Rapport de stage

Extraction de l'ensemble des murs d'une coupe de nuage de points de bâtiment

par

RICHARD Thomas

Résumé

Ce projet a pour objectif le développement d'une nouvelle fonctionnalité à ajouter au logiciel de traitement de nuage de points 3DReshaper. Plus précisément il s'agira d'améliorer la manipulation de nuages de bâtiments provenant de scanner laser ou de photogrammétrie. Ce genre de technologie est couramment utilisée par des corps de métier comme les géomètres-experts afin de mesurer l'entièreté d'un bâtiment très rapidement. L'une de leurs tâches les plus communes est le tracé de plan, pour ce genre de besoins le logiciel permet d'effectuer une coupe du nuage, puis de tracer manuellement des lignes sur les murs afin de servir de base à un dessin. Ces opérations sont longues, fastidieuses et potentiellement imprécises. L'objectif est d'automatiser ces étapes en concevant un algorithme d'extraction de lignes de l'ensemble des murs d'un bâtiment. Le problème est loin d'être trivial, nous avons donc découpé la méthode en sous-algorithmes répondant chacun à un problème spécifique. Un total de cinq algorithmes ont été développés. Tout d'abord le MarchingCylinder algorithme qui permet d'isoler tous les points d'un seul mur depuis un nuage de bâtiment complet. Le calcul de meilleure ligne qui permet d'obtenir la ligne correspondant au mieux au nuage ainsi extrait. L'extraction de section qui va utiliser le MarchingCylinder algorithme pour extraire non plus un, mais l'intégralité des murs d'un bâtiment. Le chaînage, qui va permettre de chaîner intelligemment les lignes entre elles pour améliorer la précision des angles. Et enfin l'optimisation de section qui va parfaitement réorienter chacune des lignes en fonction du nuage, afin de corriger les dernières imperfections. Finalement les résultats de la méthode ont été validés par le responsable qualité et une nouvelle fonctionnalité se basant sur l'algorithme sera prochainement incluse au logiciel.

Abstract

The main goal of this internship is to develop a new functionnality for the point cloud processing software 3DReshaper. This project is about improving building point cloud manipulation, from laser scanner device of photogrametry. It is common for fields like expert surveyor to use this type of technologies to efficiently measure a building. One of their most common task is to draw plans, for this kind of needs the software allows to perform a clipping plan of the building and manually draw each line. But these operations are slow, tedious and likely imprecise. The objective is to automatize the process by designing an algorithm able to extract lines from all walls of a building scan. The problem is pretty hard because of the noise, so we cut it out into small specific algorithms. A total of four algorithms have been developed. First the MarchingCylinder algorithm that can perform a segmentation of a building scan and extract only one wall. Then the best line computation that allows to extract a line from extracted cloud. Next we have section extraction, wich uses MarchingCylinder algorithm to extract all building lines. Now we have the chaining step, to increase angle precision. And finnaly section optimisation to perfectly rearrange each line in order to match with the cloud. The results were validate by quality experts, and a new functionnality based on this algorithm will be add soon.

EKHM51
Rapport stage de fin d'étude (15 crédits ECTS)
Juin 2019
Tuteur : David Wendland
Tuteur pédagogique : Raphaelle Chaine

Remerciements

Je souhaite avant tout remercier mon tuteur de stage, David Wendland, pour le temps qu'il a consacré à partager ses connaissances et son expérience, tout en m'accordant sa confiance et une large indépendance dans la conduite de cette recherche.

L'enseignement de qualité dispensé par le Master ID3D a également su nourrir mes réflexions et a représenté un socle de connaissances solide, merci donc aux enseignants-chercheurs.

Je remercie également mon professeur référent, Raphaelle Chaine, pour sa disponibilité et son implication. Ainsi que Gilles Monnier, le PDG de Technodigit, pour m'avoir accepté au sein de ses équipes.

Et enfin je voudrais exprimer ma reconnaissance envers les amis et collègues qui m'ont apporté leur soutien moral et intellectuel tout au long de ma démarche.

Table des matières

Table des figures

Chapitre 1

Introduction

Chaque jour des spécialistes réalisent des levés, créent des plans, construisent des routes, édifient des bâtiments et fabriquent des produits qui améliorent notre qualité de vie. Leur travail repose sur des mesures toujours plus précises et le recours à des outils de mesures numériques de pointe est de plus en plus fréquent. C'est dans cet environnement en constante évolution que les scanners laser 3D ont vu le jour. Scannage d'installations industrielles, génération de maquettes 3D "as built", calibrage de cuves à pression ou de tuyauteries, leurs applications sont innombrables. Cependant leur utilisation amène un défi technique d'un nouveau genre le traitement des nuages de points qu'ils engendrent. C'est à ce niveau que la société Technodigit se positionne, en fournissant une solution complète à la manipulation et à l'inspection de nuages de points via le logiciel 3DReshaper. Le projet décrit dans ce document vise à développer un algorithme sur lequel se basera une nouvelle fonctionnalité du programme.

L'un des corps de métier historiquement client de ce genre de technologie de mesure est le domaine des géomètres. La tâche la plus courante d'un géomètre expert est le dessin de plan intérieur afin de mener des travaux ou de définir des copropriétés. La réalisation de ce genre de document nécessite des relevés d'intérieurs et d'extérieurs précis. Avant l'utilisation de scanners lasers, les géomètres utilisaient diverses techniques manuelles. La trilatération par exemple, nécessite de mesurer la distance de toutes les diagonales d'une pièce ainsi que toutes les hauteurs et longueurs de murs. La tachéométrie est une autre méthode dans laquelle l'intégralité des angles doit être mesurée. Dans les deux cas, toutes les mesures sont reportées sur papier puis la pièce est retracé à la main. Ces méthodes sont longues, fastidieuses, et très peu automatisées donc sensibles aux erreurs humaines, voir figure 1.1. Afin de réaliser des relevés de façon bien plus fiable et efficace, les cabinets de géomètres modernes ont recours à des scanners lasers. Ils réalisent des scans de structures complètes qui sont ensuite importés puis analysés dans des logiciels tel que 3DReshaper. La prise de mesures est alors grandement accélérée et simplifiée, en revanche le passage de nuage de points à plan papier reste une étape relativement complexe. En effet il est nécessaire de couper le nuage selon un plan horizontal d'une certaine épaisseur, de le projeter sur un plan, puis de tracer manuellement les lignes une par une en estimant grossièrement leurs positions. Ces opérations restent plus rapides qu'un tracé classique, cependant la nature numérique des mesures rend l'automatisation de ces tâches envisageable. La nouvelle fonctionnalité à l'origine de ce projet répond exactement à ce besoin, il s'agira de concevoir un algorithme capable d'extraire de façon automatique l'ensemble des lignes d'une section de bâtiment. Ces lignes seront ensuite importées dans un logiciel de CAO tel qu'Autocad pour servir de base au dessin d'un plan. Cette extraction automatique n'est disponible chez aucun autre éditeur logiciel. Il existe un plugin Autocad développé par Leica Geosystems nommé CloudWorkx qui permet d'obtenir un résultat relativement équivalent. Cependant le processus n'est pas automatique, il est nécessaire de cliquer les deux extrémités de chacune des lignes que l'on souhaite extraire, l'algorithme qu'ils proposent vient simplement recaler légèrement la ligne en fonction du nuage. Notre approche permettra d'extraire l'intégralité des murs d'enceinte d'un bâtiment en un seul clic. Ce projet constitue donc une réelle innovation et une application concrète dans l'industrie.

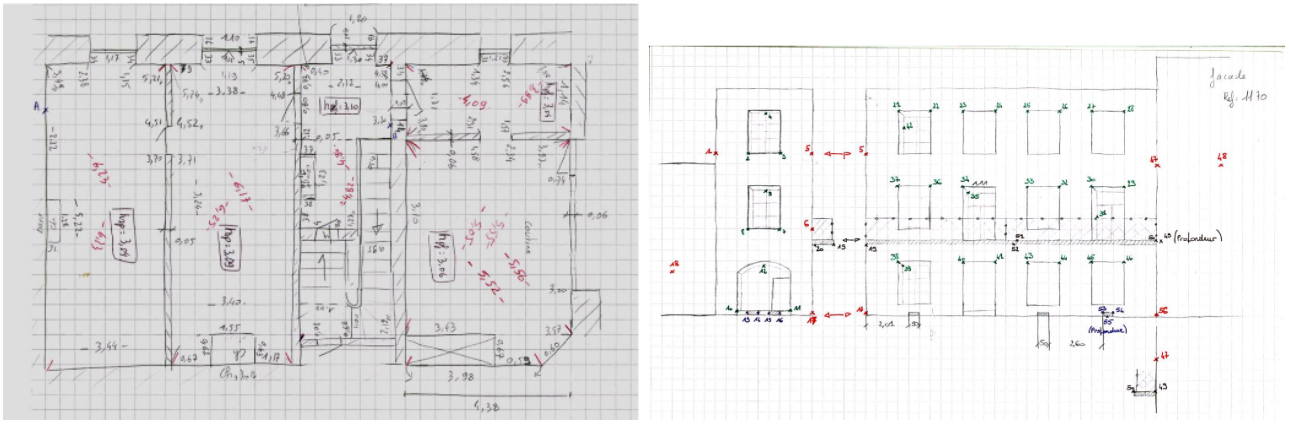


FIGURE *1.1 – Exemple de prise de mesure manuelle*

1.1 Environnement de travail

Le développement s'est déroulé sur une période de 6 mois au sein de l'équipe SDK de Technodigit. La méthode de travail a été identique à celle des autres développeurs, suivant une méthode Agile et reposant sur le gestionnaire de version Git. Le gestionnaire de ticket Jira et l'outil d'intégration continu BitBucket ont également été utilisés quotidiennement. Le code a été écrit en C++14 avec l'IDE Visual Studio 2017 et inclut dans la librairie complète comme n'importe quel développement, sur une branche dédiée. L'avancement a fait l'objet d'un suivi constant, des réunions régulières ont été organisées afin d'échanger sur les différentes idées, et le projet a été présenté au reste de l'entreprise tout au long de son développement durant des sprints démo.

1.2 Démarche

La création puis l'évaluation d'un algorithme est un processus itératif complexe, incluant entre autres des subdivisions en sous-problèmes et des pistes abandonnées. Ainsi cette section va décrire les différentes étapes de la conception ainsi que les réflexions qui les ont engendrées. Le reste du rapport présentera simplement le résultat final du travail de recherche.

Notre première approche a été une résolution globale du problème avec l'utilisation de l'algorithme Ransac, une méthode capable d'estimer les paramètres d'un modèle mathématique même dans le cas de données très bruitées. Malheureusement, l'algorithme s'est révélé très peu efficace lorsqu'un nuage contient plusieurs lignes, ce qui est toujours le cas pour un nuage de bâtiment. En effet l'intégralité des points appartenant aux lignes autres que celle que l'on souhaite extraire peuvent être considéré comme du bruit, c'est beaucoup trop pour un algorithme comme Ransac malgré sa forte robustesse. Nous avons donc abandonné cette piste et changé notre approche en procédant à un découpage en plusieurs sous-problèmes qui répondent chacun à une problématique bien spécifique. L'assemblage de tous ces algorithmes mènera à une résolution complète du problème.

Tout d'abord nous avons concentré nos efforts sur l'élaboration d'une méthode capable d'extraire la ligne d'un seul mur. Pour cela nous avons dégagé deux mécanismes importants, la segmentation du nuage pour extraire uniquement les points appartenant au mur choisi, puis le calcul d'une ligne depuis le nuage précédemment isolé. Pour le calcul nous avons développé une méthode analytique extrêmement performante, le calcul de best fit line, aussi appelé calcul de meilleure ligne. Il s'agit d'une méthode basée sur le concept mathématique de covariance permettant d'obtenir la ligne qui minimise sa distance avec tous les points d'un nuage. Pour la segmentation nous

avons conçu un tout nouvel algorithme baptisé MarchingCylinder algorithme. Cette méthode inspirée du region growing permet d'isoler tous les points appartenant à un mur grâce à une suite de sélections cylindriques successives. Le point de départ de l'algorithme est un point que l'on sait appartenant au mur et est appelé le germe.

Nous avons ensuite cherché à généraliser cette solution c'est-à-dire à extraire l'ensemble des murs et ainsi obtenir une suite de lignes chaînées, cet ensemble sera appelé une section de murs. Nous disposons d'une méthode capable d'extraire une ligne en partant d'un germe, l'extraction de section a donc simplement été conceptualisée comme la génération d'une suite de germes cohérents. Nous avons conçu un algorithme capable de parcourir l'ensemble des murs en effectuant une série de sélections à chacune de leurs extrémités afin d'estimer la position du mur suivant. Un germe est alors calculé et une ligne est ainsi extraite pour chaque mur.

Toutes les lignes sont à présent extraites, cependant le résultat est loin d'être satisfaisant. En effet l'utilisation du MarchingCylinder provoque des imprécisions au niveau des angles ce qui rend la section extraite discontinue. De plus l'extraction de section peut parfois manquer les plus petites lignes. Un algorithme a donc été ajouté afin de chaîner intelligemment les lignes entre elles de façon à corriger les imprécisions décrites précédemment. Cependant, le chaînage va modifier la position des extrémités ce qui peut altérer l'optimalité des lignes par rapport aux points du nuage.

Pour garantir des lignes parfaitement orientées nous avons donc ajouté une dernière étape permettant de replacer chaque ligne de façon optimale par rapport à son nuage tout en maintenant la contrainte d'extrémités chaînées. Une contrainte supplémentaire favorisant les angles droits a été ajouté de façon à corriger les éventuels défauts provenant des mesures.

Finalement nous avons conçu un total de cinq algorithmes qui comblent mutuellement leurs défauts. Leurs utilisations combinées permet une extraction automatisée de toutes les lignes des murs d'une section de bâtiment complet.

Chapitre 2

Extraction d'une ligne

L'extraction d'une ligne d'un nuage de bâtiment complet nous amène au développement de deux algorithmes complémentaires. Le premier est le calcul de meilleur ligne, qui permet de calculer une ligne depuis un nuage de points. Le deuxième est le MarchingCylinder algorithme, qui vient extraire tous les points appartenant à un même mur depuis un nuage de bâtiment complet.

2.1 Calcul de meilleure ligne

Ici l'objectif est simplement de calculer une ligne à partir d'un ensemble de points qui contient peu ou pas de bruit. Il existe de nombreuses manières d'obtenir une ligne depuis des points et presque autant de résultats différents. Dans notre cas nous avons choisi la ligne qui minimise sa distance au carré avec chacun des points. Nous l'appellerons la "meilleure ligne". Pour la calculer nous allons chercher la direction dans laquelle l'étalement des points est la plus forte.

2.1.1 Variance et matrice de covariance

Avant d'aborder les détails du calcul, nous allons présenté les notions de variance et covariance. La variance mesure la variation d'une variable aléatoire (comme le poids d'une personne dans une population), la covariance mesure à quel point deux variables varient entre elles (comme le poids d'une personne en fonction de sa taille). La covariance $\sigma(x, y)$ de deux variables x et y est donnée par la formule

$$\sigma(x, y) = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y}) \tag{2.1}$$

ou N est le nombre de points, $\bar{x}$ est la moyenne de la variable aléatoire x et $\bar{y}$ la moyenne de y. La matrice de covariance de la distribution des points du nuage peut ainsi être écrite comme

$$\sigma_{ij} = \frac{1}{N} \sum_{k=1}^{N} (p_i^k - \mu_i)\ (p_j^k - \mu_j) \tag{2.2}$$

avec

$$\mu_i = \frac{1}{N} \sum_{k=1}^{N} p_i^k \tag{2.3}$$

et N le nombre de points et les indices $i, j = 1, 2, 3$ représentent les directions x, y, z. Ainsi en étudiant la covariance des trois variables, nous allons pouvoir quantifier leurs écarts conjoints par rapport à leurs espérances respectives, autrement dit extraire l'orientation générale prise par le nuage. La variance d'une variable aléatoire x peut être exprimée comme la covariance d'elle-même $\sigma(x, x)$.

2.1.2 Valeurs propres

Nous savons à présent comment calculer la matrice de covariance, il nous faut maintenant en extraire la direction de la meilleure ligne. Pour cela nous allons nous servir de ses valeurs et vecteurs propres. Un vecteur $\mathbf{v}$ est l'un des vecteurs propres d'une matrice A si et seulement si il existe un scalaire λ tel que :

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v} \tag{2.4}$$

Dans notre cas, les valeurs propres de la matrice de covariance représentent une valeur d'étalement des points selon un axe, qui n'est autre que son vecteur propre associé. Ainsi le vecteur propre associé à la plus grande valeur propre indique la direction dans laquelle l'étalement des points est le plus fort. Notre meilleure ligne a donc pour direction ce vecteur propre et passe par la moyenne de tous les points.

2.1.3 Optimisation du temps de calcul

Le temps de calcul d'une meilleure ligne est critique de par son utilisation massive dans le MarchingCylinder algorithme, une méthode de segmentation centrale pour la suite. Plus précisément, l'algorithme va calculer de nombreuses meilleures lignes successives sur un nuage qui va croître progressivement. Ainsi pour chaque nuage il faudra recalculer une nouvelle matrice de covariance en reparcourant l'intégralité des points, alors qu'une grande partie d'entre eux a déjà fait l'objet d'un calcul durant le traitement du nuage de plus petite taille. Cela entraîne bien évidemment de très mauvaises performances. Nous allons donc cherché un moyen de reformuler le problème de façon à pouvoir effectuer un calcul cumulatif, ainsi il sera possible d'ajouter des points à une matrice de covariance, sans la recalculer dans son intégralité. De cette façon nous n'effectuerons qu'une seule opération par point. Pour conserver la matrice de covariance nous allons la réécrire de la manière suivante

$$\sigma_{ij} = \frac{1}{N} \underbrace{\sum_{k=0}^{n} p_i^k p_j^k}_{A_{ij}} - \mu_i \mu_j, \tag{2.5}$$

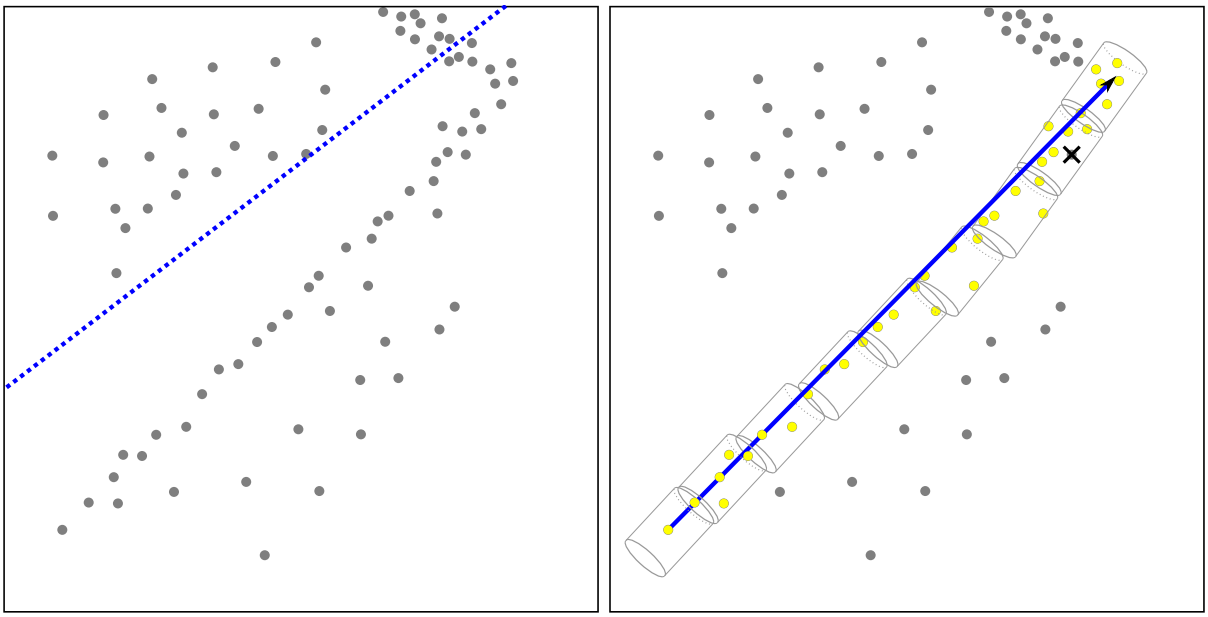
avec μ la moyenne des points. Ainsi, uniquement avec la matrice A_{ij} et la moyenne des points μ il est possible d'obtenir la matrice de covariance σ_{ij} de manière analytique, c'est-à-dire directement grâce à une formule ce qui est négligeable en temps de calcul. Donc s'il est possible d'ajouter des points à la matrice de variance et à la moyenne μ, il sera également possible d'ajouté des points à la matrice de covariance. On remarque que le calcul de la matrice de variance A_{ij} est une somme sur l'ensemble des points, y ajouter des points est donc trivial. Pour la moyenne, plutôt que de stocker directement la valeur, il suffira de conserver le nombre et la somme des points. Ainsi pour ajouter des points il suffira de les ajouter à la somme et d'incrémenter le nombre. Finalement notre nouvelle formulation permet donc bien d'ajouter des points à la matrice de covariance et ainsi de grandement accélérer le calcul.

2.2 Marching cylinder algorithme

Notes sur les appellations Afin d'alléger la notation, le mot cylindre est utilisé pour parler de sélection cylindrique, et les mots "droite" et "gauche" pour désigner deux directions opposées.

Nous disposons maintenant d'une méthode efficace pour extraire la meilleure ligne à partir d'un ensemble de points peu bruités. Cependant cette méthode n'est pas utilisable sur l'ensemble du nuage car la ligne serait alors ajustée selon l'ensemble des points et non selon les points du mur à extraire, voir figure 2.1. Une étape supplémentaire est donc nécessaire afin de segmenter un nuage de bâtiment en sous-nuages ne contenant qu'un seul mur.

Le MarchingCylinder est un algorithme conçu spécialement pour ce besoin et s'inspirant des algorithmes de *region growing*. La méthode segmente le nuage d'entrée en deux, le nuage interne et le nuage externe. À n'importe quel moment de l'algorithme tous les points contenus dans le nuage interne sont considérés comme appartenant au mur à extraire. À l'inverse tous les points du nuage externe n'appartiennent pas au mur. La méthode va étendre le nuage interne en y ajoutant les points de multiples sélections cylindriques sur le nuage externe. Ces sélections sont placées et orientées grâce à la meilleure ligne du nuage interne. Ainsi chaque sélection va apporter des points au nuage interne ce qui corrigera partiellement la direction de la prochaine sélection, l'algorithme va donc finir par converger vers une direction se rapprochant très fortement de la ligne à extraire. L'interaction entre chacun des cylindres rend l'orientation de la toute première sélection cylindrique cruciale car c'est elle qui détermine la direction générale de la ligne, de plus elle ne peut pas être estimée par une meilleure ligne car le nuage intérieur est alors vide. L'initialisation s'appuiera donc sur un point donné par l'utilisateur, et que l'on sait proche de la ligne à extraire. La longueur ainsi que le rayon des cylindres impactent très fortement le résultat de l'algorithme, ils seront donc eux aussi choisis par l'utilisateur.



(a) Meilleure ligne de l'ensemble du nuage (b) Résultat du MarchingCylinder algorithme

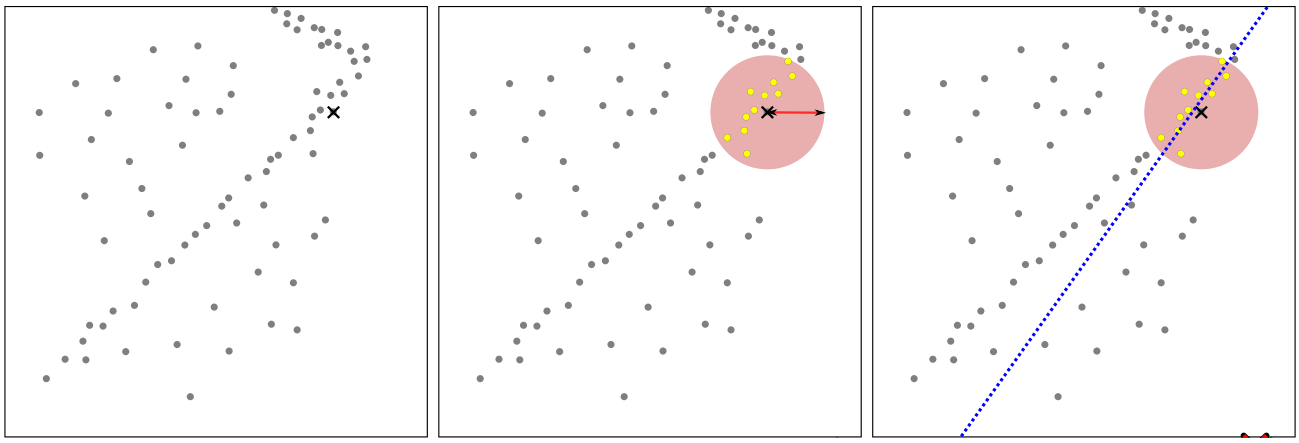
FIGURE *2.1 – Comparaison des résultats entre le MarchingCylinder algorithme et une méthode classique*

2.2.1 Initialisation

Comme expliqué précédemment, l'algorithme produit une chaîne de sélections cylindriques où chacune influe sur l'orientation de la suivante, c'est donc la première sélection qui a le plus d'impact sur l'orientation générale de la chaîne. Une première problématique est de calculer une sélection de départ qui initialisera le nuage interne et mènera à une approximation suffisamment proche de la ligne recherchée pour permettre aux sélections suivantes de corriger progressivement l'erreur.

Afin de résoudre le problème de l'orientation, la première sélection est sphérique et de rayon égal à la longueur d'un cylindre, ce qui équivaut à une sélection cylindrique dans toutes les directions. De cette façon si la position est bonne, la meilleure ligne des points contenus dans une telle sélection a de très fortes chances d'être suffisamment proche de la ligne recherchée. Une sphère a cependant plus de chances qu'un cylindre de sélectionner du bruit, mais cela à peu d'impact car le nombre d'outliers est généralement faible à proximité de la ligne à extraire.

Pour la position, le point de départ est cliqué par l'utilisateur. Ce choix s'intègre bien dans le workflow proposé pour l'intégration de l'algorithme car le nuage d'entrée peut contenir plusieurs lignes, le point servira donc aussi à choisir quelle ligne extraire. Pour maximiser la qualité du résultat, le point doit être le plus proche possible de la ligne et de préférence équidistant aux extrémités afin de laisser un maximum d'espace aux sélections pour corriger l'erreur, ces indications seront fournis à l'utilisateur.



(a) Clique du germe (b) Sélection sphérique (c) Calcul de la première meilleure ligne

FIGURE *2.2 – Initialisation du nuage interne*

2.2.2 Expansion

Le nuage interne est désormais initialisé, il s'agira maintenant d'augmenter sa taille intelligemment de façon à n'ajouter que des points appartenant à la ligne cherchée. Cette expansion se fait en trois étapes. Tout d'abord, on calcule la position et l'orientation de nouvelles sélections cylindriques supposées contenir des points cherchés, la meilleure ligne du nuage interne est utilisée comme guide. Ces sélections sont au nombre de deux, car l'algorithme se propage le long de la ligne des deux côtés du germe. Leurs contenus sont ensuite comparés, puis la meilleure des deux sélections est ajoutée au nuage interne. Le contenu du nuage interne est supposé appartenir au mur à extraire, calculer sa meilleure ligne permet d'obtenir une approximation suffisamment fiable pour orienter de futures sélections, qui viendront à leur tour améliorer la meilleure ligne. Une suite d'expansions de cette nature convergera donc vers une direction très proche de la ligne recherchée.

Calcul de nouveaux cylindres La première étape consiste à générer de nouveaux cylindres destinés à être ajoutés au nuage interne. Un cylindre doit être positionné de façon à ce qu'aucun point ne soit sélectionné deux fois, ou ne soit omis. La segmentation étant composée d'une chaîne de sélections cylindriques, la meilleure façon de positionner correctement un nouveau cylindre est de translater le centre de la dernière sélection ajoutée, et ce d'exactement la longueur d'un cylindre. La direction de la translation ne pourra cependant pas être la même que celle de l'ancien cylindre, car ce dernier a été orienté et positionné grâce à la meilleure ligne d'un nuage interne antérieur, et donc moins étendu. C'est dans cette différence d'orientation que réside le coeur de l'algorithme, car il s'agit de l'une des petites corrections qui vont successivement converger vers une orientation très proche de celle de la ligne cherchée. Le point translaté ne sera pas exactement le centre de l'ancien cylindre mais sa projection sur la nouvelle meilleure ligne, ce petit décalage permet de réaligner le centre dans un axe mieux orienté car issu d'une ligne plus récente, voir figure 2.3. Il est à noter que cette projection est négligeable si l'ancienne et la nouvelle meilleure ligne ont une orientation très proche. On applique ensuite la translation évoquée précédemment, puis il suffit de réorienter le cylindre en alignant son axe avec la nouvelle meilleure ligne pour obtenir une nouvelle sélection cylindrique correctement positionnée et avec une orientation à jour.

Le germe n'étant pas nécessairement positionné en bout de ligne, l'algorithme doit être en mesure de s'étendre des deux côtés. Les opérations de projections et de translations sont donc appliquées sur chacun des deux cylindres terminaux.

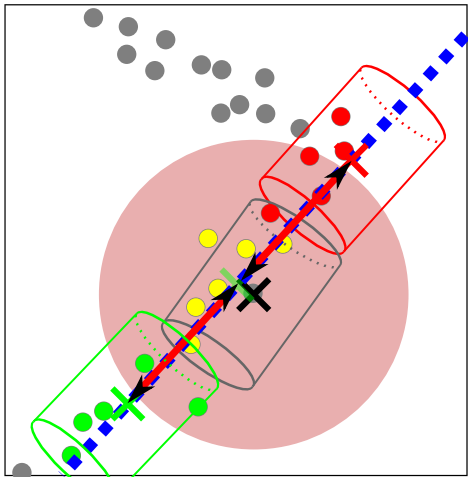


FIGURE *2.3 – Calcul de nouveaux cylindres*

Comparaison des cylindres Nous disposons maintenant de deux sélections cohérentes avec la meilleure ligne du nuage interne. On suppose que tous les points contenus dans ces sélections appartiennent au mur à extraire, cependant la fiabilité d'une telle supposition est directement corrélée à la fiabilité de la meilleure ligne, hors cette dernière peut se révéler très approximative, notamment durant les premières itérations où le nuage interne est incomplet. Pour limiter ce risque nous allons insérer uniquement la plus fiable des deux sélections. Pour quantifier la fiabilité on étudie la répartition des points dans une sélection, si leur étalement est particulièrement fort dans une direction c'est qu'ils sont disposés en forme de ligne et donc moins susceptibles d'être du bruit. L'étalement est calculé de la même manière que pour la meilleure ligne. On calcule d'abord la matrice de covariance des points du cylindre, puis ses valeurs et vecteurs propres associés. S'il y a un écart important entre la plus grande valeur propre et les deux autres, c'est que les points sont très étalés dans une direction et forment une ligne particulièrement marquée. Le cylindre avec le plus grand écart est donc considéré comme le meilleur.

Ajout du cylindre Pour ajouter le meilleur cylindre à la segmentation, on insère ses points dans le nuage interne et on ajoute sa matrice de covariance à la matrice de la meilleure ligne. Cette somme rend la comparaison de cylindres gratuite, car le calcul de covariance sera fait dans tous les cas lors du calcul de meilleure ligne intervenant plus tard. La validité de cette opération est permise par l'expression cumulative des matrices de covariances, expliquée précédemment dans la section 2.1.3. De plus, on rappelle que tous les points contenus dans le nuage externe n'appartiennent pas au mur à extraire, nous allons donc aussi retirer le contenu du cylindre du nuage externe.

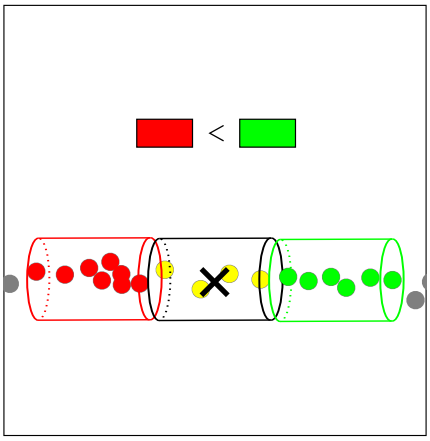


FIGURE *2.4 – Comparaison de cylindres*

Cas d'arrêt En itérant sur les deux étapes précédentes on peut progressivement étendre le nuage interne jusqu'à obtenir une meilleure ligne très proche de la ligne cherchée. On souhaite maintenant arrêter cette boucle lorsque toute la ligne a été extraite. La seule information dont dispose l'algorithme est un point de la ligne à extraire, le germe. Des données telles que la taille ou le nombre total de points de la ligne ne sont pas fournies et ne peuvent pas être estimées. En raison de cette absence de données, il est impossible de savoir si la ligne en cours d'extraction est complète. L'arrêt de l'algorithme se fera donc naturellement lorsqu'il est impossible de calculer de nouveaux cylindres, autrement dit lorsque les deux sélections terminales sont vides.

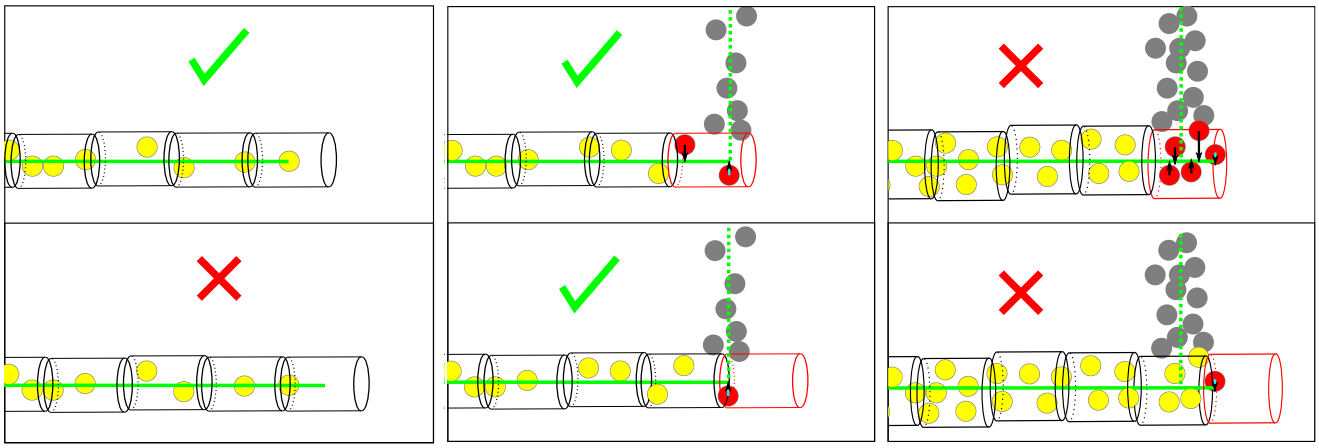
2.2.3 Finalisation

Les étapes précédentes nous ont permis d'isoler le nuage du mur à extraire ce qui remplit parfaitement l'objectif. Cependant une autre donnée nous est fournie par l'algorithme, la meilleure ligne du nuage extrait. Bien que son nom indique le contraire, cette ligne est en vérité une droite. En y plaçant intelligemment des extrémités, il devrait donc être possible d'en extraire une ligne finie, et ainsi de résoudre la quasi totalité du problème initial. Pour répondre à ce besoin trois méthodes de placements ont été imaginées.

Centre du dernier cylindre La segmentation étant extraite grâce à une chaîne de sélections cylindriques, la façon la plus naturelle de déterminer des extrémités est d'utiliser les centres des deux derniers cylindres contenant des points. Le résultat peut cependant être imprécis en fonction de la disposition des cylindres car seul le centre est utilisé, l'emplacement des points n'est pas pris en compte voir figure 2.5.a. L'un des avantages de cette méthode est l'absence de tout calcul car le centre du cylindre est déjà connu. En revanche la disposition des cylindres est directement dépendante du placement du germe , l'algorithme n'a donc aucun moyen de la modifier ce qui rend cette solution très instable.

Extrémité du nuage Ici l'idée est de trouver les deux points du nuage interne les plus éloignés du germe. Tout comme la solution précédente, nous allons travailler sur le contenu des dernières sélections, en revanche la disposition des points sera cette fois prise en compte. On projette l'ensemble des points sur la meilleure ligne, l'extrémité sera le point le plus éloigné du germe. Dans le cas de lignes fines le résultat est bien meilleur qu'en utilisant le centre des derniers cylindres, voir figure 2.5.b. Pour les lignes larges en revanche le résultat est imprécis car en prenant en compte l'épaisseur, l'extrémité ne peut pas se trouver à l'extrême bout du nuage, mais un peu avant.

Barycentre de la dernière sélection La dernière solution est de prendre le barycentre des dernières sélections. Si le cylindre final est bien positionné, le résultat est bon pour une ligne épaisse et pas trop mauvais pour une ligne fine. Pour un cylindre mal placé c'est l'inverse, le résultat est bon pour une ligne fine et plutôt mauvais pour une ligne épaisse. Selon la disposition des cylindres cette méthode peut donc potentiellement donner de bons résultats quelque soit la ligne, il s'agit d'un bon compromis. De plus, son coût est gratuit car le point a déjà été calculé durant le calcul de la meilleure ligne, voir section 2.1, il s'agit de μ.



(a) Utilisation du centre du cylindre (b) Utilisation du point le plus éloigné du germe

FIGURE *2.5 – Résultat des différentes méthodes de calcul d'extrémité*

Finalement ces trois méthodes fonctionnent dans une certaine configuration de cylindres et de tailles de lignes, mais aucune n'est générique. Il est impossible de maîtriser la disposition des cylindres car elle est directement dépendante du placement du germe, et ce dernier n'est pas modifiable. De la même façon, pour déterminer si le résultat est correct l'algorithme a besoin d'informations qu'il ne possède pas, à savoir la position de la ligne voisine et l'épaisseur du mur à extraire. Nous arrivons dans les limites du MarchingCylinder algorithme qui reste un algorithme de segmentation de nuage, le rôle d'extraction des lignes revient à deux algorithmes de post-traitements. Cependant ces deux derniers ont besoin de premières lignes pour pouvoir travailler, c'est pourquoi des lignes temporaires imprécises seront calculées. On ne souhaite donc plus un résultat parfait mais le résultat le moins mauvais possible. Dans ce contexte, c'est l'utilisation du barycentre des dernières sélections qui répond le mieux à notre besoin. Les lignes générées ont également un sens afin d'aider les algorithmes venant après ce dernier. Ainsi le point de début de la ligne sera le plus proche du germe, le point de fin sera logiquement le plus éloigné.

2.3 Résultats

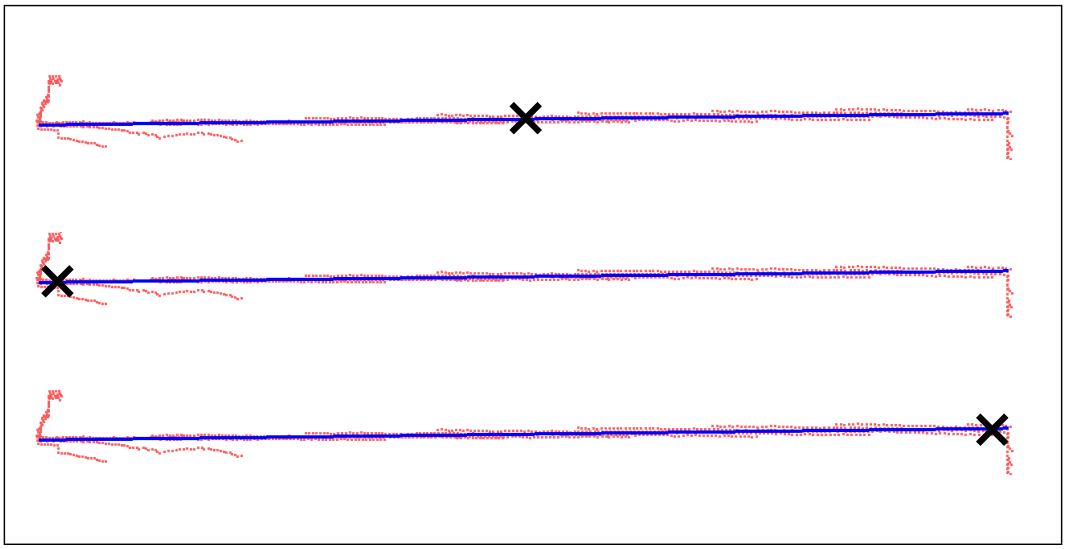


FIGURE *2.6 – Consistance du résultat selon le choix du germe*

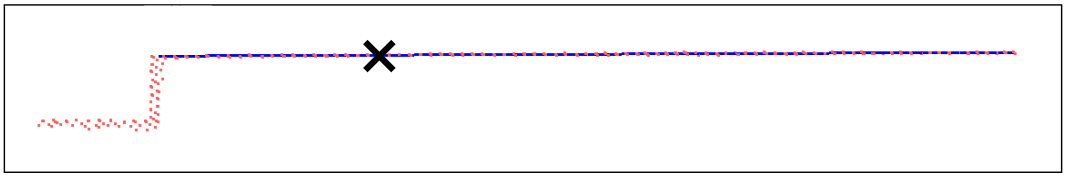


FIGURE *2.7 – Résultat sur nuage propre*

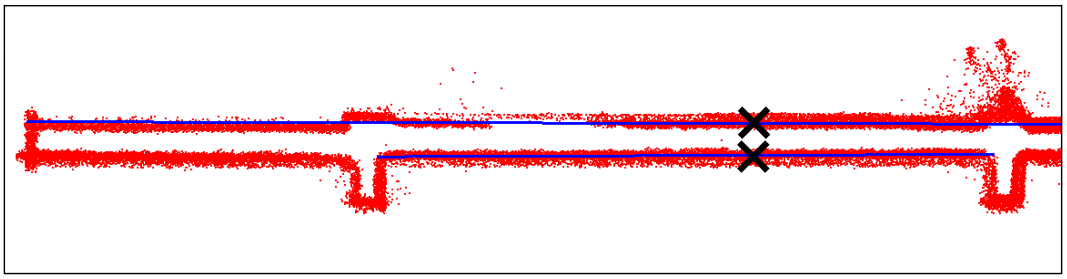


FIGURE *2.8 – Résultat sur nuage très bruité*

Voici une présentation des résultats de l'extraction de ligne unique pour différent types de nuages. Les croix noires indiquent quel point a été cliqué, autrement dit la position du germe. Les paramètres ont été adaptés en conséquence, par exemple le rayon du cylindre a été augmenté pour le nuage de la figure 2.8 car les lignes sont particulièrement épaisses.

Chapitre 3

Extraction de section

Nous sommes maintenant en mesure d'isoler le nuage d'un seul mur depuis le nuage d'un bâtiment complet. Nous souhaitons maintenant extraire automatiquement l'ensemble des murs d'enceinte, c'est-à-dire tous les murs formant des angles entre eux. Un tel ensemble de murs chaînés sera appelé une section de murs.

De plus, l'extraction de sections servira au placement précis des extrémités de la meilleure ligne de chaque nuage de murs. Bien que ce soit un autre algorithme qui réalisera cette opération, la section issue de l'extraction doit respecter certaines propriétés pour pouvoir être traitée correctement. Nous ajoutons donc deux nouvelles contraintes à notre définition de section, les lignes la composant doivent être ordonnées et correctement orientées entre elles. Autrement dit chaque ligne peut être vue comme une flèche, avec son point de début comme origine et son extrémité comme pointe. Pour qu'une section soit conforme, chaque ligne doit pointer vers le début de sa voisine, la première ligne n'est pointée par personne et la dernière ligne ne pointe personne. Ainsi tout l'enjeu de cet algorithme est de calculer une succession de nouveaux germes afin d'extraire une suite de lignes conformes à la définition de section décrite précédemment.

3.1 Recherche par sélections intelligentes

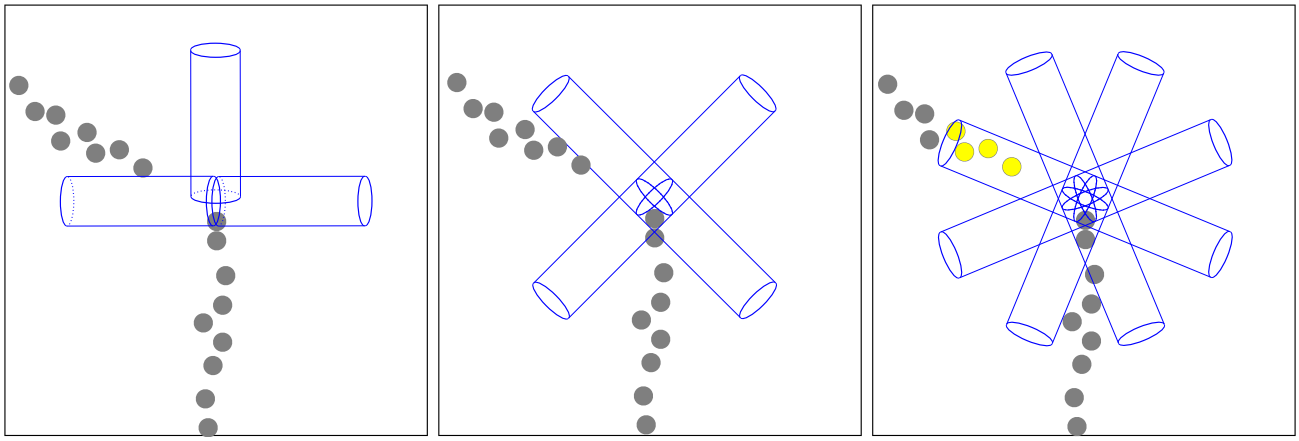
Comme expliqué précédemment l'objectif est ici d'extraire une section , autrement dit extraire un ensemble de lignes respectant une certaine disposition. Le MarchingCylinder algorithme nous permet d'obtenir une ligne imprécise à partir d'un germe, on peut donc réduire le problème de l'extraction de section au calcul d'une série de germes, ces derniers doivent cependant respecter certains critères afin de donner des lignes cohérentes. Tout d'abord il faut que les germes soient ordonnés de façon à ce que chacune des lignes qu'ils engendrent soit chaînée comme décrit dans la section . Chacun des germes devra donc se trouver relativement proche de la fin de la ligne précédente et du début de la ligne suivante. L'utilisation du MarchingCylinder ajoute deux propriétés supplémentaires aux germes qui influe sur la qualité des lignes extraites. Comme expliqué dans la section 2.2.1, le résultat est plus précis si le germe est équidistant aux extrémités et si la distance qui sépare le germe à sa ligne est petite. Finalement en prenant en compte toutes ces conditions, placer une série de germes aléatoirement à proximité de la fin de la dernière ligne ne sera pas suffisant. Il nous faut déterminer grossièrement la position de la prochaine ligne afin de placer correctement le nouveau germe.

On va simplement chercher la prochaine ligne avec une suite de sélections cylindriques possédant certaines propriétés. Tout d'abord leur origine est placée sur la fin de la dernière ligne afin de maximiser les chances de sélectionner la prochaine ligne de la chaîne plutôt qu'une ligne plus lointaine. Leur forme est cylindrique afin de limiter la sélection de bruit dans une direction. Leur rayon est le même que celui utilisé dans le MarchingCylinder algorithme, la longueur en revanche est un nouveau paramètre entré par l'utilisateur. Ce dernier pourra ainsi spécifier la distance maximale séparant deux lignes d'une même section. Une distance nulle va par exemple forcer l'extraction de lignes se touchant entre elles.

Il est à noter que chacune des sélections est effectuée sur les nuages externes des Marching-

Cylinder successifs. Autrement dit chaque ligne déjà extraite n'est plus prise en compte par les sélections ou les extractions, ainsi l'algorithme ne peut pas passer deux fois par la même ligne.

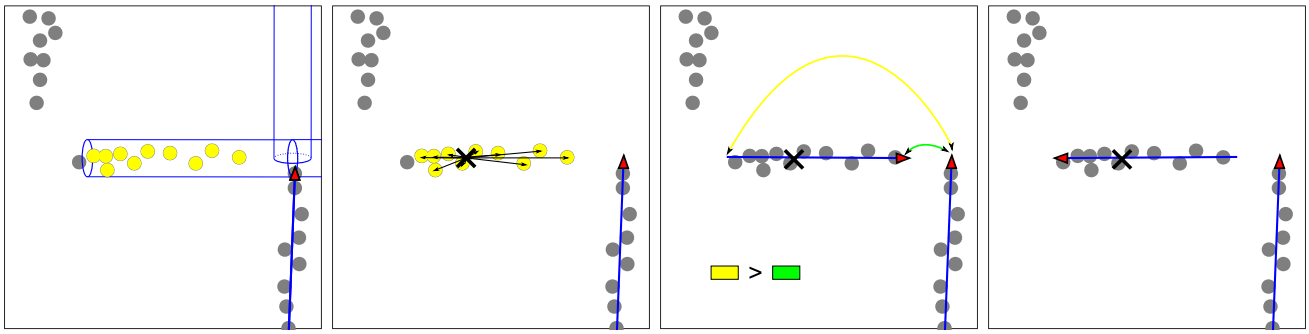
La direction de la sélection est quant à elle déterminée par l'algorithme de recherche. Ce dernier s'appuie sur une propriété commune à tous les bâtiments : l'angle entre deux murs est généralement proche d'un multiple de $\frac{\pi}{2}$, autrement dit proche d'un angle droit. On effectue donc dans un premier temps trois sélections cylindriques formant des angles communs avec la ligne précédente, à savoir $\frac{\pi}{2}$, π et $\frac{3\pi}{2}$. Il n'y a pas de sélection en 0 car il s'agit de la direction de la dernière ligne extraite. Comme les points déjà extraits ne sont plus pris en compte par la suite de l'algorithme, nous savons qu'il n'y aura pas de points. La sélection contenant le plus de points est considérée comme appartenant à la ligne cherchée, et est appelée la meilleure sélection. Si aucune sélection ne contient de points, ou si la ligne extraite est trop petite, c'est que la ligne ne se trouve pas dans des angles communs et n'a donc pas été détectée. Une deuxième série de sélections viendra rééchantillonner l'espace mais avec des angles multiple de $\frac{\pi}{4}$ donc plus serrés, les angles seront alors plus atypiques, $\frac{\pi}{4}$, $\frac{3\pi}{4}$, $\frac{5\pi}{4}$ et $\frac{7\pi}{4}$. Si toutes les sélections sont encore vides ou non valides on vient ajouter une dernière étape avec huit nouvelles sélections qui viennent combler les derniers angles morts, $\frac{\pi}{8}$, $\frac{3\pi}{8}$, $\frac{5\pi}{8}$, $\frac{7\pi}{8}$, $\frac{9\pi}{8}$, $\frac{11\pi}{8}$, $\frac{13\pi}{8}$, $\frac{15\pi}{8}$. Le prochain germe sera le barycentre de la meilleure sélection, on obtient ainsi un point proche de la ligne même en cas de sélection de bruit. De plus la sélection peut contenir une partie plus ou moins longue de la ligne, calculer un barycentre va donc mener à une position possiblement éloignée de l'extrémité et donc plus équidistant. Cet ensemble de trois séries de sélections menant à la génération d'un germe est appelé "sélections intelligentes".



(a) Première série (b) Deuxième série (c) Troisième série

FIGURE *3.1 – Recherche de la prochaine ligne durant une extraction de section*

On rappelle que deux lignes d'une section valide doivent être correctement orientées, c'est-à-dire que la fin d'une ligne doit être plus proche du début que de la fin de la ligne suivante. Hors comme expliqué dans la section 2.2.3, une ligne issue du MarchingCylinder algorithme est orientée de façon à ce que le début de la ligne soit l'extrémité la plus proche du germe et la fin la plus éloignée. On peut en déduire que pour extraire une section valide, un germe doit être plus proche de la fin de la ligne précédente que de la fin de la ligne qu'il doit extraire, hors les sélections intelligentes ne garantissent pas ce comportement. Le problème est donc corrigé après l'extraction, on calcule alors deux distances, celle entre le germe et la fin de la ligne extraite, et celle entre le germe et la fin de la ligne précédente. Si le germe est trop avancé sur la ligne extraite, et donc trop éloigné de la fin de la ligne précédente, on retourne la ligne extraite, voir figure 3.2.

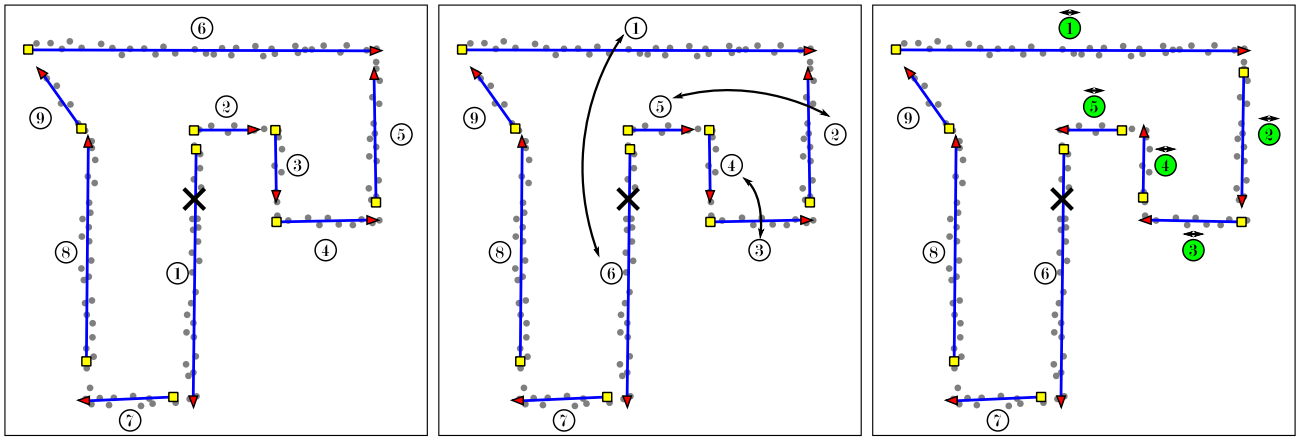


(a) Détection d'une ligne (b) germe trop proche de la fin de la ligne à extraire (c) Comparaison des distances (d) On retourne la ligne extraite

FIGURE *3.2 – Cas particulier de germe trop proche de la fin de la ligne*

3.2 Ordre et sens

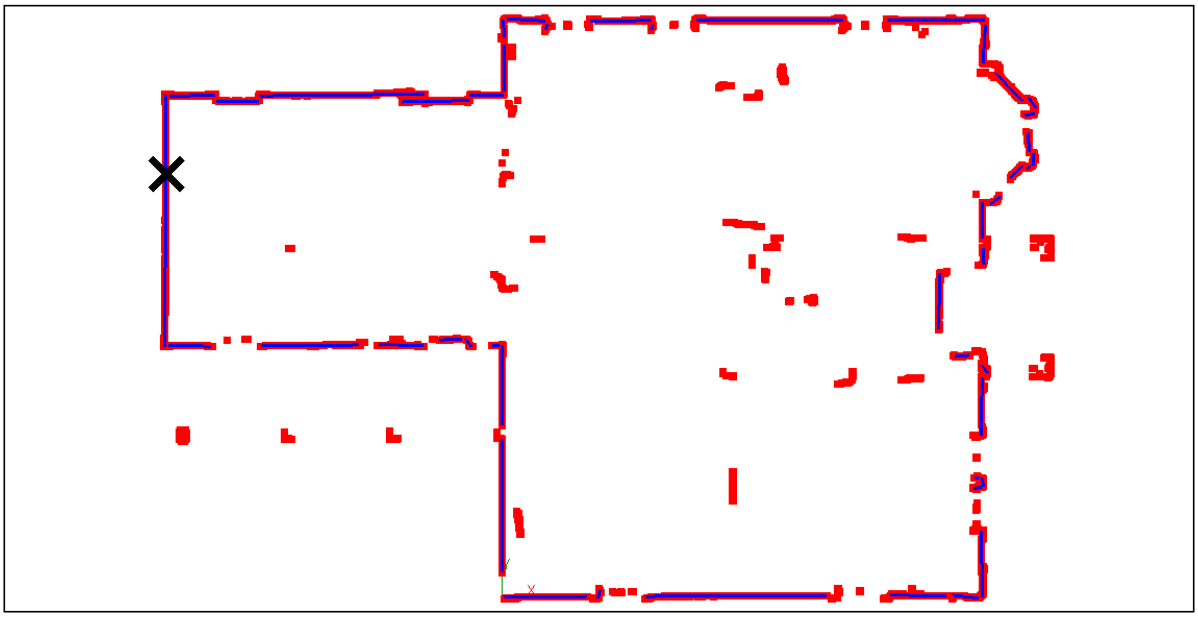
Les lignes d'une section valide doivent être ordonnées de façon à ce que chaque ligne ait comme voisine les lignes avec lesquelles elles partagent un angle, ainsi les premières et dernières lignes ne partagent qu'un seul angle avec une autre ligne. L'algorithme d'extraction de section va générer des germes à partir de sélections à proximité de la fin de la dernière ligne, l'ordre des lignes est donc garanti. Cependant tout comme le MarchingCylinder algorithme la ligne issue du tout premier germe n'est pas nécessairement la ligne en bout de chaîne, l'algorithme doit donc pouvoir extraire à gauche et à droite du premier germe. Pour se faire deux sections sont extraites, à gauche et à droite de la première ligne, puis concaténées pour ne faire qu'une section. L'ordre et le sens sont donc garantis mais indépendamment entre chaque section, comme leur direction de propagation est inversée, leur ordre et sens le sont aussi. Pour obtenir un résultat final ordonné on inverse l'ordre de toutes les lignes à "gauche" du germe, de même que leurs positions de début et de fin, voir fig 3.3.



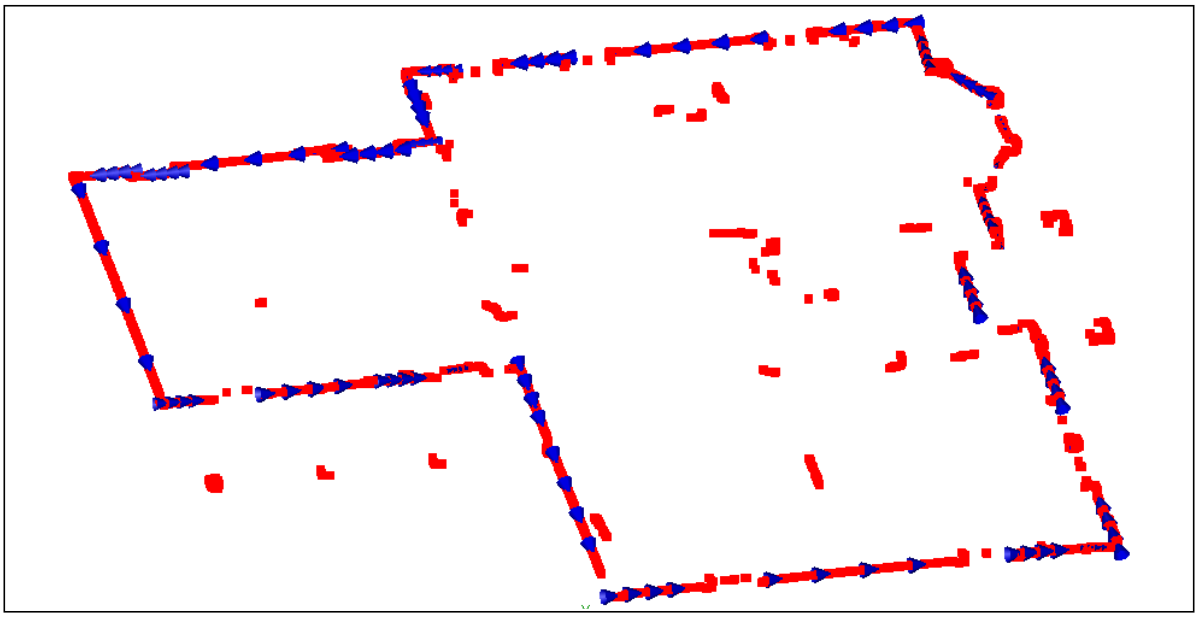
(a) Résultat avant réorientation (b) Inversion des indices des lignes de "droite" (c) Inversion du sens des lignes de "droite"

FIGURE *3.3 – Changement de l'ordre et de l'orientation des lignes*

3.3 Résultats



(a) Section extraite



(b) Orientation des lignes

FIGURE *3.4 – Résultat sur nuage fortement bruité*

On peut voir que l'intégralité des lignes ont été extraites, voir figure 3.4 (b). De plus elles sont correctement orientées, comme l'atteste les indicateurs d'orientation de lignes de 3DReshaper, voir (c) et (d).

Chapitre 4

Chaînage des lignes

Nous avons à présent une section correctement orientée, en revanche les extrémités des lignes sont imprécises car toujours issues d'une estimation du MarchingCylinder algorithme, comme expliqué dans la section 2.2.3. Le chaînage des lignes est l'une des deux étapes permettant d'améliorer la précision des extrémités. En effet comme nous savons que deux murs adjacents forment un angle de bâtiment, il suffit de modifier la position des extrémités de façon à ce que toutes les lignes soient chaînées pour améliorer la précision des lignes. Cette opération ne peut être effectuée qu'après l'extraction de section, car la position de l'intégralité des lignes est nécessaire. De plus il est nécessaire de connaître les voisins de chaque ligne pour pouvoir chaîner les bonnes extrémités, c'est la raison pour laquelle l'extraction de section doit fournir des lignes correctement ordonnées et orientées.

4.1 Classification des angles

Il a été dit précédemment que deux murs adjacents forment un angle de bâtiment. Cette affirmation est correcte, en revanche il faut rappeler que nous ne travaillons pas directement sur des murs, mais sur des lignes extraites de murs par l'algorithme de sélection intelligente. Ainsi il est possible que deux lignes adjacentes dans une section ne forment pas un angle de bâtiment, mais soient simplement reliées par une petite ligne omise durant l'extraction de section.

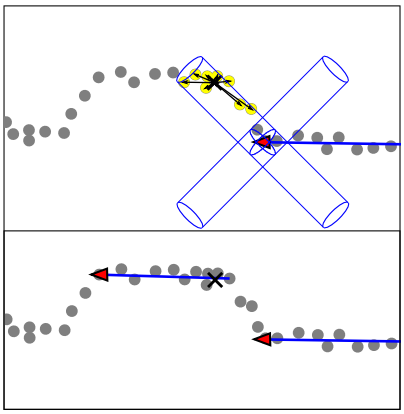
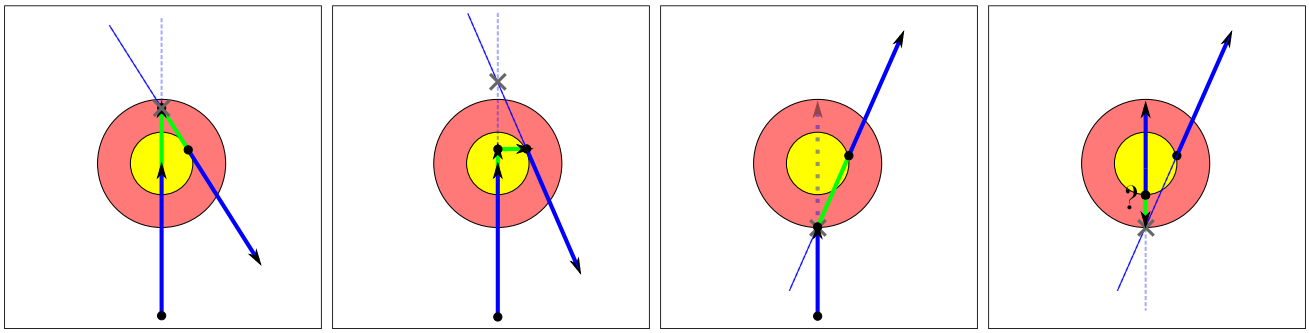


FIGURE *4.1 – Configuration dans laquel une ligne est manqué*

L'algorithme de chaînage va donc parcourir les lignes deux par deux, déterminer si elles forment un angle imprécis ou s'il manque une ligne intermédiaire, puis les chaîner en conséquence. Pour connaître la nature de l'interaction entre deux lignes nous allons utiliser leur co-linéarité. On considérera que deux lignes grossièrement parallèles ne forment pas un angle, et vice-versa. Cependant ce critère n'est pas suffisant, car nous souhaitons y ajouter une notion de distance. En effet deux lignes aux extrémités très éloignées ont plus de chances de cacher une ligne manquante plutôt que de former un angle imprécis, et ce même si elles forment grossièrement un angle droit. Pour inclure ce nouveau critère, nous allons d'abord calculer le point d'intersection entre les deux lignes. Si la distance entre cette intersection et la fin de la première ligne est supérieure au double d'une valeur seuil, alors on considère qu'il manque une ligne, sinon c'est qu'il s'agit d'un manque

de précision. Avec cette méthode si une ligne est très éloignée et forme un angle droit grossier, l'intersection sera trop éloignée et on considérera qu'il manque une ligne. En revanche si l'angle est presque parfaitement droit, alors l'intersection sera suffisamment proche et il s'agira bien d'un angle imprécis.



(a) Intersection à portée : nouvel angle

(b) Intersection hors de portée : nouvelle ligne

(c) Intersection à portée : nouvel angle

(d) Cas de retournement

FIGURE *4.2 – Classification des angles*

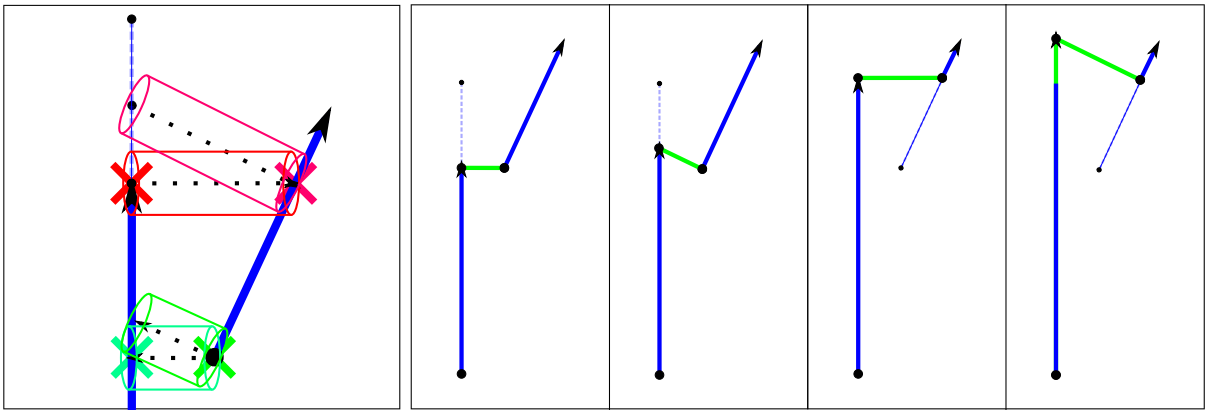
4.2 Chaînage par intersection

Si un manque de précision est détecté on effectue un chaînage par intersection, il suffit alors de placer la fin de la première ligne et le début de la seconde à l'intersection précédemment calculée. Cependant, si l'intersection est plus proche du début de la première ligne que de sa fin, cela signifie que la nouvelle ligne sera dans une direction opposé à l'ancienne, voir figure 4.2.d. Dans ce cas, on effectue un chaînage par insertion plutôt qu'une intersection.

4.3 Chaînage par insertion

On effectue un chaînage dit par insertion lorsque deux lignes forment un angle trop atypique pour être un angle de bâtiment, on en déduit qu'il manque une ligne. Il faut alors détecté sa position afin de l'insérer dans la section. Cette opération n'est pas triviale car cela signifie que ni le MarchingCylinderalgorithme, ni les sélections intelligentes ne sont parvenus à extraire cette ligne manquante, il s'agit probablement d'une ligne avec beaucoup de bruits, et/ou très petite, et/ou une incohérence du fichier. Il est néanmoins possible de déduire sa position en fonction de la disposition des deux lignes. On va différencier deux grandes configurations, les lignes qui se chevauche et celles qui ne ce chevauchent pas. Si il n'y a pas de chevauchement, on considère que la ligne manquante se trouve entre la fin de la ligne courante et le début de la ligne suivante, car c'est à cette position qu'elle a le plus de chance de se trouver. En effet, les angles entre une nouvelle ligne placée à cette position et ses voisines sont entre $\frac{\pi}{2}$ et $\frac{3\pi}{2}$, autrement dit des angles graves, très courant dans des bâtiments. En cas de chevauchement c'est l'inverse, on a des angles aigus très rare dans des bâtiments, dans ce cas on considère que l'une des deux lignes est trop longue. Il y alors quatre possibilités, décrite dans le schéma 4.3.b. Pour détecter dans quelle configuration se trouve la ligne manquante on effectue une sélection pour chacune des possibilités, si l'une d'elle contient des points c'est qu'on a détecté la ligne manquante, voir figure 4.3.a. On utilise alors ces points comme nouvelle ligne, que l'on viendra insérer dans la section. On choisira les deux extrémités de la sélection cylindrique comme extrémités pour la nouvelle ligne. Ce choix est peu

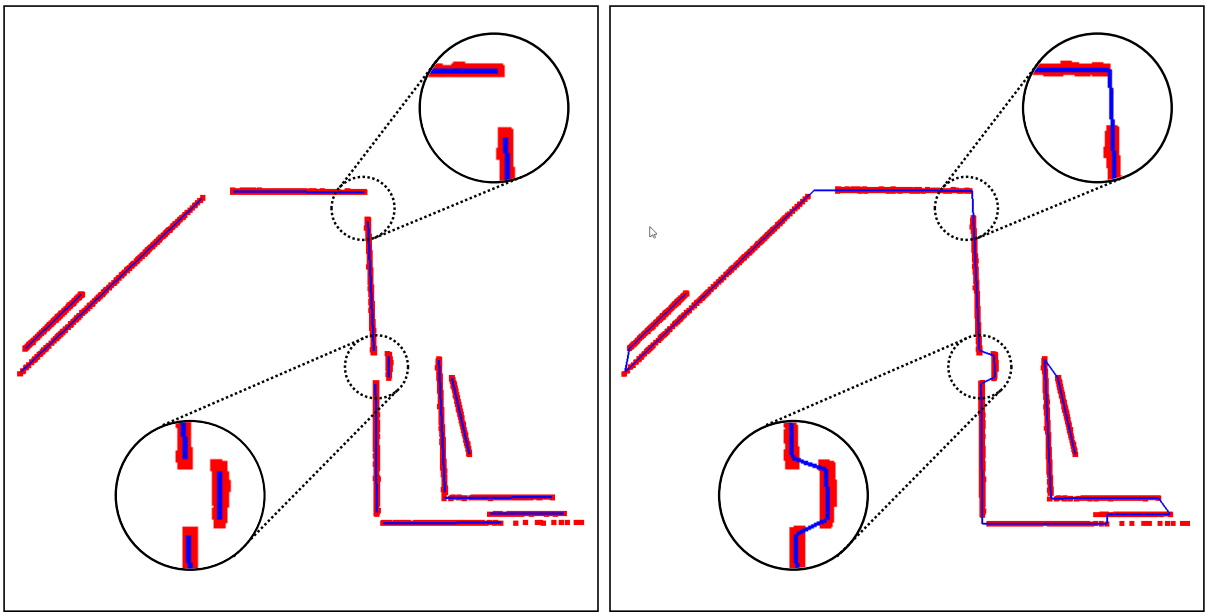
précis, mais cela n'a pas d'importance car l'algorithme d'optimisation après le chaînage viendra replacer les lignes en cohérence avec leur segmentation.



(a) Résultats possibles (b) Résultats possibles

FIGURE *4.3 – Chaînage possible en cas de chevauchement*

4.4 Résultats



(a) Avant chaînage (b) Après chaînage

FIGURE *4.4 – Résultat sur nuage synthétique peu bruité*

Le nuage utilisé pour la présentation de ces résultats est un nuage de test construit manuellement pour obtenir des angles particulièrement atypiques. On peut voir que le chaînage s'effectue correctement sans incohérence. Un chaînage par insertion d'une nouvelle ligne est effectué en cas de lignes parallèles, voir en bas à gauche de la figure 4.4. Un chaînage par intersection est effectué en cas d'angle de bâtiment détecté, voir en haut à droite de la figure 4.4, la ligne la plus courte est alors allongée.

Chapitre 5

Optimisation de sections

La méthode de chaînage décrite précédemment modifie la segmentation et fait perdre la garantie d'avoir la ligne qui minimise sa distance à tous les points. Pour corriger ce problème on ajoute une toute dernière étape de post-traitement qui vient repositionner les lignes par rapport aux points.

L'algorithme va également tenter de former un maximum d'angles droits entre les murs, tout en maintenant la proximité avec les points et le chaînage des lignes, ce qui viendra encore améliorer le résultat en lissant les potentielles imperfections.

5.1 Méthode de rejet

On souhaite calculer une section composée de lignes chaînées entre elles et optimales à leur segmentation respective, cependant aucun algorithme à notre disposition n'est capable de combiner ces deux propriétés. Le calcul de meilleure ligne permet d'obtenir des lignes optimales mais pas nécessairement chaînées, et vice-versa pour le chaînage. Concevoir un nouveau calcul capable de prendre en compte les deux contraintes serait trop complexe, nous allons donc aborder le problème sous un autre angle. Nous allons plutôt calculer l'énergie de la section, c'est-à-dire son éloignement avec la configuration désirée, puis nous allons chercher à la réduire. Par exemple si on souhaite que les angles soient les plus plats possible, un section très anguleuse va avoir une très forte énergie. Pour réduire cette énergie nous allons utiliser une méthode de rejet. Il s'agit d'un algorithme simple composé d'une simple boucle. À chaque itération une section candidate aléatoire est générée, puis son énergie est comparée avec celle de la meilleure section, si elle est plus faible alors la section candidate devient la nouvelle meilleure section. Ainsi l'algorithme va faire baisser progressivement l'énergie de la meilleure section, autrement dit réduire son éloignement avec la configuration cherchée. Les deux étapes déterminantes de l'algorithme de rejet sont donc la génération d'une section candidate ainsi que la formalisation de la configuration cherchée, c'est-à-dire le calcul de l'énergie.

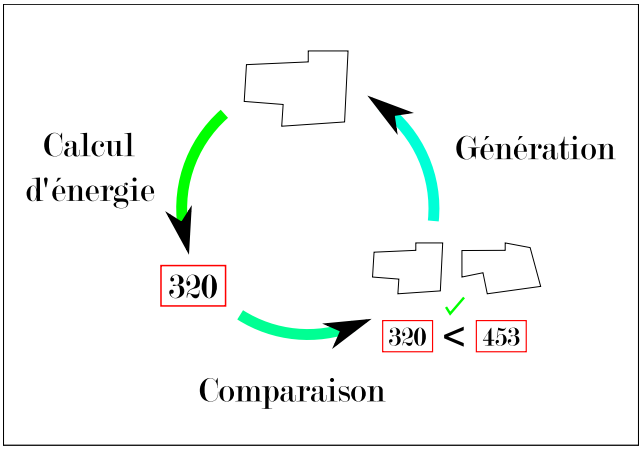


FIGURE *5.1 – Etapes de la méthode de rejet*

5.2 Génération de section aléatoire

La méthode de génération de section candidate est déterminante pour la performance de la méthode de rejet. Une génération de candidats trop éloignés de la configuration cherchée va mener à de nombreux rejets et donc à un long temps d'exécution. Grâce à la comparaison qui empêche d'obtenir une ligne plus mauvaise que la précédente, la convergence est toujours garantie même en cas de génération très peu efficace. Le contexte impose cependant de bonne performances, nous allons donc biaiser la génération aléatoire de façon à ce qu'elle génère des sections généralement plus proche de ce que l'on cherche, tout en étant différentes.

Plutôt que de recréer une section totalement aléatoire, nous allons modifier la meilleure section actuelle. On diminue ainsi les chances d'avoir une section aberrante. Cette modification est aléatoire, mais doit être légère en moyenne pour éviter d'obtenir des sections trop différentes et donc potentiellement rejetées. La méthode doit cependant être capable de générer n'importe quelle configuration de section, on s'expose sinon à un risque de blocage dans le cas où la configuration cherché est particulièrement spécifique et ne peut être générée. Nous allons donc tout de même garder une petite probabilité d'obtenir de grandes modifications, de plus en cas de section très mauvaise ces grands changements peuvent beaucoup améliorer la section en une seule itération.

Dans notre cas la génération de section candidate prend comme origine une copie de la meilleure section, on y déplace ensuite aléatoirement un point. Dans un souci d'assurer la généricité de la génération, le nombre de point déplacé est aussi aléatoire. On va cependant préférer les petits nombres de points déplacés pour limiter les grands changements et donc le nombre de rejet.

5.3 Calcul de l'énergie

Nous allons maintenant nous pencher sur le calcul de l'énergie E. On rappelle que cette valeur est utilisée comme métrique par la méthode de rejet pour évaluer l'adéquation d'une section avec la configuration souhaitée. Plus l'énergie est haute et plus la section est considéré mauvaise. Il nous faut donc modéliser la configuration que l'on souhaite dans une formule qui nous donnera de forte valeurs lorsqu'elle est appliquée sur une section avec une configuration très différente de ce que l'on souhaite.

5.3.1 Optimisation de la distance

Nous souhaitons dans un premier temps obtenir une configuration dans laquelle chaque ligne est au plus près de tous les points de sa segmentation. Dans ce cas la modélisation est triviale, car une mauvaise configuration est simplement une section dont les lignes ont une forte distance avec leurs points. On obtient alors la formule suivante

$$E = \sigma^2 \tag{5.1}$$

avec σ^2 la somme des distances de chaque ligne avec tous ses points. Pour calculer σ^2 on peut utiliser la formule

$$\sigma^2 = \sum_{i=0}^{n} \sigma_i^2 \tag{5.2}$$

ou

$$\sigma_i^2 = \sum_{j=0}^{n} d(\mathrm{p_j}, (l_i))^2 \tag{5.3}$$

avec n le nombre de points, et $d(\mathrm{p_j}, (l_i))$ la distance entre le point p_j et la droite l_i. On rappelle que cette formule va servir à comparer deux sections, elle sera donc utilisée à chaque itération de l'algorithme de rejet ce qui rend sa performance critique. Hors la formule 5.3 qui implique un parcours de chacun des points de la ligne i, est utilisée dans la formule 5.2 sur chacune des lignes de la section. On obtient donc un parcours de l'ensemble des points de la section pour un seul calcul de distance, ce qui est très coûteux. Cette formule ne sera donc pas utilisée, nous allons plutôt réécrire le problème de la manière suivante :

$$\sigma^2 = \left(n\vec{c}^2 - 2\vec{c}\,\vec{\mu} + \mathrm{Tr}(A)\right) - \left(n\,(\vec{m}\cdot\vec{c})^2 - 2\,(\vec{m}\cdot\vec{c})\,(\vec{m}\cdot\vec{\mu}) + \vec{m}\,A\,\vec{m}\right) \tag{5.4}$$

avec n le nombre de points, $\vec{m}$ la direction de la ligne, $\vec{c}$ un point de la ligne, $\vec{\mu}$ la moyenne des points et A la matrice de variance. Vous pouvez retrouver le développement de cette réécriture dans l'appendice A. On remarque que tous les composants de la formule ont déjà été utilisés par le calcul de meilleure ligne du MarchingCylinder, ils sont donc directement disponible sans aucun calcul supplémentaire. Cette réécriture a donc changé l'ancienne boucle extrêmement coûteuse en un simple calcul analytique pratiquement gratuit.

5.3.2 Optimisation des angles

La configuration que l'on va maintenant modéliser est plus complexe : on souhaite favoriser les angles droits et lisser les angles faibles, tout en maintenant les lignes au plus proche des points. Nous avons un algorithme capable de modifier une section de façon à faire baisser son énergie, et il s'agira donc de simplement modifier le calcul d'énergie en y ajoutant les angles pour que l'algorithme parvienne au résultat souhaité.

Calcul de l'angle Contrairement à σ^2 on ne pourra pas directement utiliser la somme de tous les angles. En effet si tel était le cas, l'algorithme minimiserait la valeur de tous les angles, or on souhaite favoriser les angles droits et lisser les angles faibles ce qui est un comportement différent. Il faut donc concevoir une formule qui a une valeur faible pour des angles autour de $\frac{\pi}{2}$ et 0, et une valeur forte pour les autres. Cette valeur elle appelé α, et viendra remplacer l'utilisation direct de θ. Ainsi on obtient

$$\alpha_i = |\theta| \cdot (1 - |\theta|) \tag{5.5}$$

avec les définitions

$$\theta = \hat{AB} \cdot \hat{BC} = \cos(\angle ABC) \tag{5.6}$$

ou A est le point de début de la ligne l_i, C le point de fin de la ligne l_{i+1} et B le point en commun entre l_i et l_{i+1}. La valeur maximale d'un cosinus est 1, d'après la formule 5.5, on en déduit que la valeur maximale de α_i est $\frac{1}{2} \cdot \left(1 - \frac{1}{2}\right) = \frac{1}{4}$.

Ajout de l'angle à la formule d'énergie Nous avons une formule qui modélise bien la configuration d'angle que l'on souhaite, il faut maintenant l'inclure dans le calcul d'énergie tout en maintenant le même ratio d'importance. Autrement dit les angles et les points doivent avoir le même poids dans le calcul de l'énergie. Il est à noter que l'échelle des deux valeurs est différente, la distance d'une ligne à ces points est totalement dépendante du nuage, alors qu'un angle est compris entre 0 et 2π. La formule naïve $E = \sigma^2 + \alpha$ est donc très mauvaise, car un nuage avec beaucoup de points aura une valeur σ^2 très élevée et potentiellement largement supérieure à la valeur d'angle, ce qui rend cette dernière pratiquement négligeable.

Pour régler ce problème, nous allons mettre α à l'échelle de σ^2. Tout d'abord on normalise α entre 0 et 1 en le divisant par $max(\alpha)$. Comme $0 < \theta_i < \frac{1}{4}$, nous avons donc $max(\alpha) = \sum_{i=0}^{n} \frac{1}{4} = n \cdot \frac{1}{4}$. On ramène ensuite α à la même échelle que σ^2 en la multipliant par cette dernière. On en vient donc au calcul d'énergie suivant :

$$E = \sigma^2 \left(1 + \alpha \frac{4}{n}\right) \tag{5.7}$$

Formule paramétré On souhaite maintenant pouvoir ajuster la différence de poids entre les points et les angles. On donnera une valeur λ en paramètre qui va pondérer l'importance des angles par rapport aux distances. Par exemple si $\lambda = 2$ alors les angles auront deux fois plus d'impact que les points, les lignes pourront alors potentiellement s'éloigner des points pour former de meilleur angles. Pour cela il suffit de multiplier α par λ. La formule finale sera donc

$$E = \sigma^2 \left(1 + \alpha \frac{4 \cdot \lambda}{n}\right) \tag{5.8}$$

avec les définitions suivantes

$$\sigma^2 = \sum_{i=0}^{n} \sigma_i^2 \tag{5.9}$$

$$\alpha = \sum_{i=0}^{n} \alpha_i \tag{5.10}$$

$$\tag{5.11}$$

ou σ_i^2 est la somme de toute les distances entre la ligne i et les points de sa segmentation, voir formule 5.9, α_i est le calcul d'angle entre les lignes i et $i+1$ décrit dans la formule 5.5.

5.4 Cas d'arrêt

Nous avons maintenant une méthode capable d'améliorer une segmentation au fur et à mesure des itérations, en générant aléatoirement des sections candidates pour ne retenir que celles avec une énergie moindre. Le caractère aléatoire de la génération fait qu'il est toujours possible d'obtenir de nouvelles sections candidates, le problème est donc maintenant de savoir quand la segmentation est suffisamment bonne pour arrêter l'algorithme.

Utilisation de l'énergie Une solution naïve serait d'utiliser l'énergie, on pourrait par exemple arrêter lorsque cette dernière diminue en dessous d'un seuil. Cette solution n'est pas viable car la valeur d'énergie est totalement dépendante des données, l'échelle des valeurs peut être complètement différente selon le nuage. Un nuage avec beaucoup de lignes ainsi que beaucoup de points par lignes aura une énergie largement supérieure à un nuage avec peu de lignes et peu de points. Ce phénomène est dû au calcul de l'énergie qui ne pondère pas sa valeur par le nombre de points. Même si l'énergie était normalisée entre tous les nuages, l'ajustement du poids entre angle et distance peut très fortement contraindre l'optimisation, et une valeur seuil applicable dans un cas pourrait mener à des temps de calcul excessivement long dans d'autres cas. Le nuage d'un bâtiment avec des angles atypiques par exemple, aura une valeur d'énergie sensiblement plus forte même en cas de lignes très bien ajustées à la segmentation, et peut donc ne jamais descendre en dessous du seuil. Une valeur applicable à tous les cas ne serait quant à elle pas assez restrictive, car trop haute pour des nuages à faible valeur d'énergie.

Probabilité d'acceptation Nous allons donc utiliser une autre métrique propre aux méthodes de rejet, la probabilité d'acceptation. Il s'agit d'une valeur qui quantifie la probabilité qu'a une section candidate d'être meilleure que la meilleure section actuelle. On la calcule avec la formule

$$p(A) = \frac{\text{Card}(A)}{\text{Card}(\Omega)} = \frac{\text{nombre de sections acceptées}}{\text{nombre de sections candidates}} \tag{5.12}$$

avec A l'événement de tirer une section qui a remplacé la meilleure section courante parmi toutes les sections générées, et Ω toutes les possibilités. Ainsi plus la probabilité d'acceptation est haute, plus une section candidate a des chances d'être meilleure que la meilleure section, autrement dit plus les chances d'amélioration sont grandes. Nous allons nous servir de cette valeur comme cas d'arrêt, lorsque la meilleure section a trop peu de chances de s'améliorer on considère qu'elle est est suffisamment proche de la configuration souhaitée et on arrête l'algorithme. De la même façon que pour l'énergie, il n'est cependant pas possible de déterminer une valeur seuil applicable à tous les cas. La différence est que la probabilité d'acceptation d'un nuage est calculable en quelques itérations. On peut ainsi faire une estimation rapide de l'écart du nuage avec la configuration souhaitée et adapter notre seuil en conséquence.

Finalement voici l'algorithme choisi. On réalise une première estimation en appliquant la méthode de rejet sur un temps fixe, deux secondes dans notre cas. On calcule ensuite la probabilité d'acceptation, si cette dernière est égale ou inférieure à la valeur seuil ε, on considère que la section est correcte et on arrête l'algorithme. Dans le cas contraire, on continue jusqu'à ce que la probabilité atteigne le seuil. Si la probabilité est deux fois plus grande que le seuil c'est que la section est particulièrement mauvaise, atteindre ε peut se révéler très long. Nous allons donc plutôt arrêter l'algorithme lorsque la probabilité aura atteint le double de l'estimation initiale, ce qui donnera une moins bonne section qu'avec un seuil à ε mais doublera néanmoins la qualité. Nous avons fixé $\varepsilon = \frac{1}{10}$, cette valeur a été déterminée par l'expérience.

5.5 Résultat

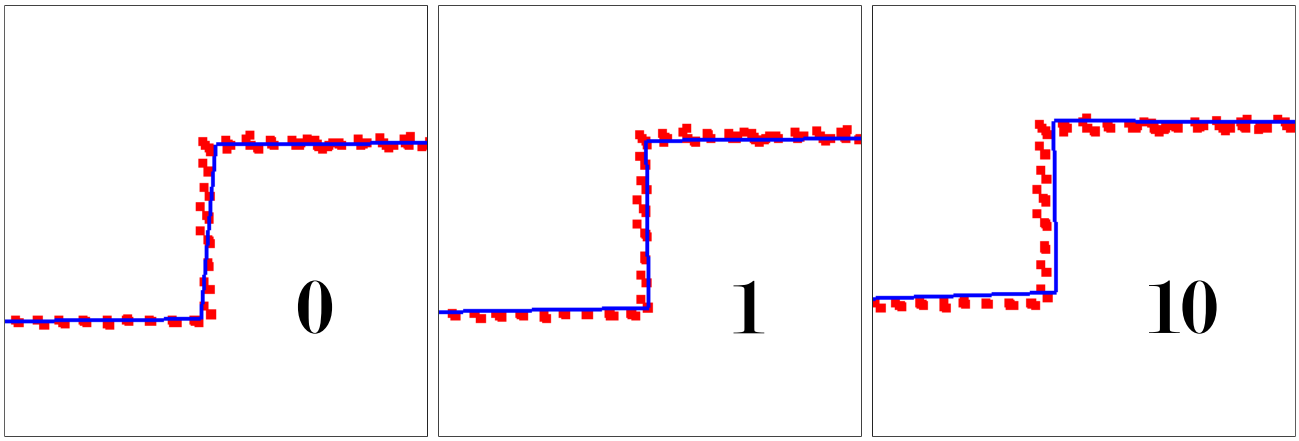


FIGURE *5.2 – Résultat de l'optimisation*

Voici les résultats obtenus suite à l'optimisation des lignes. L'indice présent sur l'image présente l'importance des angles par rapport aux lignes. On peut voir que sans optimisation les lignes suivent grossièrement les points. Lorsque les angles ont autant de poids que les points, la ligne suis toujours les points mais les angles sont plus droits. Lorsque les angles ont dix fois plus de poids que les lignes, on peut voir que les angles sont parfaitement droits, mais que les lignes dévient du nuage.

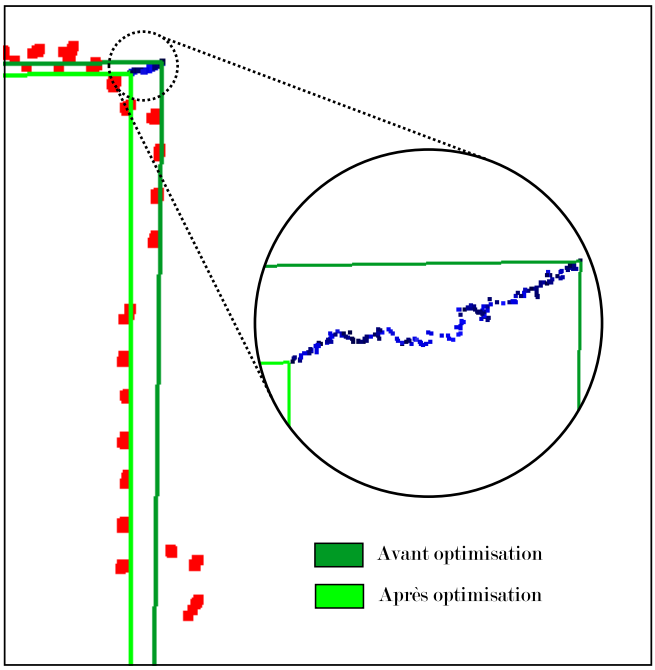


FIGURE *5.3 – Exemple de marche aléatoire*

Comme expliqué précédemment la méthode de rejet va améliorer la section en proposant une succession de sections aléatoires, la figure 5.3 illustre la cohérence des sections acceptées. Pour produire cette illustration nous avons enregistré la position de l'angle (les points bleus) à chaque fois qu'une meilleure section est retenue La forme chaotique de cette chaîne illustre bien le caractère aléatoire de la méthode, en revanche on peut voir que toute position converge vers une direction général, l'angle ne recule jamais. C'est cohérent car la méthode empêche de conservé des sections moins bonnes que les précédentes.

Chapitre 6

Conclusion

Finalement, l'intégralité de la fonctionnalité est maintenant disponible dans la librairie, avec un code conforme, testé et parfaitement fonctionnel. Plusieurs réunions avec le responsable qualité ont permis de valider les résultats, la fonctionnalité sera donc prochainement intégré à 3DReshaper et deviendra un argument de vente supplémentaire pour l'entreprise. C'est une grande satisfaction d'avoir pu contribuer au développement de ce logiciel, mais au-delà de l'aspect purement matériel ce projet m'a aussi énormément apporté en connaissances et compétences à de nombreux niveaux.

Tout d'abord j'ai eu la chance de pouvoir travailler sur un projet très intéressant d'un point de vu mathématique. Des concepts puissant ont été utilisés tel que la covariance ainsi que les valeurs et vecteurs propres. Concepts qui ont par ailleurs déjà été abordés durant la formation ID3D ce qui m'a permis de rapidement me les approprier et ainsi les inclure correctement au projet. Le projet a également un réel intérêt algorithmique, la méthode au complet comporte un algorithme de Monte-Carlo ainsi qu'un algorithme s'apparentant à du region growing. Enfin l'extraction de ligne d'une section de bâtiment est un problème très spécifique qui n'est donc pas abordé tel quel dans la littérature. La résolution de ce problème ne se résume donc pas au simple développement d'un algorithme existant mais bien à la conception d'une nouvelle solution. Ce travail de recherche, de synthèse et de réflexion a été particulièrement formateur et à changé ma façon de raisonner et d'aborder les problèmes. La notion de performances, rarement abordé durant les études car peu pertinent dans ce contexte, a également été un nouveau paramètre à prendre en compte et un défi intéressant à relever.

Ce projet a également été extrêmement formateur d'un point de vue technique. Ajouter du code à une large bibliothèque professionnelle de traitement de nuages de points nécessite tout un travail d'adaptation ainsi que la manipulation d'objets complexes. De plus la nature de la fonctionnalité a nécessité la manipulation et parfois même la modification de nombreux objets complexes comme les nuages de points, les procédures de sélection ou encore les polylignes. L'utilisation de concepts C++ avancés tels que les lambdas expressions ou les templates a également été indispensable. La encore, la plupart de ces concepts ont été abordés durant la formation ce qui a grandement accéléré leur prise en main. Finalement le contexte industriel impose un code maintenable et stable, c'est pourquoi un grand nombre de tests a également été codé en parallèle de l'algorithme afin de valider progressivement les nouveaux développements. Cette discipline a grandement accéléré le processus parfois chaotique que constitue la recherche de résolution de problème, et fait maintenant partie intégrante de ma façon de raisonner.

Le travail au sein d'une équipe de développeurs expérimentés a également amélioré ma façon de travailler en équipe. En parallèle du développement de l'algorithme, j'ai assisté à de nombreuses réunions durant lesquelles quelque corrections de bugs mineurs m'ont été assignés. Ces développements rapides m'ont permis de m'habituer aux routines de travail en entreprise, de naviguer dans une bonne partie de la librairie et d'échanger avec des développeurs d'autres équipes. De plus, un jour par moiss j'ai réalisé une présentation des avancés ainsi qu'une démonstration de l'algorithme au reste de l'entreprise, parfois en anglais, ce qui a développé mes capacités de présentation orale en public.

Finalement ce projet a été extrêmement formateur à tout point de vu, et constitue une expérience ainsi qu'un atout réel pour la suite de mon parcours professionnel.

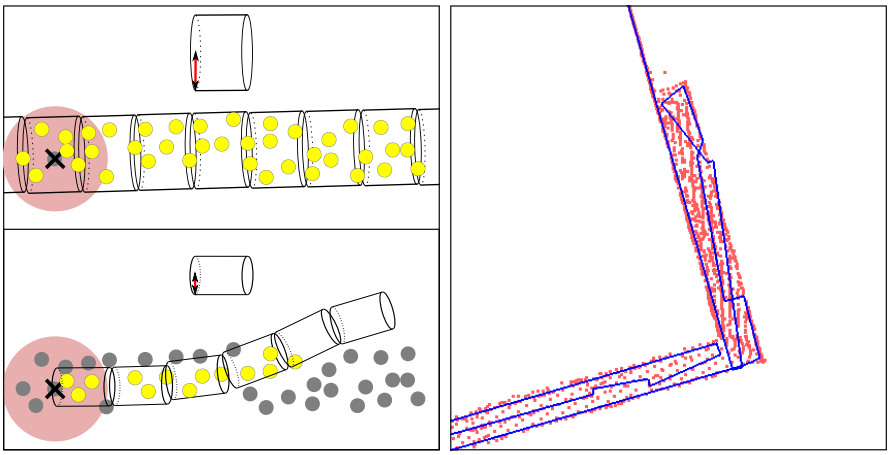
6.1 Limites et perspectives

La fonctionnalité est terminée et fonctionnelle, cependant quelques limites subsistent et un travail supplémentaire pourrait amener de nouvelles possibilités.

Génération automatique des paramètres Aujourd'hui la méthode complète présente un nombre de cinq paramètres dont le choix est crucial pour le bon fonctionnement de l'algorithme. Cette flexibilité permet d'obtenir de bon résultats sur n'importe quel nuage, mais choisir les bons paramètres peut parfois se révéler complexe sans une connaissance profonde de la méthode, surtout dans le cas de nuage très bruité et imprécis. Une amélioration pourrait être la génération automatique de certains paramètres en fonction des données d'entrée. Une piste explorée est l'utilisation des valeurs propres de la première sélection pour initialiser le rayon des cylindres, ou un système de multi-passe pour améliorer la précision des extrémités et contourner le choix de la longueur des cylindres. La génération d'un premier germe est également envisagée pour s'affranchir du clique.

Adaptation selon l'épaisseur des lignes La plus grosse limite actuelle de l'algorithme est l'impossibilité de traiter des sections avec des murs de différentes épaisseurs, voir figure 6.1. En effet, le rayon des cylindres étant une valeur fixe entrée par l'utilisateur, l'algorithme n'a aucun moyen de la modifier. Une génération automatique du rayon des cylindres pourrait permettre une adaptation de l'algorithme à l'épaisseur de la ligne en cours d'extraction. Finalement il s'agit d'une variante de l'amélioration précédemment décrite, sauf qu'il faut appliquer la génération de paramètre au début de chacune des extractions de murs et non plus uniquement au début de l'algorithme ce qui nécessite des adaptations.

Extraction d'embranchement générique L'extraction de section permet d'obtenir toutes les lignes des murs d'enceinte d'un bâtiment, en revanche il est impossible d'extraire toutes les lignes d'un embranchement de plus de deux murs. La ligne manquante doit être extraite par un lancement manuel indépendant du MarchingCylinder algorithme. Actuellement si un tel cas se présente l'extraction de sections va prioriser le mur avec l'angle le plus faible. Cette solution fonctionne dans la plupart des cas mais rien ne garantit qu'il s'agit bien de la meilleure direction à prendre, notamment si plusieurs bâtiments sont accolés.



(a) Importance d'un rayon de cylindre correct

(b) Problème d'adaptation

FIGURE *6.1 – Principale limite de l'algorithme*

Bibliographie

I. Anagnostopoulos, V. Pătrăucean, I. Brilakis, and P. Vela. Detection of walls, floors, and ceilings in point cloud data. 05 2016.

D. L. Bool, L. C. Mabaquiao, M. E. Tupas, and J. L. Fabila. Automated building detection using ransac from classified lidar point cloud data. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 10 2018.

O. Chum. Two-view geometry estimation by random sample and consensus. Thèse, 2005. URL http://cmp.felk.cvut.cz/~chum/papers/Chum-PhD.pdf.

L. Elie and B. Lapeyre. Introduction aux méthodes de monte-carlo. Cours, 2001. URL http://cermics.enpc.fr/~bl/PS/SIMULATION-X/poly-monte-carlo-x.pdf.

A. Guyader. Méthode de monte-carlo. Cours, 2010. URL http://www.lsta.upmc.fr/guyader/files/teaching/MonteCarlo/MonteCarlo.pdf.

P. G. Hélène. Macher, Tania. Landes. From point clouds to building information models : 3d semi-automatic reconstruction of indoors of existing buildings. Article, 2017.

H. Isack and Y. Boykov. Energy-based geometric multi-model fitting. Article de journal, 2012. URL http://www.csd.uwo.ca/~yuri/Papers/tr735.pdf.

N. Janakiev. Understanding the covariance matrix. Article, 2018. URL https://datascienceplus.com/understanding-the-covariance-matrix.

P. J. Rousseeuw. Least median of square regression. Article de journal, 1984. URL http://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/LeastMedianOfSquares.pdf.

J. Wittwer. A practical guide to monte carlo simulation, 2004.

B. Xu, W. Jiang, J. Shan, J. Zhang, and L. Li. Investigation on the weighted ransac approachesfor building roof plane segmentation from point clouds. Article, 2015. URL https://pdfs.semanticscholar.org/7de5/5bbd61e47d2fe2fe15d924f851dc7e0336ba.pdf.

Annexe A

Re-formulation du calcul de distance

Soit la moyenne des points μ_i tel que

$$\mu_i = \sum_{k=0}^{n} P_i^k \tag{A.1}$$

ainsi que la matrice de variance A_{ij} tel que

$$A_{ij} = \sum_{k=0}^{n} p_i^k p_j^k \tag{A.2}$$

nous allons exprimer la distance entre tous les points et leur ligne d^2 en fonction de μ et A. Tout d'abord nous avons les points $\vec{p}$, $\vec{m}$ et $\vec{c}$ tel que

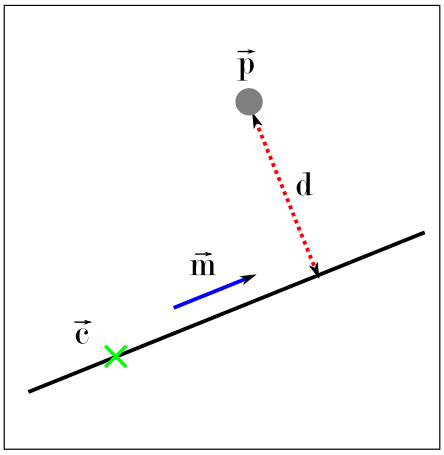


FIGURE *A.1*

Ainsi

$$\begin{aligned} d^2 &= (\vec{c}-\vec{p})^2 - [(\vec{c}-\vec{p})\,\vec{m}]^2 \\ &= \sum_{i=1}^{3}(c_i-p_i)^2 - \left(\sum_{i=1}^{3}(c_i-p_i)m_i\right)^2 \\ &= \sum_{i=1}^{3}(c_i-p_i)^2 - \sum_{i=1}^{3}(c_i-p_i)m_i \cdot \sum_{j=1}^{3}(c_j-p_j)m_j \\ &= \sum_{i=1}^{3}(c_i-p_i)^2 - \sum_{i,j=1}^{3}(c_i-p_i)m_i\,(c_j-p_j)m_j \end{aligned} \tag{A.3}$$

nous avons donc

$$d_k^2 = \underbrace{\sum_{i=1}^{3}(c_i-p_i^k)(c_i-p_i^k)}_{①} - \underbrace{\sum_{i,j=1}^{3}(c_i-p_i^k)m_i\,(c_j-p_j^k)m_j}_{②} \tag{A.4}$$

ainsi que

$$\sigma^2 = \sum_{k=0}^{n} d_k^2 \tag{A.5}$$

$$\tag{A.6}$$

L'objectif est maintenant de réécrire ① et ② pour ne plus avoir l'intervention des points, nous avons ainsi

$$\begin{aligned}
① &= \sum_{k=0}^{n}\sum_{i=1}^{3}\left[(c_i - p_i^k)(c_i - p_i^k)\right] \\
&= \sum_{k=0}^{n}\sum_{i=1}^{3}\left[c_i \cdot c_i - 2c_i p_i^k + p_i^k \cdot p_i^k\right] \\
&= \sum_{i=1}^{3}\sum_{k=0}^{n}\left[c_i \cdot c_i - 2c_i p_i^k + p_i^k \cdot p_i^k\right] \\
&= \sum_{i=1}^{3}\left[\sum_{k=0}^{n} c_i \cdot c_i - 2c_i \underbrace{\sum_{k=0}^{n} p_i^k}_{\mu_i} + \underbrace{\sum_{k=0}^{n} p_i^k\ p_i^k}_{A_{ii}}\right] \\
&= \sum_{i=1}^{3}\left[nc_i^2 - 2c_i\mu_i + A_{ii}\right] \\
&= n\vec{c}^{\,2} - 2\vec{c}\,\vec{\mu} + \mathrm{Tr}(A)
\end{aligned} \tag{A.7}$$

$$\tag{A.8}$$

$$\begin{aligned}
② &= \sum_{k=0}^{n}\sum_{i,j=1}^{3}\left[\left(c_i - p_i^k\right) m_i\ \left(c_j - p_j^k\right) m_j\right] \\
&= \sum_{k=0}^{n}\sum_{i,j=1}^{3}\left[m_i m_j\left(\left(c_i - p_i^k\right)\left(c_j - p_j^k\right)\right)\right] \\
&= \sum_{k=0}^{n}\sum_{i,j=1}^{3}\left[m_i m_j\left(c_i c_j - c_i p_j^k - p_i^k c_j + p_i^k p_j^k\right)\right] \\
&= \sum_{i,j=1}^{3} m_i m_j\left(\sum_{k=0}^{n} c_i c_j - c_i \underbrace{\sum_{k=0}^{n} p_j^k}_{\mu_j} - \underbrace{\sum_{k=0}^{n} p_i^k}_{\mu_i}\, c_j + \underbrace{\sum_{k=0}^{n} p_i^k p_j^k}_{A_{ij}}\right) \\
&= \sum_{i,j=1}^{3} m_i m_j\left(nc_i c_j - c_i\mu_j - \mu_i c_j + A_{ij}\right) \\
&= \sum_{i,j=1}^{3} n\ m_i c_i\ m_j c_j - m_i c_i\ m_j\mu_j - m_i\mu_i\ m_j c_j + m_i m_j A_{ij} \\
&= n\left(\vec{m}\cdot\vec{c}\right)\left(\vec{m}\cdot\vec{c}\right) - \left(\vec{m}\cdot\vec{c}\right)\left(\vec{m}\cdot\vec{\mu}\right) - \left(\vec{m}\cdot\vec{\mu}\right)\left(\vec{m}\cdot\vec{c}\right) + \vec{m}\ A\ \vec{m} \\
&= n\left(\vec{m}\cdot\vec{c}\right)^2 - 2\left(\vec{m}\cdot\vec{c}\right)\left(\vec{m}\cdot\vec{\mu}\right) + \vec{m}\ A\ \vec{m}
\end{aligned} \tag{A.9}$$

Nous avons réussi à écrire d_k^2 en fonction de matrices de variance ainsi que de la moyenne de

tous les points, il est donc possible de calculer la somme des distances entre des points et une ligne uniquement grâce à la matrice de covariance de ce point.

Nous avons donc finalement

$$\begin{aligned} d^2 &= (\vec{c} - \vec{p})^2 - [(\vec{c} - \vec{p})\,\vec{m}]^2 \\ &... \\ &= n\vec{c}^2 - 2\vec{c}\,\vec{\mu} + \mathrm{Tr}(A_{ii}) - \left(n\,(\vec{m}\cdot\vec{c})^2 - 2\,(\vec{m}\cdot\vec{c})\,(\vec{m}\cdot\vec{\mu}) + \vec{m}\;A_{ij}\;\vec{m}\right) \end{aligned} \tag{A.10}$$

Cette expression ne fait pas intervenir p ni la moindre somme, le calcul est donc gratuit.