Fusion de maillages triangulés issus de différents points de vue

Carteron Rémi

06/09/2005

Contents

List of Figures

List of Tables

1 Introduction

Cet article présente les travaux de recherche et d'implémentation de la **Fusion de maillages triangulés issus de différents points de vue** dans le cadre d'un stage au sein du laboratoire MATIS de l'IGN (Institut Géographique National).
L'IGN envisage de faire des campagnes de prises de vue d'images haute-résolution géoréférencés (c'est-à-dire dont les paramètres de prise de vue sont connus) de façades de bâtiments (figure 1) dans le but de disposer de modèles 3D de paysages urbains issus de ces images. De ces images

Figure 1: Exemple d'images de façades (210 209 208)

seront extraits des nuages de points qui seront triangulés. L'utilisation de modèles triangulés plutôt que celle de nuages de points permet de mieux représenter des topologies et des géométries, puisque que l'on disposera de surfaces et non plus de données discrètes (figure 2).

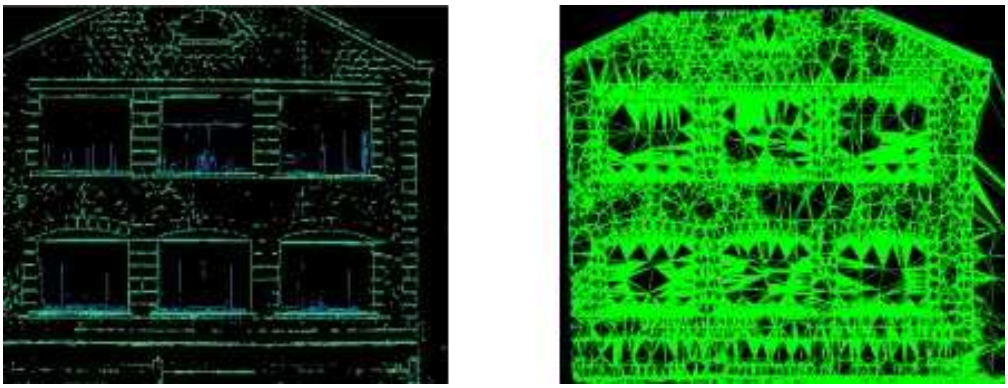

Figure 2: Exemple d'un nuage de points et de sa triangulation (façade 209)

Le but de ces recherches vise donc à implémenter un algorithme permettant de reconstruire des bâtiments en 3D (et plus généralement des objets 3D) à partir d'images 2.5D (de profondeur) ou d'acquisitions laser. L'utilisation de plusieurs vues permet évidemment de reconstruire différentes faces d'un objet mais aussi de mieux modéliser certaines de ses parties cachées. Ce qui est préférable dans un but de reconstruction s'approchant le plus possible de la réalité.
On obtient finalement en sortie, une triangulation par point de vue ou par acquisition laser, on a

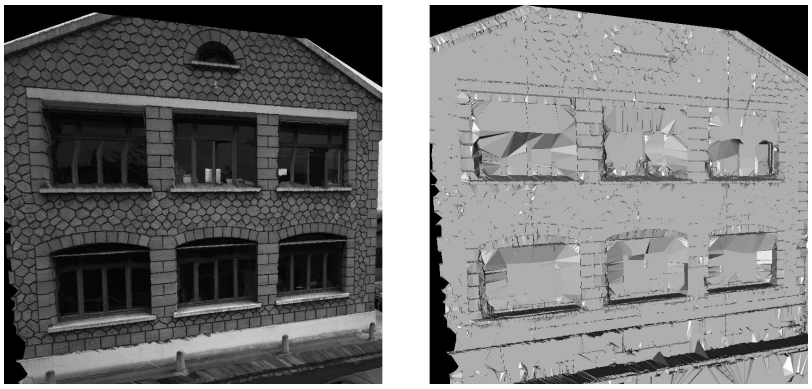

Figure 3: Exemple d'une façade triangulée avec texture (à droite) et sans texture (à gauche)

donc besoin d'une méthode automatisée qui permette d'assembler ces triangulation en une unique surface 3D. Dans le cadre de cet article on présentera les travaux ayant trait à la simplification et à la fusion de telles triangulations. Pour cela on dispose en entrée de modèles déjà triangulés (un jeu de données issues d'images de façades et un modèle simulé d'un cube troué permettant d'avoir des parties cachées -figure 4-) qui seront les données utilisées par nos algorithmes.

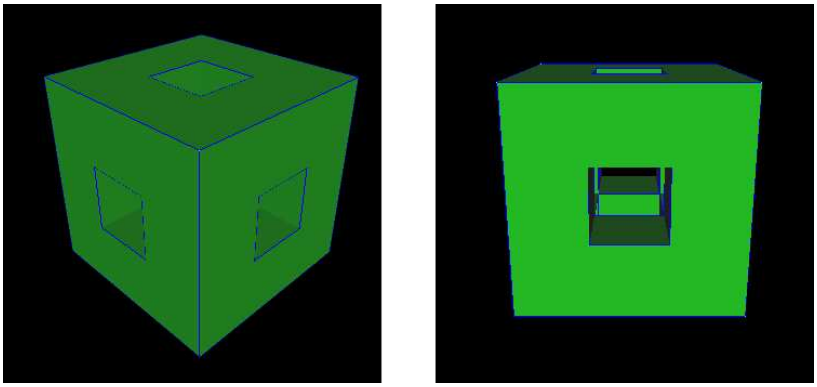

Figure 4: Simulation d'un cube troué

1.1 État de l'art

Les premières méthodes de fusion de triangulations remontent au milieu des années 80 [Boi84]. À partir de 92, de nouvelles méthodes apparaissent et se diversifient en plusieurs catégories. Voici un bref aperçu des méthodes les plus utilisées (pour plus de détails, voir [Ben98] chapitre 8, [Hil97]).

1.1.1 Intégration de maillages

Les techniques d'intégration de maillages consistent à fusionner plusieurs triangulations qui se chevauchent en une seule triangulation finale. Dans toutes ces méthodes les triangulations sont issues d'images 2.5D (c'est-à-dire des images de distances).

Geometry structures for three-dimensional shape representation [Boi84] et Merging range images of arbitrarily shaped objects [RST94] Les méthodes de Boissonnat et de Rutishauser déterminent les chevauchements de mailles à l'aide de graphes.
Rutishauser a amélioré les travaux de Boissonnat en y ajoutant une approche probabiliste. Ce qui permet d'être plus robuste aux erreurs d'acquisition. Ces deux méthodes fusionnent deux triangulations en une seule en utilisant des contraintes 2D sur les triangles.

Zippered polygon meshes from range image [TL94] Turk et Levoy proposent une méthode dite de "zippering". La détection de recouvrement entre les triangles se fait en vérifiant que la distance entre un sommet et son point le plus proche issu d'une autre triangulation soit inférieur à un seuil. Les triangulations seront alors érodées jusqu'à se qu'elles ne se recouvrent plus. Enfin il ne restera plus qu'à "zipper" (c'est-à-dire raccorder - voir fugure 5) les bords des deux triangulations pour n'en former qu'une seule.

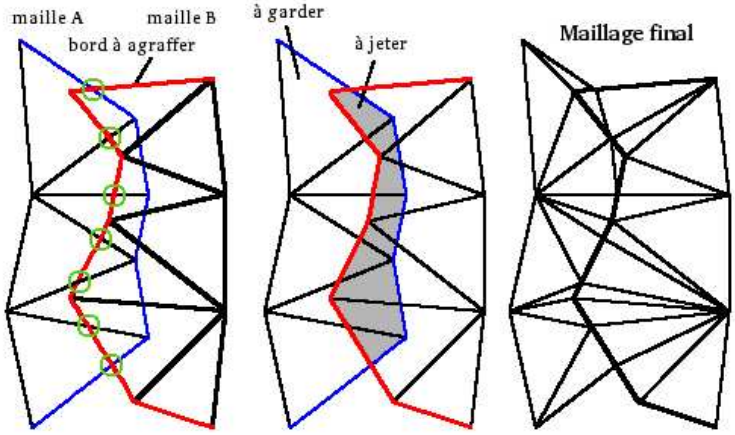

Figure 5: Principe du zippering [TL94]

A general surface approach to the integration of a set of range views [SL95] Dans sa méthode Soucy adopte une approche paramétrique pour determiner les recouvrements de nappes provenant d'images 2.5D. Ceci permet par la suite d'éliminer les parties redondantes et de fusionner les régions qui s'intersectent.

Mesh integration based on co-measurement [Pit96] Pito améliore les travaux de Soucy [SL95] et de Turk [TL94] en se servant d'un test de visibilité spécifique pour détecter les recouvrements. Cela permet de fusionner des triangles de surfaces d'orientations opposées. L'élimination

de redondance se fait de la même manière que celle de Turk [TL94] sauf que Pito introduit une mesure de fiabilité qui fait que seuls les triangles les plus fiables sont gardés. Enfin Pito raccorde les triangulations en construisant des triangles joignant les bords (approche "soudure" figure 6 : (a) les bordures sont trouvées et classées, (b) un pont est construit entre les deux bordures, (c) la "soudure" s'oppère).

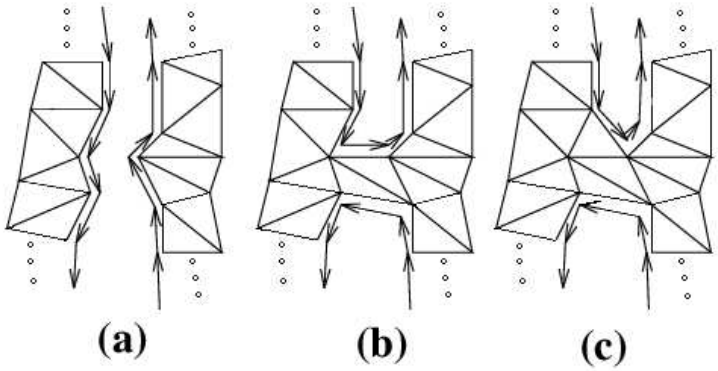

Figure 6: Principe de "Soudure" de Pito [Pit96]

1.1.2 Fusion volumique

Les méthodes de fusion volumiques diffèrent des méthodes d'intégration dans le fait que les opérations sont effectuées dans un espace 3D.
La fusion volumique construit une surface intermédiaire implicite qui représente au mieux les mesures des surfaces qui se chevauchent.
Ces méthodes consistent donc à trouver une fonction $f : R^3 \rightarrow R^3$ telle que les points à fusionner appartiennent tous à l'ensemble des zéros de f ou d'en être le plus proche possible.

Surface reconstruction from unorganized points [HDD+92] La méthode de Hoppe n'est pas réellement une méthode de fusion proprement dite mais elle est à la base de toutes les méthodes de fusion volumique. Elle permet de trianguler des nuages de points non structurés. Hoppe est le premier à introduire la notion de surface implicite de laquelle les méthodes qui vont suivre vont s'inspirer.
Pour estimer la topologie locale. Hoppe considère ensuite la fonction implicite f comme une distance signée par rapport à la surface recherchée. Enfin, pour reconstruire la surface, Hoppe utilise l'algorithme de "Marching cubes"[1] [LC87].

Reliable surface reconstruction from multiple range images [Hil95] [HI97] Hilton modifie la méthode de Hoppe en lui permettant de reconstruire une surface continue à partir de triangulations discontinues provenant de plusieurs images 2.5D. Les fonctions implicites pour chacun des maillages 2.5D (fonction distance signée par rapport à la surface recherchée - comme

dans la méthode de Hoppe) sont d'abord calculées, puis ces fonctions sont intégrées en utilisant des règles basées sur la géométrie de la surface. Enfin, l'algorithme de "Marching' cubes"[1] [LC87] est ici aussi utilisé pour retrianguler la surface.
Hilton [Hil97] étend sa première méthode [Hil95] en rendant la taille des voxels modulable à l'aide d'octree. Ce qui donne une meilleure précision pour les petits détails et permet aussi de réduire l'espace de stockage.

A volumetric method for building complex models from range images[CL96] La méthode de Curless et Levoy est proche de celle de Hilton [Hil95]. Elle se base aussi sur une représentation implicite des triangulations par une fonction distance signée (même si cette fonction diffère légèrement du fait du contexte d'acquisition de Curless). Mais Curless introduit un rééchantillonage des triangulations et une opération d'extrusion qui permet le remplissage de trous. Cette méthode utilise aussi un codage "Run-Length Encoding"[2] (RLE) pour accélérer les temps de calculs et réduire l'espace de stockage.

Les données de triangulations de l'IGN sont issues de surfaces de corrélation obtenues à partir d'images haute-résolution ou d'acquisitons laser. Il est parfois peu évident de les manipuler, de les stocker ou de les visualiser sur des machines standards (taille des données imposantes et temps de calculs trop long).
Il est nécessaire, avant la fusion proprement dite de s'interresser à la simplification du volume des données. La partie I de cet article sera donc consacrée à la simplification de maillage, ensuite la partie II concernera la fusion proprement dite.

[1] L'algorithme de Marching cubes permet de polygoniser une surface courbe. Son principe est de subdiviser l'espace en petits cubes et de s'aider des points d'intersections de la courbe avec ces derniers pour la polygoniser.

[2] Le Run-Length Encoding est un codage de compression basé sur le codage de répétitions de donnés (binaire, pixels, caractères...).

Part I
Simplification de maillage

Dans un premier temps on s'intéresse à réduire le volume des données grâce à une simplification de maillages. On cherche à savoir si on ne pourrait pas simplifier nos triangulations par diverses méthodes existantes.

1 État de l'art

Dans la littérature il existe plusieurs méthodes de simplification de maillages. Voici un bref résumé des principales méthodes.

Progressive Meshes [Hop96] Le choix des arêtes à réduire est determiné par une fonction d'énergie à minimiser. L'évaluation de cette fonction se base sur les attributs du maillage selon 2 catégories : les attributs discrets (indices de matériau et de texture) et les attributs continus (coordonnées, normales). Cette simplification s'adapte ainsi aux caractéristiques du maillage et non pas seulement à l'erreur géométrique.

Image-driven Simplification [LT00] Lindstrom et Turk ont basé leur méthode sur des comparaisons d'images. Les coûts des réductions d'arêtes sont analysés en comparant l'image du modèle simplifié avec l'image du modèle initial.

Fast and efficient Polygonal Simplification [LT98] Lindstrom et Turk choisissent la position du nouveau sommet tel que le volume original soit maintenu et qu'il minimise les changements de volume du tetraèdre formé par les triangles qui ont été déplacés. La surface près des bords est maintenue et les changements d'aire sont minimisés.

Surface simplification using Quadric Error Metrics [GH97] Garland et Heckbert [GH97] [GH98] proposent une méthode qui est très proche de la réduction d'arêtes. Elle apparie des paires de sommets qui seront classés selon un coût de contraction dans une pile. La paire de plus faible coût est dépilée puis contractée à la position optimale. Le dépilement se fait tant qu'on a pas atteint le nombre de triangles désiré.

Appearence-Preserving Simplification [COM98] La technique de simplification avec préservation de l'apparence génère une approximation qui garde l'apparence du modèle pour une résolution d'écran donnée. L'algorithme échantillonne en fait la position de la surface, la courbure et les attributs de couleurs. Le choix des arêtes à contracter est fait de façon à rester dans les limites de résolution. Au final on plaque la texture de couleur et de normales sur le maillage simplifié.

Pour un panorama plus complet sur les méthodes de simplification voir [Por04] [Lue01].

2 Algorithme personnel de simplification

2.1 Explication/contexte

Dans un premier temps on s'intéresse à savoir si on ne pourrait pas simplifier la triangulation sans altérer la géométrie.
Les façades de bâtiments ont généralement de grandes sections planes ; de nombreux points peuvent appartenir à un même plan.
On pourrait donc en supprimer un grand nombre, surtout sur des maillages très denses acquis avec des images hautes-résolutions comme celle de l'IGN.

2.2 Algorithme mis en oeuvre

2.2.1 Principe

Un point dont tous les triangles adjacents sont coplanaires n'apporte rien quand à la géométrie du maillage. Sans ce point la géométrie reste strictement identique. On peut par conséquent le supprimer.
De plus dans le contexte de nos images de façades, l'axe z est strictement horizontal et est perpendiculaire au plan de la façade. Ce qui simplifie grandement le problème, d'autant plus que cet axe est discrétisé de telle sorte que les points peuvent avoir le même z. Il suffit donc de vérifier la coordonnée z des points pour détecter la co-planarité.
Il existe aussi dans le maillage d'autres points que l'on pourrait supprimer. La recherche de coplanarité de leurs triangles aura une probabilité quasi nulle d'aboutir (les coordonnées en x et en y n'étant pas discrétisés). Ces points étant de plus supposés peu nombreux, cette recherche supplémentaire ne sera pas réalisée.
Il faut souligner que cette étape de simplification est spécifique à nos données de façades, elle n'a donc aucune pertinence pour des données laser.

2.2.2 Algorithme

```
On initialise une liste l des points que l'on gardera.
Pour (chaque point P du maillage)
   Pour (tous les voisins V de P)
      Si (au moins un voisin n'a pas le même z que P)
         On ajoute P à l.
      Sinon Si (P fait partie du bord de la maille)
         On ajoute P à l.
On retourne la liste obtenue.
Puis on retriangule les points gardés.
```

2.3 Résultats

Grâce à notre algorithme de simplification on peut diviser le volume de données par deux (voir tableau 1). Ce qui est appréciable pour la visualisation et la manipulation de ces données. De plus les façades (figure 1) sur lesquelles s'effectuent nos tests ont des surfaces rocailleuses

assez irrégulières, on peux donc espérer des résultats supérieurs sur des bâtiments à surfaces plus lisse.

Triangulations	Espace mémoire		Nombre de points		Nombre de triangles	
	Départ	Arrivée	Départ	Arrivée	Départ	Arrivée
Numéro 208	13.3 Mo	5.8 Mo	116 988	51 522	233 708	102 776
Numéro 209	21.2 Mo	8.6 Mo	182 303	76 206	364 111	151 917
Numéro 210	22.3 Mo	10.4 Mo	191 235	91974	382 102	182 980

Table 1: Résultats avec notre algorithme de simplification

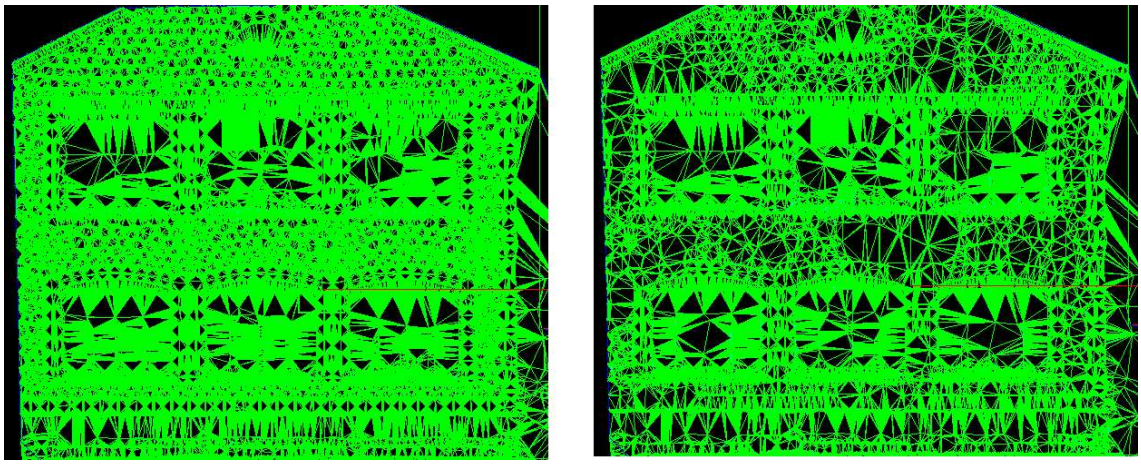

Figure 7: Triangulation de départ et celle simplifiée par notre algorithme

3 Simplification de maillage avec QSlim

Les simplifications de maillages sans perte de géométrie réduisent les données de façon "rigide". On ne peux pas déterminer à l'avance par exemple le niveau de simplification du résultat final. Comme le montrent les résultats de la section 2, le gain de volume sur nos données est appréciable mais ces données peuvent encore rester trop volumineuses lors de calculs ou dans le stockage de milliers de triangulations.
On s'intéresse donc à pouvoir simplifier encore plus les triangulations. La simplification sans perte ayant été faite, on se doit de trouver une simplification qui soit "modulable" (simplification qui permet de cibler le niveau de simplification désiré) et qui limite les déformations géométriques.

3.1 Explication du choix de qslim

La méthode de Lindstrom et Turk [LT00] basée sur les images, même si elle donne d'excellents résultats visuels n'a aucun "contrôle" sur la géométrie. Ce qui ne correspond pas aux attentes de l'IGN qui s'intéresse à restituer le plus fidèlement possible la géométrie des façades. Le rendu

visuel n'est pas prioritaire.
De même pour la méthode de simplification avec préservation de l'apparence [COM98].
La méthode de Hoppe [Hop96], n'a elle qu'un "contrôle" partiel sur la géométrie, ce qui limite aussi son utilisation pour l'IGN.

Les algorithmes de Garland [GH98] et de Lindstrom et Turk [LT98] sont les seuls à s'intéresser vraiment à la géométrie. Ces méthodes sont donc celles qui altèrent le moins la géométrie (parmi les méthodes citées - voir [RNFT01]-). Ce qui correspond bien au profil d'algorithme souhaité dans notre problématique de simplification.
Garland a publié l'implémentation de son algorithme (sous le nom de QSlim) sur le web et l'a developpé sous licence GNU ; QSlim est opensource. Ce qui fait qu'il est facilement récupérable et intégrable dans d'autres codes.
Le choix se portera donc sur QSLim ([Gar99]). QSlim est un programme régulièrement amélioré et optimisé. Si on devait implémenter l'algorithme de Lindstrom et Turk, l'integration serait plus longue et probablement moins performante.

3.2 Implémentation de QSlim

3.2.1 Contexte

Avec QSlim, il reste une interrogation sur le moyen de déterminer à l'avance le nombre de triangles final qui simplifiera le modèle au mieux sans "trop" déformer la géométrie.
On pourrait par exemple lancer QSlim en boucle (par exemple en divisant le nombre de triangles final par 2 à chaque itération) jusqu'à atteindre un niveau de rendu satisfaisant. Avec une solution de ce genre, on devra superviser les résultats visuellement. Ce qui dans notre contexte n'est pas du tout le but recherché, on doit absolument rester géométriquement proche du modèle initial. Ce qu'on ne peux pas juger simplement à l'oeil nu.
Pour trouver une meilleure solution on doit d'abord regarder en détail l'algorithme de Garland.

3.2.2 Algorithme de simplification de Garland

```
Initialisation des coûts de contraction.
Les paires de sommets sont placées dans une pile (les paires sont
triées avec les coûts les plus faibles en haut de la pile).
Tant qu'on n'a pas atteint le nombre de triangles cible
   On contracte la paire de moindre coût.
   Mise à jour des coûts.
```

3.3 Modification de l'algorithme de Garland

3.3.1 Principe

On constate que la boucle itère tant que le nombre de triangles n'est pas atteint. Les coûts de contraction sont ordonnés par ordre croissant. Donc plutôt que d'itérer sur le nombre de triangles, on pourrait itérer sur un seuil représentant l'erreur géométrique et arrêter lorsque cette erreur dépasse un seuil de déformation géométrique.

3.3.2 Algorithme

```
Initialisation des coûts de contraction.
Les paires de sommets sont placées dans une pile (les paires sont
triées avec les coûts les plus faibles en haut de la pile).
Tant que le coût de contraction est inférieur à un seuil
   On contracte la paire de moindre coût.
   Mise à jour des coûts.
```

3.3.3 Interprétation Géométrique

Pour pouvoir determiner un seuil, il faut pouvoir interpréter géométriquement ce que représente ce coût de contraction (qui est une erreur de quadriques -matrices 4x4-).
Cette erreur représente en fait la somme des carrés de la distance d'un sommet à ses plans.
Garland interprète aussi ce coût comme étant le volume carré du tétraèdre formé par le point optimal de la contraction et des triangles reliant l'un des deux sommets ([Gar99] chapitre 4).
Malheureusement, ces interprétations ne peuvent garantir la robustesse de l'algorithme à rester aussi proche de la réalité que l'on voudrait. En effet, on pourrait tout-à-fait avoir un tétraèdre dont le sommet optimal s'écarterait grandement des triangles associés tout en ayant un volume très réduit. Néanmoins un tel cas de figure est peu probable sur nos données de façades.

3.4 Résultats

On se rend compte que même avec un nombre de triangles cible assez faible par rapport au nombre de triangles de départ (5 000 contre 364 111), visuellement le modèle simplifié n'est pas si éloigné du modèle initial (modèle simplifié : figure 8 et modèle initial : figure 7). On peut repérer les fenêtres des surfaces planes. Evidemment, une division des données par plus de 60 ne peux pas donner un résultat satisfaisant pour des données devant rester le plus proche possible de la réalité.
En revanche la simplification avec une cible d'erreur de 1 cm^3 (figure 8) donne un résultat très satisfaisant vu le nombre de triangles obtenu (26 519 soit un peu plus de dix fois moins de triangles).

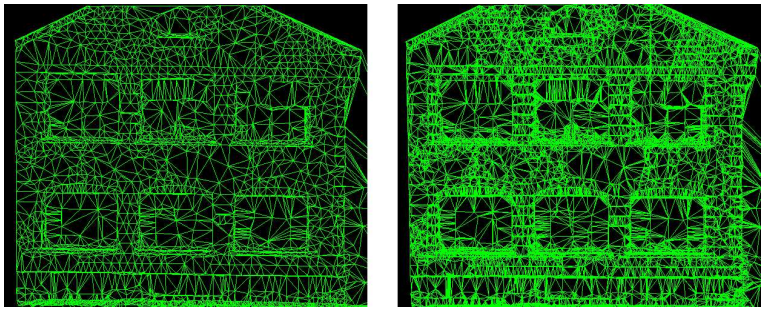

Figure 8: Simplifications avec QSlim (à gauche : cible de 5000 triangles et à droite : cible d'erreur évaluée à 1 cm^3)

Part II
Fusion

1 Filtrage

1.1 Explication du filtrage

Les triangulations possèdent, du fait des parties cachées liées aux prises de vue (figure 9), des triangles erronés sur les discontinuités des façades. Ces triangles doivent donc être éliminés pour éviter qu'ils ne créent des incohérences lors de la fusion.

1.2 Algorithme mis en oeuvre

1.2.1 Principe de base

On peut détecter un triangle erroné s'il existe un point 3D d'un autre point de vue qui se trouve derrière ce triangle (Le point P se trouvant derrière le triangle T, le triangle T sera supprimé -schma 9-). On utilise pour cela un **distance-buffer**[3] qui sera notre test de visibilité (on utilise un test de visibilité tout comme Pito [Pit96] dans sa méthode).

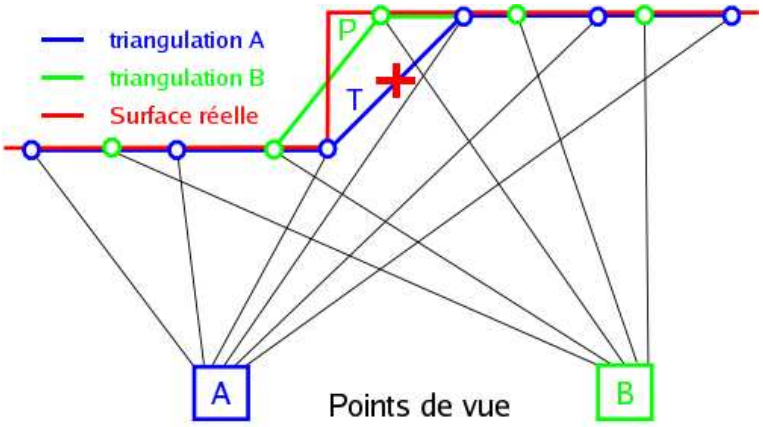

Figure 9: Schéma de filtrage

1.2.2 Algorithme de base

```
On calcule le distance-buffer de la surface reconstruite à partir du point de vue
A dans la géométrie du point de vue B.
On parcourt tous les points vue en B :
   Pour chaque point-image PI de B :
```

[3]Le Distance-Buffer est une variante du **Z-Buffer** qui utilise les distances entre les points 3D et l'origine plutôt que les coordonnées z. Le Z-Buffer est une technique permettant le calcul de l'élimination des parties cachées en ne gardant en mémoire que les points visibles d'un certain point de vue (grâce à leurs coordonnées z).

```
On calcule le point-terrain PT associé à PI.
On calcule la distance D entre PT et le point de vue B.
Si (D - distance-buffer) est supérieure à un certain seuil :
   Le triangle associé aux coordonnées images du ditance-buffer doit
   être éliminé.
```

1.2.3 Résultats

La plupart des triangles erronés ont été filtrés tant sur les données simulées que sur les données réelles (tableau 2). Ceux-ci sont bien ceux supposés erronés (aux abords des parties cachées - figure 10-), ce qui est rassurant. Cependant, on constate qu'il reste une quantité non négligeable de triangles devant être filtrés. Nous proposons quelques solutions pour éliminer les triangles restants.

1.2.4 Les problèmes

Triangles effilés. Les triangles à éliminer sont parfois trés éffilés. Tous les triangles ne peuvent pas tous être affichés dans le distance-buffer si ceux-ci sont beaucoup trop fins car ils sont masqués par d'autres triangles du voisinage.
Pour éliminer tous les triangles erronés il faut qu'ils puissent être affichés dans le distance-buffer au moins une fois. On peut par exemple recalculer un distance-buffer sur la triangulation déjà filtrée. Ce qui permet à des triangles qui n'ont pas pu être affichés dans le premier filtrage de l'être dans le second.
On a maintenant atteint les limites de la complémentarité des différents points de vue. Néanmoins, d'autres traitements permettraient d'aller plus loin.

Triangles erronés restants Pour qu'un triangle erroné soit filtré, il faut qu'un point d'un autre point de vue le permette. Il peut arriver qu'aucun point ne permette de filtrer un triangle erroné. Il faut que les points coïncident avec les triangles à simplifier, ce qui n'est pas forcément le cas et donc il peut rester des triangles qui doivent tout de même être supprimés.
Après un filtrage on se rend compte que des triangles isolés restent au milieu de zones filtrées. On peut considérer qu'un triangle se retrouvant seul dans une zone, peut être supprimé. Partant de ce constat, on se rend compte que les triangles qui ont 2 voisins déjà supprimés peuvent être supprimés à leur tour. Il suffit donc d'effectuer ce "postfiltrage" pour obtenir un résultats final plus convaincant.

1.3 Remplissage de trous

Une fois les filtrages effectués, il faut reboucher les triangles manquants avec une géométrie correcte (venant des autres points de vue). Cette étape est réalisée lors de la fusion. Pendant le filtrage, tous les points qui ont permis de filtrer un triangle sont stockés pour être raccordés à la triangulation plus tard.

1.4 Algorithme final de filtrage

```
Tant que des triangles sont supprimés :
   On calcule le distance-buffer de la surface reconstruite à partir
```

```
du point de vue A dans la géométrie du point de vue B.
Pour chaque point-image PI de B :
   On calcule le point-terrain PT associé à PI.
   On calcule la distance D entre PT et le centre du point de vue B.
   Si (D - distance-buffer) est supérieure à SEUIL.
   Le triangle associé aux coordonnées images du ditance-buffer
      doit être éliminé et on ajoute PT à la triangulation.
On parcourt tous les triangles :
Si 2 de ses voisins ont déjà été supprimés :
   Ce triangles est supprimé.
```

1.5 Résultats

On constate qu'un filtrage itéré augmente jusqu'à 10 % le nombre de triangles filtrés sur nos données de façades et de 50 % sur nos données simulées (tableau 2), ce qui n'est pas négligeable. De même, le "postfiltrage" augmente jusquà 70 % le nombre de triangles filtrés sur nos données de façades et de 15 % sur nos données simulées.
Ces deux techniques permettent donc d'améliorer sensiblement les résultats de filtrages (figure 10).
Cependant, on remarque qu'il reste encore des triangles qui pourraient être supprimés. En effet, la densité de points aux bords des discontinuité des triangulations des façades étant faible il se peut qu'aucun point ne permette de filtrer ces triangles. On peut constater que le filtrage marche parfaitement sur des données simulées (figure 11).

	Nombre de triangles filtrés		
	filtrage simple	filtrage itéré	filtrages + postfilrage
Façade 208	6 429	6 958	6958+2057
Façade 209	8 414	9 092	9082+6092
Donnée simulée (figure 4)	262	392	392+41

Table 2: Résultats de filtrages

2 Fusion

2.1 Choix d'un algorithme

Dans la littérature ([Hil95] [Ben98]), les méthodes volumiques ont souvent la faveur des comparaisons, les autres méthodes ayant des limites géométriques (petits angles et objets fins plus difficilement reconstruits par exemple) plus importantes.
Cependant dans notre contexte, on doit absolument respecter les triangulations obtenues après filtrage, ce qui n'est pas le cas de la fusion volumique. En effet, les méthodes de Hoppe [HDD+92] et celles de Hilton [Hil95] [HI97] retriangulent les maillages et celle et Curless [CL96] rééchantillonne les triangulations. On ne peut pas se permettre d'avoir des points déplacés (les points

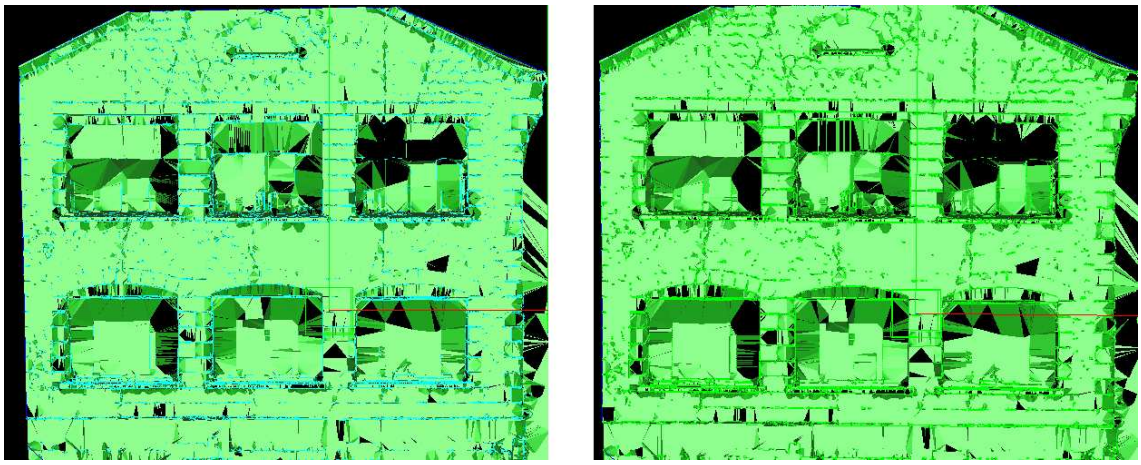

Figure 10: Filtrage simple et filtrage itéré avec "postfiltrage"

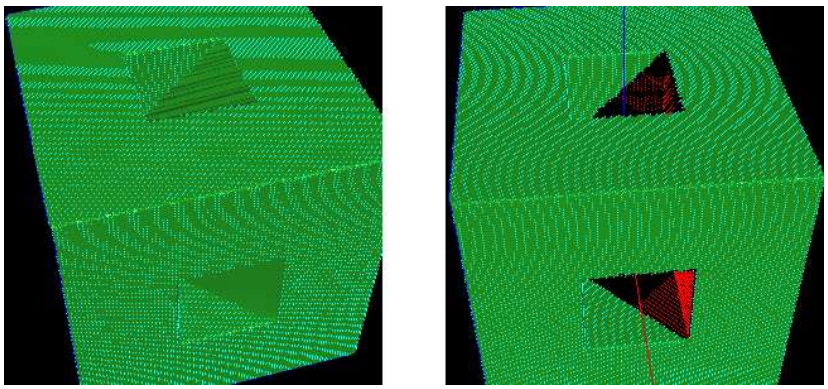

Figure 11: Vue d'un cube et sa version filtrée

acquis étant supposés bons) ou d'avoir une géométrie modifiée.

Les autres méthodes semblent donc plus appropriées.
Rutishauser [RST94] propose dans sa méthode de résoudre les problèmes d'erreurs d'acquisition à l'aide de probabilités. Comme dit précédemment les points sont supposés exacts. Des points risquent donc d'être supprimés s'ils ne correspondent pas avec la surface calculée dans la méthode, ce qui peut générer une déformation d'une géométrie qui était correcte. Ça pourrait être le cas sur des discontinuités telles que des rebords de fenêtres. Cette méthode est elle aussi peu appopriée.
Les méthodes d'intégration de maillages de Soucy et Laurendeau [SL95], de Turk et Levoy [TL94] et de Pito [Pit96] ont l'avantage de ne pas avoir ces inconvénients (déplacements de points et/ou de modification de géométrie). Les limites géométriques des méthodes d'intégration soulignés par Hilton [Hil95] tels que le trou minimal pouvant être reconstruit, les petits angles, les objets fins et les densités d'échantillonage ne sont dans notre contexte, pas des soucis majeurs. En effet, les trous ont été rebouchés lors de l'étape de filtrage de triangles erronés ; ils n'y aura plus de trous lors de la fusion. Aussi notre optique vise à fusionner des façades ; les objets très fins et les petits seront donc rares (on essaiera tout de même de se soucier de ces cas de figures). Enfin, les acquisitions des nuages de points étant de même densités, la fusion ne rencontrera pas a priori de grosse différentes de densité entre les différentes triangulations qui pourraient poser problèmes. Ces limitations étant secondaires (**dans notre contexte**) on peut donc finalement se permettre d'implémenter une des méthodes d'intégration de maillages.

2.2 Algorithme mis en oeuvre

Les algorithmes d'intégration de maillages se décomposent en trois phases.

1. Détermination des recouvrements des triangulations ;
2. Élimination des données redondantes ;
3. Raccordement des triangulations.

Ces algorithmes respectant bien ces phases, il serait tout-à-fait envisageable d'implémenter un algorithme combinant les différentes méthodes.

2.2.1 Détermination de recouvrements des triangulations

Soucy [SL95] détermine les recouvrements de nappes à l'aide d'un test de visibilité (pour des surfaces opposées) et d'un test de voisinage (pour les discontinuités). Ceci permet de savoir si un point est dans l'intersection de plusieurs triangulations.
La méthode de "zippering" [TL94], quant à elle rogne les surfaces redondantes jusqu'à ce que les bords se rejoignent. Cette technique n'utilise pas les deux surfaces pour recontruire la surface finale, ce qui peut être dommageable si l'une des surfaces contient un détail que ne contient pas l'autre. Pito [Pit96] dans son algorithme s'aide des points de vue (avec une test de visibilité) pour ne garder que les surfaces les plus probables.
On peut constater qu'il existe une analogie entre le test de visibilité de Pito et l'étape de filtrage précédement décrite (de la même façon, on ne garde que les triangles "corrects"). En effet le distance-buffer[3] utilisé lors du filtrage réalise "un test de visibilité". Grâce au distancebuffer on obtient directement les parties qui se chevauchent. Il n'est donc pas necessaire d'implémenter cette phase à l'aide d'une autre méthode.

2.2.2 Élimination des données redondantes

Dans un contexte d'un besoin de précision maximal, il peut être intéressant de garder tous les points plutôt que de les jeter. Les nuages de points étant supposés bons, on pourra très bien garder les points des triangles se chevauchant, on obtiendra ainsi une super-résolution sur les parties communes. Évidemment si un point n'apporte rien à la géométrie, il sera inutile de le garder.

2.2.3 Raccordement des triangulations

Une fois les zones de recouvrement trouvées et les redondances traitées, on peut se demander comment raccorder les triangulations. On a utilisé un distance-buffer pour determiner les recouvrements mais on peut aussi bien l'utiliser pour déterminer les parties propres à la deuxième triangulation (triangles en dehors du distance-buffer). Il suffit donc d'ajouter ces triangles à la première triangulation pour avoir la fusion des deux triangulations.
Pour finir, il ne reste plus qu'à faire la jointure entre les triangulations. En effet les triangles aux bords des recouvrements n'apparaissent pas dans le distance-buffer et l'un de leurs sommets peut avoir été supprimé. Ces triangles ne peuvent donc pas être reconstruits. On se retrouve dans le même cas de figure que dans la méthode de Pito [Pit96]. Celui-ci utilise une approche "soudure" (voir figure 6) pour joindre des bordures des triangulations. On utilise son approche dans l'implémentation de notre algorithme (on recherche d'abord les bordures, puis on crée une jonction avec le point le plus proche d'une autre bordure, et on itère tant que cette distance ne dépasse pas un seuil).

2.2.4 Algorithme final de fusion

```
On calcule le distance-buffer de la surface reconstruite à partir
du point de vue A dans la géométrie du point de vue B.
On initialise une pile L de points (qui seront triangulés plus tard).
Pour chaque point-image PI de B :
   Si PI n'appartient pas à une projection d'un triangle de la surface
   dans le distance-buffer :
      On ajoute ce point à la pile L et on passe au point suivant.
   On calcule le point-terrain PT associé à PI.
   On calcule la distance D entre PT et le centre du point de vue B.
   Si D est supérieure à celle du distance-buffer :
      On passe au point suivant (le cas ayant déjà été traité lors du filtrage).
   Si D est égale à celle du distance-buffer (à une tolérance près) :
      On passe au point suivant (le point est considéré comme inutile).
   Si la valeur absolue de (D - distance-buffer) est inférieure au seuil :
      On retriangule ce point dans le maillage.
   Sinon
      Le point se trouvant devant l'objet, on l'ajoute à la pile L.
Pour chaque triangle de B :
   Si ce triangle à ses 3 sommets dans la pile L.
      On triangule les trois sommets ensembles.
```

```
On recherche toutes les bordures que l'on met dans une pile.
Tant que la pile n'est pas vide :
   On dépile une bordure
   On calcule la distance D entre le point le plus proche PT et la bordure.
   Si D est inférieure à un certain seuil :
         On triangule la bordure avec PT.
```

2.3 Résultats

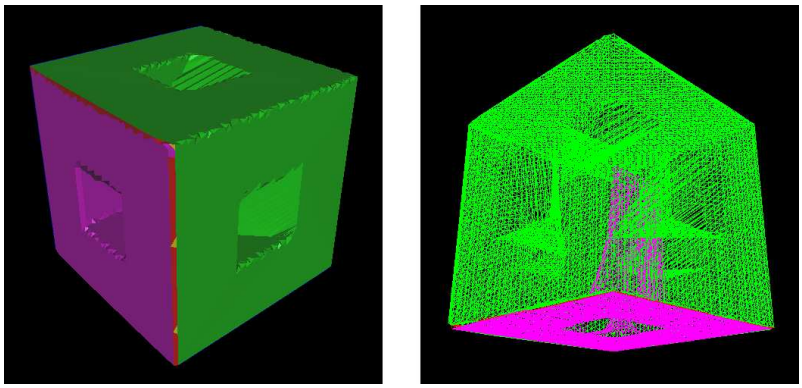

Figure 12: Fusion sur la simulation du cube (la surface violette est celle ajoutée á la surface verte et les raccords sont identifiés par les triangles rouges)

Notre algorithme de fusion donne de bon résultats sur les données simulées (figure 12). En effet les surfaces fusionnés sont bien raccordés et on peut observer (sur l'image de droite) une densité de points plus grande qu'au départ. Les résultats sur données réelles n'ont pu être présentées ici, faute de temps.

3 Conclusion

On a tenté, par cette approche, de répondre aux besoins pour la simplification et la fusion de triangulations.

Pour la simplification, on a implémenté, dans une première étape, un algorithme simple et assez efficace (n'altérant pas la géométrie et divisant le volume des données par deux) permettant de visualiser et de manipuler des données antérieurement volumineuses. Cet algorithme est spécifique aux données de façades.
Dans une seconde étape de la simplification, on a intégré dans les codes de l'IGN, l'implémentation de l'algorithme QSlim de Garland [GH97]. Sa méthode est l'une des méthodes de simplification altérant le moins la géométrie et permet via quelques modifications de pouvoir obtenir une simplification selon un critère d'altération géométrique. Malheureusement, il est difficile d'interpréter géométriquement ce critère. Pour contrôler rigoureusement l'écart géométrique, une méthode ad-hoc pourrait être développée mais sans doute au détriment de son efficacité.

Enfin, pour la fusion, la méthode utilisée est inspirée de la méthode d'intégration de mailles de Pito [Pit96]. Cependant, notre test de visibilité (basé sur un distance-buffer) ne sert pas seulement à determiner les recouvrements et les données redondantes ne sont pas éliminées. Ces adaptations ont permis de mieux répondre à notre contexte ; le filtrage permet de "réparer" des géométries erronées, et les données redondantes peuvent servir à une super-résolution. Notre algorithme de fusion donne de bons résultats (figure 12) mais dépend d'un seuil qui dépend de l'application.

Remerciements Je remercie le laboratoire MATIS et IGN de m'avoir acceuilli en tant que stagiaire. Je remercie aussi mon maître de stage, Lionel Pénard, et tous ceux qui m'ont aidé lors de mon stage.

References

[Ben98] R. BenJemaa. *Traitement de données 3D denses acquises sur des objets réels complexes.* Thése de doctorat, ENST, Paris, October 1998.

[Boi84] J. D. Boissonnat. Geometry structures for three-dimensional shape representation. *ACM Transactions on Graphics*, 1984.

[CL96] B. Curless and M. Levoy. A volumetric method for building complex models from range images. *SIGGRAPH*, 1996.

[COM98] J. Cohen, M. Olano, and D. Manocha. Appearence-preserving simplification. *SIGGRAPH*, pages 115–122, 1998.

[Gar99] M. Garland. *Quadric-Based Polygonal Surface Simplification.* Thése de doctorat, School of Computer Science of Pittsburgh, May 1999.

[GH97] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. *SIGGRAPH*, pages 209–216, 1997.

[GH98] M. Garland and P. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. *IEEE*, pages 263–270, 1998.

[HDD+92] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *SIGGRAPH*, 1992.

[HI97] A. Hilton and J. Illingworth. Multi resolution geometric fusion. *IEEE*, 1997.

[Hil95] A. Hilton. Reliable surface reconstruction from multiple range images. Technical Report VSSP-TR5/95, Departement of Electronic and Electrical Engineering, University of Surney, October 1995.

[Hil97] A. Hilton. Introduction to model building from 3d surface measurements. *CVSSP 3D Vision*, 1997.

[Hop96] H. Hoppe. Progressive meshes. *SIGGRAPH*, 1996.

[LC87] W. E. Lorensen and H. E. Cline. A high resolution 3d surface construction algorithm. *ACM Transactions on Graphics*, 1987.

[LT98] P. Lindstrom and G. Turk. Fast and efficient polygonal simplification. *IEEE*, 1998.

[LT00] P. Lindstrom and G. Turk. Image-driven simplification. *ACM Transactions on Graphics*, 2000.

[Lue01] D. P. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE*, pages 24–5, 2001.

[Pit96] R. Pito. Mesh integration based on co-measurement. *IEEE*, 1996.

[Por04] D. Porquet. *Rendu en Temps Réel de Scènes Complexes*. Thése de doctorat, Université de Limoge, Limoge, November 2004.

[RNFT01] M. Roy, F. Nicolier, S. Foufou, and F. Truchetet. Mesure de la qualité des algorithmes de simplification de maillages. *Journées de l'AFIG*, 2001.

[RST94] M. Rutishauser, M. Stricker, and M. Trobina. Merging range images of arbitrarily shaped objects. *IEEE*, pages 573–580, 1994.

[SL95] M. Soucy and D. Laurendeau. A general surface approach to the integration of a set of range views. *IEEE*, 1995.

[TL94] G. Turk and M. Levoy. Zippered polygon meshes from range images. *SIGGRAPH*, 1994.