Cependant, les plans saillants souhaités sont ceux qui se produisent dans les régions de concentration des points de l'espace paramétrique de Hough. Ces plans saillants sont associés au plan commun auquel appartiennent les points correspondants du point 3D défini dans l'espace $(O\ x\ y\ z)$. Théoriquement, les points coplanaires dans l'espace $(O\ x\ y\ z)$ devraient partager un point dans l'espace paramétrique de Hough (voir Fig.10).

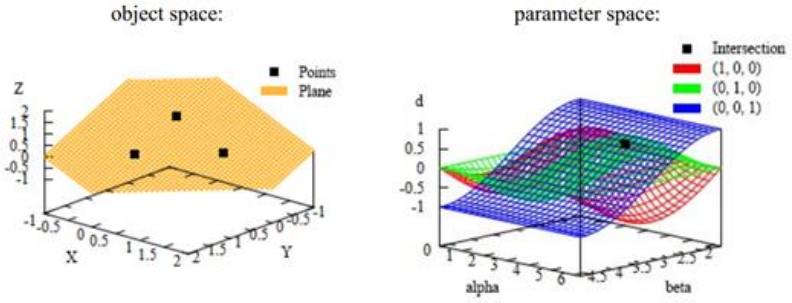


FIGURE 10 – Transformation de trois points coplanaires de l'espace objet vers l'espace de Hough. Source [26]

Les premières approches dans lesquelles la transformée 3D de Hough est utilisée pour la reconstruction

automatique de bâtiments à partir de nuages de points 3D sont présentées dans [25, 16]. Depuis, ce procédé a été fréquemment utilisée dans la littérature pour détecter des plans dans des nuages de points 3D [1, 17].

2.2.4 Croissance de régions

Une troisième méthode courante pour la segmentation d'éléments des structures de toitures est la segmentation basée sur la croissance de surface (ou de région).

Contrairement à la transformation de Hough et le paradigme de RANSAC, la croissance de surface est une méthode de segmentation qui ne prend en compte que le voisinage local. Également issue du traitement numérique des images 2D, elle consiste à identifier des « surfaces graines » (surfaces de départ) pour ensuite les faire croître vers les points voisins. L'algorithme tente alors de propager la « surface » de manière itérative dans le nuage de points. Tant que les points voisins répondent à certains critères, ils sont acceptés dans la « surface ». En l'absence de point voisin correspondant aux critères, l'algorithme s'arrête et la surface est dite close. L'un des défis principaux lorsque l'on utilise cette méthode est de choisir la bonne « surface graine », puisque ce choix va impacter directement la qualité de la segmentation. De plus, les méthodes proposées dans la littérature utilisent différents critères de similarité entre les points lors de la propagation : courbure, planearité, angles entre les normales aux points etc. [4, 20, 3, 10].

Plus récemment Wichmann [26] utilise une amélioration de cet algorithme appelée « croissance de sous-surface ». Cela consiste à utiliser un algorithme de croissance en surface pour trouver un ensemble initial de segments planaires puis de poursuivre par une procédure de croissance sous la surface qui génère des points virtuels en dessous des points de surface déjà segmentés. De cette façon, les trous dans un segment qui sont causés par les superstructures du bâtiment par exemple sont automatiquement bouchés. De plus, les superstructures sont mieux représentées lors du résultat final de la segmentation.

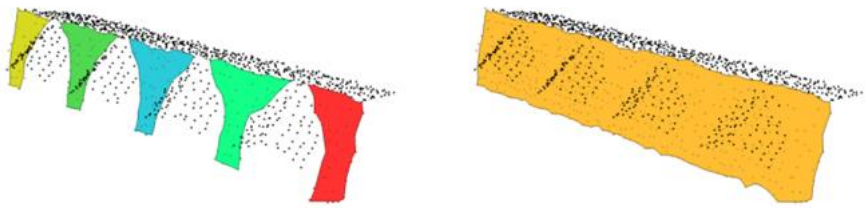


FIGURE 11 – Résultats de segmentation de plans à l'aide d'un algorithme de croissance de surface (à gauche) et de croissance "sous la surface " (à droite) selon [26]

3 Notre méthode

Dans cette section, nous allons présenter notre pipeline de traitements. En entrée, il prend un nuages de point 3D de toitures, préalablement segmenté à partir de données LIDAR aériens d'un lotissement ou d'une ville. En sortie, il fournit un modèle polygonal du toit de chaque bâtiment dans lequel chaque pan est représenté par un polygone plat et les différents pans sont correctement connectés.

Notre méthode peut être résumée en trois grandes étapes, à savoir : D'abord, nous délimitons les composantes connexes correspondant aux différentes toitures contenues dans le nuage de point du lotissement. Ensuite, nous procédons toitures par toiture et commençons par détecter les primitives planaires (les pans) de chaque toiture. A la fin, nous construisons et connectons les polygones correspondant aux pans de chaque toiture.

3.1 Étape 1 : Composantes connexes des toitures

Cette étape préliminaire permet de délimiter les composantes connexes correspondant aux différentes toitures dans le nuage de points global d'un lotissement. Pour ce faire, nous avons utilisé l'algorithme de partitionnement selon la distance Euclidienne, fourni par la bibliothèque PCL (Point Cloud Library) [19].

Cet algorithme a trois paramètres : les tailles de cluster minimale *MinClusterSize* et maximale *MaxClusterSize* et tolérance de cluster *ClusterTolerance*. Ce dernier paramètre doit être défini avec soin car si nous prenons une très petite valeur, il peut arriver qu'un objet puisse être vu comme plusieurs clusters (sur-segmentation). En revanche, si nous définissons une valeur trop élevée, il peut arriver que plusieurs objets soient considérés comme un seul cluster (sous-segmentation). Ainsi, comme recommandé par la documentation PCL, nous avons testé et déterminé quelle valeur convient à nos ensembles de données et nous avons utilisé $MinClusterSize = 100$, $MaxClusterSize = 500000$ et $ClusterTolerance = 0.5$.

La sortie de cet algorithme est l'ensemble des points d'entrée où chaque point a reçu une étiquette unique correspondant au cluster auquel il appartient. Par conséquent, chaque cluster correspond au toit d'un bâtiment unique ou d'un ensemble de bâtiments contigus, pour les toitures plus complexes (voir Fig. 12).

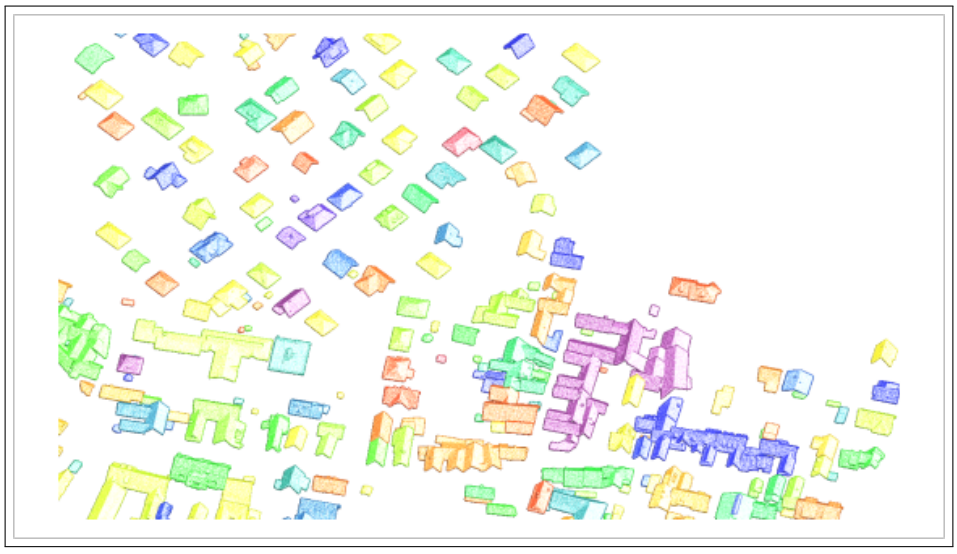


FIGURE 12 – Résultat de la segmentation du nuage de points global en composantes connexes correspondantes aux différentes toitures.

3.2 Étape 2 : Détection des primitives planaires

L'extraction des primitives géométriques planes est l'une des étapes les plus importantes en vue de la construction des polygones correspondant aux toitures. Comme nous l'avons vu précédemment, il existe plusieurs stratégies permettant de ce faire. Nous avons implémenté et testé deux parmi elles, à savoir une première approche combinant la transformée de Hough pour l'estimation des normales aux point suivie d'une croissance de surface et une seconde approche basée sur le paradigme de RANSAC que nous avons adaptée pour le cas particulier des toitures.

Avant de donner les détails de ces méthodes, notons d'abord que la détection de plans est réalisée pour chaque composante connexe issue de l'étape précédente de notre pipeline.

3.2.1 Approche 1 : transformée de Hough et croissance de régions

Cette approche commence par estimer les normales aux points d'entrée en utilisant la transformée de Hough robuste fournie par Bulch[2]. Ceci qui revient à chercher les plans tangents aux points en prenant un voisinage de 30 points (voir Fig.13).

Ensuite, nous utilisons ces normales dans un algorithme de croissance de surface afin de séparer les différents pans de la toitures. L'algorithme commence par choisir un point avec une courbure minimale de l'ensemble de points et qui est considéré comme le germe pour le pan en cours. Ensuite, il propage le pan aux points voisins tant que l'angle entre les normales n'excède pas pas 2 degrés. L'algorithme recommence jusqu'à ce qu'il n'y ait plus de points non-étiquetés.

Finalement, nous calculons l'équation cartésienne du plan pour chaque pan. Pour ce faire, nous re-estimons les normales des points d'un versant en prenant un voisinage plus grand (typiquement 20% des points du pan) et calculons la normale du plan comme étant la moyenne des normales des points qui le composent :

$$\vec{n}_{moy} = \sum_{i=1}^{k} n_i/k = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}$$

Nous en déduisons les coefficients a, b, c et d de l'équation cartésienne du plan $\mathscr{P}$: $ax + by + cz + d = 0$ comme suit : $a = n_x$, $b = n_y$, $c = n_z$, $d = -(n_x x + n_z y + n_z z)$

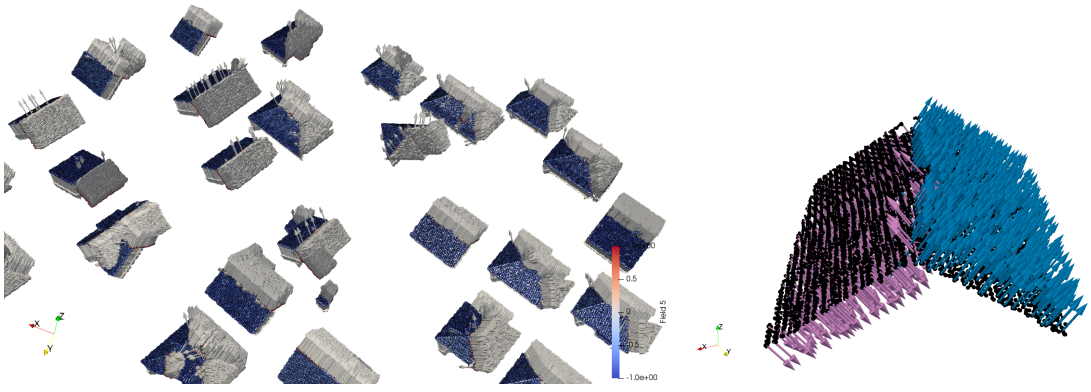
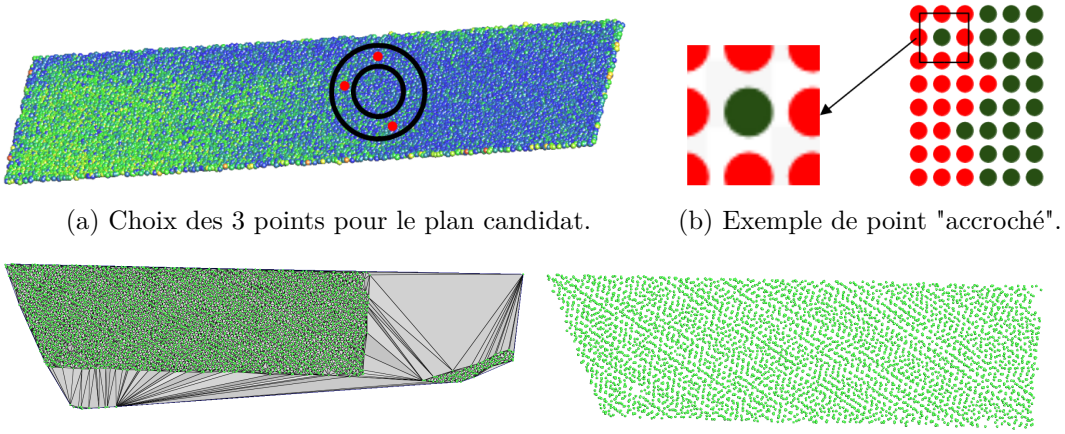


FIGURE 13 – A gauche, normales estimées par la transformée de Hough sur les toitures. A droite, deux versants obtenus par propagation de surface.



(a) Choix des 3 points pour le plan candidat.

(b) Exemple de point "accroché".

(c) Exemple de plan brute et son maillage de Delaunay

(d) Exemple de plan après raffinement

FIGURE 14 – Détails de notre algorithme de RANSAC.

3.2.2 Approche 2 : basée RANSAC

La méthode de RANSAC permet de segmenter les points en entrée en fonction du pan auquel ils appartiennent. Notre algorithme prend un point au hasard et choisit ensuite 3 points qui se trouvent dans son 128-voisinage et ne se trouvent pas dans son 64-voisinage (voir Fig.14a). Ceci revient à essayer de générer des plans en prenant les 3 points dans un voisinage suffisamment éloigné du point d'origine pour ne pas être trop sensible au bruit mais suffisamment proche pour rendre compte de la planéité locale. Ces 3 points définissent un plan candidat. Le reste des points du nuage sont alors testés dans l'équation du plan candidat avec un seuil de distance par rapport au plan (seuil = 1.0/12). Après plusieurs itérations, l'algorithme trouve les 3 points qui représentent le meilleur plan candidat. L'algorithme s'arrête lorsque le nombre de points restant est inférieur à 128 ou lorsque 64 itérations successives échouent (en détectant des pans avec moins de 100 points).

Élimination des points en trop : RANSAC détecte des plans dans l'espace 3D qui ne se résument pas forcément à des pans de toits. Etant donné que c'est une méthode paramétrique, tous les points du même plan mathématique sont rassemblé dans l'ensemble détecté, il faut donc ensuite retiré les points qui ne font pas partie du vrai pan. Comme montré sur la figure 14d, il est fréquent que le plan détecté par l'algorithme "accroche" d'autres ensembles de points, plus lointains et faisant partie d'autres pans de la toiture Ce sont, en fait, des points qui vivent dans le même plan en 3D mais qui font partie de pans différents d'une toiture complexe.

Afin de supprimer ces points en trop, on calcule la triangulation de Delaunay 2D sur le résultat brute de l'algorithme et on élimine les triangles qui ont une arête trop longue. Puis on parcourt les arêtes restantes depuis un point aléatoire du pan et on labellise les points atteints. Les points non-labellisés sont retirées du pan et ré-injectés dans l'ensemble des points à traiter.

Re-étiquetage des points isolés : Le dernier raffinement nécessaire consiste à supprimer les petits îlots de points d'un plan, entièrement contenus dans un autre. Ces configurations arrivent essentiellement le long des arêtes entre différents plans, comme montré sur la Fig.14b, et elles posent des problèmes dans la suite des traitements. Pour ce faire, nous changeons les labels de ces points en les assignant au plan

dans lequel ils ont le plus de voisins. Nous avons fait des tests avec 10, 20, 30 et 50 voisins et il s'avère que le nombre de changements effectifs est borné dans un petit voisinage. Par conséquent, notre algorithme re-étiquette les points en prenant le pan le plus présent dans leur 12-voisinage.

En sortie de cet algorithme, nous avons un fichier contenant tous les points retenus avec comme attribut leur plan et un fichier contenant les plans détectés sous forme de quatre nombres nx, ny, nz, d calculés comme pour la première approche (voir section précédente).

Notons que les petits plans, de moins de 100 points, correspondant à des fenêtres de toits et des cheminées sont ignorés par cet algorithme.

3.2.3 Comparaison des deux approches de segmentation en plans

Nous avons appliqué les deux méthodes de segmentation en plans sur notre base de données de toitures. Les expérimentations ont montré que les deux approches donnent des résultats semblables et équivalents pour les toits simples de deux et quatre pans. Par contre, il s'avère que la première approche, basée sur la transformée de Hough et de la croissance de surface, fournit des meilleurs résultats pour les toitures complexes, composées de plus d'une dizaine de pans, telles que montrées sur Fig.15 et Fig.16. On remarque que les plans détectés sont plus complets et les petits plans sont mieux détectés avec la première approche.

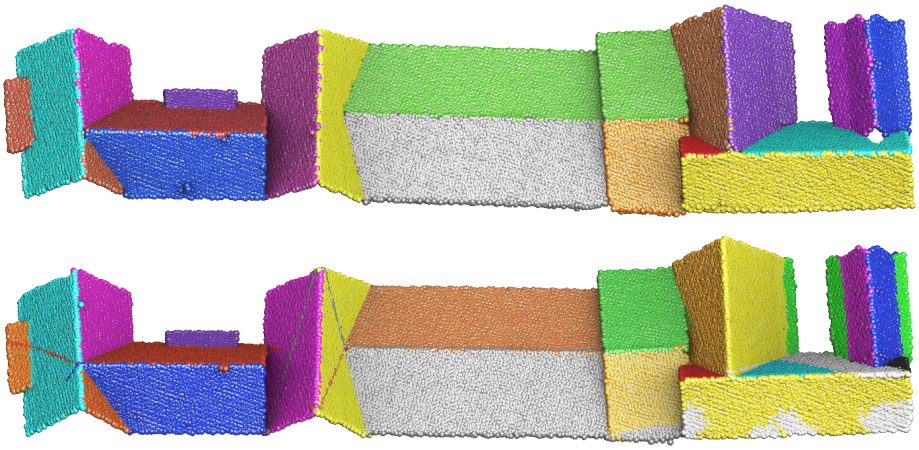


FIGURE 15 – Comparaison de la segmentation en plans avec l'approche 1 (en haut) et l'approche 2 (en bas).

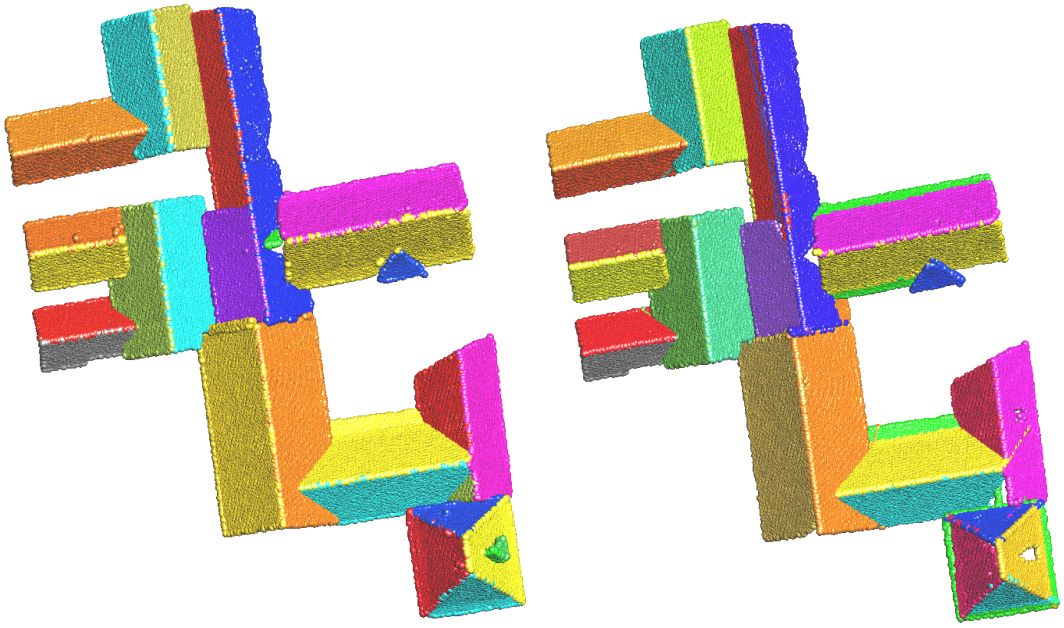


FIGURE 16 – Comparaison de la segmentation en plans avec l'approche 1 (à gauche) et l'approche 2 (à droite).

3.3 Étape 3 : Construction des polygones

Sur la base de la segmentation en pans effectuée à l'étape précédente, cette étape se charge à extraire d'abord les contours extérieurs des toitures et les segments d'intersections entre les différents pans. Ensuite, ces segments sont simplifiés et utilisés pour construire topologiquement les polygones correspondant aux différents versants. Finalement, le plongement géométrique des polygones est raffiner par projection afin qu'ils soient parfaitement planaires.

3.3.1 Extraction du contour extérieur et des arêtes entre pans

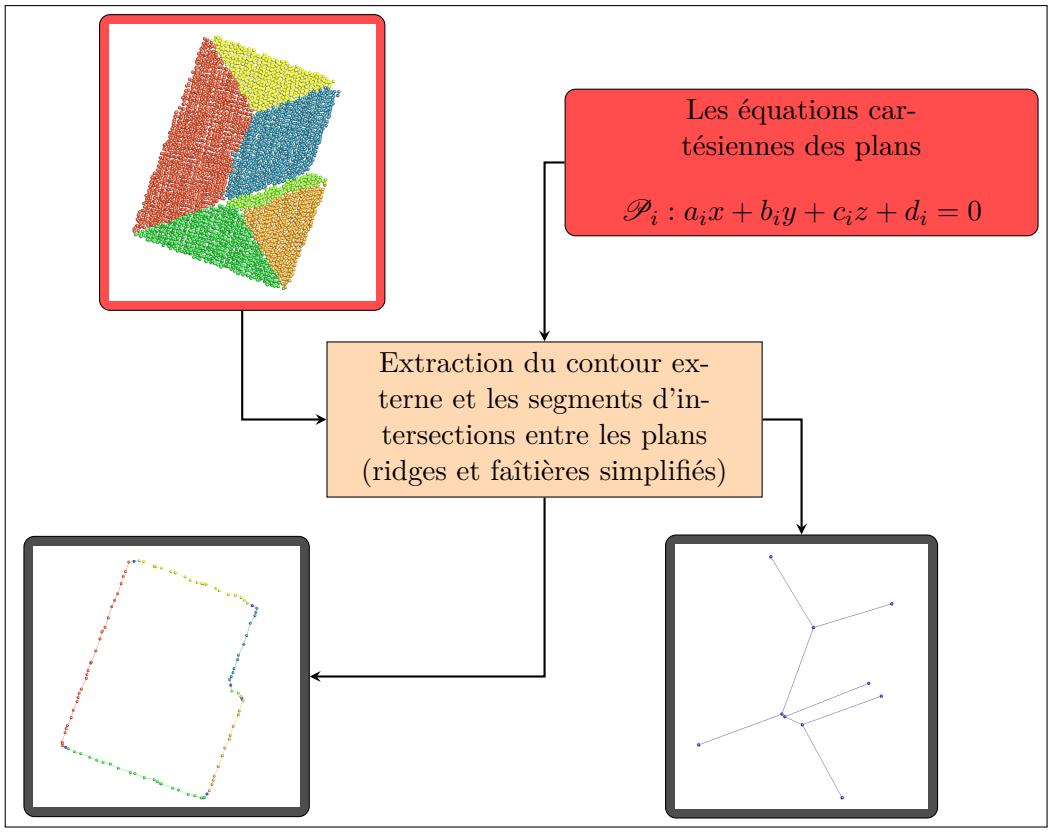


FIGURE 17 – Résumé de l'extraction des contours externes et internes.

Les segments qui décrivent la topologie de chaque toiture peuvent être séparées en 2 catégories :

— le contour extérieur qui correspond à la bordure de la toiture

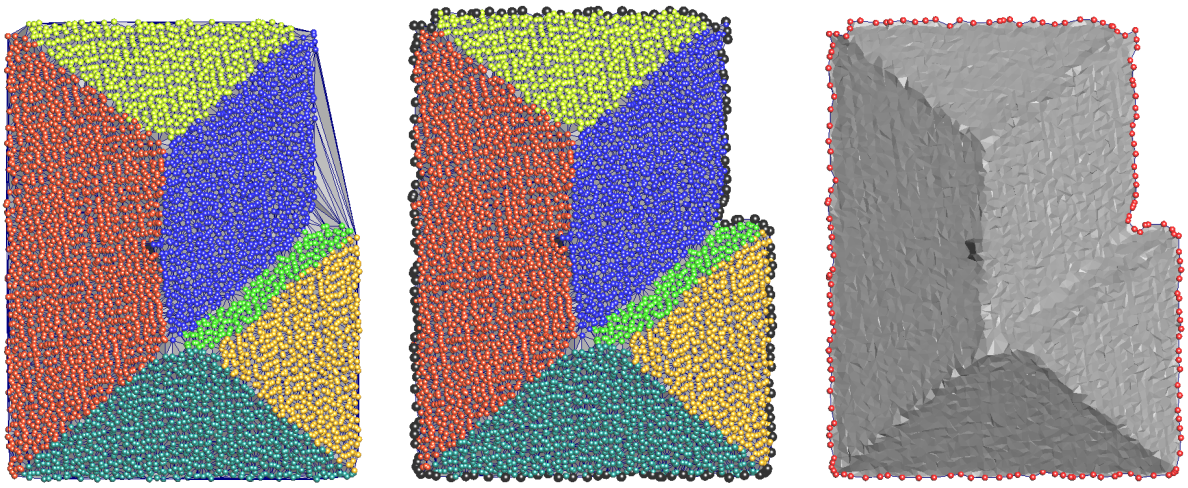
— les segments intérieurs qui correspondent à la jonction de plusieurs versants.

Pour extraire ces segments, nous utilisons le maillage correspondant au α-shape des points après projection sur le plan (x, y) (ou tout simplement en ignorant la composante en z des points).

Rappelons que la triangulation de Delaunay d'un ensemble de points 2D fournit l'*enveloppe convexe* de ces points (voir Fig.18a) alors que l'α-shape donne l'*enveloppe concave* des points qui nous intéresse ici (voir Fig.18b). En pratique, pour obtenir l'α-shape, il suffit de calculer la triangulation de Delaunay des points et d'y supprime toutes les arêtes plus longues que α.

Ainsi, la bordure extérieure de la toiture peut être définie comme l'ensemble des arêtes du maillage incidentes à un unique triangle (voir. Fig.18c).

3.3.1.1 Algorithmes d'érosion du maillage de Delaunay : L'algorithme part de la triangulation de Delaunay 2D des points projetés sur le plan $0z$. C'est une triangulation de l'enveloppe convexe des points que l'on veut éroder depuis l'extérieur pour obtenir le polygone concave correspondant à la bordure du toit. L'algorithme commence par stocker la bordure convexe dans une liste L des arêtes appartenant à un seul triangle. Il itère sur ces arêtes pour trouver les arêtes les plus *longues* ce qui lui permet de trouver les triangles à supprimer. Étant donné une arête longue ($> dmax = 0.6^2$), l'algorithme supprime son triangle incident et rajoute les deux autres arêtes de celui-ci dans le liste L à tester. Quand la liste



(a) Triangulation de Delaunay des points.

(b) α-shape après érosion du maillage initial.

(c) La bordure extérieure soulignée en rouge.

FIGURE 18 – *alpha*-shape des points d'une toiture et bordure extérieure.

L est entièrement parcourue, l'algorithme supprime les arêtes qui n'appartiennent à aucun triangle. En sortie, l'algorithme fournit la bordure extérieure de la toiture sous forme d'un ensemble d'arêtes, que nous allons devoir chaîner avant la suite des traitements.

Nous avons implémenté trois versions de l'algorithme d'érosion du maillage de Delaunay initial car la distance utilisée pour décider si une arête est *trop longue* peut être soit une distance 2D, soit une distance 3D. De plus, nous avons développé une version qui n'ignore pas les contours intérieurs. Ces derniers sont souvent liés à des fenêtres et n'apportent pas grand-chose à la modélisation du toit, tout en rendant le traitement plus compliqué. Cependant, le fait de les ignorer peux parfois fausser sensiblement le résultat.

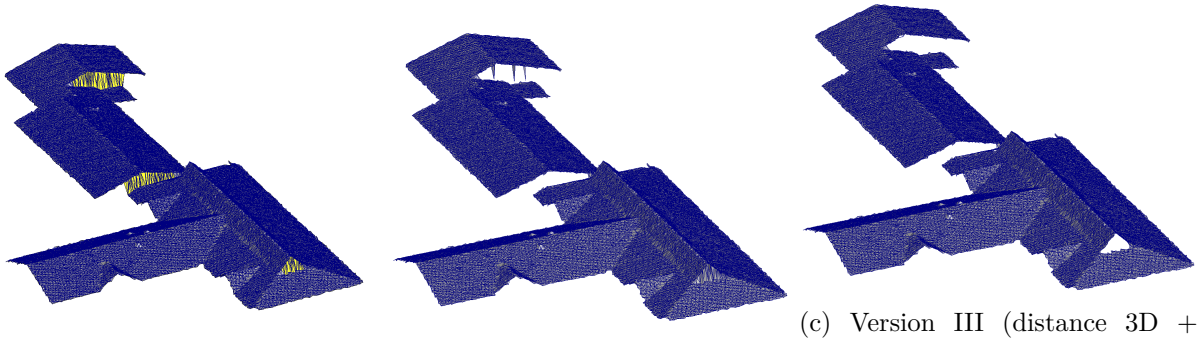
Donc la version I de l'algorithme utilise une distance 2D lors de l'érosion. Ceci a comme effet de ne pas détecter les arêtes longues en 3D telles que celles entre deux versants ayant une différence en hauteur (on parle de "step edges" en anglais), comme les triangles jaunes sur la Fig.19a.

La version II de l'algorithme d'érosion utilise une distance 3D ce qui permet d'éroder le maillage également entre les pans ayant une forte différence en hauteur, comme montré sur la Fig. 19b. Dans ce cas, les contours contiennent fréquemment des boucles qui proviennent d'une différence verticale entre deux versants.

Dans la version III, l'algorithme prend en compte aussi les labels des points ce qui lui permet d'extraire les contours intérieurs qui sont dus à des fenêtres (voir Fig.19c).

3.3.1.2 Algorithme de chaînage des arêtes de la bordure : Cette algorithme permet d'arranger les arêtes dans un ordre quelconque. En entrée, il prend l'ensemble des arêtes de la bordure et un booléen qui indique s'il doit faire un *abort* si le contour créé n'utilise pas toutes les arêtes en entrée. En sortie, il fournit une unique polyligne donnée sous la forme d'une suite d'indice de point, dont chacun est liée par une arête au suivant et au précédent. La sortie est une polyligne simple et fermée s'il n'y a pas de boucles dans le contour d'entrée. Sinon voir les explications ci-bas et la Fig.20.

Le contour rangé est stocké dans un vecteur d'index, initialisé à vide. On stock le premier $v0$ et le deuxième sommets $v1$ de la première arête comme premier et second élément du contour trié. Pour chaque sommet suivant, on cherche l'arête qui lient le point précédent et qui n'a pas déjà été utilisé dans le but



(a) Version I (distance 2D). (b) Version II (distance 3D). (c) Version III (distance 3D + trous intérieurs).

FIGURE 19 – Différentes versions de l'algorithme d'érosion du maillage initial.

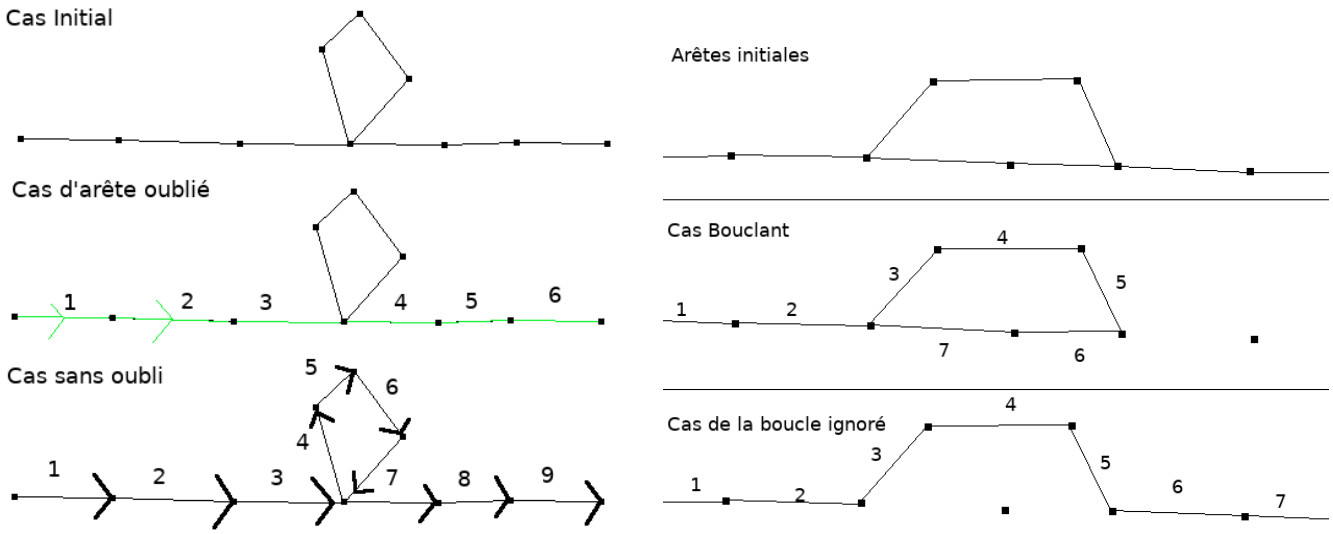


FIGURE 20 – Scénario possibles liés à la présence d'une boucle dans la bordure.

de trouver le sommet actuel. La boucle continue tant que l'on n'a pas fait un contour de la même taille que l'entrée et qu'il n'y a pas d'échec pour trouver l'arête suivante.

Enfin, avant de retourner le résultat, le programme vérifie (avec un *assert*) si les arêtes ont été toutes utilisées, et il affiche un message d'erreur (la taille du contour chaîné != la taille du contour d'entrée), et en fonction du booléen d'entrée, fait un *abort* ou pas.

Il y a deux types de boucles : les boucles liée à un seul point (4 arêtes sur un point) ou les boucles qui se rattachent à 2 points, ayant chacun 3 arêtes. À partir de là, quatre scénario peuvent se dérouler quand le programme rencontre une boucle (voir Fig.20).

Si une boucle sur un seul point est présente, deux comportements peuvent arriver :

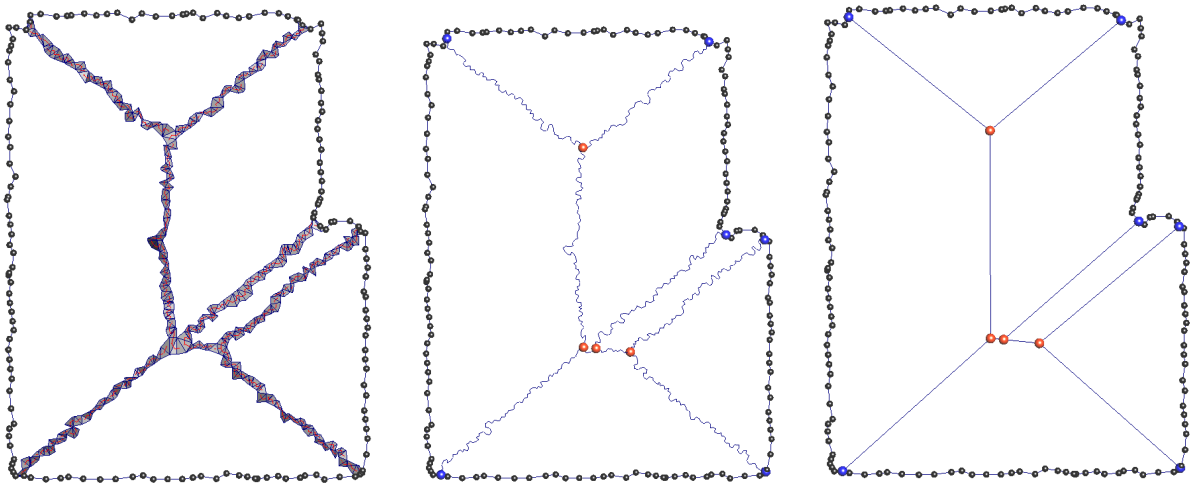
— il passe par la boucle, revient sur le point, et utilise une arête non utilisé auparavant, donc continue son chemin.

— il passe à côté de la boucle.

Si une boucle passe par plusieurs points, deux comportements peuvent arriver :

— il passe par l'un des deux chemins disponibles et oublie le second

— il passe par les deux chemins et se retrouvent sans arêtes à explorer, et s'arrête.



(a) Triangles entre plusieurs pans. (b) Arêtes intérieures brutes. (c) Arêtes après simplification.

FIGURE 21 – Arêtes internes entre les différents versants d'un toit simple.

3.3.1.3 Arêtes entre les versants Une fois que nous avons obtenu la bordure extérieure de chaque toiture, il faut encore récupérer les arêtes entre les différents versants. Pour ce faire, nous cherchons dans le maillage α-shape les triangles reliant des points appartenant à des plans différents, en utilisant les labels des points (voir Fig.21a). Si un triangle relie deux ou trois pans différents, alors on peut considérer que le centre du triangle se trouve sur la droite séparant les pans. Ainsi, nous relions les centres de ces triangles ce qui nous donne les segments intérieures sous forme d'une suite de petites arêtes (voir Fig.21b). Si une arête n'appartient qu'à un seul triangle, on la connecte avec la bordure extérieure.

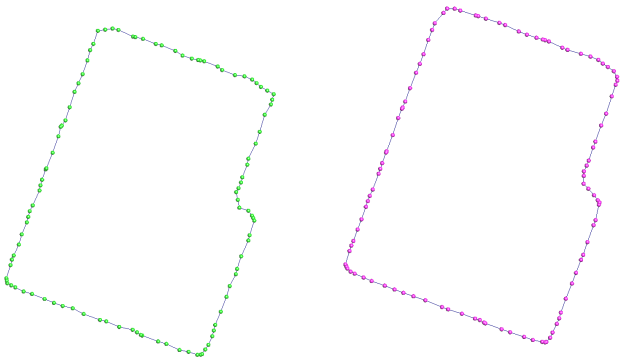
Finalement, nous simplifions les arêtes entre les versants en enlevant les sommets qui sont connectés à exactement deux sommets voisins pour ne conserver que les sommets d'intersections (en rouge sur la Fig.21c), ainsi que les sommets d'intersection avec la bordure extérieure (en bleu sur la Fig.21c). Cela permet, entre autres, d'ignorer les contours des fenêtres qui ne sont pas reliés à la bordure, car elles ne présentent en général aucun sommet d'intersection avec les arêtes intérieures.

3.3.2 Simplification du contour externe

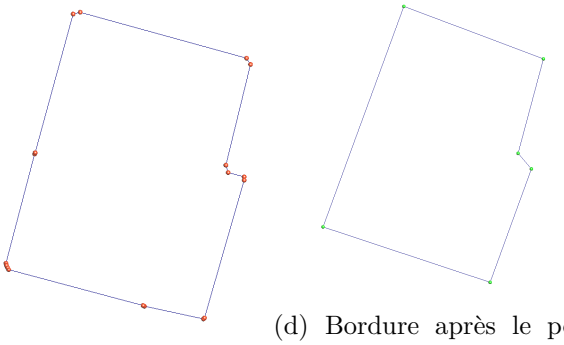
Notons que le contour extérieur extrait depuis le maillage de l'α-shape des points est composé d'une multitude de petites arêtes connectant les points (voir Fig.23a). Donc, avant de poursuivre nous devons simplifier ces bordures de sorte à ne garder que quelque points significatifs tels que les coins entre plusieurs segments.

Il existe dans la littérature plusieurs méthodes de simplification de polygones, les deux méthodes les plus connues sont l'algorithme de Douglas–Peucker [5] et l'algorithme de Visvalingam's [24]. Dans la première méthode, on garde le point le plus éloigné B d'un segment formé par le premier et le dernier point A (respectivement C) d'une polyligne s'il est supérieur à un *seuil* et on fait la même chose récursivement pour les nouveaux segments $[A, B]$ et $[B, C]$. Dans la deuxième méthode on supprime le point qui engendre le moins de changement visible sur la polyligne. L'algorithme que nous avons développé est inspiré de la deuxième méthode que nous avons adaptée à notre problème. Cet algorithme nécessite un pré-traitement (lissage) et un post-traitement (garder un seul point dans une sphère F de rayon r).

Le lissage du contour consiste à faire projeter un point P_i sur la droite qui passe par ses voisins direct



(a) Bordure extérieure brute. (b) Bordure extérieure lissée.



(c) Bordure simplifiée. (d) Bordure après le post-traitement.

FIGURE 23 – Résultat de la simplification de la bordure extérieure.

P_{i-1} et P_{i+1} (P_{i-1} P_i *et* P_{i+1} appartiennent à un seul et même plan).

Algorithm 1: Algorithme de lissage du contour

Input: $\mathscr{C}$: contour qui se compose d'un ensemble de points triés
Result: $\mathscr{C}$: contour lissé

```
foreach point P_i dans le contour C do
    if |cos(P_{i-1} P_i P_{i+1})| < 0.8 then
        Projeter P_i sur la droite (P_{i-1} P_{i+1})
    end
end
```

Le lissage sert à faire des ouvertures d'angle pour que le point P_{i+1} ne soit pas considéré comme un point du coin (voir Fig.23b).

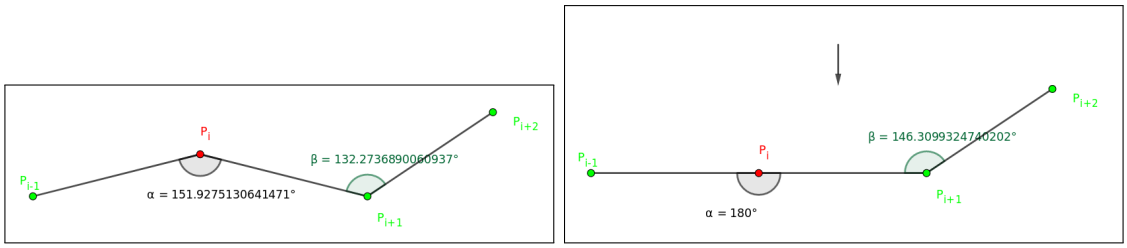


FIGURE 22 – Projection du point P_i sur la droite $(P_{i-1}P_{i+1})$

Notre algorithme de simplification du contour externe consiste à faire des suppressions successives de

petites aires en ne gardant à la fin que les sommets des coins (voir Fig.23c).

Algorithm 2: Algorithme de simplification du contour

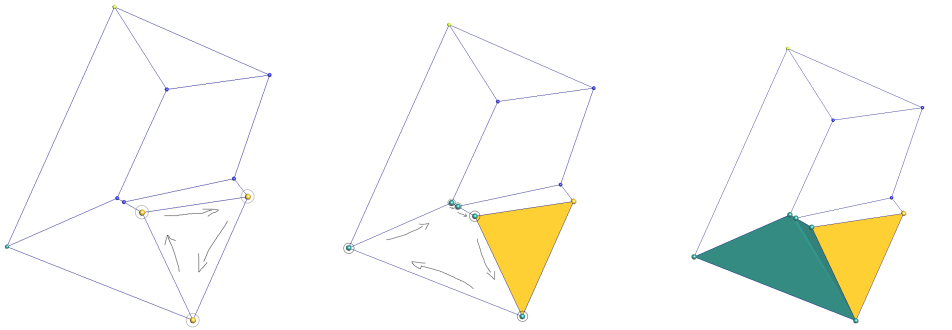
Input: $\mathscr{C}$: contour lissé
Result: Les points des coins du contour

```
1  coût = +∞ − 1
2  while coût < +∞_float do
3      foreach P_i do
4          A = P_{i−1} ; B = P_i ; C = P_{i+1} ;
5          if ÂBC est entre 45° et 135° then
6              tmp_coût = +∞
7          else
8              tmp_coût = aire(ABC)
9          end
10         if tmp_coût < coût then
11             coût = tmp_coût
12             j = i
13         end
14     end
15     Supprimer le point d'indexe j
16 end
```

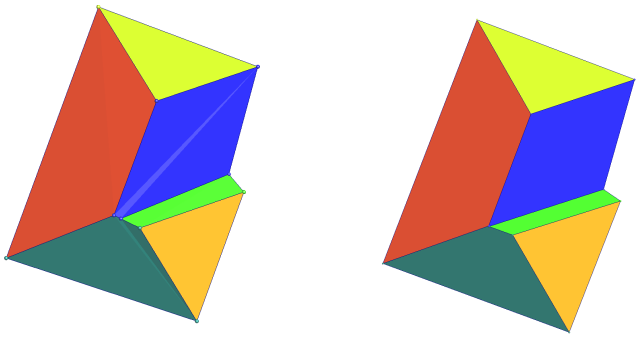
Algorithm 3: Algorithme de post-traitement sur le contour simplifié

Input: $\mathscr{C}$: contour simplifié d'un ensemble de points triés
Result: Garder un seul point dans un rayon $r = seuil$

```
1  i = 0
2  while i < taille(𝒞) do
3      d_1 = Distance(P_i, P_{i−1})
4      d_2 = Distance(P_i, P_{i−1})
5      if d_1 < seuil then
6          if P_i n'est pas un point d'intersection entre les plans then
7              Supprimer le point P_i
8          else
9              if P_{i−1} n'est pas un point d'intersection entre les plans then
10                 Supprimer le point P_{i−1}
11             end
12         end
13     end
14     if d_2 < seuil then
15         if P_i n'est pas un point d'intersection entre les plans then
16             Supprimer le point P_i
17         else
18             if P_{i+1} n'est pas un point d'intersection entre les plans then
19                 Supprimer le point P_{i+1}
20             end
21         end
22     end
23 end
```



(a) Graphe d'adjacence G. (b) Premier versant. (c) Deuxième versant.



(d) Fin de l'algorithme d'extraction des versants. (e) Résultat final, après raffinement de la géométrie.

FIGURE 24 – Modèle polygonal d'une toiture simple.

On constate après l'application de l'algorithme de simplification que les sommets qui sont proches d'un sommet important (intersection entre deux plans) ne seront pas supprimés, donc la simplification nécessite un post-traitement. Nous avons choisi de ne garder qu'un seul sommet dans un rayon r autour d'un sommet important, ce rayon r doit être inférieur à la longueur minimale que peut avoir un segment dans le contour. Sans cette condition on risque de supprimer un segment d'un polygone au lieu de juste le simplifier. Ci-dessous l'algorithme proposé pour régler ce problème dont le résultat peut être apprécié sur la Fig.23d.

3.3.3 Construction des polygones

Par la suite on assemble le contour extérieur simplifié et les segments intérieurs simplifiés dans un graphe d'adjacence. Ce graphe sera utilisé pour extraire les polygones selon leurs labels provenant de la segmentation, c'est-à-dire que chaque polygone prendra le label du pan qu'il représente. Ces labels sont stockés dans une liste de labels L.

L'algorithme commence par construire un graphe d'adjacence G avec les arêtes d'intersection entre les versants simplifiées et la bordure simplifiée (voir Fig.24a). Ensuite il choisit au hasard un sommet S_i du graphe G pour commencer l'extraction des polygones. Le nombre de polygones construit à partir du sommet S_i est égal au nombre de labels de ce sommet (nombre de plans dans lequel il appartient). Après l'extraction de tous les polygones à partir du sommet S_i, l'algorithme supprime les labels L_j utilisés de la liste L et passe au sommet suivant. Il termine quand il ne reste plus de polygones à extraire, c'est-à-dire tous les labels de la liste L ont été utilisés.

Algorithm 4: Algorithme d'extraction de polygones

Input: G : graphe d'adjacence des sommets de toit;
L : ensemble de labels du toit.
Result: Polygones labellisés

```
foreach Sommet S_i du graphe G do
    foreach Label L_j du sommet S_i do
        if L ≠ ∅ Et L_j ∈ L then
            Enlever L_j de L
            S' = S_i
            Point[] poly
            Ajouter(S_i, poly)
            Marquer le sommet S_i comme déjà visité
            Ajouter le sommet P_i au polygone labellisé avec L_j
            Sortir = faux
            while !Sortir do
                N_k = récupérer un voisin non marqué de S'
                if !visité(N_k) then
                    Ajouter(N_k, poly)
                    Marquer le point N_k comme déjà visité
                    S' = N_k
                end
                else
                    Sortir = vrai
                end
            end
            Export_polygone(poly)
            Démarquer tous les points
        end
    end
end
```

3.3.4 Raffinement de la géométrie

Pour finir, afin de raffiner le plongement géométrique (s'assuerer que les polygones sont bien plâts), on projette les sommets de chaque polygone sur les plans d'appartenance.

Projection des sommets des polygones sur leurs plans

Il y a trois cas possibles de projection d'un point (projeter sur un plan, deux plans ou trois plans), le choix de type de projection dépend de nombre de labels du point.

3.3.4.1 Cas d'un point qui appartient à un seul plan La projection orthogonale d'un point P sur un plan $\mathscr{P}$ est le point $M \in \mathscr{P}$ tel que la droite (PM) est perpendiculaire au plan $\mathscr{P}$.

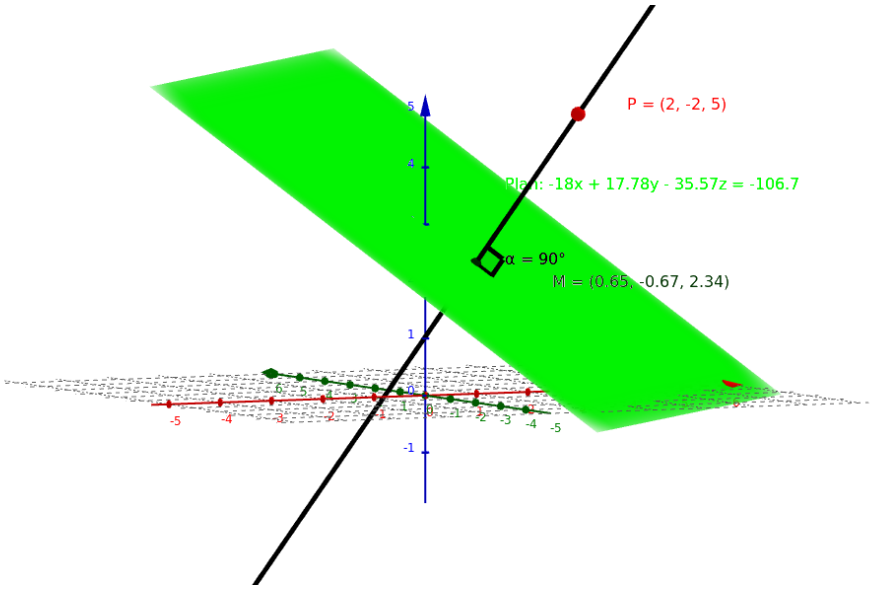


FIGURE 25 – Projection du point P sur le plan $\mathscr{P}$.

3.3.4.2 Cas d'un point qui appartient à deux plans La projection d'un point P sur deux plans $\mathscr{P}_1$ et $\mathscr{P}_2$ est le point d'intersection M des trois plans $\mathscr{P}_1$ $\mathscr{P}_2$ et $\mathscr{P}_3$ (plan perpendiculaire au deux plans $\mathscr{P}_1$ $\mathscr{P}_2$).

Trouver l'équation du plan $\mathscr{P}_3$:

$$a_3x + b_3y + c_3z + d_3 = 0$$

Pour trouver l'équation du plan $\mathscr{P}_3$ il suffit de trouver sa normale $\vec{n_3}$: (a_3, b_3, c_3)

Et sa distance : d_3

Calcul de la normale $\vec{n_3}$: Le plan $\mathscr{P}_3$ est perpendiculaire au deux autres plans $\mathscr{P}_1$ et $\mathscr{P}_2$ donc :

$$\vec{n_3} = \vec{n_1} \wedge \vec{n_2}$$

Calcul de la distance $\vec{d_3}$: le point P se trouve sur le plan $\mathscr{P}_3$ alors :

$$d_3 = -a_3x - b_3y - c_3z$$

La projection du point P sur les deux plans $\mathscr{P}_1$ et $\mathscr{P}_2$ est la solution du système d'équation suivant :

$$\begin{cases} a_1x + b_1y + c_1z + d_1 = 0 \\ a_2x + b_2y + c_2z + d_2 = 0 \\ a_3x + b_3y + c_3z + d_3 = 0 \end{cases}$$

3.3.4.3 Cas d'un point qui appartient à trois plans La projection d'un point P sur les trois plans $\mathscr{P}_1$ $\mathscr{P}_2$ et $\mathscr{P}_3$ c'est le d'intersection M de ces plans, pour trouver ce point d'intersection on résout le système d'équation suivant :

$$\begin{cases} a_1x + b_1y + c_1z + d_1 = 0 \\ a_2x + b_2y + c_2z + d_2 = 0 \\ a_3x + b_3y + c_3z + d_3 = 0 \end{cases}$$

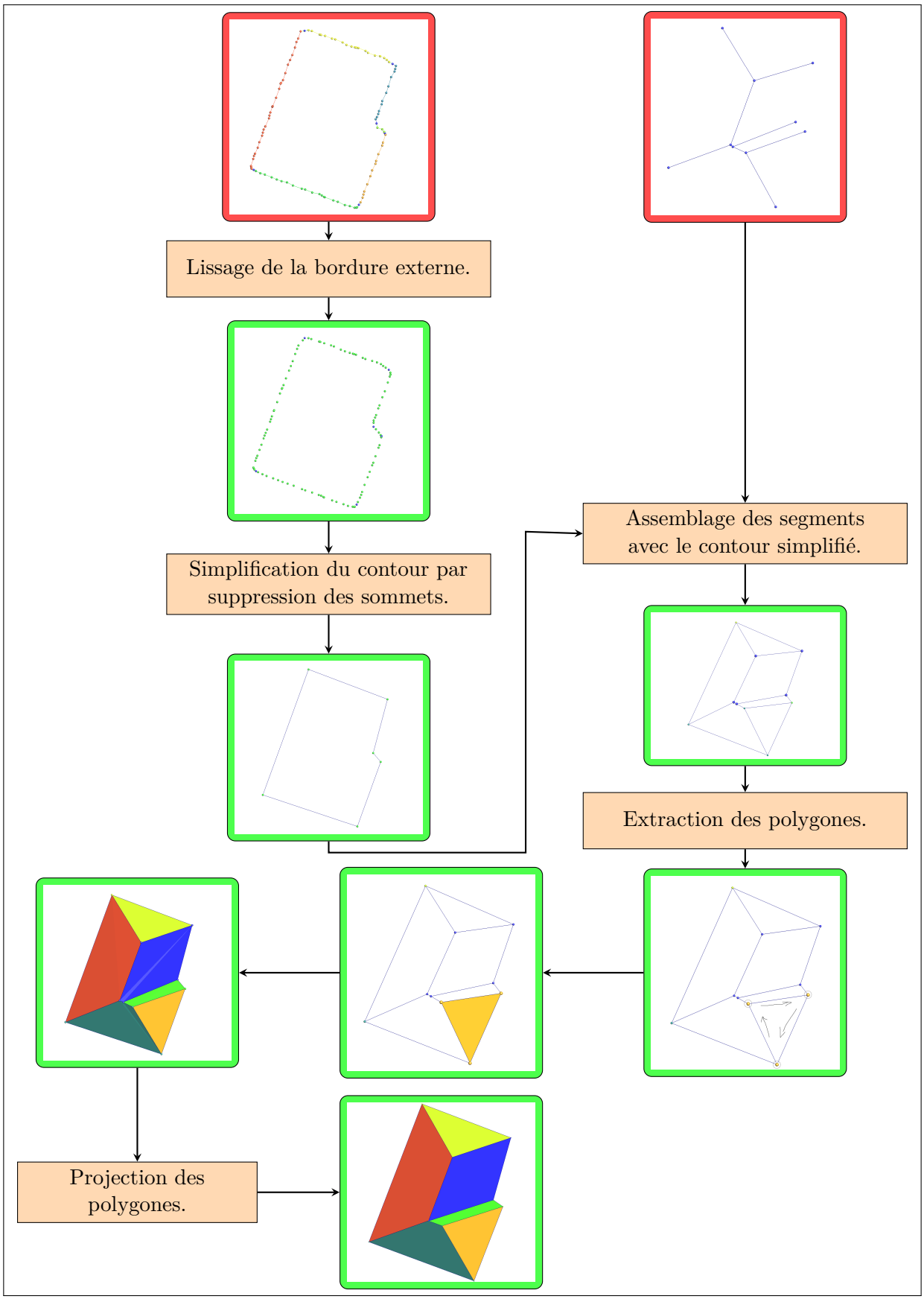


FIGURE 26 – Résumé de l'algorithme d'extraction des polygones correspondant aux versants d'une toiture.

4 Résultats et discussion

Avec la chaîne de traitements présentée dans ce rapport, avant la projection finale des polygones, on arrive à traiter 95% des toits de notre base de données. Les 5% restants sont des erreurs provenant de notre algorithme d'extraction des polygones à partir du graphe d'adjacence : soit l'algorithme n'arrive pas à revenir au sommet de départ (c'est-à-dire fermer le polygone en construction), soit il génère un polygone dégénéré.

En ajoutant la projection sur les résultats obtenus le taux de réussite descend à 68%. Pour les 32% restants les projections des sommets sont incorrectes. Ceci est typiquement causé par des incohérences d'intersections entre versants situés à des hauteurs différentes qui sont dues à la projection 2D utilisée pour la construction de l'α-shape. La Fig.27 montre une configuration où les plans 1 et 3 sont parallèles mais ne s'intersectent pas en 3D car il y a une différence en hauteur, alors que notre algorithme considère qu'il y a un point d'intersection entre les trois plans (1,2 et 3) ce qui provoque une projection erronée.

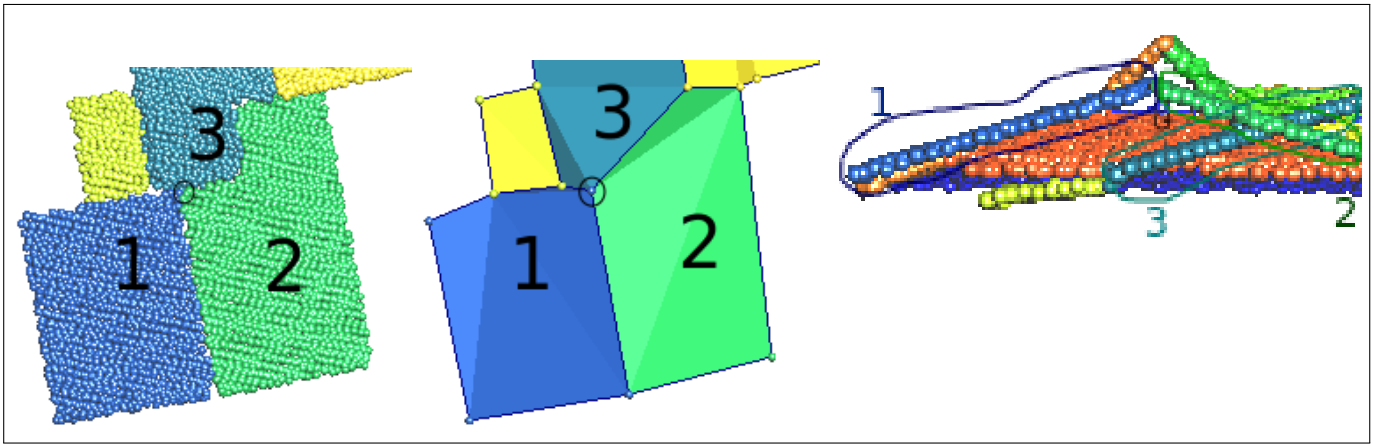


FIGURE 27 – Exemple d'intersection de trois plans qui donne une projection erronée.

5 Conclusion

Ce travail de recherche avait pour but l'automatisation d'une chaîne de traitements qui permet de passer d'un nuage de points 3D vers des modèles polygonaux de toitures. Ces modèles sont destinés à être utilisés pour la modélisation des bâtiments avec un détail LoD2 (les bâtiments sont modélisés à partir de leur structure de toit).

Notre chaîne de traitement est composée de trois étapes principales. La première consiste à segmenter le nuage de point global en composantes connexes selon la distance Euclidienne qui correspondent à des toitures simples ou de toitures complexes de plusieurs bâtiments adjacents. Ensuite, nous avons détecté les différents pans dans chaque toiture et avons trouvé l'équation de chacun des plans. Ensuite, nous avons détecté les bordures extérieures des toits ainsi que les segments d'intersections entre les différents versants qui ont été simplifiés afin de permettre la construction des polygones finaux correspondants aux toits. Nos résultats expérimentaux ont montré que la méthode permet de traiter correctement la majorité des toits de notre base de données. Dans l'avenir, nous nous concentrons, en particulier, sur le renforcement des contraintes géométriques telles que la planéarité des polygones, des faîtières parallèles au sol, les polygones englobant les points de la toiture, etc.

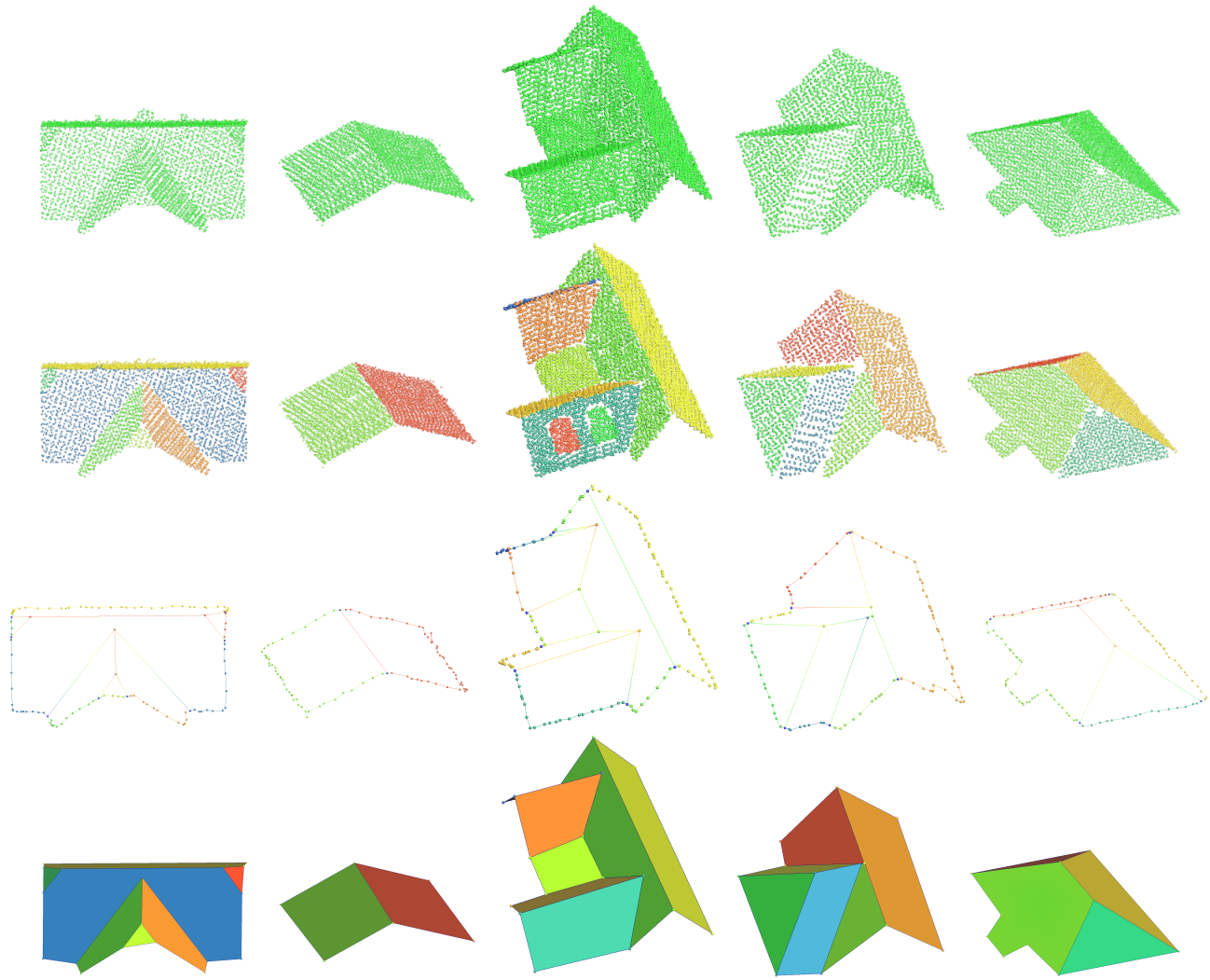


FIGURE 28 – Première ligne : composantes connexes de toitures. Deuxième ligne : segmentation en plans. Troisième ligne : graphe des arrêtes internes et les bordures. Dernière ligne : les résultats obtenus avec notre méthode.

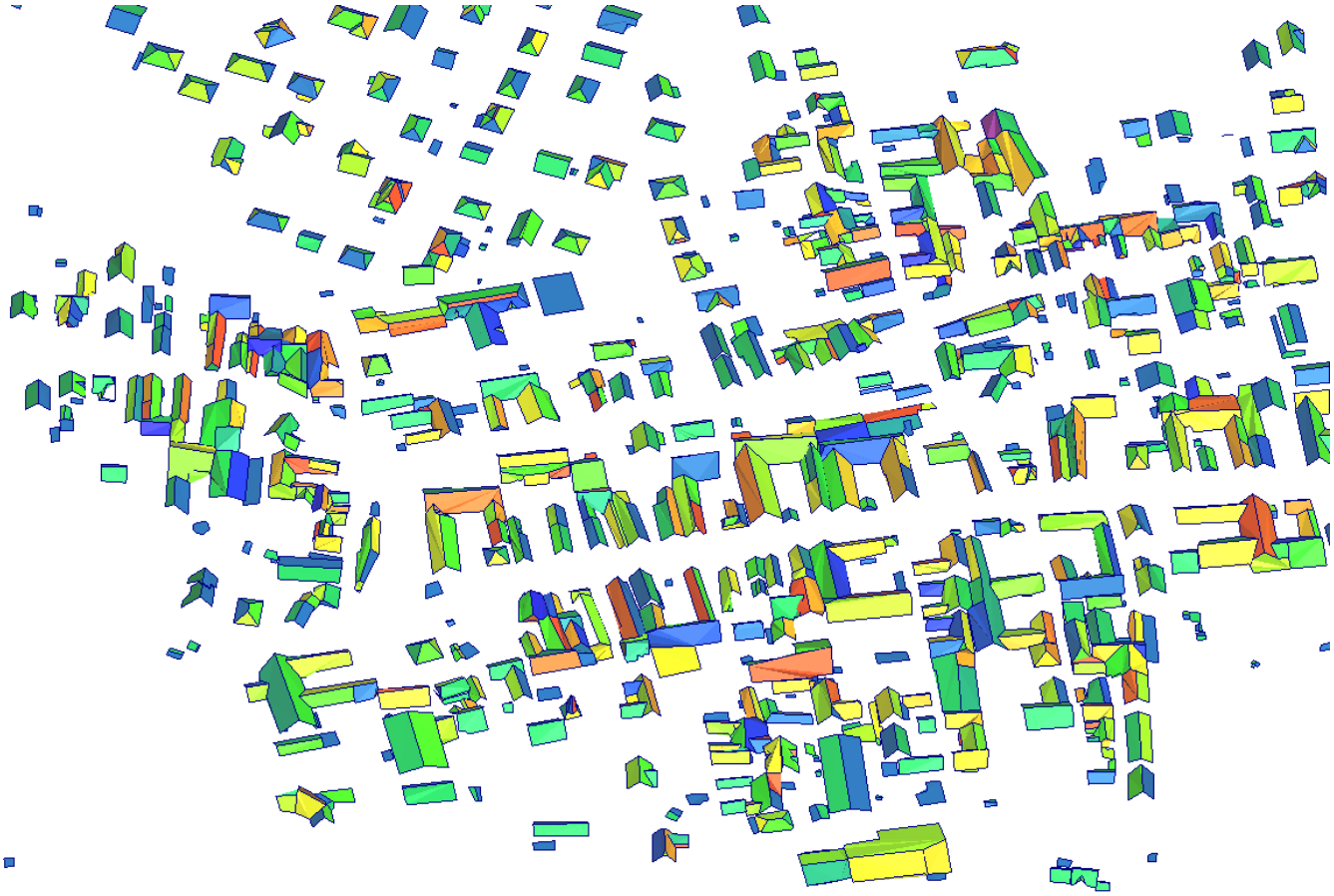


FIGURE 29 – Polygones construits automatiquement par notre méthode à partir des nuages de points de la figure 1.

Références

[1] D. BORRMANN, J. ELSEBERG, K. LINGEMANN, AND A. NÜCHTER, *The 3d hough transform for plane detection in point clouds : A review and a new accumulator design*, 3D Research, 2 (2011), p. 3, doi:10.1007/3DRes.02(2011)3, https://doi.org/10.1007/3DRes.02(2011)3.

[2] A. BOULCH AND R. MARLET, *Fast Normal Estimation for Point Clouds with Sharp Features using a Robust Randomized Hough Transform* , Computer Graphics Forum, 31 (2012), pp. 1765–1774, doi:10.1111/j.1467-8659.2012.03181.x, https://hal-enpc.archives-ouvertes.fr/hal-00732426. Proceedings of the 10th Symposium of on Geometry Processing (SGP 2012), Tallinn, Estonia, July 2012.

[3] R. CAO, Y. ZHANG, X. LIU, AND Z. ZHAO, *Roof plane extraction from airborne lidar point clouds*, International Journal of Remote Sensing, 38 (2017), pp. 3684–3703, doi:10.1080/01431161.2017.1302112, https://doi.org/10.1080/01431161.2017.1302112, arXiv:https://doi.org/10.1080/01431161.2017.1302112.

[4] J.-E. DESCHAUD AND F. GOULETTE, *A Fast and Accurate Plane Detection Algorithm for Large Noisy Point Clouds Using Filtered Normals and Voxel Growing*, in 3DPVT, Paris, France, May 2010, https://hal-mines-paristech.archives-ouvertes.fr/hal-01097361.

[5] D. H. DOUGLAS and T. K. PEUCKER, *Algorithms for the reduction of the number of points required to represent a digitized line or its caricature*, Cartographica : The International Journal for Geographic Information and Geovisualization, 10 (1973), pp. 112–222, doi:10.3138/FM57-6770-U75U-7727, https://doi.org/10.3138/FM57-6770-U75U-7727.

[6] R. O. Duda and P. E. Hart, *Use of the hough transformation to detect lines and curves in pictures*, Commun. ACM, 15 (1972), pp. 11–15, doi:10.1145/361237.361242, http://doi.acm.org/10.1145/361237.361242.

[7] H. Edelsbrunner, *Alpha shapes—a survey*, Tessellations in the Sciences, (2010).

[8] M. A. Fischler and R. C. Bolles, *Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography*, Commun. ACM, 24 (1981), pp. 381–395.

[9] M. A. Fischler and R. C. Bolles, *Random sample consensus : A paradigm for model fitting with applications to image analysis and automated cartography*, Commun. ACM, 24 (1981), pp. 381–395, doi:10.1145/358669.358692, http://doi.acm.org/10.1145/358669.358692.

[10] S. A. N. Gilani, M. Awrangjeb, and G. Lu, *Segmentation of airborne point cloud data for automatic building roof extraction*, GIScience & Remote Sensing, 55 (2018), pp. 63–89, doi:10.1080/15481603.2017.1361509, https://doi.org/10.1080/15481603.2017.1361509, arXiv:https://doi.org/10.1080/15481603.2017.1361509.

[11] P. Hough, *Method and means for recognizing complex patterns*, (1962).

[12] M. Kada and L. Mckinley, *3d building reconstruction from lidar based on a cell decomposition approach*, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 38 (2009).

[13] A. Kaiser, J. A. Y. Zepeda, and T. Boubekeur, *A survey of simple geometric primitives detection methods for captured 3d data*, Computer Graphics Forum, 38 (2019), pp. 167–196.

[14] F. Lafarge, X. Descombes, J. Zerubia, and M. Pierrot Deseilligny, *Structural approach for building reconstruction from a single DSM*, IEEE Trans. on Pattern Analysis and Machine Intelligence, 32 (2010), pp. 135–147.

[15] L. Li, F. Yang, H. Zhu, D. Li, Y. Li, and L. Tang, *An improved ransac for 3d point cloud plane segmentation based on normal distribution transformation cells*, Remote Sensing, 9 (2017), doi:10.3390/rs9050433.

[16] H. Maas and G. Vosselman, *Two algorithms for extracting building models from raw laser altimetry data*, ISPRS journal of photogrammetry and remote sensing, 54 (1999), pp. 153–163, doi:10.1016/S0924-2716(99)00004-0.

[17] E. Maltezos and C. Ioannidis, *Automatic extraction of building roof planes from airborne lidar data applying an extended 3d randomized houg5 transform*, ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, III-3 (2016), pp. 209–216, doi:10.5194/isprsannals-III-3-209-2016.

[18] W. Nguatem, M. Drauschke, and H. Mayer, *Roof Reconstruction from Point Clouds using Importance Sampling*, ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, II3 (2013), pp. 73–78, doi:10.5194/isprsannals-II-3-W3-73-2013.

[19] PCL. http://pointclouds.org/.

[20] S. Pu and G. Vosselman, *Automatic extraction of building features from terrestrial laser scanning*, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 36 (2012).

[21] R. Schnabel, R. Wahl, and R. Klein, *Efficient ransac for point-cloud shape detection*, Comput. Graph. Forum, 26 (2007), pp. 214–226, doi:10.1111/j.1467-8659.2007.01016.x.

[22] F. Tarsha-Kurdi, *Extraction et reconstruction de bâtiments en 3D à partir de relevés lidar aéroportés*, PhD thesis, Université de Strasbourg, 11 2008.

[23] V. Verma, R. Kumar, and S. Hsu, *3d building detection and modeling from aerial lidar data*, vol. 2, 02 2006, pp. 2213 – 2220, doi:10.1109/CVPR.2006.12.

[24] M. Visvalingam and J. D. Whyatt, *Line generalisation by repeated elimination of the smallest area*, `https://hull-repository.worktribe.com/output/459275`.

[25] G. Vosselman, *Building reconstruction using planar faces in very high density height data*, Int. Arch. Photogramm. Remot. Sens, 32 (2000).

[26] A. Wichmann, *Grammar-guided reconstruction of semantic 3D building models from airborne LiDAR data using half-space modeling*, PhD thesis, Université technique de Berlin, 2018.

[27] J. Yan, W. Jiang, and J. Shan, *Quality analysis on ransac-based roof facets extraction from airborne lidar data*, ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XXXIX-B3 (2012), pp. 367–372, doi:10.5194/isprsarchives-XXXIX-B3-367-2012.

[28] S. Yeon, C. Jun, H. Choi, J. Kang, Y. Yun, and N. Doh, *Robust-pca-based hierarchical plane extraction for application to geometric 3d indoor mapping*, Industrial Robot, 41 (2014), pp. 203–212, doi:10.1108/IR-04-2013-347.