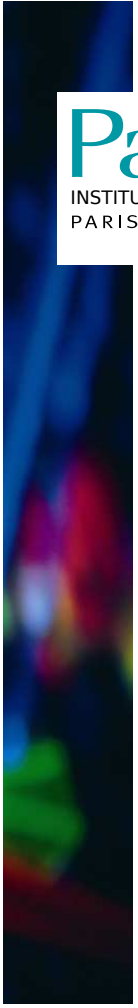
Traitements de nuages de points denses et modélisation 3D d'environnements par système mobile LiDAR/Caméra

Jean-Emmanuel Deschaud



ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

MINES ParisTech

École doctorale n°432 : Sciences des Métiers de l'Ingénieur

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École nationale supérieure des mines de Paris

Spécialité « Informatique temps-réel, robotique et automatique »

présentée et soutenue publiquement par

Jean-Emmanuel DESCHAUD

le 22 Novembre 2010

Traitements de nuages de points denses et modélisation 3D d'environnements par système mobile LiDAR/Caméra

~ ~ ~

Dense point cloud processing and 3D environment modeling from LiDAR/Camera mobile system

Directeur de thèse : **François GOULETTE**

Jury

M. Denis Laurendeau, Professeur, Université de Laval (Canada)	Rapporteur
M. Nicolas Paparoditis, Directeur de Recherche, IGN	Rapporteur
M. Martial Hébert, Professeur, Carnegie Mellon University (USA)	Examinateur
M. Pierre Grussenmeyer, Professeur, INSA Strasbourg	Examinateur
M. Pierre Charbonnier, Directeur de Recherche, LRPC de Strasbourg	Examinateur
M. Thomas Chaperon, Ingénieur de Recherche, Mensi-Trimble	Examinateur
M. François Goulette, Enseignant Chercheur, MINES ParisTech	Directeur de Thèse

T H È S E

MINES ParisTech
Centre de Robotique - Département Mathématiques et Systèmes
60 Boulevard Saint-Michel, 75272 Paris Cedex 06, France

Remerciements

Je voudrais remercier à travers ces quelques lignes les différentes personnes qui m'ont aidé et entouré lors de cette thèse. Je voudrais tout d'abord saluer François Goulette, mon directeur de thèse qui m'a montré la voie de la recherche et m'a donné de nombreux conseils tout au long de ces trois années. Je tiens aussi à remercier les différents membres du jury pour leur présence et particulièrement Pierre Charbonnier avec qui j'ai eu l'honneur d'être en collaboration sur un des projets de ma thèse.

Ce document ne serait pas aussi abouti sans l'aide de mes relecteurs, notamment François Goulette, Raoul De Charette, Taha Ridene, ma soeur Anne-Laure et ma maman.

Je profite de ces quelques lignes pour remercier la très bonne ambiance de travail qui reigne au sein du laboratoire de robotique de Mines ParisTech. Cela donne beaucoup d'énergie le matin et la possibilité de boire de nombreux cafés tout au long de la journée. De manière non exaustive, je passe le bonjour à Claude Laurgeau l'ancien directeur, Arnaud De La Fortelle le nouveau directeur, Raoul, Taha, Laure, Ayet, Oussama, Amaury, Gwennaël, Alexandre, Vincent, Omar, Fatin, Alexis, Eric, Jacky, Christine, Christophe, Fawzi, François, Hyun-Jae, Jean-François, Keerthi, Nan, Philippe, Anne-Sophie, Bogdan, Bruno, Joël, Fabien, Sébastien, Kamel, Shenle, Sungwoo et Aurélien.

J'exprime ma gratitude à mes parents qui m'ont soutenu pendant ces trois ans, ma petite soeur qui m'a hébergé et supporté sur la fin de la rédaction, mon petit frère Gilles-alexandre qui suit aussi le chemin de la thèse et enfin mon grand frère Pierre-Olivier.

Je vous laisse maintenant libre pour la lecture de ce mémoire. Encore merci à tous !

Table des matières

Introduction

La modélisation 3D d'environnements consiste à reproduire en virtuel un décor réel comme un bâtiment, une rue, l'intérieur d'un appartement ou même à plus grande échelle une ville entière. Il s'agit d'un domaine très porteur avec de nombreuses applications. En effet, nous trouvons des modèles 3D d'environnements dans des domaines aussi variés que l'architecture, la cartographie, l'aménagement du territoire, le tourisme, les jeux-vidéos, le cinéma, etc. Pour construire ces modèles, il existe de nombreux logiciels de modélisation qui nécessitent d'un *designer* plusieurs heures pour obtenir un résultat réaliste. Or certaines applications demandent des modèles de plus en plus réalistes, de plus en plus précis et pour des zones de grande étendue, le tout en un minimum de temps. Comparées à la modélisation interactive, les approches entièrement automatiques sont alors préférables pour traiter de grands volumes de données. Cette thèse se place dans cette problématique de construction automatique de grandes bases de données 3D d'environnements à l'aide de systèmes mobiles de numérisation.

La plateforme mobile que nous avons utilisée, appelée « LA Route Automatisée en 3-Dimensions » (LARA3D), a été conçue au laboratoire de robotique de MINES ParisTech [Abuhadrous, 2005]. Le véhicule, localisé en temps réel par fusion des données d'un GPS et d'une centrale inertielle permet avec un capteur LiDAR (« *LIght Detection And Ranging*) la création de nuages de points 3D de l'environnement traversé. Ce prototype a ensuite été amélioré par l'ajout de caméras fish-eye pour la création de modèles 3D texturés [Brun, 2007]. Au fil du temps, toute une équipe s'est formée autour de la thématique de recherche en lien avec LARA3D : Hyun-Jae Yoo sur le développement d'un nouveau capteur LiDAR dédié à la cartographie mobile, Keerthi Narayana sur l'amélioration de la localisation et Taha Ridene sur le recalage des données LARA3D [Ridene, 2010].

Cette thèse s'est déroulée dans le cadre de deux projets de recherche. Le premier est le projet Dialogue Infrastructure-Véhicule pour Améliorer la Sécurité routière (DIVAS) du Programme de Recherche Et d'Innovation dans les Transports terrestres (PREDIT). En collaboration avec le Laboratoire Régional des Ponts et Chaussées (LRPC) de Strasbourg, notre contribution porte sur l'amélioration de la modélisation 3D d'environnements routiers (routes, bas-côtés, arbres, panneaux, etc.) pour l'estimation de distances de visibilité. Le deuxième projet, TerraNumerica (pôle de compétitivité Cap Digital), a pour ambition de franchir les barrières technologiques sur la création de modèles 3D photo-réalistes de scènes urbaines, notamment de façades d'immeubles. Ces deux projets bien que très différents nécessitent tous deux une génération de modèles 3D de haute fidélité, tout en conservant des données suffisamment compactes. Durant notre étude, nous nous focalisons sur le problème de la création d'un modèle 3D terrestre photo-réaliste avec en entrée un nuage de points bruité et des images fish-eye géo-référencées. Des traitements spécifiques pour chacune des applications ont été pris en considération tout en restant le plus générique

possible afin de permettre le traitement de nuages de points venant d'autres systèmes.

Le projet de recherche autour de la plateforme mobile LARA3D cherche à produire des données 3D en temps réel, c'est-à-dire avec un temps de modélisation similaire au temps d'acquisition. A terme, l'avantage de ces systèmes mobiles serait la production par traitements embarqués, de bases de données 3D. Pour cela, notre travail a été de développer une chaîne complète pour construire des modèles au rendu photo-réaliste par des méthodes automatiques, le tout en un minimum de temps.

Le chapitre 1 de ce mémoire décrit l'état de l'art des technologies d'acquisition et de modélisation de nuages de points 3D ainsi que notre approche de modélisation photo-réaliste. Le chapitre 2 développe l'analyse des nuages de points créés par système mobile et plus particulièrement par la plateforme LARA3D. Les chapitres 3 et 4 détaillent les contributions majeures de la thèse concernant l'amélioration des traitements de nuages de points 3D denses. Le chapitre 5 expose l'ajout de l'information de couleur aux nuages de points et de textures aux modèles 3D. Enfin, le chapitre 6 présente trois applications pour les modèles 3D : la première concernant la modélisation de scènes urbaines (liée au projet TerraNumerica), la deuxième pour l'estimation de distances de visibilité à partir de modèles 3D d'environnements routiers (liée au projet DIVAS) et enfin la dernière application relative à la création de modèles 3D de scènes d'intérieur à partir d'une autre plateforme mobile démontrant la généricité de notre méthode.

Les contributions de ces travaux de recherche se situent principalement sur les traitements de nuages de points avec entre autre :

- une analyse des données de système mobile ;
- une nouvelle méthode de calcul de normales pour les nuages de points 3D ;
- un nouvel algorithme de débruitage de données 3D ;
- une amélioration de l'algorithme de segmentation par croissance de régions à la fois en terme de rapidité et en terme de qualité ;
- un nouvel algorithme de triangulation de nuage de points 3D ;
- une méthode de colorisation/texturation ;
- une chaîne de modélisation générique et déclinée suivant plusieurs applications.

Ces travaux ont donné lieu à plusieurs publications listées à la page 199.

Chapitre 1

Acquisition et modélisation d'environnements

Résumé

Ce premier chapitre décrit les différents moyens d'acquisitions de surfaces et plus précisément les techniques d'acquisitions terrestres d'environnements. Nous détaillons les différents systèmes terrestres mobiles de numérisation et décrivons trois plateformes de production de données 3D utilisées dans le cadre de cette thèse. Ce chapitre détaille aussi les méthodes existantes de la modélisation 3D, l'étape pour transformer les nuages de points en modèle 3D. Enfin, nous terminons par la description de notre approche globale de modélisation pour la construction de modèles photo-réalistes d'environnements terrestres.

1.1 Technologies d'acquisition

Il existe trois grandes familles de technologie de capteurs de surfaces : (1) le palpage, qui consiste à mettre en contact un appareil de mesure ; (2) les systèmes optiques passifs (avec par exemple la stéréovision) ; (3) les systèmes optiques actifs (les radars, les LiDARs, les projections de grilles lumineuses...) [Goulette et Laurgeau, 2002]. Parmi toutes ces technologies d'acquisition, la technologie LiDAR est très répandue. Elle permet de mesurer la distance à une surface à l'aide d'une onde lumineuse rétro-diffusée par la surface. La mesure de la distance se fait généralement soit par l'évaluation du temps d'aller-retour de l'onde soit par différence de phase entre l'onde aller et l'onde retour. Les LiDARs terrestres qui utilisent une onde laser sont appelés plus communément scanners laser. Nous pouvons distinguer principalement deux catégories de numérisation laser : les technologies dites fixes, c'est-à-dire où les données sont acquises avec un appareil fixe et les systèmes mobiles où les données sont acquises à l'aide d'un robot, d'une plateforme ou d'un véhicule en déplacement.

Les scanners laser fixes sont des appareils de mesure générant de l'ordre de 5 000 à 100 000 points par seconde avec une précision des points (un sigma) de l'ordre de 1 à 10 mm. Par exemple le Trimble GX (figure 1.1) permet de générer 5 000 points par seconde avec une précision (un sigma) de l'ordre de 5 à 10 mm.

FIGURE 1.1 – Scanner Laser Trimble GX.

Pour créer un modèle complet, il faut généralement plusieurs stations, c'est-à-dire plusieurs positions de l'appareil de mesure. Pour construire le nuage de points d'un site entier, les scans (données obtenues pour une station) sont recalés les uns par rapport aux autres, généralement avec un algorithme comme l'« *Iterative Closest Point* » (ICP). Le déplacement du scanner laser entre les différentes stations nécessite un temps de mise en œuvre relativement long ce qui rend les systèmes fixes peu adaptés à la numérisation de zones de grande étendue. Nous allons étudier plus en détail les systèmes mobiles terrestres qui permettent de numériser de larges environnements en peu de temps.

1.1.1 Systèmes mobiles terrestres pour la numérisation d'environnements

Nous pouvons distinguer deux classes de systèmes mobiles : ceux qui utilisent un appareil de mesure fixe en le plaçant dans un environnement mobile pour diminuer le temps entre acquisitions, et les systèmes mobiles qui utilisent le déplacement comme axe de numérisation.

Mode Stop & Go

Dans cette première catégorie, les plateformes mobiles stoppent leur déplacement le temps de l'acquisition puis se déplacent à la prochaine station : on appelle cela le mode « *Stop&Go* ». [Allen et al., 2001] ont monté le projet AVENUE dont l'objectif est l'automatisation du processus de modélisation d'environnement urbain avec un robot basé sur le principe du « *Stop&Go* ». La figure 1.2 montre le robot composé de capteurs de positionnement (GPS+odométrie+vision), de caméras et d'un scanner laser Cyrax. Un système de prédiction de trajectoire va estimer la prochaine meilleure vue pour numériser un site en entier. [Asai et al., 2004] ont développé un véhicule d'acquisition (figure 1.2) avec la même méthode « *Stop&Go* ». Ils utilisent un GPS RTK (« *Real Time Kinematic* ») pour la localisation et un scanner omnidirectionnel Riegl qui numérise l'environnement à 360° lors des différents arrêts du véhicule. Pour ces deux exemples, la localisation du mobile va servir d'initialisation pour le recalage des données laser. Ainsi cette méthode d'acquisition produit des données similaires à une méthode d'acquisition fixe. Le mobile permet seulement de se déplacer plus rapidement d'une station à une autre mais avec une information supplémentaire sur la localisation des différents scans grâce à sa localisation.

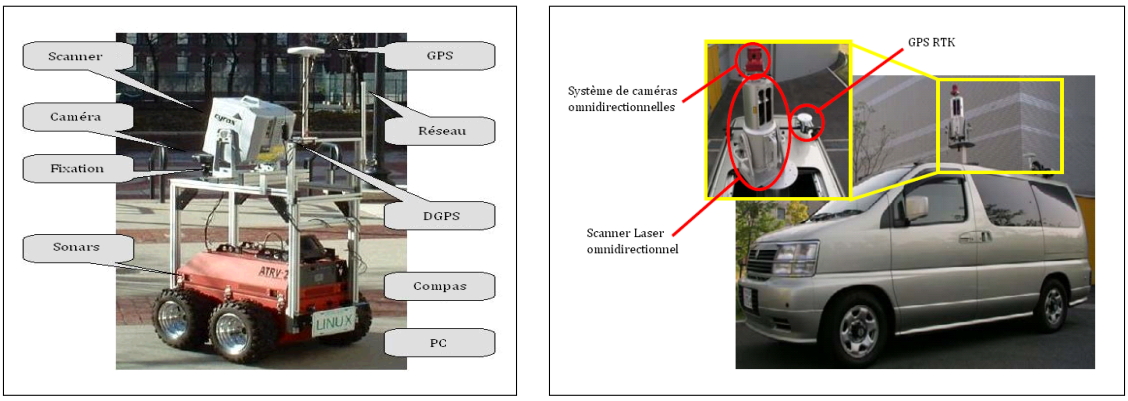


FIGURE 1.2 – A gauche, robot mobile pour le projet AVENUE (d'après [Allen et al., 2001]). A droite, véhicule de numérisation de l'équipe d'Asai *et al.* (d'après [Asai et al., 2004]).

Mode continu

Dans cette deuxième catégorie, les plateformes mobiles utilisent la trajectoire du véhicule comme axe de numérisation : on appelle cela le mode continu. Les systèmes mobiles terrestres basés sur la technologie LiDAR en mode continu peuvent être distingués

suivant leur système de localisation. Le système de positionnement de l'équipe de Zhao *et al.* [Zhao et Shibasaki, 2003] est basé sur la correspondance de profils laser horizontaux (figure 1.3). Prenant pour hypothèse que le sol est plat, un scanner balayant à l'horizontal permet de localiser le véhicule à chaque instant par correspondance de profils. Les données d'un deuxième laser balayant dans un plan vertical sont alignées dans un référentiel global pour donner un nuage de points 3D de l'environnement. [Früh et Zakhor, 2004] et [Biber et al., 2005] ont conçu des prototypes mobiles très similaires à celui de Zhao *et al.* comportant aussi deux scanners : un scanner horizontal pour la cartographie 2D et un scanner vertical pour la numérisation de la géométrie (figure 1.3). Mais la technologie d'acquisition par correspondance de cartes 2D limite généralement la vitesse d'acquisition à 5-10 km/h.

FIGURE 1.3 – A gauche, système de numérisation de l'équipe Zhao *et al.* (d'après [Zhao et Shibasaki, 2003]). Au milieu, plateforme d'acquisition de l'équipe de Früh *et al.* (source [Früh et Zakhor, 2004]). A droite, plateforme de l'équipe de Biber *et al.* (source [Biber et al., 2005]).

Les plateformes mobiles opérant en mode continu peuvent être localisées par la fusion de données de capteurs de positionnement (GPS, centrale inertielle, odomètre). Goulette *et al.* [Goulette et al., 2006], Yu *et al.*[Yu et al., 2007] et Alshawa *et al.* [Alshawa et al., 2009] ont développé des plateformes qui suivent ce même principe (figure 1.4). La localisation fonctionne par fusion de données GPS et centrale inertielle en temps réel. Avec l'information de navigation et les données des scanners laser, ils génèrent des nuages de points 3D de l'environnement traversé. La plateforme de Yu *et al.* est dédiée à la cartographie précise de routes tandis que celle de Alshawa *et al.* est conçue pour la génération de nuages de points de scènes urbaines.

La principale difficulté pour les systèmes mobiles en mode continu vient de la nécessité d'avoir une localisation précise de la plateforme pour pouvoir géo-référencer les données géométriques. Contrairement au mode « *Stop&Go* » qui utilise la localisation comme initialisation et qui peut recaler les scans après l'acquisition, il n'est généralement pas possible de faire de même en mode continu. Par contre, les systèmes en mode continu peuvent numériser un site en un temps beaucoup plus court qu'un système en mode « *Stop&Go* » (le mobile n'ayant pas besoin de s'arrêter).

L'avantage de la méthode « *Stop&Go* » réside en l'obtention de données très précises car l'acquisition se fait en statique. En mode continu, il faut tenir compte de la précision de

FIGURE 1.4 – A gauche, système de numérisation LARA3D de 2005 de l'équipe de Goulette *et al.* (source [Abuhadrous, 2005]). Au milieu, plateforme d'acquisition de l'équipe de Yu *et al.* (source [Yu et al., 2007]). A droite, plateforme de l'équipe d'Alshawa *et al.* (source [Alshawa et al., 2009]).

la localisation, des vibrations dues au déplacement du véhicule ce qui génère des données d'une qualité inférieure. Le mode « *Stop&Go* » génère de nombreuses zones peu denses entre les positions fixes d'acquisitions contrairement au mode continu qui permet d'avoir une densité à peu près équivalente tout au long de la trajectoire. L'équipe d'Asai *et al.* a intégré en 2005 [Asai et al., 2005] les deux modes « *Stop&Go* » et continu dans une seule plateforme de numérisation. Ces travaux ont confirmé qu'utiliser les deux modes permet de réduire les portions non mesurées du mode « *Stop&Go*.

Dans ce que nous venons de voir, nous nous sommes limités aux systèmes basés sur la technologie LiDAR terrestre. Il existe aussi de nombreux systèmes de numérisation fixes et mobiles basés sur d'autres technologie d'acquisition. La plus utilisée est la stéréovision. Nous pouvons prendre l'exemple des travaux de Pénard *et al.* [Pénard et al., 2005] qui reconstruisent des façades de bâtiments à partir d'images haute résolution prises à bord d'un véhicule. De même, Pollefeys *et al.* [Pollefeys et al., 2008] construisent un modèle texturé de l'environnement en temps réel à partir d'images de caméras vidéos. Ils utilisent un GPS et une centrale inertielle pour le géo-référencement des données et seulement les images pour générer la géométrie de la scène et les textures. La différence entre la technologie basée image et la technologie basée laser, c'est que l'on peut avec les LiDARs obtenir directement en temps réel des données denses très précises. Pour un état de l'art détaillé de ces systèmes, le lecteur peut se référer à [Ridene, 2010].

1.2 Description de trois plateformes d'acquisition

Nous décrivons dans cette partie trois plateformes de numérisation dont les données ont été utilisées au cours de cette thèse.

1.2.1 Description de la plateforme d'acquisition LARA3D

Le Centre de Robotique CAOR de MINES ParisTech travaille depuis 1998 sur un prototype de véhicule pour valider en situation réelle ses travaux sur la perception avancée et la fusion multi-capteurs. Les premiers travaux ont permis de transformer le véhicule en système de cartographie mobile avec l'ajout d'un capteur LiDAR placé de façon à numériser l'environnement pendant le déplacement [Abuhadrous, 2005]. Le véhicule prototype a alors été baptisé LARA3D. Depuis 2003, de nombreuses études ont été menées sur ce prototype et une équipe MMS pour « *Mobile Mapping System* » a été montée au sein du laboratoire pour continuer les recherches sur la cartographie 3D. Après les travaux d'Abuhadrous *et al.*, la thèse de Xavier Brun a porté sur la modélisation 3D à partir de données LARA3D [Brun, 2007]. Cette étude a servi de point de départ pour nos propres travaux.

Le véhicule LARA3D est configuré pour accueillir de nombreux capteurs et en particulier des caméras vidéos et des appareils photos, un ou plusieurs radars, un ou plusieurs LiDARs, une centrale inertielle, un GPS, des odomètres, des capteurs d'angle au volant, etc. Il comporte également un ordinateur embarqué équipé du logiciel RTMAPS, qui permet le traitement de données datées et synchronisées provenant en temps réel des capteurs embarqués. L'un des avantages de ce logiciel réside dans l'enregistrement des données datées et synchronisées de manière à pouvoir effectuer une relecture de ces dernières pour des tests tout en simulant les mêmes conditions au moment de l'acquisition. Ces tests permettent le prototypage d'applications qui seront par la suite directement embarquées sur le véhicule.

Le véhicule LARA3D a subi de nombreuses modifications depuis ses débuts avec principalement trois versions majeures, en 2005, en 2007 et en 2010. La figure 1.4 de gauche montre le prototype dans sa version 2005. Nous décrivons ici le système dans sa version 2010 tel qu'il a été utilisé lors de nos dernières campagnes d'acquisitions, même si nous faisons par la suite référence à des données produites par d'anciennes versions de LARA3D. La figure 1.5 présente, à gauche, une photo de la version 2010 du prototype LARA3D et à droite une photo avec un gros plan de quelques capteurs utilisés lors des acquisitions.

Voici la liste exhaustive des capteurs présents sur la version 2010 de LARA3D :
- 2 récepteurs GPS (Trimble Ag132 et GPS5012 Wi-Sys configurés en mode DGPS EGNOS) ;
- 2 centrales inertielles (Crossbow VG600 et Crossbow IMU440) ;
- 2 odomètres (roue droite et gauche : prototype du CETE Normandie Centre) ;
- 2 scanners laser (2 SICK LMS 221) ;
- 2 appareils photos (2 CANON EOS 5D) ;
- 1 caméra (Pike F421C).

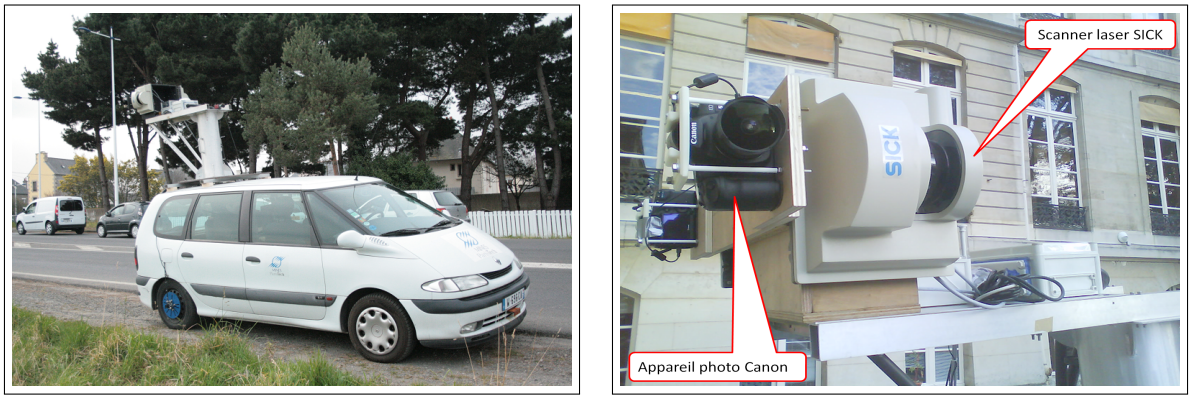


FIGURE 1.5 – A gauche, photo de la version 2010 du prototype LARA3D. A droite, détail sur quelques capteurs présents sur la plateforme embarquée sur le toit.

Nous pouvons distinguer deux catégories dans cet ensemble de capteurs : les capteurs de localisation (GPS, centrale inertielle et odomètre) et les capteurs d'imagerie (scanner laser, appareil photo et caméra). Cette distinction est faite en tenant compte de l'usage de ces capteurs dans notre application. Nous allons voir dans un premier temps la partie localisation du système qui sert de brique indispensable à la génération de données 3D.

Localisation du véhicule

Les systèmes de positionnement par satellite, tels que le GPS (le système américain) ou le GLONASS (le système russe), permettent pendant l'acquisition avec la plateforme mobile, la détermination de sa position, de sa vitesse ou encore de donner une information du temps absolu quel que soit son emplacement sur le globe terrestre, pour autant que les signaux des satellites soient disponibles. En effet dans le cas de la navigation terrestre, elle est souvent compromise à de nombreux endroits, notamment en ville. Pour améliorer cela, il est nécessaire d'adjoindre au système de positionnement par satellite un système de navigation inertiel, centrale inertielle ou INS pour « *Inertial Navigation System* ». Une centrale inertielle à six degrés de liberté (comme la Crossbow VG600) se compose de trois accéléromètres et de trois gyroscopes qui mesurent l'accélération et la vitesse angulaire du véhicule dans chacune des trois dimensions à une fréquence d'échantillonnage assez élevée (ici 84 Hz).

Les avantages de l'intégration GPS/INS sont plus qu'une amélioration de la précision de la localisation. Par exemple, les solutions d'INS peuvent être utilisées pour identifier et corriger le saut des cycles GPS, dû au blocage ou la perturbation du signal en milieu urbain. L'utilisation de l'INS pour établir des liens entre les données GPS, en cas d'anomalies, apporte une amélioration significative de la qualité du positionnement. Le filtrage de Kalman fournit un outil puissant pour créer la synergie entre deux capteurs de navigation - GPS et INS - puisqu'il peut tirer profit des avantages et des caractéristiques des deux systèmes pour fournir un système intégré de navigation présentant une performance supérieure à celle de l'un ou l'autre des sous-ensembles de capteurs pris séparément.

La localisation utilisée par le prototype LARA3D est un filtre étendu de Kalman pour intégrer un GPS différentiel (DGPS) avec une centrale inertielle de précision moyenne. Le résultat de ce filtre est la position et l'orientation du véhicule à une fréquence de 10 Hz [Abuhadrous, 2005].

Pour la précision de la localisation, nous devons distinguer deux types : la précision absolue et la précision relative. La première est liée à un biais systématique affectant toutes les données. La précision relative est liée à l'erreur de localisation une fois enlevée l'erreur absolue. La précision absolue est principalement liée au GPS, et la précision relative aux capteurs inertiels et odométriques. Il est très difficile d'avoir une mesure précise de ces erreurs car elles dépendent de nombreux facteurs : précision des capteurs (GPS ou DGPS, centrale inertielle, odomètre, etc.), paramètres du filtre d'intégration, environnement (urbain, péri-urbain, rural, etc.). Nous ne traitons pas la problématique d'estimation d'erreurs de la chaîne d'acquisition de la plateforme LARA3D par contre nous verrons dans le chapitre 2 une estimation plus précise de l'erreur mesurée sur les nuages de points produits en les comparant avec des données produites par un scanner laser fixe. Le lecteur intéressé par le sujet de l'erreur globale pourra se référer aux travaux de [Alshawa et al., 2009].

En plus de la position et de l'orientation, l'information de la datation des données est très importante à chaque occurrence : à chaque point de la trajectoire est associée une date. Les autres capteurs du véhicule sont aussi horodatés ce qui permet ensuite avec synchronisation des datations de faire un recalage dans le temps des données des capteurs. La datation de notre prototype LARA3D se fait de manière logicielle avec RTMAPS ce qui nous donne une précision d'environ 1 ms. A 130 km/h, une erreur de datation de 1 ms peut entraîner directement une erreur de localisation de 3.6 cm [1] le long de la trajectoire.

Génération de nuages de points

Une fois le véhicule localisé, c'est-à-dire une fois sa position, son orientation et la datation de ses données connus, il est alors possible de géo-référencer toutes les données issues des autres capteurs de cartographie, c'est-à-dire les données des caméras, appareils photos et LiDARs. Pour cela, il faut synchroniser la datation des données de cartographie avec la datation de la localisation, avec si nécessaire une interpolation entre deux positions connues.

Les scanners lasers de LARA3D

De 2003 à 2008, nous avons utilisé un scanner laser « LD Automotive » de la société IBEO pour numériser l'environnement proche de la voiture. Le scanner IBEO balaye, dans un plan, une zone angulaire de 270 °. C'est ce scanner qui a été utilisé pour les versions 2005 et 2007 de LARA3D. En 2009, nous avons fait l'acquisition de scanners laser SICK LMS221. Chaque SICK balaye une zone de 180 °, c'est pourquoi nous en utilisons deux placés dos à dos dans la version 2010 de LARA3D comme sur la figure 1.6. Ces deux capteurs sont des scanners laser qui fonctionnent sur le temps de vol, c'est-à-dire qu'ils mesurent le temps de parcours de l'onde entre son départ et son retour et en déduisent

1. $\frac{130}{3.6} \times \frac{1}{1000} \approx 0.036\ m$

une mesure de distance. Le LiDAR est généralement doté d'une tête rotative et fournit à chaque position angulaire durant le balayage circulaire la distance à l'obstacle le plus proche. Nous noterons que l'IBEO fournit environ 10 000 points à la seconde avec un écart-type d'environ $5-6$ cm alors que les deux SICK fournissent environ 30 000 points à la seconde avec un écart-type de l'ordre de $2-3$ cm. Le changement de scanner a ainsi permis une amélioration de la densité et de la précision des nuages de points produits par LARA3D.

FIGURE 1.6 – Plateforme de numérisation du prototype LARA3D avec scanners SICK.

Le géo-référencement des points laser

Les scanners lasers sont placés de telle sorte que leur plan d'acquisition soit perpendiculaire à la trajectoire du véhicule comme sur la figure 1.7. A chaque balayage circulaire, les scanners renvoient une coupe transversale de l'environnement traversé. Nous appelons profil l'ensemble des points d'une coupe transversale, c'est-à-dire les points d'une rotation de la tête laser. Avec les profils recalés les uns à la suite des autres suivant la trajectoire, nous avons une représentation discrète, une numérisation de l'environnement sous forme de nuage de points. Les points dont la distance à la tête laser est inférieure à une certaine distance d_{min} (par exemple les points captés sur le véhicule) ou supérieure à une certaine distance d_{max} (par exemple les ondes envoyées dans la direction du ciel qui n'ont pas eu de retour), ne sont pas conservés.

En réalité, le nuage de points est produit sous forme hélicoïdale. c'est-à-dire que pour chaque mesure du scanner, la donnée est précisément datée (avec un écart-type inférieur à 0.1 ms) et synchronisée avec les informations de localisation pour être géo-référencée. Il est nécessaire d'effectuer deux transformations, une pour passer du repère du scanner laser au repère du véhicule et une deuxième pour passer du repère véhicule au repère monde (repère géo-référencé). La première transformation est constante car le laser est fixé de manière

rigide au véhicule : un étalonnage permet d'obtenir les paramètres de cette transformation. La deuxième transformation pour passer du repère véhicule au repère monde correspond à une matrice de transformation R et un vecteur de translation T qui dépendent de la date d'acquisition t du point laser. Les coordonnées 2D (angle, distance) d'un point laser dans le repère laser sont ainsi transformées en coordonnées 3D dans le repère monde. Nous construisons alors un nuage de points 3D comme sur la figure 1.8 de l'environnement traversé par le véhicule.

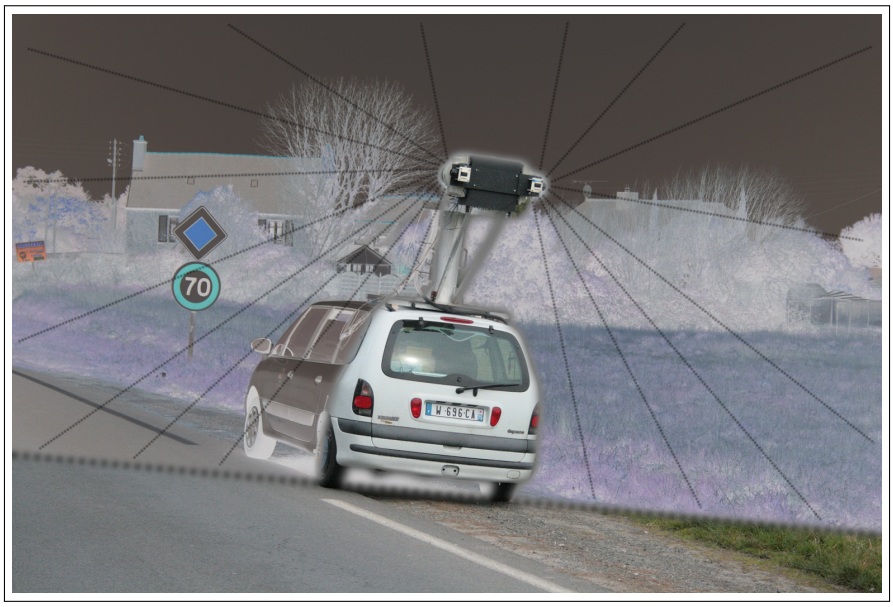


FIGURE 1.7 – Numérisation d'une scène par coupes transversales.

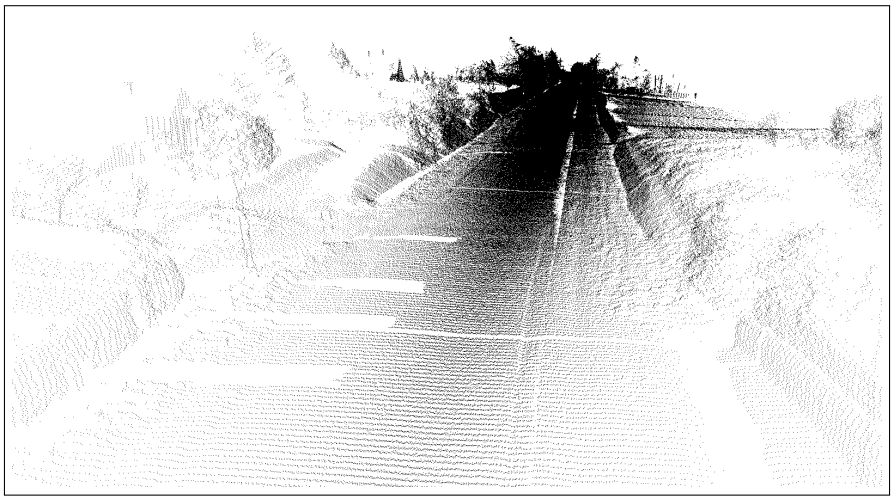


FIGURE 1.8 – Nuage de points 3D obtenu après géo-référencement.

Dans sa version 2010, le véhicule LARA3D peut produire 30 000 points à la seconde, ce

qui donne pour une vitesse d'acquisition de 20 km/h environ 5 400 points par mètre [1], donc environ 5 millions de points par kilomètre. Nous conservons seulement les points dont la distance mesurée par le LiDAR est comprise entre les deux paramètres d_{min} et d_{max}. Avec une valeur de $d_{min} = 1\ m$ et de $d_{max} = 30\ m$, nous obtenons des nuages de points compris entre 2 et 3 millions de points par kilomètre.

Ainsi, d'après la classification des systèmes mobiles terrestres d'acquisition que nous avons vu, LARA3D est une plateforme qui numérise en mode continu se basant sur un système de localisation par capteurs GPS et centrale inertielle.

L'objectif de LARA3D est de produire des données en temps réel, c'est-à-dire avec des temps de traitement similaire aux temps d'acquisition. Suivant le diagramme fonctionnel 1.9, le scanner laser génère des données à une fréquence de 10 Hz et les données de sortie sont produits à la même fréquence. Les différentes fonctions intermédiaires sont la localisation, le géo-référencement, la triangulation, la colorisation et la texturation. Les données de sortie du système sont de plusieurs sortes : des nuages de points, des nuages de points colorés, des maillages et des modèles 3D texturés.

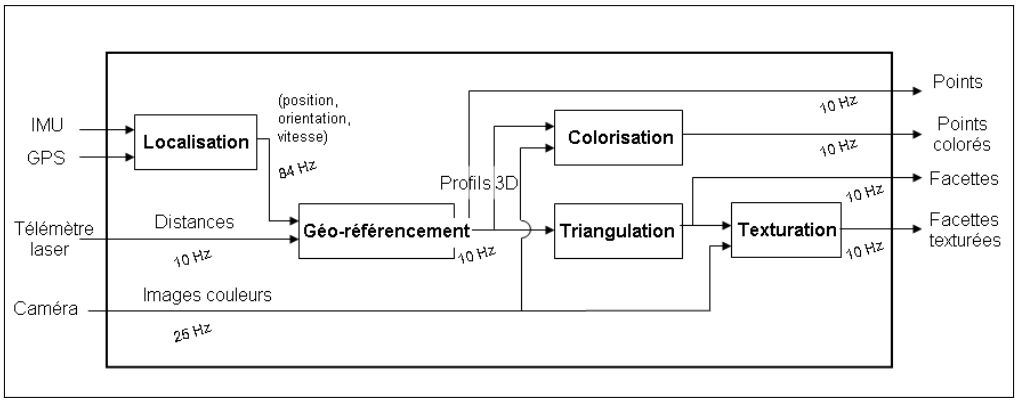


FIGURE 1.9 – Diagramme de production de données temps réel pour LARA3D.

1.2.2 Description de la plateforme Stereopolis

Stereopolis est une plateforme d'acquisition mobile initialement basée vision développée par l'IGN. Elle a subi de nombreuses modifications depuis son lancement en 2002. La version 2 dispose d'un système de localisation très précis Applanix POS-LV assurant une précision absolue inférieure à 25 cm pour une minute de masquage GPS. Pour la cartographie, le prototype dispose de 10 caméras en configuration omnidirectionnelle ainsi que de deux scanners lasers Riegl. Ces derniers possèdent une couverture de 80 ° et sont placés dos à dos comme sur la figure 1.10. Le système est globalement très similaire au système LARA3D avec une capture en mode continu des données laser. La figure 1.11

1. $\frac{30000}{\frac{20}{3.6}} \approx 5400\ pts/m$

présente un exemple de nuage de points produit par le système Stereopolis.

FIGURE 1.10 – Plateforme d'acquisition Stereopolis v2 de l'IGN.

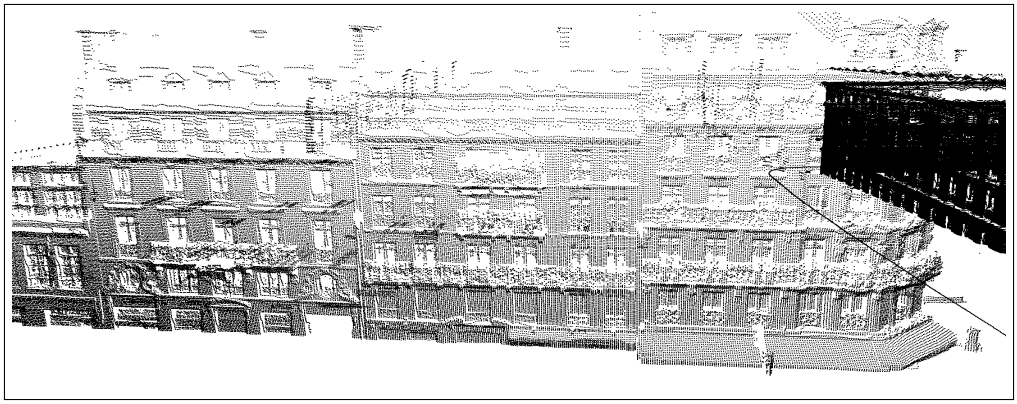


FIGURE 1.11 – Exemple de nuage de points produit par le système Stereopolis.

1.2.3 Description de la plateforme MINES_Rover

Au laboratoire de robotique de MINES ParisTech, une autre équipe travaille sur la cartographie d'environnements. Dans le cadre du projet CAROTTE de l'Agence Nationale de la Recherche, ils ont développé un robot appelé MINES_Rover et équipé de deux scanners lasers. Un premier scanner balaye dans un plan horizontal afin de localiser en temps réel le mobile. Par correspondance de scans successifs, ils obtiennent une carte 2D de l'environnement telle que celle de la figure 1.12 [Hamzaoui et Steux, 2010]. Le robot s'arrête en différentes positions et un deuxième scanner balaye en 3D l'environnement à

l'avant du robot MINES_Rover. C'est la technique de numérisation « *Stop&Go* similaire à Allen *et al.* [Allen et al., 2001] consistant à arrêter le robot en certains points pour laisser le temps au deuxième scanner de faire un scan 3D complet. Ce processus d'acquisition permet d'obtenir un nuage de points d'intérieur comme sur la figure 1.13.

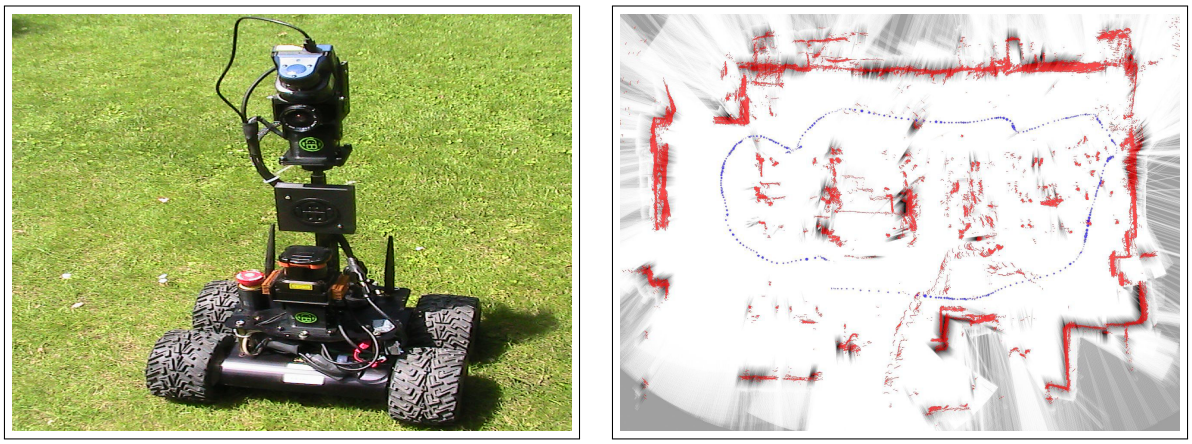


FIGURE 1.12 – A gauche, photo de la plateforme MINES_Rover. A droite, carte de localisation générée en intérieur.

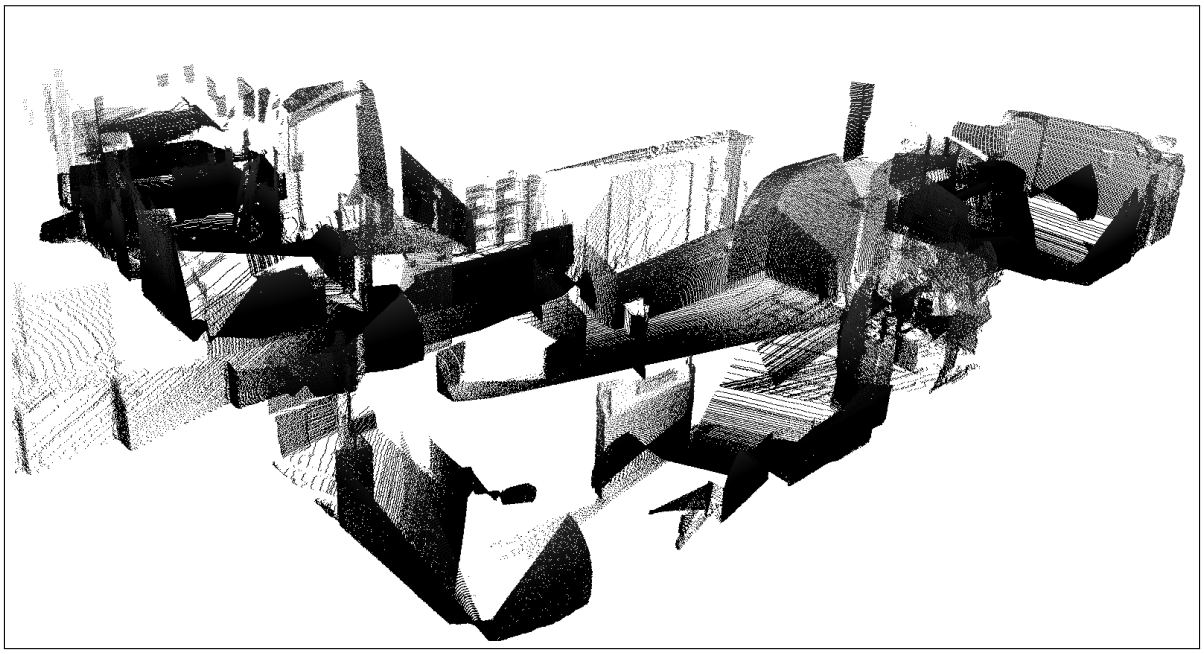


FIGURE 1.13 – Exemple de nuage de points produit par la plateforme MINES_Rover.

1.3 Modélisation 3D d'environnements à partir de nuages de points denses

A partir d'un nuage de points produit par un système terrestre fixe ou mobile, il existe une grande quantité de techniques de modélisation et de représentation de l'environnement. Il faut distinguer la notion de modélisation de la notion de représentation. La modélisation n'est qu'une étape pour produire une autre représentation de l'environnement numérisé. Il est possible de représenter un modèle directement sous forme de nuage de points, de maillage polygonal (par exemple une liste de triangles), sous forme volumique à base de voxels, etc. Le choix de la représentation de l'environnement va dépendre fortement de l'application. Dans notre cas, nous souhaitons construire à partir d'un nuage de points un maillage polygonal représentant la surface pour exporter les modèles 3D dans des moteurs de rendus traditionnels. L'étape de modélisation va donc consister à transformer un nuage de points en un modèle surfacique polygonal.

La plupart des approches de modélisation peuvent se regrouper sous trois catégories suivant la connaissance des données à traiter : une modélisation sans *a priori* (où il n'y a aucune connaissance sur les données), une modélisation basée sur des primitives géométriques (où l'on fait la supposition que l'environnement est composé de primitives simples comme des plans, des cylindres, etc.) et une modélisation basée sur des éléments métiers (où l'on essaye de reconnaître dans les données des éléments provenant de domaines architecturaux, routiers, industriels, etc.). Nous détaillons dans les sections suivantes ces trois catégories de modélisation.

1.3.1 Modélisation sans *a priori*

Pour la modélisation sans *a priori*, il n'y a aucune connaissance sur la surface à reconstruire. La représentation la plus courante pour ce type de modèle est le maillage polygonal. Les polygones les plus utilisés sont les triangles. Ce type de représentation est générique et permet de modéliser tout type d'objets complexes (comme les statues, les façades singulières, la végétation, etc.). Cela génère des modèles fidèles à la réalité mesurée, mais l'un des inconvénients est le volume des données à stocker généralement très important. A partir d'un nuage d'un million de points, un modèle reconstruit par maillage triangulaire comportera en moyenne deux millions de triangles. Il existe ensuite des techniques de simplification de maillage comme celle de [Garland et Heckbert, 1997] mais cette simplification se fait souvent au prise d'une certaine perte de qualité.

La plupart des systèmes de cartographie mobile basés LiDARs comme les travaux de [Früh et al., 2005], de [Brun, 2007] et de [Carlberg et al., 2008] construisent un maillage à partir du nuage de points par triangulation des points voisins. Ils construisent le modèle 3D en reliant les points profil par profil : deux points voisins dans un même profil sont reliés à un autre point voisin dans le profil suivant pour donner un triangle. La figure 1.14 présente à gauche le principe et à droite un exemple de maillage obtenu par la méthode de triangulation de points voisins de [Brun, 2007] sur des données de la plateforme LARA3D.

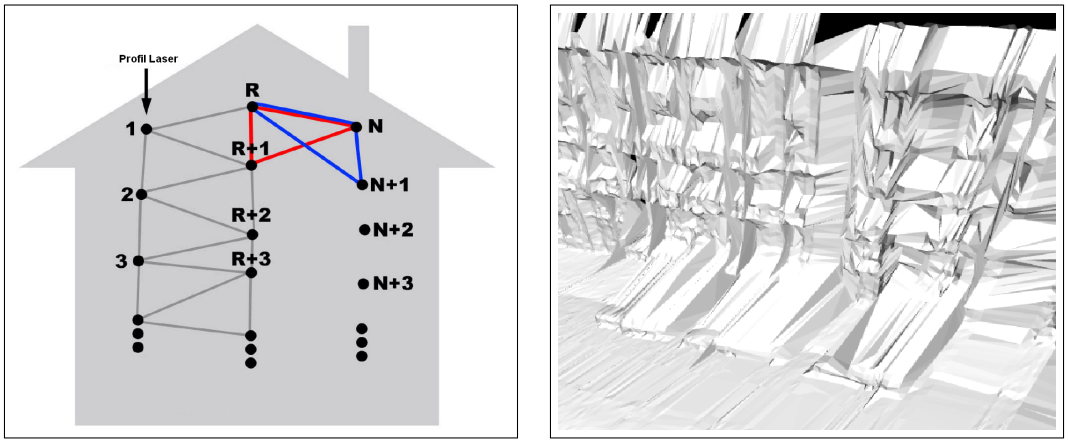


FIGURE 1.14 – A gauche, principe de modélisation par triangulation de points voisins par profil. A droite, le maillage obtenu par la méthode de triangulation de points voisins de [Brun, 2007] sur des données de la plateforme LARA3D (source [Brun, 2007]).

Cette méthode de triangulation est très rapide et permet de construire en temps réel un modèle au fur et à mesure de l'avancée du véhicule lors de l'acquisition. Mais dans les virages serrés, les profils peuvent se croiser comme sur la figure 1.15 et la triangulation profil par profil donne alors de mauvais résultats. D'autre part, cette méthode est dépendante de la configuration de la plateforme et en particulier de celle des lasers

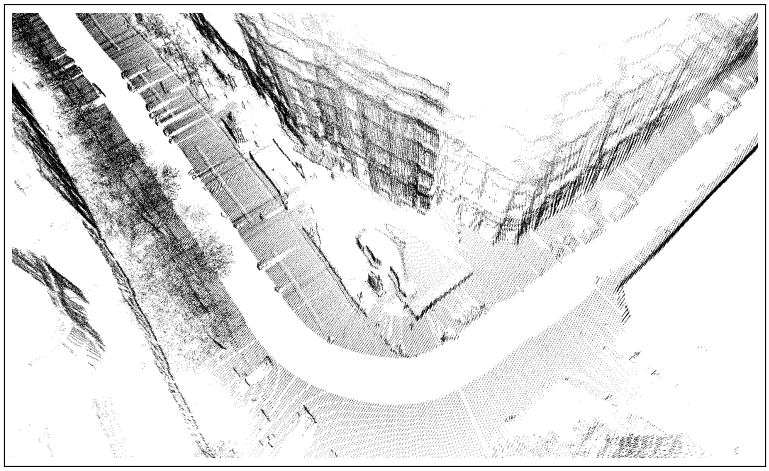


FIGURE 1.15 – Nuage de points produit par le système de cartographie mobile LARA3D dans un virage avec superposition et inversion de profils.

Dans la communauté de l'Informatique Graphique (IG), il existe de nombreuses techniques de modélisation pour passer d'un nuage de points quelconque à un maillage sans *a priori*. Cette étape est alors appelée reconstruction, triangulation ou maillage. La triangulation d'un nuage de points quelconque est un problème délicat, avec des algorithmes généralement très lents qui vont dépendre directement du nombre de points

du nuage. Nous verrons en détail au chapitre 4 les différentes méthodes de triangulation existantes. Nous pouvons citer une des méthodes les plus connues, l'algorithme « Cocone » [Amenta et al., 2002] qui consiste à créer tout d'abord une triangulation de Delaunay (ensemble de tétraèdres reliant les points) puis avec certaines heuristiques, à sélectionner les faces des tétraètres appartenant à la surface. La figure 1.16 montre un résultat de maillage pour un nuage de points d'un club de golf. Les méthodes de triangulation cherchent à obtenir un modèle le plus proche possible de la surface réelle. Mais la complexité des algorithmes de triangulation ne permet généralement pas de produire des modèles en temps réel.

FIGURE 1.16 – Triangulation d'un nuage de points avec l'algorithme « Cocone » (source [Amenta et al., 2002]).

1.3.2 Modélisation avec reconnaissance de primitives géométriques

Dans ce type de modélisation, nous supposons que l'environnement est constitué de primitives géométriques simples. L'idée est de calculer le meilleur plan, cylindre, boîte ou sphère par rapport à une portion ou même la totalité du nuage de points en entrée. La représentation de ces éléments peut alors se faire sous une forme paramétrique, ce qui permet de considérablement diminuer le nombre de données à conserver par rapport à une modélisation sous forme d'ensemble de polygones. Ce type de représentation permet aussi dans certains cas de diminuer le bruit d'acquisition.

Nous pouvons distinguer deux résultats pour ce type de modélisation. Les approches qui vont conserver uniquement les primitives géométriques détectées et celles qui vont construire une géométrie hybride avec un maillage et des primitives géométriques.

Pour la première catégorie, nous pouvons citer les travaux de [Boulaassal, 2010]. Il effectue tout d'abord une détection de primitives géométriques (ici des plans) sur un nuage de points de façade produit par un scanner laser fixe. A partir de cette segmentation, il modélise la façade suivant les contours des plans détectés. Nous observons sur la figure 1.17 le résultat de la modélisation à partir d'une portion d'un nuage de points laser. Avec ce type de modélisation, la façade peut être représentée par un nombre très restreint d'éléments : on peut passer d'un nuage d'un million de points à un ensemble de points de contour

d'un millier de points ce qui peut donner un facteur de réduction de 100. Par contre, les détails architecturaux complexes ne pourront pas être représentés. [Budroni et Bohm, 2009] obtiennent les mêmes résultats mais l'application est différente, il s'agit de la modélisation de scènes d'intérieurs. Ils segmentent le nuage de points en zones planes et le modèle final est composé uniquement des contours des plans détectés. La figure 1.18 montre le passage d'un nuage de points à un modèle CAO d'intérieur.

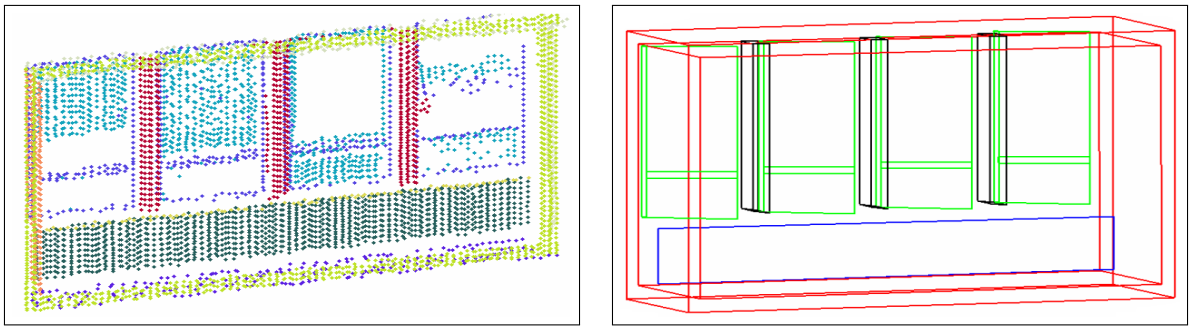


FIGURE 1.17 – A gauche, nuage de points d'une portion d'une façade segmentée en primitives géométriques. A droite, la modélisation à partir d'une détection des contours (source [Boulaassal, 2010]).

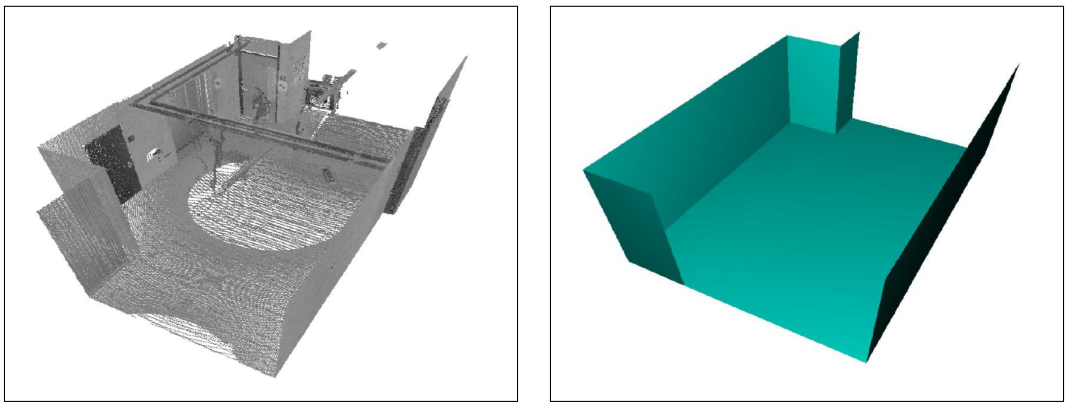


FIGURE 1.18 – A gauche, nuage de points d'intérieur. A droite, modélisation par détection de plans (source [Budroni et Bohm, 2009]).

L'autre approche consiste à construire une géométrie hybride avec une modélisation libre et une modélisation à partir de primitives géométriques. [Stamos et al., 2006] utilisent comme données d'entrée des images de profondeur d'un scanner laser fixe. Chaque image est segmentée en zones planes et non planes. Les images de profondeur sont ensuite recalées pour construire un nuage de points, et les zones planes sont fusionnées pour générer le modèle. Le modèle final est composé d'un maillage triangulaire et de primitives planes. La triangulation est basée sur l'algorithme « *Ball Pivoting Algorithm* » (BPA) de [Bernardini et al., 1999]. Ce type de représentation permet de diminuer de manière significative la taille du modèle final (surtout dans le cas de scènes urbaines qui présentent

de nombreuses zones planes) tout en restant fidèle à la réalité dans les zones complexes. La Figure 1.19 montre une scène d'intérieur modélisée avec cette méthode : nous voyons que les piliers sont modélisés par des plans et le plafond est modélisé par un maillage simple. La réduction du nombre d'éléments caractérisant le modèle peut varier d'un facteur 2 à un facteur 10 suivant les paramètres de la segmentation. Plus le facteur de réduction augmente, plus la perte de détails se fait sentir. Ainsi, une réduction d'un facteur 10 implique le remplacement de nombreuses zones par des plans.

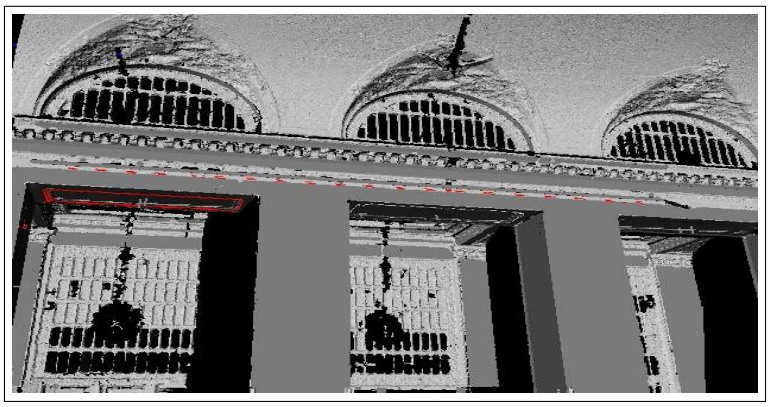


FIGURE 1.19 – Modélisation d'une scène d'intérieur avec détection des zones planes (piliers) et maillage pour les zones non planes (source [Stamos et al., 2006]).

[Lafarge et al., 2009] construisent aussi des modèles hybrides avec des primitives géométriques simples (plans, cylindres, cônes et tores), mais contrairement à [Stamos et al., 2006], ils utilisent directement le maillage comme donnée d'entrée. Suivant l'erreur accordée pour l'ajustement des primitives géométriques, la taille du modèle (nombre de données conservées pour représenter le modèle) peut aussi être réduite jusqu'à un facteur de 10. La Figure 1.20 montre le maillage d'une église qui a été segmenté avec cette méthode et nous voyons les zones planes qui ont été détectées et ajustées aux données dans le maillage représenté à droite.

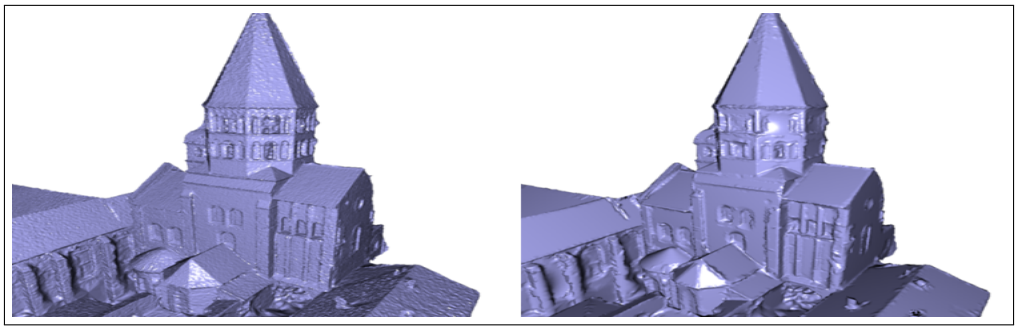


FIGURE 1.20 – A gauche, maillage original. A droite, maillage hybride composé de primitives géométriques simples et de triangles (source [Lafarge et al., 2009]).

1.3.3 Modélisation avec reconnaissance d'éléments métiers

A l'inverse des méthodes de modélisation précédemment décrites, la modélisation à partir d'éléments métiers implique une connaissance précise du domaine d'application des données en entrée. Par exemple, pour des nuages de points de scènes urbaines, nous pouvons considérer que ce sont des scènes architecturales avec un nombre limité d'éléments à reconnaître comme le toit, les fenêtres, les portes, etc. Ce type de représentation permet de réduire la taille des modèles et de contrôler parfaitement les éléments composant le modèle. Malheureusement, ce gain se fait au détriment de la capacité à modéliser des scènes complexes et variées. Il existe autant de modélisations possibles que de domaines d'applications. Nous allons voir ici quelques exemples de modélisation de nuages de points à partir d'éléments métiers.

[Pu et Vosselman, 2009] construisent des modèles de façades de bâtiments à partir de données laser terrestre. Ils utilisent un ensemble de connaissance sur les caractéristiques des toits, des portes et des façades pour les extraire comme sur la figure 1.21. Le modèle final est un ensemble de lignes polygonales représentant les bords des éléments détectés permettant de combler les éventuelles parties occludées.

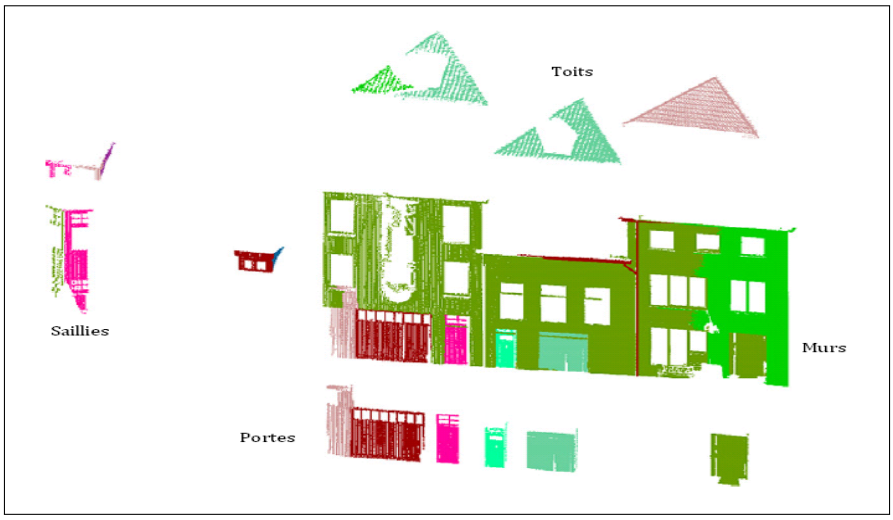


FIGURE 1.21 – Extraction d'éléments architecturaux dans un nuage de points (d'après [Pu et Vosselman, 2009]).

[Goulette et al., 2006] présentent une modélisation de l'environnement routier et urbain à partir de données laser captées par le système mobile LARA3D que nous avons présenté. Les scènes urbaines sont modélisées à partir de la détection de trois éléments : le sol, les façades et les arbres. Cette segmentation s'opère dans les profils laser simultanément à l'acquisition des données comme sur la figure 1.22.

[Rutzinger et al., 2010] détectent et modélisent des arbres dans des nuages de points provenant d'une plateforme mobile (figure 1.23). La réduction de la quantité de données est d'environ 95%.

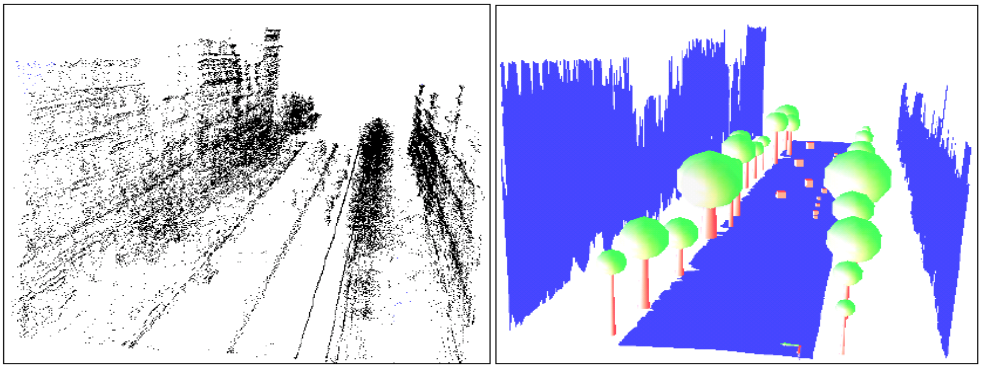


FIGURE 1.22 – A gauche, nuage de points de scène urbaine acquis par le système mobile LARA3D. A droite, modèle 3D généré par segmentation du sol, de la façade et des arbres (source [Goulette et al., 2006]).

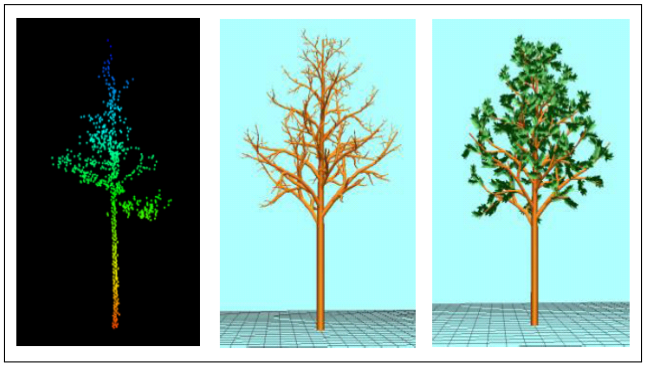


FIGURE 1.23 – A gauche, nuage de points d'un arbre provenant d'un système de cartographie mobile. Au milieu, modélisation du squelette. A droite, modèle de l'arbre enrichi et texturé (source [Rutzinger et al., 2010]).

Dans le milieu industriel, citons la thèse de Thomas Chaperon [Chaperon, 2002] qui présente une application où sont modélisé de grandes scènes industrielles par segmentation de lignes de tuyauterie ainsi que l'illustre la figure 1.24.

L'équipe de Budroni *et al.* ont modifié leur approche de modélisation de scènes d'intérieur entre 2009 [Budroni et Bohm, 2009] et 2010 [Budroni et Bohm, 2010]. En plus de la détection de plans dans les nuages de points, ils ont ajouté l'extraction de portes comme sur la figure 1.25. Leur approche qui était une modélisation avec reconnaissance de primitives géométriques est devenue une méthode basée sur la reconnaissance de modèles.

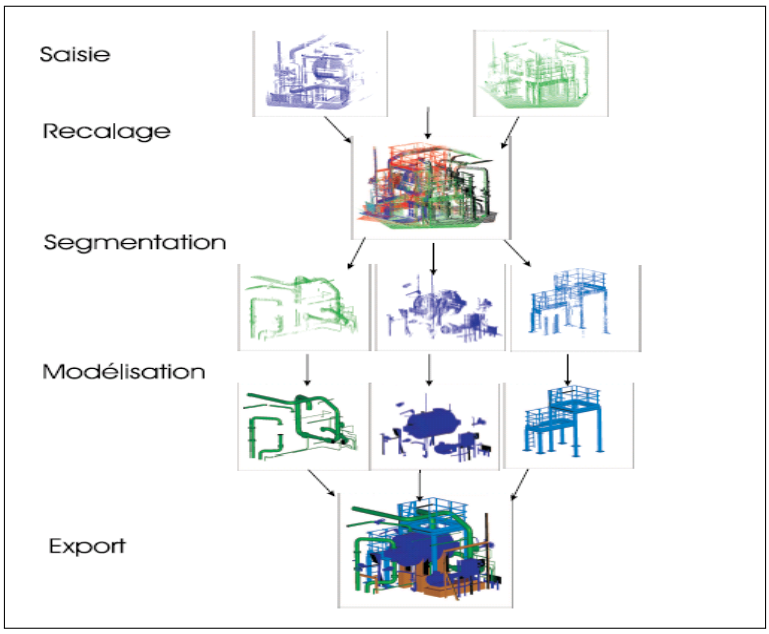


FIGURE 1.24 – Modélisation de scènes industrielles par extraction de lignes de tuyauterie (source [Chaperon, 2002]).

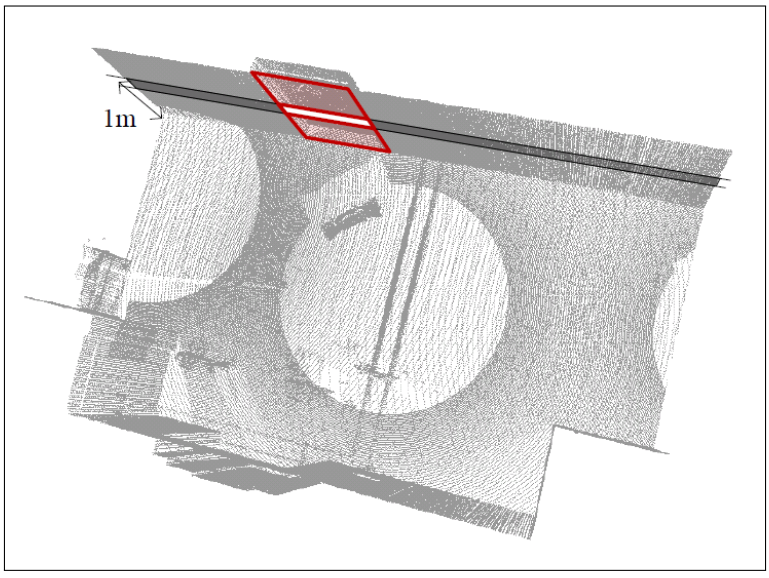


FIGURE 1.25 – Extraction de porte dans les nuages de points d'intérieur (source [Budroni et Bohm, 2010]).

1.4 Description de notre approche de modélisation 3D photo-réaliste

Comme nous l'avons vu, la plateforme mobile LARA3D génère plusieurs millions de points par kilomètre. Pour modéliser des scènes d'environnements urbains et routiers, nous avons choisi ce qui nous semble être le meilleur compromis entre la généricité, la fidélité et la compacité des modèles 3D avec une modélisation hybride sous la forme d'un maillage et de primitives planes. Nous avons cherché à obtenir une modélisation finale sous forme de primitives pour les surfaces planes et d'un maillage pour le reste du modèle, comme pour les travaux de [Stamos et al., 2006] et de [Lafarge et al., 2009]. Par contre, les données d'entrée sont différentes de celles des deux travaux précédents puisque ce ne sont ni des images de profondeur, ni des maillages, mais des nuages de points 3D provenant d'une plateforme mobile. L'objectif est de réduire la quantité d'information à stocker tout en modélisant des scènes complexes en milieu urbain et rural.

Nous avons vu que certaines méthodes de modélisation sont dépendantes du système d'acquisition comme les travaux précédents sur LARA3D [Brun, 2007] avec la triangulation en temps réel de points voisins par profils. Vu les progrès constants en matière de technologie pour les systèmes mobiles de cartographie, nous avons souhaité être indépendant du système de numérisation laser et donc être capable de modéliser un nuage de points quelconque sans utiliser les profils laser pour la triangulation. La difficulté avec cette approche vient du temps de traitement qui peut rendre difficile un traitement des données en temps réel.

Nous avons donc développé une chaîne complète de modélisation de manière à obtenir des modèles 3D photo-réalistes : calcul de normales dans les nuages de points, débruitage, segmentation par détection de zones planes, triangulation, simplification et texturation. La géométrie du modèle est alors construite à partir des données de LiDAR et les textures extraites des images des caméras.

Chapitre 2

Analyse de nuages de points de systèmes mobiles

Résumé

Ce chapitre détaille une analyse des principales caractéristiques des nuages de points produits par les systèmes mobiles LiDAR terrestres. Une mesure de la précision est exposée en comparant les nuages de points avec des données de scanner laser fixe. Enfin une comparaison est faite entre les données de plusieurs systèmes mobiles.

2.1 Analyse des nuages de points produits par LARA3D

Dans cette section, nous étudions les nuages de points produits par le prototype LARA3D. Pour cela, trois nuages de points ont été sélectionnés parmi les données disponibles provenant des différentes campagnes d'acquisitions. Nous avons un premier nuage de points *LARA3Dv2007_Soufflot* en milieu urbain avec des données peu denses et très bruitées, un deuxième nuage de points plus dense et moins bruité *LARA3Dv2010_Orsay* toujours en milieu urbain et enfin un troisième nuage de points *LARA3Dv2010_Etables* acquis en milieu rural à une vitesse plus grande de 45 km/h. Les noms des nuages de points ont comme préfixe la technologie d'acquisition (ici LARA3D), puis la version de cette technologie (ici version 2007 et 2010) et enfin la localisation du nuage de points (ici la rue Soufflot à Paris, le musée d'Orsay à Paris et enfin Etables-sur-Mer dans les Côtes d'Armor).

Tout d'abord, nous avons le nuage de points *LARA3Dv2007_Soufflot* présenté sur la figure 2.1. Ce nuage de points a été généré lors des acquisitions de décembre 2008 à Paris pour le projet TerraNumerica. La configuration du prototype LARA3D était proche de sa version 2007, c'est-à-dire avec le scanner IBEO. Seule la caméra a été modifiée entre la version de 2007 et les acquisitions de 2008 pour une caméra de meilleure résolution puisque nous sommes passés d'une caméra Marlin F046C de résolution 780x582 avec un angle de vue de 140 ° à une caméra Pike F146C de résolution 1 388x1 038 avec un angle de vue d'environ 140 °. Le nuage de points *LARA3Dv2007_Soufflot* est composé de 385 639 points acquis en 1 min 45 s par le véhicule roulant à une vitesse approximative de 5 km/h. La génération du nuage de points s'est faite en temps réel suivant la méthode décrite dans le chapitre 1.

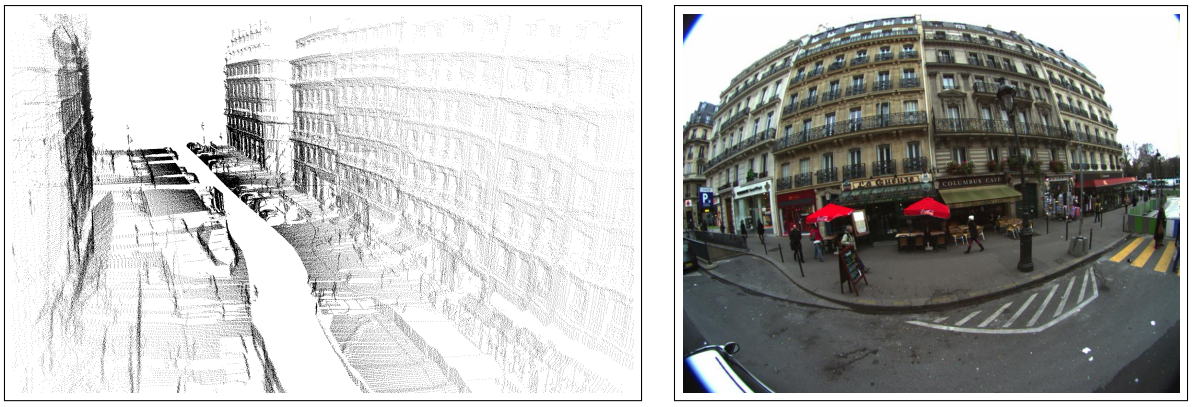


FIGURE 2.1 – A gauche, nuage de points *LARA3Dv2007_Soufflot* de la rue Soufflot à Paris. A droite, une image de la caméra Pike F145C prise pendant l'acquisition.

Ensuite, nous avons le nuage de points appelé *LARA3Dv2010_Orsay* de la façade Nord du musée d'Orsay à Paris présenté sur la figure 2.2. Ce nuage de points a été généré lors des acquisitions d'octobre 2009 à Paris toujours dans le cadre du projet TerraNumerica. La configuration du prototype était cette fois-ci très proche de la version 2010 que nous avons présentée, c'est-à-dire avec les deux scanners lasers SICK. Notons que les différences entre la version 2009 et 2010 n'ont pas d'influences sur nos travaux. Ce nuage de points

est composé de 2 440 416 points acquis en 3 min 10 s à une vitesse approximative de 5 km/h. Les images proviennent d'un appareil photo grand public Canon EOS 5D avec une résolution de 5 616x3 744 et un angle de vue de 180 ° en diagonale.

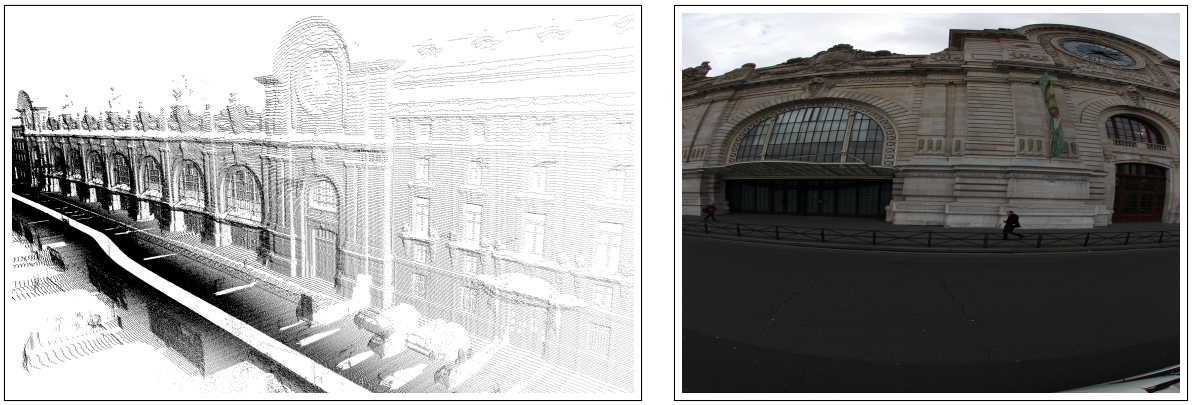


FIGURE 2.2 – A gauche, nuage de points *LARA3Dv2010_Orsay* de la façade Nord du musée d'Orsay à Paris. A droite, une image de l'appareil photo Canon EOS 5D prise pendant l'acquisition.

Enfin, nous avons aussi l'étude du nuage de points appelé *LARA3Dv2010_Etables* présenté sur la figure 2.3. Ce nuage de points a été généré lors des acquisitions de Mars 2010 sur la RD786 à côté d'Etables-Sur-Mer dans les Côtes d'Armor dans le cadre du projet DIVAS. La version du prototype était la version 2010. Ce nuage de points est composé de 3 540 911 points acquis en 4 min 41 s à une vitesse approximative de 45 km/h. Les images proviennent aussi de l'appareil photo Canon EOS 5D.

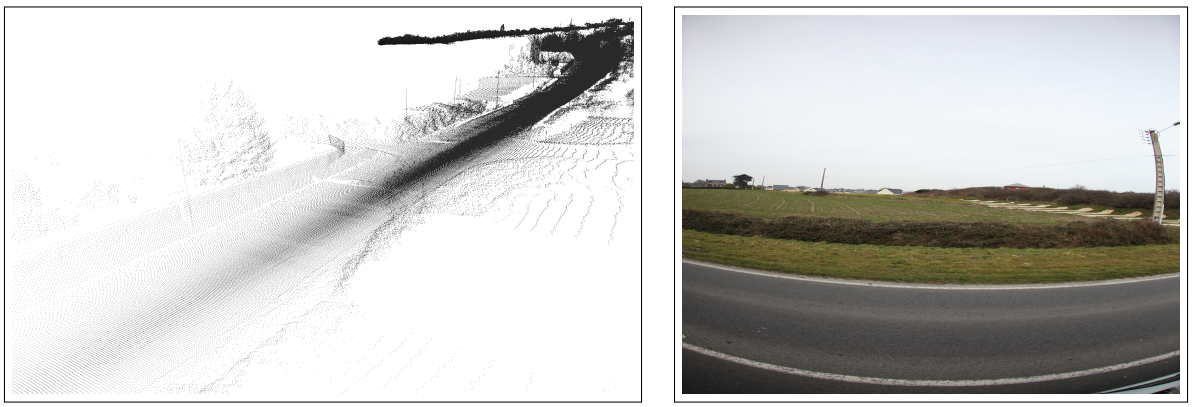


FIGURE 2.3 – A gauche, nuage de points *LARA3Dv2010_Etables* de la RD786 à côté d'Etables-Sur-Mer. A droite, une image de l'appareil photo Canon EOS 5D prise pendant l'acquisition.

2.1.1 Répartition des points dans les nuages 3D

Nous allons étudier les caractéristiques des nuages de points bruts (sans traitement) produits par LARA3D. Tout d'abord, nous voyons sur la figure 2.4 que les nuages de points ne sont pas uniformément répartis sur la surface capturée. En effet, l'espacement entre les points est plus grand dans les zones éloignées du véhicule que dans les zones proches du véhicule. Ceci est dû au fait que les lasers fonctionnent à résolution angulaire fixe et retournent ainsi des points voisins proches pour des objets proches du laser. La densité locale du nuage de points, c'est-à-dire le nombre de points par m^2 varie ainsi fortement dans le nuage de points. De plus, nous pouvons voir que la répartition locale des points est différente suivant deux directions : suivant le sens de la trajectoire du véhicule et suivant la direction orthogonale à la trajectoire (dans le sens des profils laser). Les nuages de points produits par LARA3D sont ainsi globalement hétérogènes et localement anisotropes (propriétés qui varient suivant la direction).

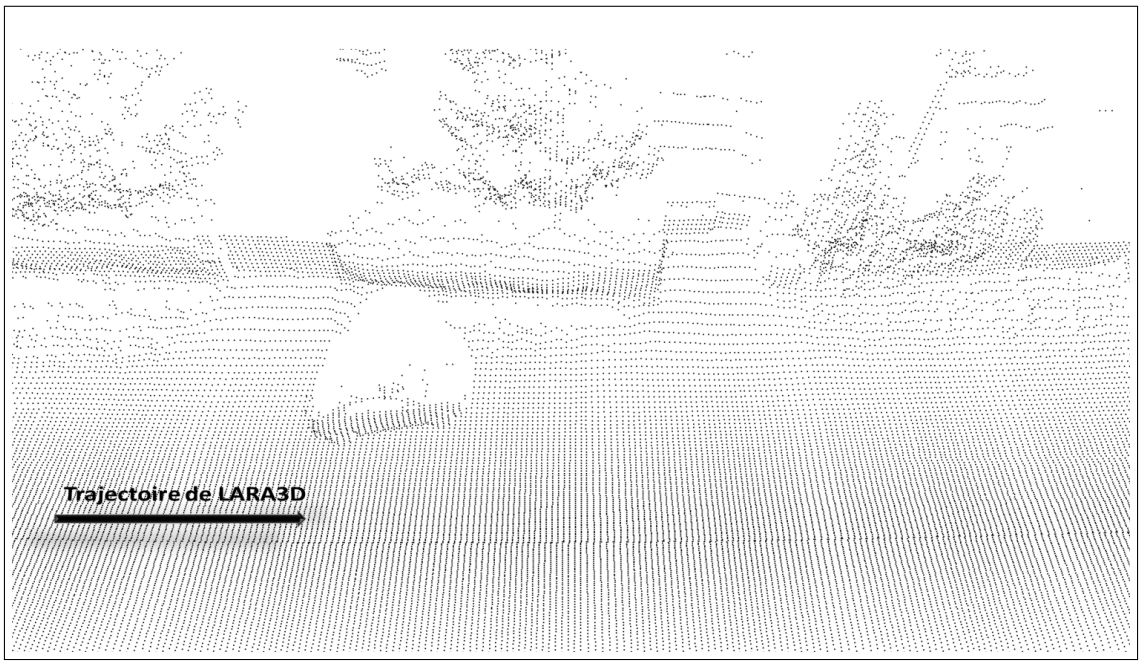


FIGURE 2.4 – Gros plan sur le nuage de points *LARA3Dv2010_Etables* pour montrer le caractère hétérogène et anisotrope du nuage de points.

Pour expliquer ce caractère hétérogène et anisotrope des données, nous devons revenir à la méthode de capture des données. Dans un profil, les scanners laser capturent des points avec une résolution angulaire fixe. Nous aurons donc des points voisins plus rapprochés à courte distance (par exemple près du véhicule) et en revanche, les points voisins seront plus espacés pour des objets captés à plus grande distance (par exemple les bords de la route ou les façades). Nous pouvons calculer la distance moyenne d_{profil} entre deux points voisins dans un profil laser en fonction de la résolution angulaire du scanner laser r_{laser} en radians, de la distance minimale d_{min} et maximale d_{max} des points conservés : $d_{profil} = \frac{d_{min}+d_{max}}{2} r_{laser}$. Par exemple, pour le nuage de points *LARA3Dv2010_Etables*,

nous avons choisi une distance minimale $d_{min} = 1\ m$, une distance maximale $d_{max} = 30\ m$ et une résolution angulaire des scanners SICK $r_{laser} = 1\,^{\circ}$, ce qui donne une distance moyenne entre points d'un même profil : $d_{profil} \approx 27.0\ cm$ [1].

Dans la direction de la trajectoire du véhicule, la distance entre points voisins va dépendre de la vitesse du véhicule, de la fréquence des lasers mais aussi de la courbure de la trajectoire et de la distance des points au laser. Pour simplifier l'étude, nous supposons que la différence de courbure de la trajectoire est nulle entre deux profils successifs et que les profils sont ainsi parallèles. La distance moyenne $d_{trajectoire}$ entre points de profils successifs va alors dépendre seulement de la vitesse $v_{vehicule}$ du véhicule (en m/s) et de la fréquence du laser f_{laser}, c'est-à-dire du nombre de tours par seconde de la tête laser : $d_{trajectoire} = \frac{v_{vehicule}}{f_{laser}}$. Toujours pour le nuage *LARA3Dv2010_Etables*, avec une vitesse du véhicule approximative $v_{vehicule} \approx 45\ km/h$ et une fréquence laser des scanners SICK $f_{laser} = 75\ Hz$, nous avons une distance moyenne entre points voisins d'un profil à un autre : $d_{trajectoire} \approx 16.6\ cm$ [1].

Dans le nuage de points, en supposant toujours que les profils successifs sont parallèles, la distance entre un point et son point le plus proche va approximativement varier entre 1.7 *cm* et 16.6 *cm* à cause de la résolution angulaire fixe des lasers et de la distance constante entre profils pour une vitesse constante du véhicule. Localement, la distance entre un point et son voisin le plus proche dans la direction du profil laser va varier entre 1.7 *cm* et 52.3 *cm* mais dans le sens de la trajectoire, cette distance avec son voisin le plus proche sera approximativement toujours identique à 16.6 *cm* pour une vitesse constante du véhicule. Ces deux remarques se vérifient visuellement sur le nuage de points de la figure 2.4.

Pour vérifier expérimentalement ces deux valeurs de résolution entre points voisins, nous avons calculé pour chaque point du nuage *LARA3D_Etables* la distance à son voisin le plus proche. Nous avons ensuite tracé l'histogramme de ces distances sur la figure 2.5. On peut tout d'abord constater le caractère non homogène de la répartition des points avec une distance moyenne entre points de 13.2 *cm* et un écart type important de 9.6 *cm*. Cet histogramme représente la distance d'un point à son voisin le plus proche dans le nuage de points. Ceci est différent des valeurs précédemment calculées, qui sont des distances moyennes d'un point à son plus proche voisin dans un profil pour d_{profil} et entre profils pour $d_{trajectoire}$. Or nous pouvons retrouver ces deux valeurs dans l'histogramme 2.5. Le deuxième pic qui se trouve à environ 15 *cm* correspond approximativement à la distance $d_{trajectoire}$ calculée. En effet, pour des points suffisamment éloignés du véhicule, le voisin le plus proche sera le voisin du profil précédant ou suivant et sera donc à une distance approximative de $d_{trajectoire} \approx 16.6\ cm$. Mais le premier pic qui se trouve à environ 5 *cm* ne correspond pas à la valeur attendue de d_{profil}. Ceci s'explique par le fait que nous avons implicitement supposé pour le calcul de d_{profil} une répartition uniforme des points entre d_{min} et d_{max}. Or, nous avons ici un nuage de points acquis en milieu rural à ciel ouvert, et donc la majorité des points se situent sur la route et ses abords, c'est-à-dire approximativement entre 1 et 5 m, ce qui nous donnerait $d_{profil} \approx 5.2\ cm$ et correspond

1. $\frac{\frac{1+30}{2}}{\frac{45}{\pi}} \times \frac{\pi}{180} \approx 27.0\ cm$
1. $\frac{3.6}{75} \approx 16.6\ cm$

bien à la valeur observée du premier pic. Ainsi les pics de l'histogramme 2.5 représentent le caractère anisotrope du nuage de points. Nous pourrions ainsi extraire des caractéristiques d'acquisition comme la vitesse moyenne du véhicule uniquement à partir du nuage de points.

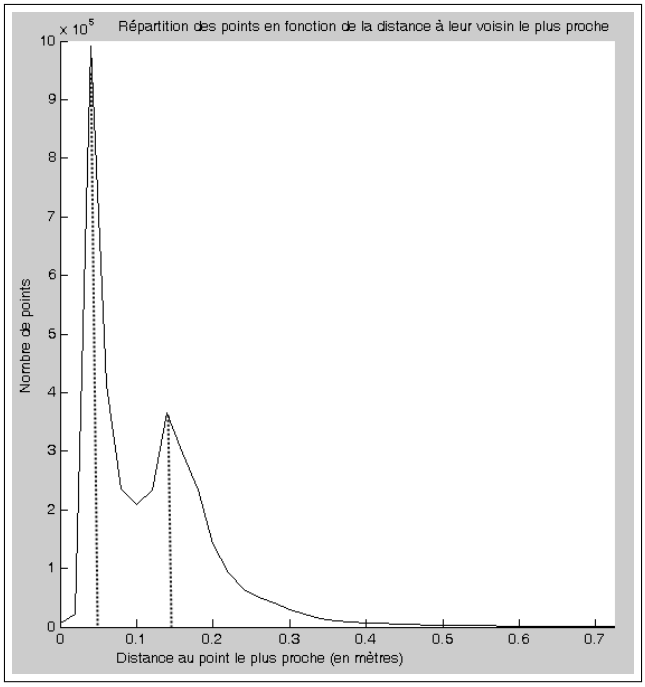


FIGURE 2.5 – Histogramme de la distance entre un point et son voisin le plus proche pour le nuage de points *LARA3Dv2010_Etables*.

Nous avons fait la même étude de répartition de points sur les nuages *LARA3Dv2007_Soufflot* et *LARA3Dv2010_Orsay* de la figure 2.6. Pour le premier, nous avons les valeurs théoriques de résolution des points $d_{profil} = 6.7\ cm$ et $d_{trajectoire} = 13.9\ cm$. Pour le second, nous avons $d_{profil} = 6.7\ cm$ et $d_{trajectoire} = 7.4\ cm$. Dans les deux cas, la vitesse du véhicule est de 5 km/h et la résolution des lasers de $0.25\,°$. Seules les fréquences des lasers sont différentes : dans le premier cas, l'IBEO fonctionne à une fréquence de 10 Hz et dans le second cas, les SICK sont à 18 Hz.

Nous avons tracé sur la figure 2.7 les histogrammes des distances au point le plus proche de ces deux nuages de points. Nous avons une distance moyenne entre points de 7.8 cm pour *LARA3Dv2007_Soufflot* avec un écart type de 6.3 cm et une distance moyenne entre points de 4.8 cm pour *LARA3Dv2010_Orsay* avec un écart type bien inférieur égal à 2.6 cm. Nous pouvons toujours distinguer les deux pics sur l'histogramme du nuage *LARA3Dv2007_Soufflot*. Par contre, il n'y a plus qu'un seul pic sur l'histogramme de *LARA3Dv2010_Orsay* et un écart-type très faible. Ceci est dû au fait que les deux résolutions moyennes de distances entre points dans les deux directions sont très proches comme nous l'avaient annoncé les valeurs théoriques $d_{profil} = 6.7\ cm$ et $d_{trajectoire} = 7.4\ cm$. La différence entre les deux histogrammes s'explique aussi par le fait que la façade du musée d'Orsay était plus proche du véhicule que les façades des bâtiments de la rue Soufflot.

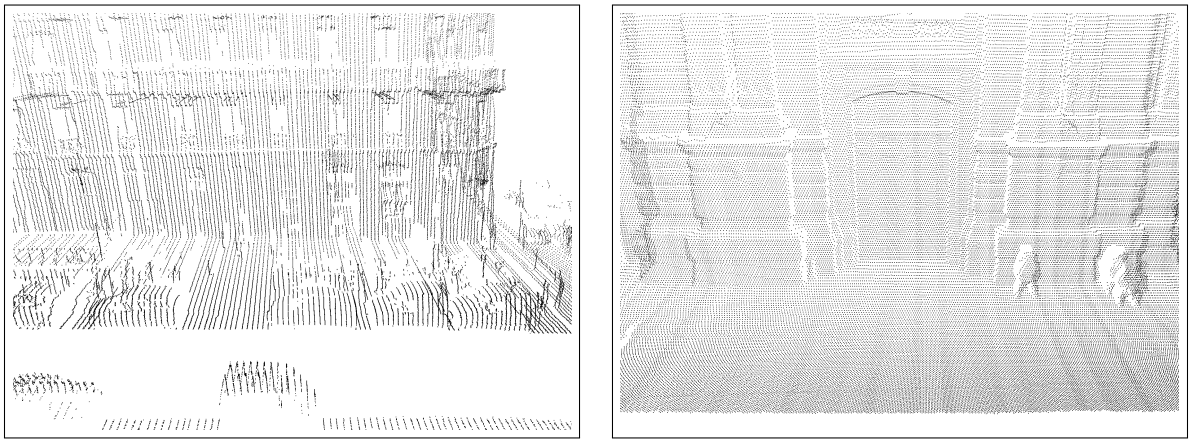


FIGURE 2.6 – A gauche, nuage de points *LARA3Dv2007_Soufflot*. A droite, nuage de points *LARA3Dv2010_Orsay*.

Le nuage de point *LARA3Dv2010_Orsay* reste tout de même hétérogène et localement anisotrope comme nous le voyons sur la figure 2.6 car la distance entre points voisins dans un profil laser va tout de même varier fortement.

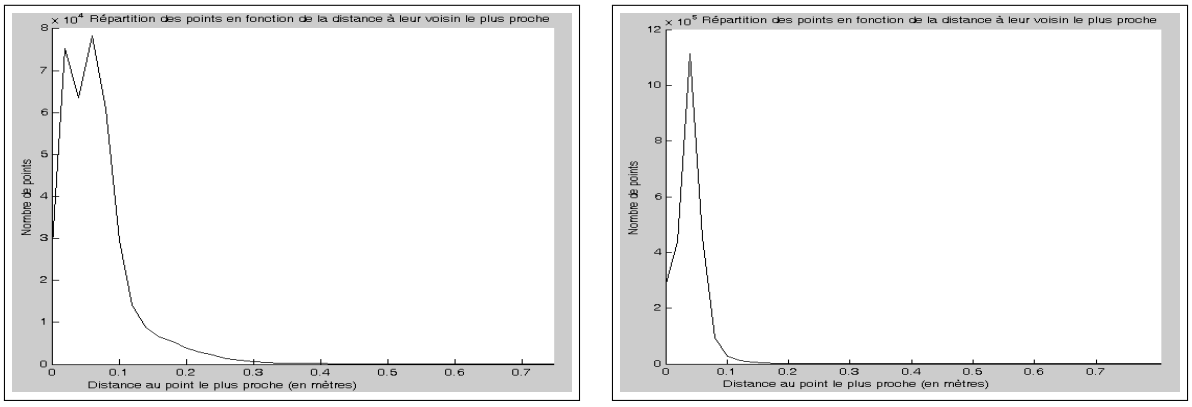


FIGURE 2.7 – A gauche, histogramme de la distance entre un point et son voisin le plus proche pour le nuage de points *LARA3Dv2007_Soufflot*. A droite, même histogramme pour le nuage de points *LARA3Dv2010_Orsay*.

Ainsi, dans le cas d'une répartition uniforme des points de l'environnement dans un profil entre d_{min} et d_{max}, il est possible de calculer une vitesse du véhicule optimale avec $d_{profil} = d_{trajectoire}$ afin d'obtenir une meilleure répartition des points sur la surface comme nous venons de le voir sur le nuage de points *LARA3Dv2010_Orsay* : $v_{vehicule} = \frac{d_{min}+d_{max}}{2} f_{laser} r_{laser}$. Ceci permet d'obtenir une répartition plus uniforme des points pour générer une meilleure reconstruction de la surface.

Nous pouvons définir la distance d_{mean} comme la moyenne des distances d'un point à son plus proche voisin. Cette distance est représentative de l'échelle du nuage de points. Pour

les nuages *LARA3Dv2007_Soufflot*, *LARA3Dv2010_Etables* et *LARA3Dv2010_Orsay*, cette distance varie entre 5 et 15 *cm*.

2.1.2 Précision des nuages de points

[Barber et al., 2008] donnent une analyse théorique pour identifier la provenance et l'importance des erreurs sur la position des points d'un nuage produit par un système de cartographie mobile. Ils concluent que cette erreur vient principalement de l'erreur de positionnement du véhicule par le système GPS/INS, mais aussi de la précision des points laser dans le repère laser (erreur sur la mesure du capteur LiDAR) et enfin de manière moins importante de la synchronisation des différentes données et des différentes imprécisions lors de l'étalonnage des capteurs. Avec des données tests produites par leur prototype StreetMapper dans deux conditions d'acquisition (zone résidentielle et zone industrielle), ils ont mesuré une précision finale moyenne des points entre 3 et 10 *cm*, distance moyenne des points entre leur position géo-réferencée du nuage et leur position réelle.

Nous avons vu deux précisions pour la localisation du véhicule : une précision absolue avec un biais identique sur tous les points de l'acquisition. Ce biais dépendra majoritairement du biais introduit par le capteur GPS. Ensuite, une fois l'erreur absolue enlevée, nous avons l'erreur relative de chaque point de la trajectoire. Nous pouvons garder ces deux définitions pour la précision d'un point d'un nuage de points 3D. La précision qui nous intéresse ici est surtout la précision relative. Pour supprimer l'erreur absolue d'un nuage de points, il est possible d'appliquer un recalage du nuage de points avec des données extérieures comme il est décrit dans la thèse de Taha Ridene [Ridene, 2010].

Pour notre prototype de cartographie LARA3D dans la version 2010, nous avons estimé approximativement l'erreur relative de chaque point à environ 5 *cm*. Dans les nuages produits par la version 2007 de LARA3D, cette erreur peut monter à $10 - 15$ *cm*. Pour vérifier ces estimations, nous avons fait une comparaison précise du nuage *LARA3Dv2010_Orsay* avec le nuage *FX_GX_Orsay* obtenu à l'aide de scanners fixes Trimble FX et Trimble GX dans le cadre du projet TerraNumerica. Ce nuage de points de la façade Nord du musée d'Orsay a été obtenu par recalage de données issues de plusieurs stations des scanners Trimble FX et Trimble GX (figure 2.8). La consolidation des nuages produits par les différentes stations ont été faites à l'aide de cibles posées manuellement. Les résultats de recalage donnent une précision relative de tous les points du nuage de l'ordre de 1 *cm*. Nous avons ainsi pris ce nuage comme référence pour mesurer la qualité du nuage de points produit par LARA3D dans la même zone. Pour mesurer la précision de notre nuage *LARA3Dv2010_Orsay*, nous avons fait un recalage à la main des données avec le nuage *FX_GX_Orsay* à l'aide du logiciel RealWorks de Trimble afin de supprimer l'erreur absolue sur le nuage. Nous avons ensuite conservé les parties de ces nuages qui se chevauchent et opéré une minimisation de la distance entre les deux nuages par ICP. Les parties recalées qui se chevauchent sont visibles sur la figure 2.9 avec en rouge le nuage *LARA3Dv2010_Orsay* et en vert le nuage *FX_GX_Orsay*. Le chevauchement se fait sur une longueur de 204 *m* avec une erreur moyenne de recalage de 17.7 *cm* calculée sur 245 892 points. Pour confirmer ces résultats, nous avons étudié les distances à la main entre

ces nuages de points à travers leur profils en z comme sur la figure 2.9. Nous avons mesuré des erreurs sur le nuage de points *LARA3Dv2010_Orsay* variant entre 0 et 30 *cm*. Nous avons obtenu que la précision relative des points ne varie pas de plus de 5 *cm* sur 5 *m*, ce qui nous assure la bonne cohérence des données au niveau local. Ainsi, nous avons une dérive de la précision des points du nuage le long de la trajectoire mais une bonne précision des points au niveau local. Cela permet d'assurer une modélisation possible à une échelle de 5-10 *cm* même si nous ne pouvons assurer une précision finale relative des points que de 20 *cm* en moyenne.

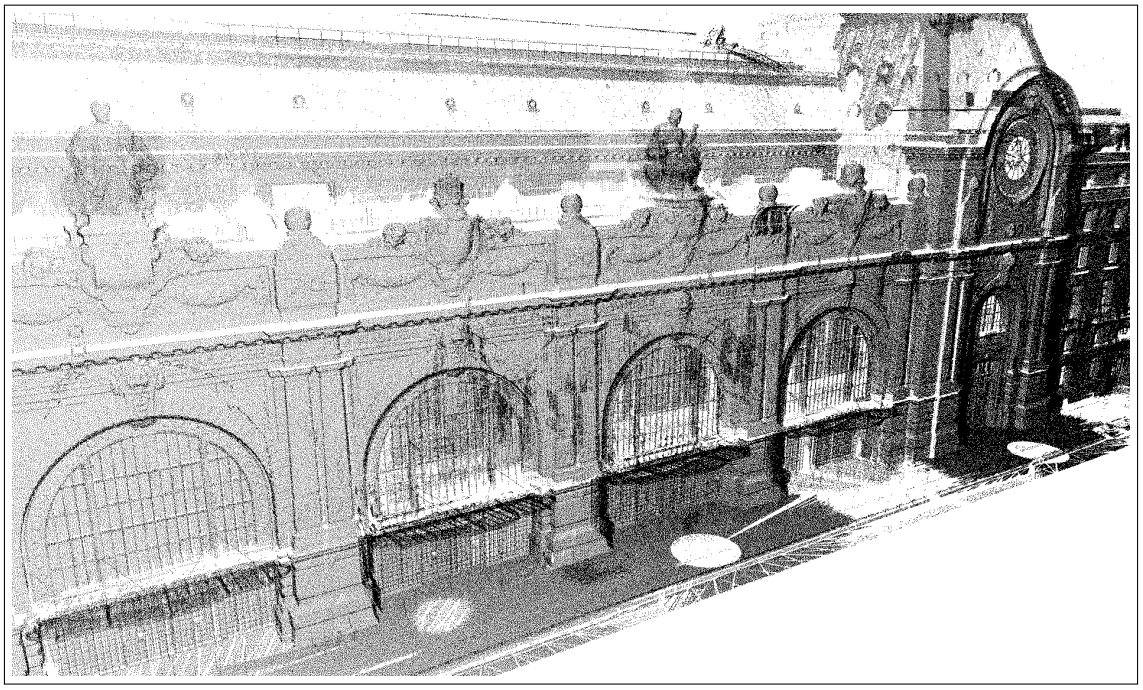


FIGURE 2.8 – Nuage de points *FX_GX_Orsay* obtenu à partir des scanners fixes Trimble FX et Trimble GX.

Nous pouvons aussi définir un troisième type d'erreur que nous allons retrouver au niveau local. En effet, si nous considérons un voisinage local et que le centre de ce voisinage est correctement recalé (en abolu et en relatif) alors il reste des erreurs sur les points principalement dues au bruit de mesure des capteurs laser. Cette erreur locale peut se mesurer en prenant un voisinage local qui correspond à un plan. Nous faisons l'hypothèse que la partie locale de la surface réelle est un plan. Si cette erreur de modélisation est négligeable devant l'erreur locale à mesurer alors nous pouvons en déduire l'erreur locale du nuage de points. Nous avons sélectionné à la main une partie des nuages *LARA3Dv2010_Orsay* et *LARA3Dv2007_Soufflot* pour analyser l'erreur locale. Nous avons dans les deux cas choisi une partie de la rue en rouge à gauche et à droite de la figure 2.10.

Pour cette étude, notre *a priori* est donc que ces deux parties sont planes (avec l'hypothèse que l'erreur de modélisation - rue plane localement - est négligeable devant l'erreur de mesure). Afin de mettre en exergue le bruit des capteurs, nous avons mesuré

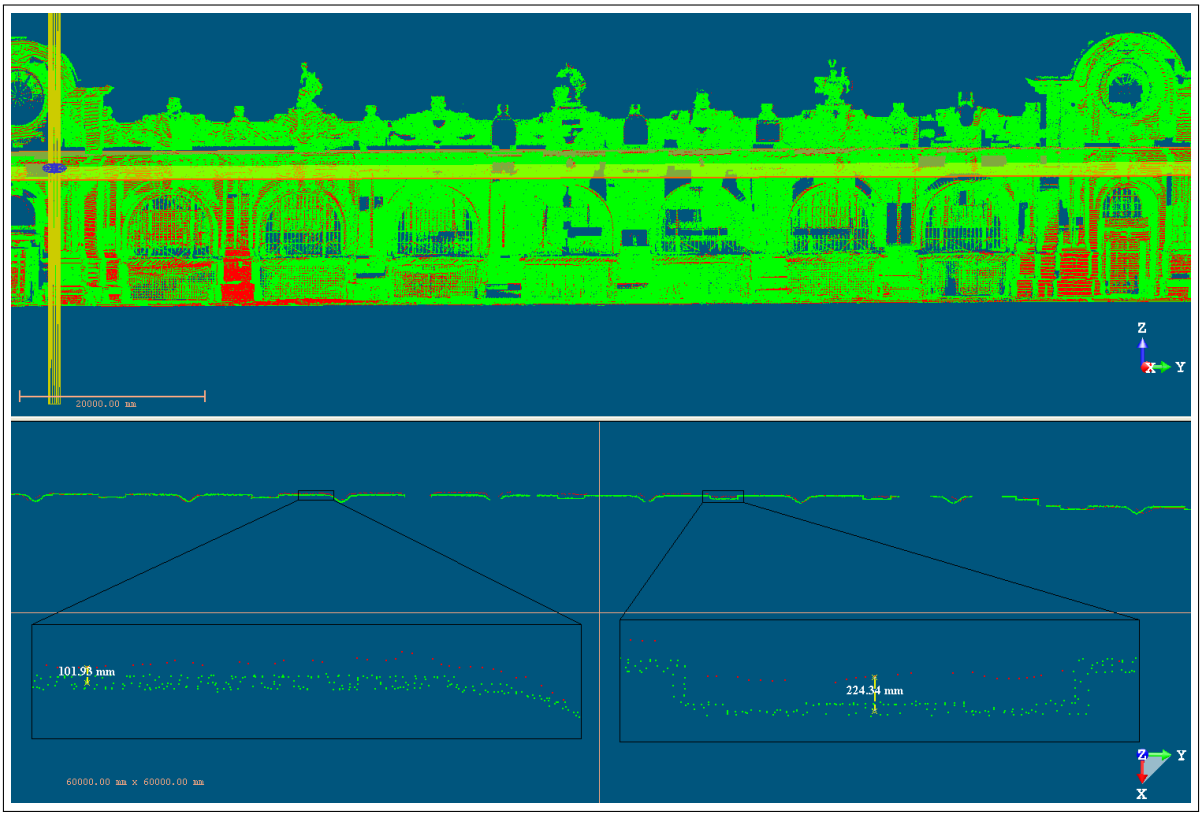


FIGURE 2.9 – En haut, recalage des nuages *LARA3Dv2010_Orsay* en rouge et *FX_GX_Orsay* en vert. En bas, coupe des nuages recalés avec un plan orthogonal à l'axe z et gros plan sur le calcul des deux erreurs sur le nuage *LARA3Dv2010_Orsay* en rouge.

les distances des points au plan et représenté les histogrammes de ces mesures sur la figure 2.11. Le résultat est qu'il apparaît dans les deux cas que l'erreur locale peut être modélisée par une gaussienne, ce qui nous permet de la considérer comme un bruit. Un écart type de 11 *mm* a été mesuré pour le nuage *LARA3Dv2007_Soufflot* et un écart-type de 4 *mm* pour le nuage *LARA3Dv2010_Orsay*. L'hypothèse de la surface réelle plane pourrait être remis en cause à la vue des écart-type obtenus.

Finalement, nous avons caractérisé trois types d'imprécision dans un nuage de points produit par une plateforme mobile. Tout d'abord, une erreur absolue sur tout le nuage de points principalement due à une erreur de géo-référencement du GPS. Ensuite, nous avons vu la notion d'erreur relative des points principalement due au caractère mobile de la technologie d'acquisition et avons mesuré cette erreur à environ 5 *cm* au maximum sur 5 *m* ou 30 *cm* au maximum sur 200 *m*. Enfin, nous avons étudié une erreur locale qui peut être modélisée comme une gaussienne d'écart-type environ 5-10 *mm* et provenant surtout du bruit du capteur laser. Ainsi, avec une distance moyenne entre points voisins de 4.8 *cm* pour un nuage de points comme *LARA3Dv2010_Orsay*, nous avons un bruit d'écart-type 4 *mm* ce qui représente 8% de la distance moyenne entre points voisins. Pour le nuage de points *LARA3Dv2007_Soufflot*, ce pourcentage de bruit

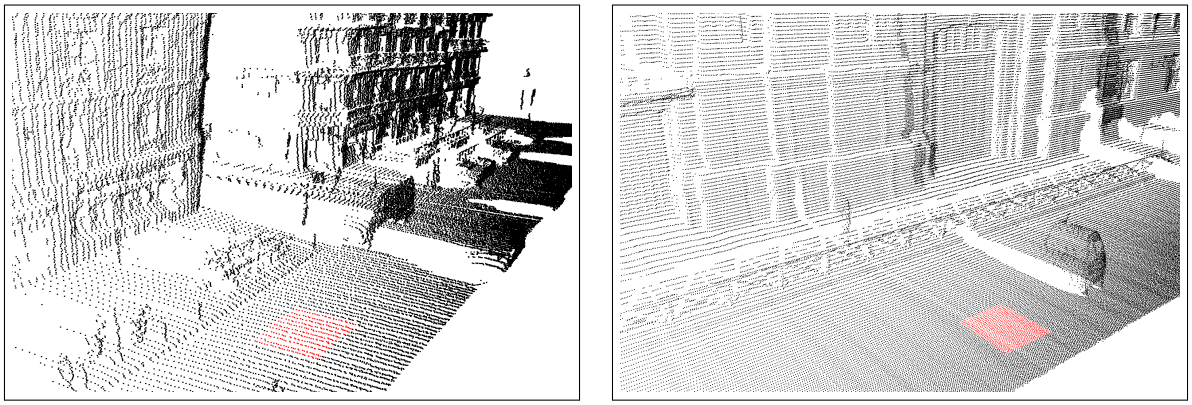


FIGURE 2.10 – A gauche, sélection en rouge d'une partie locale du nuage *LARA3Dv2007_Soufflot*. A droite, même sélection sur le nuage *LARA3Dv2010_Orsay*.

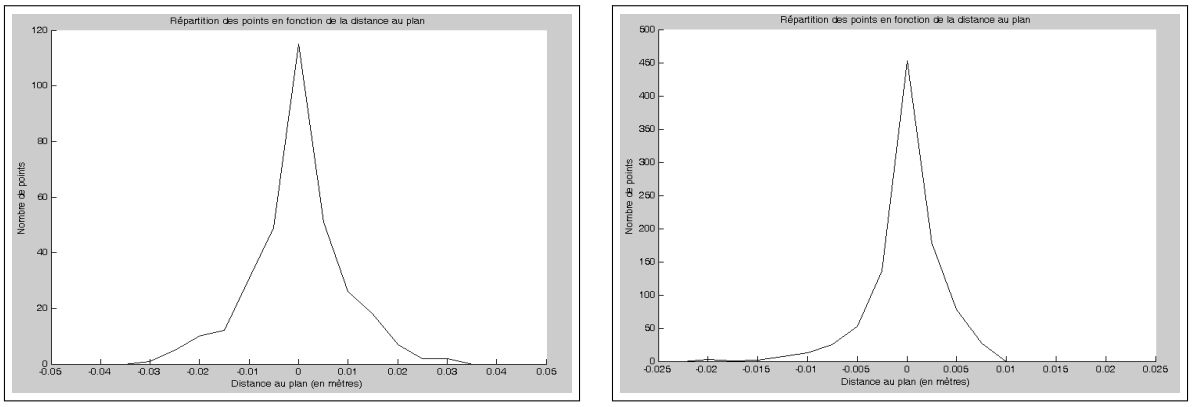


FIGURE 2.11 – A gauche, histogramme des distances des points au plan modélisé pour les parties locales de *LARA3Dv2007_Soufflot*. A droite, histogramme pour la partie locale du nuage *LARA3Dv2010_Orsay*.

monte à 14% avec une distance moyenne entre points de 7.8 *cm* et un bruit d'écart-type 11 *mm*. Nous verrons par la suite que ce pourcentage de bruit par rapport à la distance entre points influe particulièrement le choix des méthodes de traitement de nuages de points.

En revanche, nous pouvons voir qu'il n'existe pas de points fortement éloignés de la surface, appelés points aberrants ou « *outliers* » . En effet, la technologie laser étant une technologie suffisamment fiable, les erreurs sur les points mesurés sont (en supprimant l'erreur absolue et l'erreur relative) au maximum de l'ordre de quelques sigmas (écart-type du bruit de mesure du laser). Ceci nous affranchis du traitement des points aberrants, problème fréquemment rencontré avec les technologies d'acquisition basées images.

2.1.3 Orientation de la surface

Une autre donnée importante pour la reconstruction de surface est l'orientation du nuage de points. En effet, il est nécessaire de connaître en chaque point l'intérieur et l'extérieur, c'est-à-dire l'orientation de la surface représentée par les points. Pour connaitre cette orientation pour un nuage de points quelconque, il est possible comme dans [Hoppe et al., 1992] de calculer en chaque point la normale à la surface (nous verrons en détail dans le prochain chapitre la méthode de calcul) puis de trouver une orientation globale de la surface en construisant un graphe reliant les points voisins de manière à rendre l'orientation consistante. Cependant, l'intérieur et l'extérieur ainsi définis ne correspondent pas forcément aux attentes. Ensuite, il est très difficile d'avoir des orientations correctes pour des nuages de points comportant plusieurs composantes non connexes (c'est-à-dire d'ensembles de petits nuages de points qui sont à des distances plus grandes que la distance de voisinage). Nous voyons sur la figure 2.12 le calcul des normales (en bleu les points et en rose les normales) en construisant un graphe pour orienter les normales : nous voyons que les façades de gauche et de droite n'ont pas les mêmes orientations, c'est-à-dire que les points ne sont pas tous orientés vers la trajectoire (certains points n'ont pas de normales car ils ne possèdent pas assez de points dans leur voisinage).

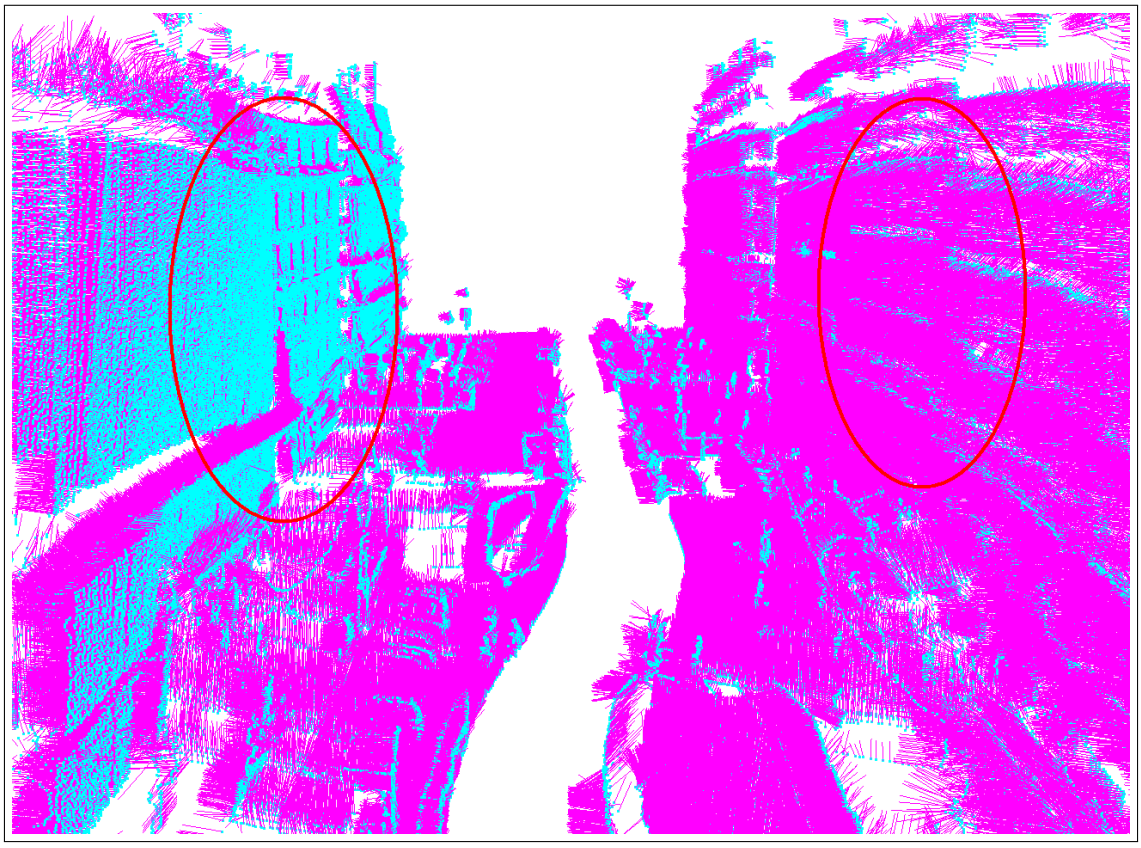


FIGURE 2.12 – Orientation des normales en construisant un graphe reliant les points du nuage *LARA3Dv2007_Soufflot* (en bleu les points et en rose les normales).

Dans le cas de points acquis par LiDAR, nous pouvons arbitrairement définir l'extérieur d'un point comme étant la direction du point de la surface vers sa source laser. Nous conservons ainsi une information supplémentaire lors de l'acquisition du nuage de points : il s'agit de la direction de l'onde laser pour chaque point capté. En effet, lorsque l'onde laser arrive sur la surface, le vecteur directeur de cette onde fait toujours un angle inférieur à 90 ° avec la normale locale à la surface. L'onde retour qui a la même direction que l'onde d'arrivée fait elle aussi un angle avec la normale locale inférieur à 90 °. Après avoir calculé la normale en chaque point du nuage, il suffit de choisir parmi les deux normales possibles celle qui fait un angle inférieur à 90 ° avec la direction du laser pour orienter le point vers l'extérieur. Cette méthode est applicable à toutes les technologies basées sur le LiDAR, y compris les systèmes mobiles puisque que l'on connait à chaque instant la position de la tête du capteur grâce à la localisation du véhicule. Le fait d'utiliser la direction du laser permet aussi d'orienter correctement la surface vers l'extérieur et non vers l'intérieur, c'est-à-dire la surface orientée vers le point de mesure. La figure 2.13 montre l'orientation des normales en utilisant la direction de l'onde laser pour chaque point : nous remarquons que les façades gauche et droite ont cette fois-ci la même orientation vers l'extérieur, c'est-à-dire vers la trajectoire du véhicule.

FIGURE 2.13 – Calcul des normales en utilisant la direction des lasers pour le nuage *LARA3Dv2007_Soufflot* (en bleu les points et en rose les normales).

En conservant la direction du laser, il est aussi possible d'estimer l'angle d'incidence du laser, c'est-à-dire l'angle entre la surface réelle et la direction du laser lors de la mesure. [Soudarissanane et al., 2009] a montré que cet angle a en moyenne une participation de 20% sur le bruit de mesure d'un point pour un scanner laser, et que pour un angle

d'incidence de plus de 60° il devenait le principal facteur d'imprécision d'un point par rapport à l'étalonnage du laser, aux conditions atmosphériques et aux propriétés de l'objet mesuré. Cette information d'angle peut être utilisée pour en déduire un facteur de qualité des points du nuage. Nous pouvons par exemple supprimer les points dont l'angle d'incidence est trop élevé. Nous avons représenté sur les figures 2.14 et 2.15 cet angle : la couleur d'un point varie du vert au rouge proportionnellement à l'angle d'incidence (en vert les points dont la direction du laser est proche de la normale calculée et en rouge les points qui ont été acquis avec une direction de l'onde laser proche de 90° à la normale à la surface). On observe sur ces figures que ce sont surtout les points des façades des rues perpendiculaires à la trajectoire du véhicule qui sont en rouge, c'est-à-dire les points pour lesquels le grand angle d'incidence implique une plus grande imprécision.

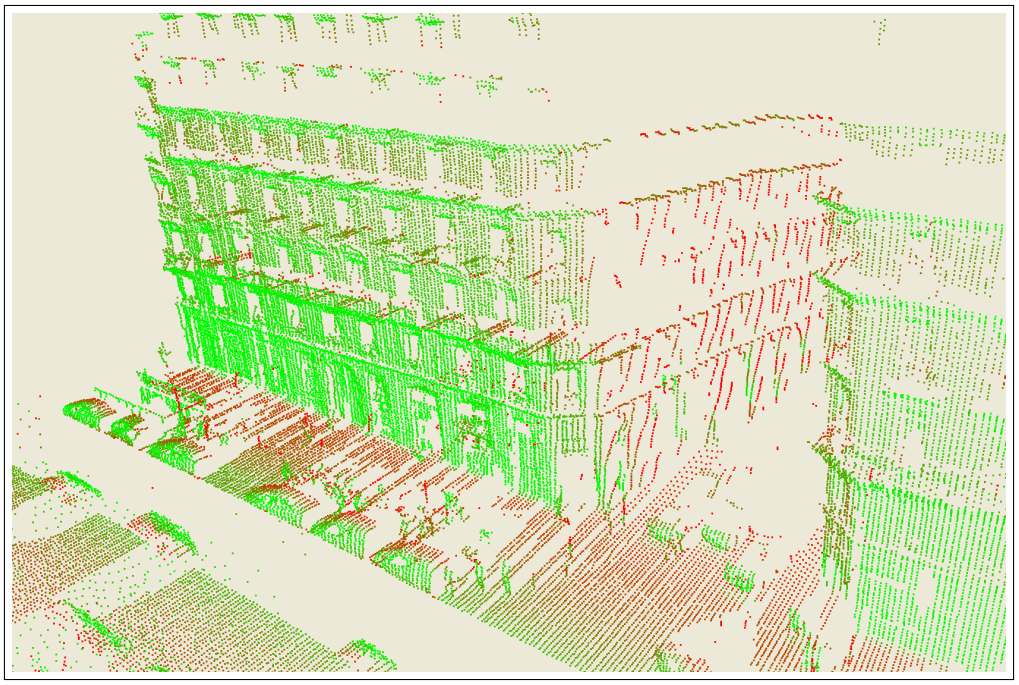


FIGURE 2.14 – Représentation de l'angle entre la direction de l'onde laser enregistrée pendant la capture et la normale calculée en chaque point (couleur proportionnelle à l'angle avec en vert un angle proche de 0° et en rouge proche de 90°) pour le nuage *LARA3Dv2007_Soufflot*.

FIGURE 2.15 – Représentation de l'angle entre la direction de l'onde laser enregistrée pendant la capture et la normale calculée en chaque point (couleur proportionnelle à l'angle avec en vert un angle proche de 0 ° et en rouge proche de 90 °) pour le nuage *LARA3Dv2010_Orsay*.

2.2 Nuages de points d'autres plateformes d'acquisitions

Nous détaillons ici quelques caractéristiques de nuages provenant de surfaces réelles acquises par d'autres systèmes de numérisation que LARA3D. Nous disposons de données provenant de plateformes mobiles d'acquisition comme la plateforme Stereopolis et le robot MINES_Rover ainsi que de données provenant de scanners lasers fixes tels que les Trimble VX, FX, GX et le scanner fixe Leica Cyrax.

2.2.1 Stereopolis et MINES_Rover

Stereopolis

Dans le cadre du projet TerraNumerica, l'IGN a conduit plusieurs acquisitions dans Paris. Nous avons retenu deux tracés pour nos tests : le nuage de points *Stereopolis_Soufflot* de la rue Soufflot et le nuage de points *Stereopolis_Orsay* de la façade Nord du musée d'Orsay. Ces deux nuages de points sont représentés à la figure 2.16.

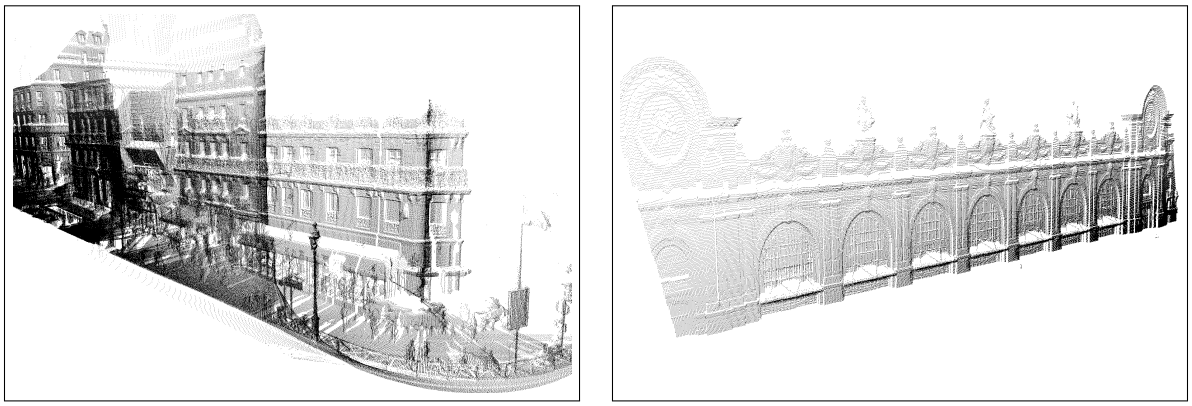


FIGURE 2.16 – A gauche nuage de points *Stereopolis_Soufflot*. A droite nuage de points *Stereopolis_Orsay* tous deux produits par le système Stereopolis.

Nous avons fait une analyse des nuages de points de Stereopolis de la même manière que pour les nuages de la plateforme LARA3D. Comme pour les données LARA3D, les données Stereopolis sont hétérogènes et anisotropes. La distance moyenne entre un point et son plus proche voisin est de 4.6 *cm* avec un écart-type de 2.8 *cm* pour le nuage *Stereopolis_Soufflot* et une distance moyenne de 11.8 *cm* et d'un écart-type de 2.2 *cm* pour le nuage *Stereopolis_Orsay*. Nous avons tracé pour ces deux nuages la répartition des points en fonction de la distance d'un point à son voisin le plus proche à la figure 2.17. Il faut noter que la configuration des lasers étant différente entre l'acquisition de la rue Soufflot et l'acquisition du musée d'Orsay, cela explique la différence des histogrammes de distance. On constate en effet que la dispersion des points est très grande pour le nuage *Stereopolis_Soufflot*.

Nous avons aussi mesuré la précision relative des nuages de points en comparant les

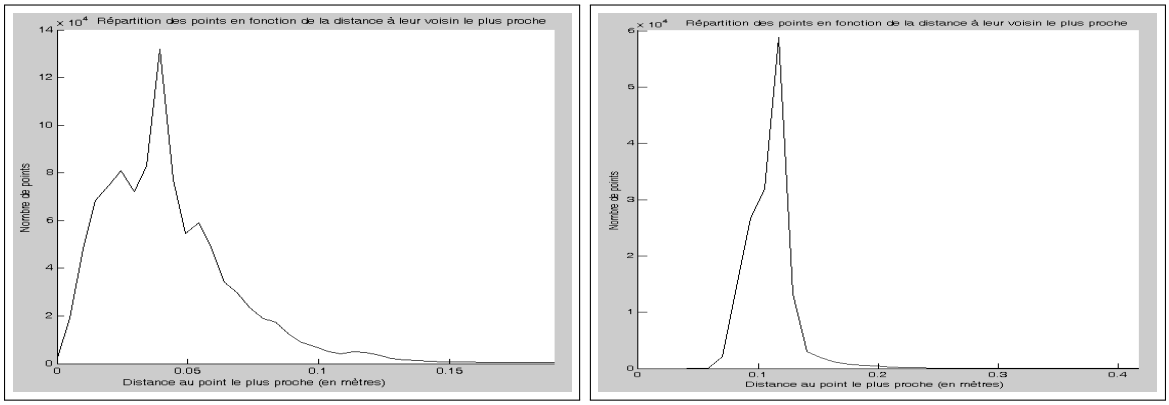


FIGURE 2.17 – Histogramme de la distance entre un point et son voisin le plus proche pour le nuage de points *Stereopolis_Soufflot* à gauche et pour le nuage *Stereopolis_Orsay* à droite.

données du musée d'Orsay avec le nuage *FX_GX_Orsay* produit par les scanners Trimble FX et GX. Pour cela, nous avons effectué un recalage des zones de chevauchement de ces deux nuages de points afin d'enlever l'erreur absolue sur le nuage et avons minimisé la distance entre ces deux nuages par ICP. Nous avons trouvé une erreur moyenne de 5.6 *cm* calculée sur 226 865 points en commun. La figure 2.18 montre la zone de chevauchement des deux nuages de points.

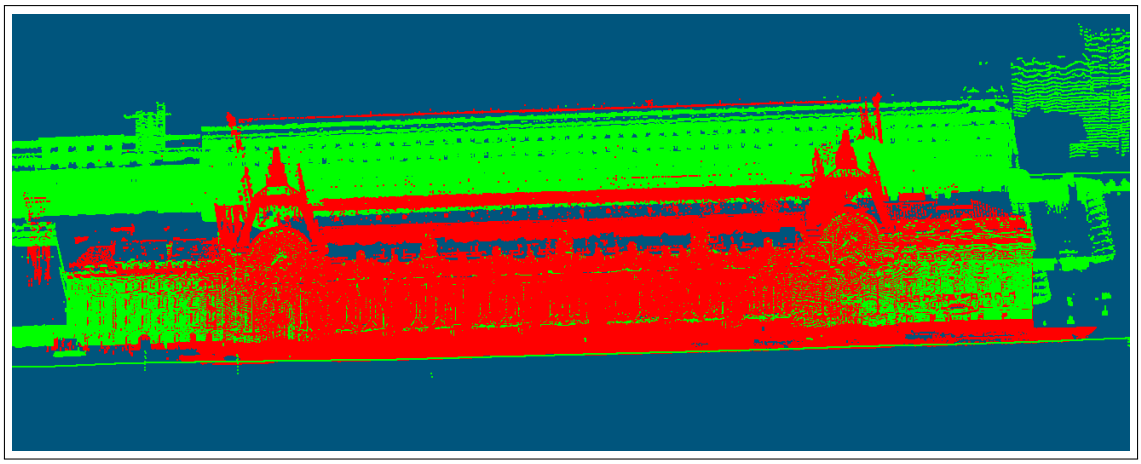


FIGURE 2.18 – Recalage du nuage *Stereopolis_Orsay* en vert avec le nuage *FX_GX_Orsay* en rouge.

La figure 2.19 représente la répartition des points autour d'une zone considérée comme plane afin de calculer l'erreur locale de mesure. Une fois de plus l'erreur locale prend la forme d'un bruit gaussien avec un écart-type de 6.6 *mm* pour *Stereopolis_Orsay*. En revanche, pour le nuage *Stereopolis_Soufflot*, l'écart-type des points autour du plan est bien plus important avec une valeur de 1.6 *cm* et le bruit ne semble pas être gaussien. Ceci est dû à la configuration différente des lasers pour cette acquisition. En effet, ceux-ci

ont été placés pour capturer le même côté de sorte qu'ils scannent la plupart des zones de l'environnement avec deux angles différents. Il y a donc une redondance d'information qui permet de limiter les zones d'occlusion mais augmente le niveau de bruit local. Dans ce cas, le niveau de bruit représente 35% de la distance moyenne entre points alors que celui-ci se limite à 6% pour le nuage *Stereopolis_Orsay*.

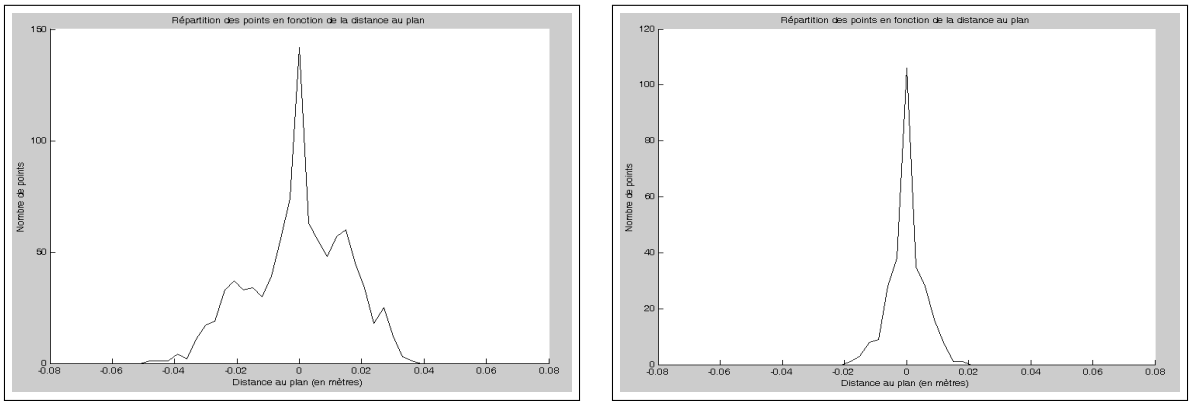


FIGURE 2.19 – A gauche, histogramme des distances des points à un plan local pour le nuage *Stereopolis_Soufflot*. A droite, même histogramme pour le nuage *Stereopolis_Orsay*.

En étudiant les systèmes LARA3D et Stereopolis, nous voyons qu'ils produisent des données très similaires même si la précision finale absolue et relative des données de Stereopolis est supérieure grâce à un système de positionnement très précis. De manière générale, les systèmes de cartographie produisent des données précises localement avec un bruit compris environ entre 5% et 50% de la distance moyenne entre points mais qui peuvent avoir des dérives dépendant du système de navigation. Dans le cas où plusieurs scanners sont utilisés pour couvrir une même zone (à différents moments de l'acquisition), le nuage de points sera plus complet avec moins de zones cachées par des objets placés entre le scanner et la surface à numériser (comme un camion devant une façade) mais sera beaucoup plus bruité localement impliquant une modélisation de la surface moins précise et, dans certains cas, l'utilisation d'algorithmes de modélisation différents.

MINES_Rover

La figure 2.20 montre le nuage de points *MINES_Rover_Indoor* de l'intérieur du laboratoire CAOR de MINES ParisTech produit par le robot *MINES_Rover*.

Avec le mode d'acquisition « *Stop&Go* », nous obtenons un nuage très hétérogène avec des densités très importantes près des positions où le MINES_Rover s'est arrêté et des densités très faibles dans les zones de transition. Nous pouvons voir cette répartition hétérogène des points sur la figure 2.20. C'est pourquoi nous obtenons une distance moyenne entre points très faible de 5.6 mm avec un écart-type très grand de 5.5 mm. Nous remarquons aussi cela sur la répartition des distances d'un point à son plus proche voisin

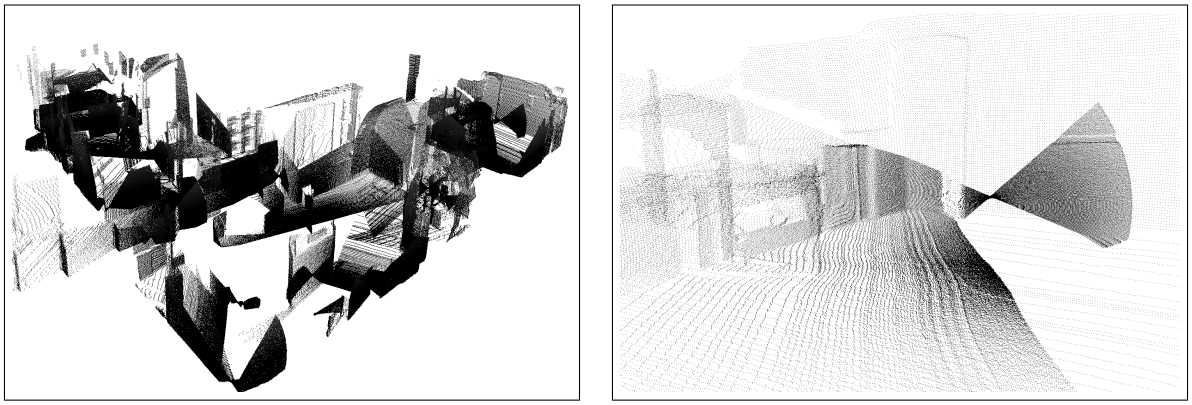


FIGURE 2.20 – A gauche, nuage de points *MINES_Rover_Indoor*. A droite, détail du nuage.

de la figure 2.21 avec une grande dispersion des distances entre points.

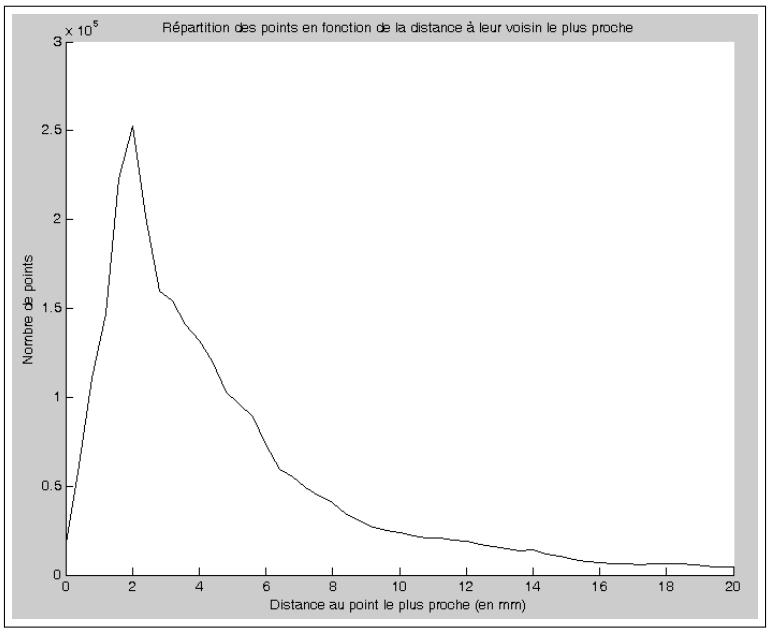


FIGURE 2.21 – Répartition des distances entre points dans le nuage *MINES_Rover_Indoor*.

2.2.2 Systèmes de numérisation laser fixes

Nous avons testé dans le cadre de cette thèse des nuages de points provenant de scanners fixes. Nous disposons notamment d'un nuage de points *VX_Soufflot* de la rue Soufflot acquis avec une station totale Trimble VX (technologie à temps de vol avec 15 pts/s, écart-type de 10 mm à 150 m) provenant du projet TerraNumerica, d'un nuage de points *FX_GX_Orsay* de la façade nord du musée d'Orsay par la consolidation de plusieurs scans de Trimble FX (technologie à différence de phase avec 200 000 pts/s, écart-type de 2 mm à

50 m) et GX (technologie à temps de vol avec 5 000 pts/s, écart-type de 12 mm à 100 m) provenant aussi du projet TerraNumerica et enfin d'un nuage de points *Cyrax_Chateau* de la façade d'un château par un scanner laser Leica Cyrax 2500 (technologie à temps de vol avec $4k$ pts/s, écart-type de 6 mm à 50 m) provenant du répertoire « *AIM@SHAPE* » de l'INRIA. Ces appareils de mesure ont des écart-types très similaires de l'ordre de 5-10 mm à 100 m sauf le Trimble FX basé sur une technologie différente. Mais ils ont des vitesses de numérisation très différentes entre 15 pts/s et 200 000 pts/s et des densités allant de 15 pts/m^2 à 12 000 pts/m^2.

La figure 2.22 montre les nuages de points pour les scanners Trimble VX, Trimble FX et Leica Cyrax. Nous remarquons visuellement la grande différence de densité de ces nuages de points. La densité moyenne d'un nuage de points n'est pas une valeur très indicative pour les systèmes mobiles car celle-ci varie fortement suivant la position d'un point par rapport à la trajectoire du mobile : un objet proche (par exemple la route) peut avoir de nombreux points alors que les points représentant l'objet à modéliser (par exemple une façade) peuvent ne pas être suffisamment denses.

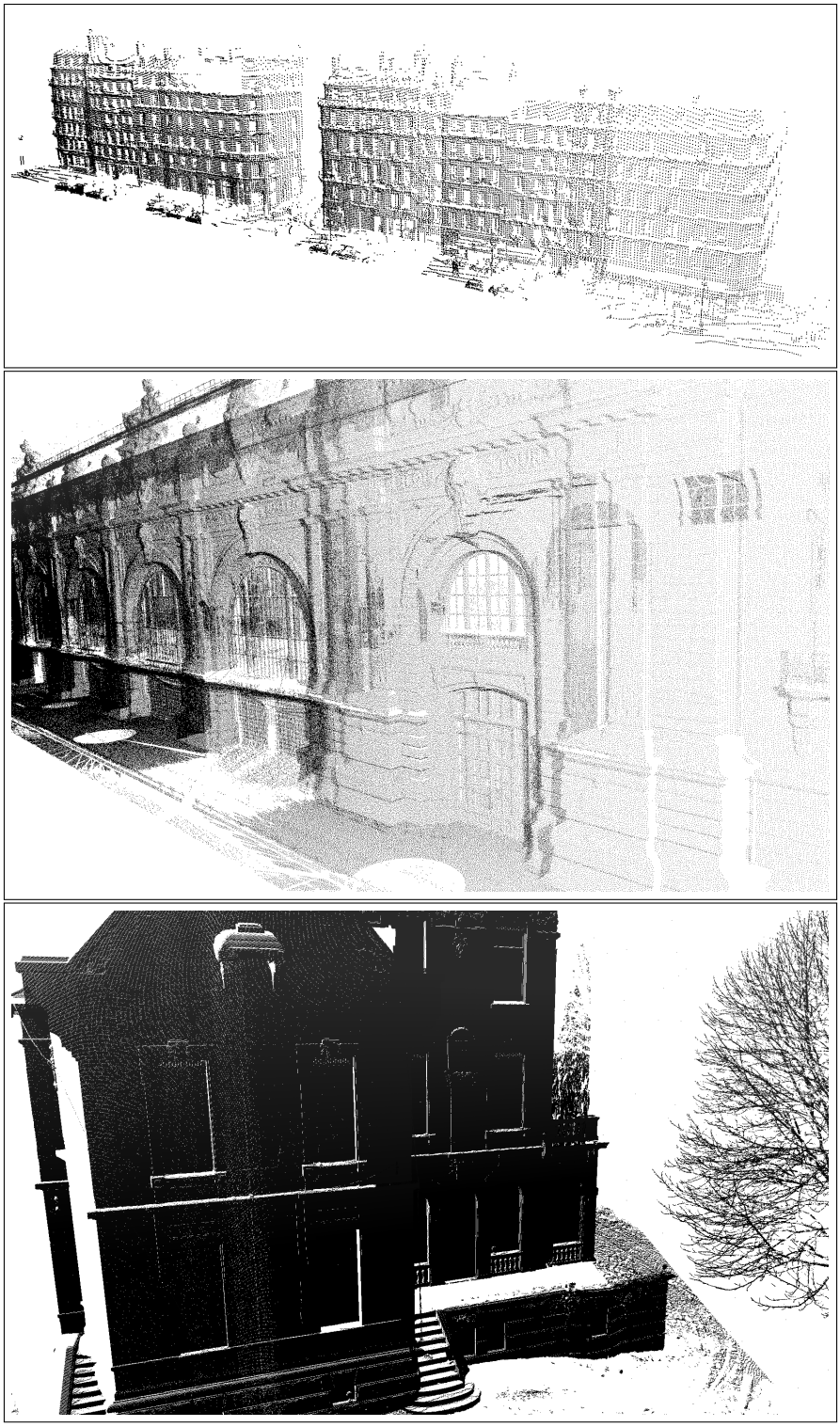


FIGURE 2.22 – En haut, nuage de points *VX_Soufflot* d'un Trimble VX. Au milieu, nuage de points *FX_GX_Orsay* des Trimble FX et GX. En bas, nuage de points *Cyrax_Chateau* d'un Leica Cyrax.

2.3 Récapitulatif

L'ensemble des caractéristiques des nuages de points produits par les différentes technologies étudiées dans ce chapitre sont récapitulés dans le tableau 2.1. Nous remarquons que les nuages *LARA3Dv2010_Orsay* et *FX_GX_Orsay* qui représentent la même surface réelle (la façade Nord du musée d'Orsay) sont très similaires en terme de résolution (d_{mean}) et de densité même si la précision des points est bien meilleure pour le nuage *FX_GX_Orsay*.

	Nb points	d_{mean}	$\sigma_{d_{mean}}$	Densité
LARA3Dv2007_Soufflot	385 639	7.8 cm	6.3 cm	162.9 pts/m^2
LARA3Dv2010_Etables	3 540 911	13.2 cm	9.6 cm	64.1 pts/m^2
LARA3Dv2010_Orsay	2 440 416	4.8 cm	2.6 cm	419.7 pts/m^2
Stereopolis_Soufflot	1 007 747	4.6 cm	2.8 cm	459.7 pts/m^2
Stereopolis_Orsay	156 191	11.8 cm	2.2 cm	71.0 pts/m^2
MINES_Rover_Indoor	2 836 369	5.6 mm	5.5 mm	31 887.7 pts/m^2
VX_Soufflot	42 477	24.4 cm	14.1 cm	16.7 pts/m^2
FX_GX_Orsay	1 804 648	5.9 cm	2.0 cm	287.0 pts/m^2
Cyrax_Chateau	5 535 038	8.9 mm	6.0 mm	12 489.6 pts/m^2

TABLE 2.1 – Tableau résumant les caractéristiques des nuages de points étudiés.

Chapitre 3

Traitements de nuages de points denses

Résumé

Ce chapitre expose les traitements nécessaires avant l'étape de modélisation. Ce sont des traitements applicables pour des nuages de points denses quelconques. Nous étudions tout d'abord une méthode de décimation de nuage de points afin de réduire le nombre de points à traiter. Sachant que les normales sont nécessaires pour notre approche de segmentation et de triangulation, nous exposons une nouvelle méthode de calcul de normales en chaque point. Nous proposons aussi dans ce chapitre un algorithme de débruitage applicable directement sur les nuages de points.

3.1 Quelques notions sur les nuages de points

Nous faisons l'hypothèse qu'un nuage de points est un échantillonnage d'un ensemble de surfaces ou 2-variétés de classe C^2. L'intersection de deux surfaces est une courbe de $\mathbb{R}^3$ appelée arête et l'intersection de trois surfaces ou plus est un point appelé sommet. A titre d'exemple, le nuage de points *S_Cube* de la figure 3.1 est composé de 6 surfaces de classe C^2 (ici 6 plans), de 8 arêtes (ici des droites) et de 8 sommets. Les points appartenant à des arêtes sont aussi appelés points de discontinuités par abus de langage. La normale en un point p_i est égale à la normale à la surface en p_i. Pour un point de discontinuité, la normale peut être définie comme la moyenne des normales des différentes surfaces auxquelles le point appartient. La figure 3.2 montre les normales de trois points p_i (en rouge). A gauche, tous les points du voisinage (défini plus loin) du point p_i en rouge appartiennent à la même surface et donc la normale de p_i est la normale à la surface. Au milieu, certains points du voisinage du point p_i en rouge n'appartiennent pas à la même surface que p_i mais la normale ne va dépendre que de la surface à laquelle p_i appartient. Enfin à droite, le point p_i est un point de discontinuité et sa normale est égale à la moyenne des normales des deux surfaces qui s'intersectent en p_i.

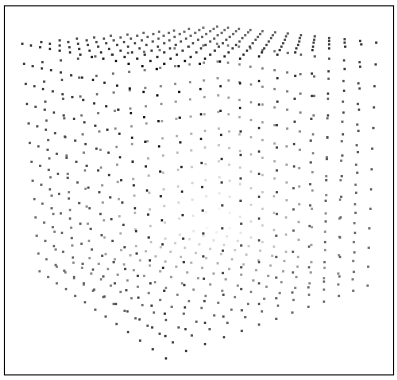


FIGURE 3.1 – Nuage de points *S_Cube* d'un cube.

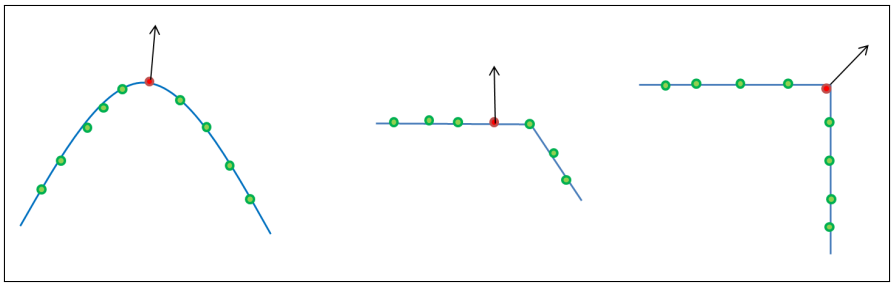


FIGURE 3.2 – Normales de trois points suivant leur voisinage (point p_i en rouge, points du voisinage en vert, normale en noir).

A partir d'un nuage de points de référence, nous considérons les nuages de points bruités. Nous rajoutons pour cela en chaque point p_i un bruit gaussien d'écart-type σ suivant x, y

et z pour générer le point p'_i qui hérite des propriétés de p_i. Nous considérons par exemple que p'_i appartient à la même surface que p_i et si p_i est un point de discontinuité alors nous considérons aussi que p'_i est un point de discontinuité. Par cette méthode, nous simulons ainsi des nuages de points bruités permettant de tester et de valider nos méthodes de traitements de nuages de points en comparant les résultats avec ceux du nuage de points d'origine.

3.1.1 Voisinage d'un point

Tous les traitements sur les nuages de points que nous voyons dans la suite sont appliqués localement. En effet, que cela soit pour calculer la densité locale ou la normale en un point, il est nécessaire de savoir comment sélectionner les points voisins. Nous devons donc définir la notion de voisinage local. Deux définitions sont majoritairement utilisées dans la littérature, comme détaillé dans [Rabbani et al., 2006]. La première consiste à utiliser les k voisins les plus proches comme voisinage avec k généralement compris entre 15 et 20. L'avantage de cette définition est d'être indépendante de la distance entre les points. A l'inverse, celle-ci présente l'inconvénient majeur de dépendre de la densité. Ainsi, en un point de densité faible, prendre les k plus proches voisins n'a pas de signification physique car le $k$ième voisin peut être très éloigné de p_i et ne devrait pas être utilisé comme point de voisinage local. L'autre définition pour le voisinage local est de considérer comme voisins tous les points inclus dans une sphère de rayon δ (que nous appelons distance locale). Une telle définition donne au voisinage un sens géométrique et permet donc d'être indépendant de la répartition locale des points. En revanche, le voisinage va inclure trop de points dans les régions denses et pas assez de points dans les zones peu denses. Entre ces deux méthodes, nous avons choisi de définir le voisinage local comme l'ensemble des points inclus dans la sphère de rayon δ, car c'est la définition qui s'adapte le mieux aux nuages de points denses.

La densité locale est définie à partir du voisinage local et est égale au nombre de voisins inclus dans la sphère de rayon δ divisé par l'aire de la surface intersectée par la sphère, ce qui donne une densité $\rho = \frac{k}{\pi\delta^2}$.

En prenant un ensemble de points répartis de manière uniforme, isotrope et homogène sur un plan, nous pouvons établir des liens entre les notions de densité ρ, de distance entre points voisins d et de distance locale δ. En effet, pour un carré de côté l avec N points répartis uniformément comme sur la figure 3.3, l'aire du carré est égale à $A = l^2 \approx Nd^2$. La densité est alors égale à $\rho = \frac{N}{l^2} \approx \frac{N}{Nd^2} = \frac{1}{d^2}$. Nous avons donc un lien entre la densité ρ et la distance moyenne entre points d. Pour un disque qui contient k points, son aire est égale à $A = \pi\delta^2 = \frac{k}{\rho} \approx kd^2$. Nous avons ainsi trouvé dans un cas simple un lien entre un voisinage à k points fixes et un voisinage avec une distance locale δ, où $\delta = \sqrt{\frac{k}{\pi}}d$.

La figure 3.4 montre les densités locales du nuage de points *LARA3Dv2010_Etables* avec en rouge les zones de forte densité et en vert les zones peu denses. Nous voyons que la route est une zone très dense. Nous avons calculé la densité moyenne du nuage de points $\rho_{mean} = 64\ pts/m^2$ et l'écart type de la densité de 46 pts/m^2. Cet écart-type très important montre encore une fois l'hétérogénéité de la répartition des points du nuage. Avec la relation précédente, nous pouvons déduire une distance moyenne entre points à partir de la densité

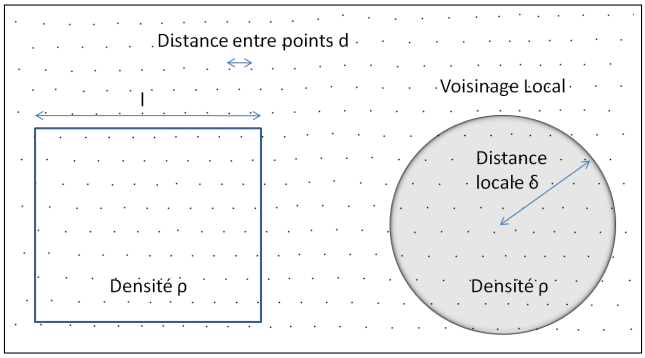


FIGURE 3.3 – Nuage de points répartis uniformément sur un plan.

locale moyenne : $d_{mean} \approx \sqrt{\frac{1}{\rho_{mean}}} = 12.5\ cm$ ce qui correspond bien à la valeur trouvée pour la distance moyenne entre points voisins de 13.2 cm.

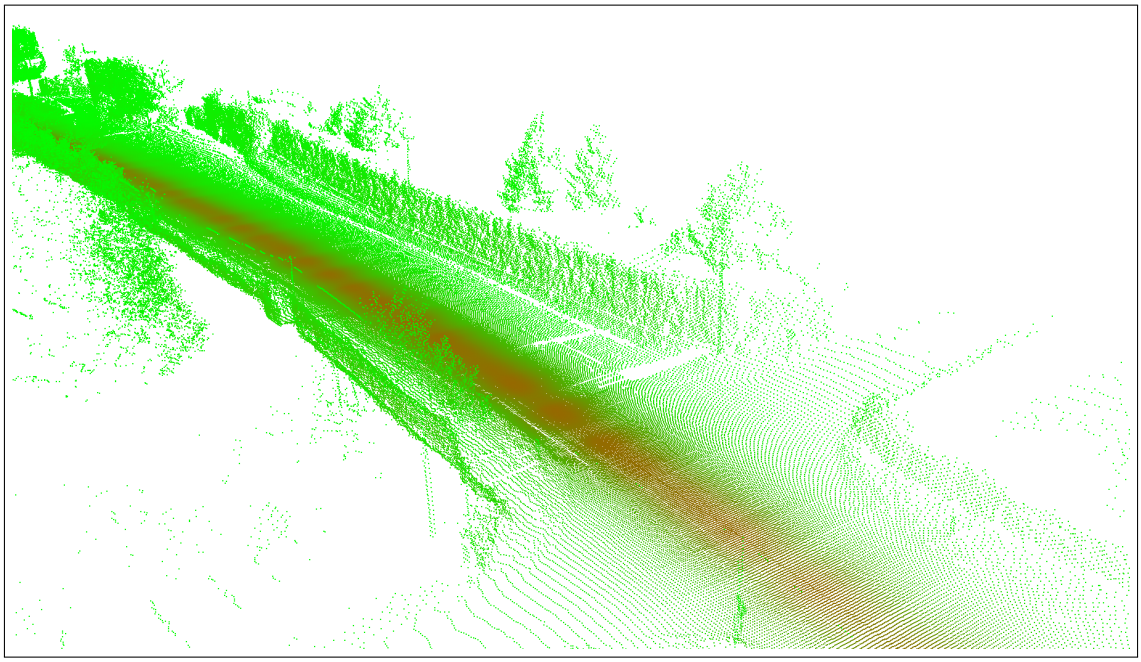


FIGURE 3.4 – Densité locale du nuage de points *LARA3Dv2010_Etables* avec en rouge les zones de forte densité et en vert les zones peu denses.

3.2 Décimation de nuages de points

Nous avons vu au chapitre précédent plusieurs caractéristiques des nuages de points produits par système mobile. Tout d'abord, ce sont de gros volumes de données avec plusieurs millions de points pour quelques centaines de mètres. Sachant que les temps de traitements des nuages de points dépendent directement du nombre de points, nous souhaitons en diminuer le nombre dès que possible, c'est-à-dire dès le début de la phase de modélisation. Du fait de la technologie d'acquisition, de nombreux points sont à une distance parfois négligeable devant le niveau de détail souhaité. Par exemple, pour une reconstruction de l'environnement à une échelle de 10 *cm*, c'est-à-dire avec des points dont la distance entre voisins est en moyenne de 10 *cm*, nous n'avons pas besoin de conserver deux points voisins à moins d'1 *cm*. Ceci vient du fait que nous souhaitons reconstruire une surface qui interpole les points mesurés et donc nous voulons créer un maillage dont les sommets des triangles sont des points du nuage. Ceci explique la nécessité de notre première étape de traitement consistant à diminuer le nombre de points du nuage. Cela sert à accélérer les temps de traitement des étapes ultérieures, et ainsi diminuer le nombre de triangles du modèle final.

3.2.1 Etat de l'art

Il existe de nombreuses méthodes de simplification pour décimer les nuages de points. Un point du nuage peut être supprimé sur des critères géométriques comme la distance du point à la surface, la courbure locale, la distance à l'axe médian de la forme. [Alexa et al., 2003] diminuent le nombre de points d'un nuage en enlevant ceux qui apportent le moins d'information sur la surface, c'est-à-dire en estimant la différence locale entre la surface avec et sans le point. [Song et Feng, 2008] ont ajouté dans le processus de réduction de points l'information du nombre final de points retenus, qui consiste à faire des regroupements de points sur des critères géométriques. Mais leur méthode est très lente et difficile à paralléliser. [Sheung et Wang, 2009] et [Cohen-Steiner et al., 2004] minimisent une fonction d'énergie avec un terme de distance et un terme de forme.

La plupart de ces méthodes ont besoin en chaque point de la courbure, de la normale ou d'un opérateur de projection sur la surface pour connaître les points à conserver. Or ces grandeurs nécessitent le calcul en chaque point d'un plan tangent par moindres carrés. Cette étape est très coûteuse en temps de calcul et n'est alors pas compatible avec le traitement de très gros volumes de données. De plus, avec les données de la plateforme mobile LARA3D, nous avons des nuages de points fortement anisotropes comme on le constate sur la figure 3.5. Ainsi, la disposition locale des points (en lignes) va donner des plans par moindres carrés qui ne seront plus tangents à la surface. Nous avons donc besoin d'une quantité mesurant approximativement la variation géométrique locale des points qui ne nécessite pas le calcul d'un plan par moindre carré et qui se calcule très rapidement.

3.2.2 Méthode de décimation proposée

Pour réduire la taille de nos nuages de points, nous effectuons une décimation par distance minimale entre points voisins, c'est-à-dire que si deux points sont plus proches

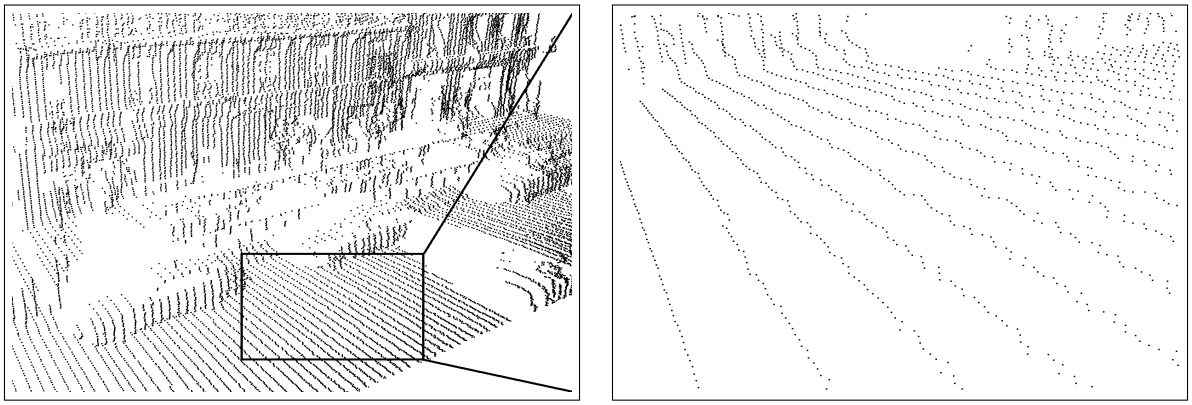


FIGURE 3.5 – A gauche, nuage de points *LARA3Dv2007_Soufflot*. A droite, un gros plan sur une partie fortement anisotrope.

qu'une certaine distance d_{deci}, nous supprimons un des deux points. Nous construisons ainsi un sous-échantillonnage du nuage de points dans lequel la distance entre un point et son plus proche voisin est égale ou supérieure à d_{deci}.

Cette distance de décimation est un paramètre qui dépend pour les nuages de points provenant de systèmes mobiles, de trois résolutions :

- de la résolution des points suivant un profil, c'est-à-dire de la résolution angulaire des lasers : d_{profil} ;
- de la résolution des points suivant la trajectoire dépendant elle-même de la vitesse du véhicule et de la fréquence du scanner : $d_{trajectoire}$;
- du niveau de détail souhaité : $d_{details}$.

Nous avons étudié les deux premières résolutions lors du chapitre précédent avec $d_{profil} = \frac{d_{min}+d_{max}}{2} r_{laser}$ la distance entre points suivant les profils lasers, et $d_{trajectoire} = \frac{v_{vehicule}}{f_{laser}}$ la distance entre points suivant la direction de la trajectoire où r_{laser} désigne la résolution angulaire du laser (en radians), $v_{vehicule}$ la vitesse du véhicule (en m/s) et f_{laser} la fréquence de rotation du laser (en Hz ou $tours/s$). Or notre objectif est de diminuer fortement l'aspect anisotrope des répartitions locales des points. C'est pourquoi nous prenons une distance de décimation $d_{deci} = \frac{d_{profil}+d_{trajectoire}}{2}$; l'idée étant de couper le premier pic d'anisotropie que nous avons vu sur les histogrammes de distances au chapitre 2. Cela permet aussi d'enlever les points trop proches les uns des autres et évite ainsi la création de triangles trop petits ou trop allongés lors du maillage.

$d_{details}$ correspond au niveau de détails minimum souhaité pour le modèle final. Pour conserver le maximum de détails par rapport aux informations capturées, nous prenons la valeur $d_{details} = 0$. Augmenter la valeur de $d_{details}$ permet de diminuer considérablement le nombre de points et donc d'accélérer les temps de traitements pour toutes les étapes ultérieures.

Pour récapituler, la distance de décimation finale peut s'écrire sous la forme :

$$d_{deci} = \max(\frac{d_{profil} + d_{trajectoire}}{2}, d_{details}) \tag{3.1}$$

Pour les nuages de points simulés ou provenant d'autres méthodes d'acquisition (telles que les stations laser fixes), on ne dispose pas de données d_{profil} et $d_{trajectoire}$. A la place, nous prenons la valeur $\frac{d_{mean}}{2}$ avec d_{mean} exprimant la valeur moyenne des distances d'un point à son voisin le plus proche. Cela donne un paramètre de décimation $d_{deci} = \max(\frac{d_{mean}}{2}, d_{details})$. Ainsi, la suppression des points dont la distance est inférieure à $\frac{d_{mean}}{2}$ permet de s'assurer du bon calcul des normales et d'avoir un maillage avec des triangles plus proches de triangles équilateraux.

En effectuant cette décimation, nous améliorons la répartition locale des points en la rendant plus uniforme. A l'inverse, nous risquons d'enlever des points importants pour la reconstruction de la surface : les points de forte courbure et notamment les points appartenant à des arêtes. Pour remédier à cela, nous calculons en chaque point une approximation de la courbure locale. [Gumhold et al., 2001] calculent une approximation de la courbure locale en chaque point p_i avec $\kappa_i = \frac{2d_i}{\mu_i^2}$ où μ_i est la distance moyenne des points p_j du voisinage à p_i et d_i est la distance du point p_i au plan des moindres carrés du voisinage de p_i. Or comme nous l'avons dit, nous ne pouvons pas calculer le plan par moindres carrés en certains points à cause de la répartition locale anisotrope. Nous allons donc faire une approximation de d_i. Puisque nous savons que le barycentre b_i des points du voisinage de p_i appartient au plan des moindres carrés, il est possible de faire l'approximation locale $d_i \approx \|b_i - p_i\|$. Ce qui nous permet de calculer très rapidement la courbure moyenne en chaque point p_i du nuage $\kappa_i \approx \frac{2\|b_i - p_i\|}{\mu_i^2}$ afin de conserver uniquement les points p_i du nuage qui ont une courbure maximale par rapport aux points de leurs voisinage. Nous avons testé notre approximation de courbure locale sur le nuage de points simulé *S_Fandisk*. La figure 3.6 illustre le résultat de ce test et nous remarquons que les sommets et les points sur les arêtes sont bien sélectionnés par notre courbure locale (points encerclés en vert).

Pour la décimation d'un nuage de points, nous enlevons tout d'abord, autour des points qui maximisent la courbure moyenne locale, tous les points dont la distance est inférieure à d_{deci}. Ensuite parmi les points qui restent, si la distance entre deux points est inférieure à d_{deci}, alors nous enlevons le point qui a la plus faible courbure moyenne approximée.

Le fait d'approximer la courbure locale permet de discriminer très rapidement les points et donc de diminuer le nuage de points pour de grand volumes de données avec un temps de traitement d'environ 300 000 points par seconde sur une plateforme QuadCore Q9300 2.50 GHz, 2 Go de RAM, sur un seul *thread* et sur CPU.

Pour montrer l'importance de conserver les points qui maximisent notre approximation de la courbure moyenne, nous avons testé notre méthode de décimation sur un nuage de points représentant un cube. Avec une distance moyenne entre points telle que $d_{mean} = 0.8$ et une distance de détails fixée à $d_{details} = 2.6$ afin de décimer fortement le nuage de points, nous obtenons : $d_{deci} = 2.6$. La figure 3.7 montre le résultat de la décimation sans tenir compte des points qui maximisent la courbure. Après la décimation, il reste 69 points sur

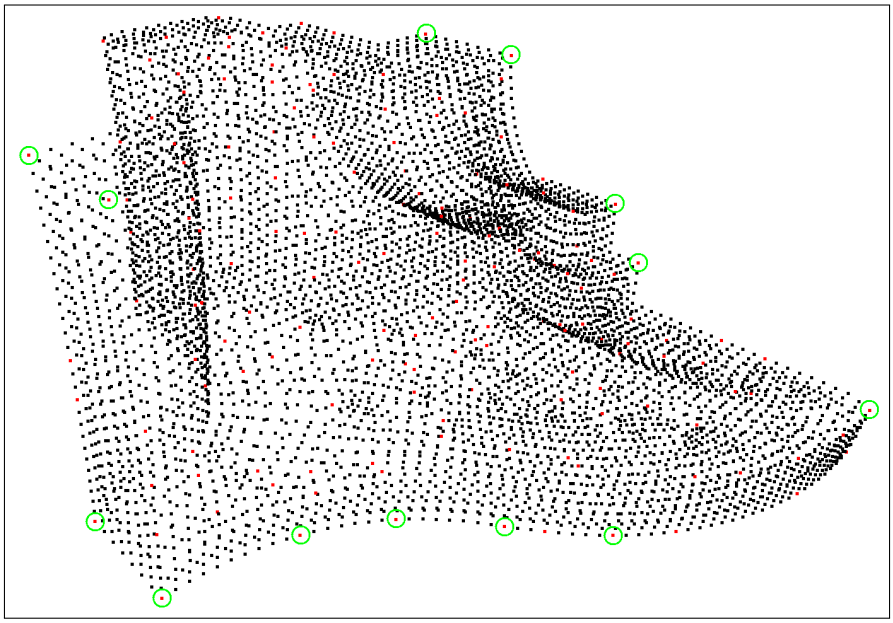


FIGURE 3.6 – Nuage de points *S_Fandisk* avec en rouge les points qui maximisent notre approximation de la courbure moyenne dans leur voisinage (quelques points en rouge présents sur des arêtes ont été encerclé en vert pour plus de visibilité).

les 866 points initiaux et certains sommets du cube n'ont pas été conservés. Nous avons donc à droite de la figure 3.7 un maillage (construit avec la méthode de triangulation du chapitre 4) qui ne représente pas la surface initiale. Sur la figure 3.8, nous montrons la décimation en effectuant tout d'abord une sélection des points qui maximisent notre approximation de la courbure moyenne. Les points en rouge sont les points qui maximisent la courbure moyenne dans leur voisinage et qui ont été conservés lors de la décimation. Les 8 sommets du cube ayant été sélectionnés par ce critère, notre méthode permet une reconstruction de la surface après décimation identique à la surface d'origine.

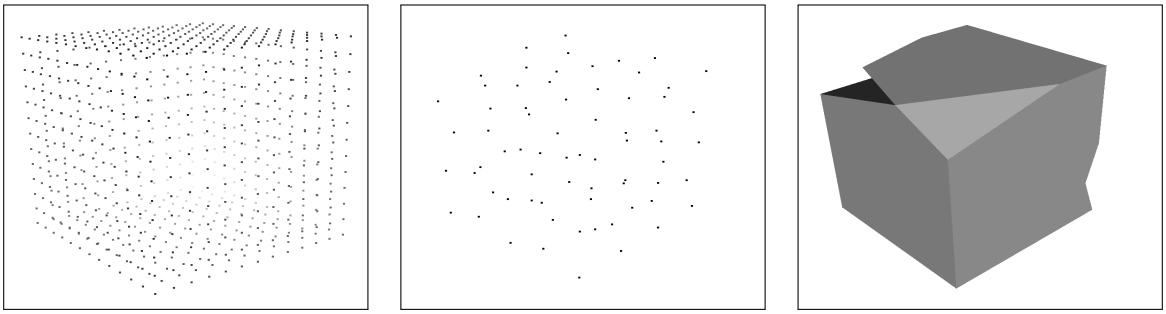


FIGURE 3.7 – A gauche, nuage de points brut *S_Cube*. Au milieu, nuage de points décimés avec $d_{deci} = 2.6$ sans sélection des points par courbure moyenne. A droite, la reconstruction de la surface par triangulation à partir du nuage décimé.

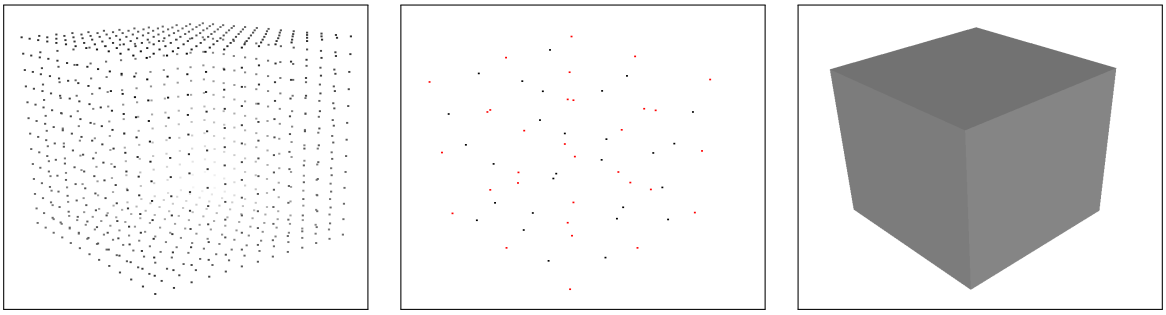


FIGURE 3.8 – A gauche, nuage de points brut *S_Cube*. Au milieu, nuage de points décimés avec $d_{deci} = 2.6$ et sélection des points en rouge qui maximisent localement la courbure moyenne. A droite, la reconstruction de la surface par triangulation à partir du nuage décimé.

3.2.3 Exemples de décimation de nuages de points

Pour le nuage *LARA3Dv2007_Soufflot*, nous savons, avec les informations de vitesse du véhicule et les informations techniques sur le laser utilisé que $d_{profil} = 6.7\ cm$ et $d_{trajectoire} = 13.9\ cm$. Pour garder le maximum de détails, nous avons choisi $d_{details} = 0$ ce qui nous donne une distance de décimation : $d_{deci} = 10.3\ cm$. Cette valeur nous permet d'éliminer de nombreux points de la route qui ne sont pas significatifs et de rendre la répartition des points beaucoup plus homogène au niveau local. Le résultat de cette décimation est visible sur la Figure 3.9. Au final, nous avons supprimé 113 056 points des 385 639 du nuage brut soit environ 30% des points. Nous pouvons étudier l'impact de la décimation sur le caractère anisotrope du nuage de points en calculant après décimation les histogrammes des distances aux points les plus proches. Nous voyons le résultat sur la figure 3.10 : il n'y a plus qu'un seul pic à droite sur l'histogramme de distance après décimation. Avant la décimation, la distance moyenne entre points était de 7.8 *cm* avec un écart-type important de 6.3 *cm*. Après la décimation, nous avons perdu un peu en résolution avec une distance moyenne de 11.1 *cm* mais avec un écart-type beaucoup plus faible par rapport à la nouvelle distance moyenne : 6.0 *cm*.

La figure 3.11 illustre la décimation du nuage brut *LARA3Dv2010_Orsay* avec une distance $d_{deci} = 7.05\ cm$ ($d_{details} = 0$, $d_{profil} = 6.7\ cm$ et $d_{trajectoire} = 7.4\ cm$), ayant résulté en la suppression de 793 693 points sur les 2 440 416 points d'origine (décimation d'environ 30% des points). Nous voyons que la décimation ne va opérer que pour les points proches du véhicule, c'est-à-dire les points sur la route et donc laisser intact les points de la façade pour la reconstruction.

Dans le cas du nuage *MINES_Rover_Indoor*, la méthode d'acquisition est différente de celle de LARA3D, nous utilisons donc la définition de d_{deci} à partir de d_{mean} ce qui nous donne $d_{mean} = 5.6\ \ mm$. Nous avons choisi le paramètre $d_{details} = 20\ mm$ de manière à être à un niveau de détail plus grossier et donc d'obtenir une cartographie 3D plus légère, mais aussi pour homogénéiser le nuage qui possède de grandes différences de densités locales suivant la position du robot MINES_Rover par rapport à l'objet mesuré. Cela nous a donc donné un paramètre de décimation $d_{deci} = \max(\frac{5.44}{2}, 20) = 20$. Le résultat de cette

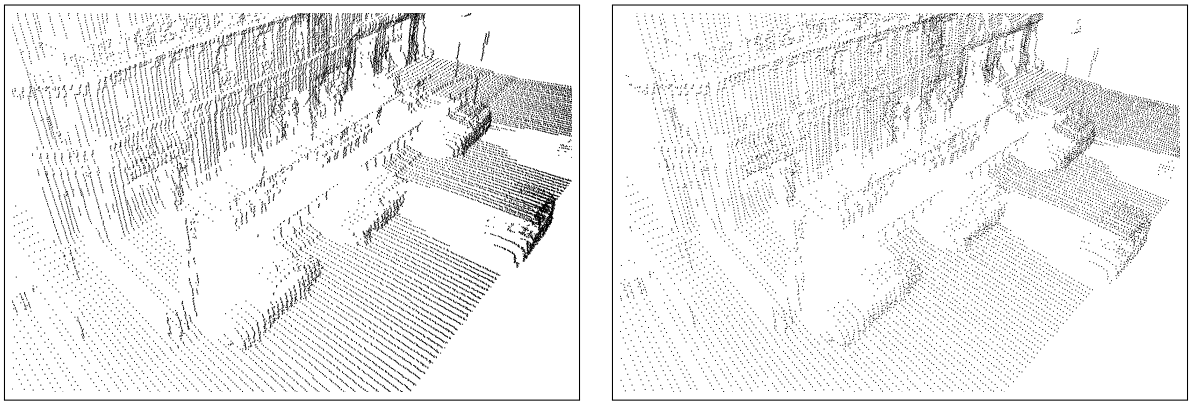


FIGURE 3.9 – A gauche, nuage de points brut *LARA3Dv2007_Soufflot*. A droite, le nuage de points décimé avec la distance $d_{deci} = 10.3\ cm$ (décimation d'environ 30% des points).

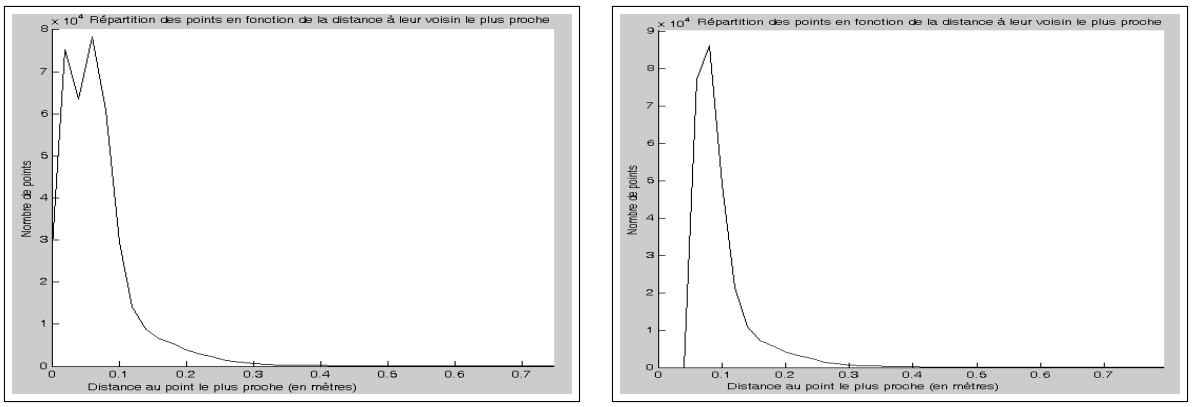


FIGURE 3.10 – A gauche, histogramme des distances du nuage de points brut *LARA3Dv2007_Soufflot*. A droite, histogramme des distances après décimation avec la distance $d_{deci} = 10.3\ cm$.

décimation est représenté sur la figure 3.12 qui a enlevé 2 269 834 points sur 2 836 369 points que comptait le nuage brut (décimation d'environ 80% des points). Ainsi, notre étape de décimation a permis pour ce nuage de points de passer très rapidement d'un nuage de points de 2 836 369 points à un nuage de points de seulement 566 535 points en seulement 10.5 s tout en gardant un très bon niveau de détail.

3.2.4 Décimation par facteur de qualité

Avec le calcul des normales (méthode décrite dans la section 3.3), nous pouvons calculer le facteur de qualité de chaque point comme défini au chapitre 2, c'est-à-dire l'angle entre la direction du laser et la normale. Nous décimons ensuite les points qui ont un angle supérieur à un paramètre θ et ce pour deux raisons : d'une part, ce sont des points dont la position n'est pas fiable à cause de l'angle d'incidence du laser (ils peuvent donc engendrer des erreurs dans la reconstruction) et d'autre part, ils peuvent avoir été incorrectement orientés.

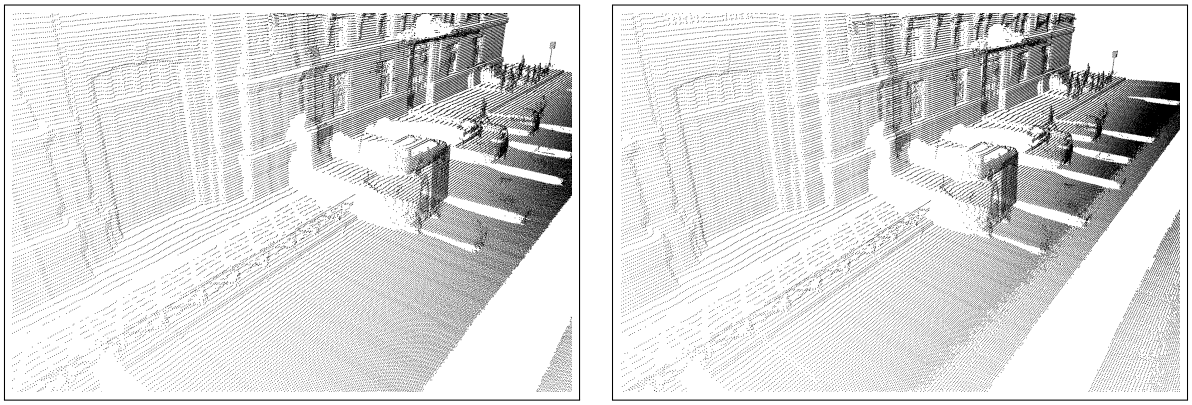


FIGURE 3.11 – A gauche, nuage de points brut *LARA3Dv2010_Orsay*. A droite, le nuage de points décimé avec la distance $d_{deci} = 7.05\ cm$ (décimation d'environ 30% des points).

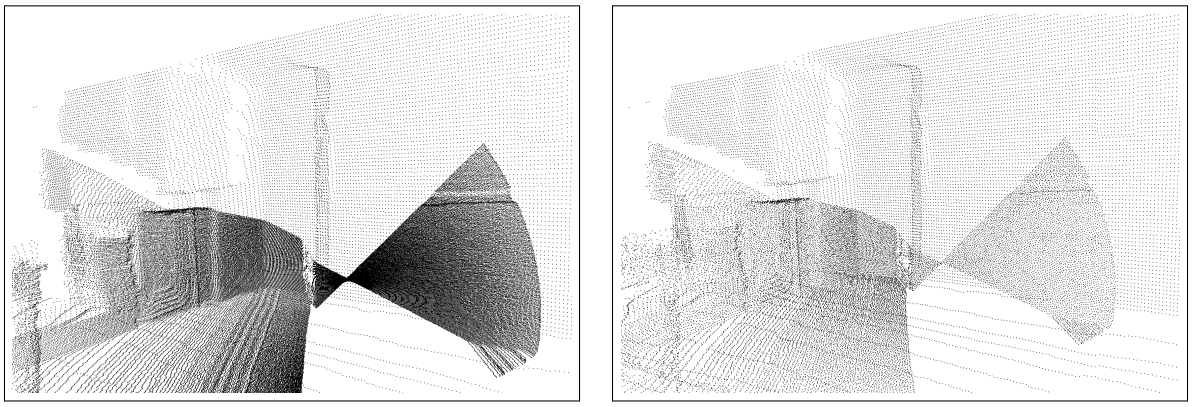


FIGURE 3.12 – A gauche, nuage de points brut *MINES_Rover_Indoor*. A droite, le nuage de points décimé avec la distance $d_{deci} = 20\ mm$ (décimation d'environ 80% des points).

En effet, pour des points qui ont un angle important entre la normale et la direction du laser, il suffit d'un peu de bruit pour que l'orientation de la normale soit inversée. Nous avons choisi $\theta = 80\,^\circ$ car nous avons vu que pour des bruits entre 0 et 20% de la distance moyenne entre points, l'erreur sur l'estimation de la normale est en moyenne compris entre 2 et $10\,^\circ$. Nous supprimons aussi les points qui n'ont pas de normale, c'est-à-dire les points qui n'ont pas assez de voisins dans leur voisinage local car ils n'ont pas assez de voisins pour pouvoir représenter une partie de la surface. La figure 3.13 montre en rouge les points décimés par ce facteur qualité pour le nuage *LARA3Dv2010_Orsay*. Dans ce cas, 28 561 points ont été supprimés. Cette décimation permet notamment d'enlever tous les points capturés dans les rues perpendiculaires à la rue acquise par le véhicule.

3.2.5 Conclusion sur la décimation proposée

La méthode étudiée dans cette section permet de diminuer le nombre de points d'un nuage. Une étude du paramètre de décimation a été apportée pour les nuages de points

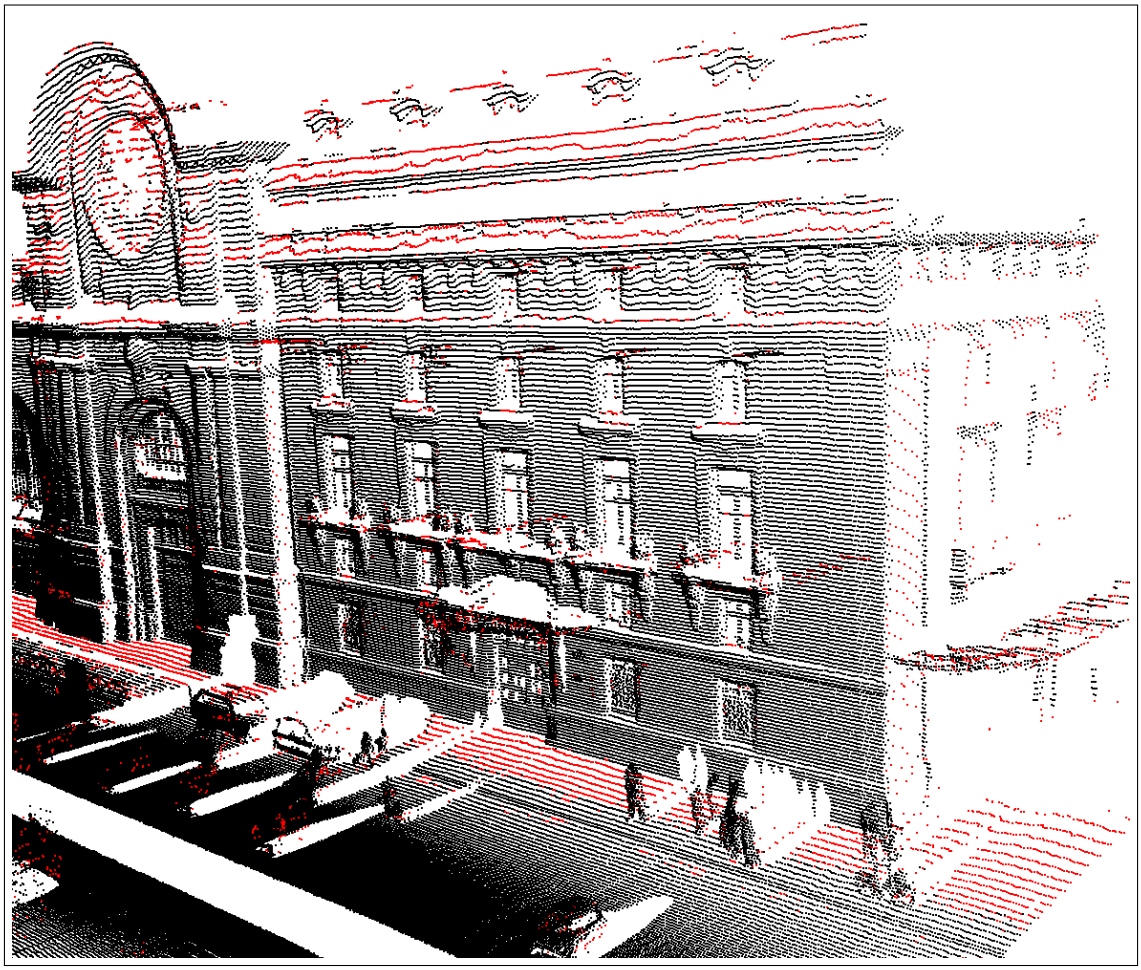


FIGURE 3.13 – Nuage de points *LARA3Dv2010_Orsay* avec en rouge les points décimés par facteur qualité.

de systèmes mobiles. Cette méthode de décimation est particulièrement rapide tout en offrant une qualité supérieure par rapport à une décimation aléatoire et cela à l'aide d'une sélection des points maximisant une approximation de la courbure locale. Nous proposons en outre l'utilisation d'une décimation supplémentaire permettant d'enlever les points potentiellement imprécis (imprécision liée à l'angle d'incidence du laser lors de la numérisation).

3.3 Calcul de normales

Nous allons voir dans cette partie l'étape de calcul de normales dans un nuage de points. Il s'agit de faire une estimation de la normale en chaque point. Cette information est nécessaire pour notre méthode de segmentation ainsi que pour notre méthode de reconstruction de surfaces. De la bonne estimation des normales dépendra la qualité de la reconstruction.

3.3.1 Etat de l'art

[Hoppe et al., 1992] présentent une méthode classique de calcul de normales, que nous détaillons ici car elle permet de présenter les notions de base du calcul de normales.

En chaque point p_i du nuage de points, on calcule tout d'abord son voisinage local V_i (soit les k plus proches voisins, soit les points dont la distance à p_i est inférieure à δ). Ensuite, on calcule les paramètres du plan minimisant les distances orthogonales des points du voisinage au plan : on cherche un plan des moindres carrés minimisant la somme $\sum_{j\in V_i}(n_i \cdot p_j + d_i)^2$ où n_i est la normale du plan et d_i la distance du plan à l'origine. Comme expliqué dans [Sharkarji, 1998], le barycentre b_i des points du voisinage appartient au plan et la résolution du plan des moindres carrés correspond à chercher le vecteur propre de la plus petite valeur propre de la matrice de covariance C_i du voisinage. On a alors $C_i = \sum_{j\in V_i} {}^t(p_j - b_i)(p_j - b_i)$. On note $\lambda_i^1, \lambda_i^2, \lambda_i^3$ avec $\lambda_i^1 \leq \lambda_i^2 \leq \lambda_i^3$ les valeurs propres de la matrice de covariance et v_i^1, v_i^2, v_i^3 les vecteurs propres correspondants. On a au point $p_i : \lambda_i^1 = \sum_{j\in V_i}(n_i \cdot p_j + d_i)^2$ et donc on choisit la normale au point $p_i : n_i = v_i^1$. La normale au point p_i correspond donc à la normale du plan des moindres carrés du voisinage. Nous noterons cette méthode PF pour « *Plane Fitting* ».

Cette méthode de calcul est l'une des plus rapides et reste relativement robuste au bruit lorsque l'on augmente la taille du voisinage. Une des difficultés de cette méthode est justement la bonne sélection du voisinage (le paramètre k ou le paramètre δ) pour avoir une estimation la plus proche possible de la normale à la surface : un voisinage trop petit est source d'erreur à cause du bruit et un voisinage trop grand va lisser les données surtout dans les zones de fortes courbures. [Mitra et al., 2004] font une analyse des valeurs propres de la matrice de covariance de manière à majorer l'erreur, plus exactement l'angle entre la normale estimée et la normale à la surface. Avec un bruit qui suit une loi normale de variance σ, une valeur optimale pour la majoration est d'avoir un voisinage qui va varier suivant la densité locale et la courbure locale : $\delta_i = (\frac{1}{\kappa_i}(c_1\frac{\sigma}{\sqrt{\epsilon\rho_i}} + c_2\sigma^2))^{\frac{1}{3}}$ où κ_i est la courbure moyenne locale au point p_i, ρ_i la densité locale et c_1 et c_2 des constantes. Mais la courbure moyenne locale κ_i et la densité locale ρ_i sont définis aussi à partir du voisinage local, c'est pourquoi [Mitra et al., 2004] font plusieurs itérations de leur algorithme. Une première valeur de δ identique en tout point permet de déterminer ρ_i et κ_i en fonction du point p_i avec $\rho_i = \frac{k}{\pi\delta^2}$ et $\kappa_i = \frac{2d_i}{\mu_i}$, courbure calculée suivant la méthode de [Gumhold et al., 2001]. Un deuxième passage permet de calculer δ_i en chaque point en fonction de ρ_i et de δ_i calculés lors de la première itération. Avec cette nouvelle valeur de δ_i, on peut améliorer les grandeurs de ρ_i et de δ_i en chaque point puis de nouveau évaluer δ_i et ainsi de suite jusqu'à

convergence. Ce schéma itératif permet d'estimer le voisinage local, la densité locale et la courbure locale mais nécessite de nombreuses itérations dans le seul but d'améliorer le calcul du voisinage.

[Pauly et al., 2003] ont amélioré la méthode de calcul de plan local sur un nuage de points en ajoutant un poids à chaque point dans le voisinage local. Plus un point est proche de p_i, plus son poids est grand et proche de 1. Inversement, plus un point est loin et plus son influence sur l'estimation de la normale doit être minime et donc son poids proche de 0. Pour cela, ils utilisent un poids sous forme de gaussienne. On calcule le plan en minimisant la somme : $\sum_{j \in V_i} \omega_j \, (n_i \cdot p_j + d_i)^2$ où $\omega_j = \theta(\|p_j - p_i\|)$ et $\theta(r) = e^{-\frac{r^2}{h^2}}$. h étant un paramètre pour désigner la taille de la fenêtre locale. Dans le cas d'un voisinage défini avec les k plus proches voisins, on prend h comme fonction de la distance entre p_i et le $k$ième voisin. Dans le cas d'un voisinage défini par un rayon δ, h est fonction de δ. Nous noterons cette méthode WPF pour « *Weighted Plane Fitting* ».

[Dey et al., 2005b] présentent et comparent trois méthodes d'estimation de normales. Ils comparent les deux méthodes précédentes avec une méthode de [Dey et al., 2005a] basée sur le diagramme de Voronoï du nuage de points. L'idée est pour un point p_i, de prendre l'extrémité la plus éloignée de sa cellule de Voronoï (définition de cellule de Voronoï au chapitre 4). La droite reliant p_i et cette extrémité est une bonne approximation de la normale à la surface. Pour être robuste au bruit, [Dey et al., 2005a] suggèrent de ne pas choisir l'extrémité la plus éloignée dans la cellule de Voronoï de p_i mais l'extrémité la plus éloignée dont la boule de Delaunay duale est suffisamment *grande*. Le critère *grand* est fonction de la distance moyenne du point p_i à ses k plus proches voisins. Cette méthode est appelée BDB pour « *Big Delaunay Balls* ». [Dey et al., 2005b] concluent que WPF est meilleur dans les cas où le bruit est plus faible et BDB est meilleur avec un bruit plus élevé. Dans le cas de nuage de points avec plus de 5 millions de points, WPF est aussi à préférer pour sa rapidité. BDB nécessite la construction du diagramme de Voronoï du nuage de points, ce qui en fait une méthode globale et non locale.

Au lieu d'ajuster un plan, [Meek et Walton, 2000] et [OuYang et Feng, 2005] ajustent une quadrique sur les points du voisinage et calculent la normale à la surface quadrique pour estimer la normale du point. Il est possible d'étendre les surfaces quadriques à l'approximation locale de surfaces polynomiales de degré d, appelées d-jets dans [Cazals et Pouget, 2003] et appelées « *Moving Least Squares* » (MLS) dans [Levin, 2003] lorsque l'on rajoute un poids aux points du voisinage en fonction de leur distance au point p_i. Nous allons expliciter la méthode d'ajustement locale de surfaces polynomiales dans le cas des MLS et l'estimation de quantités différentielles de la surface polynomiale pour le calcul de la normale.

Une surface paramétrée est définie de manière explicite pour chaque point $p_i : X(u, v) = (u, v, g_i(u, v))$ où (u, v) est un plan de paramétrisation et g_i est un polynôme à deux variables, de degré d. Pour commencer, nous devons définir un plan de paramétrisation pour la surface et pour cela, calculer le plan local par moindre carré pondéré (c'est-à-dire WPF de [Pauly et al., 2003]). Ce plan local va servir de repère pour définir la courbe polynomiale approximant la surface. Sachant que la matrice de covariance pondérée C_i du voisinage est une matrice symétrique réelle donc diagonalisable dans une base orthonormée, alors les vec-

teurs propres (v_i^1, v_i^2, v_i^3) de la matrice C_i forment un repère orthogonal. Nous définissons le repère local du point p_i composé des vecteurs (v_i^2, v_i^3, v_i^1) et le point p_i comme origine. Comme expliqué dans [Levin, 2003], nous pouvons calculer un polynôme à deux variables $g_i(u,v) = \sum_{k+l \leq d} a_{k,l} u^k v^l$ qui va minimiser la somme pondérée suivante :

$$\sum_{j \in V_i} (g_i(u_j, v_j) - f_j)^2 e^{-\frac{\|p_j - p_i\|^2}{h^2}} \tag{3.2}$$

où (u_j, v_j, f_j) sont les coordonnées locales du point p_j dans le repère (v_i^2, v_i^3, v_i^1), f_j correspondant à la distance signée du point p_j au plan local.

La figure 3.14 montre en 2D le calcul d'une surface MLS (« *Moving Least Squares* »). Le plan local en noir est calculé par moindres carrés à partir des points du voisinage. Un repère local est alors défini à partir du point p_i et des vecteurs propres de la matrice de covariance. La surface MLS g_i de degré d en rouge approxime les points du voisinage en fonction de leur distance au point p_i.

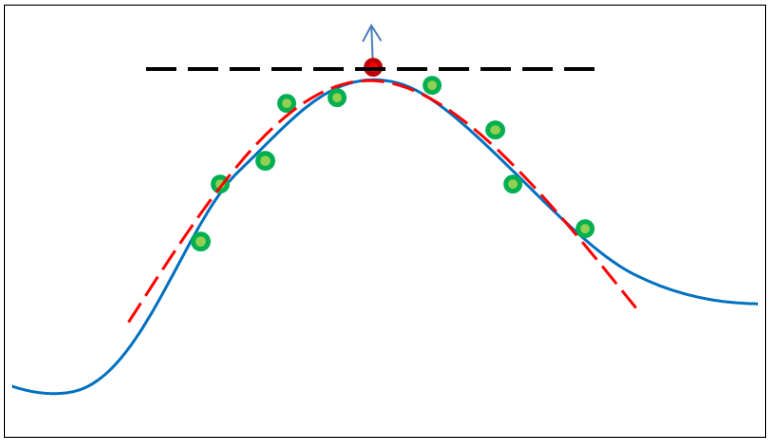


FIGURE 3.14 – Point p_i en rouge, points du voisinage en vert, plan local par moindre carré en noir, projection du point p_i sur le plan local en noir (centre du repère local), surface MLS calculée en rouge et surface réelle en bleu.

Avec les propriétés des surfaces différentielles, nous connaissons la normale à la surface paramétrée $X(u,v) = (u, v, g_i(u,v))$ au point $(0,0)$ qui correspond au point p_i. Pour une surface paramétrée $X(u,v)$, la normale à la surface en $(0,0)$ est égale à $n_i = \dfrac{\partial_u X(0,0) \wedge \partial_v X(0,0)}{\|\partial_u X(0,0) \wedge \partial_v X(0,0)\|}$. [Cazals et Pouget, 2003] démontrent que pour un polynôme de degré d et pour une surface sans bruit, la normale en p_i est estimée avec une approximation en $O(\mu_i^d)$ où μ_i est la distance moyenne des points du voisinage à p_i. En présence de bruit, cette approximation ne sera plus valable. Mais ceci nous permet de faire la conjecture que pour des données avec un bruit faible, il est possible que l'approximation locale par des surfaces polynomiales de degré 2 ou 3 permettent de mieux estimer la normale qu'un plan tangent.

En plus de toutes ces méthodes d'approximation locale, il existe une deuxième catégorie d'estimation de normales par calcul de moyenne. Pour cela, on construit tout d'abord une triangulation locale autour du point p_i. Pour une triangulation locale composée de K triangles autour de p_i, la normale $n_i = \frac{\sum_{j=1\to K} w_j n_j^t}{\sum_{j=1\to K} w_j}$ où n_j^t et w_j sont les normales et les poids associés à chaque triangle. w_j peut être égal à 1 (méthode Mean_Tri), à l'aire de chaque triangle voisin (méthode Area_Tri) ou l'angle de chaque triangle au point p_i (méthode Angle_Tri) comme expliqué dans [Meek et Walton, 2000]. La triangulation locale peut être construite de plusieurs façons, par projection des points dans le plan tangent puis triangulation de Delaunay en 2D ou alors à l'aide du diagramme de Voronoï.

Enfin, une troisième catégorie d'estimation consiste à améliorer les normales par des méthodes de filtrage. Tout d'abord, une première estimation des normales est effectuée avec l'une des méthodes présentées plus haut (souvent la méthode la plus simple, robuste et la plus rapide PF ou WPF). Dans une seconde étape, la normale en chaque point p_i est gardée comme la moyenne pondérée des normales des points de son voisinage, c'est-à-dire $n'_i = \frac{\sum_{j\in V_i} w_j n_j}{\sum_{j\in V_i} w_j}$. Pour le filtre bilatéral, on prend les poids $w_j = e^{-\frac{\|p_j-p_i\|^2}{h^2}} e^{-\frac{\|n_j-n_i\|^2}{a^2}}$. [Oztireli et al., 2009] proposent un schéma itératif appelé IRLS pour « *Iteratively Reweighted Least Squares* » avec la normale filtrée n'_i la limite de la suite $n_i^k = \frac{\sum_{j\in V_i} w_j n_j}{\sum_{j\in V_i} w_j}$ avec $w_j = e^{-\frac{\|p_j-p_i\|^2}{h^2}} e^{-\frac{\|n_j-n_i^{k-1}\|^2}{a^2}}$. Cela ressemble au filtre bilatéral avec une minimisation par itérations successives. Comme initialisation, [Oztireli et al., 2009] suggèrent de prendre $n_i^0 = \frac{\sum_{j\in V_i, j\neq i} w_j n_j}{\sum_{j\in V_i, j\neq i} w_j}$ avec $w_j = e^{-\frac{\|p_j-p_i\|^2}{h^2}}$, qui consiste à prendre la moyenne pondérée des normales du voisinage excepté la normale en p_i de manière à être plus robuste au bruit. Ces filtrages se montrent très efficaces lorsque les données sont très bruitées mais à l'inverse risquent de lisser les normales lorsqu'il y a peu de bruit dans le nuage de points.

3.3.2 Méthode proposée pour le calcul de normales : Filtered MLS

La plupart des méthodes d'estimation de normales font l'hypothèse que tous les points d'un voisinage appartiennent à la même surface de classe C^2. La figure 3.15 montre à gauche un exemple de normales telles que nous l'avons défini au début de ce chapitre pour deux plans orthogonaux. Les normales sont bien orthogonales aux plans sauf au point de discontinuité, où la normale est la moyenne des normales des deux plans. A droite de la figure 3.15, nous montrons ce que donnerait une méthode d'approximation comme WPF (« *Weighted Plane Fitting* »). En considérant tous les points du voisinage comme appartenant à la même surface de classe C^2, l'erreur sur l'estimation des normales augmente à l'approche du point de discontinuité.

[Fleishman et al., 2005] ont étudié cette question en classant les points d'un voisinage en deux catégories : les « *inliers* », c'est-à-dire les points qui appartiennent à la même surface que le point courant p_i, et les « *outliers* », les points qui n'appartiennent pas à la même surface que p_i. Pour cela, ils utilisent un estimateur robuste (« *Least Median of*

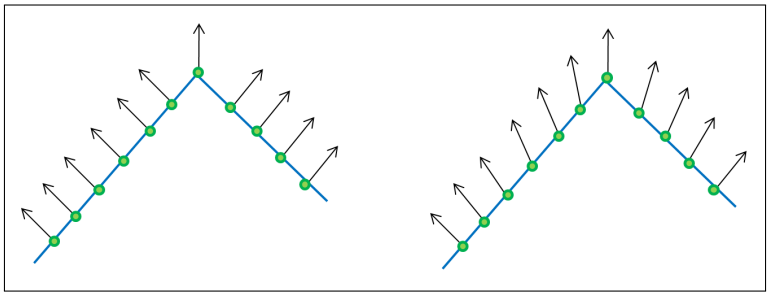


FIGURE 3.15 – A gauche, normales telles que nous l'avons défini. A droite, normales calculées par une méthode comme WPF (« *Weighted Plane Fitting* »).

Squares ») pour déterminer les paramètres de la surface MLS locale avec les « *inliers* » et les « *outliers* ». Ils sélectionnent p échantillons au hasard (où p est le nombre de paramètres du modèle) parmi les points du voisinage et calculent le médian des résidus des points restants dans le voisinage. Ce processus est répété T fois afin de garder le modèle qui minimise le médian des résidus. La méthode « *Least Median of Squares* » est en statistique un algorithme qui permet d'approximer un modèle même en présence de 50% d'« *outliers* ». [Sheung et Wang, 2009] utilisent un autre estimateur plus robuste, le MDPE (« *Maximum Density Power Estimator* ») de [Wang et Suter, 2004], capable de tolérer jusqu'à 85% d'« *outliers* ». Mais l'inconvénient de tous ces estimateurs robustes réside dans le grand nombre d'itérations nécessaires en chaque point p_i pour obtenir un bon modèle approximant les données. [Sheung et Wang, 2009] nous donne le chiffre d'une heure et demi de traitement pour un nuage de 250 000 points sur un bon PC de 2009 avec un seul *thread* et uniquement sur CPU. Il semble clair que pour des questions de temps de calcul une telle méthode est inutilisable sur nos données de plusieurs millions de points.

Pour notre méthode de calcul des normales, nous souhaitons aussi être capable pour chaque voisinage de déterminer les « *inliers* » et les « *outliers* », c'est-à-dire sélectionner les voisins qui appartiennent à la même surface, mais de manière plus rapide que les méthodes de [Fleishman et al., 2005] et de [Sheung et Wang, 2009]. Pour cela, nous opérons en deux étapes. La première étape va permettre de déterminer une première approximation de la normale en chaque point en utilisant la méthode PF (« *Plane Fitting* ») de [Hoppe et al., 1992]. Dans la seconde étape, nous partons du voisinage V_i de p_i et cherchons le sous-ensemble V_i' des « *inliers* ». Au lieu d'estimer un modèle de manière robuste avec un grand nombre d'itérations comme les méthodes de [Fleishman et al., 2005] et de [Sheung et Wang, 2009], nous allons faire une approximation et sélectionner dans le voisinage les points qui semblent appartenir à la même surface que p_i à l'aide des normales calculées lors de la première étape. En chaque point p_j du voisinage V_i, nous calculons le poids $w_j = e^{-\frac{\|p_j - p_i\|^2}{h^2}} e^{-\frac{\|n_j - n_i\|^2}{a^2}}$ qui représente la différence de normales et la distance entre p_i et p_j. Si ce poids est supérieur à 0.5, alors nous considérons le point p_j comme « *inlier* » sinon nous le considérons comme « *outlier* ». Ceci permet d'obtenir immédiatement un nouveau voisinage V_i' sous-ensemble de V_i. C'est à partir de ce nouveau voisinage que nous calculons la surface MLS locale comme expliqué plus haut en minimisant la somme : $\sum_{j \in V_i'} (g_i(u_j, v_j) - f_j)^2 e^{-\frac{\|p_j - p_i\|^2}{h^2}}$. La

normale est alors égale à $n_i = \dfrac{\partial_u X(0,0) \wedge \partial_v X(0,0)}{\|\partial_u X(0,0) \wedge \partial_v X(0,0)\|} = \frac{1}{\sqrt{1+a_{1,0}^2+a_{0,1}^2}}(-a_{1,0}, -a_{0,1}, 1)$ dans le repère (v_i^2, v_i^3, v_i^1) et où $a_{k,l}$ sont les coefficients de la surface MLS polynomiale obtenue par moindres carrés : $g_i(u,v) = \sum_{k+l \leq d} a_{k,l} u^k v^l$.

Les poids w_j pour la sélection des « *inliers* » sont très semblables aux poids du filtre bilatéral. Mais contrairement à celui-ci, nous n'utilisons pas les poids comme facteur d'estimation de la nouvelle normale mais simplement pour savoir si un point est considéré ou non comme appartenant à la même surface. Ce filtrage permet d'enlever les points qui ont une normale très différente de celle de p_i et donc qui ont une grande chance de ne pas appartenir à la même surface. Nous montrons à gauche de la figure 3.16, pour un point p_i (en rouge) les « *inliers* » sélectionnés (en vert). A droite, la nouvelle normale n_i du point p_i est égale à la normale de la surface MLS en noire calculée sur le sous-ensemble des « *inliers* » V_i'. Le fait de sélectionner un sous-ensemble V_i' du voisinage de p_i permet ainsi de mieux estimer la normale.

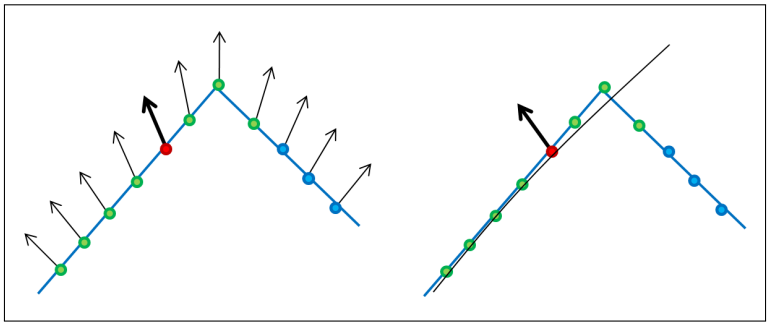


FIGURE 3.16 – A gauche, sélection des « *inliers* » (en vert)dans le voisinage de p_i (en rouge). A droite, estimation de la surface MLS en noire et de la normale à partir du nouveau voisinage V_i' des « *inliers* » (en vert).

Nous appelons notre méthode « Filtered Moving Least Squares » ou FMLS. Elle s'apparente à des méthodes de filtrage comme le filtrage bilatéral car nous utilisons des normales déjà calculées. Mais elle diffère dans le sens où ce filtrage sert uniquement pour extraire un sous-échantillon du voisinage.

Une première version de cette méthode a été publiée à la conférence [$3DPVT, 2010$].

3.3.3 Analyse et comparaison des résultats sur des données simulées et réelles

Pour évaluer notre méthode de calcul de normales, nous avons testé et comparé les algorithmes décrits plus haut : PF, WPF, MLS, Area_Tri, PF+Bilateral, PF+IRLS avec notre méthode FMLS. Pour les tests, nous avons pris trois exemples de nuages de points : *S_Cube* et *S_Fandisk* qui sont des données simulées et dont nous avons le nuage de points sans bruit et le nuage de points *CYBERWARE_Bunny* de l'Université de Stanford acquis avec un scanner Cyberwxare 3030 MS et peu bruité. Pour chaque nuage de points,

nous avons généré des normales de référence. Pour cela, nous avons construit le maillage de *S_Cube* et de *S_Fandisk* à partir du nuage sans bruit et le maillage à partir de *CYBERWARE_Bunny* sans ajouter de bruit. Nous avons ensuite estimé la normale en chaque point avec la méthode Area_Tri qui donne les meilleurs résultats lorsque il y a un minimum de bruit dans les données. La figure 3.17 présente les étapes de calcul des normales de référence. Nous pouvons voir avec le gros plan à gauche que les normales de référence calculées de cette façon représentent une très bonne approximation des normales telles que nous les avons définis au début de ce chapitre.

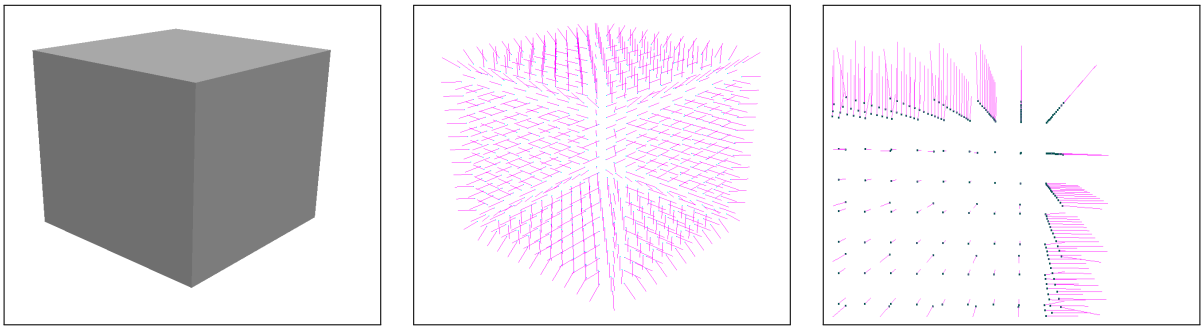


FIGURE 3.17 – A gauche, maillage construit à partir de *S_Cube* sans bruit. Au milieu, normales calculées avec la méthode Area_Tri pour générer les normales de référence. A droite, gros plan sur un des sommets.

Pour chaque nuage de points, nous avons ajouté du bruit entre 0% et 100% de la distance moyenne entre points avec un pas de 5%. Et nous avons testé toutes les méthodes pour chaque nuage bruité. Nous nous sommes limité à un bruit variant entre 0% et 100% car comme nous l'avons vu au chapitre 2, l'échelle de bruit que nous avons pour des données LiDAR et particulièrement pour nos données de cartographie mobile varie entre 5% et 20% de la distance entre points. Afin de ne pas dépendre du choix de la distance locale pour la comparaison des méthodes, nous avons calculé pour chaque méthode et pour chaque niveau de bruit le voisinage local permettant d'obtenir les meilleurs résultats (en faisant varier la distance locale δ_{normal} entre 1 et 8 fois la distance moyenne entre points d_{mean}). Les résultats des normales de chaque méthode pour chaque nuage de points bruité sont comparés aux normales de référence en calculant la moyenne sur tous les points de l'angle entre la normale estimée et la normale de référence.

Pour tous les nuages de points, nous avons effectué une première étape de décimation comme détaillé dans la section précédente avec $d_{deci} = \frac{d_{mean}}{2}$. Pour toutes les méthodes à base de noyau gaussien utilisant h, nous avons pris $h = \frac{\delta_{normal}}{2}$. Pour le filtre bilatéral, le filtre IRLS (« *Iteratively Reweighted Least Squares* ») et notre méthode FMLS, nous avons pris $a = \frac{1}{\sqrt{2}}$. Pour les méthodes basées sur les surfaces MLS, nous avons considéré des surfaces polynomiales de degré 2, ce qui donne 6 coefficients à déterminer par moindre carré c'est pourquoi les voisinages locaux devaient contenir au moins 7 points. Toutes les méthodes ont été exécutées sur un QuadCore Q9300 2.50 GHz, 2 Go de RAM, sur un seul *thread* et sur CPU.

La figure 3.18 représente les résultats de toutes les méthodes testées sur le nuage de points *S_Cube* pour des niveaux de bruit (écart-type) variant de 0% à 100% avec un pas de 5%. La méthode FMLS donne de meilleurs résultats que toutes les autres méthodes, sauf à très faible niveau de bruit. Pour un niveau de bruit très faible, c'est-à-dire entre 0% et 8% de la distance moyenne entre points, c'est la méthode Area_Tri qui donne les meilleurs résultats ce qui s'explique par le fait que le maillage représente très bien la surface et les normales des triangles sont une bonne approximation de la normale à la surface. En revanche, quand le niveau de bruit augmente, l'erreur avec cette méthode devient très importante car les triangles vont être très bruités. On ne peut alors plus utiliser seulement les triangles voisins pour calculer la normale en un point. Quand le niveau de bruit dépasse 8% de la distance moyenne entre points, nous voyons que c'est notre méthode FMLS qui donne les meilleurs résultats et ceci jusqu'à 100% de la distance moyenne entre points. La grande différence de résultat entre FMLS et les autres méthodes s'explique par le fait que le nuage *S_Cube* comporte de nombreux points de discontinuité et notre méthode FMLS va alors mieux estimer les normales dans les voisinages qui comportent de nombreux « *outliers* » par rapport aux autres méthodes qui ne font pas la distinction « *inliers* »/« *outliers* ».

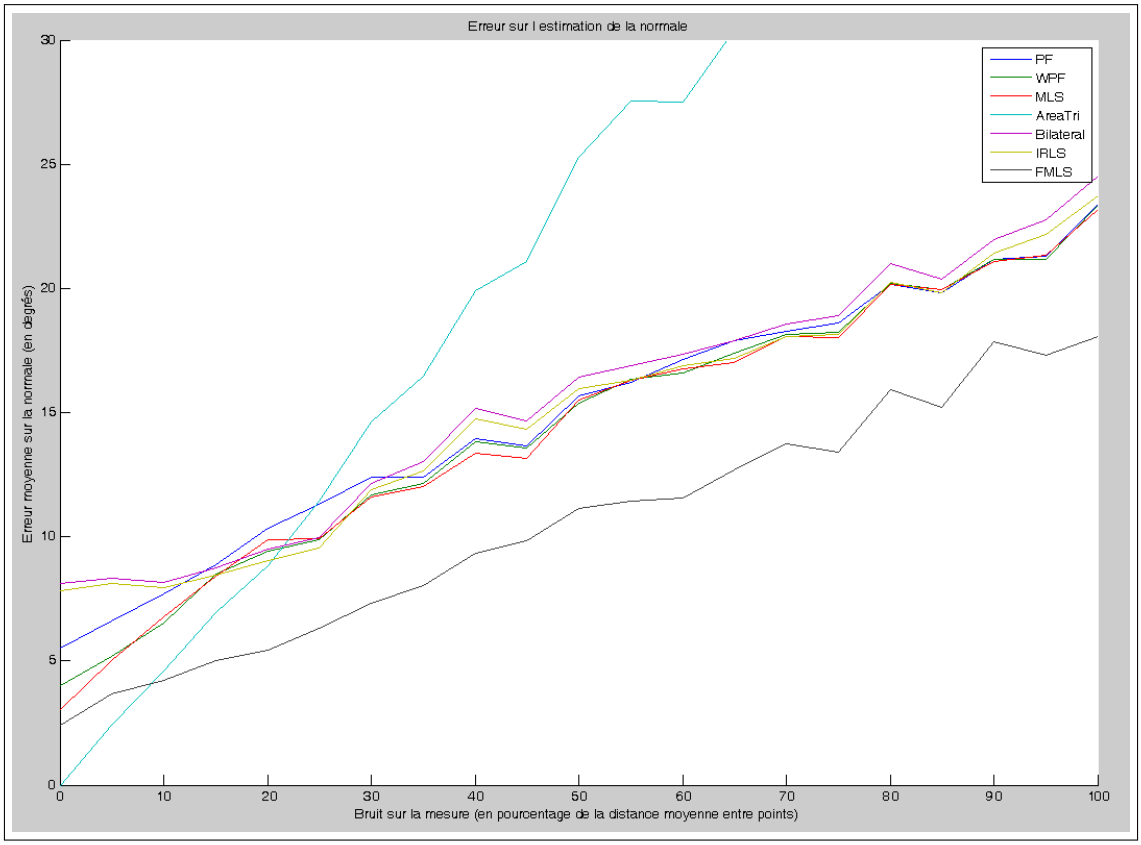


FIGURE 3.18 – Représentation de l'erreur sur l'estimation de normales pour le nuage de points *S_Cube* avec un bruit variant de 0% à 100% de d_{mean} pour 6 méthodes et notre méthode FMLS en gardant le meilleur résultat pour un voisinage variable entre d_{mean} et $8d_{mean}$.

Le graphique 3.18 a été généré en conservant à chaque test l'erreur minimale pour des voisinages variant entre 1 et 8 fois d_{mean}. Ceci nous a permis de comparer les méthodes de calcul de normales sans tenir compte du voisinage. Or, nous voulons aussi comparer ces méthodes pour une distance de voisinage fixée car dans la pratique nous ne connaissons pas exactement le bruit et nous ne pouvons pas optimiser le voisinage local sans référence. Le graphique 3.19 représente les résultats sur le même nuage *S_Cube* en fixant le voisinage local à $\delta_{normal} = 4d_{mean}$ quel que soit le bruit. Nous voyons que les résultats sont sensiblement différents. Area_Tri est toujours la meilleure méthode pour des bruits faibles puis à partir de 8% de bruit, c'est FMLS qui donne les meilleurs résultats jusqu'à un peu plus de 50%. Au delà de cette limite, c'est MLS (« *Moving Least Squares* ») qui donne les meilleurs résultats et enfin la méthode la plus basique PF (« *Plane Fitting* ») à partir de 80% de bruit. Ainsi, les méthodes d'estimation de normales sont sensibles au choix du voisinage. La nôtre est d'autant plus sensible à ce choix que c'est dans le voisinage que nous allons sélectionner les « *inliers* ». Si notre méthode est supplanté pour les très faibles bruits et les très forts bruits, elle reste cependant la plus performante dans la plage correspondant au bruit de nos données de cartographie, c'est-à-dire de 10% à 50%.

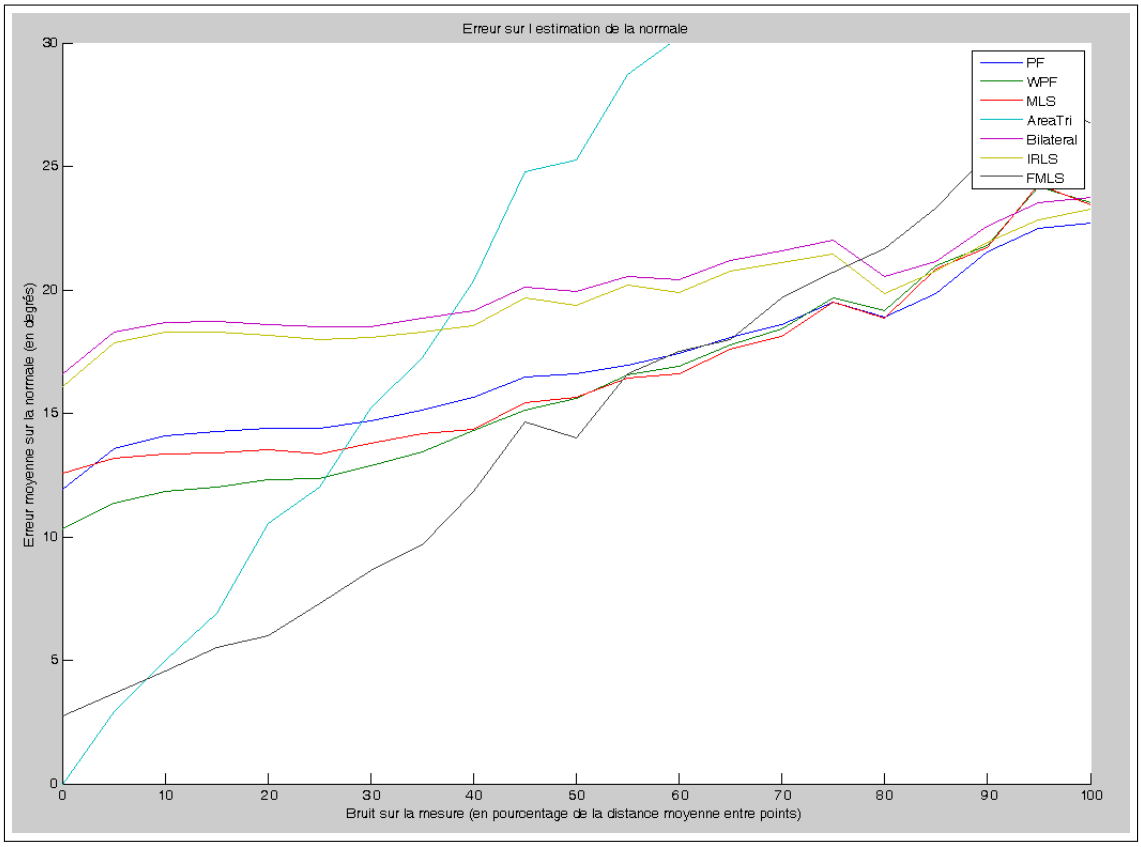


FIGURE 3.19 – Représentation de l'erreur sur l'estimation de normales pour le nuage de points *S_Cube* avec un bruit variant entre 0% et 100% de d_{mean} pour 6 méthodes et notre méthode FMLS avec $\delta_{normal} = 4d_{mean}$.

La figure 3.20 compare les résultats de calcul de normales pour les méthodes WPF

(« *Weighted Plane Fitting* ») et notre méthode FMLS (« *Filtered Moving Least Squares* ») pour un bruit de 10% et les compare avec les données de référence. Nous voyons que les normales calculées avec notre méthode sont plus proches des normales de référence.

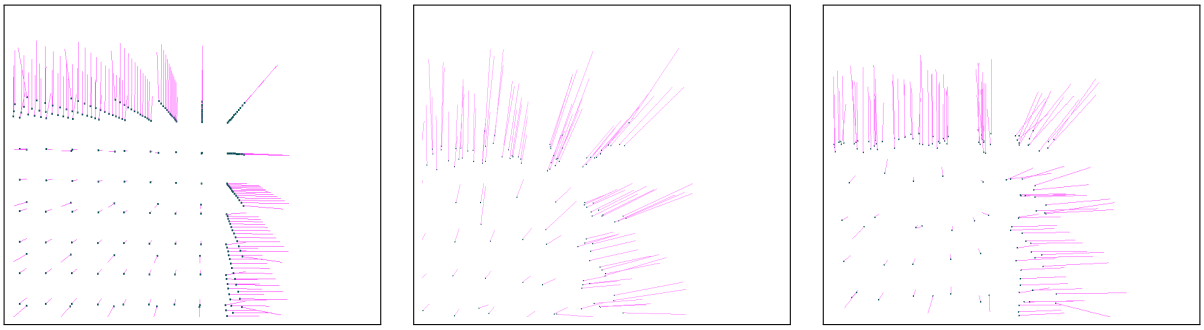


FIGURE 3.20 – A gauche, gros plan sur les normales de référence de *S_Cube*. Au milieu, normales calculées avec la méthode WPF. A droite, normales calculées avec notre méthode FMLS pour un bruit de 10%.

Nous avons effectué les mêmes comparaisons avec le nuage *S_Fandisk*. La figure 3.21 montre le calcul des normales de référence. Le graphique 3.22 présente les résultats des différentes méthodes avec un voisinage donnant le meilleur résultat à gauche et un voisinage fixe à droite. Les résultats sont très similaires à ce que nous avons obtenu avec le nuage de points *S_Cube*. Notre méthode donne les meilleurs résultats pour des niveaux de bruit entre 10 et 50% correspondant aux niveaux de bruits de nos données. Par contre, c'est la méthode IRLS (« *Iteratively Reweighted Least Squares* ») qui donne les meilleurs résultats lorsque le niveau de bruit est élevé. Ceci confirme le fait que ce sont normalement les méthodes de filtrage (comme le filtre bilatéral et l'IRLS) qui donnent les meilleurs résultats lorsque les données sont très bruitées. Nous pouvons aussi remarquer que pour un nuage de points avec un niveau de bruit de 10%, l'erreur sur l'estimation de la normale est inférieure à 5 °. La figure 3.23 montre les résultats de calcul de normales pour les méthodes IRLS et notre méthode FMLS pour un bruit de 10% et les compare avec les données de référence. Nous observons que les normales calculées avec notre méthode sont plus proches des normales de référence.

Afin de tester et de comparer les méthodes d'estimation de normales avec des données réelles, nous avons fait les mêmes évaluations sur le nuage de points *CYBERWARE_Bunny*. Nous avons pris un nuage de points avec peu de bruit pour que notre calcul de normales de référence avec la méthode Area_Tri reste suffisamment valable. Nous ne pouvions pas utiliser le nuage *LARA3D_Orsay* pour évaluer les méthodes de calcul de normales car le bruit n'était pas suffisamment bas pour permettre de calculer une normale de référence avec la méthode Area_Tri. La figure 3.24 montrent les étapes pour le calcul des normales de référence avec la méthode Area_Tri. Le graphique 3.25 présente les résultats des différentes méthodes avec un voisinage optimisé à gauche et un voisinage fixe à droite. Nous voyons que les résultats sont un peu différents de ceux obtenus avec *S_Cube* et *S_Fandisk* : la différence de résultat entre notre méthode FMLS et les autres méthodes est moins nette. Ceci s'explique par le fait que le nuage de points *CYBERWARE_Bunny* contient peu de

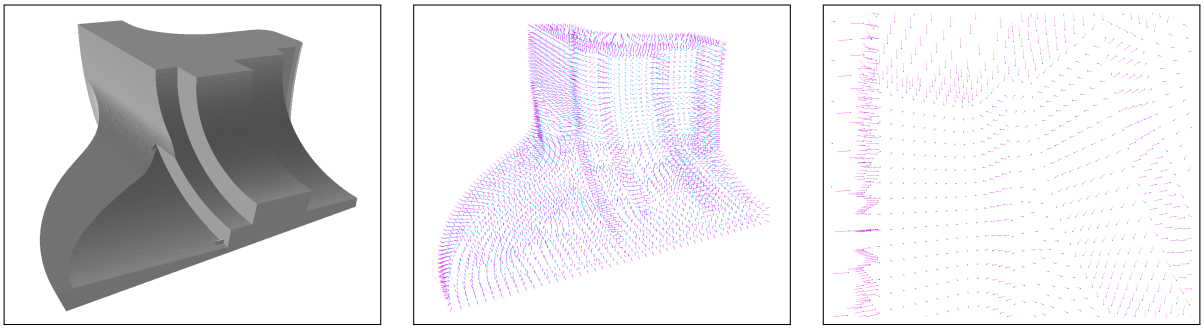


FIGURE 3.21 – A gauche, maillage construit à partir de *S_Fandisk* sans bruit. Au milieu, normales calculées avec la méthode Area_Tri pour générer les normales de référence. A droite, gros plan sur un détail.

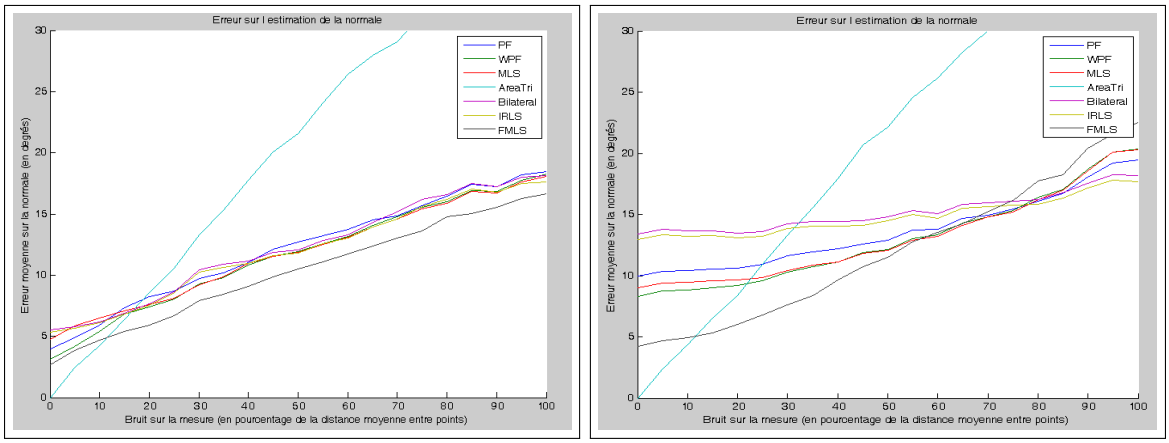


FIGURE 3.22 – A gauche, représentation de l'erreur sur l'estimation de normales pour le nuage de points *S_Fandisk* avec un bruit variant entre 0% et 100% de d_{mean} pour 6 méthodes et notre méthode FMLS en gardant le meilleur résultat pour un voisinage variable entre d_{mean} et $8d_{mean}$. A droite, même comparaison en fixant le voisinage $\delta_{normal} = 4d_{mean}$ quel que soit le bruit.

points de discontinuité. Or c'est au niveau des points de discontinuité que notre méthode de filtrage permet d'obtenir les meilleurs résultats. C'est pourquoi FMLS est adaptée au calcul de normales dans des scènes urbaines contenant de nombreux points de discontinuités (intersections de plans). Il est intéressant de voir sur le graphique de droite de la figure 3.25 que presque toutes les méthodes se croisent en un point autour de 66% de niveau de bruit et inversent presque leur ordre de résultat. La figure 3.26 montre les résultats de calcul de normales pour les méthodes Area_Tri et notre méthode FMLS pour un bruit de 10% et les compare avec les données de référence. Nous remarquons que les méthodes Area_Tri et FMLS donnent des résultats visuels très similaires pour un niveau de bruit de 10%.

Nous avons comparé les temps de calcul des différentes méthodes que nous avons testées. Le tableau 3.1 donne les temps de calcul des différentes méthodes moyennés pour les 3 nuages de points et pour plusieurs voisinages. L'étape de calcul de normale est l'étape la plus longue dans tout le processus de modélisation et représente jusqu'à 50% du temps total

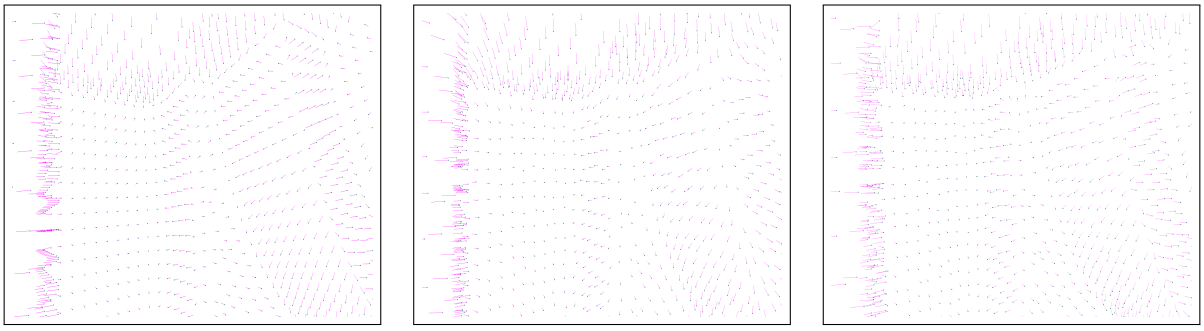


FIGURE 3.23 – A gauche, gros plan sur les normales de référence de *S_Fandisk*. Au milieu, normales calculées avec la méthode IRLS. A droite, normales calculées avec notre méthode FMLS pour un bruit de 10%.

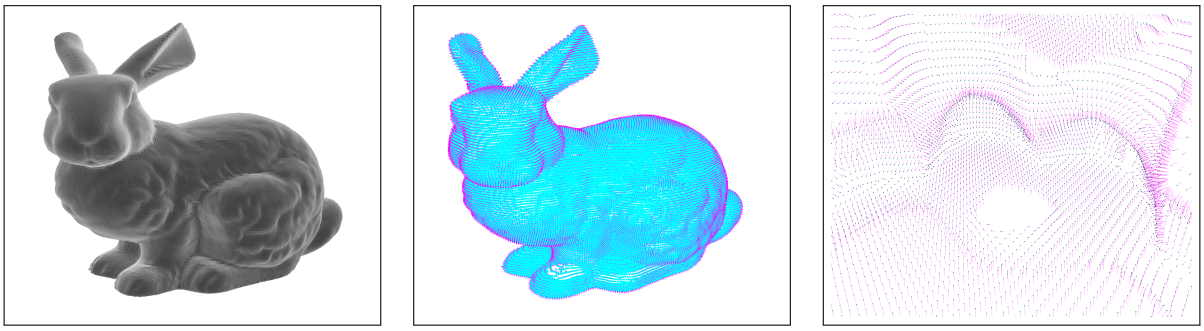


FIGURE 3.24 – A gauche, maillage construit à partir de *CYBERWARE_Bunny* sans bruit. Au milieu, normales calculées avec la méthode Area_Tri pour générer les normales de référence. A droite, gros plan sur un détail

de modélisation c'est pourquoi il est important de choisir une méthode suffisamment rapide. Nous constatons que la méthode Area_Tri est très lente à cause du calcul de voisinage par triangulation locale. Notre méthode FMLS est environ deux fois plus lente que la méthode la plus rapide PF mais le gain en qualité se situe entre 50% et 100% par rapport à cette même méthode pour des niveaux de bruit de 10%. Même pour un nuage de points sans bruit, notre méthode FMLS est une bonne alternative à la méthode Area_Tri car nous n'avons pas besoin de calculer de triangulation locale et nous obtenons des résultats bien meilleurs que le reste des méthodes de calcul de normales. En revanche, dès que le niveau de bruit dépasse 50%, nous devons optimiser le voisinage pour pouvoir utiliser notre méthode (par exemple en fonction du niveau de bruit avec une méthode comme [Mitra et al., 2004] ce qui augmenterait les temps de calcul) ou alors nous devons utiliser une méthode comme IRLS.

3.3.4 Conclusion sur la méthode proposée

La méthode de calcul de normales proposée a donné les meilleurs résultats sur les nuages testés pour des niveaux de bruit compris entre 10% et 50%. Nous pouvons donc supposer

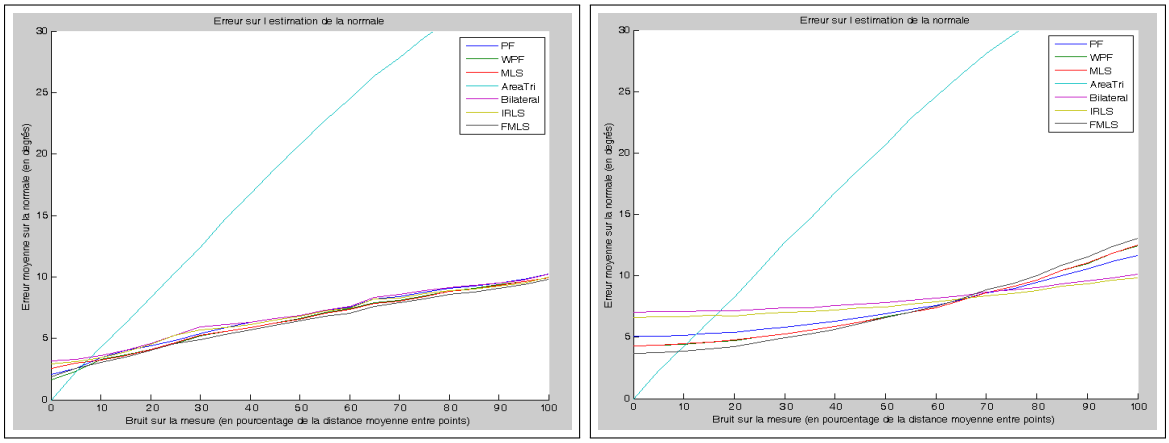


FIGURE 3.25 – A gauche, représentation de l'erreur sur l'estimation de normales pour le nuage de points *CYBERWARE_Bunny* avec un bruit variant entre 0% et 100% de d_{mean} pour 6 méthodes et notre méthode FMLS en gardant le meilleur résultat pour un voisinage variable entre d_{mean} et $8d_{mean}$. A droite, même comparaison en fixant le voisinage $\delta_{normal} = 4d_{mean}$ quel que soit le bruit.

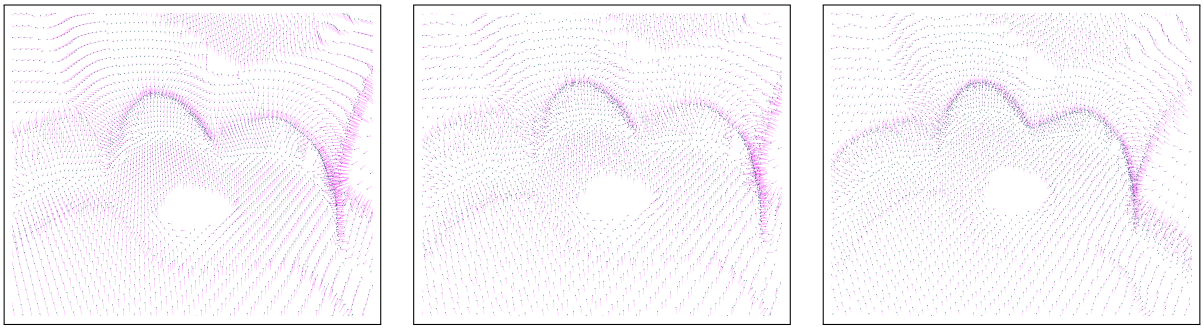


FIGURE 3.26 – A gauche, gros plan sur les normales de référence de *CYBERWARE_Bunny*. Au milieu, normales calculées avec la méthode Area_Tri. A droite, normales calculées avec notre méthode FMLS pour un bruit de 10%.

que nous obtenons de bons résultats pour nos données LiDAR de plateforme mobile qui ont des niveaux de bruit compris entre 10% et 20%.

	Temps de calcul/point
PF	0.1 ms
WPF	0.1 ms
MLS	0.2 ms
Area_Tri	0.6 ms
PF+Bilateral	0.1 ms
PF+IRLS	0.2 ms
FMLS	0.2 ms

TABLE 3.1 – Tableau résumant les temps de calcul de différentes méthodes d'estimation de normales.

3.4 Débruitage non local : méthode NLD3D

Les données laser contiennent du bruit avec un écart-type plus ou moins grand suivant le type de technologie d'acquisition. Un traitement important consiste donc à réduire ce bruit présent dans les données. Le débruitage fait intrinsèquement partie des méthodes de reconstruction par approximation ce qui n'est pas le cas des méthodes de reconstruction par interpolation. Le maillage construit par interpolation va conserver le bruit du nuage en entrée. Le débruitage du nuage de points est ainsi une étape essentielle avant d'effectuer une reconstruction de la surface par interpolation. De nombreuses méthodes de débruitage des nuages de points existent dans la littérature, mais peu d'entre elles parviennent à préserver les détails de la surface tout en enlevant correctement le bruit. Ce sont surtout les points de discontinuité qui posent problème lors d'un débruitage. Or pour construire un modèle réaliste de l'environnement à partir de données de LiDAR, il est important de conserver et de récupérer les arêtes même en présence de bruit. Notre objectif est ainsi d'être capable de débruiter un nuage de point de LiDAR (peu d'« *outliers* », bruit faible) tout en conservant le maximum de détail. Nous présentons ici une méthode innovante pour le débruitage. Elle a été adaptée à partir d'une méthode pour le débruitage d'images appelée « Non Local Denoising » (NLD) de [Buades et Morel, 2005], elle même basée sur la redondance dans les données.

3.4.1 Etat de l'art

Comme décrit dans [Wang et al., 2008] et [Schall et al., 2008], nous pouvons classer les méthodes de débruitage en plusieurs catégories : les techniques spectrales, les filtres par voisinage, les approches par approximation locale et les techniques basées sur les statistiques.

Pour les techniques spectrales, [Taubin, 1995] réduit le problème de débruitage à un filtre passe-bas en généralisant l'analyse de Fourier aux surfaces discrètes. [Pauly et Gross, 2001] introduisent le concept de fréquence locale pour la géométrie et décomposent ces fréquences par transformée de Fourier. La modification des coefficients de la transformée de Fourier permet de lisser les données. [Alexa, 2002] utilise le filtre de Wiener et l'adapte à la géométrie des maillages pour le débruitage.

Parmi les méthodes basées sur le voisinage, certaines sont des extensions de filtres 2D à la 3D comme [Yagou et al., 2002] qui étendent aux maillages le filtre moyen et médian du traitement de l'image. Le filtre bilatéral défini dans [Tomasi et Manduchi, 1998] a aussi été adapté aux maillages par [Fleishman et al., 2003]. [Choudhury et Tumblin, 2003] introduisent le concept de filtrage trilatéral d'images et l'adaptent pour les maillages. [Yoshizawa et al., 2006] présentent une extension aux maillages du filtre NLD introduit par [Buades et Morel, 2005] pour les images : ils calculent en chaque point p_i une approximation locale par une « *Radial Basis Function* » (RBF). Un poids de similarité est défini entre p_i et un point p_j de son voisinage en comparant la fonction radiale autour de p_i et de p_j. Le point p_i est ensuite débruité en utilisant le filtre bilatéral. [Wang et al., 2008] ont aussi adapté le filtre non local aux nuages de points. En utilisant le filtre trilatéral, ceux-ci calculent une intensité géométrique en chaque point. Ensuite, en calculant le plan local

p_i d'un point et en projetant les points du voisinage dans le plan local, ils débruitent le point p_i en comparant la similarité d'intensité géométrique entre deux points voisins. [Schall et al., 2008] et [Huhle et al., 2008] appliquent aussi le filtre non local pour débruiter des données de LiDAR mais ils l'appliquent sur des données d'images de profondeur ce qui leur permet d'utiliser directement la méthode de [Buades et Morel, 2005]. De manière générale, les méthodes basées sur le filtre bilatéral ou trilatéral et leurs dérivées débruitent les points suivant la normale aux points. Ils ont donc besoin d'une information précise sur la normale. Ces méthodes peuvent aussi introduire un effet de réduction du volume de la surface car les points de courbure maximale vont utiliser les points voisins qui ont une courbure inférieure.

Pour les méthodes par approximation locale, il s'agit de déterminer tout d'abord une approximation locale de la surface au point p_i. Nous connaissons des méthodes d'approximation locale de surfaces comme le « *Moving Least Squares* » (MLS) introduit par [Levin, 1998] ou le « *Radial Basis Function* » (RBF). [Mederos et al., 2003] introduisent un nouvel opérateur de débruitage inspiré des MLS et des théories de statistiques robustes. Comme [Fleishman et al., 2005], ils détectent dans chaque voisinage local les « *inliers* » et les « *ouliers* » et estiment de manière robuste les paramètres des surfaces locales pour projeter les points sur ces surfaces. [Oztireli et al., 2009] utilisent une version implicite des surfaces MLS et ajoutent un noyau basé sur des statistiques robustes pour former le « *Robust Implicit Moving Least Squares* » (RIMLS).

Pour les méthodes basées uniquement sur les statistiques, [Pauly et al., 2004] présentent une représentation statistique qui quantifie la probabilité pour chaque point d'appartenir à la surface débruitée. [Jenke et al., 2006] utilisent les statistiques bayésiennes pour inférer une probabilité maximale de reconstruction. Mais les techniques basées sur des statistiques bayésiennes nécessitent généralement d'optimiser un modèle statistique global. Il est dans ce cas difficile de traiter rapidement des nuages de points de plusieurs millions de points.

3.4.2 Méthode de débruitage proposée : NLD3D

Notre méthode de débruitage est une méthode proche de [Yoshizawa et al., 2006] et de [Wang et al., 2008]. Comme ces deux méthodes, nous nous sommes inspirés de l'algorithme de traitement d'image « *Non Local Denoising* » de [Buades et Morel, 2005]. Mais nous définissons une mesure de similarité géométrique différente de [Yoshizawa et al., 2006] et de [Wang et al., 2008]. Elle est basée sur les approximations locales MLS. Nous nous distinguons aussi de ces travaux dans la mesure où nous n'effectuons pas de filtrage bilatéral avec les poids de similarité géométrique. Notre méthode n'est pas basée sur le voisinage puisque nous appliquons un filtrage qui peut s'effectuer entre deux points quelconques du nuage de points. Nous pouvons ainsi trouver plus de points similaires. Nous avons donc un voisinage local (pour l'approximation locale) et un voisinage de débruitage (généralement plus grand).

Nous allons décrire ici en détail la méthode de débruitage de [Buades et Morel, 2005] fonctionnant sur les images 2D.

« *Non Local Denoising* » (NLD) pour les images

Le « *Non Local Denoising* » est un algorithme de débruitage pour les images introduit par [Buades et Morel, 2005]. L'objectif est de débruiter les pixels en utilisant d'autres pixels dont les voisinages sont similaires. Une image est un ensemble de pixels v_i et pour chaque pixel v_i, nous avons un voisinage local V_i. Nous comparons les voisinages locaux des pixels dans une fenêtre de débruitage W_i autour du pixel v_i. La nouvelle valeur du pixel v_i est $v'_i = \sum_{j \in W_i} w_{i,j} v_j$. Le poids $w_{i,j}$ est une mesure de similarité entre le pixel i et le pixel j avec $0 \leq w_{i,j} \leq 1$ et $\sum_{j \in W_i} w_{i,j} = 1$. La similarité entre les pixels i et j va dépendre de la similarité $D(i,j)$ des intensités de gris des pixels voisins de i et de j. Nous avons donc :

$$D(i,j) = \sum_{i+k \in N_i} e^{-\frac{\|k\|^2}{h^2}} \mid v_{j+k} - v_{i+k} \mid^2 \tag{3.3}$$

$$w_{i,j} = \frac{1}{\sum_{j \in W_i} e^{-\frac{D(i,j)}{a^2}}} e^{-\frac{D(i,j)}{a^2}} \tag{3.4}$$

où N_i est le voisinage local de v_i dans le but de faire une comparaison de similarité avec v_j. Ce voisinage doit être suffisamment large pour être robuste au bruit et suffisamment petit pour être représentatif des détails de l'image au pixel v_i. h est le paramètre du noyau gaussien autour du pixel v_i, c'est-à-dire représente l'influence locale des points autour de v_i. a est un paramètre de contrôle de l'influence de la similarité. Pour éviter les recherches de similarité avec tous les autres pixels de l'image, on se restreint à une fenêtre large W_i. W_i doit être suffisamment large pour permettre de trouver d'autres pixels similaires mais pas trop large pour limiter le temps de calcul.

Nous voyons sur la Figure 3.27 que pour débruiter un pixel p, on compare les voisinages des autres pixels de l'image avec le voisinage de p. Nous voyons ici que les poids de similarité $w(p, q1)$ entre p et $q1$ et les poids $w(p, q2)$ entre p et $q2$ seront proches de 1 alors que le poids $w(p, q3)$ entre p et $q3$ sera plutôt proche de 0.

De la 2D à la 3D

[Buades et Morel, 2005] définissent la notion de similarité entre pixels v_i et v_j en comparant les valeurs en niveaux de gris des pixels des voisinages V_i et V_j. Or cette notion n'est pas extensible en 3D. Nous avons donc défini une notion de similarité de voisinage dans les nuages de points. Contrairement à [Yoshizawa et al., 2006] et [Wang et al., 2008] qui comparent deux points uniquement lorsqu'ils sont voisins, nous voulons pouvoir comparer la similarité de tous les couples de points dans le nuage de points pour avoir plus de possibilités de trouver des points similaires. Et contrairement à [Wang et al., 2008] qui décrivent la géométrie locale d'un point avec une seule valeur d'intensité, nous voulons pouvoir décrire la géométrie locale d'un point de manière plus précise pour donner un descripteur de géométrie locale. Ainsi, les points qui ont la même similarité géométrique pourront être utilisés pour réduire le bruit tout en préservant les détails de la surface. Nous avons donc besoin que ces descripteurs soient robustes au bruit et représentent bien la géométrie locale. Notre descripteur local doit être également invariant par rotation pour permettre de trouver plus de points similaires. Par exemple, si nous voulons débruiter les points d'une

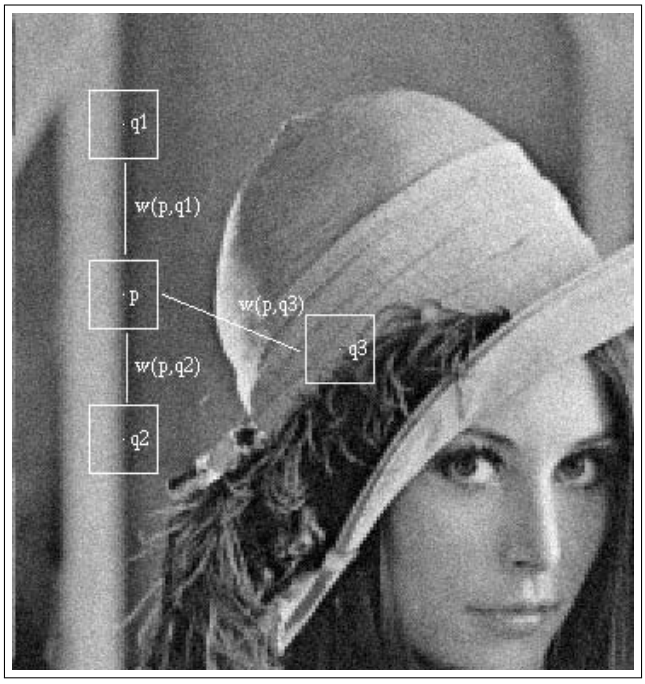


FIGURE 3.27 – Stratégie du NLD (source [Buades et Morel, 2005]).

sphère, nous aurons besoin d'un descripteur invariant par rotation pour trouver que tous les points ont des similarités géométriques locales et ainsi réduire le bruit du nuage de points.

Aperçu de l'algorithme

Notre méthode s'effectue en deux étapes. Lors de la première étape, nous définissons en chaque point p_i un repère local et dans ce repère local, nous calculons un polynôme à deux variables g_i qui correspond à notre MLS. Nous en déduisons un descripteur géométrique local du point p_i. Lors de la deuxième étape, nous modifions le point p_i en fonction des points p_j et des poids $w_{i,j}$ dans une large fenêtre W_i. Les poids $w_{i,j}$ entre deux points représentent une mesure de la différence entre les deux descripteurs locaux de p_i et de p_j.

Descripteur local

Comme nous l'avons vu dans la section sur le calcul de normales, pour déterminer la surface MLS en un point p_i, nous calculons en chaque point p_i un plan local minimisant la distance pondérée des points du voisinage au plan local. Puis nous définissons un repère local composé des vecteurs propres de la matrice de covariance et dont le centre est le point p_i sur le plan local. Nous calculons en p_i la surface MLS $g_i = \sum_{k+l \leq d} a^i_{k,l} u^k v^l$ polynomiale à deux variables approximant la surface réelle en minimisant la somme :

$$\sum_{j \in V_i} (g_i(u_j, v_j) - f_j)^2 e^{-\frac{\|p_j - p_i\|^2}{h^2}} \tag{3.5}$$

où (u_j, v_j, f_j) sont les coordonnées locales du point p_j dans le repère (v_i^2, v_i^3, v_i^1), f_j correspondant à la distance signée du point p_j au plan local.

Une fois les surfaces MLS calculées, nous définissons notre descripteur local comme les coefficients du polynôme g_i, c'est-à-dire 6 coordonnées pour une surface polynomiale de degré 2, 10 coordonnées pour une surface polynomiale de degré 3...

Invariance du descripteur local

Dans une première version de notre algorithme de débruitage, nous avions choisi les vecteurs propres de la matrice de covariance comme repère pour la surface MLS afin que le descripteur local soit invariant par rotation. Or ces vecteurs dépendent de la distribution locale des points alors que nous souhaiterions être invariant par rotation suivant la géométrie locale autour du point p_i. Nous avons donc modifié notre méthode pour tenir compte des directions principales autour du point p_i.

Dans une première étape, nous calculons une première surface MLS comme nous l'avons décrit précédemment, avec un plan local de référence suivant les vecteurs propres de la matrice de covariance, c'est-à-dire suivant la répartition des points autour de p_i. A partir de cette surface polynomiale, nous pouvons calculer les directions principales de la surface au point p_i, c'est-à-dire au point $(0, 0)$ dans le repère local. Les directions principales correspondent aux vecteurs propres de l'endomorphisme symétrique du plan tangent à la surface paramétrée $X(u, v) = (u, v, g_i(u, v))$ (appelé endomorphisme de Weingarten). Nous choisissons de garder la direction principale d_i maximisant la courbure locale au point p_i.

La figure 3.28 montre pour le nuage *S_Fandisk* à gauche le vecteur propre v_i^2 de la matrice de covariance dans le repère local et à droite la direction principale d_i selon la courbure maximale. Nous remarquons notamment que pour les points entourés en rouge, la direction principale permet de mieux estimer une direction de la géométrie locale alors que le vecteur v_i^2 va dépendre de la répartition des points. Lorsque nous ajoutons du bruit au nuage de points, nous voulons que ces directions restent stables. Nous voyons sur la figure 3.29 les vecteurs v_i^2 et la direction principale d_i en ajoutant un bruit correspondant à 30% de la distance moyenne entre points. Nous pouvons voir que la direction principale d_i est plus robuste au bruit que le vecteur v_i^2.

A partir de ce nouveau vecteur d_i, on peut définir un nouveau repère local $(d_i, n_i \wedge d_i, n_i)$ où $n_i = v_i^1$ représente la normale au point p_i. Ce nouveau repère nous permet de calculer une nouvelle surface MLS et donc de nouveaux coefficients pour g_i. Nous définissons le descripteur local $G_i = (a^i_{0,0}, a^i_{0,1}, a^i_{1,0}, a^i_{0,2}...)$ à partir des coefficients de la surface locale. Ce changement de repère permet de construire un descripteur local invariant par rotation.

Débruitage

Afin de débruiter un point p_i, nous le comparons avec tous les points qui sont dans le voisinage de débruitage W_i en définissant un poids de similarité entre les points p_i et p_j :

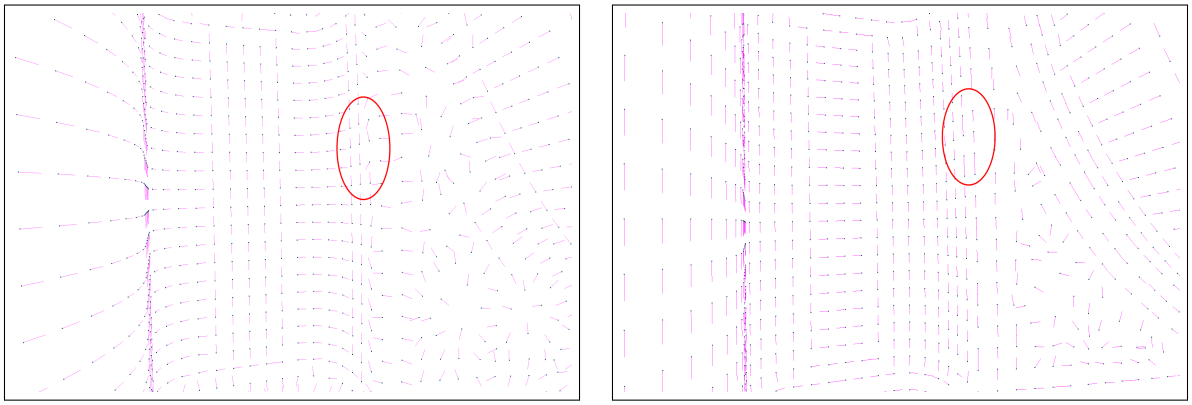


FIGURE 3.28 – A gauche, vecteur propre v_i^2 du repère local. A droite, direction principale d_i selon la courbure maximale.

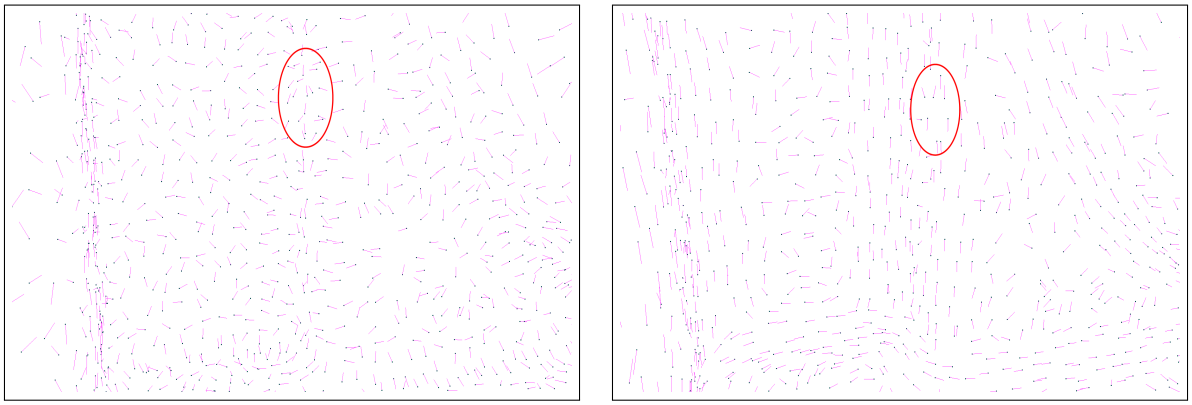


FIGURE 3.29 – A gauche, vecteur propre du repère local. A droite, direction principale selon la courbure maximale (avec un bruit de 30% de la distance moyenne entre points).

$$D(i,j) = \|G_i - G_j\|^2 = \sum_{k+l \leq d} (a^i_{k,l} - a^j_{k,l})^2 \tag{3.6}$$

$$w_{i,j} = \frac{1}{\sum_{j \in W_i} e^{-\frac{D(i,j)}{a^2}}} e^{-\frac{D(i,j)}{a^2}} \tag{3.7}$$

où $D(i,j)$ est la mesure de similarité entre les surfaces autour du point i et du point j. a est le paramètre de contrôle de similarité.

Les nouvelles coordonnées du point p_i, notées p'_i sont calculées de la façon suivante :

$$p'_i = p_i + (a^i_{0,0} - \sum_{j \in W_i} w_{i,j} a^j_{0,0}) n_i \tag{3.8}$$

où $a^i_{0,0} = g_i(0,0)$ est le premier coefficient de g_i, c'est-à-dire la distance de la surface MLS au point i.

Dans le cas où un point p_i ne trouve pas de points similaires, c'est-à-dire si $w_{i,j} \approx 0$ pour tout $j \neq i$, alors le point p_i ne sera pas modifié. Les points atypiques des nuages ne seront donc pas modifiés, ce qui permet de conserver le maximum de détails mais aussi les « *outliers* » et les points très bruités. Nous notons avec cette remarque que notre méthode ne fonctionnera pas pour les nuages de points très bruités ainsi que pour ceux présentant peu d'auto-similarité. Par contre, dans le cas de nuages peu bruité et présentant de nombreuses similarités géométriques locales, elle permettra de débruiter les points tout en conservant les détails de la surface.

Nous avons appelé notre algorithme de débruitage NLD3D pour « *Non Local Denoising 3D* ». Cet algorithme de débruitage a été publié à la conférence [$PCV, 2010$] sans l'amélioration de l'invariance par rotation du descripteur local suivant les directions principales. La version complète de la méthode NLD3D, décrite dans ce mémoire, est en cours de soumission à une revue [$IJPRS$].

3.4.3 Analyse et comparaison des résultats sur des données simulées et réelles

Notre méthode NLD3D comporte quatre paramètres : la taille du voisinage local V_i (sphère de rayon δ_{NLD3D}), la taille du voisinage de débruitage W_i (sphère de rayon Δ_{NLD3D}), le paramètre h du noyau gaussien et le paramètre a du noyau de similarité.

Nous avons pris un voisinage local fixe où $\delta_{NLD3D} = 4d_{mean}$. Pour le voisinage de débruitage W_i, nous avons choisi $\Delta_{NLD3D} = 3\delta_{NLD3D}$ ce qui permet d'inclure suffisamment de points afin de trouver des voisins localement similaires tout en étant suffisamment petit pour limiter le temps de calcul. Nous avons pris $h = \frac{\delta_{NLD3D}}{2}$ comme dans le cas du calcul des normales. Le paramètre a est le paramètre le plus difficile à fixer car il dépend du nuage de points. Lors des expérimentations, nous avons trouvé des résultats optimaux pour des valeurs de a entre 0.1 et 10. Pour le polynôme g_i, nous avons choisi des polynômes de degré 2 pour être suffisamment robuste au bruit. Toutes les méthodes ont été exécutées sur un QuadCore Q9300 2.50 GHz, 2 Go de RAM, sur un seul *thread* et sur CPU.

Résultats sur des données simulées

La méthode de débruitage NLD3D que nous proposons a été testée sur des données réelles et des données simulées. Nous avons utilisé comme nuage de points simulés *S_Angle*, *S_Anneau* et *S_Fandisk* et nous avons ajouté un bruit gaussien avec un écart-type égal à 15% de la distance moyenne entre points, ceci afin de représenter le niveau de bruit mesuré dans des données réelles. Nous avons comparé nos résultats avec deux autres méthodes classiques de débruitage : une méthode basée sur le voisinage (le filtre bilatéral de [Fleishman et al., 2003]) et une méthode basée sur un opérateur de projection et d'approximation locale (le RIMLS de [Oztireli et al., 2009]). Dans toutes ces méthodes, les données d'entrée sont des nuages de points bruités et les données de sortie sont des nuages de points débruités. Pour faciliter la comparaison visuelle, nous avons ajouté une étape de triangulation après l'étape de débruitage. Ce maillage a été obtenu avec notre algorithme

de triangulation décrit dans le chapitre 4.

Les résultats des méthodes de débruitage sur le nuage *S_Angle* sont visibles sur la figure 3.30. Notre algorithme NLD3D est capable de conserver et de débruiter l'angle grâce aux similarités de géométrie locale en améliorant les points sur l'arête.

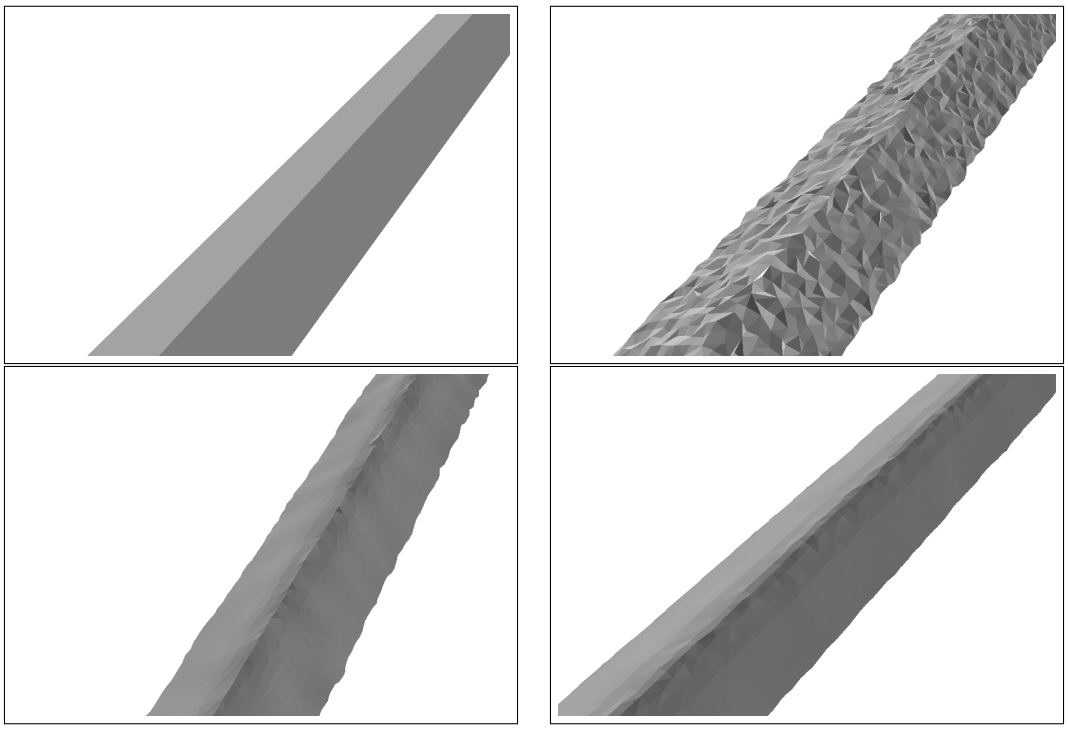


FIGURE 3.30 – Résultats des algorithmes de débruitage sur un nuage de points composé de deux plans orthogonaux (en haut à gauche : original, en haut à droite : bruité, en bas à gauche : débruité par RIMLS, en bas à droite : débruité par NLD3D)

Nous avons testé les méthodes de débruitage sur le nuage *S_Anneau*. La Figure 3.31 en montre les résultats et nous pouvons faire la comparaison des méthodes. Nous voyons que notre algorithme a un bon comportement sur les bords de l'anneau car chaque point du bord peut trouver de nombreux autres points similaires. Mais nous avons de mauvais résultats de débruitage au moment de la petite variation du bord. Cela s'explique par le fait qu'il n'y a pas d'autres points avec la même géométrie locale dans le voisinage et les points ne peuvent pas être débruités. Il s'agit d'une limitation de notre algorithme. En effet, tous les points singuliers qui ne trouvent pas de points correspondants dans le nuage de points ne pourront pas être débruités.

Pour faire une comparaison quantitative, nous avons utilisé la distance de Hausdorff entre surfaces : $d(X,Y) = max\{\underset{y\in Y}{sup}\ \underset{x\in X}{inf}\ \delta(x,y), \underset{x\in X}{sup}\ \underset{y\in Y}{inf}\ \delta(x,y)\}$. Nous calculons la distance de Hausdorff entre les surfaces débruitées triangulées et la surface originale triangulée. La distance de Hausdorff est la distance la plus utilisée pour mesurer la distance entre surfaces.

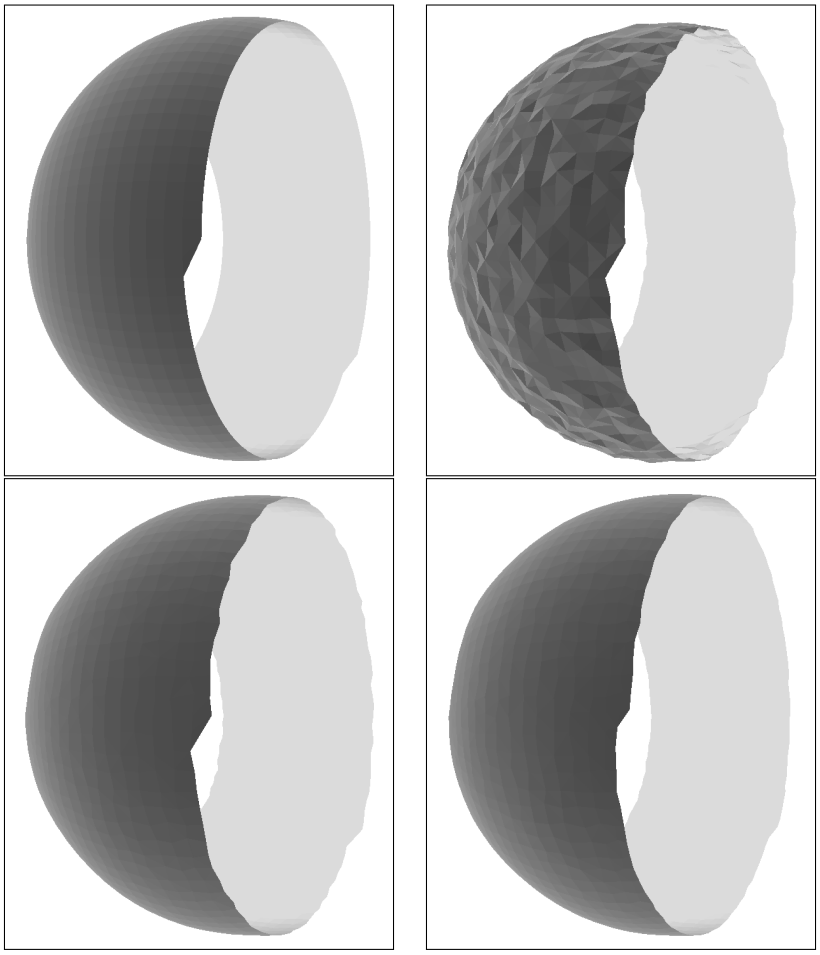


FIGURE 3.31 – Résultats du débruitage d'un nuage de points formant un anneau *S_Anneau* (en haut à gauche : original, en haut à droite : bruité, en bas à gauche : débruité par le filtre bilatéral, en bas à droite : débruité par NLD3D)

Les résultats obtenus sont reportés dans le tableau 3.2. Nous observons que la surface débruitée avec NLD3D est plus proche de la surface originale que les surfaces débruitées avec le filtre bilatéral ou avec le RIMLS. On note aussi que les temps de calcul sont très similaires pour toutes ces méthodes bien qu'il faille aussi rappeler la forte dépendance des paramètres dans les temps de calcul. Ainsi, pour NLD3D, si nous choisissons une fenêtre de comparaison W_i très grande alors notre méthode va devenir beaucoup plus lente que les autres algorithmes.

Nous avons aussi testé notre méthode sur le nuage de points *S_Fandisk*. Afin de visualiser la notion de similarité de voisinage entre points. Nous avons choisi arbitrairement deux points encerclés en vert dans la figure 3.32 et avons visualisé la similarité de leurs points voisins. Les points voisins en couleur sont ceux dont le poids de similarité $w_{i,j} \geq 0.5$. Le point en vert encerclé sur une arête n'a sélectionné des voisins similaires que sur l'arête, ce qui montre que la géométrie locale a été correctement comparée. Les résultats de débruitage sont présentés sur la figure 3.33.

	S_Angle	*S_Angle*	*S_Anneau*	*S_Anneau*
Surface	Distance de Hausdorff	Temps	Distance de Hausdorff	Temps
Bruitée	0.159103 m	-	0.113102 m	-
Bilateral	0.126050 m	1.5 s	0.081701 m	1.4 s
RIMLS	0.095922 m	1.7 s	0.069357 m	1.8 s
NLD3D	0.073229 m	1.2 s	0.06316 m	1.1 s

TABLE 3.2 – Comparaison des méthodes de débruitage sur le nuage *S_Angle* et le nuage *S_Anneau* à l'aide de la distance de Haussdorff entre surface originale et surface débruitée.

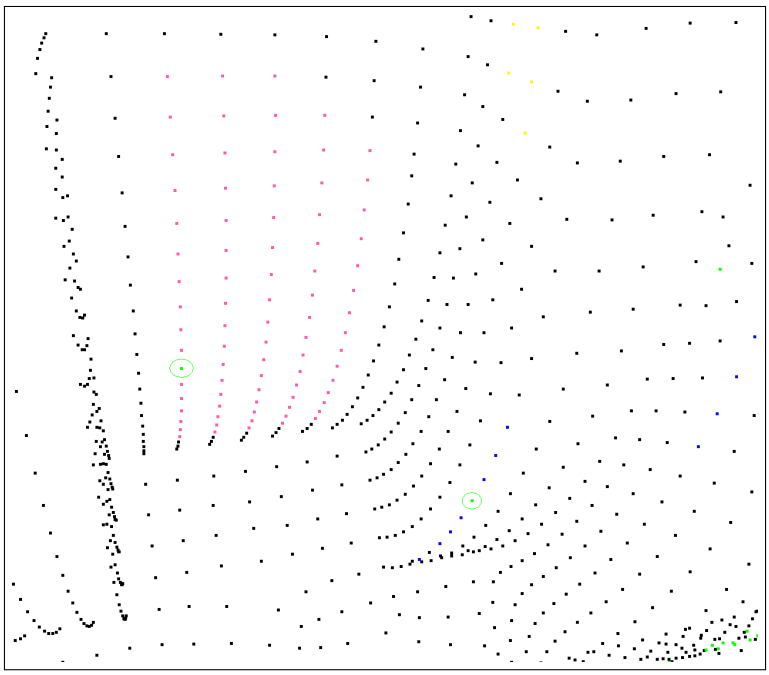


FIGURE 3.32 – Exemples de poids $w_{i,j}$ pour le nuage *S_Fandisk*.

Résultats sur des données réelles

Outre les tests sur des données simulées, nous avons testé notre algorithme sur des données réelles provenant de scanners fixes et de scanners mobiles. Les résultats de ces tests sont visibles sur les figures 3.34, 3.35 et 3.36. Pour ces trois cas, nous constatons visuellement que le bruit a largement diminué tout en conservant les détails de la surface. La qualité du débruitage est visible dans les zones de fortes similarités, comme les zones planes, les arêtes, c'est-à-dire particulièrement pour les nuages de points de scène urbaine *LARA3D_Orsay* et *FX_GX_Orsay*. Nous observons d'ailleurs sur la figure 3.35 une nette amélioration de l'aspect de la surface avec notre méthode de débruitage par rapport à la méthode RIMLS.

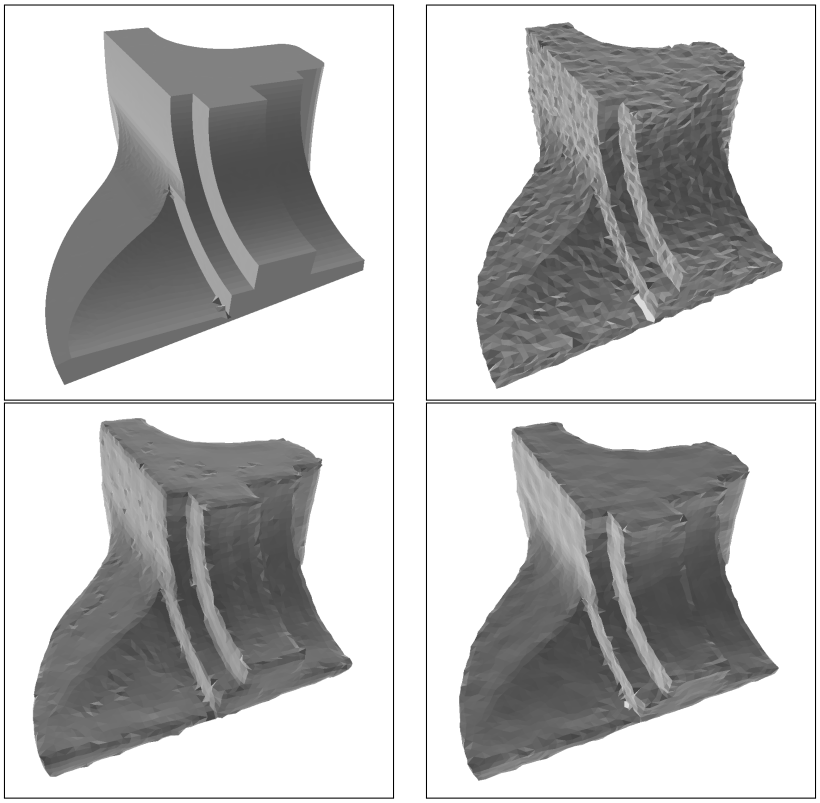


FIGURE 3.33 – Résultats du débruitage du nuage de points *S_Fandisk* (en haut à gauche : original, en haut à droite : bruité, en bas à gauche : débruité par le RIMLS et en bas à droite : débruité par NLD3D).

3.4.4 Conclusion sur le débruitage non local

Nous avons proposé une méthode innovante pour débruiter un nuage de points 3D qui est une extension d'un filtre 2D appelé « *Non Local Denoising* ». Des descripteurs locaux de surface invariants par rotation ont été définis ainsi qu'une formule permettant de trouver les nouvelles coordonnées d'un point en utilisant tous les points du voisinage de débruitage et si possible tous les points du nuage. Notre méthode a été testée sur des données simulées et réelles et les résultats comparés à deux autres méthodes classiques de débruitage. Nous avons obtenu des performances similaires aux méthodes existantes pour des nuages de points quelconques et de meilleurs résultats sur des données présentant de fortes zones de redondance et d'auto-similarité, notamment les données de scènes urbaines. En revanche, il apparaît que pour des nuages de points très bruités, notre méthode n'est pas adaptée car notre descripteur local n'est alors plus invariant par rotation. Notre méthode de débruitage est ainsi bien adaptée aux nuages de points LiDAR peu bruités, denses et qui contiennent de nombreuses zones d'auto-similarité comme c'est le cas pour les nuages de points de la plateforme mobile LARA3D.

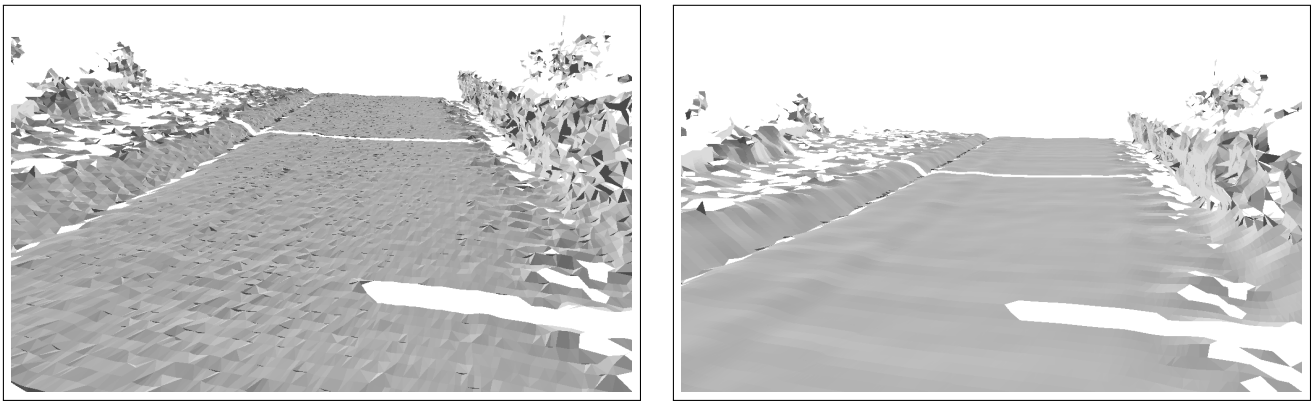


FIGURE 3.34 – Résultats du débruitage du nuage de points *LARA3D_Etables* provenant d'un système de cartographie mobile (à gauche : maillage sans débruitage, à droite : débruité par NLD3D).

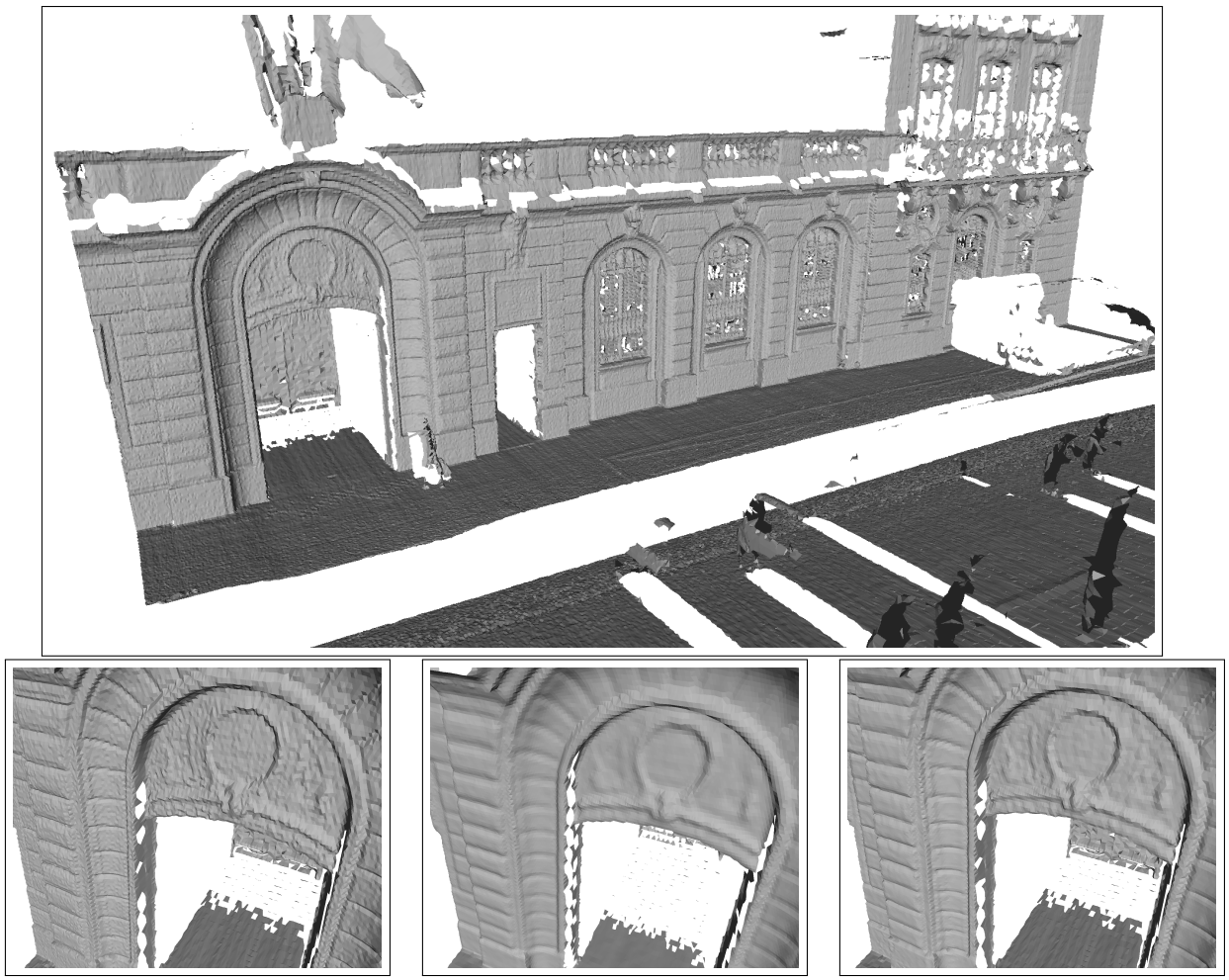


FIGURE 3.35 – En haut, maillage du nuage *LARA3D_Orsay* sans débruitage. En bas, détails des maillages (à gauche : sans débruitage, au milieu : débruité par RIMLS, à droite : débruité par notre méthode NLD3D).

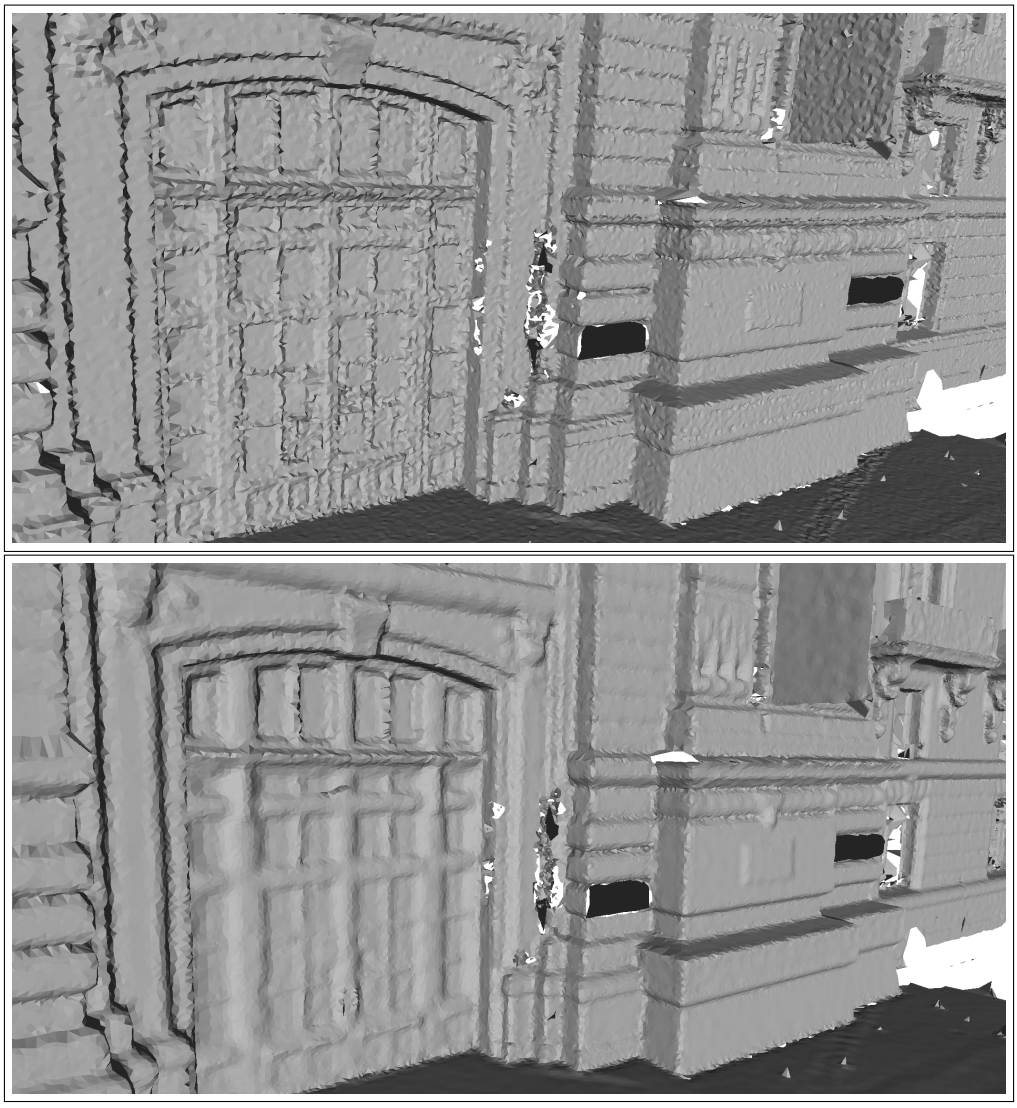


FIGURE 3.36 – Résultats du débruitage du nuage de points *FX_GX_Orsay* provenant d'un système fixe de numérisation (en haut : sans débruitage, en bas : débruité par NLD3D).

Chapitre 4

Modélisation 3D d'environnements

Résumé

Nous décrivons dans ce chapitre quelques éléments de notre méthode générale de modélisation d'environnement. Après les pré-traitements que nous avons vus au chapitre précédent (décimation, calcul des normales et débruitage), nous présentons l'étape de segmentation et de triangulation. L'étape de segmentation a pour but de détecter les zones planes de l'environnement. Notre méthode de triangulation construit un maillage par interpolation à partir d'un nuage de points débruité.

4.1 Détection de plans dans les nuages de points

La détection des zones planes est une étape importante permettant de segmenter l'environnement en deux : les zones planes et les zones non planes. Les zones planes qui vont être détectées peuvent être celles de l'environnement (qui peuvent ne plus être planes dans le nuage de point à cause d'erreurs introduites lors de l'acquisition) mais aussi des zones non planes de l'environnement qui seront modélisées en zone plane afin de rendre le modèle plus compact (minimiser le nombre de triangles).

4.1.1 Etat de l'art

Il existe de nombreuses méthodes de segmentation pour les images, les nuages de points et les maillages. Nous pouvons citer quelques exemples de méthode comme la croissance de région [Poppinga et al., 2008], les estimateurs robustes comme RANSAC (« *RANdom SAmple Consensus* ») de [Fischler et Bolles, 1981], les méthodes basées sur la sphère de Gauss [Liu et Xiong, 2008] ou la transformée de Hough [Tarsha-Kurdi et al., 2007], la morphologie mathématique [Gorte et Pfeifer, 2004], la segmentation hiérarchique [Deveau, 2006], les détections de contours par gradient ou laplacien [Gorte, 2007], les champs de Markov [Munoz et al., 2008], etc.

Dans le domaine plus spécifique de la détection de plans dans les données 3D de scènes urbaines, nous pouvons décrire quelques méthodes existantes. [Bauer et al., 2003] et [Boulaassal et al., 2007] détectent les façades dans des nuages de points denses avec un algorithme basé sur RANSAC. [Vosselman et al., 2004] étudient et comparent les méthodes de croissance de région et la transformée de Hough pour détecter les primitives géométriques comme les plans, les cylindres et les sphères. [Tarsha-Kurdi et al., 2007] détectent les toits des bâtiments dans les nuages de points en comparant les résultats obtenus avec la transformée de Hough et l'algorithme RANSAC. Ils concluent sur le fait que ce dernier est plus efficace pour les traitements de données 3D que Hough. [Pu et Vosselman, 2006] utilisent la méthode de [Vosselman et al., 2004] pour détecter et extraire les zones planes des nuages de points de scènes urbaines et pour ensuite reconnaître sur les zones extraites des éléments particuliers (toit, porte, fenêtre, etc.). [Stamos et al., 2006] et [Chen et Stamos, 2007] présentent aussi des algorithmes de détection de régions planaires dans des scènes urbaines mais ici pour des données qui sont des images de profondeur. Les méthodes de [Stamos et al., 2006] et de [Chen et Stamos, 2007] sont basées sur un algorithme de croissance de région dans chaque image de profondeur et sur la fusion des résultats pour la segmentation du nuage de points correspondant.

Ainsi, la majorité des méthodes pour la détection de plans dans les nuages de points denses sont basées sur la croissance de région et RANSAC. C'est pourquoi nous présentons ici une analyse de ces deux méthodes très étudiées dans la littérature. Pour comparer la complexité en temps de calcul des algorithmes, nous prenons comme exemple un cas simple de nuage de points contenant N points avec un seul plan P qui compte n points. Nous supposons que nous voulons détecter ce plan P et définissons n_{min} comme la taille minimale (nombre de points) des plans que nous voulons détecter.

RANSAC

RANSAC est un algorithme initialement développé par Fischler et Bolles et qui permet d'ajuster un modèle à un ensemble de données discrètes sans pour autant devoir tester toutes les possibilités. RANSAC est basé sur la probabilité de détecter un modèle à partir d'un ensemble minimal d'échantillons suffisants pour définir ce modèle.

Pour détecter un plan avec RANSAC, nous choisissons trois points aléatoires (ensemble minimal et suffisant pour définir un plan). Nous calculons les paramètres du plan avec ces trois points. Ensuite un score est défini pour déterminer la bonne correspondance du modèle avec les données. Usuellement, le score d'un plan correspond au nombre de points appartenant à ce plan. On dit qu'un point appartient à un plan si la distance du point au plan est inférieure à une distance γ. Nous verrons plus tard que nous remettrons en cause cette définition d'*appartenance*. A la fin, nous conservons le modèle de plan avec le meilleur score, c'est-à-dire contenant le plus de points. En choisissant 3 points au hasard, la probabilité que le plan formé soit le plan recherché est $p = (\frac{n}{N})^3$ en supposant que la sélection de n'importe quels 3 points parmi les n points du plan est suffisante pour déterminer correctement les paramètres du plan (ce qui n'est pas le cas pour des nuages de points bruités). La probabilité d'obtenir le plan recherché en T essais est $p = 1 - (1 - (\frac{n}{N})^3)^T$. En supposant $\frac{n_{min}}{N} \ll 1$, nous savons que le nombre T_{min} d'essais minimums pour avoir une probabilité p_t d'obtenir les plans d'une taille supérieure à n_{min} est donné par l'équation 4.1.

$$T_{min} = \frac{log(1-p_t)}{log(1-(\frac{n_{min}}{N})^3)} \approx log(\frac{1}{1-p_t})(\frac{N}{n_{min}})^3. \tag{4.1}$$

A chaque essai, nous testons tous les points du nuage pour calculer le score du plan, c'est-à-dire le nombre de points appartenant au plan. La complexité de l'algorithme RANSAC est en $O(N(\frac{N}{n_{min}})^3)$ quand $\frac{n_{min}}{N} \ll 1$ et $O(N)$ quand $n_{min} \to N$. Ainsi, RANSAC est très efficace pour détecter les grands plans, c'est-à-dire quand $\frac{n_{min}}{N} \simeq 1$ mais est très lent pour détecter les petits plans dans les grands nuages de points, c'est-à-dire quand $\frac{n_{min}}{N} \ll 1$. Un des avantages de RANSAC est d'être robuste au bruit car le modèle est calculé avec seulement 3 points, le plan peut donc contenir de nombreux points aberrants.

[Schnabel et al., 2007] apportent deux optimisations à RANSAC, la première portant sur la sélection locale des points et la seconde sur un calcul optimisé du score. Pour cela et avant tout autre chose, un octree est créé à partir du nuage de points. Les points pour l'estimation des paramètres du plan ne sont donc plus choisis aléatoirement dans tout le nuage de points mais seulement dans un noeud de l'octree à une profondeur aléatoire. Pour le calcul de score, au lieu de tester tous les points du nuage, on teste le modèle uniquement sur un sous-échantillonage du nuage de points et le score global est estimé par interpolation. La complexité de l'algorithme est en $O(\frac{N}{r}\frac{4Nd}{n_{min}})$ où r est le nombre de sous échantillons du nuage de points pour le calcul du score et d est la profondeur maximale de l'octree. Leur algorithme améliore la vitesse de détection des plans mais la complexité reste élevée pour détecter de petits plans dans de grands nuages de points.

Croissance de régions

Les algorithmes de croissance de région fonctionnent bien sur les images de profondeur comme dans [Poppinga et al., 2008]. Le principe de la croissance de région est de partir d'une région graine et de la faire croître par voisinage lorsque les voisins satisfont certaines conditions. Cependant, si dans les images de profondeurs, il est possible de connaître directement les voisins de chaque point avec les coordonnées du pixel. Ce n'est pas le cas pour un nuage de points 3D qui ne comporte pas directement d'information de voisinage. Les deux méthodes les plus utilisées sont de prendre soit les k points les plus proches, soit les points dont la distance au point d'origine est inférieure à un paramètre δ comme nous l'avons vu au début du chapitre 3. Pour un nuage de densité fixe, la recherche du voisinage peut être considérée comme étant en temps constant $O(C)$. Pour un plan de taille n, à partir d'une région graine correcte, le temps de détection du plan sera de $O(Cn)$. Si nous nous plaçons dans le cas extrême où $n_{min} \geq N$, c'est-à-dire que la taille minimale recherchée est plus grande que la taille du nuage de points, alors nous allons tester chaque point du nuage comme point graine et le temps de détection sera en $O(C(N-n)+Cn^2)$. L'avantage de la méthode de croissance de région est qu'elle est rapide lorsqu'il y a de nombreux petits plans à extraire. Mais nous voyons que l'algorithme de croissance tend à être de plus en plus lent lors de la détection de plans de plus en plus grands (par exemple une façade dans un nuage de points) et lorsque nous aurons de nombreuses régions graines incorrectes qui obligent à réitérer une croissance de région. Une autre difficulté de la croissance de région provient de la sélection des régions graines. Si, par exemple, nous sélectionnons comme région graine les points du nuage qui ont un résidu local minimal alors nous devons faire en plus un tri des points du nuage ce qui amène une complexité en $O(NlogN)$.

[Rabbani et al., 2006] ont développé une méthode de segmentation de zones par contrainte lisse avec une croissance de région. Après avoir estimer une normale en chaque point comme dans [Hoppe et al., 1992], les points avec un résidu minimal sont utilisés comme point de départ pour la croissance de région. Ils testent les k plus proches voisins du dernier point ajouté : si l'angle entre la normale du point initial et la normale du point courant est inférieure à un paramètre α alors ils ajoutent ce point au plan.

Nous voyons ainsi que les méthodes RANSAC et de croissance de région sont des méthodes très différentes dans leurs approches mais aussi complémentaires. Pour notre application, nous souhaitons pouvoir détecter de grands plans (grand signifiant par exemple un plan composé de plus d'un cinquième du nuage : les façades des bâtiments, le sol, etc.) comme les petits plans (par exemple les fenêtres, les balcons, etc.) tout en étant suffisamment rapide pour traiter de nombreux points.

4.1.2 Méthode proposée de détection de zones planes

Aperçu

Notre algorithme de détection de zones planes est basé sur la croissance de région avec une sélection des régions graines optimisée et une étape de croissance modifiée pour pouvoir détecter des plans plus rapidement. Notre méthode fonctionne directement sur des nuages

de points avec seulement l'information de normales. Elle ne dépend pas de la configuration du laser (c'est-à-dire pas de segmentation dans les profils laser comme [Früh et al., 2005] ou [Brun, 2007]). La méthode proposée s'applique en deux étapes. Premièrement, en chaque point p_i, nous déterminons si le voisinage local forme un plan et si tel est le cas nous commençons l'étape de croissance en ce point. Lors de la seconde étape de croissance, au lieu de faire croître le plan initial point par point, nous le faisons par listes de points, plus exactement par voxels (cubes partitionnant l'espace).

Détails

Nous verrons ici en détails les différentes étapes de notre méthode de détection de plans.

Choix des plans initiaux

Notre étape de choix de plans initiaux ou plans graines s'opère de la façon suivante : en chaque point p_i du nuage, nous définissons un voisinage local V_i ainsi que décrit au chapitre 2, c'est-à-dire les points p_j dont la distance $\|p_i - p_j\| \leq \delta$. Si tous les points de ce voisinage local suivent certains critères alors on considére ce voisinage comme un plan local qui servira directement de plan initial pour l'étape de croissance. Nous appelons point d'origine p_0 le point ayant servi à calculer le plan initial. Nous ne faisons pas de calcul de plan graine sur tous les points du nuage pour rechercher le meilleur plan graine, ceci afin d'éviter une étape de tri coûteuse en temps de calcul. En effet, si un point et son voisinage local satisfont les critères que nous allons voir ci-dessous alors nous commençons directement l'étape de croissance de région en ce point.

Critères d'appartenance d'un point à un plan

Dans de nombreux algorithmes de détection de plans, un seul ou deux critères sont utilisés parmi les trois suivants pour définir l'appartenance d'un point à un plan défini par un ensemble de points :

1. La distance orthogonale du point au plan doit être inférieure à un paramètre γ.
2. Les angles entre la normale du point et les normales de tous les autres points appartenant au plan doivent être inférieurs à un paramètre α.
3. L'angle entre la normale du point et la normale du plan doit être inférieur à un paramètre α.

A notre connaissance, aucune méthode de détection de plans n'utilise les trois critères réunis pour définir l'appartenance d'un point à un plan. Or nous allons voir des exemples de configuration de points où la seule utilisation d'un ou deux de ces critères pour la définition d'appartenance d'un point n'est pas suffisante pour détecter correctement un plan. Dans tous ces exemples théoriques, nous introduisons des surfaces composées de régions planes et courbes dont nous voulons extraire les zones planes et nous supposons que les normales ont été parfaitement estimées (suivant la définition expliquée au début chapitre 3).

Sur la figure 4.1, nous avons représenté en projection 2D des configurations de points et de plans en définissant l'appartenance d'un point à un plan suivant un seul des trois critères. Dans l'exemple de gauche, on considère seulement le critère de distance d'un point au plan, c'est-à-dire le critère 1. En vert, ce sont l'ensemble des points appartenant à notre plan suivant ce critère et en pointillé noir, c'est le plan des moindres carrés calculé à partir de cet ensemble de points. Nous avons représenté en bleu la vraie surface que nous souhaiterions idéalement détecter. On note clairement que des points proches des intersections appartenant en réalité à des plans orthogonaux sont considérés comme appartenant au plan en noir. Ceci s'explique par le fait que nous tenons compte uniquement de la distance des points au plan et non de la normale des points. Pour l'exemple du milieu, nous avons considéré le critère 2 comme définition d'appartenance et nous voyons que des points appartenant à deux plans distincts de la surface réelle sont détectés comme appartenant au même plan. Enfin, pour l'exemple de droite, avec le critère 3, c'est-à-dire en définissant l'appartenance d'un point à un plan seulement avec l'angle entre la normale de ce point et la normale du plan, nous détectons un seul plan au lieu de 3 plans distincts.

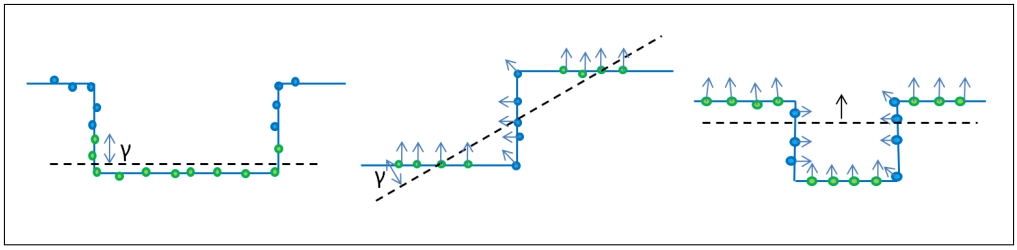


FIGURE 4.1 – Définition d'appartenance de points suivant un seul critère : à gauche les points dont la distance est inférieure à un paramètre γ, au milieu les points dont les normales font un angle inférieur à α et à droite les points dont la normale fait un angle inférieur à α avec la normale au plan (en vert les points détectés appartenant à un seul même plan, en pointillé en noir le plan des moindres carrés, les flèches représentent les normales et en bleu la surface réelle que nous souhaitons détecter).

Si nous choisissons deux des trois critères d'appartenance, nous pouvons aussi arriver à détecter des plans qui ne vont pas correspondre à la surface. L'exemple du milieu de la figure 4.1 est un contre-exemple de la définition d'appartenance pour les critères 1 et 2. En effet, les points en vert sont effectivement proches du plan en pointillé noir (distance inférieure à γ) et les normales de tous les points sont identiques. Nous détectons un plan alors que nous devrions en détecter deux. L'exemple de droite de la figure 4.1 est aussi un contre-exemple pour la définition d'appartenance suivant les critères 2 et 3, tous les points ayant une normale identique entre eux et à la normale du plan. Nous détectons alors un plan en pointillé noir qui ne correspond pas à la surface.

Avec une définition d'appartenance d'un point à un plan suivant les critères 1 et 3 (distance des points au plan inférieur à γ et angle entre normale du plan et normale des points inférieur à α), nous pouvons trouver un contre-exemple lors du processus de croissance comme sur la figure 4.2. Sur cette figure en rouge est indiqué le point d'origine et en vert les points sélectionnés comme appartenant au plan à chaque étape de la

croissance ; le point en bleu étant le point testé. La distance du point en bleu au plan et l'angle entre sa normale et la normale du plan valident l'ajout de ce point au plan. Nous voyons à droite qu'une fois ajouté, nous recalculons le plan par moindres carrés ce qui fait pivoter le plan. La nouvelle normale du plan permet de rajouter d'autres points par croissance de région qui appartiennent en réalité à une surface courbe. Ainsi, lors de la croissance, en utilisant seulement les critères 1 et 3, les surfaces avec une faible courbure sont considérés comme des plans. L'utilisation conjointe des critères 1, 2 et 3 permet de rejeter le point bleu et donc d'arrêter la croissance de région pour une surface qui n'est pas plane.

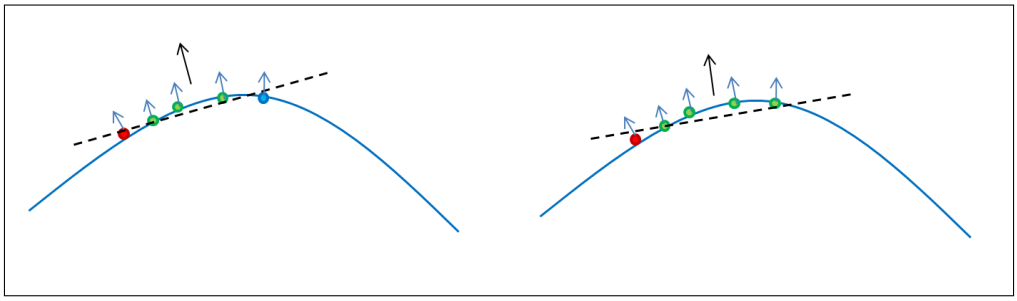


FIGURE 4.2 – Croissance de région avec la définition d'appartenance de points à un plan suivant deux critères, c'est-à-dire la distance des points au plan inférieure à γ et l'angle entre la normale des points et la normale du plan inférieur à α (en vert les points appartenant au plan, en pointillé noir le plan des moindres carrés, les flèches représentent les normales et en bleu la surface réelle).

Ainsi, nous définissons un point comme appartenant à un plan si la distance de ce point au plan est inférieure à γ, que l'angle entre la normale du point et la normale du point d'origine est inférieure à α (équivalent au critère 2) et que l'angle entre la normale du point et la normale du plan est inférieure à α. Si ces trois conditions sont validées pour tous les points du voisinage local du point d'origine alors nous considérons qu'il s'agit d'un plan initial et commençons alors l'étape de croissance de région. Il est important de préciser que ces trois critères seront vérifiés à nouveau à l'ajout de chaque point lors de l'étape de croissance.

Croissance par voxels

Pendant l'étape de croissance, nous utilisons une grille de voxels qui accélère le processus de détection de plans. Les voxels sont de petits cubes de longueur d qui partitionnent l'espace : chaque voxel contenant une liste de points.

Au lieu de croître par voisinage avec les k plus proches voisins ou les points inférieurs à une distance δ, nous effectuons une croissance par voxels. Nous testons tout d'abord tous les points dans le voxel v_0 du point d'origine p_0. Les points qui répondent aux trois critères d'appartenance à un plan sont ajoutés au plan initial. Ensuite, nous testons tous les points dans chacun des 26 voxels voisins toujours suivant les 3 critères définis précédemment. Dans chaque voxel, tous les points sont testés et si au moins un point a été détecté comme appartenant au plan, alors nous itérons le processus et nous testons les 26 voxels voisins de ce nouveau voxel. Parmi tous les voxels à tester, nous prenons le voxel dont le centre

est le plus proche du point d'origine p_0. La figure 4.3 montre un exemple de croissance par projection d'une grille de voxels 3D en 2D. Les voxels en rouge ont déjà été testés et les voxels en vert sont les voxels à tester.

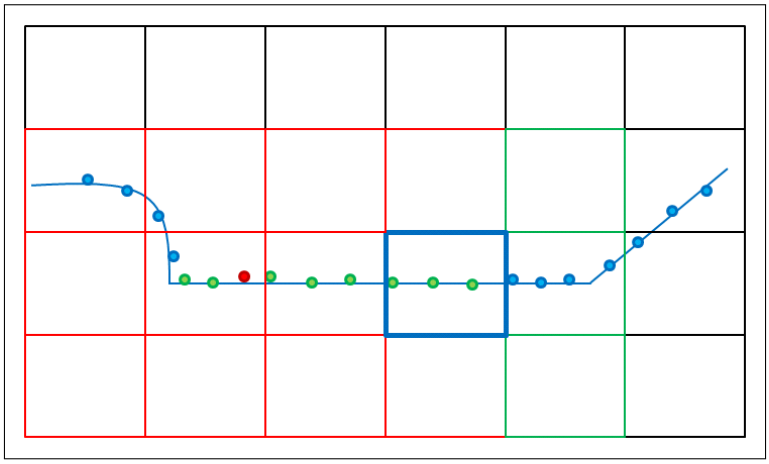


FIGURE 4.3 – Projection en 2D d'une grille 3D avec croissance par voxels (point en rouge : le point d'origine de la croissance de voxels, points en vert : les points sélectionnés comme appartenant à un plan, voxel en bleu : voxel courant de la croissance, voxels en rouge : voxels déjà testés, voxels en vert : voxels à tester).

A chaque point ajouté au plan P, nous devons recalculer ses nouveaux paramètres par moindres carrés en minimisant la somme $\sum_{j\in P}(n_i \cdot p_j + d_i)^2$. Cela permet lors de la croissance par voxels d'améliorer l'estimation des paramètres du plan. Pour ne pas avoir à minimiser cette somme à chaque nouveau point ajouté, nous utilisons une variante itérative de mise à jour du barycentre b et de la matrice de covariance C des points du plan, cette variante est inspirée des travaux de [Poppinga et al., 2008] :

$$\begin{aligned} &b_0 = 0_{3x1} \quad C_0 = 0_{3x3}. \\ &b_{n+1} = \frac{1}{n+1}(nb_n + p_{n+1}). \\ &C_{n+1} = C_n + \frac{n}{n+1}{}^t(p_{n+1} - b_n)(p_{n+1} - b_n). \end{aligned} \tag{4.2}$$

où C_n est la matrice de covariance d'un ensemble de n points, b_n est le barycentre d'un ensemble de n points et p_{n+1} est un vecteur représentant le $(n+1)$ième point ajouté à l'ensemble.

Cette méthode de croissance de voxels donne un ensemble P de points du plan sous forme de composante connexe car les points ont été ajoutés par voxels connectés. Dans notre cas, la distance entre un point de P et son voisin le plus proche dans P sera inférieure à d jusqu'à $2\sqrt{3}d$ (avec d le paramètre de notre grille de voxel) et ceci dépendra de la position du point dans le voxel. C'est pourquoi le paramètre d représente aussi la connectivité des points dans les plans détectés et dans le cas de la croissance par voxels, la connectivité

n'est pas fixe à d mais varie entre d et $2\sqrt{3}d$.

Les voxels ne forment pas simplement une structure alternative pour l'étape de croissance mais l'idée est de remplacer la connaissance exacte de la connectivité lors d'une croissance par voisinages par une connaissance « plus floue » de la connectivité lors d'une croissance par voxels (distance maximale entre points variant entre d et $2\sqrt{3}d$) pour permettre d'accélérer la propagation des points du plan lors de l'étape de croissance en évitant de calculer pour chaque point son voisinage.

Pour pouvoir limiter la taille des plans détectés, nous ajoutons le paramètre n_{max}. Si, au cours de la croissance de voxels, le nombre de points du plan P est supérieur à n_{max} alors nous arrêtons l'étape de croissance, le plan étant validé à la taille maximale autorisée.

Validation des plans

Une fois la croissance par voxels terminée, le plan est conservé si et seulement s'il contient suffisamment de points, c'est-à-dire si le nombre n de points de P est supérieur à n_{min}. Si cela n'est pas le cas, nous conservons les points qui pourront être détectés lors d'une prochaine étape de croissance. De nombreuses méthodes de segmentation rejettent l'ensemble des points du nuage si le plan détecté ne répond pas au critère de taille. Or ces points peuvent appartenir à un plan plus grand. Nous montrons sur la figure 4.4 deux résultats de croissance de région avec deux points d'origine différents. Si nous voulons conserver seulement les plans de grande taille (dans cet exemple plus de 15 points) et que la situation de la figure du haut arrive avant celle de la figure du bas, alors les points ne doivent pas être rejetés pour pouvoir être détectés lorsque la situation du bas se présentera (ce qui sera le cas puisque nous testons tous les points du nuage comme points d'origine). Ceci ne vient pas du fait que le plan initial ait été mal choisi (mauvais point d'origine ou critères insuffisants) car deux plans initiaux peuvent avoir les mêmes caractéristiques locales et donner des plans différents lors de la croissance.

Si le plan est validé car de taille suffisante, tous les points sont classés comme appartenant au plan et ne peuvent plus appartenir à un autre plan. Si le plan est invalidé, alors tous les points sont de nouveau disponibles. Il existe des configurations de nuage de points et des choix de paramètres pour la segmentation qui vont nécessiter un grand nombre de croissances de région. Par exemple, dans un nuage de taille N composé d'un seul plan de taille N, en fixant le paramètre n_{min} à $2N$, alors nous allons pour chaque point du plan faire la croissance par région et détecter le plan de taille N qui sera rejeté : cela va nécessiter N croissances par région. Il est donc important pour pouvoir à la fois détecter de grands plans ou de petits plans d'avoir une croissance très rapide, c'est pourquoi nous avons développé une croissance par voxels et une méthode itérative de mise à jour de la matrice de covariance.

Classification des points d'un plan et des points de bordure

Si nous nous restreignons à notre définition de plan et notre méthode de détection, alors la détection des plans serait identique à la figure 4.5. Nous observons que 5 plans ont été détectés mais les points sur les arêtes ne sont pas classés comme appartenant à un plan.

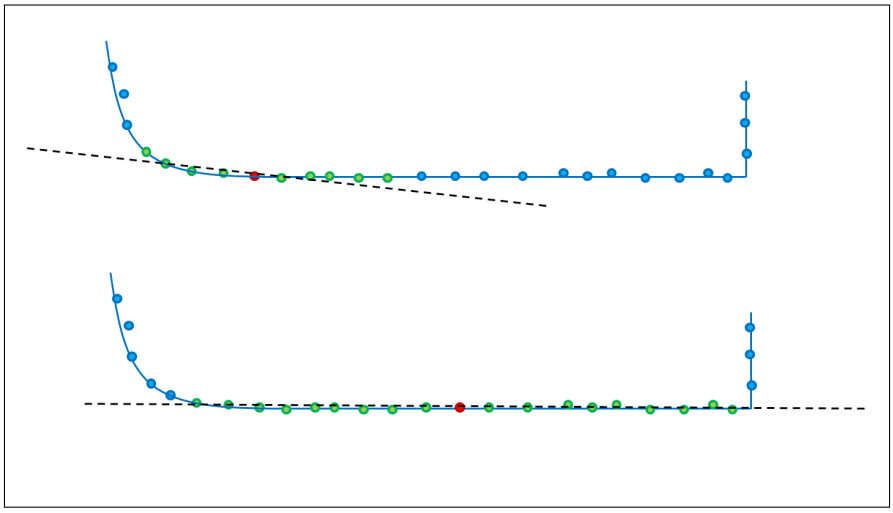


FIGURE 4.4 – Deux résultats différents de la croissance pour deux points d'origine différents (en rouge, le point d'origine, en vert les points du plan, en noir les paramètres du plan par moindres carrés et en bleu les points non sélectionnés).

En effet, les normales des points sur les arêtes ne vont pas permettre de valider les critères d'appartenance pour ces points. Or, comme nous l'avons défini dans le chapitre 3, les points sur les arêtes appartiennent à l'intersection de deux surfaces et ici à l'intersection de deux plans. Les points en bleu devraient donc appartenir à deux plans.

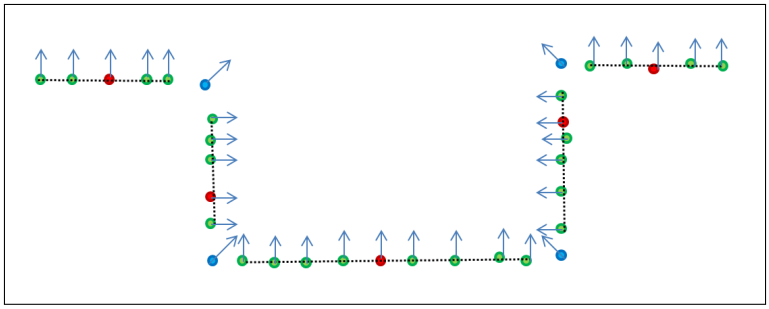


FIGURE 4.5 – Exemple de détection de plans avec notre méthode sans classification des points (en rouge les points d'origine de chaque plan, en vert les points des plans, en bleu les points non sélectionnés, en pointillé noir les plans détectés et les flèches représentent les normales telles que définies au début du chapitre 3).

Pour corriger cela, nous modifions notre définition d'appartenance pour classer les points d'un plan en deux catégories : les points du plan qui satisfont les critères 1, 2 et 3 comme dans les paragraphes précédents et les points de bordure d'un plan qui répondent seulement au critère $1'$ suivant : la distance des points au plan est inférieure à $\frac{\gamma}{2}$. Les points de bordure sont des points que nous ajoutons à l'ensemble des points du plan lors de la croissance qui ne satisfont pas les critères 1, 2 et 3 mais seulement le critère $1'$. Les points de bordure peuvent appartenir à plusieurs plans et ne sont donc pas rejetés du nuage après

la validation du plan. Ces points de bordure supplémentaires permettent de retrouver les points des arêtes et donc de retrouver une segmentation correcte de la surface comme sur la figure 4.6.

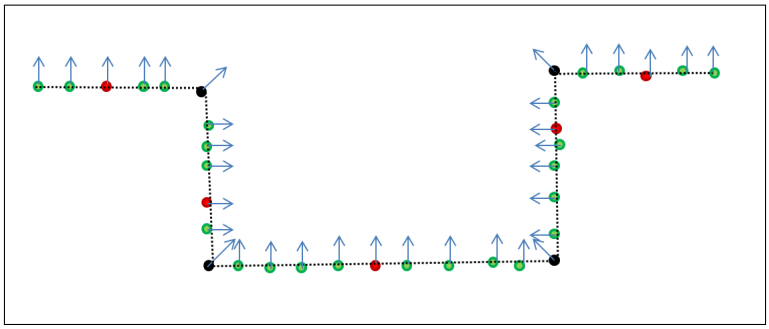


FIGURE 4.6 – Exemple de détection de plans avec classification des points (en rouge les points d'origine, en vert les points des plans, en noir les points de bordures et en pointillé noir les plans détectés).

Dans notre méthode de segmentation, nous voyons que les normales jouent un rôle important pour le critère d'appartenance d'un point à un plan. C'est pourquoi nous avons travaillé sur l'amélioration de l'estimation des normales avec notre algorithme par filtrage de voisinage local (cf. chapitre 3).

Pseudo-code

Pour la détection de plans, nous avons besoin en entrée : d'un nuage de points relativement dense suivant le niveau de détails que nous souhaitons modéliser pour lequel les normales en chaque point ont été calculées avec la méthode décrite dans le chapitre 3. En sortie, nous obtenons des ensembles de points appartenant à des plans.

L'algorithme 1 est le pseudo-code de notre méthode de détection de plans telle décrite dans la section précédente.

4.1.3 Analyse et comparaison des résultats de segmentation sur des données simulées et réelles

Notre méthode de détection de plans comporte six paramètres : δ_{plan}, γ_{plan}, α_{plan}, d_{voxel}, n_{min} et n_{max}. Les paramètres γ_{plan} et α_{plan} vont dépendre du nuage de points et de la segmentation souhaitée tandis que d_{voxel}, n_{min} et n_{max} vont dépendre seulement de la segmentation souhaitée.

δ_{plan} est le rayon du voisinage local et représente aussi les points qui sont sélectionnés pour tester la validation de chaque plan initial.

Algorithme 1 Méthode de détection de plans

Début
  Découpage du nuage en une grille de voxels de taille d_{voxels}
  PourChaque point p_i **De** Nuage **Faire**
    { *Test des points du voisinage de p_i comme plan initial* }
    Voisinage V_i de p_i : $V_i = \{p_j, d(p_j, p_i) < \delta_{plan}\}$
    Calcul du plan P de normale n_P des points de V_i par moindres carrés
    Si $\forall p_j \in V_i \; d(p_j, P) < \gamma_{plan}$ ET $n_j \cdot n_i > cos(\alpha_{plan})$ ET $n_j \cdot n_P > cos(\alpha_{plan})$ **Alors**
      { *Plan initial validé : début de l'étape de croissance* }
      Liste des voxels à explorer $VoxelList \leftarrow v_i$ le voxel contenant p_i
      TantQue $VoxelList \neq \emptyset$ **Faire**
        Prendre le voxel v dans $VoxelList$ le plus proche de p_i
        PourChaque point p_j **De** v **Faire**
          Si $d(p_j, P) < \gamma_{plan}$ ET $n_j \cdot n_i > cos(\alpha_{plan})$ ET $n_j \cdot n_P > cos(\alpha_{plan})$ **Alors**
            $P \leftarrow p_j$
          SinonSi $d(p_j, P) < \frac{\gamma_{plan}}{2}$ **Alors**
            p_j point de bordure de P
          Si Au moins un point a été ajouté dans P **Alors**
            $VoxelList \leftarrow$ voxels voisins du voxel v
            Nouveau calcul de la normale n_P de P par moindres carrés
      Si $n > n_{max}$ (n nombre de points du plan P) **Alors**
        Arrêter l'étape de croissance : $VoxelList \leftarrow \emptyset$
    Si $n > n_{min}$ (n nombre de points du plan P) **Alors**
      Conserver P et enlever les points détectés du nuage
Fin

d_{voxel} correspond à la taille des voxels utilisés dans la croissance par voxels et représente la connectivité des points dans les plans détectés. Cette connectivité est différente de celle introduite par la croissance de région classique. Pour une croissance par région, les points sont ajoutés s'ils sont dans le voisinage de rayon d d'un point. La distance entre un point et son point le plus proche dans le plan détecté sera donc compris entre 0 et d. Pour la croissance par voxels, la distance entre un point et son plus proche voisin sera compris entre 0 et $2\sqrt{3}d$ sachant que tous les points entre d et $2\sqrt{3}d$ du voisinage de chaque point n'auront pas tous été testés. Il s'agit du seul point différenciant en « théorie » le résultat de la segmentation entre croissance par régions et croissance par voxels car dans le deuxième cas, nous testons plus de points pour une même distance d de croissance. Nous avons ainsi une définition de la connectivité moins stricte que celle de la croissance par régions classique. Pour un plan de n points, la complexité de l'algorithme de croissance de régions est un $O(Cn)$ où $O(C)$ est la complexité de recherche du voisinage d'un point. Dans le cas de la croissance par voxels, la complexité est en $O(n)$. Cette différence peut devenir importante lorsqu'il y a

de nombreux plans à détecter et/ou de nombreux plans invalidés après l'étape de croissance.

γ_{plan} est la distance maximale entre le point appartenant à un plan et le modèle du plan. Il représente donc l'épaisseur du plan et sera lié au niveau de détail de détection souhaité et au bruit du nuage de points. Si nous souhaitons détecter uniquement les plans réels de la surface alors nous avons un lien entre γ_{plan} et le bruit du nuage σ. En effet, nous connaissons la distribution statistique des distances à un plan si tous les points suivent une même distribution. D'après [Hartley et Zisserman, 2003] et comme expliqué dans [Boulaassal, 2010], le paramètre γ_{plan} peut se déduire de : $\gamma^2_{plan} = F^{-1}_{\chi^2_2}(p)\sigma^2$ où p est la probabilité d'avoir les points comme appartenant au plan et où $F_{\chi^2_2}$ est la fonction de répartition de la loi χ^2_2 (loi chi-2 avec 2 degrés de liberté). Si nous souhaitons récupérer 99% des points d'un plan, alors nous avons $p = 0.99$ et donc $F^{-1}_{\chi^2_2}(0.99) \approx 9.21$. Nous pouvons ainsi choisir $\gamma_{plan} \approx 3\sigma$. En revanche, pour modéliser l'environnement par des plans si la surface réelle ne correspond pas à un plan, nous devrons choisir un $\gamma_{plan} \geq 3\sigma$.

Le paramètre α_{plan} va déterminer l'angle maximal entre la normale d'un point du plan et la normale du plan mais aussi l'angle entre la normale d'un point et la normale du point d'origine. Ce paramètre va dépendre du bruit du nuage de points mais aussi du niveau de détail souhaité pour la modélisation. Comme pour le paramètre γ_{plan}, si nous souhaitons détecter uniquement les plans correspondant à une zone plane de l'environnement alors le paramètre α_{plan} va dépendre uniquement du bruit σ.

n_{min} représente le nombre minimal de points pour les plans que nous voulons conserver. Pour une densité uniforme ρ du nuage de points, le nombre de points d'un plan va être lié à l'aire de ce plan par la relation $n = \rho a$ où a est l'aire du plan. Il est plus intuitif de fixer le paramètre a_{min} de l'aire minimale des plans à détecter. Dans le cas où n_{min} est fixé à 0, alors tous les plans seront validés après l'étape croissance et auront une aire minimale de $\pi\delta^2_{plan}$, où δ_{plan} est la donnée de voisinage local utilisée pour l'estimation des plans initiaux.

n_{max} représente le nombre maximal de points d'un plan. Cela permet de limiter le nombre de points pour pouvoir faire un découpage précis de zones planes. Si au cours de la croissance de voxels, le nombre de points d'un plan dépasse n_{max} alors nous arrêtons la croissance de voxels. Si ce paramètre est trop petit, alors cela va engendrer une sur-segmentation des plans (un seul grand plan sera découpé en plusieurs petits plans).

Une première version de cette méthode a été publiée à la conférence [$3DPVT$, 2010].

Évaluation de notre méthode de segmentation pour des données simulées

Nous avons tout d'abord testé notre méthode de segmentation sur un nuage de points simple, un cube *S_Cube* composé de 866 points et de 6 faces. Nous voyons sur la figure 4.7 les plans initiaux ou plans graines qui ont été validés après l'étape de croissance. Comme attendu, 6 plans ont été détectés lors de la segmentation. Conformément à ce que nous souhaitions, les points sur les arêtes du cube n'ont pas été détectés comme appartenant aux plans mais bien détectés comme points de bordure.

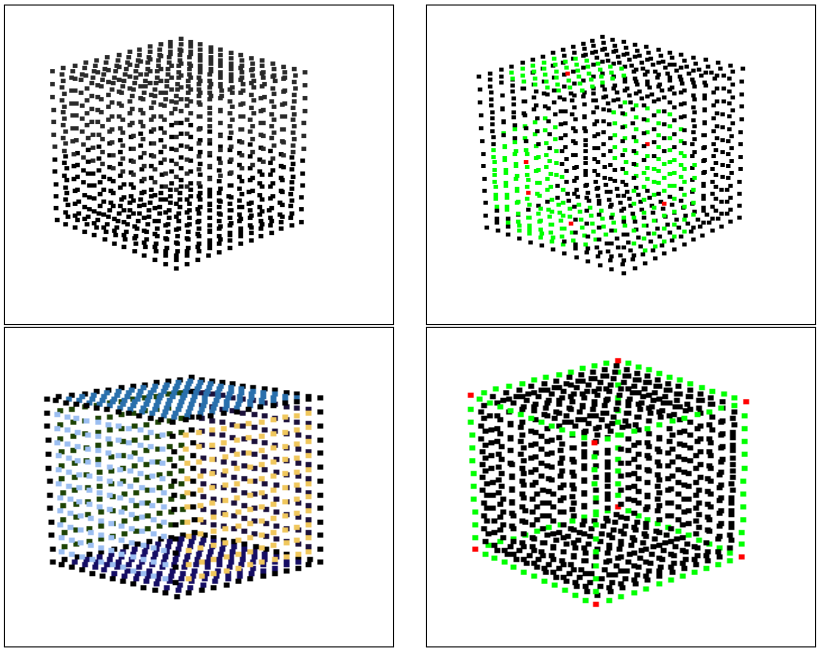


FIGURE 4.7 – En haut à gauche, nuage de points *S_Cube* d'un cube, en haut à droite, en rouge les points d'origine et en vert les plans graines qui ont servi à l'étape de croissance, en bas à gauche les plans détectés avec des couleurs aléatoires et en bas à droite, les points de bordures avec en vert les points qui appartiennent à deux plans et en rouge à trois plans.

Pour évaluer précisément notre méthode de détection de plan, nous avons construit un modèle de façade à l'aide d'un logiciel de modélisation comme sur la figure 4.8 à gauche. Nous avons échantillonné les faces du maillage pour construire un nuage de points *S_Facade* de 136 318 points visible au milieu sur la figure 4.8. Les points appartiennent à des faces d'un maillage simulé donc à des éléments géométriques parfaits (correspondants à des plans, des quadriques, etc.). Ainsi, nous avons pu segmenter toutes les données à la main pour classer les points qui appartiennent à un plan de plus de 1000 points et les points qui n'appartiennent à aucun plan ou à un plan de taille inférieure à 1000 points. Le résultat de cette segmentation est affichée à droite sur la figure 4.8. Cette segmentation nous donne le nuage vérité terrain avec 17 plans extraits. Nous voyons que dans cette segmentation, nous avons voulu que les plans des fenêtres soient détectés comme un seul plan pour chaque fenêtre tandis que la façade est décomposée en deux plans.

Nous appelons les nuages de points segmentés par une méthode automatique nuages SA (pour Segmentation Automatique) et la segmentation vérité, le nuage de points segmenté à la main est appelé nuage VT (pour Vérité Terrain). Pour la comparaison entre nuage SA et nuage VT, nous avons défini en 3D les mêmes notions introduites par [Hoover et al., 1996] pour la segmentation d'images de profondeur. Les plans du nuage SA peuvent être classés en 5 catégories : détection correcte si un pourcentage suffisant de points correspondent entre les deux nuages de points, sur-segmentation si un plan du nuage VT a été segmenté en plusieurs plans du nuage SA, sous-segmentation si un plan du nuage SA correspond à plusieurs plans du nuage VT, manqué pour les plans du nuage VT qui ne sont pas dans

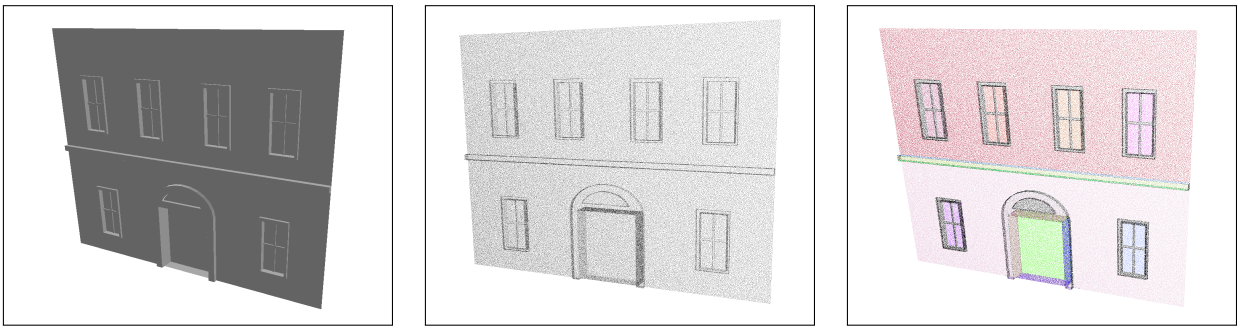


FIGURE 4.8 – A gauche, maillage d'une façade construit sous un logiciel de modélisation, au milieu un nuage de points *S_Facade* extrait par échantillonnage et à droite une segmentation manuelle en zones planes et non planes.

l'une des trois premières classes et enfin bruité pour les plans du nuage SA qui ne sont pas dans l'une des trois premières catégories. Le seuil T est défini de telle sorte que $0.5 \leq T \leq 1$ et représente la précision demandée pour la classification des points. Pour toute la suite, nous avons pris un taux de tolérance $T = 80\%$.

Les principaux apports de notre méthode sont les critères d'appartenance d'un point à un plan et la croissance par voxels. Pour évaluer cela, nous avons testé deux variantes de notre méthode. Comme première variante, nous avons utilisé seulement le critère de distance pour définir l'appartenance d'un point à un plan (seulement le critère 1 à la place des critères 1, 2 et 3) sans modifier les autres apports. Nous appelons cette variante « Sans Normales » car cela consiste à ne pas tenir compte des normales du nuage de points. L'autre variante consiste à tester l'impact de la croissance par voxels sur la qualité de la segmentation. C'est pourquoi nous avons remplacé la croissance par voxels par une méthode classique de croissance de région par voisinage. Cette variante 2 est appelée « Croissance par région ». De la même façon que pour la première variante, nous modifions seulement une partie de notre méthode pour permettre de tester l'influence de la croissance sur les résultats de la segmentation.

Notre algorithme a aussi été comparé à une autre méthode basée sur l'estimateur robuste RANSAC. Pour RANSAC, nous avons aussi défini l'appartenance d'un point à un plan suivant les 3 critères que nous avons vu. Le nombre d'itérations est $T_{min} = \frac{log(1-p_t)}{log(1-(\frac{n_{min}}{N})^3)}$ avec $p_t = 0.95$. Pour tous ces tests, les normales des nuages de points en entrée sont calculées avec notre méthode FMLS (« *Filtered Moving Least Squares* ») détaillée au chapitre 3. Nous avons voulu voir la différence de résultat sur la segmentation en utilisant une méthode classique de calcul de normales comme WPF (« *Weight Plane Fitting* ») à la place de FMLS. Enfin, la robustesse de notre méthode de détection au bruit a été vérifiée pour trois niveaux de bruit : 0%, 10% et 30% de la distance moyenne entre points. Tous ces calculs ont été effectués sur un QuadCore Q9300 2.50 GHz, 2 Go de RAM, sur un seul *thread* et sur CPU.

Le Tableau 4.1 présente les résultats de la segmentation automatique pour le nuage de

points *S_Facade* avec un niveau de bruit de 0% de la distance moyenne entre points. Même sans bruit, RANSAC et la variante 2 « Sans Normales » ne permettent pas de retrouver le bon nombre de régions. La variante 1 « Croissance de Région » et notre méthode (avec les normales WPF et FMLS) détectent le bon nombre de région (17). Mais nous constatons qu'il y a une différence dans le pourcentage de détection correcte des plans. Notre méthode complète apporte les meilleurs résultats avec 98.6% de détection correcte entre les régions VT et SA. Avec une détection de 97.6% en utilisant les normales WPF, nous remarquons que le fait de calculer les normales avec notre méthode FMLS donne une meilleure segmentation. La différence de 1 ou 2% entre FMLS et WPF va se situer principalement sur les bords des plans et ce sont les points les plus importants car ce sont eux qui vont déterminer le contour du plan final. Ceci est dû à notre méthode de calcul des normales par sélection d'« *outliers* » dans les voisinages locaux qui permet de mieux estimer les normales autour des points de discontinuité, c'est-à-dire souvent aux bords des plans. Le taux de détection de la variante 1 « Croissance de Région » de 98.3% est presque équivalent à celui de notre méthode. Mais il y a un facteur 4 de temps entre la croissance de région et la croissance par voxels. Il faut noter que cette différence de temps va beaucoup dépendre de la configuration du nuage de points et de la segmentation souhaitée.

	Régions SA	Régions Correctes	Sur-Seg	Sous-Seg	Manqué	Bruit	Temps
Variante 1 : « Croissance par Région »	17	17 (98.3%)	0	0	0	0	188 *s*
Variante 2 : « Sans Normales »	16	15 (95.0%)	0	1	0	0	116 *s*
Méthode RANSAC	11	8 (97.0%)	0	2	1	1	287 *s*
Notre méthode avec normales WPF	17	17 (97.6%)	0	0	0	0	39 *s*
Notre Méthode avec normales FMLS	17	17 (98.6%)	0	0	0	0	41 *s*

TABLE 4.1 – Résultats de notre méthode de segmentation et comparaison avec deux variantes, la méthode RANSAC et en modifiant les données d'entrée (calcul des normales avec WPF à la place de FMLS) pour le nuage *S_Facade* (17 Régions VT) et pour un niveau de bruit de 0%.

Les tableaux 4.2 et 4.3 présentent les mêmes résultats de segmentation mais pour des niveaux de bruit différents de 10% et de 30% de la distance moyenne entre points. Nous pouvons faire les mêmes remarques : la variante 2 « Sans Normales » et l'algorithme RANSAC ne permettent d'atteindre le résultat de segmentation souhaité, le calcul des normales avec FMLS permet d'améliorer la segmentation et la variante 1 « Croissance par région » est de 2 à 4 fois plus lente que notre méthode de détection de plans.

Segmentation sur une base de données d'images de profondeur avec vérité terrain

Nous avons testé notre méthode de segmentation sur une base de données d'images de profondeur disposant d'une vérité terrain et généré avec un LiDAR Perceptron (le Perceptron est un scanner portable qui produit une image de profondeur de son environnement). Cette base de donnée a été créée au laboratoire CESAR de l'ORNL (« *Oak Ridge National Laboratory* ») et décrite dans [Hoover et al., 1996]. Puisque notre méthode fonctionne sur des données 3D, nous avons converti toutes les images de profondeur en nuages de points.

	Régions SA	Régions Correctes	Sur-Seg	Sous-Seg	Manqué	Bruit	Temps
Variante 1 : « Croissance par Région »	17	17 (98.1%)	0	0	0	0	123 *s*
Variante 2 : « Sans Normales »	16	14 (95.0%)	0	1	1	1	104 *s*
Méthode RANSAC	10	6 (96.7%)	0	3	1	1	281 *s*
Notre méthode avec normales WPF	17	17 (97.6%)	0	0	0	0	26 *s*
Notre Méthode avec normales FMLS	17	17 (98.4%)	0	0	0	0	26 *s*

TABLE 4.2 – Résultats de notre méthode de segmentation et comparaison avec deux variantes, la méthode RANSAC et en modifiant les données d'entrée (calcul des normales avec WPF à la place de FMLS) pour le nuage *S_Facade* (17 Régions VT) et pour un niveau de bruit de 10%.

	Régions SA	Régions Correctes	Sur-Seg	Sous-Seg	Manqué	Bruit	Temps
Variante 1 : « Croissance par Région »	17	17 (96.7%)	0	0	0	0	136 *s*
Variante 2 : « Sans Normales »	21	9 (90.3%)	0	1	6	11	67 *s*
Méthode RANSAC	11	5 (95.4%)	0	3	2	3	326 *s*
Notre méthode avec normales WPF	17	17 (96.1%)	0	0	0	0	56 *s*
Notre Méthode avec normales FMLS	17	17 (97.1%)	0	0	0	0	49 *s*

TABLE 4.3 – Résultats de notre méthode de segmentation et comparaison avec deux variantes, la méthode RANSAC et en modifiant les données d'entrée (calcul des normales avec WPF à la place de FMLS) pour le nuage *S_Facade* (17 Régions VT) et pour un niveau de bruit de 30%.

Après la segmentation, les points labellisés sont projetés sur une image ce qui donne l'image de segmentation et nous comparons cette image avec l'image de vérité terrain. La figure 4.9 montre les images segmentées avec à gauche la vérité terrain générée à la main, au milieu, la segmentation avec notre méthode en utilisant les normales calculées avec WPF et à droite notre méthode de segmentation mais cette fois en utilisant les normales calculées par FMLS. Les points noirs dans les images représentent les points non classés. Nous voyons que le fait d'utiliser notre méthode de calcul de normale FMLS permet d'avoir moins de points noirs (non classés) sur les bords des objets et donc une meilleure estimation des bords des plans détectés.

Le Tableau 4.4 compare les résultats de segmentation de 30 images tests de Perceptron avec deux méthodes classiques : UE est une méthode de [Hoover et al., 1996] et UFPR est une méthode de [Gotardo et al., 2003]. Il est important de noter que UE et UFPR sont des méthodes de segmentation d'images de profondeur et que notre méthode n'est pas adaptée pour les images de profondeur mais seulement pour les nuages de points. Néanmoins, ces données nous permettent de faire des comparaisons quantitatives et comme on le constate sur le tableau 4.4, la précision de notre méthode est très proche de l'état de l'art de la segmentation d'images de profondeur. Plus précisément, notre méthode manque plus de plans que la méthode UFPR (10.9 régions correctes pour notre méthode et 11.0 pour la méthode UFPR) : ce sont surtout les très petits plans qui ne sont pas détectés à cause de la sélection stricte des plans initiaux. Malgré cette imprécision, on remarque que notre segmentation génère moins de plans bruits, c'est-à-dire moins de plans qui n'existent pas

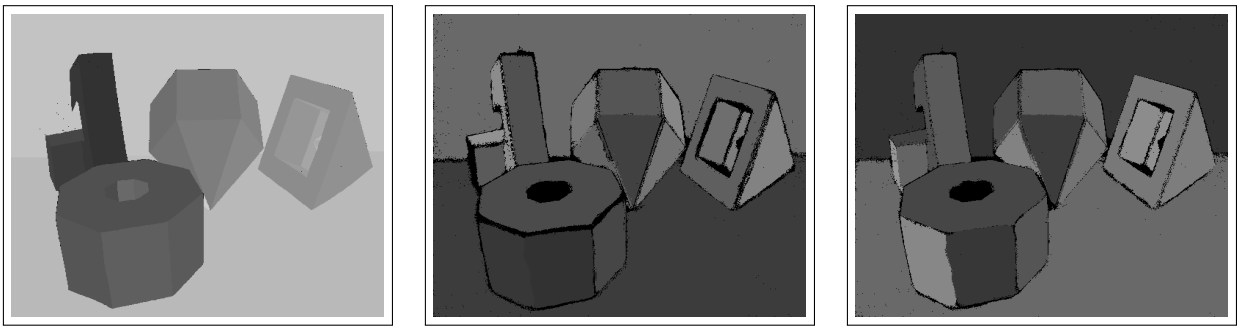


FIGURE 4.9 – A gauche image VT (Vérité Terrain), au milieu notre méthode de segmentation en utilisant les normales calculées avec WPF et à droite en utilisant les normales calculées avec FMLS.

dans la vérité terrain (0.7 régions bruits pour notre méthode et 2.1 pour la méthode UE) : ceci est principalement dû à nos critères stricts d'appartenance d'un point à un plan.

	Régions VT	Régions Correctes	Sur-Seg	Sous-Seg	Manqué	Bruit
UE	14.6	10.0	0.2	0.3	3.8	2.1
UFPR	14.6	11.0	0.3	0.1	3.0	2.5
Notre méthode	14.6	10.9	0.2	0.1	3.3	0.7

TABLE 4.4 – Moyenne des résultats de différentes méthodes de segmentation sur 30 images tests de Perceptron avec un taux de tolérance de 80%.

Segmentation sur des données réelles

Nous avons testé notre méthode sur des données réelles provenant d'acquisitions produites par des technologies différentes. Nous avons des données de notre plateforme mobile LARA3D en environnement urbain *LARA3D_Orsay*, des données de la plateforme mobile Stereopolis de l'IGN pour la rue Soufflot, *Stereopolis_Soufflot*, des données de scanners fixes précis comme les Trimble FX et GX, *FX_GX_Orsay* mais aussi moins précis avec le scanner Trimble VX, *VX_Soufflot*.

La figure 4.10 illustre les résultats de la segmentation pour des données provenant du système de cartographie mobile LARA3D et représentant la façade nord du musée d'Orsay. Nous voyons que les points de bordures des plans correspondent approximativement aux arêtes de la façade.

La figure 4.11 met en évidence les résultats de segmentation pour le nuage de points *Stereopolis_Soufflot* qui provient du système mobile de cartographie de l'IGN Stereopolis.

La figure 4.12 expose les résultats de segmentation pour le nuage de points *FX_GX_Orsay* qui provient des scanners fixes Trimble FX et GX avec des données

très précises.

La figure 4.13 montre les résultats de segmentation pour le nuage de points *VX_Soufflot* qui provient d'un scanner fixe Trimble VX avec des données de précision moyenne. Nous avons souhaité extraire uniquement les façades lors de cette segmentation, c'est pourquoi nous avons choisi $n_{min} = 10\ 000$.

Nous pouvons constater visuellement sur toutes ces images que les segmentations correspondent à ce qui est attendu, c'est-à-dire que nous obtenons des plans dont les bordures correspondent à des arêtes des façades.

Tous les résultats de segmentation sont détaillés dans le tableau 4.5. Les 4 premières lignes détaillent les caractéristiques des nuages de points avec le nombre de points, la distance moyenne d'un point à son plus proche voisin, la densité moyenne et l'écart-type $\sigma_{d_{mean}}$ sur la distance moyenne. Les 5 lignes suivantes donnent les paramètres de la méthode de segmentation pour chaque nuage de points. Certains paramètres ont été choisis pour segmenter le nuage en plans de petites tailles comme pour la segmentation du nuage *LARA3D_Orsay* et d'autres paramètres ont été choisis de manière à montrer que l'on peut aussi utiliser cette méthode pour segmenter les nuages en plans de grande taille comme pour *VX_Soufflot* où nous gardons seulement les plans de plus de 10 000 points. Nous pouvons voir sur ce tableau 4.5 que notre méthode est approximativement linéaire en fonction du nombre de points avec un temps de traitement d'environ 0.2 ms/pt.

	LARA3D_Orsay	*Stereopolis_Soufflot*	*FX_GX_Orsay*	*VX_Soufflot*
Nombre de points	2 440 416	1 007 747	1 804 648	42 477
Distance moyenne	4.8 cm	4.6 cm	5.9 cm	24.4 cm
$\sigma_{d_{mean}}$	2.6 cm	2.8 cm	2.0 cm	14.1 cm
Densité moyenne	419.7 pts/m^2	459.7 pts/m^2	287.0 pts/m^2	16.7 pts/m^2
δ_{plan}	30 cm	30 cm	30 cm	1.5 m
γ_{plan}	7 cm	7 cm	7 cm	50 cm
α_{plan}	10 °	10 °	10 °	30 °
d_{voxel}	50 cm	50 cm	50 cm	2.0 m
n_{min}	100	100	100	10 000
n_{max}	100 000	100 000	100 000	100 000
Nb plans détectés	159	359	861	2
Durée totale	377 s	184 s	305 s	17 s
Temps/point	0.16 ms	0.18 ms	0.17 ms	0.4 ms

TABLE 4.5 – Synthèse des résultats de segmentation sur des données réelles.

4.1.4 Projection des points sur les plans détectés

L'étape de projection consiste une fois les zones planes détectés à projeter les points sur leur plan respectif. Pour les points qui appartiennent à un seul plan, c'est une projection

orthogonale. Pour les points de bordure qui appartiennent à deux plans, alors il s'agit d'un point appartenant à une arête et nous le projetons sur la droite d'intersection des deux plans dont il fait parti. Si un point de bordure appartient à au moins trois plans, alors il s'agit d'un coin et nous le projetons sur le point minimisant les distances orthogonales aux plans (dans le cas de trois plans, il s'agit de l'intersection des trois plans).

4.1.5 Conclusion sur la méthode de segmentation proposée

La méthode de segmentation proposée est une amélioration de la croissance de région avec une nouvelle définition de l'appartenance d'un point à un plan, une sélection stricte des plans initiaux et une étape de croissance basée sur les voxels. Nous avons détaillé en quoi et comment les critères d'appartenance d'un point à un plan peuvent changer les résultats d'une segmentation. Nous avons aussi vu que la croissance par voxels améliore significativement d'un facteur 2 à 4 les temps de traitement sans diminuer la qualité de segmentation. Enfin, nous avons aussi appliqué notre méthode de calcul de normales FMLS développée dans le chapitre 3 et nous avons constaté qu'elle est bien adaptée pour améliorer les résultats de segmentation, surtout sur les bords des plans, ce qui est une des difficultés majeures de la segmentation.

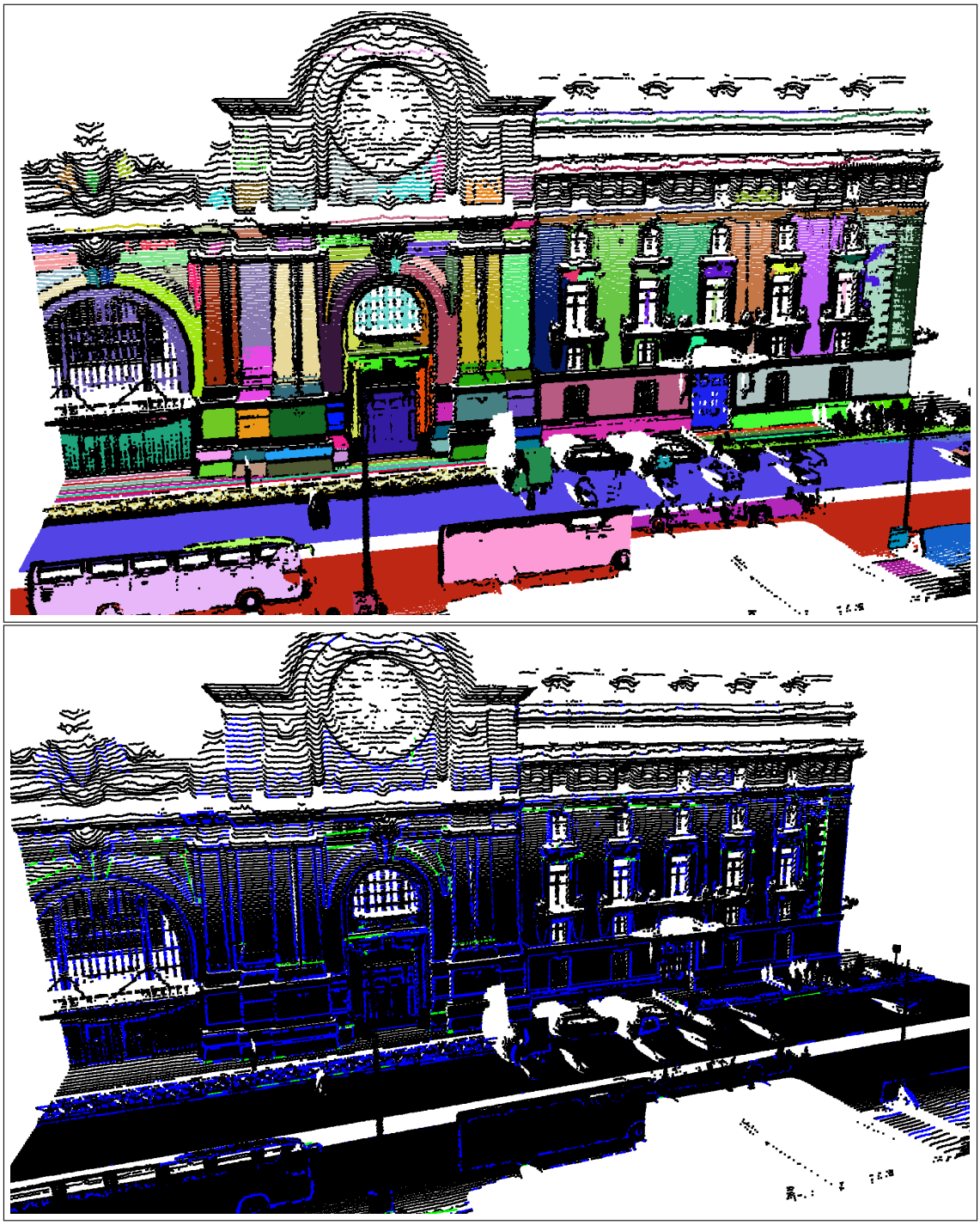


FIGURE 4.10 – En haut nuage de points *LARA3D_Orsay* avec les plans détectés dans des couleurs aléatoires (en noir les points non segmentés) et en bas, même nuage de points segmenté avec en bleu les points des bordures des plans détectés.

FIGURE 4.11 – Segmentation du nuage de points *Stereopolis_Soufflot* avec les plans dans des couleurs aléatoires et les points non classés en noir.

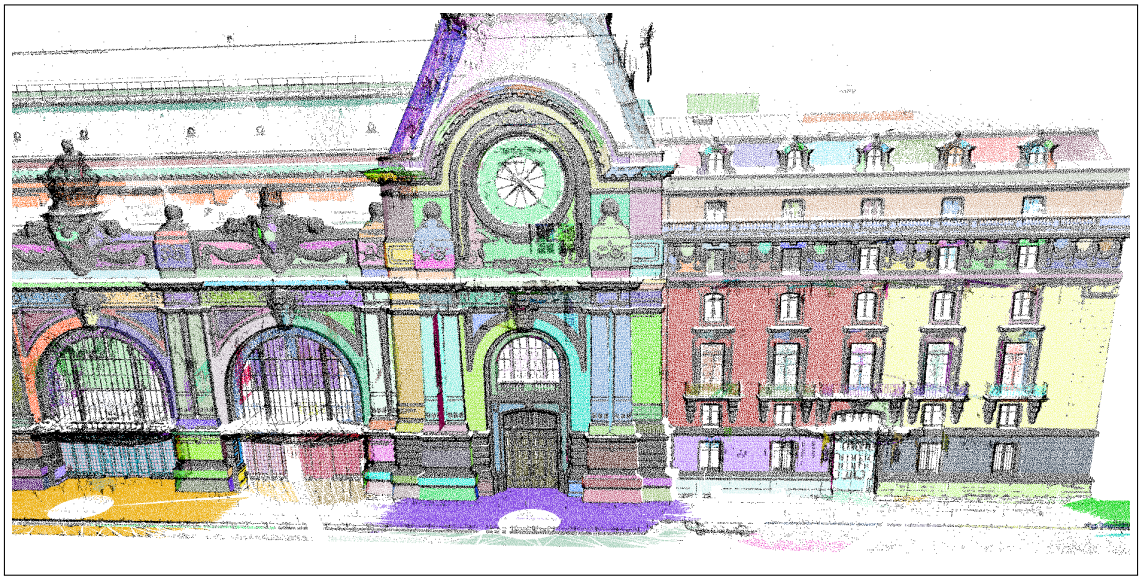


FIGURE 4.12 – Segmentation du nuage de points *FX_GX_Orsay* avec les plans dans des couleurs aléatoires et les points non classés en noir.

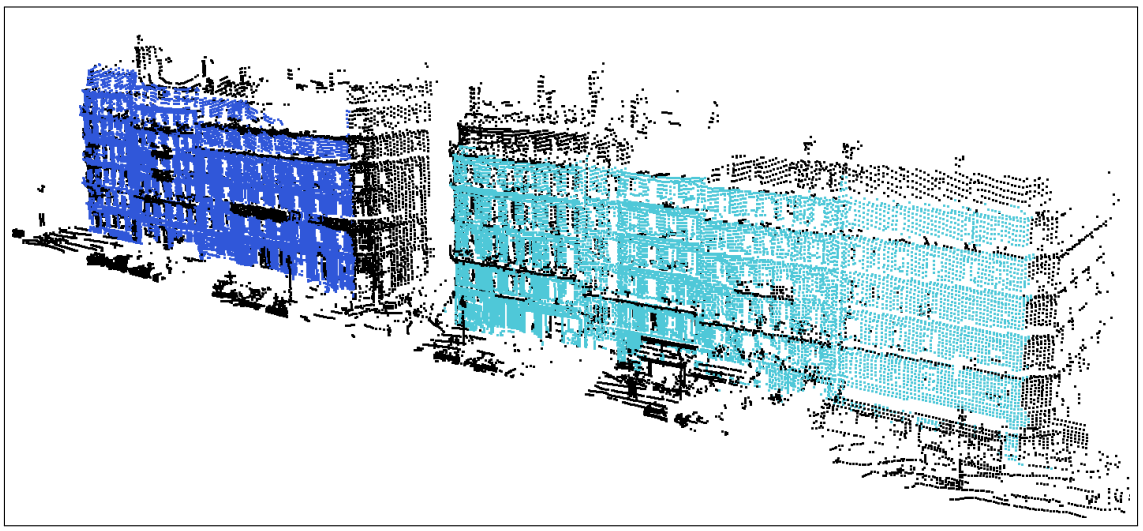


FIGURE 4.13 – Segmentation du nuage de points *VX_Soufflot* (2 plans).

4.2 Triangulation de nuages de points

Nous présentons dans cette section notre méthode de maillage de nuage de points basée sur une triangulation rapide dans le plan tangent. Nous considérons comme entrée un nuage de points débruité, avec l'information des normales aux points. Le but est alors de construire un maillage par interpolation, c'est-à-dire dont les sommets des triangles sont des points non modifiés du nuage. L'objectif est d'obtenir un maillage qui approche au mieux la surface réelle.

4.2.1 Etat de l'art

De nombreuses méthodes de triangulation existent dans la littérature. On distingue deux grandes catégories : les méthodes par interpolation et les méthodes par approximation ainsi que l'illustre la figure 4.14. Nous avons comme donnée d'entrée un ensemble de points dont nous cherchons à reconstruire la surface. A gauche, l'interpolation va relier les points par des segments en supposant que les points appartiennent à la surface. A droite, l'approximation va créer une surface en minimisant une fonction vis à vis des points d'entrée. On note que les deux résultats sont très différents et il est possible qu'aucune des deux reconstructions ne corresponde à la surface réelle. La première méthode sera plus utilisée pour des données denses, précises, contenant peu de points aberrants (« *outliers* ») - comme les données de LiDAR - tandis que la seconde méthode sera plus utilisée avec des données d'entrée moins denses, moins précises et contenant de nombreux « *outliers* » comme les nuages de points produits par stéréovision. Dans le cadre de notre modélisation de données laser provenant de systèmes mobiles, mêmes si les données sont plus bruitées, moins uniformément réparties que les données provenant de scanners fixes, la méthode par interpolation est plus appropriée pour reconstruire les surfaces. En effet, celle-ci permet notamment de reconstruire plus facilement les arêtes, fréquentes dans la modélisation d'environnements urbains. Nous notons aussi l'importance d'avoir un nuage de points débruité ou tout du moins, avec un bruit suffisamment faible pour minimiser l'erreur de reconstruction. C'est pourquoi certains nuages de points ont été débruité par la méthode de débruitage NLD3D décrite au chapitre 3 avant l'étape de triangulation.

L'objectif de notre reconstruction est de créer un « maillage variété » orienté avec bords. Rappelons qu'un maillage triangulaire est un ensemble désordonné de triangles. En ajoutant le terme orienté, nous voulons que chaque triangle soit orienté vers l'extérieur de la surface. L'ordre des sommets $p_1p_2p_3$ signifie que l'extérieur est dans le sens $(p_2 - p_1) \wedge (p_3 - p_1)$. Cela est nécessaire par exemple pour connaître les faces à éclairer lors du rendu du maillage. La terminologie « maillage variété » est impropre mais elle fait référence au terme anglais « *manifold mesh* » expliqué dans [Schroeder et al., 1992]. Une 2-variété avec bords est une notion qui vient de la géométrie différentielle : chaque point de la surface doit être localement homéomorphe à $\mathbb{R}^2$ ou $\mathbb{R} \times \mathbb{R}_+$. Dans le cas d'un ensemble de triangles représentant une surface discrète, variété avec bord signifie comme expliqué dans [Schroeder et al., 1992] que chaque arête pq orientée de la triangulation doit appartenir à un seul triangle et que pour chaque point p, l'ensemble des triangles dont un des sommets est p forment un cycle ou un demi-cycle (point du bord) comme sur la figure 4.15. Le caractère variété est nécessaire pour de nombreux algorithmes de traitements

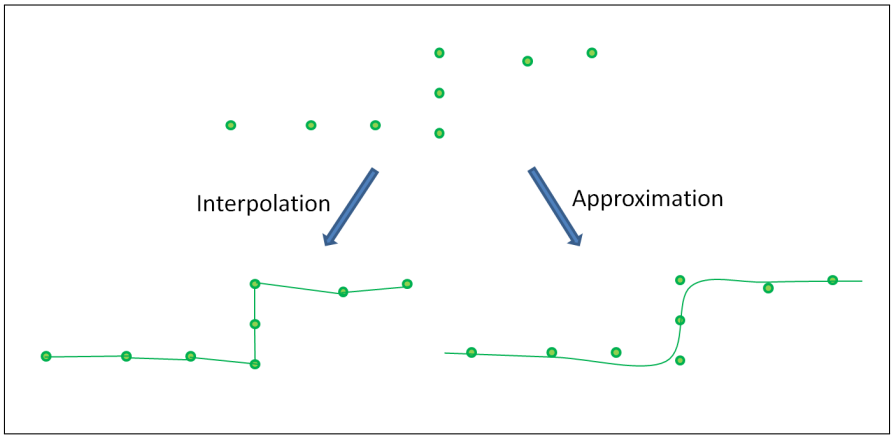


FIGURE 4.14 – Méthodes de reconstruction d'un ensemble de points.

de maillages et permet aussi d'éviter certains artefacts lors de la visualisation du modèle. Avec bords signifie que le modèle n'est pas complètement fermé et peut contenir des trous. Les trous du maillage vont alors correspondre à des trous présents dans l'environnement à modéliser (par exemple une fenêtre ouverte dans une façade). En plus de créer un maillage variété orienté avec bords, nous souhaitons construire des triangles avec une forme la plus proche possible d'un triangle équilatéral, c'est-à-dire en maximisant le plus petit des angles de la triangulation. En 2D, la triangulation de Delaunay est une triangulation qui maximise le plus petit angle de l'ensemble des angles des triangles.

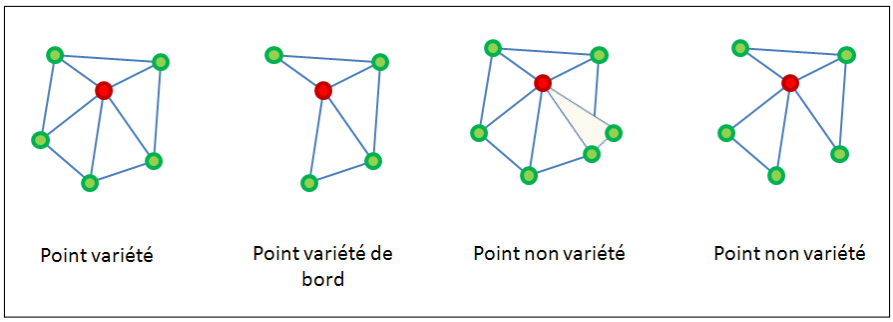


FIGURE 4.15 – Configurations possibles de triangles autour d'un point.

Méthodes par approximation

Les méthodes par approximation peuvent être classées en trois catégories : les méthodes par approximation locale, les méthodes par approximation globale et les méthodes basées sur les statistiques. La plupart de ces méthodes se basent sur le calcul d'une isosurface locale ou globale représentant la surface, c'est-à-dire l'ensemble des points vérifiant la relation $f(x, y, z) = 0$. C'est pourquoi ces méthode sont aussi souvent appelées méthodes implicites. Une fois la fonction f connue, la surface peut être retrouvée à l'aide d'un algorithme comme

les « Marching Cubes » de [Lorensen et Cline, 1987]. Cet algorithme partitionne l'espace en voxels ou cubes. Chacun des 8 sommets d'un voxel est évalué par f pour savoir s'il est à l'intérieur ou l'extérieur du volume défini par le modèle. Les 8 sommets donnent $2^8 = 256$ possibilités qui peuvent par symétrie se ramener à 15 cas. Suivant les cas, 0, 1 ou plusieurs triangles sont créés à l'intérieur du voxel avec comme sommets des triangles des points sur les arêtes du voxels dont la position est interpolée linéairement par les deux valeurs scalaires de f des sommets de cette arête. La figure 4.16 montre les étapes de l'algorithme des « Marching Cubes ».

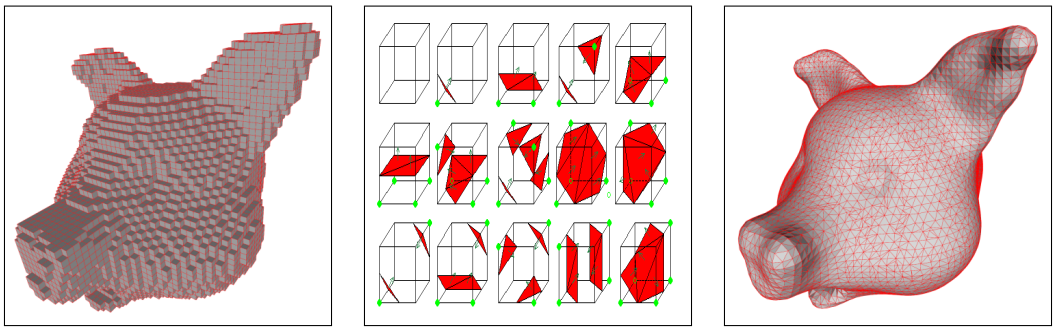


FIGURE 4.16 – Algorithme des « Marching Cubes » avec à gauche la décomposition en voxels, au milieu les 15 configurations possibles et à droite le maillage final.

Les méthodes par approximation locale définissent localement une fonction approximant la surface. Cette fonction peut être des combinaisons linéaires de gaussiennes comme [Taubin et al., 1994], des variantes des Radial Basis Functions (RBF) comme dans [Carr et al., 2003], [Turk et O'Brien, 2002], [Samozino et al., 2006], [Kojekine et al., 2004] ou des variantes des Moving Least Squares (MLS) comme dans [Levin, 1998], [Fleishman et al., 2005], [Kolluri, 2005] ou [Oztireli et al., 2009]. Les premiers travaux de [Hoppe et al., 1992] définissent comme fonction implicite la distance signée des points à leurs plans tangents. Cette distance signée peut être accumulée dans une grille volumique pour construire le maillage d'un ensemble d'images de profondeur commme dans [Curless et Levoy, 1996].

Pour les méthodes par approximation globale, nous pouvons citer [Kazhdan et al., 2006] qui reconstruisent une surface à l'aide d'une méthode, appelée méthode de Poisson, basée sur l'extraction d'une fonction indicatrice valant 1 dans le modèle et 0 à l'extérieur. Le gradient de cette fonction indicatrice doit s'approcher au mieux des normales des points. Ce problème de minimisation se ramène alors à résoudre une équation de Laplace-Poisson.

Pour les méthodes basées sur les statistiques, [Jenke et al., 2006] et [Diebel et al., 2006] utilisent les statistiques bayésiennes pour inférer une probabilité maximale de reconstruction. Enfin, le « *Power Crust* » de [Amenta et al., 2001] est une méthode basée sur le diagramme de Voronoï qui donne une triangulation par approximation et qui offre des garanties théoriques de reconstruction sous certaines conditions d'échantillonnage. Cette méthode a été améliorée dans [Mederos et al., 2005] pour fonctionner sur des nuages de points bruités.

Ces méthodes de reconstruction par approximation permettent pour la plupart de reconstruire des surfaces à partir de données très bruitées, avec de nombreux « *outliers* » et même avec des données non denses et non uniformément réparties. Ces méthodes sont ainsi très robustes aux différents types de données en entrée. Par contre, avec un modèle exact, c'est-à-dire un nuage de points dont les points sont sur la surface, elles vont alors faire de nombreuses erreurs de reconstruction, notamment sur les arêtes. La plupart des méthodes par approximation nécessitent la connaissance des normales en chaque point et de connaître l'intérieur et l'extérieur, c'est-à-dire d'avoir une bonne orientation des normales. Elles nécessitent aussi souvent l'hypothèse que le modèle soit fermé, c'est-à-dire que la surface soit représentée par un volume. Les maillages obtenus sont alors des modèles qui ne comportent pas de trou, ce qui peut devenir un inconvénient pour obtenir des modèles 3D correspondant à des environnements réels où il y a de nombreux trous. Certaines méthodes par approximation, notamment les techniques basées sur des statistiques bayésiennes, nécessitent d'optimiser un modèle global : il est alors difficile de pouvoir traiter de grands volumes de données avec par exemple des nuages de plus d'un million de points.

Méthodes par interpolation

Les méthodes par interpolation peuvent aussi être classées en deux catégories : celles qui fonctionnent par propagation et celles qui utilisent un diagramme de Voronoï. Un diagramme de Voronoï est une décomposition de l'espace en régions à partir d'un ensemble P de n points d'un espace vectoriel normé E (ici $E = \mathbb{R}^2$ ou $E = \mathbb{R}^3$). Une région ou cellule de Voronoï pour un point p de P est l'ensemble des points qui sont plus proches de p que de tout autre point de P : $Vor(p) = \{x \in E, \forall q \in P, \|p - x\| \leq \|q - x\|\}$. La triangulation de Delaunay est le dual du diagramme de Voronoï : deux points p et q créent une arête dans le graphe de Delaunay si et seulement si les régions de Voronoï associées à p et à q sont adjacentes : $D(P) = \{(p, q) \in P^2, Vor(p) \cap Vor(q) \neq \emptyset\}$. En 2D, la triangulation de Delaunay $D(P)$ est l'ensemble des triangles tels qu'aucun point de P n'est à l'intérieur du cercle circonscrit du triangle. En 3D, la triangulation de Delaunay $D(P)$ est l'ensemble des tétraèdres dont les sphères circonscrites ne contiennent aucun point de P. Les réciproques des deux dernières propriétés sont vraies, c'est-à-dire que si un ensemble de triangles ou de tétraèdres vérifient les mêmes conditions sur le cercle circonscrit ou la sphère circonscrite alors il s'agit d'une sous-partie de la triangulation de Delaunay. Les figures 4.17 et 4.18 montrent les diagrammes de Voronoï et les triangulations de Delaunay d'un ensemble de points dans $\mathbb{R}^2$ et $\mathbb{R}^3$. Il n'y a pas toujours existence d'une triangulation de Delaunay d'un ensemble de points (par exemple 3 points alignés), ni unicité (par exemple 4 points cocycliques).

[Boissonnat, 1984] a été un des premiers à présenter une méthode de reconstruction par propagation en construisant une triangulation dans le plan tangent local. [Bernardini et al., 1999] avec sa méthode appelée BPA pour « *Ball Pivoting Algorithm* » sélectionne tout d'abord un triangle initial répondant à certains critères puis fait tourner une boule de rayon fixe sur la surface pour créer de nouveaux triangles. [Huang et Menq, 2002] restreint la propagation aux k plus proches voisins et fait varier localement le paramètre

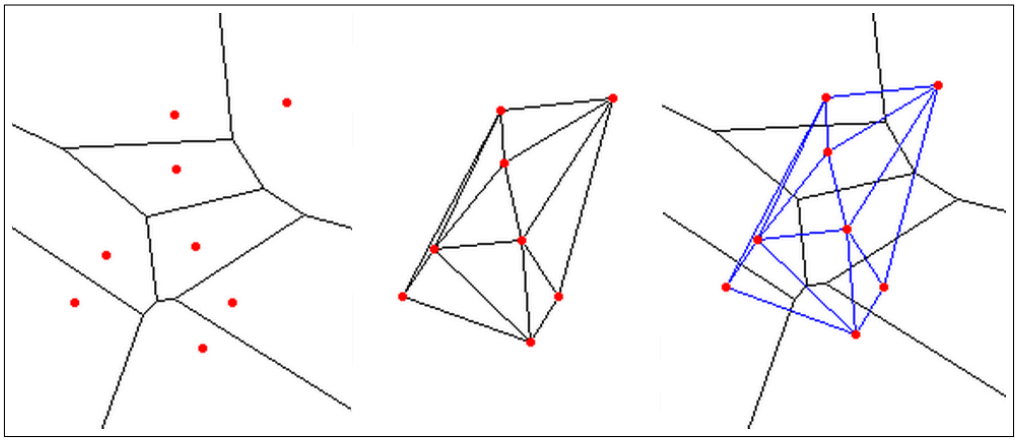


FIGURE 4.17 – A gauche, diagramme de Voronoï d'un ensemble de points de $\mathbb{R}^2$, au milieu, sa triangulation de Delaunay et à droite le diagramme et la triangulation.

rayon. Un autre algorithme de [Lin et al., 2004] est basé sur l'IPD pour « Intrinsic Property Driven », c'est-à-dire le ratio des longueurs des plus petites et plus longues arêtes incidentes en un point. [Gopi et al., 2000] et [Gopi et Krishnan, 2002] utilisent aussi une projection dans le plan tangentiel pour calculer une triangulation de Delaunay locale en décimant les points dans le voisinage local suivant les courbures principales. [Crossno et Angel, 1999] construisent aussi une triangulation par propagation en projetant localement les points dans l'espace tangent.

Les méthodes par propagation ont besoin de nuages peu ou pas bruités en entrée. Même si le nuage n'est pas bruité, les méthodes peuvent laisser des trous indésirables dans le maillage souvent dus à des situations ambigües comme par exemple 4 points cocycliques, qui peuvent engendrer plusieurs triangulations de Delaunay possibles ou lorsque deux surfaces sont très proches. Ces méthodes souffrent aussi souvent de leur manque de preuves théoriques et des erreurs topologiques qu'elles peuvent générer. Malgré cela, ce sont des méthodes souvent très rapides et qui permettent de bien interpoler le modèle au niveau des arêtes.

Les méthodes basées sur les diagrammes de Voronoï ont été initiées par le travail de [Amenta et Bern, 1999]. Ce dernier fonctionne en deux passes après avoir construit le diagramme de Voronoï du nuage de points : une première pour estimer les normales aux points et d'autres caractéristiques locales tandis que la seconde itération permet de récupérer les triangles. La méthode a été améliorée dans [Amenta et al., 2002] et appelée « Cocone ». Avec certaines heuristiques, celle-ci sélectionne les faces des tétraèdres de la triangulation de Delaunay appartenant à la surface. [Boissonnat, 1984] propose une méthode où il sculpte le volume des tétraèdres de la triangulation de Delaunay en supposant que la surface est fermée. Une variante connue est la méthode des α-shapes de [Edelsbrunner et Mucke, 1994] avec les récents α-shapes conformes de [Cazals et al., 2005] et les α-shapes pondérés de [Park et al., 2005]. La méthode des α-shapes considère une sphère de rayon α qui supprime tous les tétraèdres de la triangulation de Delaunay quand elle ne contient aucun point de

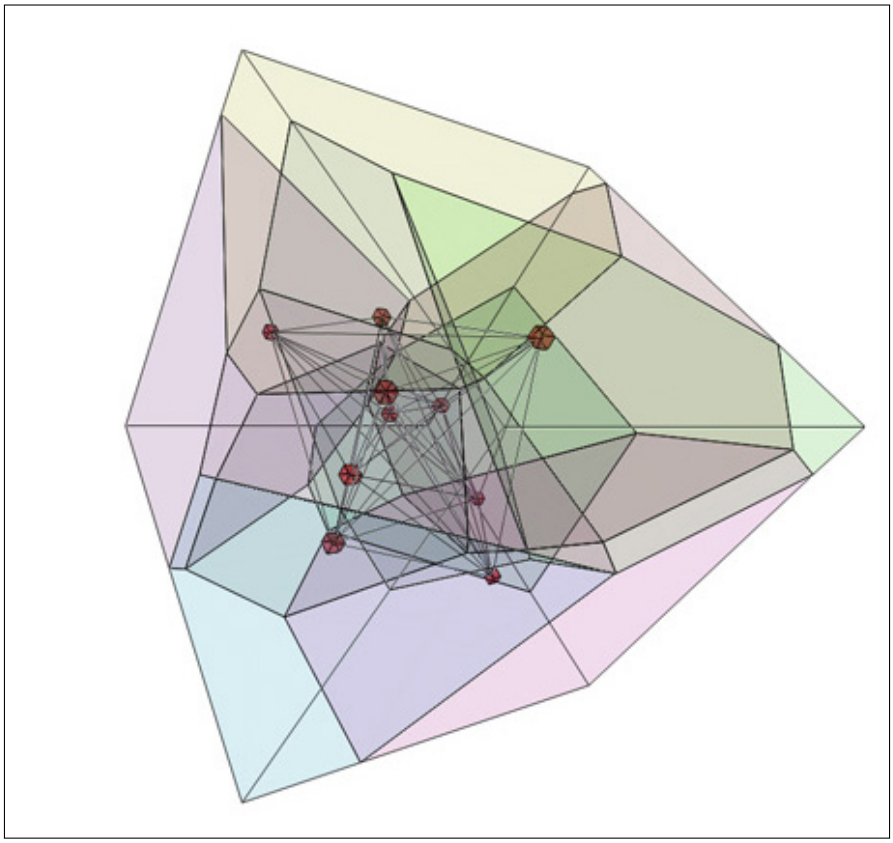


FIGURE 4.18 – Diagramme de Voronoï d'un ensemble de points de $\mathbb{R}^3$.

P. Pour une valeur de $\alpha = 0$, aucun triangle n'est créé et pour une valeur $\alpha = \infty$, nous récupérons l'enveloppe convexe de la triangulation de Delaunay. Pour des objets avec trous, il faut plusieurs expérimentations pour trouver une valeur de α qui donne une surface appropriée.

Les méthodes basées sur les diagrammes de Voronoï offrent généralement de bonnes garanties topologiques sur la surface reconstruite. Par contre, ce sont des méthodes généralement très sensibles au bruit et coûteuses en temps de calcul dû au temps nécessaire pour générer le diagramme de Voronoï.

La plupart des méthodes que nous venons de voir ne permettent pas de reconstruire des surfaces pour de grands volumes de données. Or, les nuages de points provenant de scanners peuvent contenir plusieurs millions de points. L'algorithme BPA de [Bernardini et al., 1999] permet de traiter les nuages de points en « *out-of-core* », c'est-à-dire en chargeant seulement une partie des données en mémoire et en figeant les triangles dans les zones de transition lors de la propagation. « *Super Cocone* » de [Dey et al., 2001] utilise un octree pour subdiviser le nuage et lance l'algorithme de base « *Cocone* » plusieurs fois avec les données d'une cellule de l'octree en zone mémoire pour obtenir la surface finale. Une méthode de [Bolitho et al., 2007] basée sur la méthode de Poisson de [Kazhdan et al., 2006] utilise un flux de données pour être capable de gérer jusqu'à 400 millions de points orientés.

La méthode de [Chang et al., 2009] basée sur les transformations de l'axe médian est aussi capable de produire des maillages à partir de grands volumes de données grâce à un partitionnement de l'espace. Enfin, la méthode de [Buchart et al., 2008] qui est une variante de la méthode de [Gopi et al., 2000] (triangulation par projection locale dans le plan tangent) utilise le GPU pour accélérer les calculs.

4.2.2 Méthode de triangulation proposée

Notre méthode de maillage est basée sur une triangulation dans le plan tangent. Elle est inspirée de [Gopi et al., 2000] et de [Medeiros et al., 2004] (variante 2D de [Bernardini et al., 1999]). Nous allons présenter ici en détail ces deux méthodes.

Méthodes GOPI et BPA

La méthode de [Gopi et al., 2000], que nous appelons GOPI, est basée sur une triangulation locale de Delaunay dans l'espace tangent en chaque point du nuage. Pour cela, en chaque point p_i, un sous-ensemble des points voisins est sélectionné suivant un critère d'échantillonnage basé sur les courbures principales. Ce sous-ensemble des points voisins est ensuite projeté dans l'espace tangent suivant la normale au point p_i. Une triangulation de Delaunay en 2D dans l'espace tangent permet de trouver les points connectés de la triangulation finale. Deux points sont connectés s'ils appartiennent chacun à la triangulation locale de l'autre point. Les triangles créés sont les triangles dont les sommets sont des points connectés deux à deux. [Buchart et al., 2008] a implémenté cet algorithme sur GPU pour accélérer les temps de traitement. La figure 4.19 montre un maillage obtenu par la triangulation GOPI.

GOPI est une très rapide car elle est de complexité linéaire suivant le nombre de points. Mais les maillages construits par la méthode GOPI contiennent souvent des trous. Cela est dû à la construction des triangles. Un triangle est ajouté au maillage seulement si les trois sommets sont deux à deux connectés. Or dans certaines situations (points cocycliques ou presque cocycliques), deux points peuvent ne pas être connectés parce que dans le plan tangent de l'un des points, l'autre point n'est pas dans la triangulation locale en 2D. Le triangle n'est pas alors pas construit laissant un trou indésirable dans le maillage.

La méthode BPA (« *Ball Pivoting algorithm* ») de [Bernardini et al., 1999] construit une triangulation à partir d'un ensemble P de points avec l'information de normales orientées. Un triangle est initialement sélectionné avec la condition suivante : la sphère circonscrite de rayon ρ dont le centre est dans le demi-plan indiqué par l'orientation des normales des 3 sommets ne doit contenir aucun autre point de P. Ensuite la sphère va pivoter suivant les 3 arêtes du triangle comme sur la figure 4.20. Pour une arête v_1v_2, la sphère de rayon ρ pivote jusqu'à rencontrer un nouveau point v_3 pour former le triangle $v_1v_3v_2$. Les arêtes v_1v_3 et v_3v_2 sont ajoutées à la liste des arêtes à explorer. Certaines conditions sont vérifiées lorsque la sphère rencontre un point déjà touché : ceci permet d'éviter de créer une surface non variété. Si la sphère de rayon ρ ne rencontre

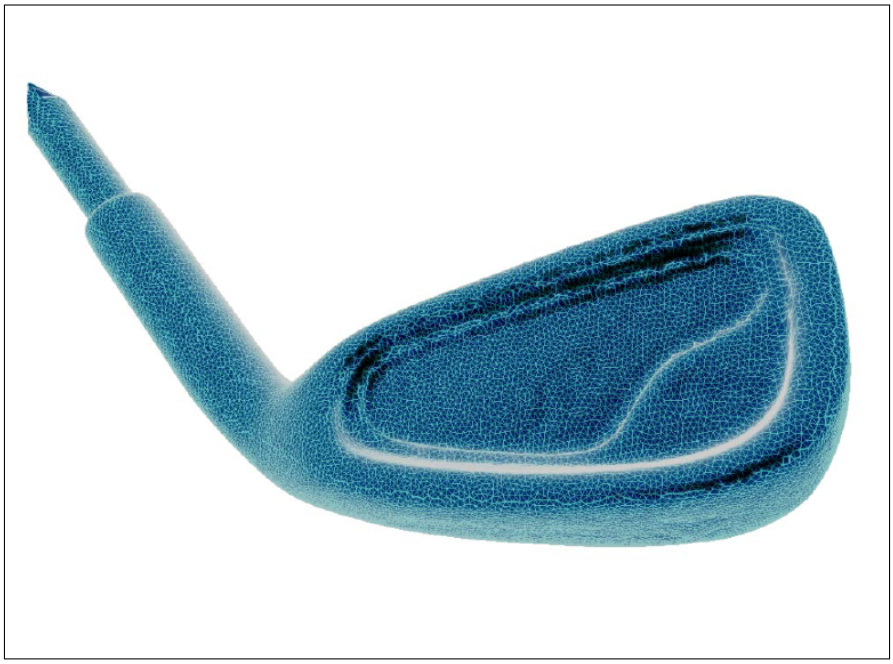


FIGURE 4.19 – Maillage d'un nuage de points de Club de golf par la méthode GOPI (d'après [Gopi et al., 2000]).

pas de point lors de la rotation, alors il n'y a pas de nouveaux triangles, ni de nouvelles arêtes dans la liste à explorer et les deux points de l'arête sont des points de bords. Cet algorithme permet ainsi de créer des composantes connexes donc des surfaces à trous. Même sans garanties théoriques, les surfaces reconstruites au cours de nos expérimentations sont bien des variétés à bords. La figure 4.21 montre un résultat de maillage obtenu par BPA.

Nous avons étudié et analysé l'algorithme BPA, notamment dans le cadre du stage d'Andrea M. Fonger [Fonger, 2009]. Par construction, chaque triangle doit être sur la sphère circonscrite de rayon ρ. Donc le rayon du cercle circonscrit de chaque triangle du maillage est compris entre 0 et ρ. En supposant que le nuage de points ait été décimé avec le paramètre d_{deci}, c'est-à-dire que la distance entre un point et son point le plus proche soit toujours supérieure ou égale à d_{deci}, alors les angles de tous les triangles de la triangulation BPA sont supérieurs à un angle minimal α_{min} qui vérifie la relation : $sin(\alpha_{min}) \geq \frac{d_{deci}}{2r}$ (par la loi des sinus d'un triangle) où r est le rayon du cercle circonscrit du triangle. Sachant que ce rayon est compris entre 0 et ρ, alors $sin(\alpha_{min}) \geq \frac{d_{deci}}{2\rho}$. Avec une décimation des points, la triangulation BPA permet de connaître un minorant des angles de la triangulation. Ceci permet de s'assurer de ne pas construire de triangles trop « allongés ».

BPA est très rapide et de complexité linéaire par rapport au nombre de points du nuage mais a le désavantage d'être très sensible au bruit et à l'échantillonnage des points sur la surface comme la plupart des méthodes de triangulation par interpolation. Un inconvénient majeur de la méthode BPA est de ne pas reconstruire correctement la surface aux points où la courbure est supérieure à $\frac{1}{\rho}$ comme on le voit sur la figure 4.22. En effet, la sphère

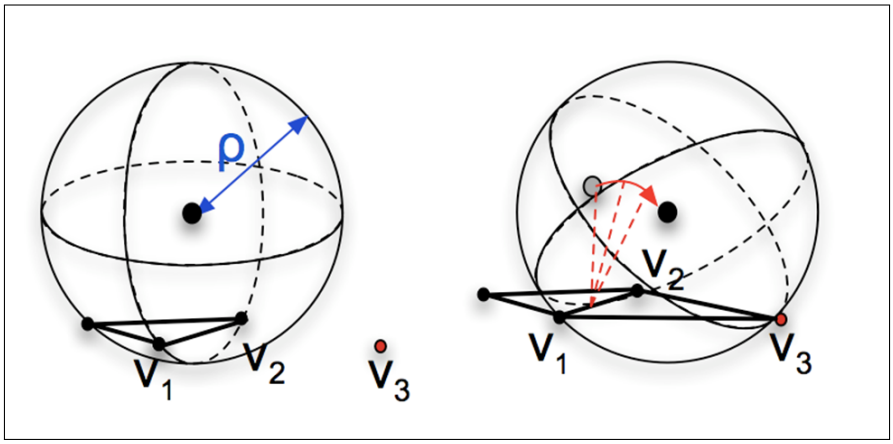


FIGURE 4.20 – Construction des triangles de la méthode BPA par propagation : une sphère de rayon ρ pivote selon une arête du triangle vv_1v_2 pour créer le triangle $v_1v_3v_2$.

de rayon ρ lorsqu'elle pivote ne peut pas atteindre les points où la courbure est supérieure à $\frac{1}{\rho}$.

Ainsi, les deux méthodes de triangulation par interpolation GOPI et BPA sont rapides et bien adaptés pour les nuages de plusieurs millions de points mais la première laisse de nombreux trous dans le maillage tandis que la seconde ne reconstruit pas correctement la surface aux points de courbure supérieure à $\frac{1}{\rho}$. Notre objectif est de produire une triangulation par interpolation, linéaire suivant le nombre de points qui laisse le minimum de trous et qui reconstruise au mieux les arêtes.

[Medeiros et al., 2004] ont proposé une variante en 2D de l'algorithme BPA appelé R-BPA (« *Restricted-Ball Pivoting Algorithm* ») pour générer une triangulation de points en 2D. Le fait d'être en 2D permet de simplifier la méthode : pour construire un triangle à partir d'une arête v_1v_2, il n'est plus nécessaire de faire une rotation d'une sphère de rayon ρ mais il suffit de chercher le point v_3 d'angle $v_1v_3v_2$ maximal. [Medeiros et al., 2004] donnent des garanties théoriques sur la triangulation : il s'agit d'une sous-partie de la triangulation de Delaunay en 2D. Cette triangulation est très rapide puisqu'elle se fait en temps linéaire suivant le nombre de points.

Notre algorithme de maillage

L'idée de notre algorithme de triangulation est de tenir compte des avantages des deux méthodes : BPA et GOPI. Pour cela, nous effectuons comme dans [Gopi et al., 2000] pour chaque p_i une projection des points du voisinage sur le plan tangent. Pour sélectionner les points du voisinage, nous gardons les points inclus dans une sphère de rayon 2ρ où ρ est un paramètre identique à BPA. Nous gardons les points dont la normale fait un angle inférieur à $60\,^\circ$ avec la normale au point p_i. Cela permet de ne pas garder les points qui pourraient appartenir à une autre surface. Nous ne voulons pas décimer les points du voisinage suivant la densité locale ou la courbure comme dans [Gopi et al., 2000] car nous voulons tenir compte des cas où la répartition est localement anisotrope (ce qui est le cas

FIGURE 4.21 – Maillage d'un nuage de points de dragon par la méthode BPA (d'après [Bernardini et al., 1999]).

avec des données provenant de systèmes mobiles) comme sur la figure 4.23.

Phase 1 de construction de la triangulation

Nous projetons les points du voisinage dans le plan tangent. Nous avons ainsi un ensemble de points en 2D et nous pouvons appliquer l'algorithme R-BPA (version 2D de l'algorithme BPA de [Medeiros et al., 2004]) pour calculer rapidement la triangulation locale autour du point p_i. Pour cela, nous commençons par le point le plus proche q_0 de p_i car nous savons que l'arête q_0p_i appartient à la triangulation de Delaunay de P comme expliqué dans [Boissonnat, 1984]. Nous recherchons ensuite parmi les points du voisinage le point q qui maximise l'angle (q_0qp_i) avec une orientation du triangle q_0qp_i identique à la normale en p_i. Cela nous donne le point q_1 comme à gauche de la figure 4.24 et nous continuons avec l'arête q_1p_i et ainsi de suite jusqu'à ne plus trouver de nouveaux points ou retomber sur le point q_0. Si nous retombons sur le point q_0, c'est que nous avons construit une suite de triangles avec les points $(q_0, q_1, q_2, q_3, ..., q_n, q_0)$ qui forment un cycle complet autour du point p_i comme au milieu de la figure 4.24. Si à un moment, nous ne trouvons pas de point d'angle maximum, alors nous repartons du point q_0 et nous allons dans l'autre sens

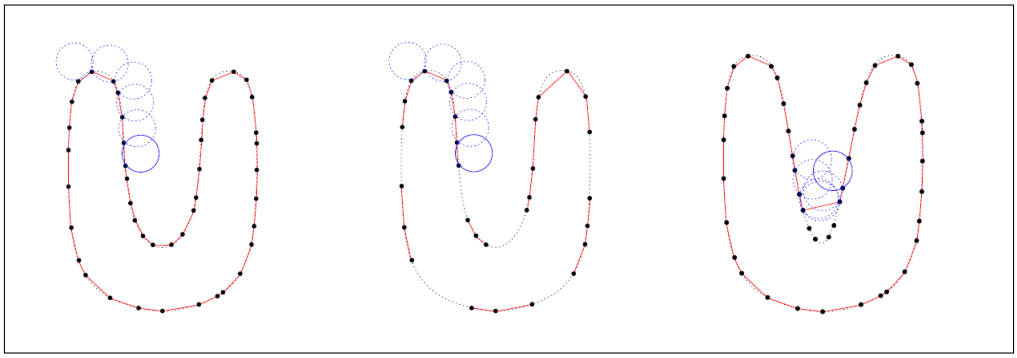


FIGURE 4.22 – Principe du Ball Pivoting Algorithm avec à droite un cas où la reconstruction ne correspond pas à la surface (d'après [Bernardini et al., 1999]).

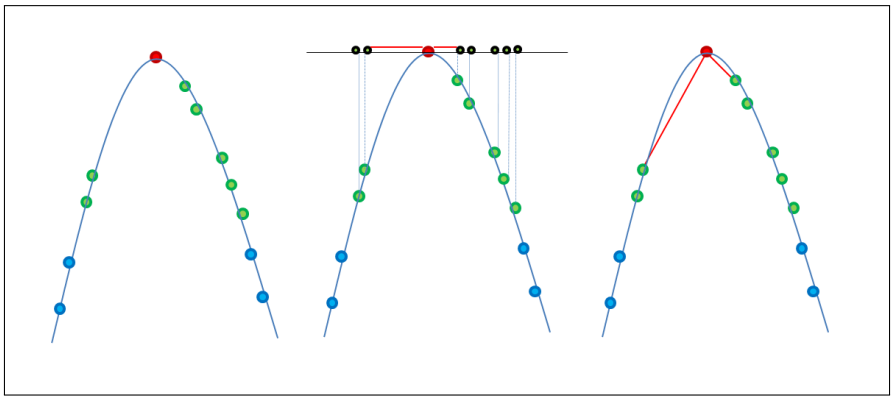


FIGURE 4.23 – Principe de projection dans le plan local pour la recherche de points voisins dans la triangulation.

pour trouver les points $(q_0, q'_1, q'_2, ...q'_n)$. Les triangles autour du point p_i forment alors un demi-cycle comme sur la figure 4.24.

Une fois trouvés les triangles autour de p_i dans le plan tangent en 2D, nous considérons directement que ces triangles sont valides en 3D, c'est-à-dire que ce sont les mêmes points connectés en 3D. Cette étape est différente de [Gopi et al., 2000] qui construit les triangles uniquement lorsque trois sommets sont connectés deux à deux, c'est-à-dire une fois les triangulations locales de chacun des points calculées et que chacun des points se trouve dans la triangulation 2D des deux autres points. Dans notre cas, nous n'avons aucune garantie théorique sur la validité des triangles construits, nous construisons directement les triangles dès qu'ils ont été trouvés dans la triangulation locale. Notre maillage dans cette première étape n'est donc pas une variété et peut contenir trois fois le même triangle (si les points sont connectés deux à deux).

Phase 2 de suppression de triangles pour obtenir un maillage variété

Premièrement, pour ne pas avoir deux triangles identiques, nous conservons après la

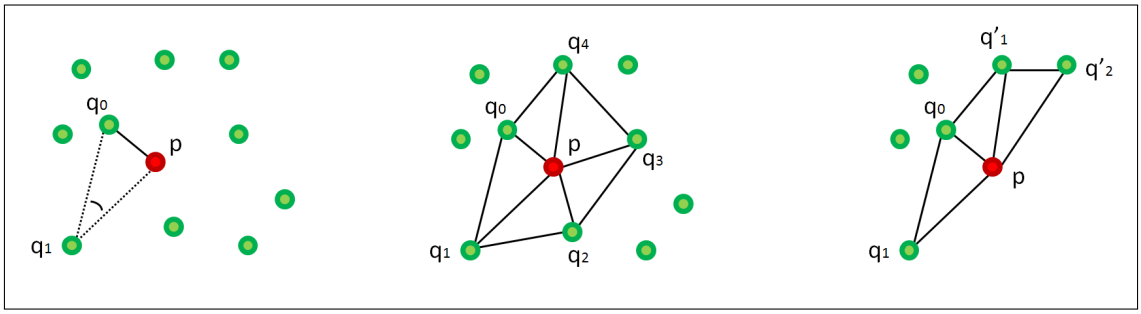


FIGURE 4.24 – Principe de triangulation en 2D avec R-BPA autour d'un point p_i dans le plan tangent : à gauche, sélection du point q_1 d'angle (q_0q_1p) maximal, au milieu, la triangulation terminée avec un cycle $(q_0, q_1, q_2, q_3, q_4, q_0)$ et une autre configuration avec une triangulation terminée par un demi-cycle $(q_0, q_1; q_0, q'_1, q'_2)$.

triangulation en 2D un triangle en 3D uniquement si le point p_i a un indice (par exemple un indice de stockage permettant de différencier les points) supérieur aux indices des deux autres points. Un triangle orienté peut être représenté par trois combinaisons : $p1p2p3$, $p2p3p1$ et $p3p1p2$, nous choisissons celle qui commence par l'indice le plus grand. Par exemple, dans le schéma de droite de la figure 4.24, nous avons en 2D, par R-BPA, les triangles pq_0q_1, pq'_1q_0 et $pq'_2q'_1$. Si les points sont stockés dans un tableau avec comme indices : $p = 15$, $q_0 = 5$, $q_1 = 12$, $q'_1 = 200$ et $q'_2 = 7$, alors nous conservons en 3D seulement le triangle pq_0q_1 car l'indice de p est supérieur aux indices de q_0 et de q_1.

Deuxièmement, nous vérifions que le rayon du cercle circonscrit du triangle en 3D est bien inférieur à ρ pour ne pas construire de triangles allongés et ainsi obtenir la même condition que BPA, c'est-à-dire un minimum des angles de la triangulation finale avec $sin(\alpha_{min}) \geq \frac{d_{deci}}{2\rho}$ après une décimation de paramètre d_{deci}.

Notre méthode de triangulation produit des maillages qui ne sont pas des variétés. Par exemple, lors de la projection dans le plan tangent d'un point p, il est possible de construire le triangle pq_0q_1. Mais lors de la projection dans le plan tangent de q_0, il est possible de construire le triangle q_0q_1q. Ces deux triangles ont une arête orientée commune q_0p_1 ce qui est incompatible avec la définition de variété. Nous avons donc une phase où nous supprimons certains triangles de la triangulation obtenue pour la rendre variété. Pour cela, nous testons tout d'abord chaque arête de la triangulation. Une arête d'un triangle a un sens donné par l'orientation de son triangle. Un triangle $p_1p_2p_3$ est orienté vers $(p_2 - p_1) \wedge (p_3 - p_1)$ et possède les arêtes p_1p_2, p_2p_3 et p_3p_1. Si une arête est partagée par au moins deux triangles alors seul le triangle qui maximise son angle minimal est conservé. Ensuite, chaque point p est testé pour savoir si les triangles qui ont pour sommet p forment bien un cycle ou un demi-cycle. Si ce n'est pas le cas, alors le cycle ou le demi-cycle contenant le plus de triangles est conservé. Par cette méthode, nous construisons une triangulation de la surface qui est variété mais qui contient certains trous non souhaités.

Phase 3 de remplissage des trous du maillage

Nous avons enfin une dernière phase pour remplir les trous laissés par la phase 1 et les trous générés par la phase 2 de suppression de triangles. Pour détecter les trous, nous construisons les suites fermées de 3 ou 4 arêtes. Les suites fermées de 3 arêtes sont de la forme (p_1p_2, p_2p_3, p_3p_1) où p_ip_j n'est pas l'arête d'un triangle et p_jp_i est l'arête d'un triangle. Les suites fermées de 4 arêtes sont de la forme $(p_1p_2, p_2p_3, p_3p_4, p_4p_1)$ avec les mêmes conditions sur p_ip_j. Nous ajoutons alors le triangle $p_1p_2p_3$ à la triangulation pour les suites de 3 arêtes et les deux triangles qui maximisent leur angle minimal pour les suites de 4 arêtes.

Pour boucher les trous de plus grande longueur ainsi que les trous des bords, nous avons mis en place un algorithme itératif qui teste tous les demi-cycles de le triangulation. Prenons comme exemple le demi-cycle $(q_0, q_1; q_0, q'_1, q'_2)$ de la figure 4.24. Si l'angle $(q_1pq'_2)$ est inférieur à $60\,^\circ$ et le rayon du cercle circonscrit du triangle $pq_1q'_2$ est inférieur à ρ, alors nous ajoutons le triangle $pq_1q'_2$ au maillage et testons de nouveau les 2 autres sommets q_1 et q'_2 pour voir si de nouveaux triangles peuvent être ajoutés.

Pseudo-code

Pour la méthode de triangulation, nous avons besoin en entrée d'un nuage de points relativement dense pour ne pas créer de trous dans le modèle. Nous avons aussi besoin d'un nuage décimé (méthode décrite au chapitre 3), des normales en chaque point (par exemple avec la méthode FMLS décrite dans le chapitre 3) et de données débruitées (par exemple avec la méthode NLD3D du chapitre 3). En sortie, nous obtenons un ensemble de triangles représentant le maillage de la surface.

L'algorithme 2 est le pseudo-code de notre méthode de triangulation décrite dans la section précédente.

4.2.3 Analyse et comparaison des résultats sur des données simulées et réelles

Nous avons deux hypothèses pour notre méthode de triangulation. La distance entre un point et son voisin le plus proche doit être supérieure à d_{deci}. Cela garantit de ne pas créer des triangles avec des angles trop petits. De même, pour qu'un point soit connecté à un autre, il faut que leur distance soit inférieure à 2ρ et dans ce cas, la triangulation recouvre toute la surface. Nous avons donc un seul paramètre pour notre méthode de triangulation : $d_{maillage} = 2\rho$ pour définir la distance maximale reliant deux points. Cette distance définit alors la densité minimale du nuage de point nécessaire pour une reconstruction correcte : $\rho_{density} \geq \frac{1}{d^2_{maillage}}$.

Aucune garantie théorique ne permet de certifier que le maillage obtenu par cette méthode est un maillage variété mais dans tous les tests que nous avons effectués, cette condition a été vérifiée. De manière générale, l'idée de cette méthode est de construire plus de triangles que les méthodes classiques pour ensuite faire des ajustements locaux par suppression et addition de triangles sur des heuristiques simples. Nous allons voir par la suite que les résultats sont meilleurs que ceux des méthodes classiques sur de nombreux exemples.

Un article sur cette méthode de triangulation est en cours de préparation.

Données simulées

Nous avons testé notre algorithme de triangulation sur des données simulées et réelles et avons comparé les résultats obtenus avec BPA de [Bernardini et al., 1999], GOPI de [Gopi et al., 2000], la méthode Poisson de [Kazhdan et al., 2006] et RIMLS de [Oztireli et al., 2009]. Pour toutes ces méthodes, nous avons utilisé les mêmes données d'entrée, c'est-à-dire des nuages de points décimés, débruités et dont les normales ont été calculées avec notre méthode FMLS du chapitre 3. Nous avons testé les nuages de points simulés *S_Fandisk*, *S_Horse*, *S_Engine* et *S_Hand*. Le tableau 4.6 présente les résultats pour les différents algorithmes avec le nombre de triangles obtenus et les temps de calcul. Les maillages ont été produits avec un QuadCore Q9300 2.50 GHz, 2 Go de RAM, sur un seul *thread* et sur CPU.

Pour notre méthode, nous avons pris pour chaque nuage de points le paramètre $d_{maillage} = 10d_{mean}$ où d_{mean} est la distance moyenne entre un point et son point le plus proche. Tous les maillages sont testés pour vérifier qu'ils sont bien variétés.

	S_Fandisk	*S_Horse*	*S_Engine*	*S_Hand*
Taille	6 475 points	100 000 points	11 246 points	327 323 points
BPA	12 583 triangles (0.4 s)	103 595 triangles (27.9 s)	22 124 triangles (1.3 s)	542 437 triangles (470.0 s)
GOPI	12 526 triangles (0.3 s)	166 895 triangles (2.3 s)	22 108 triangles (0.6 s)	532 255 triangles (46.2 s)
RIMLS	144 368 triangles (9.1 s)	491 337 triangles (89.4 s)	151 019 triangles (7.5 s)	-
Poisson	13 282 triangles (1.4 s)	13 282 triangles (9.7 s)	10 514 triangles (1.1 s)	-
Notre méthode	12 906 triangles (0.8 s)	133 979 triangles (27.0 s)	22 074 triangles (1.9 s)	559 505 triangles (407.2 s)

TABLE 4.6 – Résultats de notre méthode triangulation et comparaison avec d'autres méthodes.

La figure 4.25 présente les maillages des méthodes BPA, GOPI, RIMLS et Poisson pour le nuage *S_Fandisk*. Pour BPA, il y a peu ou pas de trous mais les zones de fortes courbures et surtout les arêtes ne sont pas correctement reconstruites (encerclés en rouge). Pour GOPI, nous observons de nombreux trous, surtout dans les zones où la répartition des points est très uniforme comme sur le dessus du nuage *S_Fandisk*. Les méthodes d'approximation RIMLS et Poisson n'arrivent pas à reconstruire correctement les arêtes de la surface et dans le cas de RIMLS nous obtenons trop de triangles par rapport à la taille du maillage (environ 140 000 triangles pour environ 6 000 points). Notre méthode de triangulation ne laisse pas de trous et reconstruit bien la surface au niveau des arêtes. En ce qui concerne les temps de traitement, GOPI est la méthode la plus rapide avec 0.3 s, notre triangulation s'opère en 0.8 s alors que RIMLS est très lente avec 9.1 s pour traiter environ 6 000 points comme le montre le tableau 4.6.

La figure 4.26 montre les maillages pour le modèle *S_Horse*. Ce nuage de points est un nuage non dense et nous voyons que les méthodes par interpolation (BPA, GOPI et

notre méthode) ne reconstruisent pas correctement la surface. En revanche, les méthodes d'approximation comme RIMLS et Poisson donnent de bien meilleurs résultats. C'est la méthode Poisson qui s'approche le plus de la surface réelle en construisant un maillage fermé.

La figure 4.27 donne les résultats de triangulation des 5 méthodes exposées plus haut pour un nuage de points de moteur *S_Engine*. Ce nuage de point a la particularité d'avoir été généré à partir d'une surface non fermée. Les méthodes BPA et notre méthode donnent les meilleurs résultats. RIMLS n'arrive à retrouver correctement les bords, Poisson donne un maillage fermé qui ne correspond pas à la surface réelle et GOPI laisse des trous à cause de la répartition uniforme des points du nuage.

La figure 4.28 présente la triangulation d'un nuage de points de plus grande taille *S_Hand* comportant environ 300 000 points. On constate que notre méthode est nettement plus lente que GOPI (à cause des phases 2 et 3 de suppression de triangles et de remplissage des trous) mais est équivalente en temps à BPA. A l'inverse, on constate visuellement que notre méthode produit un maillage incontestablement de meilleur qualité car notre méthode arrive à reconstruire la surface sans trous et avec des arêtes vives.

Données réelles

Nous avons aussi testé notre méthode de modélisation sur des données réelles. Le tableau 4.7 résume les résultats de la triangulation. Les figures 4.29, 4.30, 4.31 et 4.32 illustrent respectivement les maillages des nuages *CYBERWARE_Bunny*, *LARA3D_Orsay*, *LARA3D_Etable* et *FX_GX_Orsay*.

	CYBERWARE_Bunny	*LARA3D_Orsay*	*LARA3D_Etable*	*FX_GX_Orsay*
Taille	35 947 points	2 440 416 points	3 540 911 points	1 804 648 points
Notre méthode	70 505 triangles (70.9 *s*)	3 139 141 triangles (490 *s*)	1 330 836 triangles (68.1 *s*)	3 387 480 triangles (488.3 *s*)

TABLE 4.7 – Résultats de notre méthode de triangulation sur des données réelles.

4.2.4 Conclusion sur la méthode de triangulation proposée

Nous avons proposé dans cette section un algorithme de triangulation par interpolation de points. Cette méthode est rapide, de complexité linéaire suivant le nombre de points et donne de meilleurs résultats que les méthodes GOPI et BPA tant du point de vue des données simulées que des données réelles de LiDAR. Nous obtenons surtout de très bonnes reconstructions de la surface à proximité des points de discontinuité.

Algorithme 2 Méthode de triangulation

Début
  { *Phase 1 de construction de la triangulation* T_{3D} }
  PourChaque point p **De** Nuage **Faire**
    Projection dans le plan tangent des points p_j dont $d(p_j, p) < 2\rho$ ET $n_j \cdot n_i > cos(\frac{\pi}{3})$
    Triangulation locale T_{2D} par R-BPA autour du point p
    PourChaque triangle $t = p_1p_2p_3$ **De** T_{2D} **Faire**
      $T_{3D} \leftarrow t$
{ *Phase 2 de suppression de triangles pour obtenir un maillage variété* }
PourChaque triangle $p_1p_2p_3$ **De** T_{3D} **Faire**
  { *Soit* r *le rayon du cercle circonscrit du triangle* $p_1p_2p_3$ }
  Si $r > \rho$ OU $p_1 < p_2$ OU $p_1 < p_3$ **Alors**
    supprimer $p_1p_2p_3$ de T_{3D}
  Si les arêtes p_1p_2, p_2p_3 ou p_3p_1 sont déjà utilisées par un autre triangle **Alors**
    Supprimer le triangle qui a le plus petit angle
PourChaque point p **De** Nuage **Faire**
  Conserver les triangles du plus grand demi-cyle ou cycle de triangles autour de p
{ *Phase 3 de remplissage des trous du maillage* }
PourChaque suite (p_1p_2, p_2p_3, p_3p_1) où p_ip_j n'est pas l'arête d'un triangle et p_jp_i est l'arête d'un triangle **De** triangulation T_{3D} **Faire**
  { *Soit* r *le rayon du cercle circonscrit du triangle* $p_1p_2p_3$ }
  Si $r < \rho$ **Alors**
    $T_{3D} \leftarrow p_1p_2p_3$
{ *Idem pour les suites* $(p_1p_2, p_2p_3, p_3p_4, p_4p_1)$ *avec les mêmes conditions sur* p_ip_j }
PourChaque demi-cycle **De** triangulation T_{3D} **Faire**
  { *Soit* $p_1p_2p_3$ *le triangle manquant dans le demi-cycle autour de* p_1 }
  { *Soit* r *le rayon du cercle circonscrit du triangle* $p_1p_2p_3$ }
  Si $Angle(p_2p_1p_3) < 60\,^\circ$ ET $r < \rho$ **Alors**
    $T_{3D} \leftarrow p_1p_2p_3$
    Tester les cycles ou demi-cycles des points p_2 et p_3
Fin

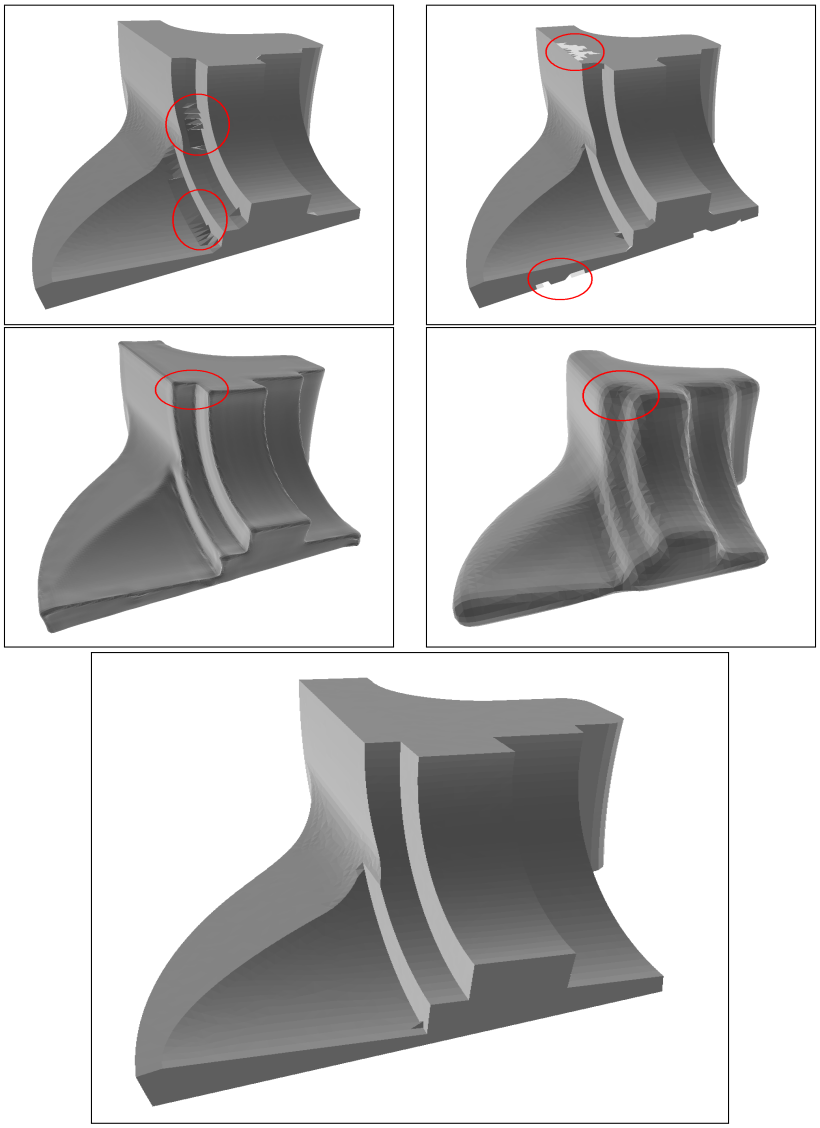


FIGURE 4.25 – Maillage du nuage de points *S_Fandisk* avec différentes méthodes (en haut à gauche BPA, en haut à droite GOPI, en bas à gauche RIMLS et en bas à droite Poisson et tout en bas notre méthode).

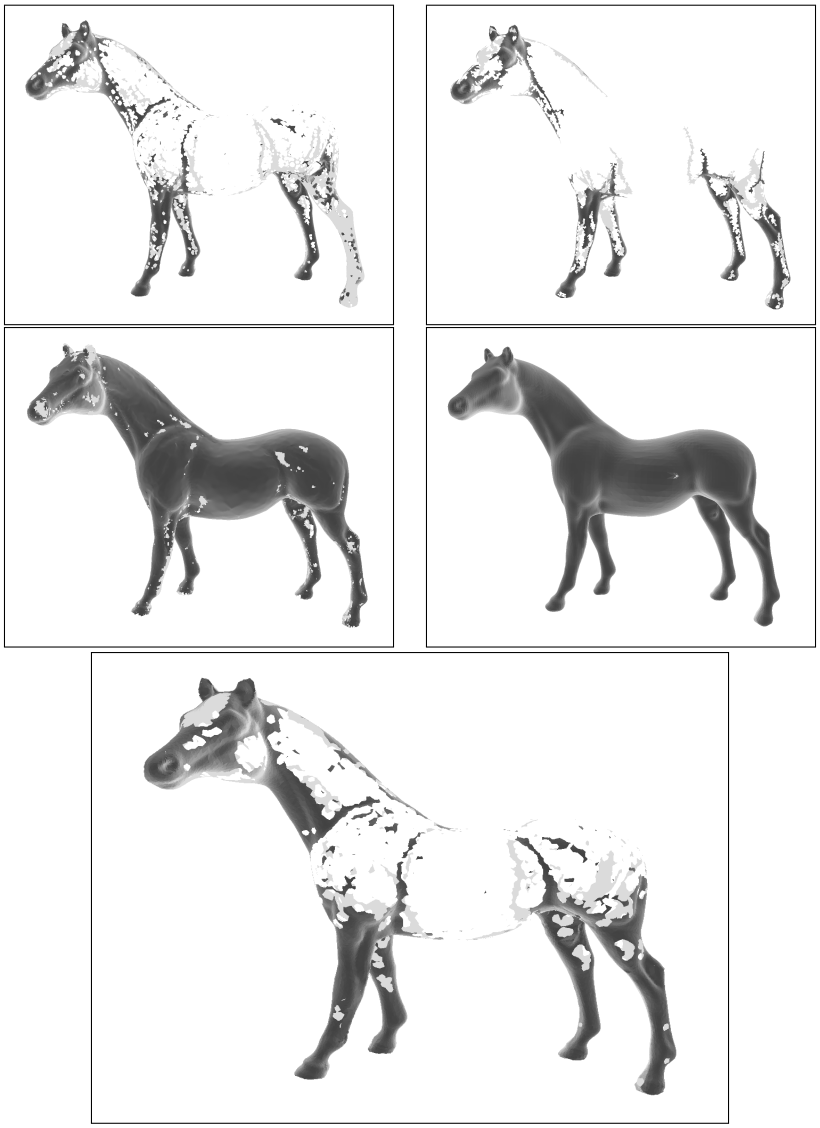


FIGURE 4.26 – Maillage du nuage de points *S_Horse* avec différentes méthodes (en haut à gauche BPA, en haut à droite GOPI, en bas à gauche RIMLS et en bas à droite Poisson et tout en bas notre méthode).

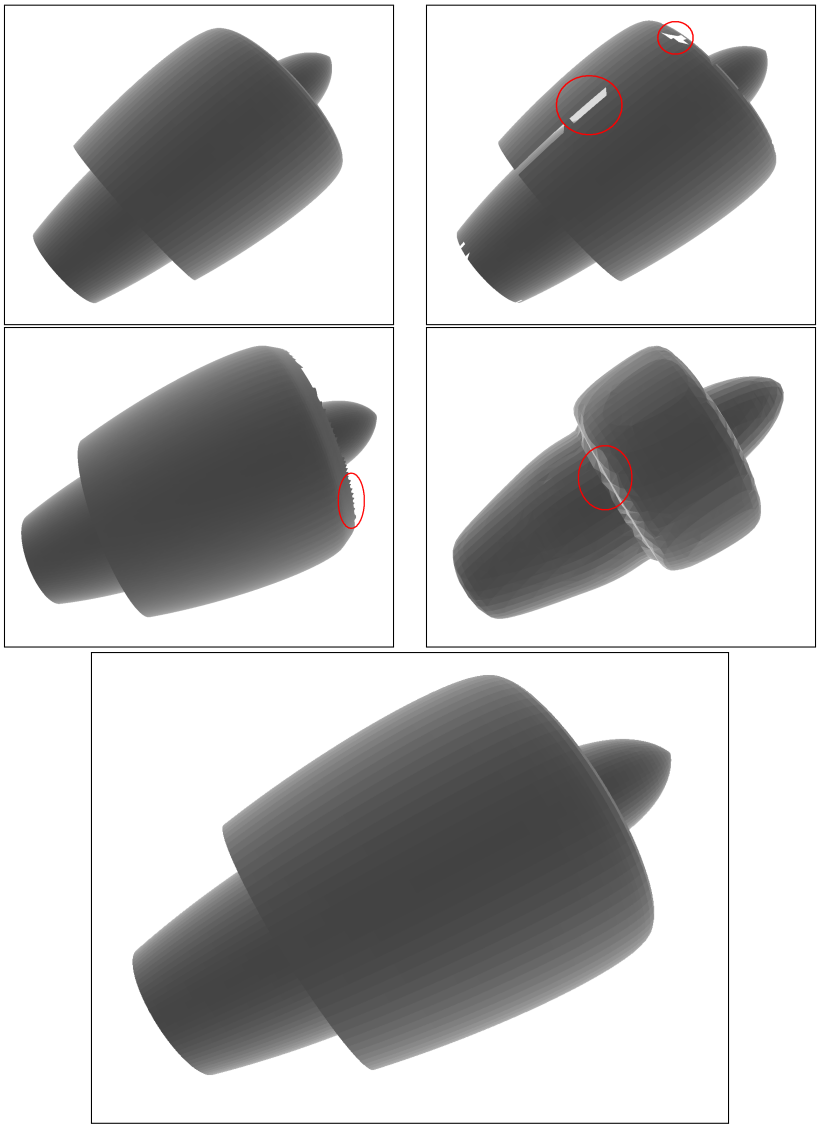


FIGURE 4.27 – Maillage du nuage de points *S_Engine* avec différentes méthodes (en haut à gauche BPA, en haut à droite GOPI, en bas à gauche RIMLS et en bas à droite Poisson et tout en bas notre méthode).

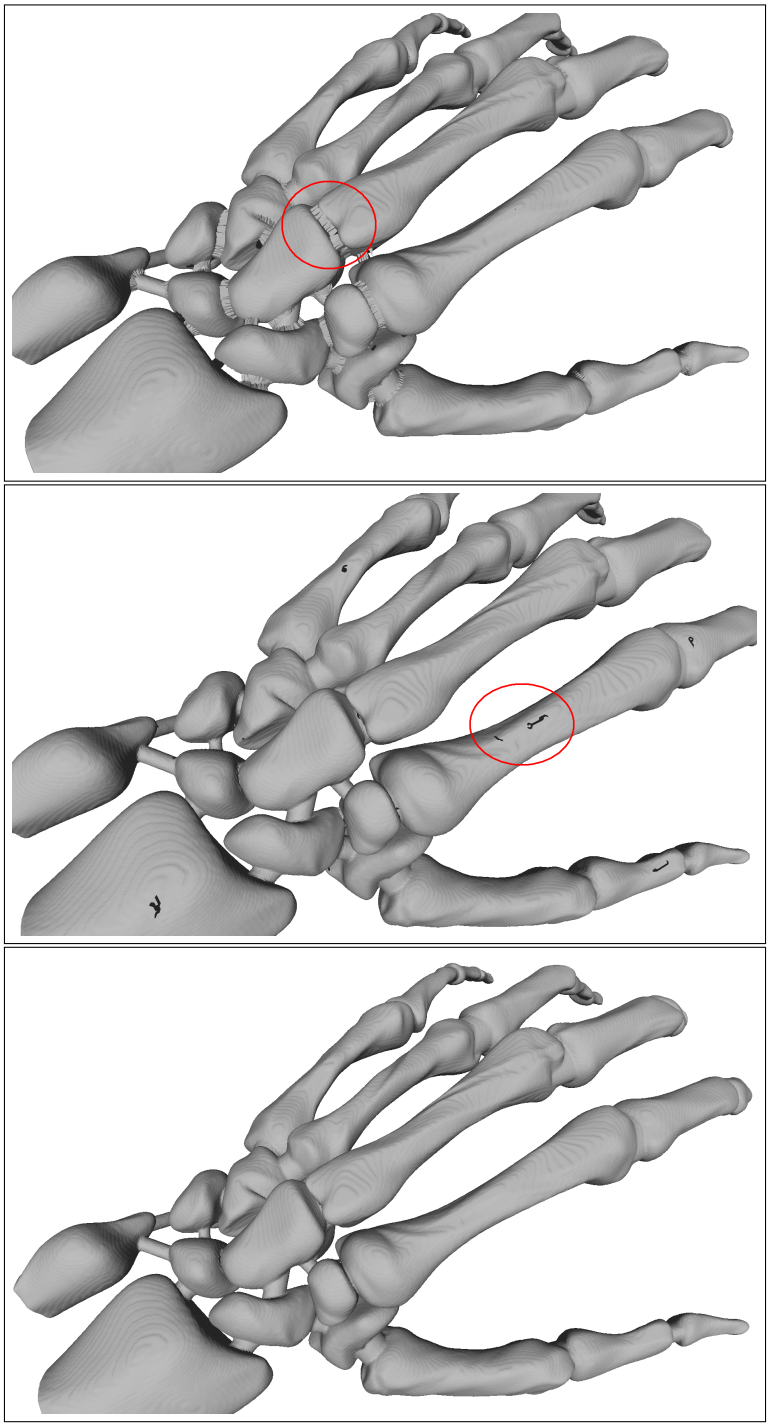


FIGURE 4.28 – Maillage du nuage de points *S_Hand* avec différentes méthodes (en haut BPA, au milieu GOPI et en bas notre méthode de triangulation).

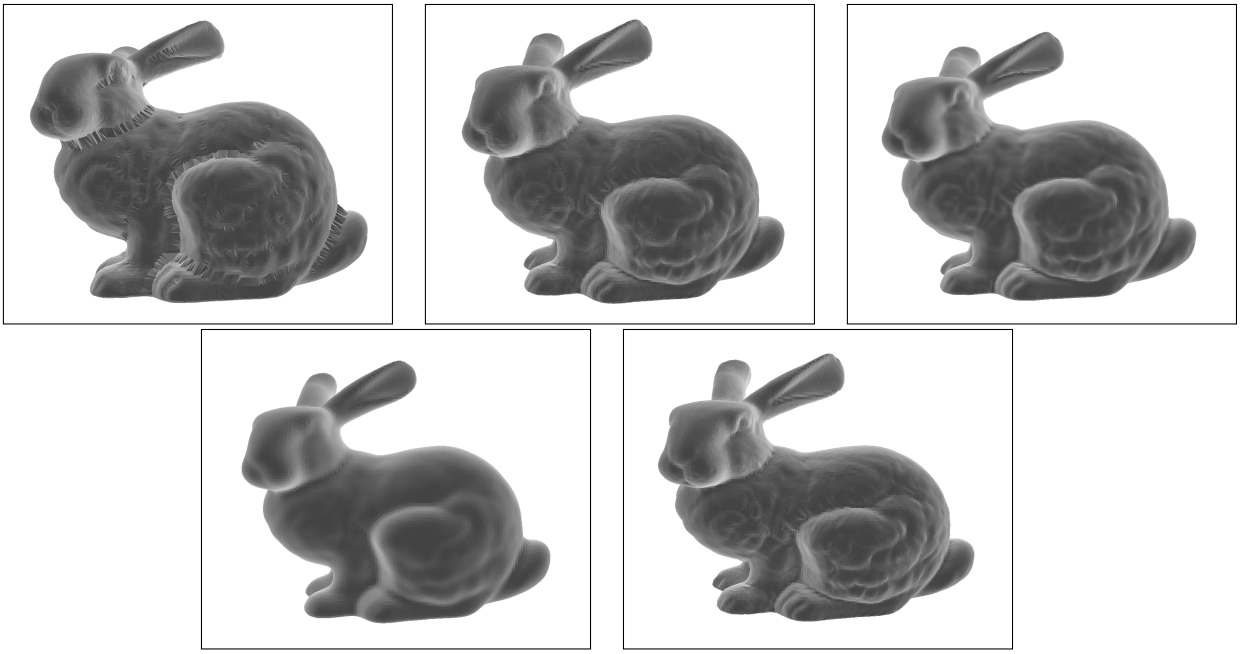


FIGURE 4.29 – Maillage du nuage de points *CYBERWARE_Bunny* avec différentes méthodes (en haut à gauche BPA, en haut au milieu GOPI, en haut à droite RIMLS, en bas à gauche Poisson et enfin en bas à droite notre méthode de triangulation).

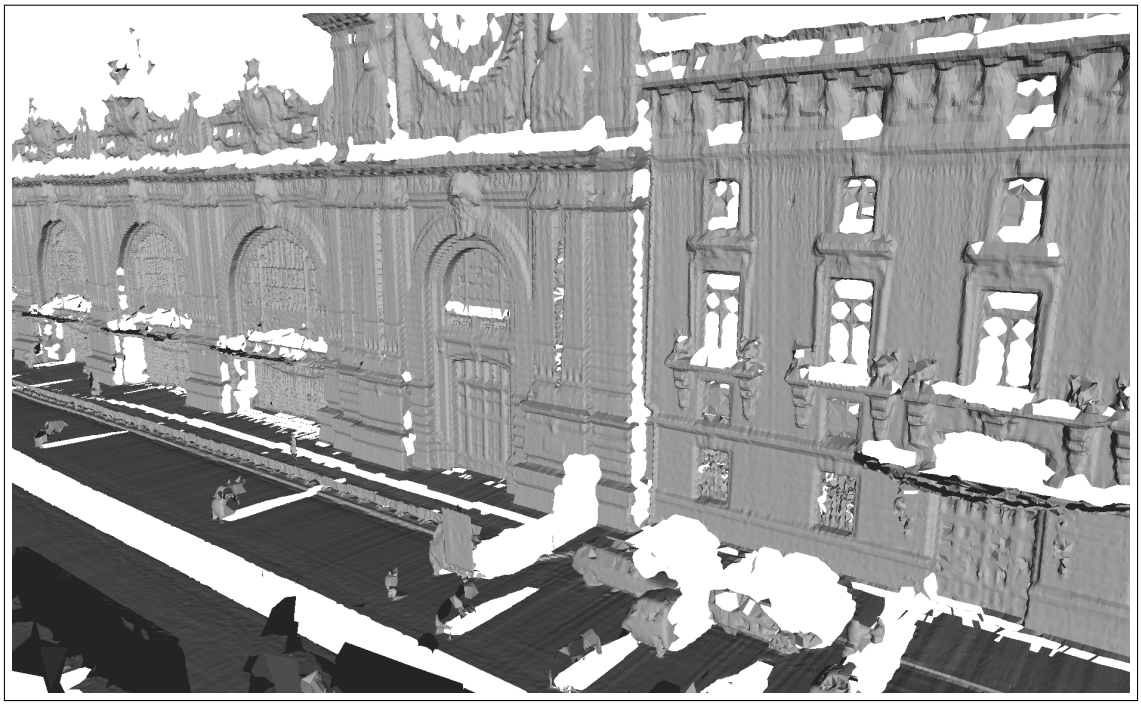


FIGURE 4.30 – Maillage du nuage de points *LARA3D_Orsay* avec notre méthode de triangulation.

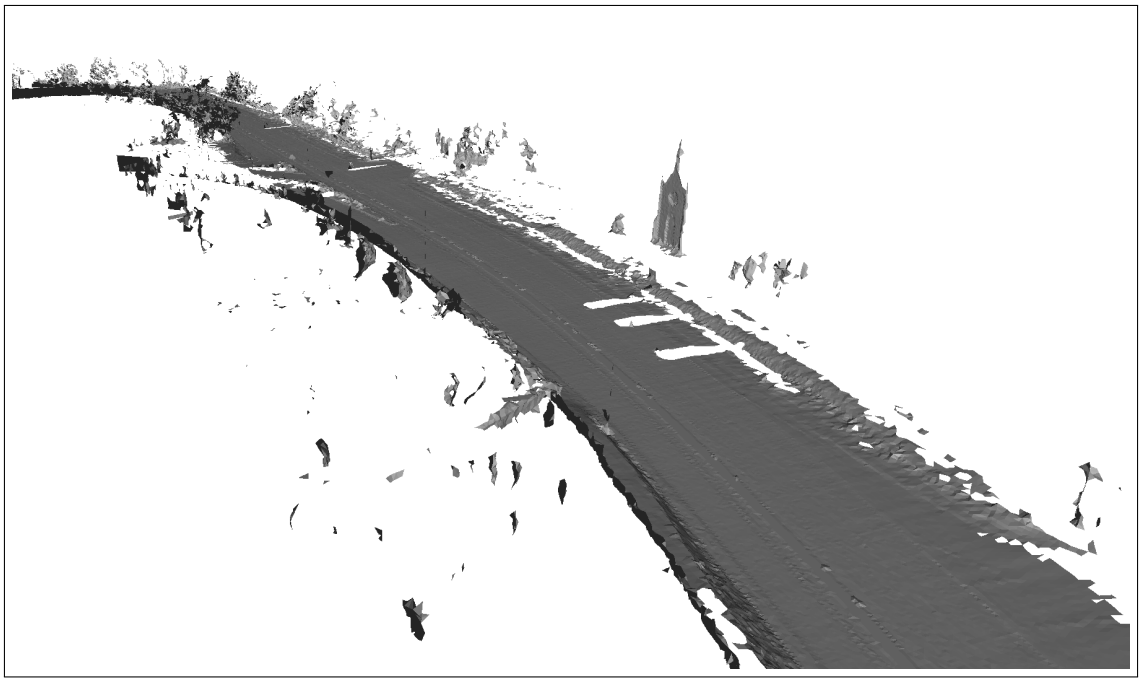


FIGURE 4.31 – Maillage du nuage de points *LARA3D_Etable* avec notre méthode de triangulation.

FIGURE 4.32 – Maillage du nuage de points *FX_GX_Orsay* avec en haut un gros plan sur le maillage de différentes méthodes (à gauche BPA, au milieu GOPI et à droite notre méthode de triangulation) et en bas une vue large du modèle triangulé par notre méthode.

Chapitre 5

Colorisation et Texturation

Résumé

Ce chapitre présente l'ajout de la couleur à l'information de géométrie apportée par les capteurs LiDAR. Nous voyons dans un premier temps la création de nuages de points colorés (nuages de points de LiDAR avec l'information de couleur des caméras) et la création de modèles 3D texturés (modèles 3D créés à partir de nuages de points et textures calculées à partir des images des caméras).

5.1 Etalonnages

5.1.1 Caméras et optiques choisis

Notre plateforme mobile de cartographie accueille plusieurs caméras pour capturer une information visuelle des environnements traversés par le véhicule. Pour couvrir une zone maximale tout en limitant le nombre de caméras, nous avons étudié plusieurs possibilités et avons finalement opté pour la configuration suivante : une caméra vers l'avant avec objectif grand angle pour filmer la zone avant et une caméra filmant chaque côté avec objectif fish-eye pour une capture très grand angle. Les objectifs fish-eye sont des lentilles spécifiques utilisant des principes optiques différents des lentilles classiques et permettant d'obtenir des longueurs de focales très courtes et ainsi de capturer la moitié de l'espace avec une seule caméra. Pour couvrir la même zone que le LiDAR, seulement deux caméras avec objectif fish-eye sont nécessaires, alors qu'un objectif classique aurait nécessité au moins quatre caméras. En effet, une caméra fish-eye peut couvrir une zone jusqu'à 180° dans toutes les directions. La figure 5.1 montre deux versions de la plateforme de LARA3D avec en 2007 une caméra Marlin d'AVT et en 2010 un appareil photo Canon EOS 5D.

FIGURE 5.1 – A gauche plateforme de la version 2007 de LARA3D avec une caméra Marlin d'AVT et un scanner IBEO LD. A droite la plateforme de la version 2010 avec un appareil photo Canon EOS 5D et un scanner SICK LMS 221.

Nous avons testé et utilisé différents modèles de caméras. Les premiers résultats ont été obtenus avec une caméra Marlin F046C avec une résolution de 776x580 et une fréquence de 30 Hz (image visible à gauche de la Figure 5.2). Pour augmenter la résolution des images, nous sommes ensuite passés à une caméra plus haute résolution Pike F421C de résolution 2048x2048 (image visible au milieu de la Figure 5.2). Mais augmenter en résolution a un coût sur la fréquence d'acquisition. Avec la caméra Pike, nous sommes descendus à une fréquence de 5 Hz ce qui nous a obligé à revoir nos méthodes de traitement des images pour la colorisation et la texturation comme nous le verrons dans la suite. Les caméras Marlin et Pike sont des caméras professionnelles, conçues pour être pilotées de l'ordinateur. L'horodatage des images est très précis avec une précision inférieure à la milliseconde.

Dans un troisième temps, nous avons voulu augmenter la qualité des images tout en

diminuant le coût. Pour cela, nous avons choisi de diminuer la fréquence des images en remplaçant les caméras par des appareils photos. Nous verrons plus loin dans ce chapitre que nous n'avons pas besoin de 30 images par seconde pour effectuer la colorisation et la texturation. C'est pourquoi nous avons installé un appareil photo Canon EOS 5D de résolution 5616x3744 avec objectif fish-eye Canon sur la plateforme LARA3D (image sur la droite de la Figure 5.2) avec un coût divisé par 2 par rapport à la caméra Pike précédente. Le fait d'utiliser un appareil photo grand public à la place d'une caméra industrielle complique singulièrement le procédé. En effet, les appareils photos grand public ne sont pas pilotables automatiquement depuis un ordinateur. Canon nous a gracieusement donné l'accès dans le cadre de ces recherches à un SDK permettant de prendre automatiquement des photos depuis un ordinateur. Nous avons ainsi mis en place un système permettant de prendre des photos à une cadence fixe de 1 Hz avec une précision de l'horodatage des images inférieure à 5 ms, ce qui était suffisant dans le cadre de nos travaux.

FIGURE 5.2 – A gauche image de la caméra Marlin d'AVT. Au milieu image de la caméra Pike d'AVT. A droite, image de l'appareil photo Canon (toutes ces caméras sont équipées d'un objectif fish-eye).

Pour coloriser et texturer l'intégralité de nos modèles 3D, les images doivent se chevaucher entre deux prises successives. Or la distance entre prises va dépendre de la fréquence de la caméra et de la vitesse du véhicule. Pour une caméra Marlin à 30 Hz, cela ne pose pas de limitation en terme de vitesse du véhicule et nous disposons d'une grande redondance photogrammétrique. A une fréquence de 5 Hz pour les caméras Pike et 1 Hz pour les appareils photos Canon, la vitesse du véhicule doit être limitée pour permettre un chevauchement suffisant des images ainsi que l'illustre les deux prises de vue de la figure 5.3.

5.1.2 Etalonnage de la caméra

Nous avons équipé nos caméras d'un objectif fish-eye dans le but d'avoir un grand angle d'ouverture. En conséquence, le modèle standard de projection perspective ne peut plus être utilisé, nous avons besoin d'un autre modèle de projection. Le modèle de projection standard pour une optique classique est la projection perspective avec $r = ftan\theta$ où θ est l'angle entre l'axe principal et le rayon incident et où r est la distance entre le projeté du rayon incident sur l'image et le centre de l'image. Les optiques fish-eye utilisent généralement un autre modèle, le modèle équidistant avec $r = f\theta$. La figure 5.4 montre la

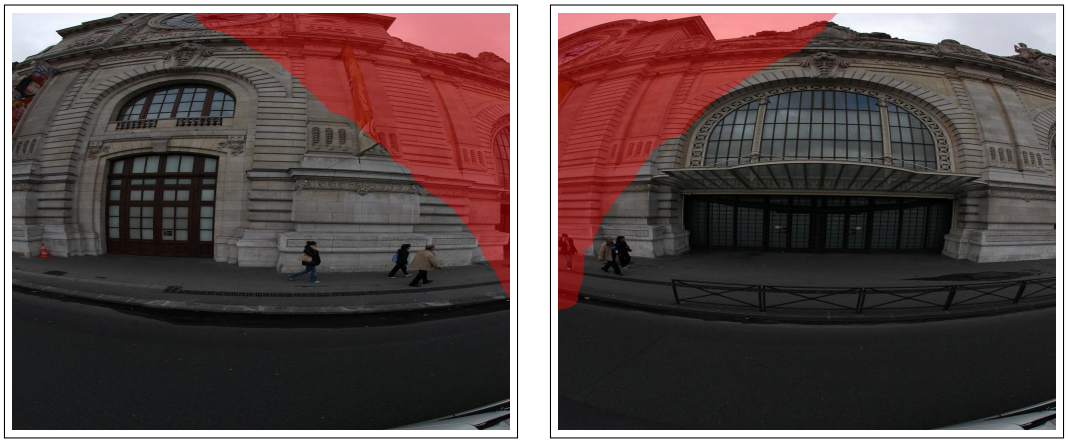


FIGURE 5.3 – Deux prises successives d'images avec l'appareil photo Canon EOS 5D photographiant la gauche du véhicule roulant à 10 km/h (les zones en rouge sont les parties communes des deux images).

différence de projection entre ces deux modèles.

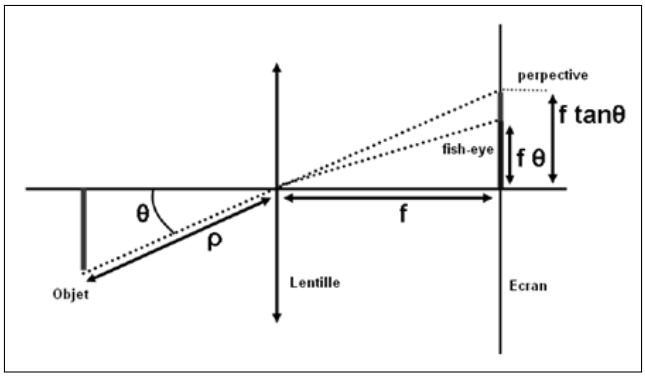


FIGURE 5.4 – Projection perspective pour une optique classique et modèle équidistant pour une optique fish-eye.

[Kannala et Brandt, 2006] introduisent une nouvelle forme de projection pour être plus générique (avoir un même modèle pour les objectifs classiques et les objectifs fish-eye). Le modèle de projection suit l'équation : $r = k_1\theta + k_2\theta^3 + k_3\theta^5 + k_4\theta^7 + k_5\theta^9 + ...$ ce qui donne plus de paramètres mais en théorie une plus grande précision sur la projection. Ils proposent alors trois modèles à 6, 9 et 23 paramètres prenant en compte les déformations de nombreux objectifs. Le dernier modèle à 23 paramètres tient compte des distorsions radiales et tangentielles. En calibrant la caméra Pike F421C, nous avons obtenu une erreur résiduelle moyenne de 0.50 pixel pour le modèle à 6 paramètres, 0.46 pixel pour le modèle à 9 paramètres et 0.44 pixel pour celui à 23 paramètres. Cette erreur représente la racine carrée de la moyenne des carrés des distances entre les centroïdes modélisés et mesurés de la mire d'étalonnage (figure 5.5). Nous avons choisi le meilleur compromis entre simplicité du modèle et qualité de reprojection. Entre le modèle à 6 paramètres et à 9 paramètres, il y

a une amélioration de 9% sur l'erreur résiduelle moyenne alors que l'on note seulement une amélioration de 4% entre le modèle à 9 paramètres et celui à 23. Nous avons donc choisi le modèle de projection à 9 paramètres pour garder une bonne qualité de la projection tout en ayant une plus grande simplicité pour le calcul du modèle de projection inverse. Avec l'appareil photo Canon EOS, nous avons obtenu une erreur RMS (« *Root Mean Square* ») de 0.85 pixels pour une image de 5616x3744 avec le modèle à 9 paramètres. Nous obtenons ainsi un modèle permettant de projeter un point 3D dans le référentiel caméra en coordonnées pixel dans chaque image.

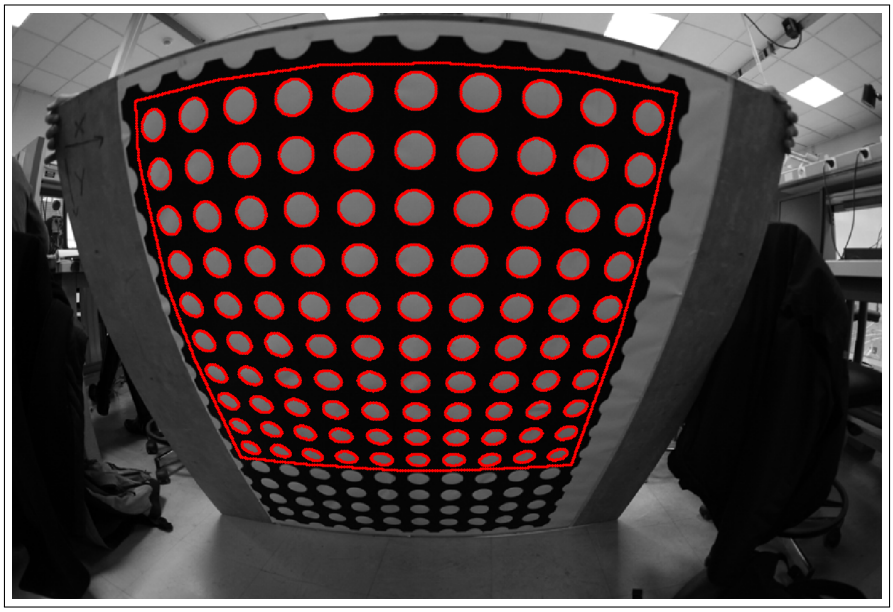


FIGURE 5.5 – Mire d'étalonnage pour la caméra (pour l'étalonnage, la zone de la mire a été encadrée à la main en rouge et les cercles en rouge ont alors été détectés automatiquement).

L'étalonnage correct de la caméra nous permet de corriger les images fish-eye en simulant une projection perspective comme sur la Figure 5.6.

5.1.3 Etalonnage scanner laser/caméra

Pour utiliser les données de la caméra avec celles du scanner laser, nous avons besoin de connaître les transformations géométriques pour passer du référentiel laser au référentiel caméra. Cette transformation est rigide puisque les capteurs sont fixés sur la même plateforme (figure 5.1) et doit être déterminée de façon précise par un étalonnage. Nous avons utilisé une méthode développée et détaillée dans [Dupont et al., 2005].

L'idée de base est d'utiliser une mire d'étalonnage plane. Dans notre cas, nous utilisons la même mire que celle utilisée pour l'étalonnage de la caméra (figure 5.5). Les points du laser dans un profil 2D (figure 5.7) doivent intersecter la mire plane dont la position est estimée par les images de la caméra. Nous avons alors une contrainte géométrique

sur la transformation rigide entre le système de coordonnées de la caméra et le système de coordonnées du laser. Nous effectuons les mesures conjointes du capteur image et du capteur laser une quinzaine de fois avec à chaque fois une position différente de la mire. Une optimisation non linéaire nous permet d'obtenir une erreur RMS de 5 *cm* entre les positions des points lasers dans le référentiel caméra et leurs projetés sur le plan de la mire. Nous obtenons ainsi une matrice de rotation et un vecteur de translation pour transformer les coordonnées d'un point du repère laser au repère caméra.

Tous ces étalonnages de la caméra et du laser sont effectués en amont des acquisitions et nous considérons qu'ils restent valables lors des acquisitions car le système laser/caméra est rigidement lié puisque monté et démonté en bloc sur le toit du véhicule (figure 5.1).

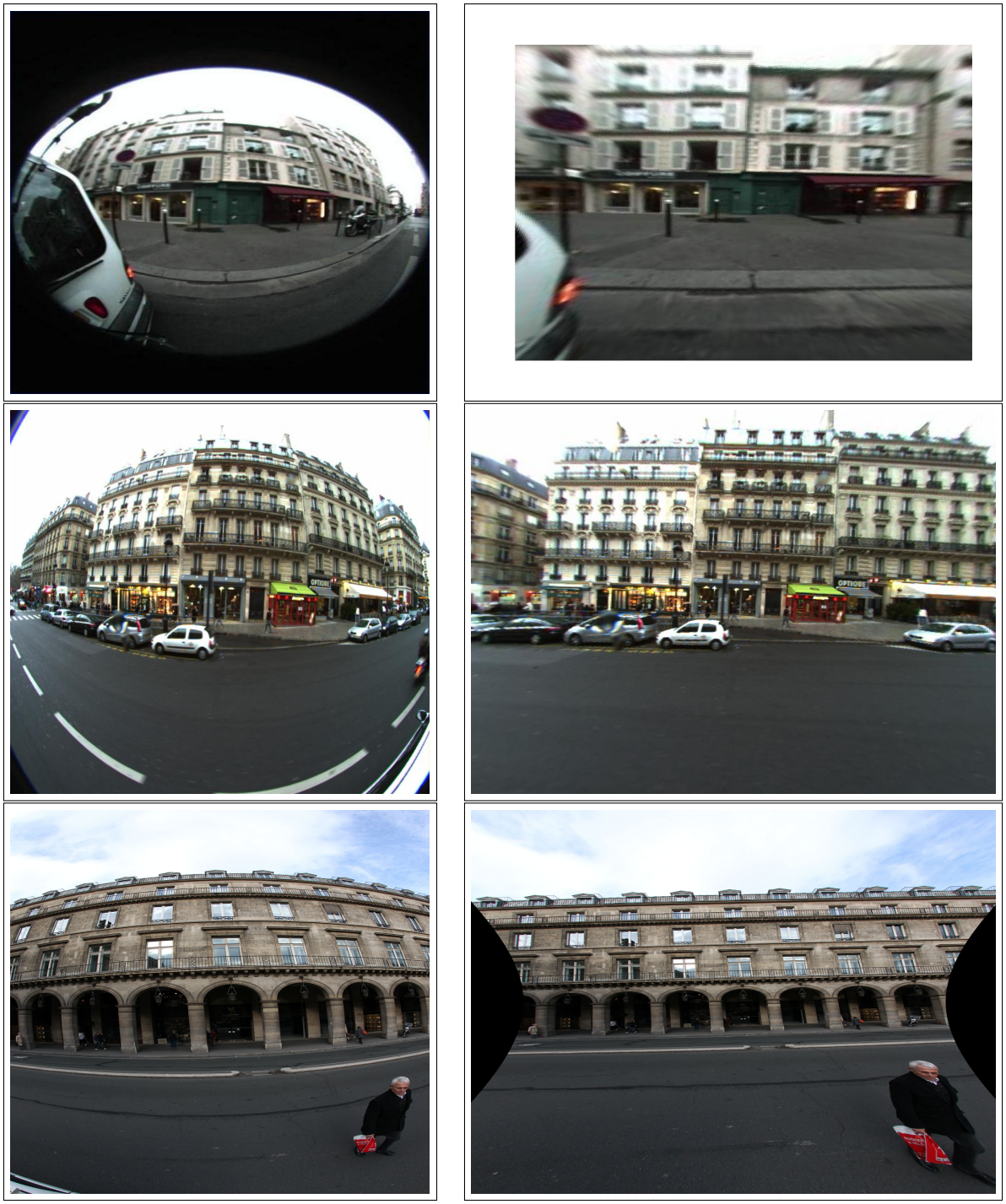


FIGURE 5.6 – Correction de la déformation des images due aux objectifs très grand angle fish-eye pour les caméras Marlin (en haut), Pike (au milieu) et l'appareil photo Canon (en bas).

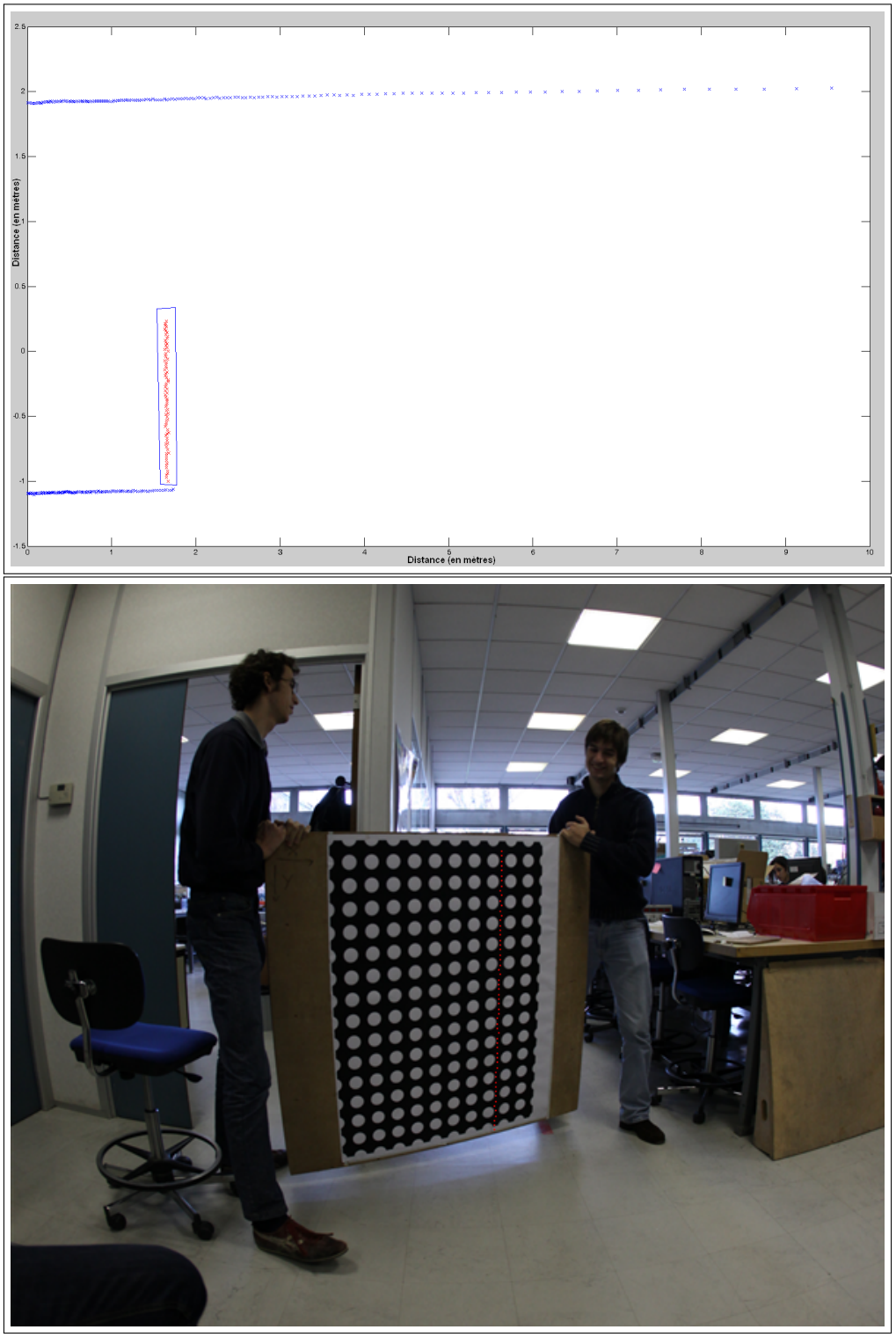


FIGURE 5.7 – En haut, profil laser avec sélection des points de la mire (points en rouge). En bas projection des points laser dans l'image après étalonnage (points en rouge, projection des points laser sur la mire).

5.2 Colorisation temps réel de nuages de points

Nous présentons dans cette partie la colorisation temps réel des nuages de points. Pour cela, nous avons développé deux méthodes : une première pour des caméras avec une haute fréquence d'acquisition (comme la caméra Marlin) et une seconde pour une fréquence faible d'acquisition des images (comme la caméra Pike et l'appareil photo Canon).

Pour préciser les ordres de grandeur, nous avons environ 1 000 points dans chaque profil toutes les 100 ms avec le scanner laser, 30 images par seconde pour la caméra Marlin F046C (résolution de 776x580), 5 images par seconde pour la caméra Pike F421C (résolution de 2 048x2 048) et une image par seconde pour l'appareil photo Canon EOS 5D (résolution de 5 616x3 744). Le véhicule se déplace à une vitesse lente (environ 5 km/h) pour disposer d'une plus grande résolution horizontale.

5.2.1 Colorisation par couplage direct « profil laser-image »

Avec une caméra de fréquence suffisante (30 Hz pour une Marlin F046C), il est possible d'associer à chaque profil laser une image de la caméra avec un temps d'acquisition suffisamment proche du temps d'acquisition du profil laser. On raisonne alors par couple « profil laser-image » .

Plus précisément, chaque profil 2D du scanner laser et chaque image de la caméra sont datés par notre logiciel de synchronisation de données. On sélectionne parmi les dernières images reçues de la caméra l'image acquise au moment le plus proche de la date d'acquisition du profil laser et obtenons ainsi le couple « profil laser-image ». On utilise alors les paramètres de la transformation rigide (déterminés lors de l'étape d'étalonnage scanner/caméra) pour transposer les points du repère laser au repère caméra. Dans le repère caméra, les points sont projetés à l'aide du modèle de projection dont les paramètres ont été définis à l'étape d'étalonnage du fish-eye. Nous obtenons alors pour chaque point laser ses coordonnées en pixels dans l'image fish-eye correspondante et pouvons donc lui attribuer une couleur correspondante. Ceci permet de construire un nuage de points 3D coloré en temps réel. Une illustration de cette colorisation est représentée sur la figure 5.8.

Pour obtenir une précision suffisante, la différence de temps d'acquisition entre une image et un profil laser se doit d'être la plus petite possible. Il est donc nécessaire d'avoir une fréquence d'image suffisante pour que les images soient acquises à un temps suffisamment proche du profil laser. Avec une fréquence du scanner de 10 Hz pour un IBEO LD et une caméra Marlin à une fréquence de 30 Hz, la différence de temps d'acquisition sera au pire de $+/-16$ ms [1], ce qui fait en valeur moyenne absolue un écart de 8 ms. Sachant que le véhicule se déplace à 5 km/h lors de nos acquisitions avec le scanner IBEO et la caméra Marlin (dû à la faible fréquence du scanner IBEO et non dû à la caméra), un point fixe sur la façade aura bougé d'environ 1.1 cm dans le repère mobile de la caméra pendant les 8 ms. L'erreur de projection due à la synchronisation peut donc être négligée.

1. $\frac{1}{2} \times \frac{1}{30} \approx 16.6\ ms$

FIGURE 5.8 – Nuage de points de la rue Gay-Lussac à Paris coloré avec une caméra basse résolution Marlin par couplage direct.

5.2.2 Colorisation par mise en correspondance de données images et laser géo-référencées

En passant à une caméra moyenne ou haute résolution, nous ne pouvions plus utiliser de couplage « scanner laser-caméra » comme décrit précédemment, dû aux faibles fréquences d'acquisition. Une fréquence de 1 Hz induit un manque d'information visuelle qui ne permet plus de faire directement la synchronisation des images avec les données laser : nous avons en effet une différence moyenne de 250 ms (500 ms au pire) sur la synchronisation scanner/laser. En utilisant le même raisonnement développé dans le paragraphe précédent, avec une vitesse du véhicule de 5 km/h, 250 ms va représenter un déplacement d'environ 34.7 cm, ce qui ne permet plus d'exploiter les images pour colorer les points avec notre première méthode.

Nous avons donc considéré les données image comme une source de données indépendante des données laser. Avec la localisation précise du véhicule (à une fréquence de 10 Hz et une date « d'acquisition » pour chaque position qui va être calculée par l'algorithme de localisation), nous pouvons géo-localiser chaque image de la caméra dans le même référentiel que celui du nuage de points 3D. A chaque image reçue de la caméra, nous récupérons la position du véhicule dont la date d'acquisition est la plus proche de la date d'acquisition de l'image. Avec une localisation à 10 Hz et une fréquence d'image de 1 Hz, cela donne une erreur moyenne de synchronisation de 25 ms (50 ms [1] au pire).

Cette erreur de projection due à la synchronisation localisation/image peut être

1. $\frac{1}{2} \times \frac{1}{10} = 50\ ms$

améliorée par interpolation linéaire de la position et de l'orientation entre deux points de la trajectoire. Plus précisément, pour chaque image, on trouve les deux positions dont les dates d'acquisitions encadrent la date d'acquisition de l'image. On estime ensuite par interpolation linéaire entre ces deux positions, la position dont la date d'acquisition correspond à la date d'acquisition de l'image. Cette étape permet de diminuer l'erreur de projection due à la synchronisation mais cette amélioration reste difficile à quantifier. De plus, la précision de localisation de l'image va aussi dépendre de la précision relative de la localisation du véhicule.

Dans une deuxième étape, pour chaque point 3D du nuage de points dans le même repère terrestre, nous récupérons l'image géo-localisée la plus proche géométriquement. Nous calculons pour cela la distance entre le point du nuage et le centre des caméras pour chaque image. Une fois l'image la plus proche sélectionnée, nous pouvons passer le point 3D du repère terrestre dans le repère de la caméra puis projeter le point dans l'image à l'aide du même modèle de projection et récupérer sa couleur. Le nuage de points provenant d'un scanner IBEO LD coloré avec la caméra moyenne résolution Pike est visible sur la figure 5.9. Plusieurs exemples de nuages de points colorés avec cette méthode en utilisant l'appareil photo Canon EOS 5D et les scanner SICK LMS sont visibles sur les figures 5.10, 5.11 et 5.12.

FIGURE 5.9 – Nuage de points coloré de la rue Soufflot à Paris, utilisation de la caméra moyenne résolution Pike par mise en correspondance de données image et laser.

Ces traitements se font en « temps réel différé », c'est-à-dire que nous stockons les données (position et orientation du véhicule, données des lasers et des caméras) dans une mémoire tampon (de l'ordre de quelques secondes) de manière à avoir suffisamment de données pour faire les traitements. Cela signifie que le nuage de points 3D coloré sera

FIGURE 5.10 – Nuage de points coloré de la rue Soufflot à Paris, utilisation de l'appareil photo Canon par mise en correspondance de données image et laser.

obtenu en moins de quelques secondes après l'arrêt des acquisitions à bord du véhicule, le temps de traiter la fin des données. Cela s'explique par le fait qu'avec une fréquence de 1 Hz pour la caméra, nous pouvons acquérir des profils laser dont l'image correspondante n'a pas encore été prise. Il faut donc stocker toutes ces données temporairement avant de les traiter. Le temps de latence va dépendre de la fréquence du capteur de plus basse fréquence, ici les caméras. Avec une fréquence de 1 Hz, il convient de traiter les données des lasers avec un temps de latence de 500 ms pour avoir une image correspondante au profil laser acquis, c'est-à-dire l'image dont l'horodatage est le plus proche de l'horodatage du profil laser.

5.2.3 Comparaison

Nous venons de voir nos deux méthodes pour coloriser un nuage de points à partir d'images de caméras. Pour ces deux méthodes, la précision colorimétrique de la colorisation comme la précision géométrique du nuage de points 3D vont dépendre fortement de la précision des étalonnages présentés plus haut (étalonnage caméra et scanner laser/caméra) mais aussi de la précision de la localisation (position et orientation). Par exemple, une erreur de 1 ° sur l'estimation du tangage va entraîner une erreur de 15 cm sur la position du point du scanner laser. Cette précision va donc aussi se répercuter sur la précision colorimétrique de la colorisation.

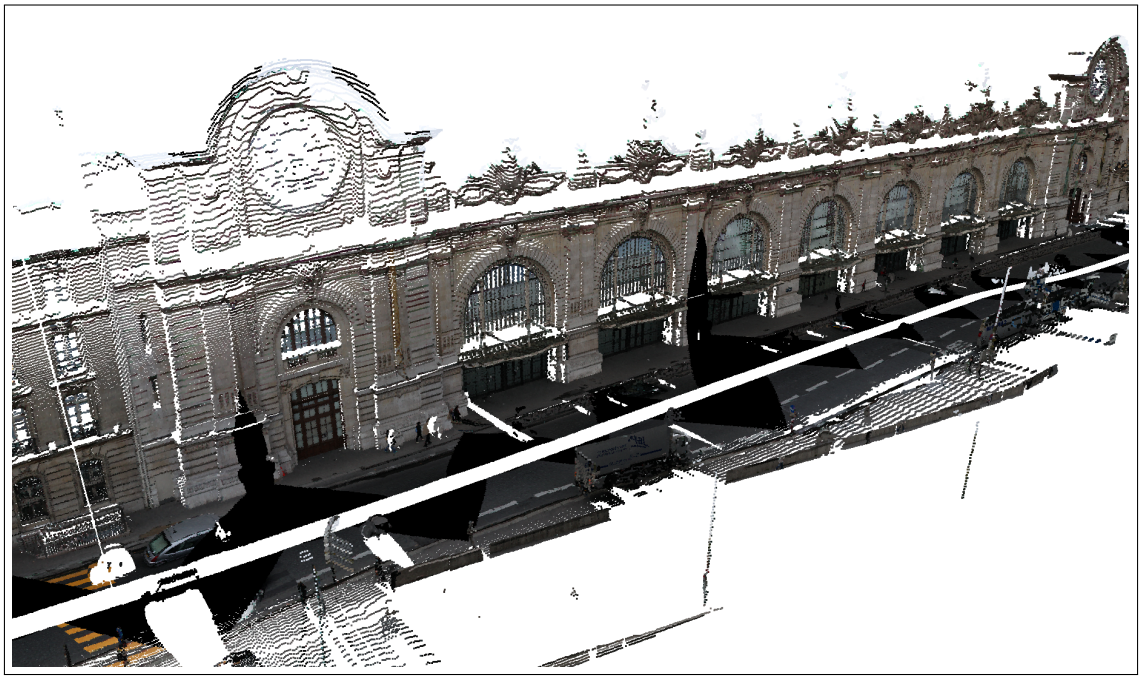


FIGURE 5.11 – Nuage de points coloré de la façade Nord du musée d'Orsay à Paris, utilisation de l'appareil photo Canon par mise en correspondance de données image et laser.

La première méthode de couplage direct « profil laser-image » nous permet « en théorie » d'obtenir des points colorés avec une plus grande précision colorimétrique. En effet, la précision de la couleur d'un point (c'est-à-dire la bonne correspondance de la couleur avec la géométrie) dépendra uniquement de la précision des étalonnages et de la synchronisation entre un profil laser et l'image associée. Seule la localisation d'un point en 3D et non la couleur dépendra de la bonne localisation du véhicule. Cela nécessite des données bien synchronisées et donc une grande fréquence d'acquisition des images, ce qui implique souvent des images de basse résolution.

L'avantage de la deuxième méthode est de pouvoir utiliser n'importe quel type de caméra puisque les données laser et images sont acquises et géo-référencées de manière indépendante. Mais la précision de la couleur va dépendre de la précision du géoréférencement des deux sources. Même en utilisant un géoréférencement identique pour les deux sources de données (images et laser) comme cela est le cas dans nos expérimentations, la précision sera plus grossière que dans le cas du couplage direct.

L'avantage de ces deux méthodes est de traiter les données en temps réel ou « temps réel différé » ce qui permet d'obtenir un nuage de points 3D coloré directement à la fin des acquisitions.

Ces travaux ont fait l'objet d'une publication à la conférence [$SFPT, 2009$] et dans la revue [$RFPT, 2010$].

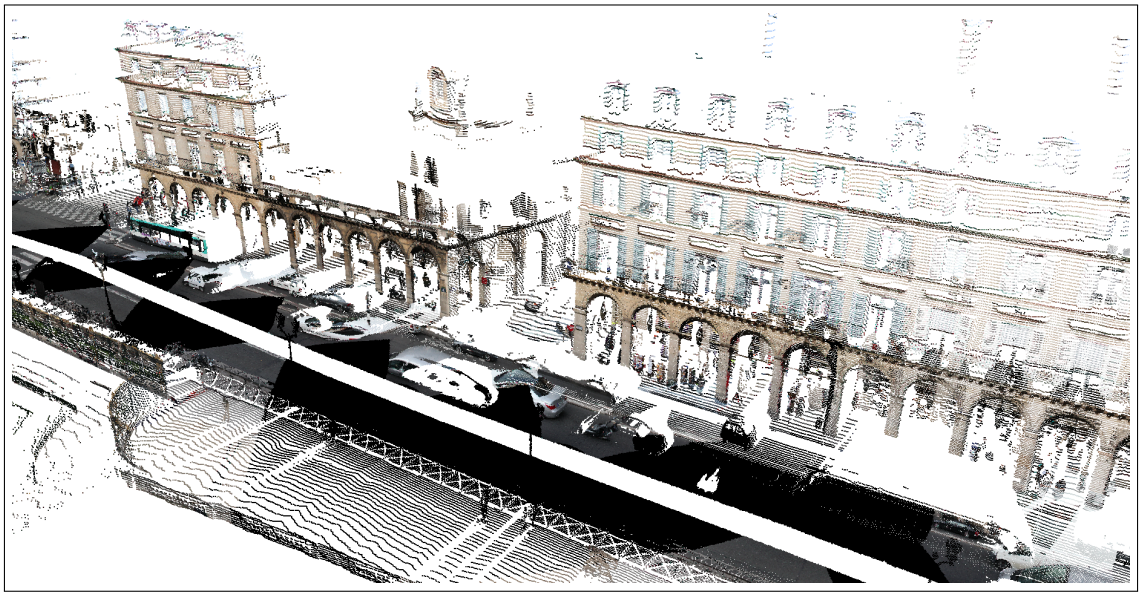


FIGURE 5.12 – Nuage de points coloré de la rue de Rivoli à Paris, utilisation de l'appareil photo Canon par mise en correspondance de données image et laser.

5.3 Texturation

Nous utilisons maintenant les images pour texturer les modèles 3D. Comme dans le cas de la colorisation, nous avons deux méthodes de texturation : une pour les caméras haute fréquence et une pour les caméras basse fréquence. Un modèle 3D texturé est formé d'un modèle 3D qui représente la géométrie de l'environnement et d'une texture (ou carte de texture) représentant une information visuelle de l'environnement.

5.3.1 Modèle 3D texturé à partir de caméras haute fréquence

Nous présentons dans cette partie notre méthode de texturation temps réelle pour les images de la caméra Marlin haute fréquence. Pour cela, nous n'utilisons pas la construction des modèles 3D telle que décrite dans les chapitres 3 et 4 mais la méthode développée par [Brun, 2007] qui est temps réel. Le modèle 3D est construit profil par profil au cours de l'acquisition. Pour cela, deux points voisins dans un même profil sont reliés à un autre point voisin dans le profil suivant pour former un triangle.

Au fur et à mesure de la construction des triangles du modèle 3D, nous calculons leurs textures respectives. Un triangle est défini par trois points et d'après l'algorithme de triangulation, deux points sont voisins dans le même profil et un point appartient au profil adjacent. Pour la texture, nous choisissons une image dont la date d'acquisition est la plus proche de la date d'acquisition moyenne des deux profils. On projette ensuite les sommets du triangle du repère laser dans l'image correspondante avec la même méthode que celle décrite pour la colorisation, comme présenté sur la figure 5.13. On obtient ainsi une texture du triangle que l'on pourra plaquer sur le triangle du maillage lors du rendu. Lors du rendu, un moteur comme OpenGL fait une interpolation bilinéaire des pixels de textures à l'intérieur d'un triangle en fonction des coordonnées de textures des trois sommets. Or avec une image fish-eye, l'interpolation n'est pas bilinéaire. Mais la déformation de la texture due à l'image fish-eye à l'intérieur d'un seul triangle est considérée comme négligeable vue la petite taille des triangles, c'est pourquoi nous ne faisons pas de correction des images.

Le même algorithme est appliqué pour chaque triangle du maillage. Nous enregistrons les textures des triangles dans des fichiers appelés cartes de textures. Pour cela, nous entourons chaque texture de triangles par un rectangle dans l'image (boîte englobante) et nous enregistrons chaque rectangle contenant la texture d'un triangle côte à côte dans des fichiers textures comme sur la figure 5.14. A titre d'exemple, cette méthode produit 302 cartes de textures de résolution 256x256 pour un modèle de rue de 150 m ayant un seul côté de la rue texturé.

Le nombre de cartes de textures (ici 302) est important dû à notre méthode pour stocker les textures des triangles (sans optimisation, les uns à la suite des autres quel que soit leur taille). Or le nombre de cartes de textures joue un rôle important lors du rendu car la mémoire de la carte graphique est limitée. Un modèle ne pourra alors plus être affiché en temps réel à partir d'une certaine taille (ici la longueur de la rue). Pour pallier à cela, nous avons développé un algorithme pour diminuer le nombre de cartes de

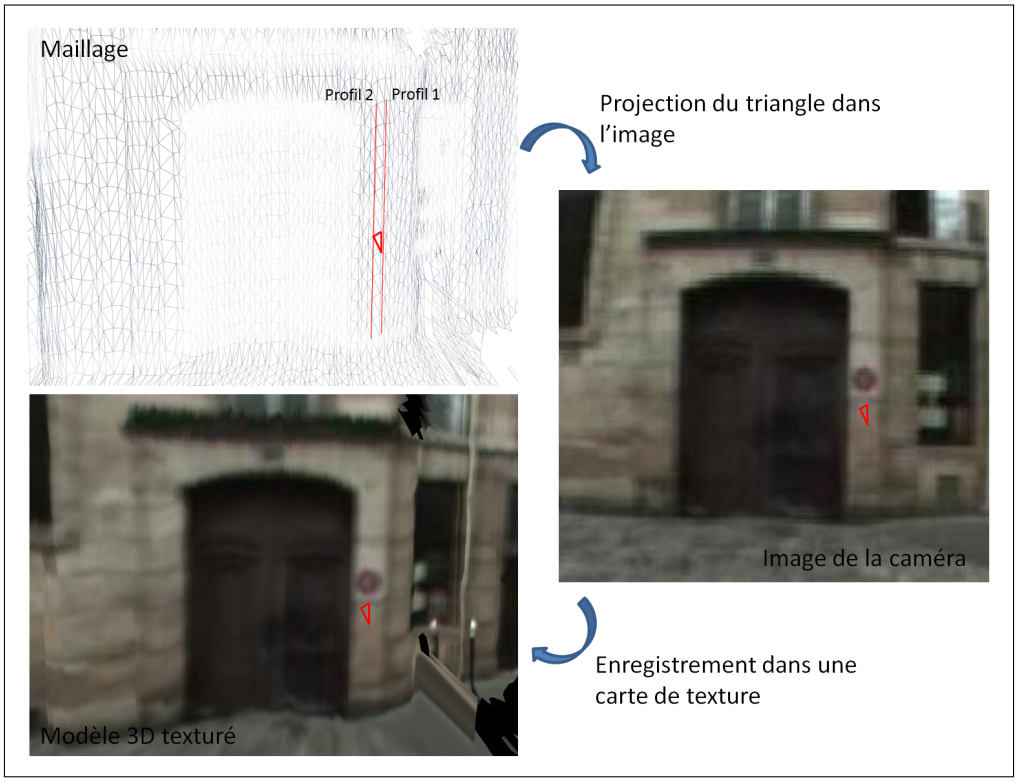


FIGURE 5.13 – Illustration de la projection des triangles du maillage sur les images de la caméra.

texture (sachant que la taille d'une carte de texture est fixée à 256x256 pixels). Pour cela, nous effectuons une classification des rectangles (englobant les textures des triangles) en les regroupant suivant leur hauteur. Par exemple, cela donne une carte de texture avec les rectangles dont la hauteur est comprise entre 2 et 5 pixels. Une autre carte de texture comportera elle des rectangles dont la hauteur est comprise entre 6 et 8 pixels et ainsi de suite jusqu'à classification de toutes les textures. Avec cette classification, nous sommes passés de 302 cartes de texture de résolution 256x256 à seulement 108 cartes de résolution 256x256 pour le modèle d'une rue de 150 m de longueur avec un seul côté de la rue texturée. Un modèle de plus grande taille peut ainsi être affiché en temps réel.

Un moteur de rendu comme OpenGL peut alors afficher le modèle 3D texturé avec les cartes de texture et le maillage triangulé. Ce travail a été effectué pour la caméra basse-résolution Marlin et un résultat de modèle 3D de la rue Saint-Jacques à Paris est visible sur la figure 5.15.

Tous ces traitements sont effectués en « temps réel différé » , c'est-à-dire les calculs se font à la même vitesse que l'acquisition des données mais nous devons garder un délai de quelques profils pour le calcul de la triangulation et de la texturation. En revanche, cette méthode ne fonctionne qu'avec un modèle généré profil par profil.

Ces travaux sur la texturation temps réelle à partir de caméras haute fréquence ont fait

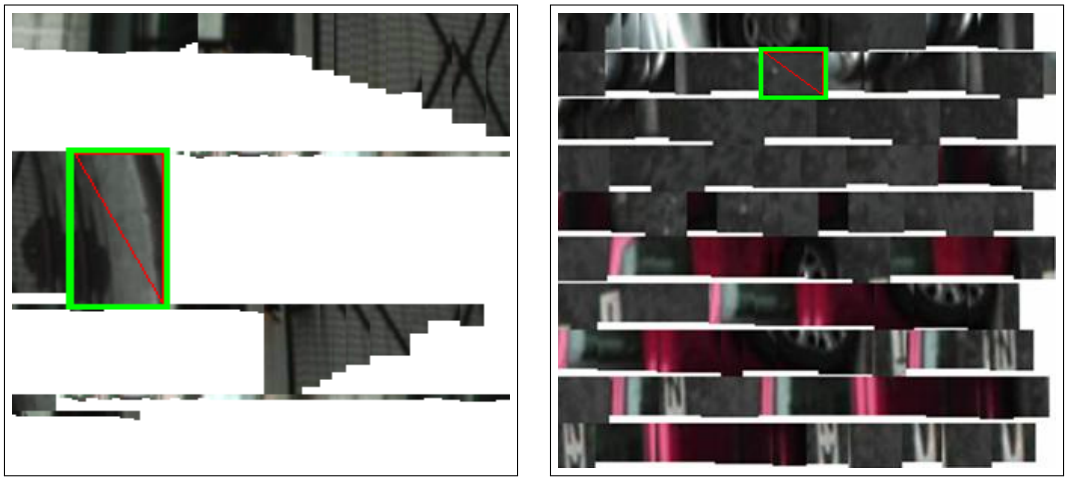


FIGURE 5.14 – A gauche, carte de texture générée pour stocker les textures des triangles et à droite amélioration du stockage par classification suivant la taille (texture d'un triangle du maillage : triangle en rouge et boîte englobante : rectangle en vert).

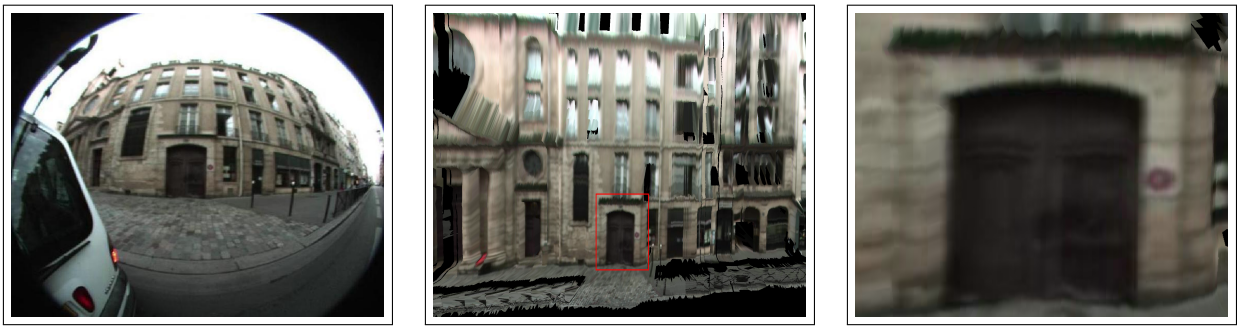


FIGURE 5.15 – A gauche, image de la caméra Marlin utilisée pour les textures, au milieu modèle 3D généré et à droite détail sur une partie du modèle.

l'objet d'une publication à la conférence [$AFIG, 2007$] et [$MMT, 2007$].

5.3.2 Modèle 3D texturé à partir de caméras basse fréquence

Avec une caméra plus haute résolution, la fréquence des images et les temps de traitement ne permettent pas de créer des cartes de texture en temps réel comme la méthode précédente. Nous avons donc modifié notre stratégie de création de cartes de texture pour créer des modèles en 3D temps réel tout en minimisant l'espace occupé par les cartes de textures.

Comme nous l'avons vu lors de la section précédente, nous ne corrigeons pas les images fish-eye lors de la création des cartes de textures. Nous faisons l'hypothèse que la déformation induite par l'objectif est négligeable à l'échelle d'un triangle d'un maillage. Plutôt que de créer de nouvelles cartes de textures, nous avons donc décidé de garder

directement les images de la caméra comme cartes de textures. Ceci est avantageux sur deux points : d'une part, nous n'avons pas à calculer des cartes de textures ce qui nous permet de faire du temps réel même avec des images 5 616x3 744 puisque les images ne sont pas modifiées et d'autre part si la fréquence des images est bien choisie par rapport à la vitesse du véhicule alors le nombre de cartes de texture créées sera minimal.

Nous pouvons voir sur la figure 5.16 que, pour une acquisition avec l'appareil photo Canon, le chevauchement est suffisant pour capter toute la scène. Au contraire, avec une caméra Pike fonctionnant à 5 Hz, nous avons augmenté la taille des zones de chevauchements entre images successives comme on le voit sur la figure 5.17. Dans ce cas, plus de cartes de textures sont stockées pour un modèle 3D. Toute la difficulté réside dans le bon compromis entre le chevauchement des images et le nombre de cartes de textures.

FIGURE 5.16 – Trois prises successives d'images avec l'appareil photo Canon EOS 5D photographiant la droite du véhicule (roulant à 5 km/h).

FIGURE 5.17 – Trois prises d'images successives avec la caméra Pike à une fréquence de 5 Hz photographiant la droite du véhicule (roulant à 5 km/h).

Pour faire le lien entre le modèle 3D et les textures, nous avons appliqué la méthode sur la colorisation à basse fréquence. Les images et les données laser sont géo-référencées

indépendamment. Le modèle 3D est créé à partir du nuage de points avec la méthode décrite dans les chapitres 3 et 4. Lors de l'étape de texturation, chaque sommet d'un triangle est projeté dans l'image lui offrant la meilleure vue, c'est-à-dire maximisant l'aire du triangle projeté. Cette dernière étape diffère de la colorisation. En effet, pour la colorisation, nous choisissions la caméra dont le centre optique est le plus proche du point à colorer. Or cette stratégie peut ne pas être optimale pour la texture d'un triangle en fonction de l'angle de vue, d'où le choix de l'image maximisant l'aire du triangle projeté.

Cette méthode de texturation a été testée avec un maillage de 835 006 triangles à une résolution de 15 cm généré à partir d'un nuage de points de 2 440 416 points (voir le chapitre 6 sur la génération de ce modèle). Nous avons conservé 65 cartes de textures (correspondant aux images de la caméra) pour un modèle de rue d'une longueur de 400 m pour lequel nous avons texturé les deux côtés. Pour le rendu, toutes les textures ont été diminuées à une résolution de 1 124x749. En effet, OpenGL ne peut charger 65 cartes de textures à une résolution de 5 616x3 744 pour faire du rendu temps réel sur une carte graphique standard ATI Radeon HD 3870. Avec une résolution des textures de 1 124x749, il est par contre possible d'afficher le modèle 3D entier en temps réel avec OpenGL. La figure 5.18 présente le modèle 3D texturé de la façade Nord du musée d'Orsay et la figure 5.19 le maillage correspondant. Le modèle 3D texturé (géométrie+textures) a été généré en 7 min sachant que le temps d'acquisition des données a été de 3 min 10 s. Nous sommes donc pour l'instant à environ 40% du temps réel pour la génération de nos modèles 3D texturé sur un QuadCore Q9300 2.50 GHz, 2 Go de RAM, sur un seul *thread* et sur CPU en sachant que le temps de calcul est extrêmement variable suivant les paramètres comme nous le verrons au chapitre 6.

Des erreurs de recalage entre textures (encerclé en rouge) sont visibles sur l'image de bas de la figure 5.18. Ces erreurs de recalage viennent principalement des imprécisions sur la position et l'orientation du véhicule. Il serait possible d'améliorer ces transitions par une technique de « *Bundle Adjustment* » qui consiste à minimiser l'erreur de reprojection de points observés dans plusieurs images. Cette technique est possible pour des images qui se chevauchent ce qui est le cas pour ces acquisitions mais qui ne serait plus valable si le véhicule roulait à grande vitesse (pour une caméra à une fréquence de 1 Hz). De même, des erreurs de recalage existent entre le maillage et les textures. Ces erreurs pourraient aussi être diminuées par des techniques de recalage image-laser comme celle de [Stamos et Allen, 2002] (par correspondance de droites 3D dans le nuage et de droites 2D dans les images) ou celle de [Deveau, 2006] (par appariement de points image et de points 3D) d'autant que la localisation du véhicule donne une bonne initialisation pour le recalage.

5.3.3 Conclusion sur la génération de modèles texturés

L'approche proposée de texturation permet d'afficher les images brutes de l'acquisition sur un maillage construit par les données des scanners laser. Il s'agit d'un résultat hybride de modélisation et de visualisation. La modélisation correspond à la création de la géométrie à partir des données de LiDAR et la visualisation correspond au rendu des images brutes comme textures du modèle 3D. Cette méthode permet de conserver les images brutes mais

aussi de les visualiser avec tous les détails sans nécessiter d'étape intermédiaire. Tout le processus de modélisation et de texturation n'est pas temps réel mais est suffisamment rapide (environ 40% du temps réel) pour espérer le devenir avec quelques optimisations.

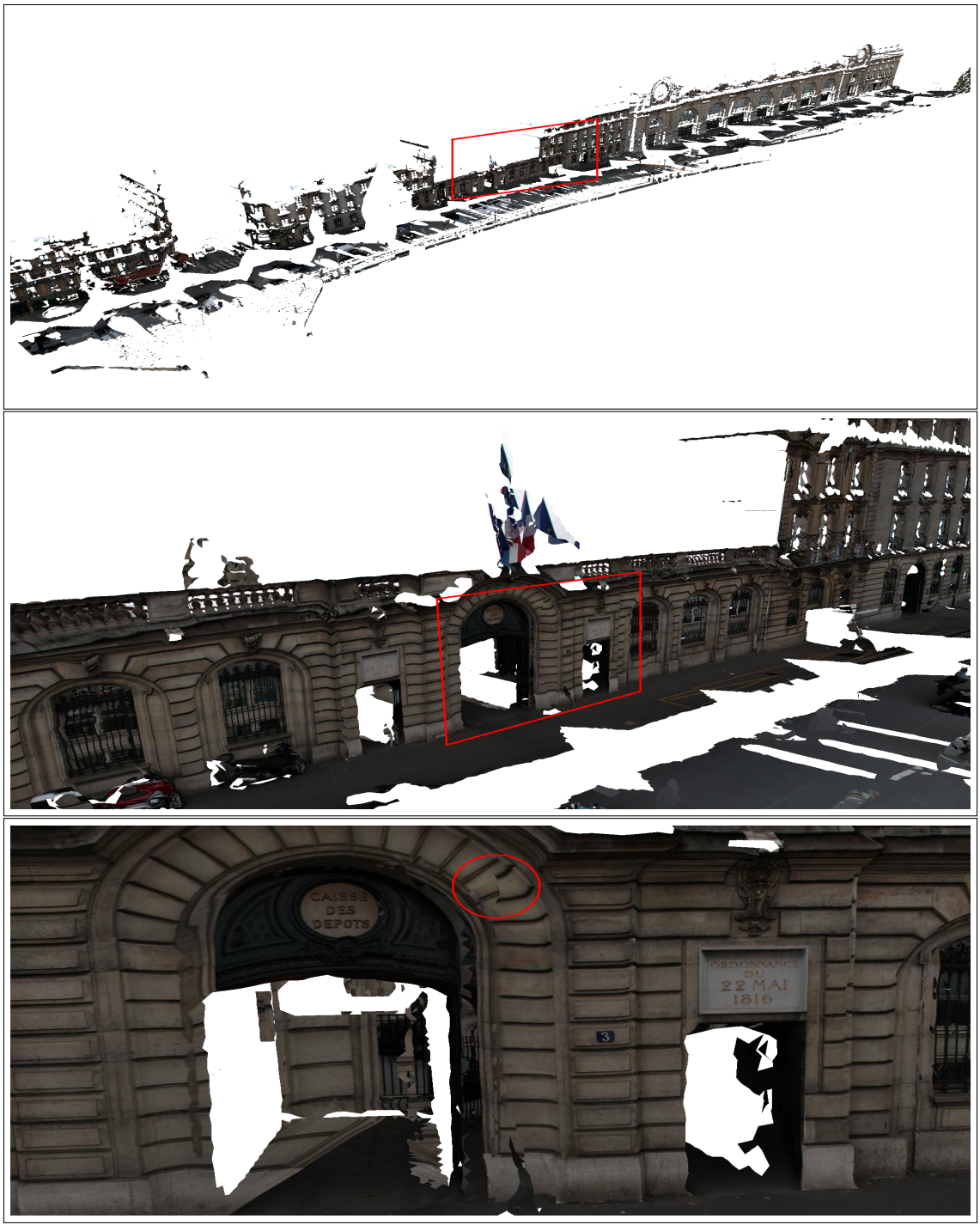


FIGURE 5.18 – Modèle 3D de la façade Nord du musée d'Orsay avec au milieu un détail du modèle et en bas un gros plan.

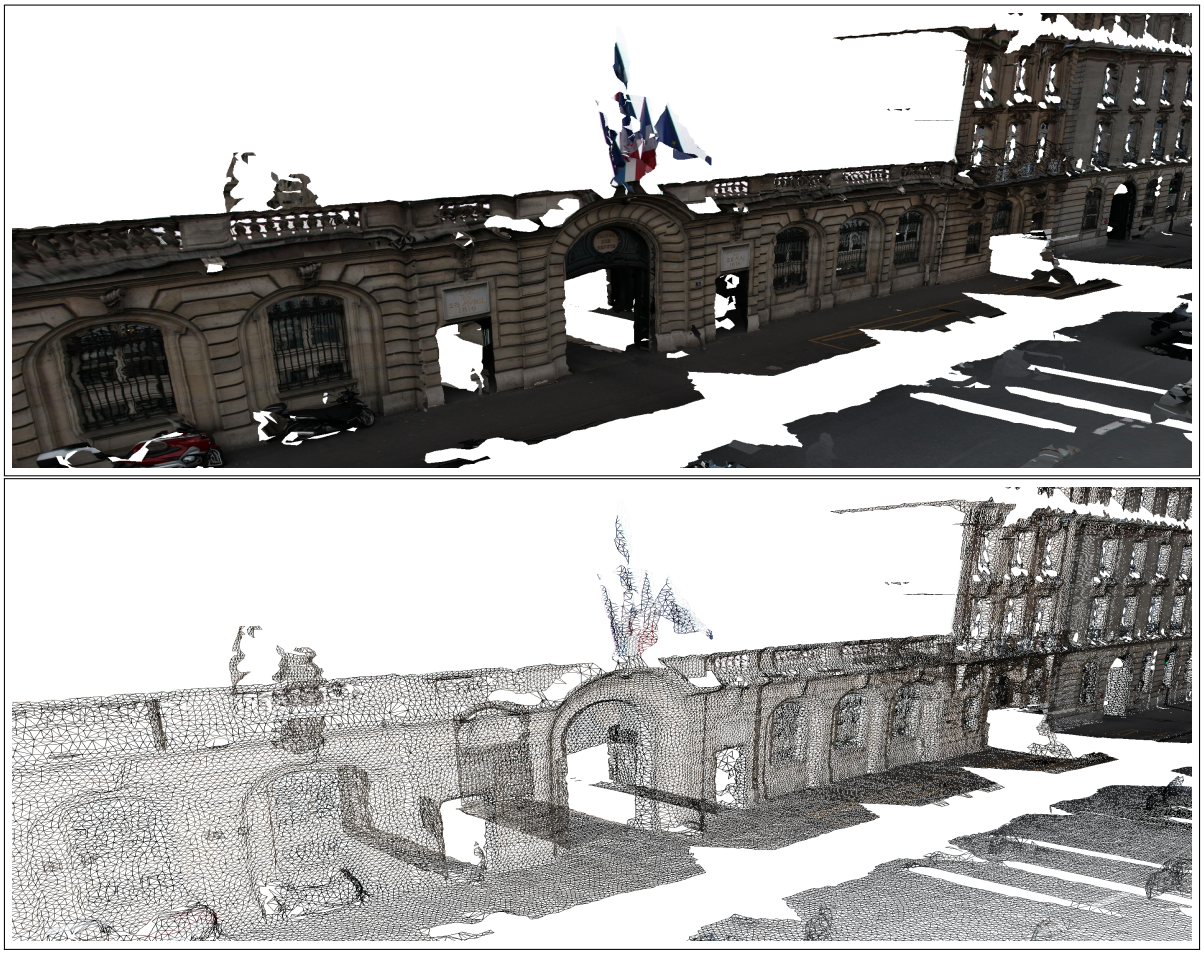


FIGURE 5.19 – Modèle 3D de la façade Nord du musée d'Orsay avec le maillage correspondant en bas.

Chapitre 6

Applications à différents environnements

Résumé

Ce chapitre présente toute la chaîne de modélisation utilisant l'ensemble des traitements que nous avons présentés aux chapitres précédents. Nous exposons trois applications de ces modèles avec la création de modèles 3D de scènes urbaines, la génération de cartes de distances de visibilité géométrique à partir de modèles de routes et la création de modèles 3D d'intérieurs.

6.1 Chaîne complète de modélisation proposée

Comme nous l'avons vu en introduction, nous nous sommes focalisés durant cette étude sur la création de modèles 3D terrestres photo-réalistes avec en entrée un nuage de points bruité et des images fish-eye référencées par rapport au nuage de points. Nous construisons un modèle texturé le plus proche des données terrains à partir de ces deux sources d'information, le tout en un minimum de temps.

Pour traiter les données de système mobile en temps réel tout en restant suffisamment générique vis à vis de la technologie d'acquisition, nous avons fait le choix de modéliser l'environnement par petites sections de nuages de points au cours de l'acquisition comme sur la figure 6.1. Une section est une petite zone géographique qui peut être traitée en une seule fois, lorsque tous les scanners sont passés dans la zone. Une mémoire tampon conserve la position 3D des points laser non traités. Une fois que le véhicule est « suffisamment éloigné », on opère sur les points laser non traités dans la « zone » la plus éloignée du véhicule. La Figure 6.1 présente un découpage en sections le long de l'abscisse curviligne de la trajectoire. On pose comme hypothèse que le véhicule ne passe pas deux fois au même endroit. Si l'ensemble des traitements d'une section se fait à la même vitesse que le temps d'acquisition alors le processus est appelé « temps réel différé », ce qui correspond à du temps réel avec une certaine latence.

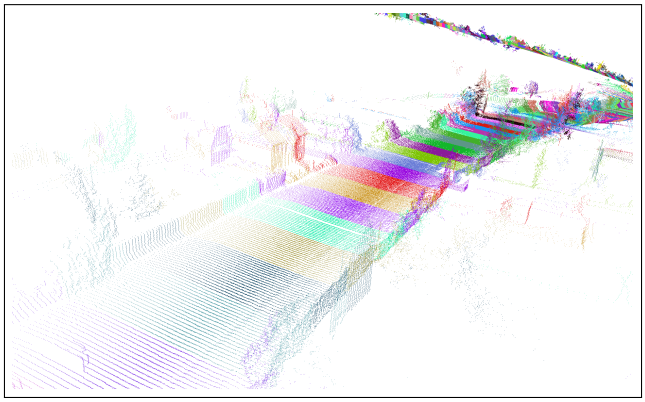


FIGURE 6.1 – Découpage du nuage de points en sections

Dans notre étude, sachant que les acquisitions ont été faites sur de petits tronçons (moins de 3 km), tous les traitements ultérieurs ont été opérés en une seule fois sur le nuage entier. Le problème des raccords entre sections n'a pas été étudié dans le cadre de cette thèse.

6.1.1 Etapes du processus

La chaîne complète de traitement de nuage de points est présentée au tableau 6.1. Le processus comporte 7 étapes pour passer d'un nuage de points et d'images à un modèle 3D débruité, maillé, simplifié et texturé. Certains paramètres dépendent directement de la distance d_{mean}, distance moyenne d'un point à son plus proche voisin. Ces paramètres

peuvent donc être calculés automatiquement.

Etapes	**Paramètres**	**Remarques**
1. Décimation	$d_{details}$, $d_{deci} = \max(\frac{d_{profil}+d_{trajectoire}}{2}, d_{details})$	Sélection des points de plus fortes courbures
2. Estimation des normales	$\delta_{normal} = 4d_{mean}$	Orientation des normales avec la direction des lasers
3. Débruitage par NLD3D	$\delta_{NLD3D} = 4d_{mean}$ et $\Delta_{NLD3D} = 12d_{mean}$	-
4. Détection des plans	δ_{plan}, γ_{plan}, α_{plan}, d_{voxel}, n_{min} et n_{max}	Projection des points sur leurs plans
5. Triangulation des points	$d_{maillage} = 10d_{mean}$	-
6. Simplification des zones planes	$\theta_{simpli} = 1\,^\circ$	-
7. Texturation	-	Sélection de la meilleure vue

TABLE 6.1 – Chaîne de traitement pour la modélisation de nuages de points.

Nous avons vu en détail dans les chapitres précédents les étapes de décimation, de calcul de normales, de débruitage, de détection de zones planes, de triangulation et de texturation. Les avantages et les limitations sont aussi détaillées dans les sections correspondantes ainsi que l'influence des paramètres sur les résultats. Nous ne décrivons ici que l'étape de simplification.

6.1.2 Etape de simplification

L'étape de simplification consiste à simplifier les triangles dans les zones planes. Pour cela, nous utilisons la méthode de [Hoppe et al., 1993] supprimant les arêtes du maillage formant localement une zone plane. Soit pq une arête du maillage et $Star(p)$ l'ensemble des triangles qui ont p comme sommet. Nous testons $Star(p)$ pour savoir s'il forme un cycle et si les normales des triangles ont deux à deux un angle inférieur à θ. Nous testons de la même façon $Star(q)$. Une fois ces conditions vérifiées, nous vérifions que le remplacement de l'arête pq par le sommet de position $\frac{1}{2}(p+q)$ n'entraîne pas de renversement de triangle. Si cela n'est pas le cas, nous conservons la nouvelle configuration. La figure 6.2 illustre la suppression de l'arête pq par le nouveau point de position $\frac{1}{2}(p+q)$.

L'objectif est de simplifier seulement les triangles dans les zones planes qui ont été détectées, c'est pourquoi nous prenons une petite valeur de $\theta_{simpli} = 1\,^\circ$. En effet, avant l'étape de simplification, les points sont projetés sur leurs plans respectifs donc les normales des triangles sont égales aux normales des plans. Une fois la simplification terminée, nous pouvons définir un taux de compression pour le maillage : il s'agit du ratio entre le nombre de triangles supprimés et le nombre de triangles avant la simplification.

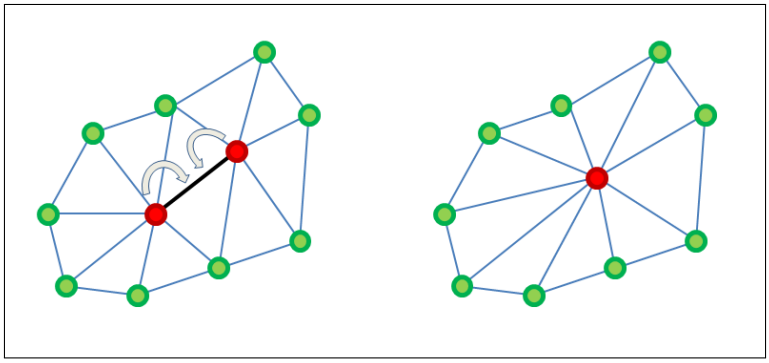


FIGURE 6.2 – Simplification du maillage par suppression d'arêtes.

6.2 Modèles 3D d'environnement urbain par système mobile LiDAR/caméra

6.2.1 Acquisitions et modélisations

Dans le cadre du projet TerraNumerica, nous avons étudié la création de modèles de scènes urbaines. Des acquisitions ont eu lieu dans Paris en 2008 dans la rue Soufflot, la rue de l'Abbé de l'Epée, autour du Panthéon, dans la rue Saint-Jacques et en 2009 dans la rue Soufflot, sur la place du Chatelet, dans la rue de Rivoli et devant la façade Nord du musée d'Orsay.

Pour tester notre méthode de modélisation, nous effectuons le traitement des données de la façade Nord du musée d'Orsay *LARA3Dv2010_Orsay*, nuage de points d'une longueur d'environ 350 m acquis en 3 min 10 s. Nous avons choisi de présenter ces résultats car nous disposons de données sur la même zone provenant de scanners fixes avec le nuage *FX_GX_Orsay*. Nous pouvons ainsi comparer les modélisations créées par ces deux technologies d'acquisition.

Nous définissons la résolution des modèles générés comme étant la distance de décimation d_{deci}. Le paramètre d_{deci} influence le nombre de points du maillage et donc le niveau de détail du modèle final. Nous avons construit deux modèles à partir du nuage de points *LARA3Dv2010_Orsay* : un modèle de résolution 7 cm et un modèle de résolution 15 cm. Pour le premier modèle, nous souhaitons garder le plus de détails, c'est pourquoi le paramètre $d_{details}$ a été fixé à 0. D'après l'algorithme de décimation du chapitre 3, $d_{details} = 0$ implique $d_{deci} = 7.05$ cm et donc un modèle de résolution 7 cm. Pour le deuxième modèle, nous avons choisi $d_{details} = 15$ cm ce qui donne $d_{deci} = 15$ cm ainsi qu'un niveau de détails de 15 cm. Avec $d_{details} = 0$, la résolution de 7 cm correspond à la meilleure résolution du nuage *LARA3Dv2010_Orsay* pour notre méthode de modélisation.

Le tableau 6.2 présente les différentes étapes du processus de modélisation avec les paramètres choisis ainsi que les temps de traitement pour le modèle avec une résolution de 7 cm. A partir d'un nuage de points de 2 440 416 points, nous obtenons un maillage décimé composé de 1 521 147 triangles pour un temps total d'environ 15 min. Pour le modèle avec une résolution de 15 cm (en gardant tous les autres paramètres identiques), nous construisons un maillage de 835 006 triangles en 7 min. Sachant que le temps d'acquisition des données est de 3 min 10 s, le premier modèle avec la résolution de 7 cm est construit à environ 20% du temps réel tandis que le second est à environ 40% du temps réel.

La figure 6.3 présente sur l'image du haut le maillage décimé et sur l'image du bas le modèle 3D texturé du nuage de points *LARA3Dv2010_Orsay* à une résolution de 7 cm. Nous pouvons voir que nous obtenons un modèle photo-réaliste grâce à la géométrie du maillage et aux textures. En revanche, le modèle final présente encore de nombreux trous non désirés.

La figure 6.4 compare les maillages des modèles de résolution 7 cm et de résolution 15 cm. Nous remarquons visuellement une différence de qualité entre ces deux modèles. Le choix de la résolution dépendra finalement de l'application et donc du compromis

Etapes	**Paramètres choisis**	**Temps**
1. Décimation	$d_{details} = 0\ cm$ donc $d_{deci} = 7.05\ cm$	4 s
2. Estimation des normales	$\delta_{normal} = 4d_{mean} = 19.2\ cm$	343 s
3. Débruitage par NLD3D	$\delta_{NLD3D} = 4d_{mean} = 19.2\ cm$ et $\Delta_{NLD3D} = 12d_{mean} = 57.6\ cm$	149 s
4. Détection des plans	$\delta_{plan} = 30\ cm$, $\gamma_{plan} = 7\ cm$, $\alpha_{plan} = 10.0^\circ$, $d_{voxel} = 50\ cm$, $n_{min} = 100$ et $n_{max} = 10\ 000$	134 s
5. Triangulation des points	$d_{maillage} = 10d_{mean} = 48\ cm$	265 s
6. Simplification des zones planes	$\theta_{simpli} = 1^\circ$	18 s
7. Texturation	-	2 s

TABLE 6.2 – Chaîne de traitement pour la modélisation du nuage de points *LARA3Dv2010_Orsay* à une résolution de 7 *cm*.

choisi entre la compression du maillage et la qualité de la reconstruction. Les modèles d'environnements urbains à une résolution de 15 *cm* auront environ 2 500 000 triangles par kilomètre tandis que ceux à une résolution de 7 *cm* environ 4 500 000 de triangles par kilomètre.

Le modèle de résolution 7 *cm* est constitué de 3 063 697 triangles avant la détection et simplification des zones planes et de 1 521 147 triangles après ce qui nous donne un ratio de compression d'environ 50%. Le fait de détecter les plans et de simplifier le modèle uniquement au niveau des plans nous permet de diminuer la taille du maillage tout en gardant la même qualité au niveau des détails. Afin d'illustrer ces propos, la figure 6.5 compare le modèle *LARA3Dv2010_Orsay* à une résolution de 7 *cm* avant et après simplification.

Nous avons construit un modèle 3D simplifié à partir du nuage *FX_GX_Orsay* à une résolution de 7 *cm* avec les mêmes paramètres que le modèle précédent. Les figures 6.6 et 6.7 présentent la comparaison des modèles produits à partir des données du système mobile LARA3D et du système fixe avant et après la simplification. Nous pouvons constater que les résultats sont visuellement presque similaires entre plateforme mobile et fixe à résolution identique.

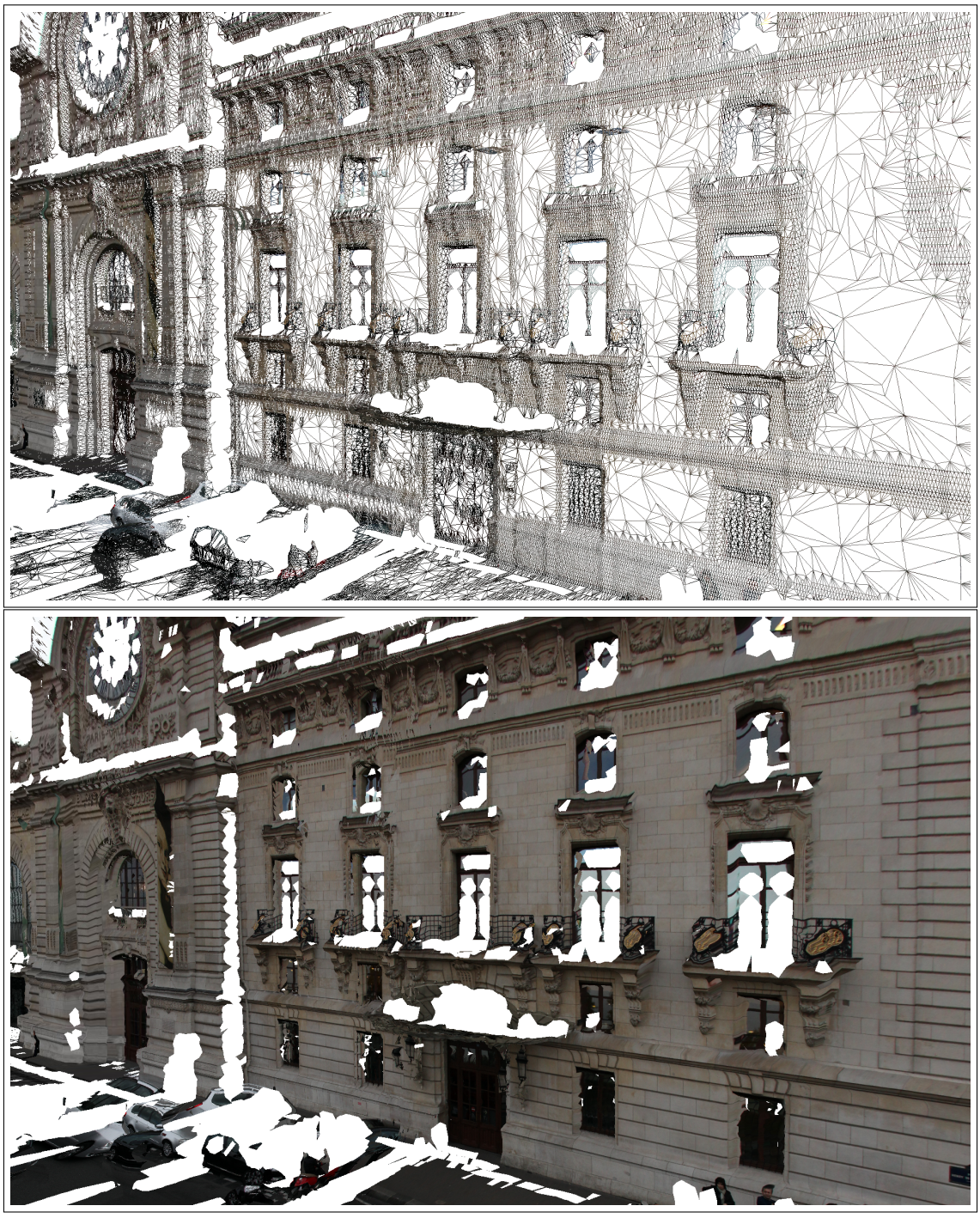


FIGURE 6.3 – En haut, maillage en fil de fer simplifié du nuage *LARA3Dv2010_Orsay* à une résolution de 7 *cm*. En bas, modèle 3D texturé.

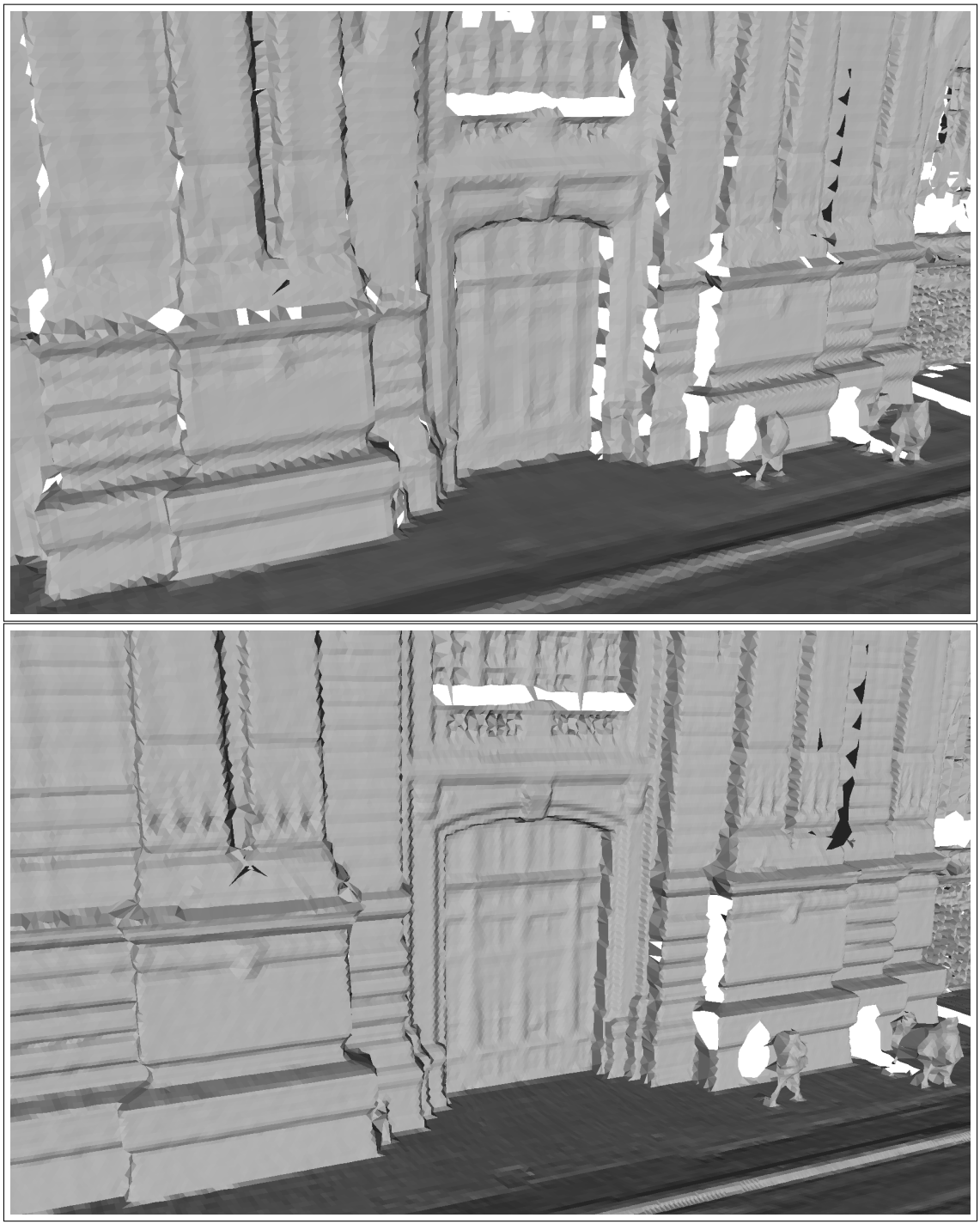


FIGURE 6.4 – En haut, maillage non simplifié de *LARA3Dv2010_Orsay* à une résolution 15 *cm*. En bas, maillage non simplifié de *LARA3Dv2010_Orsay* à une résolution de 7 *cm*.

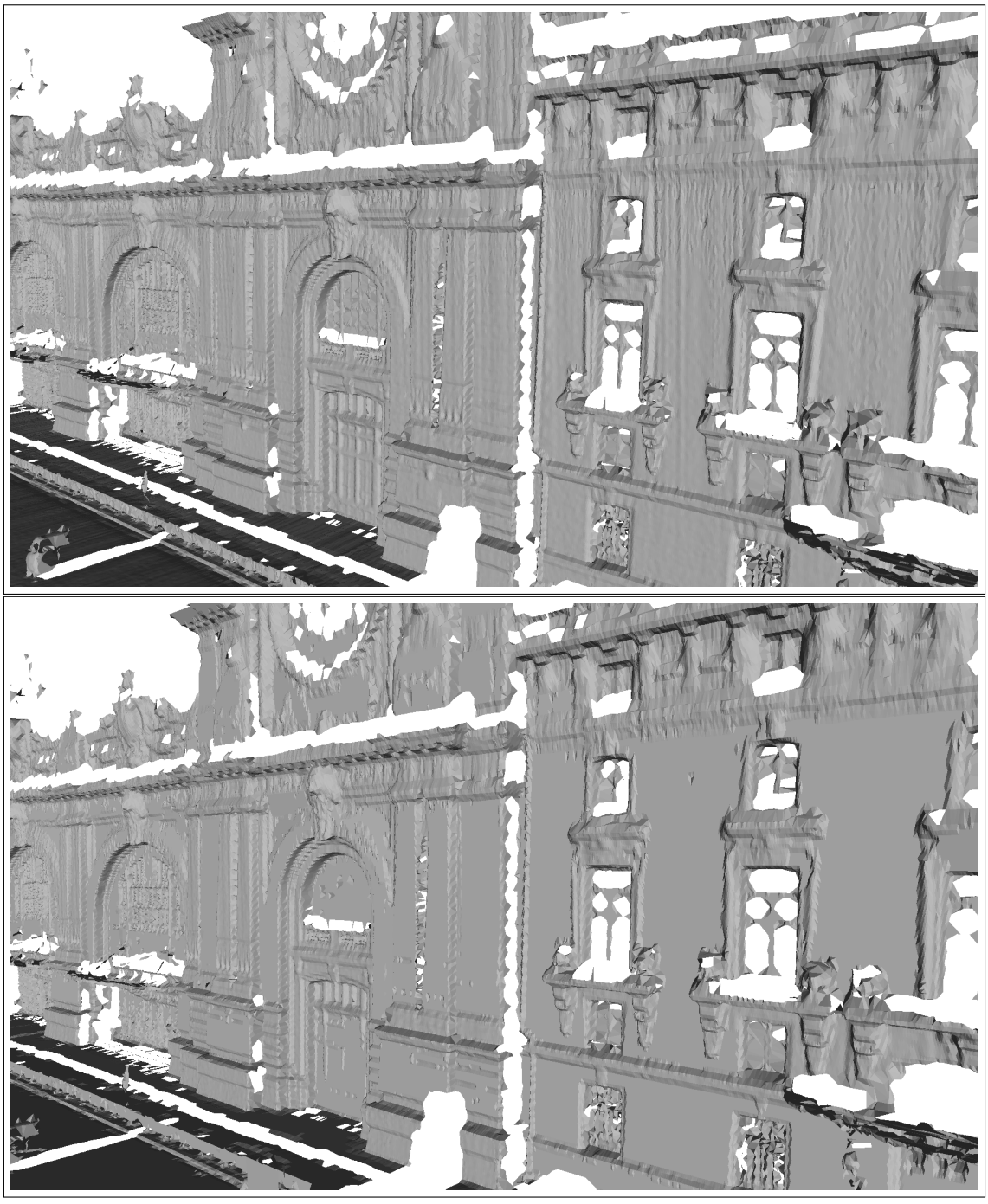


FIGURE 6.5 – En haut, maillage non simplifié de *LARA3Dv2010_Orsay* à une résolution 7 *cm*. En bas, même maillage simplifié par détection des zones planes.

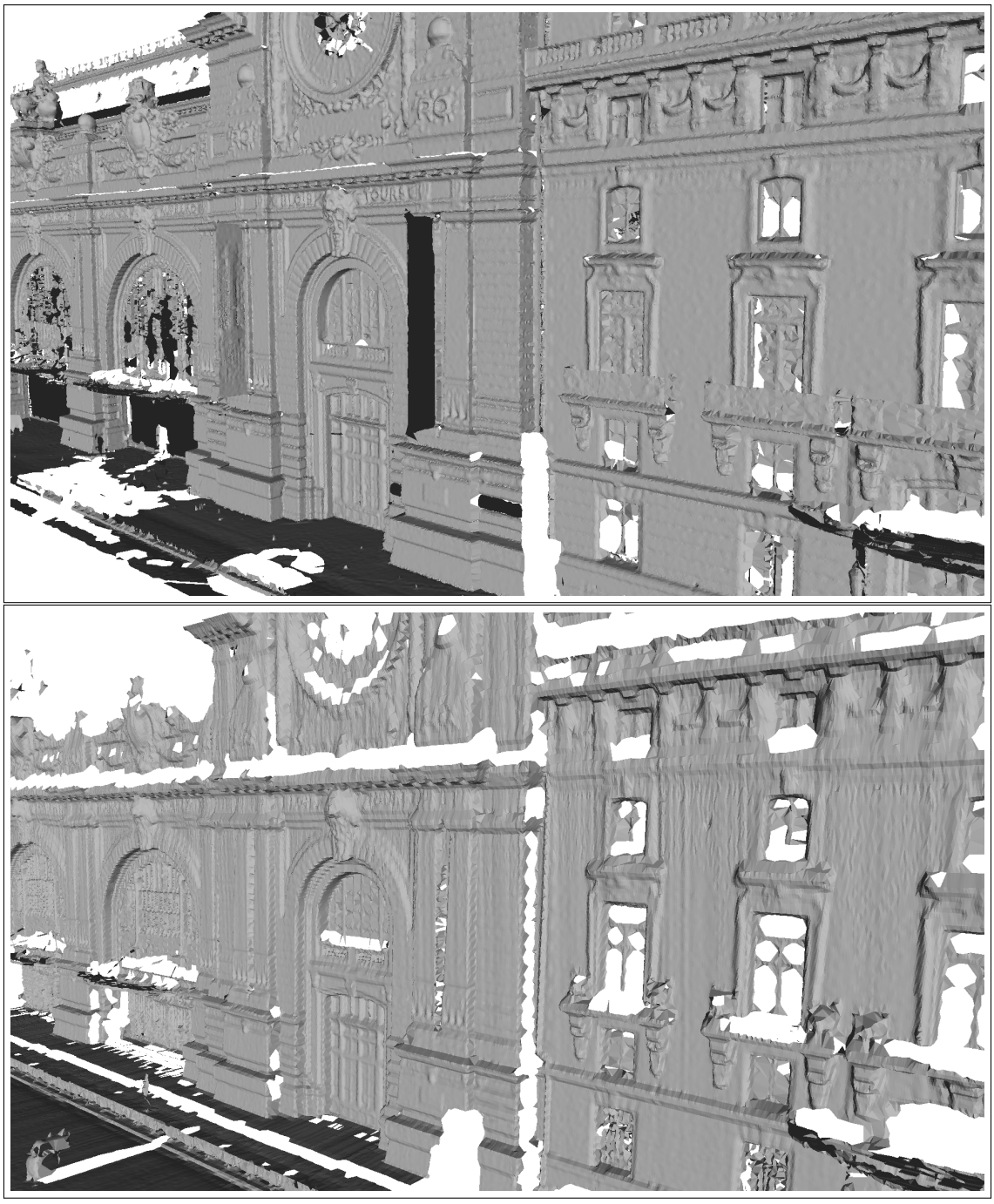


FIGURE 6.6 – En haut, maillage non simplifié du nuage *FX_GX_Orsay* à une résolution de 7 *cm*. En bas, maillage non simplifié du nuage *LARA3Dv2010_Orsay* à la même résolution.

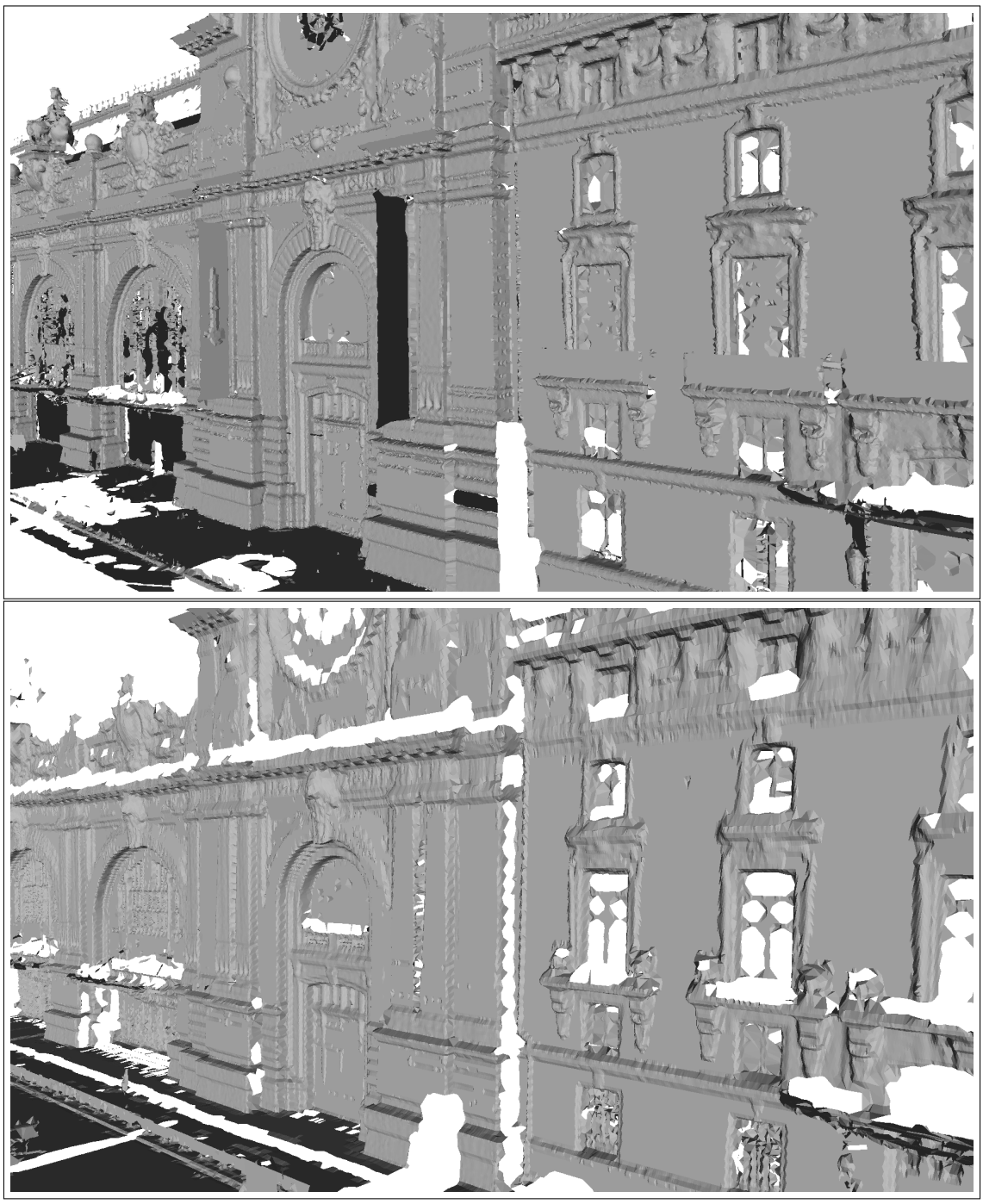


FIGURE 6.7 – En haut, maillage simplifié du nuage *FX_GX_Orsay* à une résolution de 7 *cm*. En bas, maillage simplifié du nuage *LARA3Dv2010_Orsay* à la même résolution.

6.2.2 Comparaison avec une édition manuelle

Nous avons comparé les résultats de modélisation automatique à un processus d'édition manuelle. Un infographiste 3D a modélisé une petite portion de la façade Nord du musée d'Orsay en partant des mêmes données initiales, c'est-à-dire du nuage de points *LARA3Dv2010_Orsay*. Ce travail a demandé 7h d'édition sur la portion présentée à la figure 6.8. Par extrapolation, la modélisation manuelle de toute la façade du nuage *LARA3Dv2010_Orsay* aurait demandé 35h. La modélisation manuelle de bâtiment est généralement très rapide mais dans le cas de modélisation avec une exactitude architecturale, le temps nécessaire est bien plus important puisqu'il faut se baser sur les nuages de points comme référence. En comparaison, le maillage par modélisation automatique de toute la façade a été obtenu en 15 *min*. Une telle différence dans le temps de traitement peut avoir beaucoup d'implications dans les nombreux domaines d'application de ces modèles.

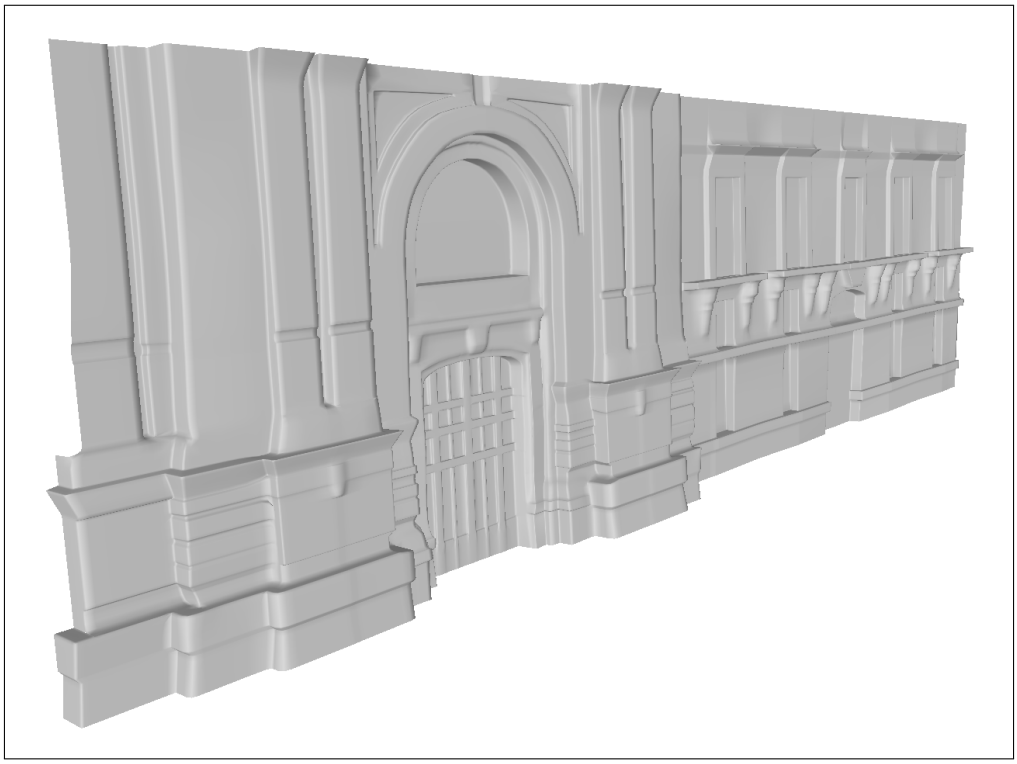


FIGURE 6.8 – Modélisation manuelle de la façade Nord du musée d'Orsay.

Les figures 6.9 et 6.10 montrent une comparaison entre la façade générée automatiquement et la façade modélisée manuellement. On y voit que l'aspect général entre les deux modèles reste très semblable même si moins de détails sont modélisés par l'approche manuelle. En revanche le maillage par édition manuelle est très régulier comparé au maillage obtenu après notre simplification automatique comme on le voit sur la figure 6.10. De plus, de nombreux trous subsistent dans notre maillage.

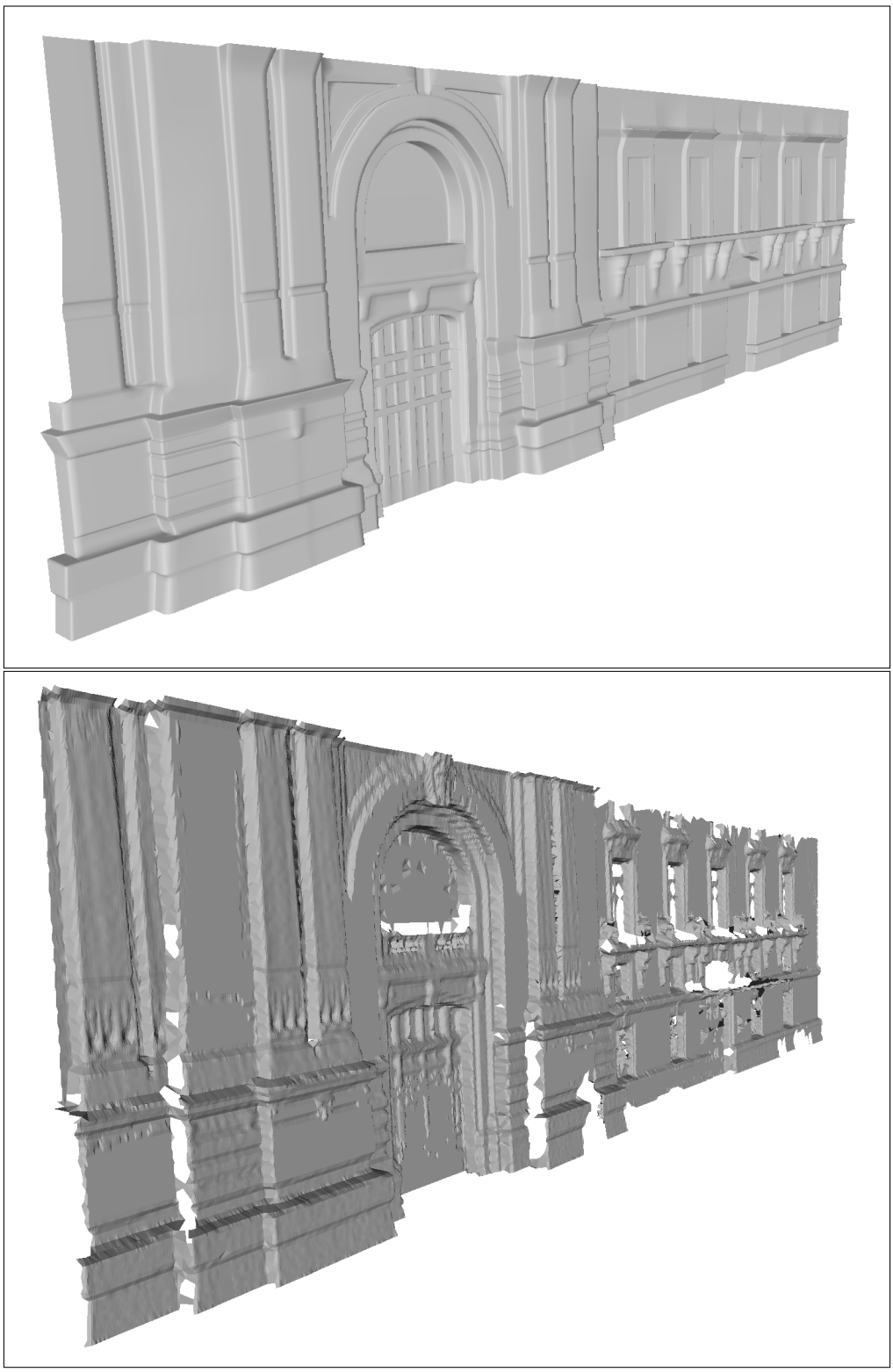


FIGURE 6.9 – En haut, maillage du nuage *LARA3Dv2010_Orsay* par édition manuelle. En bas, maillage simplifié du nuage *LARA3Dv2010_Orsay* par modélisation automatique à une résolution de 7 cm.

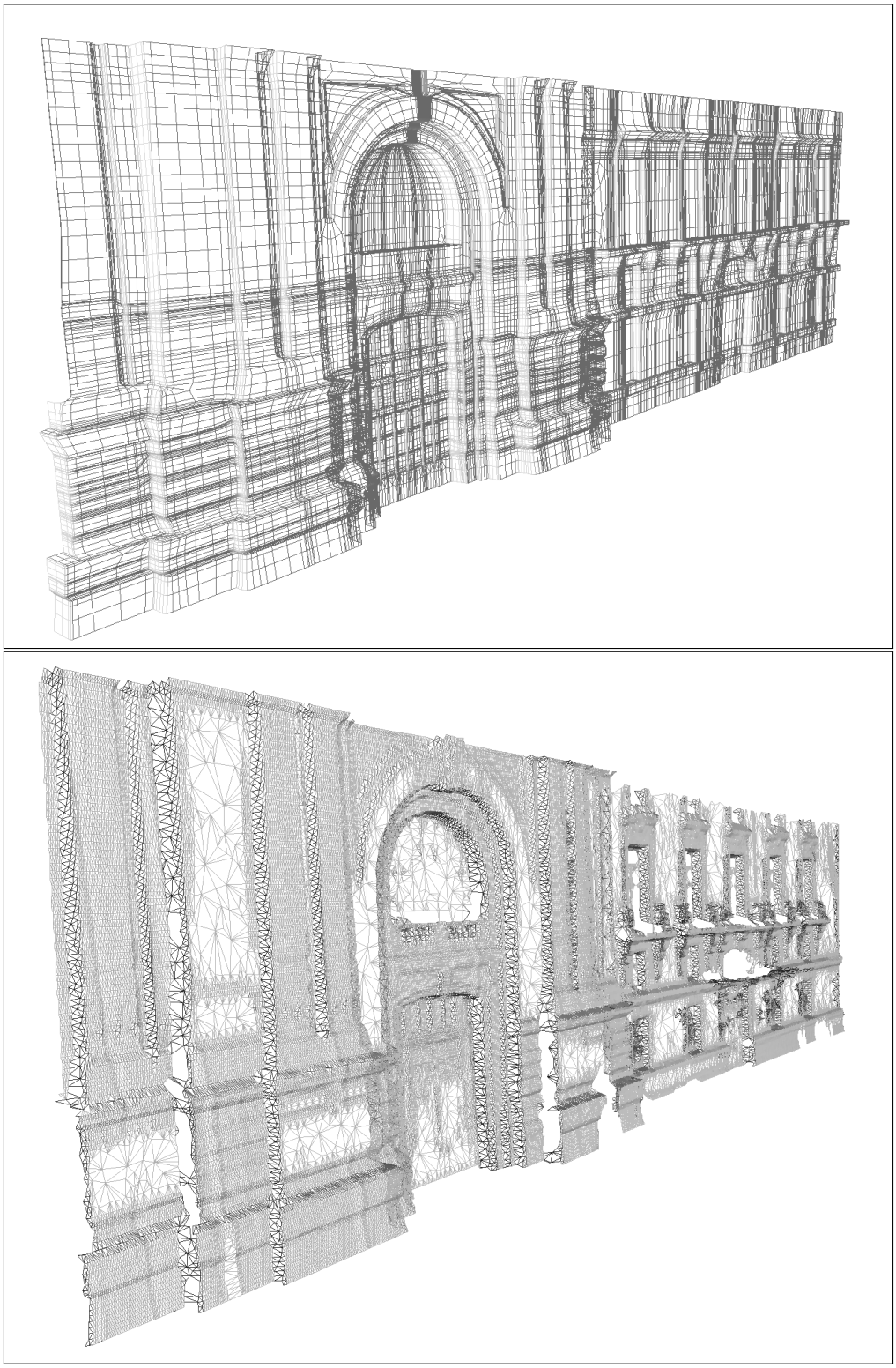


FIGURE 6.10 – En haut, maillage en fil de fer du nuage *LARA3Dv2010_Orsay* par édition manuelle. En bas, maillage simplifié en fil de fer du nuage *LARA3Dv2010_Orsay* par modélisation automatique à une résolution de 7 *cm*.

6.3 Modèles de routes et distances de visibilité

Dans le cadre du projet DIVAS, nous avons travaillé sur la génération de modèles 3D d'environnements routiers. L'objectif est d'utiliser ces modèles pour effectuer des calculs de visibilité géométrique sur route. Pour cela, nous avons eu deux campagnes d'acquisition en Février 2008 et en Mars 2010 qui ont porté sur le même itinéraire : la Route Départementale (RD) 786 dans les Côtes d'Armor. Sachant que le prototype LARA3D n'est pas encore opérationnel de façon robuste et sûre pour caractériser tout un itinéraire, il a été choisi de ne faire des relevés que sur une zone présentant des caractéristiques potentiellement intéressantes en termes de visibilité géométrique et d'accidentologie. Un autre critère de choix est la disponibilité de données de comparaison en termes de visibilité géométrique. Pour ces raisons, le Conseil Général des Côtes d'Armor (CG 22) a suggéré de retenir un tronçon situé à l'entrée d'Etables-sur-Mer, sur la RD 786, situé entre les Points de Repère (PR) 27 et 28 (figure 6.11). Nous avons effectué plusieurs séries d'acquisitions sur ce tronçon en 2008 et en 2010. Que ce soit en 2008 ou en 2010, les acquisitions ont été faites sur un seul tronçon de la RD786 dans plusieurs conditions : 20, 45 et 70 km/h et dans les deux sens à chaque fois, ce qui nous a permis de répondre à la fois à la demande de caractérisation 3D, et aussi nous a permis d'étudier l'impact de différentes conditions opérationnelles sur le résultat.

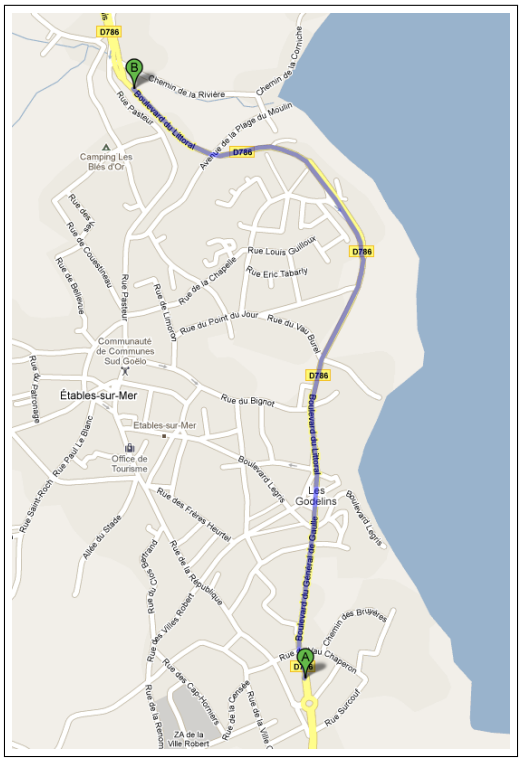


FIGURE 6.11 – Itinéraire du CG22 sur la RD786 à l'entrée d'Etables-sur-Mer (source Google Maps).

Dans le cadre du projet DIVAS, nous avons légèrement modifié notre approche sur le traitement des points pour la création de modèles 3D. Les problèmes rencontrés au niveau

de la modélisation sont principalement de deux sortes : la présence d'artefacts (voitures, camions ou autres obstacle mobiles sur la route lors de l'acquisition) et la taille trop importante des modèles pour les calculs de visibilité. Pour les artefacts dûs aux obstacles mobiles, nous avons développé une méthode de détection d'objets sur la route en utilisant la configuration des lasers. Pour diminuer la taille des modèles (environ deux millions de triangles par kilomètre de routes à partir du nuage de points brut), nous utilisons notre méthode de segmentation pour détecter les zones planes de l'environnement et ensuite les modéliser avec un minimum de triangles.

6.3.1 Suppression des artefacts

Lors de la numérisation des routes, le véhicule LARA3D roule en trafic normal. Les voitures dépassant ou roulant en sens inverse sont ainsi numérisées avec l'environnement. Or, si ces artefacts ne sont pas supprimés, ils sont modélisés en tant qu'objet de l'environnement et introduisent alors un biais pour le calcul des distances de visibilité.

Dans la création de modèle 3D d'environnements à partir d'une plateforme mobile, nous entendons comme artefact tout objet dynamique dans l'environnement (voitures, motos, camions, piétons en mouvements). Sachant que dans le cadre de DIVAS, nous numérisons des environnements routiers, nous nous sommes limités aux artefacts présents sur la route, c'est-à-dire principalement aux autres véhicules. Dans un cadre urbain, il faudrait supprimer tous les objets dynamiques aussi présents sur les trottoirs.

Nous pouvons voir des exemples d'artefacts présents dans les nuages de points bruts entourés en rouge sur la figure 6.12. Ce sont ici des voitures captées par le laser lors de la numérisation mais qui doivent être supprimées avant la modélisation pour permettre un calcul correct des distances de visibilité. Grâce à la position haute du laser (fixé en hauteur à environ 2,7 m du sol), nous pouvons minimiser l'effet d'occlusion de ces artefacts sur le reste de l'environnement et ainsi capter l'environnement situé derrière les véhicules.

Nous avons développé une méthode pour supprimer les artefacts présents sur la route. Pour cela, nous travaillons par profil laser (couche transversale à la route) avant notre chaîne de modélisation. Avec comme donnée la hauteur h du laser (la distance entre le point origine du laser et le sol) et les distances du véhicule aux bords de la route r_1 à droite et r_2 à gauche, nous segmentons automatiquement, dans le profil, les objets sur la route (Figure 6.13). Nous prenons ainsi comme hypothèse qu'aucun objet appartenant à l'environnement n'est présent sur les voies.

Les points laser des objets sur la route sont ainsi supprimés comme on le voit sur la figure 6.14. Notre méthode de suppression des artefacts nécessite la connaissance de la hauteur laser qui est une donnée fixe tout au long de l'expérimentation et donc mesurée une seule fois. Les deux autres paramètres r_1 et r_2 qui sont les distances du véhicule aux bords de la route à droite et à gauche sont dépendants du lieu d'expérimentation. Nous nous sommes restreints à une route de largeur fixe et avons admis l'hypothèse que le véhicule LARA3D roule au centre de la voie de droite. Avec une largeur de voie de

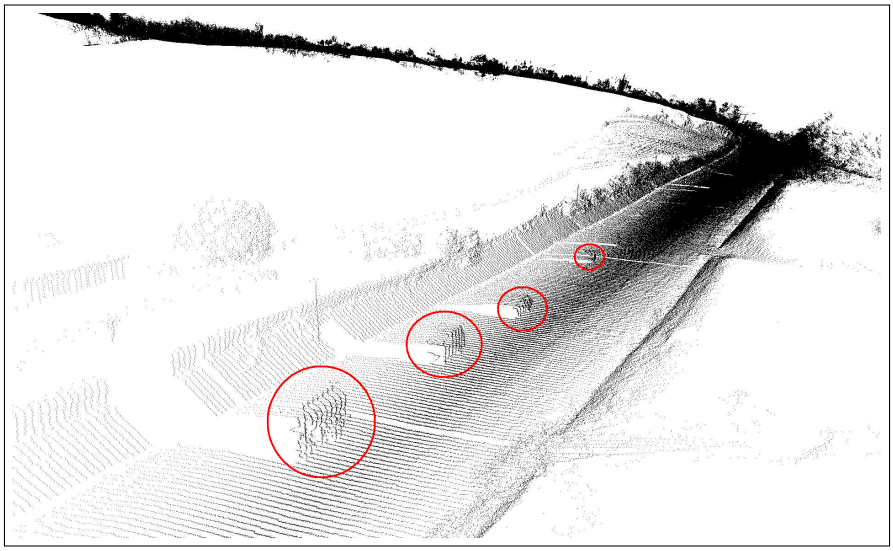


FIGURE 6.12 – Artefacts présents dans les nuages de points

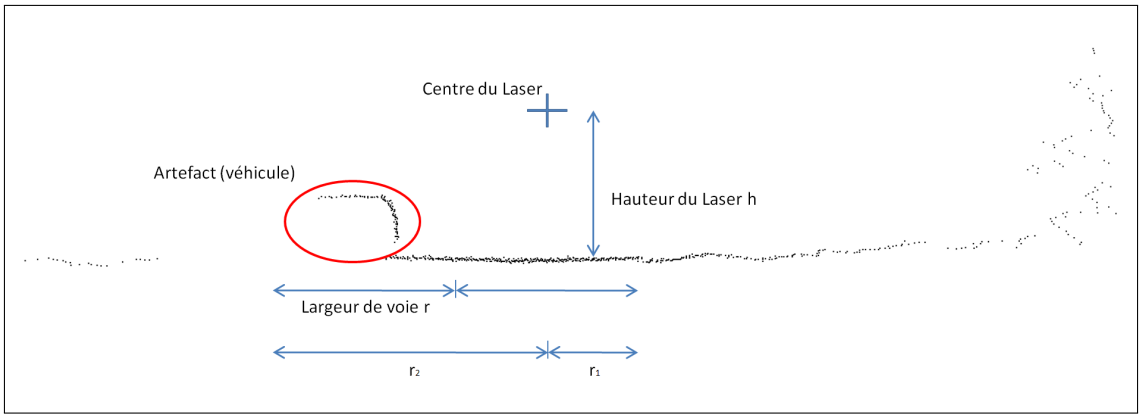


FIGURE 6.13 – Suppression des artefacts dans les profils laser

3.5 m pour les routes nationales, cela donne $r_1 = 1.75\ m$ et $r_2 = 5.25\ m$. Nous travaillons actuellement sur une méthode de détection automatique des deux paramètres r_1 et r_2 à partir d'une détection des bords de la route. Une autre limitation de cette méthode est d'être dépendante de la configuration expérimentale des lasers. La décimation ne fonctionne qu'avec des lasers qui donnent une coupe transversale de la route.

6.3.2 Génération de modèle 3D de routes

Une fois les artefacts supprimés dans un profil laser, nous créons le nuage de points 3D par géo-référencement des points laser comme détaillé au chapitre 1. Les coordonnées 3D de chaque point laser sont calculées à partir de ses coordonnées 2D dans un profil et de la localisation du véhicule au moment de la mesure. Nous appliquons alors notre chaîne de

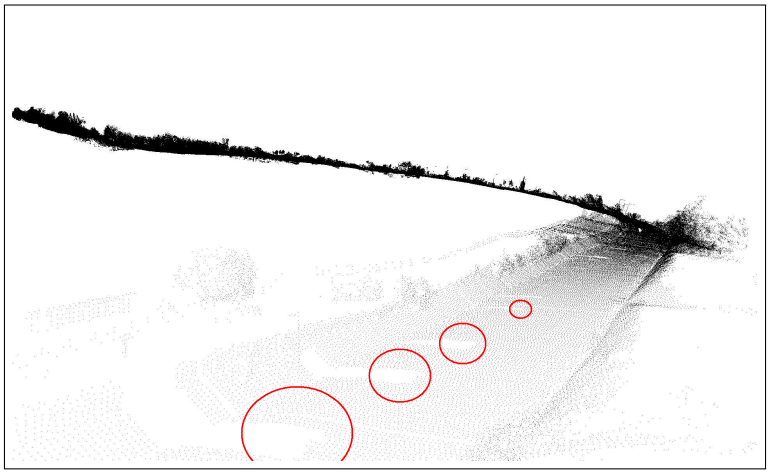


FIGURE 6.14 – Nuage de points après la suppression des artefacts

traitement : décimation, calcul de normales, débruitage, segmentation et triangulation.

Nous présentons ici les résultats sur le tronçon de la RD786 acquis en 2010. Il s'agit d'un parcours d'une longueur de 3 km, acquis en 4 min 41 s à la vitesse de 45 km/h. Le nuage de points *LARA3Dv2010_Etables* correspondant est composé de 3 540 911 points. Sachant que les temps de traitement du logiciel de calcul de visibilité dépendent fortement du nombre de triangles du maillage, nous avons choisi de construire le modèle 3D à faible résolution. C'est pourquoi nous avons choisi une distance $d_{details} = 30\ cm$, ce qui nous donne avec $d_{profil} = 27.0\ cm$ et $d_{trajectoire} = 16.6\ cm$, une distance de décimation : $d_{deci} = 30\ cm$. Le modèle 3D est ainsi généré à une résolution de 30 cm. Nous avons obtenu un maillage de 861 169 triangles en 12 min (environ 40% du temps réel).

Les différents paramètres utilisés pour la modélisation sont : $d_{details} = 30\ cm$, $\delta_{normal} = 52.8\ cm$, $\delta_{NLD3D} = 52.8\ cm$, $\Delta_{NLD3D} = 1.58\ m$, $\delta_{plan} = 1.0\ m$, $n_{min} = 100\ pts$, $n_{max} = 1000\ pts$, $d_{voxel} = 60\ cm$, $\gamma_{plan} = 10\ cm$, $\alpha_{plan} = 10.0\,^{\circ}$, $d_{maillage} = 1.3\ m$, $\theta_{simpli} = 1.0\,^{\circ}$.

La décimation a enlevé 2 480 563 points. Le nombre de triangles était de 1 290 651 avant la simplification et de 861 169 après. Le ratio de compression de la simplification par zones planes est d'environ 30% avec la détection des plans et la simplification. Nous obtenons un taux de compression inférieur à celui des scènes urbaines. Cela s'explique par le fait qu'un environnement rural dispose de moins d'éléments architecturaux plans qu'un environnement urbain. Finalement, nous générons un modèle de l'environnement routier avec environ 300 000 triangles par kilomètre au lieu de 2 000 000 de triangles par kilomètre en triangulant directement le nuage de points brut, la chaîne complète ayant permis de compresser les informations à hauteur de 85%.

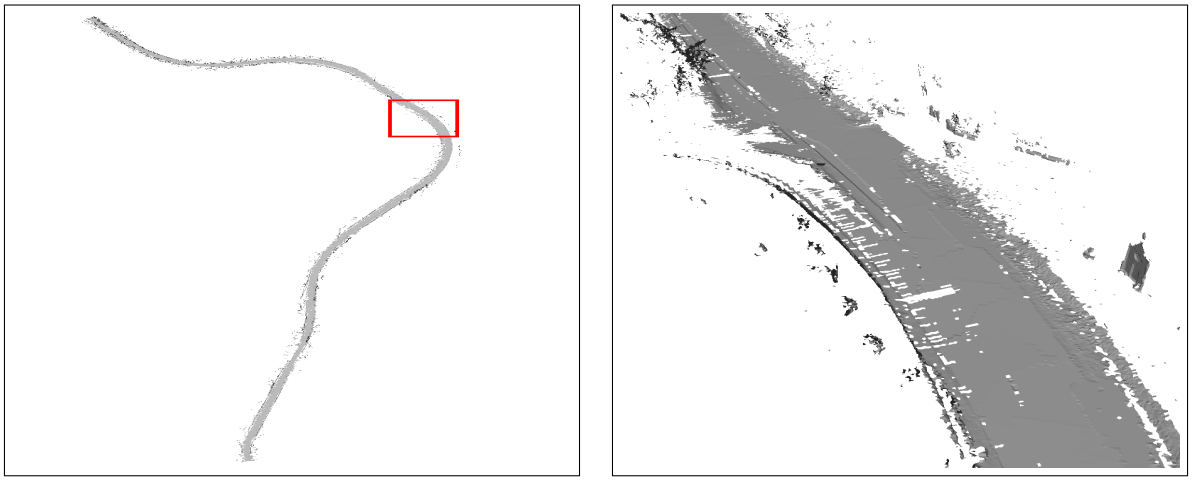


FIGURE 6.15 – A gauche, modèle 3D du nuage *LARA3Dv2010_Etables* à une résolution de 30 *cm* vu du haut. A droite, gros plan sur le maillage.

6.3.3 Génération de cartes de distance de visibilité

Le LRPC de Strasbourg a mis au point un logiciel de calcul, nommé « Qt-Ballad » qui exploite les modèles numériques route-environnement pour évaluer des distances de visibilité sur itinéraires [Charbonnier et al., 2007]. Ce logiciel permet de vérifier la disponibilité d'une visibilité minimale, la distance entre le point d'observation et la cible étant fixée à une valeur pré-définie. Il permet également l'évaluation de la distance de visibilité maximale disponible, en faisant varier la distance entre le point d'observation et la cible. Cette dernière peut être constituée d'une paire de points, représentant les feux arrières d'un véhicule, selon les définitions utilisées par les documents de référence en matière d'aménagement routier. Il peut également s'agir d'un parallélépipède représentant un véhicule ou un piéton. La distance de visibilité disponible ainsi évaluée peut être comparée avec la distance de visibilité requise, calculée par le logiciel à partir des caractéristiques de la trajectoire, et qui correspond à la distance d'arrêt sur obstacle.

A partir du modèle 3D triangulé, les distances de visibilité géométrique sont calculées avec le logiciel Qt-Ballad présenté à la figure 6.16 par un calcul de lancer de rayon entre l'observateur et l'objet cible. Une fois l'environnement modélisé, le logiciel Qt-Ballad permet de faire des calculs de distance de visibilité sur route d'un conducteur suivant différents scénarios. Il est par exemple possible de modifier les paramètres de l'observateur (décalage du bord droit et hauteur de vue), les paramètres de l'objet cible (objet sous forme de feux arrières, de parallélépipède...), de faire des calculs dans les deux sens et sur les deux voies du modèle 3D, etc. Dans le cadre de DIVAS, nous avons choisi trois scénarios pour l'observateur répondant à trois types de véhicule : une voiture, une moto et un bus.

- Voiture : hauteur des yeux : 1 m, décalage du bord droit : 2 m ;
- Moto : hauteur des yeux : 1.5 m, décalage du bord droit : 1.8 m ;
- Bus : hauteur des yeux : 1.9 m, décalage du bord droit : 2.2 m.

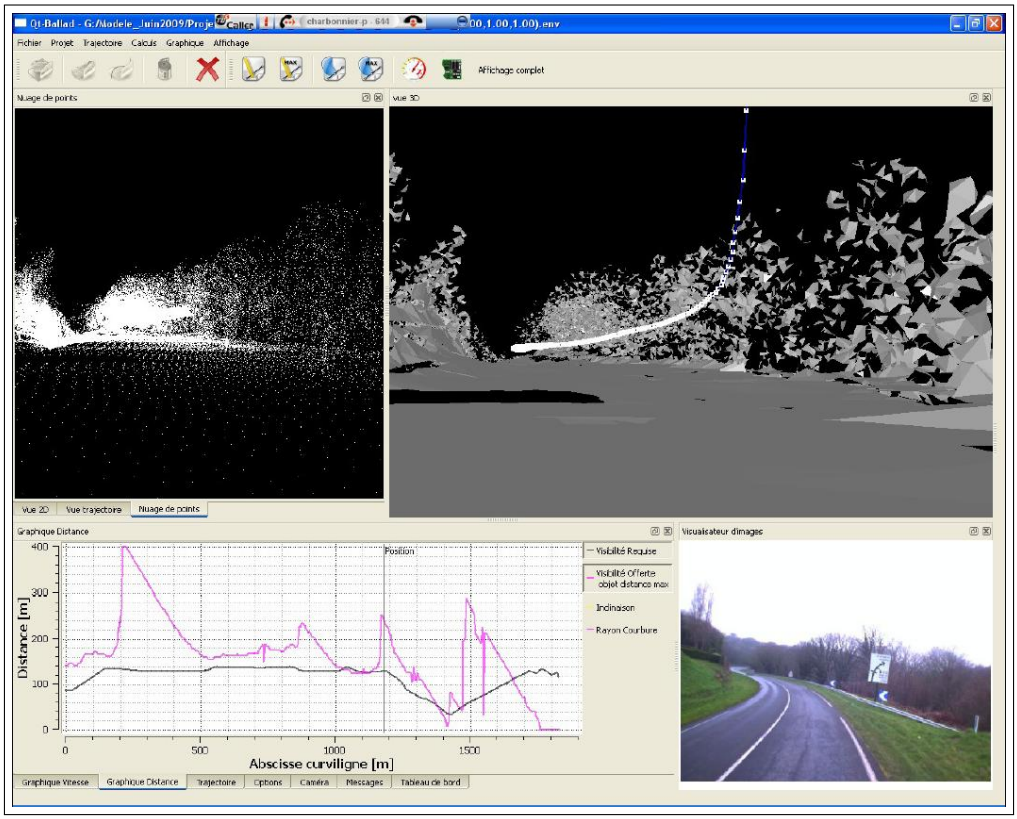


FIGURE 6.16 – Capture d'écran du logiciel Qt-Ballad (LRPC de Strasbourg) utilisé pour les calculs de visibilité.

La figure 6.17 présente les résultats de calculs de visibilité suivant les trois scénarios sur des acquisitions de 2008 à une vitesse d'acquisition de 45 km/h. Nous pouvons voir que les résultats sont très semblables avec une distance de visibilité plus grande pour le bus.

6.3.4 Comparaison des résultats avec une vérité terrain et conclusion

Les cartes de visibilité produites par LARA3D et QtBallad ont été comparées avec les données d'un autre système dédié à la mesure de distance de visibilité sur route appelé VISULINE. VISULINE est une méthode expérimentale de mesure de visibilité à partir de deux véhicules éloignés d'une distance fixe. Le deuxième véhicule note quand le premier véhicule disparaît du champ de vision. Les deux véhicules font plusieurs passages avec différentes distances séparant les voitures (100 m, 110 m, 120 m, etc.) et le résultat donne une carte de visibilité de la route. Nous pouvons considérer comme une vérité terrain les données produites par VISULINE.

La figure 6.18 montre les résultats des estimations de distances de visibilité pour des acquisitions faites en 2008 à une vitesse de 45 km/h et des acquisitions faites en 2010 aussi à une vitesse de 45 km/h et les compare avec les données de VISULINE sur le même parcours. Nous observons que les résultats sont tout d'abord cohérents entre les acquisitions

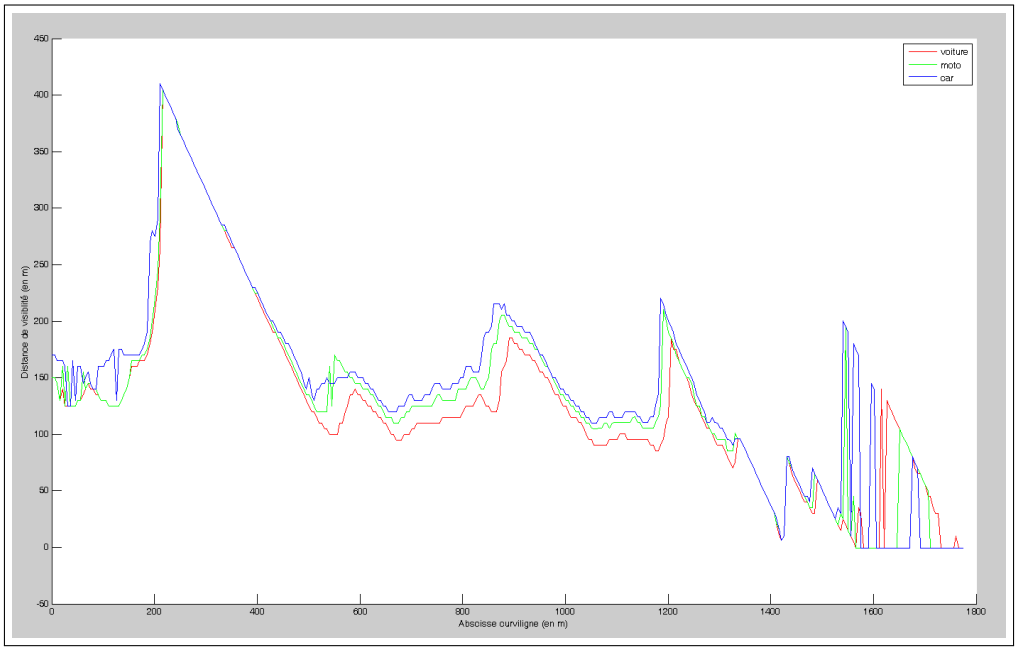


FIGURE 6.17 – Estimation de la distance de visibilité pour un itinéraire donné avec différents scénarios, en utilisant les acquisitions de 2008.

de 2008 et de 2010. Nous remarquons aussi que les distances calculées sont proches des distances obtenues par le système VISULINE. Nous notons même une amélioration sur les acquisitions de 2010 qui résulte probablement de l'amélioration du système LARA3D.

La Figure 6.19 montre les résultats des estimations de distances de visibilité pour différentes vitesses de passage en 2010 (45 et 70 km/h) comparés à VISULINE. Nous voyons que nous obtenons un résultat presque meilleur à une vitesse de 70 km/h. Cela peut s'expliquer par le fait qu'à 70 km/h, nous avons beaucoup moins d'artefacts (voitures qui doublent) dans le modèle 3D généré.

Ces résultats prouvent l'intérêt de la méthode proposée. Des méthodes optiques ont été testées pour le calcul de distances de visibilité à partir d'images mais elles échouent sur le problème de la détection et de la suppression automatique des artefacts (principalement les véhicules sur la route). Contrairement à ces dernières, notre méthode peut résoudre cette difficulté grâce aux lasers comme cela a été montré. Nous obtenons des résultats qui s'approchent de ceux obtenus par une vérité terrain permettant de valider les résultats.

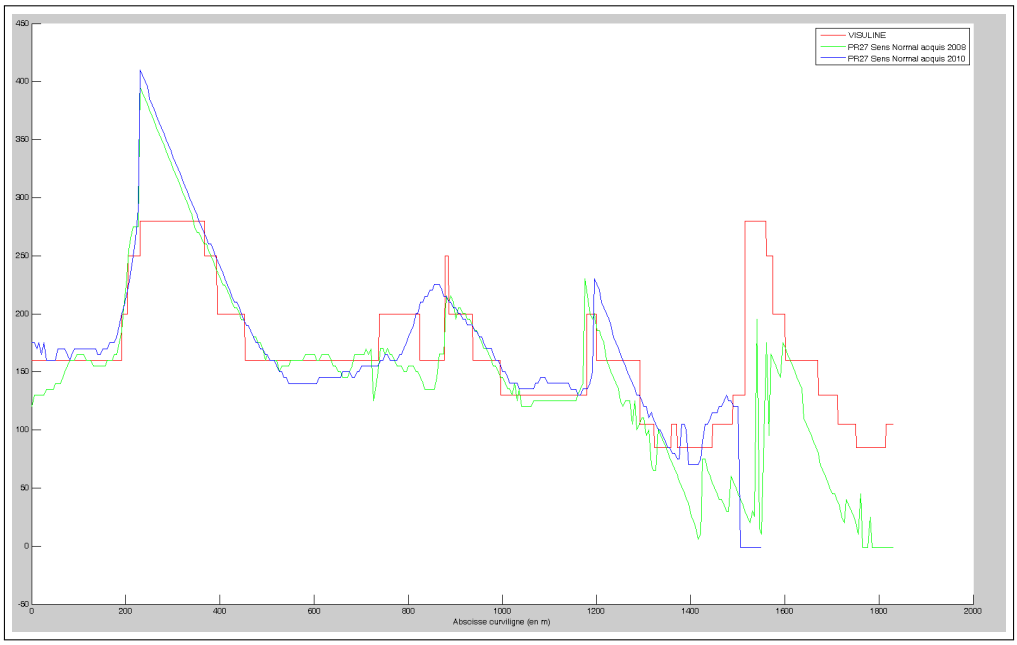


FIGURE 6.18 – Comparaison des estimations de distances de visibilité entre les acquisitions de 2008 (en vert) et de 2010 (en bleu) avec des données vérités terrains (en rouge).

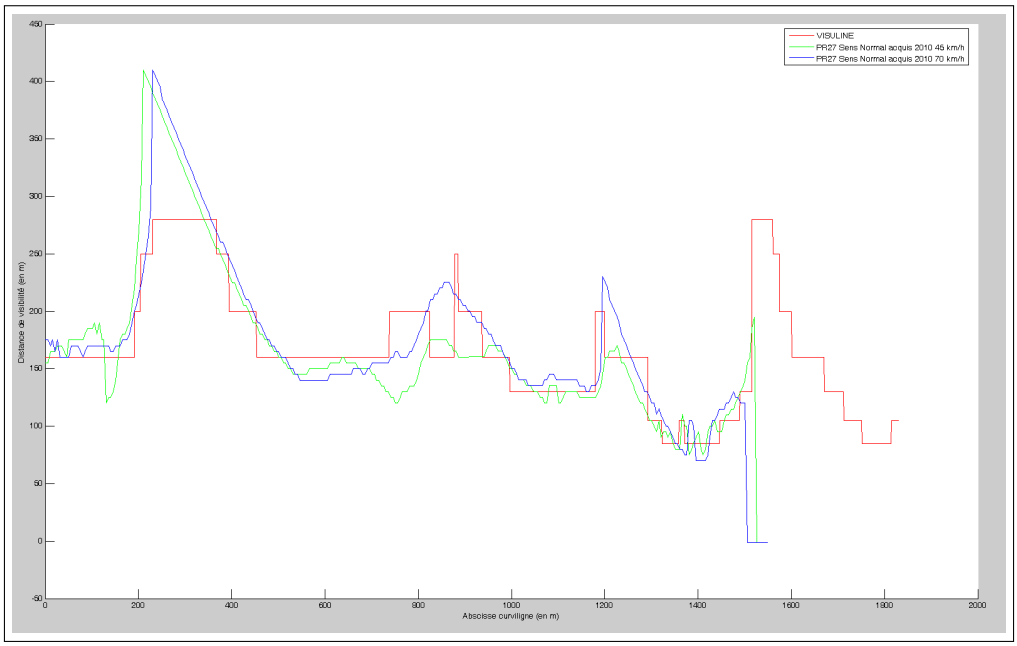


FIGURE 6.19 – Comparaison des estimations de distances de visibilité à différentes vitesses de passage pour les acquisitions de 2010.

6.4 Génération de modèles 3D d'intérieurs

Afin d'éprouver la généricité de notre méthode de modélisation, nous l'avons aussi testée sur les nuages de points provenant du robot MINES_Rover, figure 6.20. La difficulté de ce type de données acquis en mode « *Stop&Go* » est la très grande hétérogénéité des données, le bruit, ainsi que les nombreux trous comme nous pouvons le voir sur un détail du nuage de points, sur la figure 6.21.

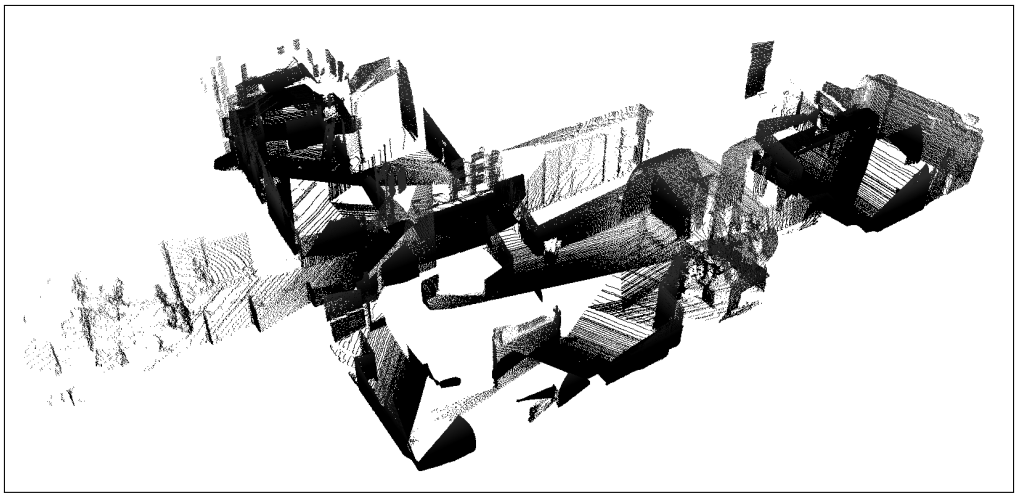


FIGURE 6.20 – Nuage de points *MINES_Rover_Indoor* d'un environnement d'intérieur obtenu par les scans recalés du robot MINES_Rover.

Notre objectif est de modéliser les murs des bâtiments en retrouvant les angles et en conservant les objets présents dans les salles. Le nuage de points testé, *MINES_Rover_Indoor*, est composé de 2 836 369 points. Nous avons choisi une distance $d_{details} = 5\ cm$ pour rendre le nuage plus homogène ce qui implique une distance de décimation $d_{deci} = 5\ cm$. Pour ce modèle dont la technologie d'acquisition est différente, nous n'avons pas pris les paramètres en fonction de d_{mean}. Ces derniers ont été trouvés après plusieurs expérimentations : $d_{details} = 5\ cm$, $\delta_{local} = 15\ cm$, $\delta_{NLD3D} = 15\ cm$, $\Delta_{NLD3D} = 50\ cm$, $\delta_{plan} = 30\ cm$, $n_{min} = 100\ pts$, $n_{max} = 100\ 000\ pts$, $d_{voxel} = 50\ cm$, $\gamma_{plan} = 10\ cm$, $\alpha_{plan} = 30.0\,°$, $d_{maillage} = 30\ cm$, $\theta_{simpli} = 1.0\,°$.

Le temps de modélisation total du nuage est de 4 *min* avec une étape de décimation qui a enlevé 2 745 059 points. La figure 6.22 présente sur l'image du haut le maillage du nuage de points sans débruitage et sur l'image du bas le maillage après débruitage mais sans simplification. Les différences visuelles entre les deux maillages montrent l'importance de l'étape de débruitage.

La figure 6.23 montre l'étape de détection de plans et de projection des points sur leurs plans. Tout le sol a été détecté comme un seul plan grâce à notre algorithme de croissance par voxels. Nous voyons sur la deuxième image la projection des points sur leurs plans avec en vert les points appartenant à deux plans et en rouge les points appartenant à trois

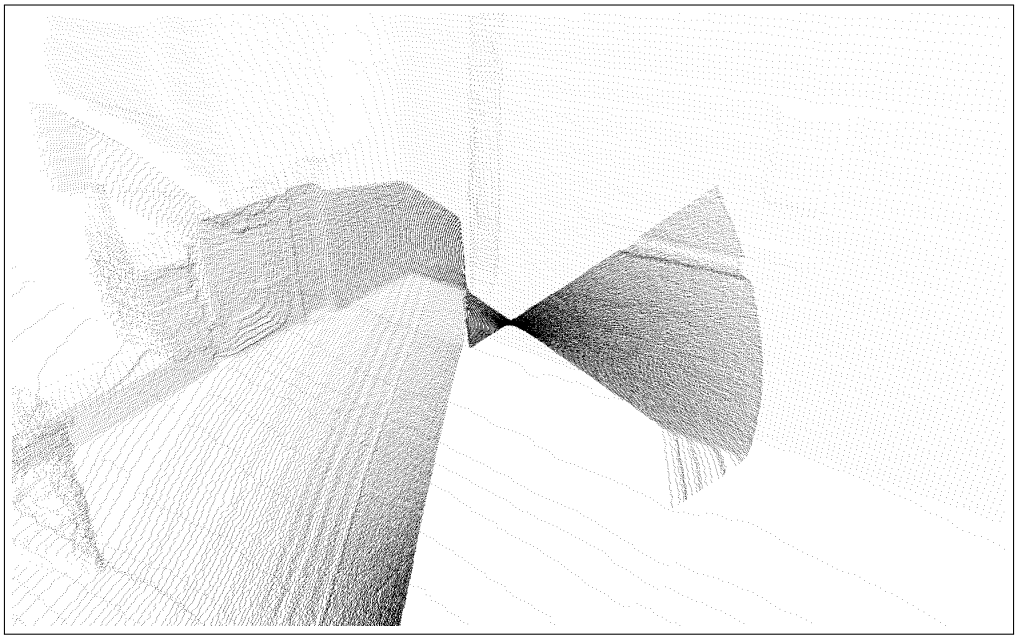


FIGURE 6.21 – Détail du nuage de points *MINES_Rover_Indoor* d'intérieur.

plans. Ces points vont être projetés sur leurs plans respectifs et vont permettre de retrouver les angles des murs. Nous voyons le résultat sur la dernière image avec des murs qui sont maintenant plans, et des angles reconstruits. La figure 6.24 illustre le maillage en fil de fer du nuage *MINES_Rover_Indoor* avant et après simplification.

La figure 6.24 présente la dernière étape de simplification des zones planes avec la comparaison des maillages avant et après simplification. Le maillage était composé de 122 791 triangles avant la simplification et de 27 205 triangles après. Nous avons un fort ratio de compression du maillage de 78%, ce qui semble logique au vu de l'environnement composé de nombreux plans.

Finalement, nous obtenons une modélisation de l'environnement avec des murs plans, en retrouvant les angles et un maillage simplifié composé de seulement 27 205 triangles pour représenter l'intérieur du bâtiment. Notre résultat se rapproche des travaux de [Budroni et Bohm, 2009] pour la modélisation d'environnements d'intérieur sauf que leur maillage est simplifié au maximum puisqu'ils ne modélisent que les plans comme nous l'avons vu au chapitre 1 (figure 1.25).

6.5 Conclusion

Ces résultats montrent que notre méthode de modélisation s'adapte à de nombreuses techniques d'acquisition de données et peut être utilisée pour des scènes urbaines, routières mais aussi d'intérieur générées par des systèmes mobiles et fixes.

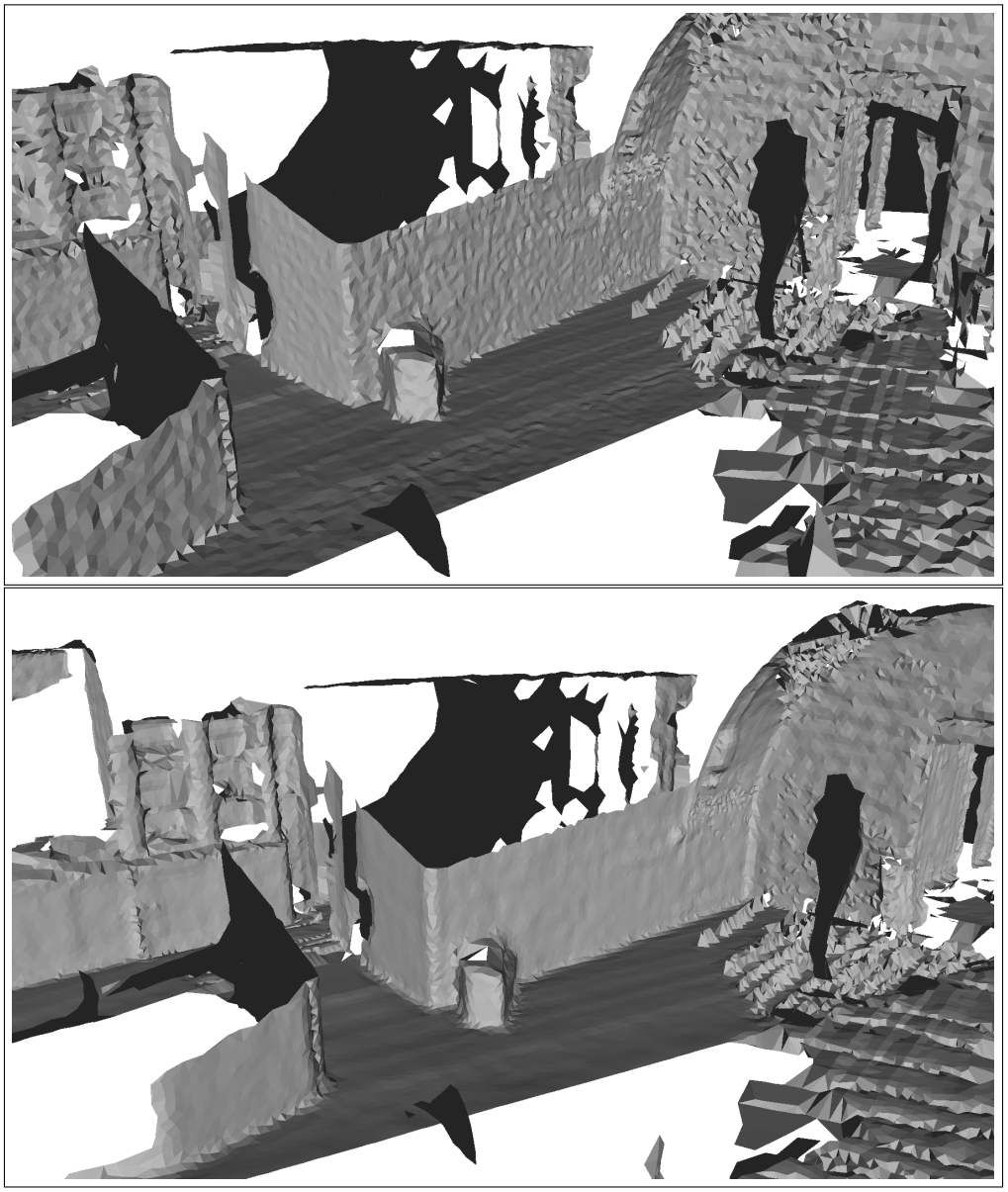


FIGURE 6.22 – En haut, maillage du nuage *MINES_Rover_Indoor* sans débruitage. En bas, maillage du même nuage après débruitage sans simplification.

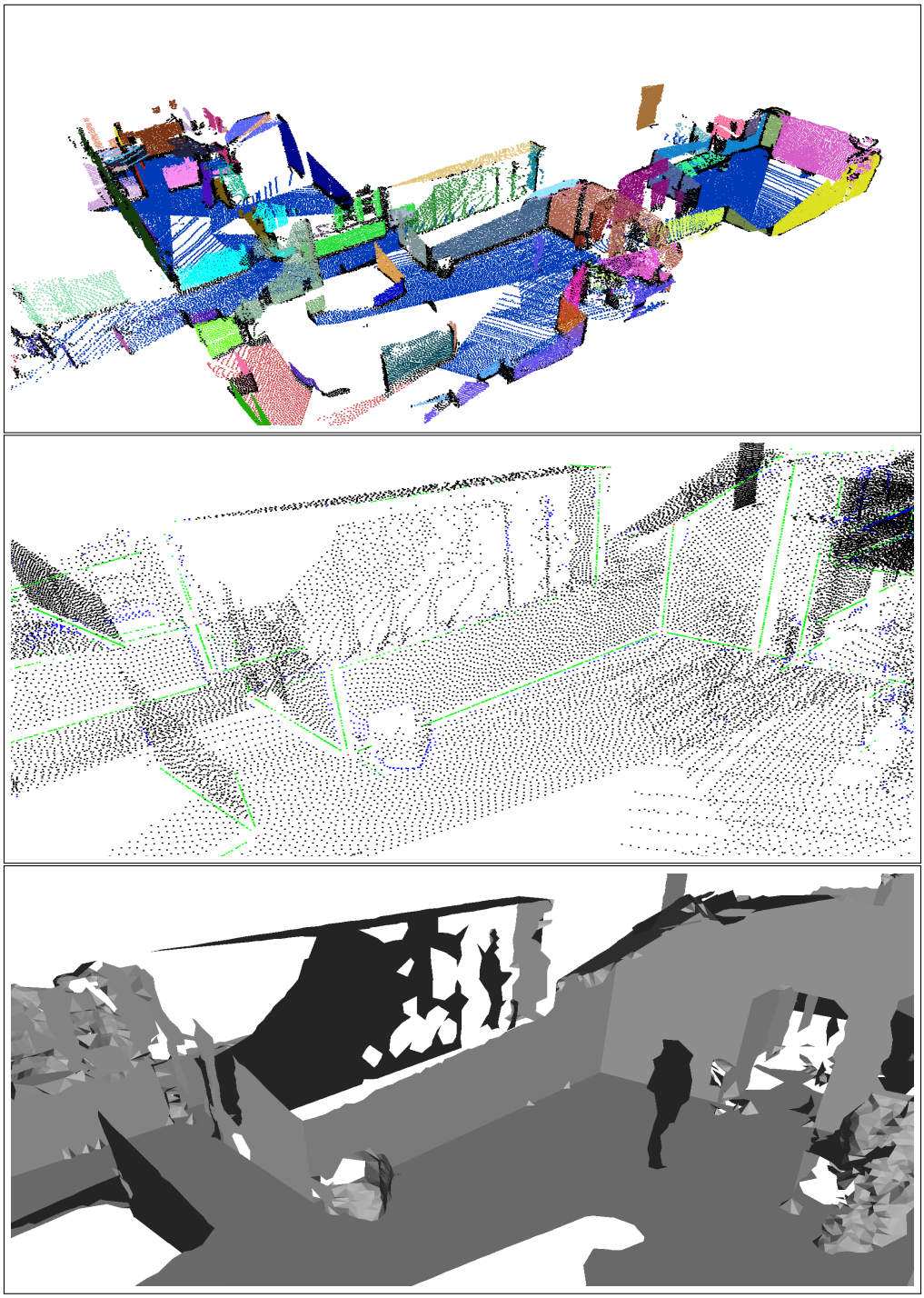


FIGURE 6.23 – En haut, détection des plans dans le nuage *MINES_Rover_Indoor*. Au milieu, projection des points sur leurs plans (en bleu les points de bordure, en vert les points d'intersection de deux plans et en rouge intersection de trois plans). En bas, maillage après l'étape de détection de plans, de projection des points et de simplification à une résolution de 5 *cm*.

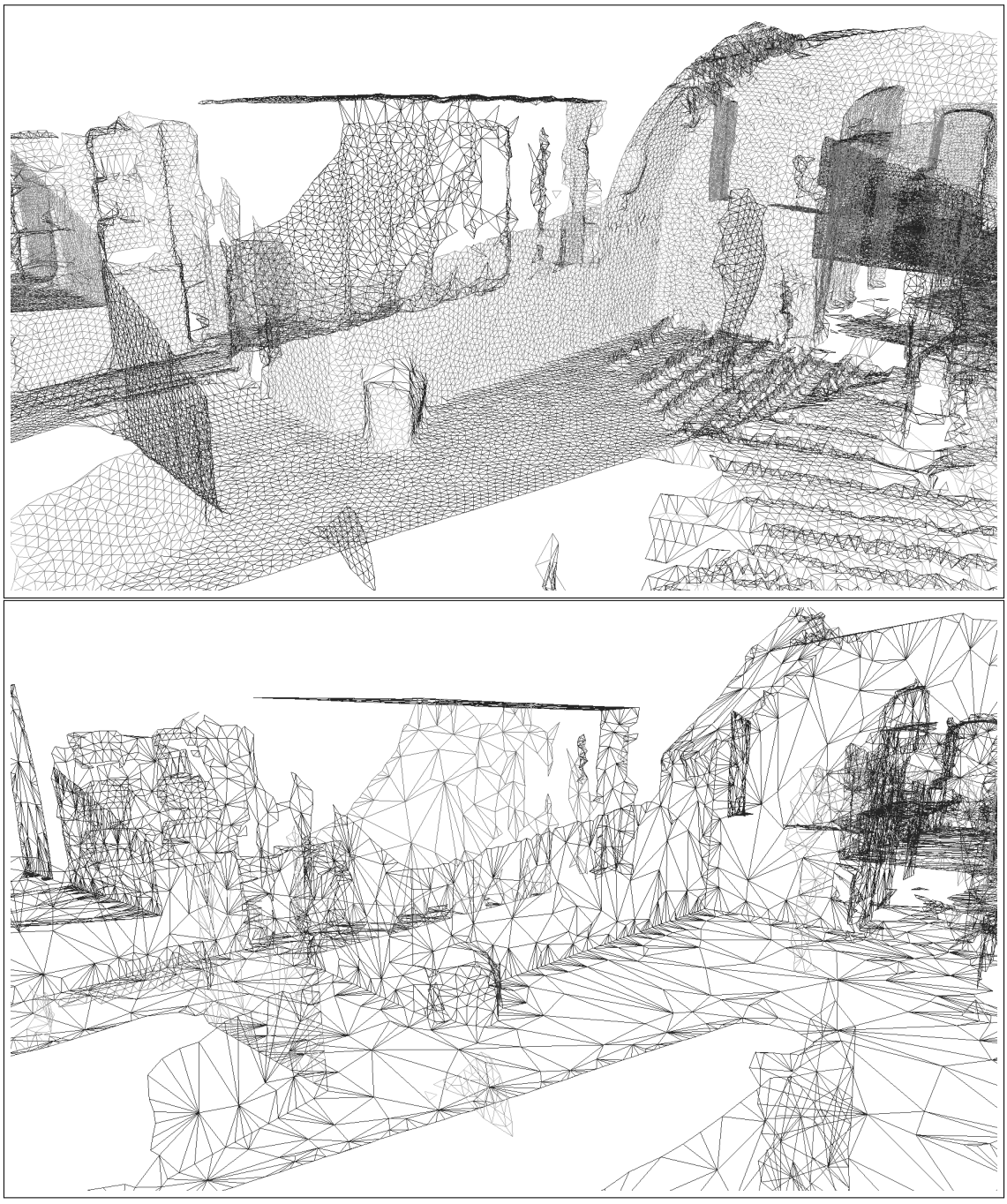


FIGURE 6.24 – En haut, maillage du nuage *MINES_Rover_Indoor* sans simplification à une résolution de 5 *cm*. En bas, même maillage après l'étape de détection et de simplification des zones planes.

Conclusion

Nous avons exposé dans ce mémoire un ensemble de techniques permettant de construire un modèle 3D texturé d'environnements terrestres. Dans un premier temps, nous avons vu les différentes technologies d'acquisition et principalement les systèmes mobiles basés LiDAR comme le prototype LARA3D de MINES ParisTech, la plateforme Stereopolis de l'IGN et le robot MINES_Rover. L'analyse des caractéristiques des nuages de points produits par ces plateformes mobiles nous a permis de constater que ces nuages sont denses et comportent peu ou pas de points aberrants. Ils sont aussi très hétérogènes avec des densités très variables et anisotropes à cause de la technologie d'acquisition. Enfin, malgré le fait que les données soient acquises en mouvement, le bruit reste relativement faible, généralement de l'ordre de 5 à 20% de la distance moyenne entre points. Toutes ces remarques ont eu une grande importance dans le développement des méthodes de traitement de nuages de points.

Ces données 3D servent d'entrée pour une série de traitements afin de construire un modèle 3D de l'environnement. La transformation de nuage en modèle 3D, appelée modélisation, peut comme nous l'avons vu, présenter de nombreuses formes selon l'application. Dans notre cas, nous souhaitons représenter des environnements à la fois ruraux et urbains tout en minimisant la quantité d'information à conserver, de manière à pouvoir créer des bases de données 3D sur de grandes étendues. Nous avons donc choisi de développer une modélisation hybride à l'aide d'une segmentation de l'environnement en zones planes et non planes.

Le premier traitement que nous avons vu au début du chapitre 3 consiste en la décimation du nuage de points en entrée de manière à réduire la taille de ce dernier. Plutôt que de faire une décimation aléatoire, nous avons ajouté une phase de sélection des points qui maximisent une approximation de la courbure locale. Nous avons ainsi une étape de décimation très rapide avec une meilleure conservation des variations locales du nuage comparé à une décimation aléatoire.

Ensuite, nous effectuons un calcul des normales en chaque point. Pour cela, nous calculons une surface locale MLS approximant la géométrie locale du point. La normale en un point est alors la normale à la surface. Pour tenir compte des voisinages contenant des points de discontinuité, nous avons introduit un filtrage du voisinage local afin d'enlever les points aberrants dans le calcul de normales, c'est pourquoi nous avons appelé notre méthode Filtered MLS. Nous avons testé et comparé notre méthode FMLS avec d'autres méthodes et les résultats montrent une amélioration sur l'erreur de l'estimation de la normale allant de 50% à 100% par rapport à une méthode classique.

Après le calcul de normales, les nuages de points sont débruités. Nous avons exposé notre méthode NLD3D de débruitage basée sur la comparaison et la similarité de voisinages locaux de points. Les tests ont montré que pour un bruit faible ou moyen comme pour les nuages de points de LiDAR, notre méthode offre de meilleurs résultats comparée aux méthodes existantes, en particulier dans les zones de forte auto-similarité.

Pour la segmentation, nous avons exposé un algorithme de détection de plans par croissance de voxels en axant la réflexion sur la nécessité d'utiliser des normales pour améliorer significativement les résultats de la segmentation. Pour la triangulation, nous avons développé une méthode rapide de triangulation donnant de meilleurs résultats sur les nuages testés par rapport aux méthodes existantes.

Tout au long de cette thèse, nous avons mis en place des techniques de traitement de nuages de points qui ne s'appliquent pas seulement aux nuages de points d'environnements extérieurs par système mobile mais à tout nuage de points dense. Plus particulièrement, les algorithmes de calcul de normales, de débruitage, de segmentation et de triangulation ont été conçus de manière suffisamment générique de sorte à être applicable à de nombreux nuages de points denses quel que soit le domaine d'application.

Dans le cadre de notre application, nous avons ajouté à ce maillage des textures en utilisant directement les images brutes de l'acquisition. Il s'agit d'une approche hybride entre de la modélisation et de la visualisation avec une géométrie de la surface représentée par le maillage et des images brutes plaquées directement dessus sans aucune perte d'information.

Les techniques présentées dans cette thèse ont été appliquées à la génération de modèles 3D. Tout d'abord, nous avons construit des modèles 3D texturés de scènes urbaines à partir de données de LARA3D. La comparaison de nos modèles avec un modèle construit par plateforme fixe nous a permis de constater que les résultats sont visuellement presque similaires. Nous avons aussi généré des modèles de scènes routières pour des estimations de distances de visibilité. Dans cette application, le système LARA3D couplé à un logiciel d'exploitation des modèles fournit des résultats de distance de visibilité comparables aux données fournies par une vérité terrain. La généricité de nos techniques nous a permis d'utiliser la même chaîne de modélisation pour produire des modèles 3D de scène d'intérieur à partir de données d'un robot mobile.

Nous avons ainsi mis au point une chaîne complète de modélisation et l'avantage majeur de notre travail est d'être générique, c'est-à-dire capable de produire des modèles avec différentes technologies d'acquisition (système mobile en mode continu, système mobile en mode « *Stop & Go* », scanners fixes) et pour différentes applications (urbain, rural, scènes d'intérieur). Avec notre processus de modélisation, nous arrivons à traiter les données de la plateforme LARA3D produisant 30 000 pts/s à 20% du temps-réel.

Avec les résultats obtenus, il est possible d'imaginer plusieurs pistes d'amélioration. Outre l'amélioration individuelle de chacune des techniques de la chaîne de modélisation,

il est possible d'améliorer la cohérence du modèle final notamment avec le recalage des données images et laser comme nous l'avons évoqué au chapitre 5. Enfin, le point essentiel restant encore à améliorer concerne la rapidité des algorithmes d'autant que l'évolution rapide de la technologies des scanners entraîne un besoin de production de données à grand volume. Une piste envisageable consisterait ainsi à implémenter tous les algorithmes présentés sur GPU.

A la vue des résultats et des temps de calcul obtenus à ce jour, il semble possible d'imaginer dans un avenir proche l'apparition de systèmes complets mobiles LiDAR/Caméra, intégrant des traitements de nuages de points denses embarqués, permettant la création automatique de bases de données 3D.

Glossaire

Area_Tri : Calcul des normales par la moyenne des normales des triangles voisins (pondération par l'aire des triangles).

Bilatéral : Filtre bilatéral qui peut être adapté pour le calcul des normales ou le débruitage de nuages de points [Fleishman et al., 2003].

BPA : Ball Pivoting Algorithm. Méthode de triangulation de nuages de points de [Bernardini et al., 1999].

FMLS : Filtered Moving Least Squares. Méthode de calcul de normales décrite au chapitre 3.

GOPI : Algorithme de maillage de [Gopi et al., 2000] par triangulation de Delaunay dans le plan tangent.

IRLS : Iteratively Reweighted Least Squares. Schéma itératif de calcul de normales de [Oztireli et al., 2009].

LARA3D : LA Route Automatisée 3D. Véhicule prototype du CAOR équipé de capteurs de positionnement (GPS, centrale inertielle) et de capteurs de modélisation (lidars, caméras).

LiDAR : Light Detection And Ranging. Technologie de mesure des propriétés de la lumière réfléchie par une cible pour obtenir des informations (souvent la distance) sur cette cible.

MINES_Rover : Robot mobile du CAOR utilisant la technique « *Stop&Go* ».

MLS : Moving Least Squares. Méthode d'approximation du voisinage local de [Levin, 2003] par une surface polynomiale.

NLD : Non Local Denoising. Technique de débruitage d'images de [Buades et Morel, 2005].

NLD3D : Non Local Denoising 3D. Débruitage de nuages de points décrite au chapitre 3.

Poisson : Méthode de maillage de [Kazhdan et al., 2006] basée sur l'extraction d'une fonction indicatrice et la résolution de l'équation de Laplace-Poisson.

PF : Plane Fitting. Méthode de calcul de normales de [Hoppe et al., 1992] par ajustement local de plans .

RIMLS : Robust Implicit Moving Least Squares. Méthode de maillage de nuage de points de [Oztireli et al., 2009] incluant un calcul de normales et le débruitage du nuage de points.

Stereopolis : Véhicule prototype de l'IGN équipé de capteurs de positionnement (GPS, centrale inertielle) et de capteurs de modélisation (lidars, caméras).

WPF : Weighted Plane Fitting. Méthode de calcul de normales de [Pauly et al., 2003] par ajout de poids aux points du voisinage.

Publications

Publications dans des revues avec comité de lecture

[$RFPT, 2010$] Deschaud, J.-E., Brun, X. et Goulette F., 2010. Colorisation et texturation temps réel d'environnements urbains par système mobile avec scanner laser et caméra fish-eye. Revue Française de Photogrammétrie et de Télédétection (RFPT). Numéro 192, pp. 29-37.

[$IJPRS$] Deschaud, J.-E. et Goulette, F. LiDAR Point Cloud Denoising using local geometric self-similarities. International Journal of Photogrammetry and Remote Sensing (IJPRS), **En cours de soumission**.

Publications dans des conférences avec actes

[$AFIG, 2007$] Deschaud, J.-E., Brun X. et Goulette, F., 2007. Texturation d'environnements urbains par système mobile avec un couplage Caméra/Télémètre Laser. Journées de l'Association Francophone d'Informatique Graphique (AFIG), Marne la Vallée, France.

[$MMT, 2007$] Brun, X., Deschaud, J.-E. et Goulette, F., 2007. On-the-way City Mobile Mapping Using Laser Range Scanner and Fisheye Camera. In : 5th International Symposium on Mobile Maping Technology (MMT), Padova, Italie.

[$CMOI, 2010$] Goulette, F., Deschaud, J.-E. et Senpauroca, J., 2010. Modélisation 3D de parcours routiers par relevé laser mobile pour la mesure de visibilité. Colloque du Club « Contrôles et Mesures Optique pour l'Industrie » (CMOI), Toulouse, France.

[$3DPVT, 2010$] Deschaud, J.-E. et Goulette, F., 2010. A Fast and Accurate Plane Detection Algorithm for Large Noisy Point Clouds. In : 3D Processing, Visualization and Transmission (3DPVT), Paris, France.

[$PCV, 2010$] Deschaud, J.-E. et Goulette, F., 2010. Point Cloud Non Local Denoising using Local Surface Descriptor Similarity. In : Photogrammetric Computer Vision and Image Analysis (PCV), ISPRS Commission III Symposium, Paris, France.

[$PRAC, 2010$] Senpauroca, J., Deschaud, J.-E. et Goulette, F., 2010. Mise en œuvre d'une plateforme expérimentale mobile de relevé laser pour le calcul de distance de visibilité. Conférence Sécurité Routière : Prévention des Risques et Aides à la Conduite (PRAC),

Paris, France.

Séminaires sans actes

[$JRSIG, 2008$] Deschaud, J.-E., Brun, X. et Goulette, F., 2008. Texturation d'environnements urbains par système mobile avec un couplage Caméra/Télémètre Laser, Journée Robotique et Systèmes d'Information Géographiques (JRSIG), GDR Robotique, Amiens, France.

[$JV, 2009$] Deschaud, J.-E. et Goulette, F., 2009. Modélisation d'environnements routiers par système mobile pour le calcul de distances de visibilité, Journée Visibilité (JV), GDR Robotique, Paris, France.

[$SFPT, 2009$] Deschaud, J.-E., Brun, X. et Goulette, F., 2009. Colorisation et texturation d'environnements urbains par système mobile avec scanner laser et caméra. Colloque Techniques Laser pour l'étude des environnements naturels et urbains, Société Française de Photogrammétrie et de Télédétection (SFPT), Le Mans, France.

[$ISIS, 2010$] Deschaud, J.-E. et Goulette, F., 2010, Détection de plans dans les nuages de points 3D générés à partir d'une plateforme mobile avec scanner laser. Journées GDR ISIS - Thème D - Action 3D, « De l'acquisition à la compression des objets 3D : un état des lieux en 2010 » , Ile de Porquerolles, France.

Bibliographie

[Abuhadrous, 2005] Abuhadrous, I., 2005. Système embarqué temps réel de localisation et de modélisation 3D par fusion multi-capteur. PhD thesis, MINES ParisTech.

[Alexa, 2002] Alexa, M., 2002. Wiener filtering of meshes. Dans les actes de la conférence : Shape Modeling International.

[Alexa et al., 2003] Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D. et Silva, C., 2003. Computing and rendering point set surfaces. IEEE Transactions on Visualization and Computer Graphics 9(1), pp. 3–15.

[Allen et al., 2001] Allen, P., Stamos, I., Gueorguiev, A., Gold, E. et Blaer, P., 2001. Avenue : Automated site modeling in urban environments. Dans les actes de la conférence : 3-D Digital Imaging and Modeling (3DIM).

[Alshawa et al., 2009] Alshawa, M., Boulaassal, H., Landes, T. et Grussenmeyer, P., 2009. Acquisition and automatic extraction of facade elements on large sites from a low cost laser mobile mapping system. Dans les actes de la conférence : 3D Virtual Reconstruction and Visualization of Complex Architectures (3D-ARCH).

[Amenta et al., 2002] Amenta, N., Choi, S., Dey, T. et Leekha, N., 2002. A simple algorithm for homeomorphic surface reconstruction. International Journal of Computational Geometry and its Applications 12(1–2), pp. 213–222.

[Amenta et al., 2001] Amenta, N., Choi, S. et Kolluri, R., 2001. The power crust, unions of balls, and the medial axis transform. International Journal of Computational Geometry and its Applications 19(2–3), pp. 127–153.

[Amenta et Bern, 1999] Amenta, N. et Bern, M., 1999. Surface reconstruction by voronoi filtering. Dans les actes de la conférence : Discrete and Computational Geometry.

[Asai et al., 2004] Asai, T., Kanbara, M. et Yokoya, N., 2004. 3d modeling of wide area outdoor environments by integrating omnidirectional range and color images. Dans les actes de la conférence : International Symposium on Mixed and Augmented Reality.

[Asai et al., 2005] Asai, T., Kanbara, M. et Yokoya, N., 2005. 3d modeling of outdoor scenes by integrating stop-and-go and continuous scanning of rangefinder. Dans les actes de la conférence : 3D Virtual Reconstruction and Visualization of Complex Architectures (3D-ARCH).

[Barber et al., 2008] Barber, D., Mills, J. et Smith-Voysey, S., 2008. Geometric validation of a ground-based mobile laser scanning system. Journal of Photogrammetry and Remote Sensing (JPRS) 63(1), pp. 128–141.

[Bauer et al., 2003] Bauer, J., Karner, K., Schindler, K., Klaus, A. et Zach, C., 2003. Segmentation of building models from dense 3d point-clouds. Dans les actes de la conférence : Austrian Association for Pattern Recognition.

[Bernardini et al., 1999] Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C. et Taubin, G., 1999. The ball-pivoting algorithm for surface reconstruction. IEEE Transactions on Visualization and Computer Graphics 5(4), pp. 349–359.

[Biber et al., 2005] Biber, P., Fleck, S., Wand, M., Staneker, D. et Straßer, W., 2005. First experience with a mobile plateform for flexible 3d model acquisition in indoor and outdoor environments - the wägele. Dans les actes de la conférence : 3D Virtual Reconstruction and Visualization of Complex Architectures (3D-ARCH).

[Boissonnat, 1984] Boissonnat, J., 1984. Geometric structures for three-dimensional shape representation. ACM Transactions on Graphics 3(4), pp. 266–286.

[Bolitho et al., 2007] Bolitho, M., Kazhdan, M., Burns, R. et Hoppe, H., 2007. Multilevel streaming for out-ofcore surface reconstruction. Dans les actes de la conférence : ACM Symposium on Computational Geometry.

[Boulaassal, 2010] Boulaassal, H., 2010. Segmentation et modélisation géométrique de façades de bâtiments à partir de relevés laser terrestres. PhD thesis, INSA de Strasbourg.

[Boulaassal et al., 2007] Boulaassal, H., Landes, T., Grussenmeyer, P. et Tarsha-Kurdi, F., 2007. Automatic segmentation of building facades using terrestrial laser data. Dans les actes de la conférence : Laser Scanning.

[Brun, 2007] Brun, X., 2007. Modélisation 3D texturée en temps réel d'environnements urbains et routiers, et application au calcul de distance de visibilité routière. PhD thesis, MINES ParisTech.

[Buades et Morel, 2005] Buades, A. et Morel, J.-M., 2005. A non-local algorithm for image denoising. Dans les actes de la conférence : Computer Vision and Pattern Recognition (CVPR).

[Buchart et al., 2008] Buchart, C., Borro, D. et Amundarain, A., 2008. Gpu local triangulation : an interpolating surface reconstruction algorithm. Computer Graphics Forum 27(3), pp. 807–814.

[Budroni et Bohm, 2009] Budroni, A. et Bohm, J., 2009. Automatic reconstruction of interiors from laser data. Dans les actes de la conférence : 3D Virtual Reconstruction and Visualization of Complex Architectures (3D-ARCH).

[Budroni et Bohm, 2010] Budroni, A. et Bohm, J., 2010. Automatic 3d modeling of indoor manhattan-world scenes from laser data. Dans les actes de la conférence : Close Range Image Measurement Techniques.

[Carlberg et al., 2008] Carlberg, M., Andrews, J., Gao, P. et Zakhor, A., 2008. Fast surface reconstruction and segmentation with ground-based and airborne lidar range data. Dans les actes de la conférence : 3D Processing, Visualization and Transmission (3DPVT).

[Carr et al., 2003] Carr, J., Beatson, R., McCallum, B., Fright, W., McLennan, T. et Mitchell, T., 2003. Smooth surface reconstruction from noisy range data. Dans les actes de la conférence : ACM GRAPHITE.

[Cazals et Pouget, 2003] Cazals, F. et Pouget, M., 2003. Estimating differential quantities using polynomial fitting of osculating jets. Dans les actes de la conférence : Eurographics/ACM SIGGRAPH Symposium on Geometry Processing.

[Cazals et al., 2005] Cazals, F., Giesen, J., Pauly, M. et Zomorodian, A., 2005. Conformal alpha shapes. Dans les actes de la conférence : Eurographics Symposium on Point-Based Graphics.

[Chang et al., 2009] Chang, M., Leymarie, F. et Kimia, B., 2009. Surface reconstruction from point clouds by transforming the medial scaffold. Computer Vision and Image Understanding 113(11), pp. 1130–1146.

[Chaperon, 2002] Chaperon, T., 2002. Segmentation de nuage de points 3D pour la modélisation automatique d'environnements industriels numérisés. PhD thesis, MINES ParisTech.

[Charbonnier et al., 2007] Charbonnier, P., Brun, X. et Goulette, F., 2007. Mesure de la visibilité géométrique à partir d'un modèle 3d de la route. Rapport technique, Livrable 1.2B, Consortium VIZIR, projet PREDIT SARI.

[Chen et Stamos, 2007] Chen, C. C. et Stamos, I., 2007. Range image segmentation for modeling and object detection in urban scenes. Dans les actes de la conférence : 3-D Digital Imaging and Modeling (3DIM).

[Choudhury et Tumblin, 2003] Choudhury, P. et Tumblin, J., 2003. The trilateral filter for high contrast images and meshes. Dans les actes de la conférence : Eurographics : Symposium on Rendering.

[Cohen-Steiner et al., 2004] Cohen-Steiner, D., Alliez, P. et Desbrun, M., 2004. Variational shape approximation. Dans les actes de la conférence : Special Interest Group on GRAPHics and Interactive Techniques (SIGGRAPH).

[Crossno et Angel, 1999] Crossno, P. et Angel, E., 1999. Spiraling edge : Fast surface reconstruction from partially organized sample points. Dans les actes de la conférence : Visualization.

[Curless et Levoy, 1996] Curless, B. et Levoy, M., 1996. A volumetric method for building complex models from range images. Dans les actes de la conférence : Special Interest Group on GRAPHics and Interactive Techniques (SIGGRAPH).

[Deveau, 2006] Deveau, M., 2006. Utilisation conjointe de données image et laser pour la segmentation et la modélisation 3D. PhD thesis, Université René Descartes - Paris 5.

[Dey et al., 2001] Dey, T., Giebsen, J. et Hudson, J., 2001. Delaunay based shape reconstruction from large data. Dans les actes de la conférence : IEEE Symposium on Parallel and Large-Data Visualization and Graphics.

[Dey et al., 2005a] Dey, T. K., Goswami, S. et Sun, J., 2005a. Extremal surface based projections converge and reconstruct with isotopy. Dans les actes de la conférence : Technical Report OSU-CISRC-05-TR25.

[Dey et al., 2005b] Dey, T. K., Li, G. et J.Sun, 2005b. Normal estimation for point clouds : A comparison study for a voronoi based method. Dans les actes de la conférence : Eurographics : Symposium on Point-Based Graphics.

[Diebel et al., 2006] Diebel, J. R., Thrun, S. et Brunig, M., 2006. A bayesian method for probable surface reconstruction and decimation. ACM Transactions on Graphics 25(1), pp. 39–59.

[Dupont et al., 2005] Dupont, R., Keriven, R. et Fuchs, P., 2005. An improved calibration technique for coupled single-row telemeter and ccd camera. Dans les actes de la conférence : 3-D Digital Imaging and Modeling (3DIM).

[Edelsbrunner et Mucke, 1994] Edelsbrunner, H. et Mucke, E., 1994. Three-dimensional alpha shape. ACM transactions on Graphics 13(1), pp. 43–72.

[Fischler et Bolles, 1981] Fischler, M. A. et Bolles, R. C., 1981. Random sample consensus : A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24(6), pp. 381–395.

[Fleishman et al., 2005] Fleishman, S., Cohen-Or, D. et Silva, C. T., 2005. Robust moving least-squares fitting with sharp features. Dans les actes de la conférence : Special Interest Group on GRAPHics and Interactive Techniques (SIGGRAPH).

[Fleishman et al., 2003] Fleishman, S., Drori, I. et Cohen-Or, D., 2003. Bilateral mesh denoising. ACM Transactions On Graphics 22(3), pp. 950–953.

[Fonger, 2009] Fonger, A. M., 2009. Applications of the Ball-Pivoting Algorithm to data from an Outdoor Environment. Rapport interne, caor, mines paristech edn.

[Früh et Zakhor, 2004] Früh, C. et Zakhor, A., 2004. An automated method for large-scale, ground-based city model acquisition. International Journal of Computer Vision (IJCV) 60(1), pp. 5–24.

[Früh et al., 2005] Früh, C., Jain, S. et Zakhor, A., 2005. Data processing algorithms for generating textured 3d building facade meshes from laser scans and camera images. International Journal of Computer Vision (IJCV) 61(2), pp. 159–184.

[Garland et Heckbert, 1997] Garland, M. et Heckbert, P. S., 1997. Surface simplification using quadric error metrics. Dans les actes de la conférence : Special Interest Group on GRAPHics and Interactive Techniques (SIGGRAPH).

[Gopi et Krishnan, 2002] Gopi, M. et Krishnan, S., 2002. A fast and efficient projection based approach for surface reconstruction. Dans les actes de la conférence : Graphics, Patterns and Images (SIBGRAPI).

[Gopi et al., 2000] Gopi, M., Krishnan, S. et Silva, C., 2000. Surface reconstruction based on lower dimensional localized delaunay triangulation. Computer Graphics Forum.

[Gorte, 2007] Gorte, B., 2007. Planar feature extraction in terrestrial laser scans using gradient based range image segmentation. Dans les actes de la conférence : ISPRS Workshop on Laser Scanning.

[Gorte et Pfeifer, 2004] Gorte, B. et Pfeifer, N., 2004. Structuring laser-scanned trees using 3d mathematicel morphology. The International Archives of the Photogrammetry Morphology 35(B5), pp. 929–933.

[Gotardo et al., 2003] Gotardo, P. F. U., Bellon, O. R. P. et Silva, L., 2003. Range image segmentation by surface extraction using an improved robust estimator. Dans les actes de la conférence : Computer Vision and Pattern Recognition (CVPR).

[Goulette et Laurgeau, 2002] Goulette, F. et Laurgeau, C., 2002. Chapitre 1 de l'ouvrage « Images de profondeur ». Hermes sciences publications edn.

[Goulette et al., 2006] Goulette, F., Nashashibi, F., Abuhadrous, I., Ammoun, S. et Laurgeau, C., 2006. An integrated on-board laser range sensing system for on-the-way city

and road modelling. International Archives of the Photogrammetry, Remote Sensing and Spacial Information Sciences 185, pp. 78–83.

[Gumhold et al., 2001] Gumhold, S., Wang, X. et Macleod, R., 2001. Feature extraction from point clouds. Dans les actes de la conférence : International Meshing Roundtable.

[Hamzaoui et Steux, 2010] Hamzaoui, O. E. et Steux, B., 2010. A fast scan matching for grid-based laser slam using streaming simd extensions. Dans les actes de la conférence : International Conference on Control, Automation, Robotics and Vision (ICARCV).

[Hartley et Zisserman, 2003] Hartley, R. et Zisserman, A., 2003. Multiple View Geometry in Computer Vision. Cambridge university press edn.

[Hoover et al., 1996] Hoover, A., Jean-Baptiste, G. et al., 1996. An experimental comparison of range image segmentation algorithms. Dans les actes de la conférence : IEEE Transactions on Pattern Analysis and Machine Intelligence.

[Hoppe et al., 1992] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. et Stuetzle, W., 1992. Surface reconstruction from unorganized points. Dans les actes de la conférence : Special Interest Group on GRAPHics and Interactive Techniques (SIGGRAPH).

[Hoppe et al., 1993] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. et Stuetzle, W., 1993. Mesh optimization. Dans les actes de la conférence : Special Interest Group on GRAPHics and Interactive Techniques (SIGGRAPH).

[Huang et Menq, 2002] Huang, J. et Menq, C., 2002. Combinatorial manifold mesh reconstruction and optimization from unorganized points with arbitrary topology. Computer Aided Design 34(2), pp. 149–165.

[Huhle et al., 2008] Huhle, B., Schairer, T., Jenke, P. et Strasser, W., 2008. Robust non-local denoising of colored depth data. Dans les actes de la conférence : Computer Vision and Pattern Recognition (CVPR) Workshops.

[Jenke et al., 2006] Jenke, P., Wand, M., Bokeloh, M., Schilling, A. et Strasser, W., 2006. Bayesian point cloud reconstruction. Computer Graphics Forum 25(3), pp. 379–388.

[Kannala et Brandt, 2006] Kannala, J. et Brandt, S. S., 2006. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(8), pp. 1335–1340.

[Kazhdan et al., 2006] Kazhdan, M., Bolitho, M. et Hoppe, H., 2006. Poisson surface reconstruction. Dans les actes de la conférence : Symposium on Geometry Processing.

[Kojekine et al., 2004] Kojekine, N., Savchenko, V. et Hagiwara, I., 2004. Surface reconstruction based on compactly supported radial basis functions. Dans les actes de la conférence : Geometric modeling : techniques, applications, systems and tools.

[Kolluri, 2005] Kolluri, R., 2005. Provably good moving least squares. Dans les actes de la conférence : Special Interest Group on GRAPHics and Interactive Techniques (SIGGRAPH).

[Lafarge et al., 2009] Lafarge, F., Keriven, R. et Brédif, M., 2009. Combining meshes and geometric primitives for accurate and semantic modeling. Dans les actes de la conférence : British Machine Vision Conference (BMVC).

[Levin, 1998] Levin, D., 1998. The approximation power of moving least-squares. Mathematics of Computation 67, pp. 807–815.

[Levin, 2003] Levin, D., 2003. Mesh-independant surface interpolation. Dans les actes de la conférence : Geometric Modeling for Scientific Visualization.

[Lin et al., 2004] Lin, H., Tai, C. et Wang, G., 2004. A mesh reconstruction algorithm driven by an intrinsic property of point cloud. Computer Aided Design 36(1), pp. 1–9.

[Liu et Xiong, 2008] Liu, Y. et Xiong, Y., 2008. Automatic segmentation of unorganized noisy point clouds based on the gaussian map. Computer Aided Design 40(5), pp. 576–594.

[Lorensen et Cline, 1987] Lorensen, W. et Cline, H., 1987. Marching cubes : A high resolution 3d surface construction algorithm. Special Interest Group on GRAPHics and Interactive Techniques (SIGGRAPH) 21(4), pp. 163–169.

[Medeiros et al., 2004] Medeiros, E., Velho, L. et Lopes, H., 2004. Restricted bpa : Applying ball-pivoting on the plane. Dans les actes de la conférence : Graphics, Patterns and Images (SIBGRAPI).

[Mederos et al., 2005] Mederos, B., Amenta, N., Velho, L. et de Figueiredo, L., 2005. Surface reconstruction from noisy point clouds. Dans les actes de la conférence : Symposium on Geometry Processing.

[Mederos et al., 2003] Mederos, B., Velho, L. et de Figueiredo, L. H., 2003. Robust smoothing of noisy point clouds. Dans les actes de la conférence : Geometric Design and Computing.

[Meek et Walton, 2000] Meek, D. S. et Walton, D. J., 2000. On surface normal and gaussian curvature approximations given data sampled from a smooth surface. Dans les actes de la conférence : Computer Aided Geometric Design.

[Mitra et al., 2004] Mitra, N. J., Nguyen, A. et Guibas, L., 2004. Estimating surface normals in noisy point cloud data. Special issue of International Journal of Computational Geometry and Applications 14(4–5), pp. 261–276.

[Munoz et al., 2008] Munoz, D., Vandapel, N. et Hébert, M., 2008. Directional associative markov network for 3-d point cloud classification. Dans les actes de la conférence : 3D Data Processing, Visualization and Transmission (3DPVT).

[OuYang et Feng, 2005] OuYang, D. et Feng, H.-Y., 2005. On the normal vector estimation for point cloud data from smooth surfaces. Computer-Aided Design 37(10), pp. 1071–1079.

[Oztireli et al., 2009] Oztireli, A. C., Guennebaud, G. et Gross, M., 2009. Feature preserving point set surfaces based on non-linear kernel regression. Computer Graphics Forum.

[Park et al., 2005] Park, S., Lee, S. et Kim, J., 2005. A surface reconstruction algorithm using weighted alpha shapes. Lecture Notes in Computer Science 3616, pp. 1141–1150.

[Pauly et Gross, 2001] Pauly, M. et Gross, M., 2001. Spectral processing of point-sampled geometry. Dans les actes de la conférence : Special Interest Group on GRAPHics and Interactive Techniques (SIGGRAPH).

[Pauly et al., 2003] Pauly, M., Keiser, R., Kobbelt, L. P. et Gross, M., 2003. Shape modeling with point-sampled geometry. Dans les actes de la conférence : Special Interest Group on GRAPHics and Interactive Techniques (SIGGRAPH).

[Pauly et al., 2004] Pauly, M., Mitra, N. et Guibas, L., 2004. Uncertainty and variability in point cloud surface data. Dans les actes de la conférence : Symposium on Point-Based Graphics.

[Pénard et al., 2005] Pénard, L., Paparoditis, N. et Pierrot-deseilligny, M., 2005. 3d building facade reconstruction under mesh form from multiple wide angle views. Dans les actes de la conférence : 3D Virtual Reconstruction and Visualization of Complex Architectures (3D-ARCH).

[Pollefeys et al., 2008] Pollefeys, M., Nister, D., Frahm, J.-M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S.-J., Merrell, P., Salmi, C., Sinha, S., Talton, B., Wang, L., Yang, Q., Stewenius, H., Yang, R., Welch, G. et Towles, H., 2008. Detailed real-time urban 3d reconstruction from video. International Journal of Computer Vision (IJCV) 78(2–3), pp. 143–167.

[Poppinga et al., 2008] Poppinga, J., Vaskevicius, N., Birk, A. et Pathak, K., 2008. Fast plane detection and polygonalization in noisy 3d range images. Dans les actes de la conférence : Intelligent Robots and Systems (IROS).

[Pu et Vosselman, 2006] Pu, S. et Vosselman, G., 2006. Automatic extraction of building features from terrestrial laser scanning. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 36(part 5), pp. 5.

[Pu et Vosselman, 2009] Pu, S. et Vosselman, G., 2009. Knowledge based reconstruction of building models from terrestrial laser scanning data. Journal of Photogrammetry and Remote Sensing (JPRS) 64(6), pp. 575–584.

[Rabbani et al., 2006] Rabbani, T., den Heuvel, F. A. V. et Vosselman, G., 2006. Segmentation of point clouds using smoothness constraint. Dans les actes de la conférence : Image Engineering and Vision Metrology.

[Ridene, 2010] Ridene, T., 2010. Co-Recalage de données hétérogènes 3D géo-référencées : contributions à la correction de relevés laser mobiles. PhD thesis, MINES ParisTech.

[Rutzinger et al., 2010] Rutzinger, M., Pratihast, A. K., Elberink, S. O. et Vosselman, G., 2010. Detection and modelling of 3d trees from mobile laser scanning data. Dans les actes de la conférence : International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences.

[Samozino et al., 2006] Samozino, M., Alexa, M., Alliez, P. et Yvinec, M., 2006. Reconstruction with voronoi centered radial basis functions. Dans les actes de la conférence : Symposium on Geometry Processing.

[Schall et al., 2008] Schall, O., Belyaev, A. et H.-P.Seidel, 2008. Adaptive feature-preserving non-local denoising of static and time-varying range data. Computer-Aided Design 40(6), pp. 701–707.

[Schnabel et al., 2007] Schnabel, R., Wahl, R. et Klein, R., 2007. Efficient ransac for point-cloud shape detection. Computer Graphics Forum 26(2), pp. 214–226.

[Schroeder et al., 1992] Schroeder, W., Zarge, J. et Lorensen, W., 1992. Decimation of triangle meshes. Dans les actes de la conférence : Special Interest Group on GRAPHics and Interactive Techniques (SIGGRAPH).

[Sharkarji, 1998] Sharkarji, C. M., 1998. Least-squares fitting algorithms of the nist algorithm testing system. Dans les actes de la conférence : Journal of Research of the National Institute of Standards and Technology.

[Sheung et Wang, 2009] Sheung, H. et Wang, C., 2009. Robust mesh reconstruction from unoriented noisy points. Dans les actes de la conférence : ACM Symposium on Solid and Physical Modeling.

[Song et Feng, 2008] Song, H. et Feng, H.-Y., 2008. A global clustering approach to point cloud simplification with a specified data reduction ratio. Computer Aided Design 40(3), pp. 281–292.

[Soudarissanane et al., 2009] Soudarissanane, S., Lindenbergh, S., Menenti, R. et Teunissen, M., 2009. Incidence angle influence on the quality of terrestrial laser scanning points. Dans les actes de la conférence : ISPRS Workshop Laserscanning.

[Stamos et Allen, 2002] Stamos, I. et Allen, P. K., 2002. Geometry and texture recovery of scenes of large scale. Computer Vision and Image Understanding 88(2), pp. 94–118.

[Stamos et al., 2006] Stamos, I., Yu, G., Wolberg, G. et Zokai, S., 2006. 3d modeling using planar segments and mesh elements. Dans les actes de la conférence : 3-D Data Processing, Visualization and Transmission (3DPVT).

[Tarsha-Kurdi et al., 2007] Tarsha-Kurdi, F., Landes, T. et Grussenmeyer, P., 2007. Hough transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data. Dans les actes de la conférence : Laser Scanning.

[Taubin, 1995] Taubin, G., 1995. A signal processing approach to fair surface design. Dans les actes de la conférence : Special Interest Group on GRAPHics and Interactive Techniques (SIGGRAPH).

[Taubin et al., 1994] Taubin, G., Cukierman, F., Sullivan, S., Ponce, J. et Kriegman, D., 1994. Parameterized families of polynomials for bounded algebraic curve and surface fitting. IEEE Transactions on Pattern Analysis and Machine Intelligence 16(3), pp. 287–303.

[Tomasi et Manduchi, 1998] Tomasi, C. et Manduchi, R., 1998. Bilateral filtering for gray and color images. Dans les actes de la conférence : International Conference on Computer Vision (ICCV).

[Turk et O'Brien, 2002] Turk, G. et O'Brien, J., 2002. Modelling with implicit surfaces that interpolate. ACM Transactions on Graphics 21(4), pp. 855–873.

[Vosselman et al., 2004] Vosselman, G., Gorte, B. G. H., Sithole, G. et Rabbani, T., 2004. Recognising structure in laser scanner point clouds. Dans les actes de la conférence : International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences.

[Wang et Suter, 2004] Wang, H. et Suter, D., 2004. Mdpe : A very robust estimator for model fitting and range image segmentation. International Journal of Computer Vision (IJCV) 59(2), pp. 139–166.

[Wang et al., 2008] Wang, R.-F., Chen, W.-Z., Zhang, S.-Y., Zhang, Y. et Ye, X.-Z., 2008. Similarity-based denoising of point-sampled surfaces. Journal of Zhejiang University - Science A 9(6), pp. 807–815.

[Yagou et al., 2002] Yagou, H., Ohtake, Y. et Belyaev, A., 2002. Mesh smoothing via mean and median fitering applied to face normals. Dans les actes de la conférence : Geometric Modeling and Processing.

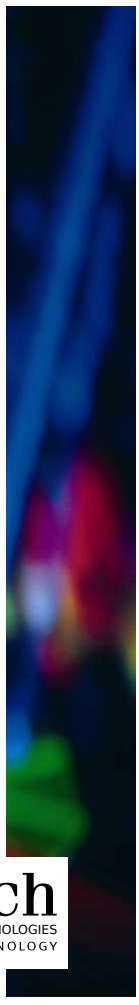
[Yoshizawa et al., 2006] Yoshizawa, S., Belyaev, A. et Seidel, H.-P., 2006. Smoothing by example : Mesh denoising by averaging with similarity-based weights. Dans les actes de la conférence : Shape Modeling and Applications.

[Yu et al., 2007] Yu, S.-J., Sukumar, S. R., Koschan, A. F., Page, D. L. et Abidi, M. A., 2007. 3d reconstruction of road surfaces using an integrated multi-sensory approach. Optics and lasers in engineering 45(7), pp. 808–818.

[Zhao et Shibasaki, 2003] Zhao, H. et Shibasaki, R., 2003. A vehicle-borne urban 3-d acquisition system using single-row laser range scanners. IEEE Transactions on Systems, Man and Cybernetics 33(4), pp. 658–666.

Traitements de nuages de points denses et modélisation 3D d'environnements par système mobile LiDAR/Caméra

Résumé : Le laboratoire de robotique CAOR de MINES ParisTech a mis au point une technique de numérisation 3D d'environnements extérieurs, utilisant un système mobile appelé LARA3D. Il s'agit d'un véhicule équipé de capteurs de localisation (GPS, Centrale Inertielle) et de capteurs d'imagerie (LiDARs et caméras). Ce dispositif permet de construire des nuages de points 3D denses et des images géo-référencées de l'environnement traversé. Dans le cadre de cette thèse, nous avons développé une chaîne de traitement de ces nuages de points et de ces images capable de générer des modèles 3D photo-réalistes. Cette chaîne de modélisation est composée d'une succession d'étapes avec dans l'ordre : le calcul des normales, le débruitage des données, la segmentation en zones planes, la triangulation, la simplification et la texturation. Nous avons testé notre méthode de modélisation sur des données simulées et des données réelles acquises dans le cadre des projets TerraNumerica (en environnement urbain) et DIVAS (en environnement routier).
Mots clés : modélisation, nuage de points, lidar, caméra, débruitage, segmentation, maillage



Dense point clouds processing and 3D environment modeling from LiDAR/Camera mobile system

Abstract: The Robotics Laboratory CAOR from MINES ParisTech has developed a technique for 3D scanning of outdoor environments, using a mobile system called LARA3D. This is a vehicle with location sensors (GPS, inertial unit) and imaging sensors (LiDARs and cameras). This device can build 3D point clouds and geo-referenced images of the environment. As part of this thesis, we have developed a processing chain of these point clouds and images for generating photo-realistic 3D models. This modeling chain has many processing steps with, in order : normal computation, denoising, planar area segmentation, triangulation, simplification and texturing. We have applied our modeling method on synthetic data and on real data acquired through the TerraNumerica (for urban environment) and the DIVAS (for road environment) projects.
Keywords: modeling, point cloud, lidar, camera, denoising, segmentation, meshing

MINES ParisTech

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY