REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohammed Khider-BISKRA-

Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie

Département d'Informatique

N°d'ordre :……

Série :……

Mémoire

Présenté en vue de l'obtention du diplôme de Magister en Informatique

Option**: Synthèse d'Images et Vie Artificielle (SIVA)**

Sous le thème :

Modélisation de surfaces sur la base d'une représentation à base de points

Par :

Mr. CHELLOUAI Abdelhey

Devant le jury :

Dr BAARIR Zineddine	**MCA**	**Université de Biskra**	**Président.**
Pr DJEDI NourEddine	**Pr**	**Université de Biskra**	**Rapporteur.**
Dr BABAHENINI Med Chaouki	**MCA**	**Université de Biskra**	**Examinateur.**
Dr CHERIF Foudil	**MCA**	**Université de Biskra**	**Examinateur.**

Soutenue le : 17/10/ 2011

Résumé

La modélisation géométrique de surface issue de points est un des domaines de la synthèse d'image qui consiste à étudier la façon de représenter des objets 3D. Un objet est donc constitué uniquement de points sans aucune information d'adjacence ni de topologie. Ce paradigme est très utilisé ces dernières années par suite du développement considérable des périphériques d'acquisitions, tels que les scanners 3D. Dés lors, ces outils permettent en quelques minutes d'acquérir un nuage de points et permettra donc de modéliser l'objet par des surfaces constitués globalement de points. Par rapport aux modèles géométrique classique (maillages polygonales, surfaces splines, surfaces implicites) les surfaces définies à base de points présentent des avantages spécifiques tels que la transmissions progressives des données ou la représentation multi-échelles, mais également des défauts importants principalement la présence de trous lors de la visualisation.

Notre travail s'inscrit dans le cadre de la modélisation de surfaces sur la base d'une représentation à base de points. Notre principale axe de recherche est d'étudier ce que permet le rapprochement entre ces deux domaines de représentations des surfaces (classiques et à base de points).

Pour cela nous avons développé une technique de reconstruction de surfaces, en partant d'un nuage de points d'un objet 3D à une représentation polygonale. L'idée est inspirée des techniques de contours et plans d'ajustement. L'un des avantages majeur est celui de pouvoir exploiter directement le GPU ainsi que la représentation en multi représentation. Nos résultats sont satisfaisants dans la reconstruction de surface.

Mots Clés

Nuage de points, Modélisation géométrique, Encodage de surfaces, Edition de surface.

Abstract

The geometrical modeling of surface resulting from points is one of the fields of the synthesis of image which consists in studying the way of representing objects 3D. An object thus is composed only of points without any information neither of adjacency nor of topology. These last year's this paradigm is very much used in consequence of the considerable development of the acquisitions peripherals, such as the scanners 3D. Consequently, these tools make it possible in a few minutes to acquire a point's cloud and will thus make it possible to model the object by surfaces overall made up of points. Compared to the models geometrical traditional (grids polygonal, surfaces splines, implicit surfaces) the surfaces defined containing points have specific advantages such as the progressive transmissions of the data or the representation multi-scales, but also of the important defects mainly the presence of holes during display.

Our work is part of the surface modeling based on performance-based points. Our main line of research is to study what makes the connection between these two areas of representation of surfaces (traditional and point-based).

For that we have developed a technique of surfaces rebuilding, on the basis from a point's cloud of object 3D to a polygonal representation. The idea is inspired by the techniques of contours and plans of adjustment. One of the major advantages is that to be able directly exploit the GPU and the representation in multi representation.
Our results are satisfactory in the surface rebuilding.

Keywords

Point clouds, Geometrical modeling, surface Encoding, Editing surface.

ملخص

وضع نموذج لسطح هندسي فرع من فروع إنشاء الصورة و الذي يعتمد على دراسة طريقة تمثيل الأجسام ثلاثية الأبعاد. إذا الجسم يتكون من نقاط فقط و بدون أي معلومة تقريبية و لا فضاء هندسي (طوبولوجيا). هذا المسار أستخدم على نطاق واسع في السنوات الأخيرة نتيجة لسلسلة من التطورات التي اكتسبت في عدة مجالات مثل الناسوخ (الماسح الضوئي) ثلاثي الأبعاد. منذ ذلك الوقت هذه الوسائل سمحت في بضع دقائق بالحصول على سحابة من النقاط ستسمح إذا بوضع نموذج لجسم يمثل بسطوح تتكون عموما من نقاط. بالمقارنة مع النماذج الهندسية الكلاسيكية (المضلعات المنسجمة, السطوح الخدية, السطوح المخفية) السطوح المعرفة بالأساس على نقاط محددة لها ميزات خاصة مثل الانتقال التدريجي للمعطيات أو التمثيل المتعدد المستويات, لكن على حد سواء هناك عيوب هامة أهمها وجود ثقوب عند عرضها.

عملنا ينطوي تحت إطار النمذجة السطحية المستند على رسم يعتمد على نقاط. محور هدفنا الرئيسي في هذا البحث هو دراسة المقاربة بين هذين المجالين المتعلقان بتمثيل السطوح (سواء الطريقة الكلاسيكية أو التي تعتمد على النقاط).

لهذا الغرض قمنا بتطوير تقنية لإعادة إنشاء السطوح, بدءا من سحابة نقطية لجسم ثلاثي الأبعاد إلى تمثيل متعدد الأضلاع. و الفكرة مستوحاة من تقنيات الحدود و مخططات المطابقة. واحدة من أهم المزايا تكمن في الاستغلال المباشر لمعالج الصورة GPU و كذلك التمثيل بأشكال متعددة.

نتائجنا كانت مرضية في ما يخص إعادة إنشاء السطوح.

كلمات المفتاح:

مجموعة نقاط، النمذجة الهندسية، ترميز السطوح، توزيع السطح.

TABLE DES MATIERES

TABLE DES FIGURES

LISTE DES TABLEAUX

CHAPITRE I

INTRODUCTION

I/ INTRODUCTION

Le progrès réalisé du point de vue de la technologie des ordinateurs en termes de puissance de calcul et de capacité de stockage a permis d'envisager des applications de plus en plus complexes. C'était l'un des facteurs qui a initié la pensée d'exploiter le graphisme sur ordinateur. Depuis, la communauté scientifique a mis l'accent sur cet axe de recherche et des efforts considérables ont été déployés conduisant à 'avènement d'une nouvelle discipline désignée aujourd'hui par la dénomination **synthèse d'images**. Un des objectifs de la synthèse d'images est de reproduire la réalité visuelle par ordinateur, puisque les images sont une source d'informations extrêmement importante, d'une part. Et d'autre part, le principe d'utiliser le canal visuel pour communiquer des idées complexes relève de l'efficacité et de la perspicacité, ce qui a conduit à l'envahissement de l'univers par les images de synthèse.

Dans le domaine de la postproduction cinématographique, par exemple, l'avantage principal de ces images est que leur coût de production est bien moindre que leur équivalent réel : il est en effet plus facile de créer informatiquement une légion Romaine en ordre de marche en dupliquant l'image d'une vingtaine de figurants, que de rassembler effectivement cinq mille hommes. Dans le même ordre d'idées, les artistes peuvent créer des situations telles que la rencontre avec des dinosaures avec un réalisme impossible à obtenir à l'aide de marionnettes télécommandées.

Un autre domaine d'application des images de synthèse concerne les applications d'immersion, de *Réalité Virtuelle*, qui permettent, à l'opposé du cinéma, une interaction directe de l'utilisateur. Par exemple, les chirurgiens peuvent s'exercer sur une reconstitution informatique à chercher le meilleur chemin pour une intervention, l'apprentissage du pilotage d'un nouvel avion se fait couramment sur des simulateurs de vol, les présentations commerciales d'un nouveau modèle de voiture peuvent se faire sur image avant même que le moindre prototype n'ait été

construit, la visite de sites archéologiques disparus ou les jeux vidéo en sont autant d'applications. Cette forme de simulation est plus immersive car l'utilisateur est libre d'interagir avec l'environnement simulé. Elle est également plus délicate à mettre en œuvre car la génération des images doit satisfaire des contraintes de temps afin de préserver l'interactivité.

Actuellement, la montée en puissance des stations de travail personnelles et la disponibilité de technologies avancées, intégrées dans leur environnement, ont largement favorisé la percée des micro-ordinateurs dans le monde de l'image et contribué, par la même occasion, à l'élargissement des champs d'application disponibles.

Le processus de création des images se décompose en deux étapes : la *modélisation* et le *rendu*. La modélisation consiste à définir la scène que l'on souhaite visualiser en données interprétables par l'ordinateur. Ces données, telles que la position des surfaces et leur nature ou l'éclairement de la scène, sont utilisées lors de l'étape de rendu pour obtenir des *image*s.

Ces images sont la projection visuelle, compréhensible par l'utilisateur, de modèles mathématiques et physiques de la réalité. Dans ce cadre, le *réalisme* d'une image est dépendant de la précision avec laquelle on définit la scène, ainsi que de la précision avec laquelle on simule la propagation de la lumière. En d'autres termes, pour obtenir une image d'une scène visuellement réaliste, il faudra d'une part la définir le plus précisément possible et d'autre part appliquer les modèles physiques de propagation de la lumière les plus complets. Idéalement, le but à atteindre est le *photo-réalisme*, c'est-à-dire une image de synthèse indiscernable d'une photographie.

Il est évident, en pratique, qu'il soit impossible de prendre en compte tous les phénomènes physiques : les ordinateurs ont un espace de stockage limité et les calculs doivent être accomplis dans un temps raisonnable. Ainsi, de très nombreuses méthodes existent dont le but est d'approximer les modèles physiques ou de contourner les

limitations matérielles. Ces méthodes permettent d'obtenir différents niveaux de réalisme, en fonction des besoins.

La synthèse d'image fait une grande consommation de modèles afin de permettre la production d'une présentation graphique. On peut répertorier au moins les modèles suivants :

- le **modèle géométrique**, qui permet de définir et représenter les formes des objets : les différentes familles de modèles géométriques diffèrent par les primitives utilisées, les possibilités de structuration de la scène et les facilités de passage d'un modèle à un autre ;
- le **modèle d'aspect**, qui permet de décrire l'apparence extérieure d'un objet : on range essentiellement dans cette catégorie les techniques de modélisation de textures ;
- le **modèle d'illumination**, qui tient compte de tous les paramètres définissant les jeux d'ombre et de lumière sur un objet : les images dites réalistes sont caractérisées par la grande richesse du modèle d'illumination ;
- le **modèle de vision**, qui correspond à la notion de prise de vue : il autorise la définition du type de projection employée, du placement de l'observateur, et dans certains cas, des mouvements de la caméra (ou de l'appareil photographique) ;
- le **modèle d'affichage**, qui tient compte des potentialités des surfaces de visualisation. On distingue ainsi couramment les dispositifs de production de dessins au trait et ceux produisant des images. D'autres paramètres peuvent encore intervenir : le nombre de points pour une image donnée, le nombre de couleurs différentes susceptibles d'être utilisées.

C'est la collaboration entre ces différents modèles qui permet de construire l'image finale.

Nous nous intéressons particulièrement au modèle géométrique qui représente la phase la plus délicate à satisfaire vis-à-vis des autres modèles. Plus la scène est bien représentée plus la qualité de l'image est satisfaisante.

En réalité la représentation d'objets est un domaine varié en techniques de modélisation ou de représentation d'objets. Nous allons présenter dans la partie état de l'art les différentes techniques de représentation puis nous introduirons dans le détail notre domaine de recherche.

CHAPITRE II

DOMAINE DE REPRESENTATIONS

II/ DOMAINE DE REPRESENTATIONS

1. Modélisation géométrique

La technique de représentation des objets la plus utilisée auparavant est celle qui a tiré profit d'une grande partie de la géométrie 2D ou 3D et présente aussi une large gamme d'outils mathématiques destinés à définir le mieux possible des objets de la scène à construire. Partant du simple tracé de droite, courbe et aussi les primitives géométriques telles qu'un carré, rectangle, polygones et enfin des surfaces et volumes complexes. Cette technique repose sur des aspects mathématiques très robustes.

Parmi les représentations classiques connues, nous pouvons citer:

La représentation fil de fer : premier modèle à être utilisé, dans celui-ci les objets en fil de fer sont constitués d'éléments filiformes, tels que des segments de droites ou des courbes. Ils sont décrits par la donnée des points de l'espace (les sommets) et les arêtes qui joignent certains sommets entre eux. Si ces objets sont extrêmement simples à manipuler, il est évident qu'ils ne suffisent pas pour décrire le monde réel. En particulier, la notion de parties cachées n'existe pas.

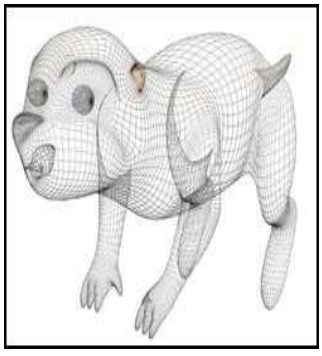


Figure 1-Représentation en fil de fer.

Par contre, la réalité est d'autant plus complexe et ne se limite pas à des formes simples telles qu'un triangle ou un carré, cependant il existe des modèles plus complexes et permettant de mieux représenter ou façonner un objet de la réalité.

1 .1- Modélisation surfacique

On distingue ici 3 formes de modélisation de surfaces [Thal03] :

a- *Les surfaces par balayage* telles que les cônes, les cylindres, les surfaces de révolution.

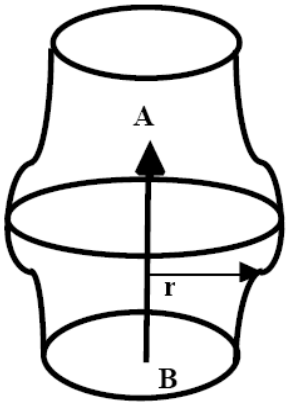


Figure 2- Balayage Rotationnel [Thal03].

b- *Les surfaces paramétriques* données par leurs équations :

$$y = Y(u, v); z = Z(u, v)$$

c- *Les surfaces libres* ce sont ces dernières qui permettent le plus convenablement de modeler des objets compliqués; on distingue 4 principaux types de surfaces libres:

Les surfaces de Coons, les surfaces de Bézier, les surfaces B-splines et les surfaces β-splines.

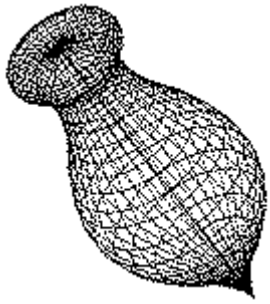


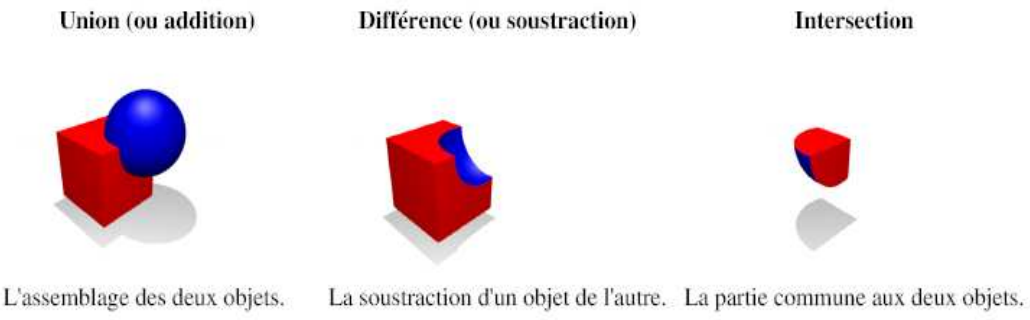
Figure 3- Représentation de surface libre.

1.2- Modélisation volumique

La génération de solides ou volumes est très importante dans le domaine de la Conception Assistée par Ordinateur (CAO). En effet, comment considérer une pièce mécanique ou un pont du point de vue purement surface alors qu'il faut être capable de faire des ouvertures ou de combiner les volumes. Pour cette raison, la géométrie des solides a été développée. Elle repose principalement sur deux types de représentation: *constructive solid geometry (CSG) et les modèles de représentation par les frontières.*

En image de synthèse et en animation tridimensionnelle, il y a aussi un besoin de représenter des objets 3D solides. Par exemple, il devient beaucoup plus facile d'effectuer des opérations ensemblistes (union, intersection et différence) sur des solides. Le stockage d'objets complexes sous forme CSG ou octree est particulièrement efficace. Ces techniques sont aussi très utiles dans la méthode du lancer de rayons. Il existe quatre types de modeleurs principaux (octree, CSG, Brep, hybride et paramétrique) [Thal03].

a- Opérations booléennes (CSG)



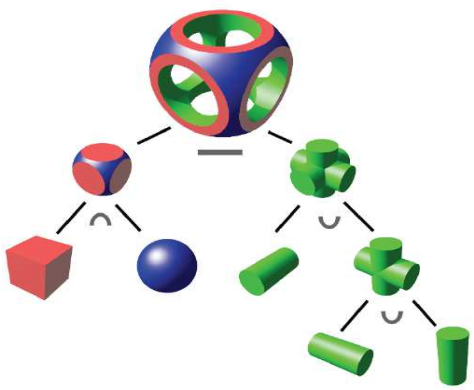
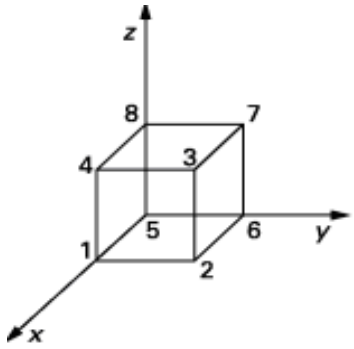
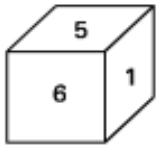


Figure 4. Arbre du modèle CSG.

b- Enumération des facettes (Brep)



N° des sommets	Ordonnées x	y	z
1	1	0	0
2	1	1	0
3	1	1	1
4	1	0	1
5	0	0	0
6	0	1	0
7	0	1	1
8	0	0	1



N° des faces	Sommets			
1	2	3	7	6
2	6	5	8	7
3	5	1	4	8
4	2	6	5	1
5	3	7	8	4
6	1	2	3	4

1.3- Modélisation complexe

Il existe un certain nombre de mécanismes permettant de décrire de nombreux phénomènes de croissance et de développement dont ceux que l'on rencontre dans la nature. Ces mécanismes sont qualifiés de mécanismes de réécriture. La technique de base consiste, à partir d'un élément initial ou d'une configuration initiale, à remplacer successivement des parties en fonction d'un ensemble de règles. Ceci évoque, immédiatement, la possibilité de formaliser cette technique suivant des grammaires et des langages formels. Plusieurs approches, ayant des liens, d'ailleurs, ont été proposées: les fractales, les IFS, les automates cellulaires et les L-systèmes [Thal03].

1.4- Techniques de rendu

On peut découper le travail de rendu en deux phases : déterminer les surfaces visibles à l’écran, puis calculer la couleur de l’élément de surface (ou la couleur moyenne des éléments) apparaissant en chaque pixel en tenant compte des caractéristiques d’orientation et de matière de celles-ci, ainsi que des conditions

d'éclairage. De nombreuses techniques ont été développées dans ce but ; on ne détaillera que les deux les plus utilisées, à savoir le tampon de profondeur (*Z-buffer*) et le lancer de rayons (*ray-tracing*), on évoquera ensuite le rendu volumique par lancer de rayon.

***a-** Le Z-buffer et ses extensions*

L'origine du *Z-buffer* est donnée à Catmull [Cat74]. Le principe consiste à laisser à une extension du concept de mémoire d'image le soin de déterminer la visibilité des surfaces. Le *Z-buffer* est une mémoire, "un tampon des profondeurs" qui sert à stocker la profondeur de chaque pixel visible à l'écran. Durant le rendu, la profondeur d'un élément candidat à paraître dans le pixel (fragment) est comparée à la valeur de la profondeur déjà stockée pour ce pixel : si cette comparaison indique que le nouveau fragment est devant celui stocké, alors celui-ci remplace le contenu du pixel écrit dans la mémoire image et le *Z-buffer* est mis à jour. La complexité de cet algorithme est proportionnelle aux nombres de primitives traitées et à leur surface à l'écran. Pour en améliorer encore l'efficacité, Greene proposa une version hiérarchique dans [GK93].

De nombreuses extensions à cet algorithme existent, la plus intéressante est peut être le *A-buffer* [Car84] qui conserve pour chaque pixel une liste des fragments candidats à y apparaître et calcule pour chaque fragment la proportion du pixel qu'il recouvre. Ceci permet de traiter les problèmes d'aliassage et d'intégrer des objets semi-transparents. Une autre amélioration [HA90] est le tampon d'accumulation (*accumulation buffer*) qui permet d'accumuler plusieurs images en déplaçant légèrement la caméra entre chaque rendu, soit pour réduire l'aliassage, soit pour obtenir un effet de flou de mouvement (*motion blur*) ou de mise au point (profondeur de champ). L'autre solution pour réduire l'aliassage est de subdiviser les pixels. La couleur finale du pixel est obtenue en moyennant les sous pixels. À noter que cette technique revient à effectuer le rendu à une résolution supérieure de celle de l'image (sur-échantillonnage).Cet algorithme

ne traite pas les ombres, ni les réflexions et le traitement de la transparence nécessite un tri des polygones avant de les envoyer à la carte graphique. Un fragment de transparence α et de couleur c_f ayant réussi le test de profondeur sera composé de la manière suivante avec la couleur c_p déjà stockée dans le pixel: $c_p = \alpha \times c_\alpha + (1 - \alpha) \times c_p$, ce qui correspond bien à un calcul de transparence si tous les fragments se trouvant derrière celui-ci ont déjà été tracés (d'où le tri des fragments en prétraitement).

Un des avantages du *Z-buffer* est la possibilité de réaliser facilement des implémentations matérielles très efficaces. Aujourd'hui les moteurs graphiques à base de *Z-buffer* permettent le rendu et l'affichage de scènes de plusieurs centaines de milliers de polygones en temps réel.

***b**- Lancer de rayons*

Le principe du lancer de rayons (*ray-tracing*) [Whi80, GAC+89] est de calculer les objets visibles en remontant le chemin de la lumière parvenant à la caméra, *i.e.* en lançant des rayons à travers tous les pixels de l'image (fonctionnement inverse d'un appareil photo). Comme le principe du lancer de rayons est très simple à programmer, on peut lui faire simuler toute l'optique géométrique et ainsi prendre en compte de nombreuses caractéristiques optiques comme la réflexion, la réfraction, l'ombrage, etc. Notamment les reflets de la scène sur une surface lisse sont obtenus en envoyant un rayon secondaire depuis la surface dans la direction miroir à celle d'arrivée par rapport à la normale. Comme le *Z-buffer*, la deuxième partie du processus consiste alors à calculer la couleur que vont avoir les objets en faisant appel à un modèle d'illumination pondéré par l'éclairage (*i.e.* l'ombrage).

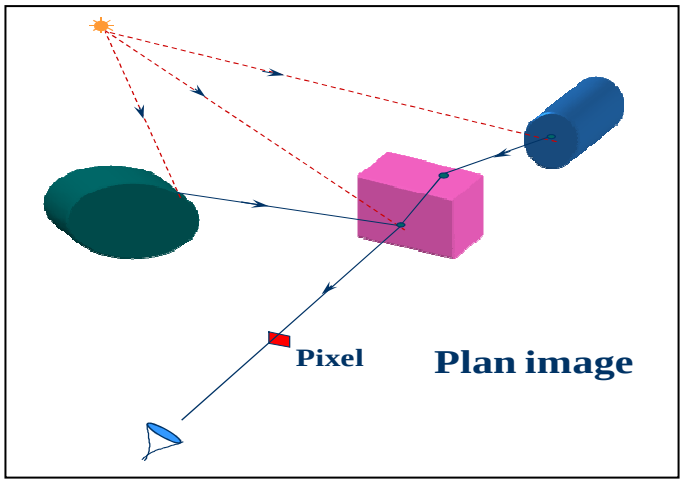


Figure 5. Lancer de rayons.

Le principe de calcul des ombres est identique : pour savoir si un objet est éclairé, on lance un rayon d'ombre vers les sources de lumière.

Le coût de cet algorithme est important : pour chaque pixel de l'écran il est nécessaire de lancer un rayon, voir plusieurs avec la technique de sur-échantillonnage. Avec un lancer de rayons non optimisé, pour chaque rayon on calcule son intersection avec toutes les primitives de la scène. De plus, pour chaque primitive intersectée d'autres rayons sont lancés (réflexions, ombres, etc.). Pour diminuer le nombre d'intersections à calculer, et donc le coût, il est indispensable de recourir à des techniques d'optimisations. Un des problèmes important que doit traiter le moteur de rendu est l'aliassage. Un arbre à lui seul compte plusieurs centaines de branches et plusieurs milliers de feuilles ; on peut donc imaginer la quantité de primitives que compte une forêt et donc le nombre moyen de primitives par pixel (le pire étant à l'horizon). Lors du rendu d'une image il est évident que toutes les primitives ne sont pas visibles explicitement, par contre elles contribuent toutes à l'aspect global de la scène (par exemple les aiguilles d'une forêt de sapins ne sont pas toutes visibles individuellement mais si on les supprime il ne reste que les branches !). Plusieurs centaines de primitives contribuent à la couleur d'un pixel, il faut donc, à défaut de toutes les traiter, en considérer suffisamment pour diminuer au maximum les problèmes d'aliassage.

La solution couramment utilisée à ce problème d'aliassage en lancer de rayons est le sur-échantillonnage. Cela revient à lancer plusieurs rayons par pixel, distribués aléatoirement, puis à moyenner les résultats pour donner la couleur finale. Le nombre de rayons lancés peut-être adaptatif, c'est-à-dire varié d'un pixel à l'autre en fonction du nombre d'objets présents dans le pixel : ce nombre n'étant pas connu à l'avance, on décide, après avoir lancé plusieurs rayons, de poursuivre tant que l'écart type des couleurs est supérieur à un certain seuil paramétrable. Cette méthode statistique donne un bon rapport efficacité/coût pour des scènes d'intérieur ou des scènes peu fouillées, c'est-à-dire lorsque le nombre de rayons lancés par pixel n'excède pas dix ou vingt. Cependant, dès que la complexité de la scène est trop importante son efficacité est limitée :

Soit on lance un nombre de rayons suffisant, mais les temps de calcul explosent, soit l'aliassage se fera rapidement sentir, surtout lors du calcul d'animation (car la cohérence temporelle n'est pas respectée).

L'autre solution au problème d'aliassage est de calculer l'intégrale de l'ensemble des objets se trouvant dans le pixel : c'est ce que se propose de faire le lancer de faisceaux [HH84] (beamtracing) ou de cônes [Ama84] (cône-tracing) en lançant un rayon pyramidal ou conique par pixel.

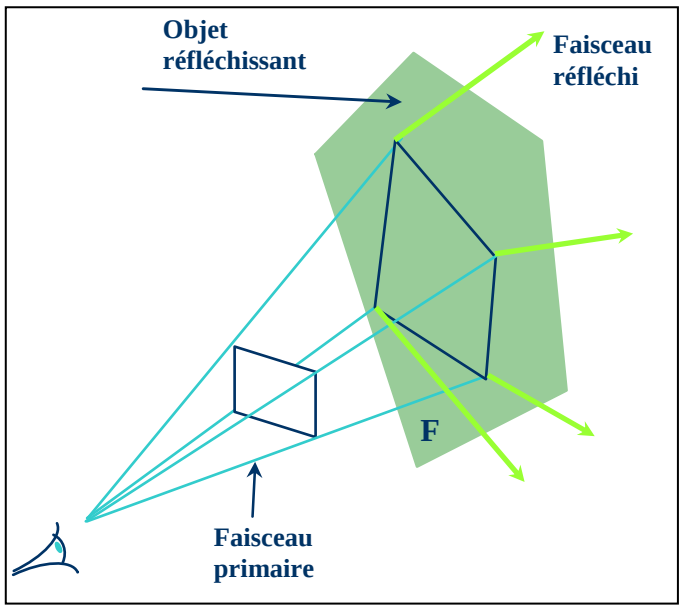


Figure 6. Lancer de faisceaux.

L'avantage de cette technique de beamtracing est de capturer tous les objets d'un pixel en lançant un unique rayon, l'inconvénient est qu'elle augmente la complexité du calcul pour chaque primitive. En effet, l'intersection entre une pyramide (ou un cône) et une primitive de la scène est plus compliquée à calculer que l'intersection entre une droite et cette primitive, comme c'est le cas avec le lancer de rayons classique. De plus, pour un rayon représenté par une droite, le calcul de l'illumination s'arrête à l'intersection de l'objet le plus proche de l'œil, alors que dans le cas du cône-tracing, on doit tenir compte de la surface du pixel recouverte par la primitive, et continuer avec les autres primitives se trouvant derrière s'il reste une portion non couverte.

Le sur-échantillonnage est plus simple à implémenter que le beamtracing, mais pour une scène comportant de nombreuses primitives, petites par rapport à la taille d'un pixel une fois projetées, la technique de beamtracing donnera de meilleurs résultats en termes d'aliassage et sûrement même en termes de coût de calcul.

La technique du lancer de rayon offre une meilleure qualité de rendu avec un réalisme impressionnant mais les coûts de calcul sont très élevés. Cependant les techniques d'optimisation du lancer de rayon sont nombreuses. Le principe général est la subdivision de l'espace objet avec pour but la limitation du nombre d'intersections rayon-objet à calculer. Les deux principales techniques sont les volumes englobant (sphères ou boîtes) et les grilles, éventuellement récursives (octree) [SB87, JW89]. De nombreux travaux portent sur ce domaine, pour un éventail de toutes ces techniques, se référer à [GAC+ 89, Hai].

c- *Rendu volumique par lancer de rayons*

Le rendu volumique est destiné à calculer une image 2D d'une structure de données représentée par une grille 3D (chaque case étant appelée un voxel). Un exemple typique de données est une grille où chaque voxel contient une

densité (i.e. une densité de tissus humains). Comme la technique classique de rendu du ray-tracing vu précédemment, le rayon parcourt le volume de données en utilisant un algorithme de tracer de droite discrète et accumule au fur et à mesure la transparence, ainsi que la couleur si celle-ci est une donnée de la grille. Pour obtenir la quantité de lumière que reçoit chaque voxel il faut lancer un rayon vers chaque source de lumière (comme dans le cas du lancer de rayons classique).

1.5- Pipeline graphique

On présente ici un modèle général du pipeline implanté dans les cartes graphiques résumant ainsi toutes les étapes depuis la définition d'un objet jusqu'à l'étape de son *rendu*. Chaque primitive doit franchir toutes les étapes et le pipeline peut être implémenté de diverses manières avec des étapes en hardware et d'autre en software.

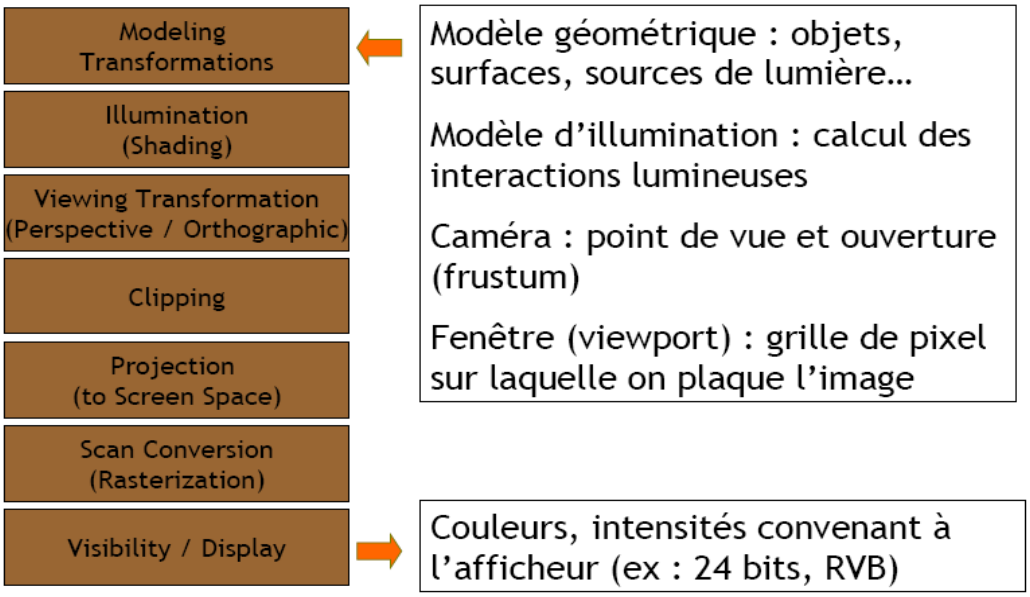


Figure 7. Pipeline graphique.

1.6- Simplification de maillage

La représentation classique bénéficie largement des avantages du pipeline graphique et s'adapte efficacement pour son rendu mais lorsque la scène devient complexe le temps de calcul du rendu devient aussi long et dépendra de la capacité du GPU surtout dans un rendu temps réel. Par exemple, pour une scène composée de plusieurs objets représentés par plusieurs millions voire milliards de polygones, le temps que mettra le GPU à rendre cette image dépendra énormément de sa capacité de traitement. Pour remédier à cette complexité, des techniques ont été proposées afin de contourner les limitations des GPU en termes de rendu (affichage) en temps réel qui permettent d'optimiser l'affichage d'objets composés d'un très grand nombre de polygones en en réduisant le nombre, ce sont les techniques de simplification de maillage. Dans le cas de la simulation d'environnements virtuels, ces techniques sont utilisées quand l'objet à visualiser est loin de la caméra et que les détails fins de sa géométrie ne sont pas ou sont peu apparents et ne nécessitent donc pas d'être affichés. Par exemple, la figure 7 montre différentes versions d'un modèle de base composé de 10000 polygones. Chacune des versions simplifiées du modèle de base constitue un niveau de détail (LOD, Level Of Detail) de celui-ci.

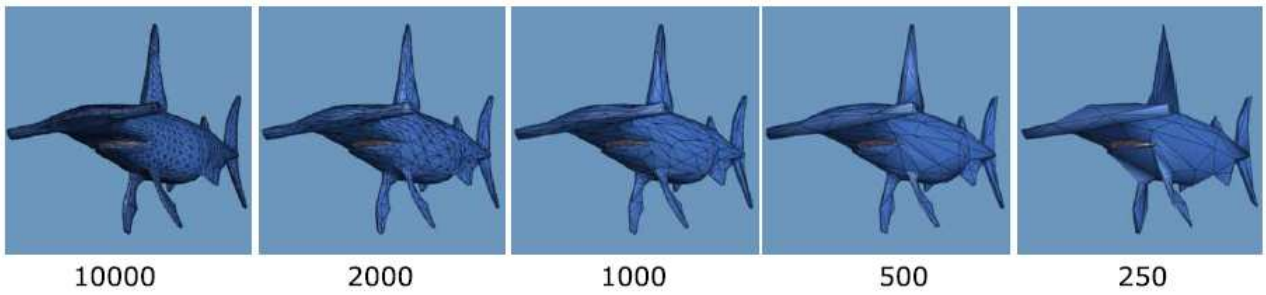


Figure 8. Simplification Géométrique [Hop96].

Ces techniques sont adaptées à différents cas de figures et peuvent être différenciées selon des critères tels que :

- Le traitement de la topologie du maillage original.
- La continuité de la transition entre deux niveaux de détails.

- La prise en compte des attributs des sommets, comme les coordonnées de texture, qui sont modifiées lors de la simplification et affectent le résultat visuel final.
- L'adaptation au point de vue, c'est à dire le but de la simplification : Obtenir à tout prix un nombre fixé de polygones (orientation budget) ou obtenir un maillage simplifié minimisant l'erreur commise (orientation erreur).

On distingue trois classes d'algorithmes : les algorithmes à niveaux de détails statiques, les algorithmes à niveaux de détails continus et ceux qui s'adaptent au point de vue [PD04].

1.6.1 Simplification à niveaux de détails statiques

Principe : La simplification statique correspond aux méthodes générant un ensemble discret de versions simplifiées d'un modèle complexe donné. Lors du rendu, il suffit de choisir le bon modèle à afficher en fonction de la distance d'observation. Différents types d'algorithmes ont été développés pour générer ces maillages simplifiés. Nous ne détaillerons pas leur fonctionnement, mais citons quelques exemples de ces méthodes telles la décimation de maillage [SZL92] qui consiste à choisir des sommets dans le maillage complexe et de les supprimer. Les trous ainsi créés sont remaillés en employant diverses méthodes qui réduisent le nombre final de faces tout en préservant la topologie du modèle. L'algorithme procède en plusieurs passes en choisissant à chaque fois si un sommet peut être supprimé selon deux critères : le sommet sera enlevé si sa suppression préserve la topologie du maillage et si la retriangulation du trou qu'il crée donne des faces à une distance du maillage originel inférieure à un certain seuil.

Avec cette méthode, les maillages simplifiés créés sont des sous-ensembles du maillage de base, ce qui permet de préserver les attributs des sommets (couleur, normale, coordonnées de texture). De plus, la qualité des modèles simplifiés peut être améliorée en déplaçant les sommets créés pour qu'ils améliorent l'aspect visuel final.

Néanmoins le fait qu'ils préservent la topologie du maillage empêche une très grande simplification. D'autre part, ces algorithmes sont très longs à calculer.

Un autre type d'algorithme, le regroupement de sommets (Vertex Clustering) et procède, comme son titre l'indique, en regroupant certains sommets entre eux. Pour cela dans un premier temps, une importance est attribuée à chaque sommet du maillage complexe selon divers critères tels que la courbure de la surface en ce point ou la taille des faces qui lui sont connexes. Ainsi, des sommets attachés à de grandes faces ou des sommets de grande courbure seront considérés plus importants que les autres et donc seront gardés le plus longtemps possible. Ensuite, une grille 3D est superposée au modèle puis tous les sommets appartenant à une cellule sont regroupés au sommet le plus important de la cellule. Ainsi, plus la grille est grosse, plus la simplification est importante.

Néanmoins, la topologie du modèle original n'étant pas préservée, les résultats visuels sont moins satisfaisants qu'avec la méthode précédente. De plus, on ne peut pas prévoir le nombre de polygones générés.

Avantages : De façon générale, ces algorithmes sont faciles à utiliser dans une application de réalité virtuelle interactive car les modèles sont pré calculés à l'avance : on génère les différents niveaux de détails d'un modèle et on les utilise directement lors du rendu, en choisissant l'un ou l'autre en fonction de la distance d'observation.

Inconvénients : Le plus gros problème de ces modèles pré-générés utilisés dans le cadre d'une simulation est que la transition entre un niveau de détail et un autre peut donner un effet de saut (popping) très gênant. Ceci est dû au fait qu'entre un niveau de détail et un autre, des polygones apparaissent ou disparaissent brusquement.

1.6.2 Simplification à niveaux de détails continus

Principe : Pour résoudre ce problème de transition brutale entre les différents niveaux de détails, une autre classe de méthode est apparu. Le principe est non plus de générer séparément des maillages, mais de créer une structure générant au vol un maillage adapté au niveau de détail que l'on souhaite.

Une telle structure est dite à niveaux de détails continus. (CLOD, “Continuous Level of Detail”). Plusieurs méthodes utilisent le principe de réduction des arêtes pour simplifier le maillage. Elles diffèrent sur le choix des arêtes à réduire, ou le moment de le faire. La réduction d'arête, et son dual la séparation de sommet, sont présentés sur la figure 8.

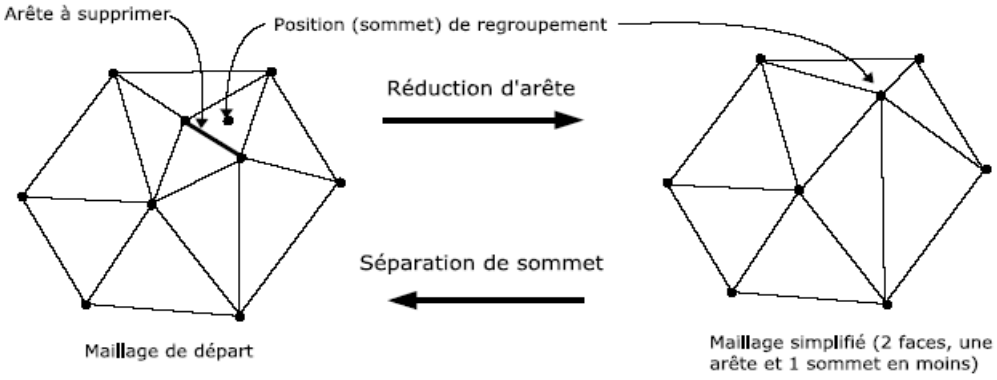


Figure 9. Réduction d'arête et séparation de sommets [PD04].

L’une des méthodes connue est celle des maillages progressifs (Progressive Meshes) de Hoppe [Hop96], qui procèdent également par réduction d'arête mais généralisent le processus de plusieurs façons. Le choix des arêtes à réduire est basé sur l'évaluation d'une fonction d'énergie à minimiser. Par ailleurs, il enregistre dans une structure de données adaptée les opérations de réduction de façon à ce que l'opération inverse de création de sommet puisse s'effectuer avec la même fonction. De plus, il guide 'évaluation de cette fonction sur les attributs du maillage, qu'il sépare en deux catégories : les attributs discrets (indice du matériau, identifiant de texture) et continus

(position, coordonnées de texture, normale). Ainsi, la simplification s'adapte aux caractéristiques du maillage et plus uniquement à une erreur géométrique.

La rapidité du calcul est telle que l'algorithme peut être utilisé en temps réel directement (employer dans la bibliothèque DirectX). De plus, la simplification adaptée aux attributs du maillage de façon unifiée donne de très bons résultats : la silhouette et l'apparence générale sont préservées. Néanmoins, comme tous les algorithmes respectant la topologie du modèle originel, les grosses simplifications ne sont pas possibles : les trous restent et jamais deux parties du modèle ne fusionnent.

Avantages : Les avantages de la simplification continue de maillage sont donc en premier lieu que les problèmes de saut entre niveaux de détails disparaissent. De plus, il n'est pas nécessaire de stocker en mémoire les différents maillages car ils sont générés au vol directement.

Inconvénients: Néanmoins ces algorithmes demandent dans l'ensemble beaucoup de temps de calcul. Même les Maillages Progressifs de Hoppe qui sont très rapides à calculer ne sont guère employés encore car dans un environnement virtuel ce n'est pas un seul modèle qui doit être simplifié mais souvent des dizaines en même temps. Dans ce cadre, les maillages statiques sont toujours utilisés car la vitesse de l'application prime sur l'espace mémoire demandé.

Un autre problème dont souffrent ces méthodes est leur inadéquation avec un environnement virtuel complet : elles ne simplifient qu'un modèle isolément alors que l'on imagine aisément que des groupes d'objets vus de loin pourraient se ramener à un seul maillage simplifié.

1.6.3 Simplification dépendante du point de vue

Principe : Ces méthodes adaptent la simplification en fonction de la position d'observation. Ainsi, un modèle occupant différent niveau de profondeurs à l'écran verra sa partie en avant-plan très détaillée et sa partie en arrière plan simplifiée.

Hoppe étend sa méthode des Maillages Progressifs en les raffinant en fonction du point de vue [Hop97]. Il utilise trois types de simplification : d'une part il supprime classiquement les parties du modèle hors du champ de vision ainsi que les faces non orientées vers l'observateur, d'autre part il adapte sa fonction d'énergie de façon à ce que l'erreur qu'elle évalue soit basée sur une déviation visuelle à l'écran. L'erreur visuelle commise par la déviation géométrique du maillage simplifié est bornée par un seuil fixé. Un des aspects positifs de cette évaluation d'erreur sur l'écran est qu'en particulier les bordures du maillage sont bien préservées.
Cet algorithme fonctionne assez rapidement pour du temps réel, mais est toujours adapté à un seul modèle.

Avantages : Le fait d'adapter la résolution du maillage au point de vue permet d'éviter une simplification arbitraire : les critères de simplification sont adaptés au dispositif d'affichage, et correspondent à une erreur visuelle concrète.

Inconvénients : Comme pour toute méthode évoluée, l'adaptation au point de vue de la simplification est plus lourde à calculer. Néanmoins, les avantages peuvent être drastiques, comme typiquement dans le cas de terrains.

Conclusion :

La simplification de maillage ne résout pas la totalité des problèmes de l’affichage en temps réel de scènes complexes étant données que le calcul de

correspondance en pixel des polygones à affichés sur scènes peut se limiter à un seul et tout le calcul pour le rendu devient inutiles et conduit à une perte énormes de temps de traitement.

Une autre représentation alternative pour remédier à cette complexité prend suffisamment de l'intention des chercheurs et exploré intensivement. Cette technique se base sur la notion de point comme primitive universelle de rendu que nous allons détailler dans la partie suivante.

2. Représentation à base de points

Ces dernières années, les primitives ponctuelles ont reçu une intention croissante à l'infographie. Il y a deux principales raisons de ce nouvel intérêt pour les points: D'une part, on a assisté à une augmentation spectaculaire de la complexité des modèles graphiques polygonaux. Les frais généraux de gestion, le traitement et la manipulation de très grandes mailles polygonales ont conduit de nombreux chercheurs à remettre en question l'utilité future du polygone comme primitive graphique fondamentale. D'autre part, les 3D modernes photographie numérique et les systèmes de balayage 3D qui permettent aisément l'acquisition des objets complexes du monde réel. Ces techniques génèrent d'énormes volumes de points échantillonnés et ont créé le besoin d'un traitement numérique avancé. Conceptuellement, les points constituent les éléments numériques atomiques de la géométrie d'objet et l'apparence - tout comme les pixels qui forment les éléments numériques d'images en 2D.

Historiquement les points n'ont pas été utilisés beaucoup dans le graphisme. En dépit de quelques exceptions, tel que les systèmes de particule de Reeves, le rapport de Levoy et Whitted en 1985 constitue probablement le premier sérieux essai d'employer les points comme primitifs de rendu et donc marque le commencement du graphisme basée sur le point.

Cependant, leur concept réellement révolutionnaire a été abandonné par les auteurs bientôt après et c'était seulement dans la fin des années 90 que les points ont regagné l'intérêt dans le graphisme. Bien sûr, les disciplines apparentées, telles que la géométrie computationnelle, ont couvèrent constamment sur les années, principalement pour le but de la (triangulation) reconstruction de surface [Gros06].

Cette motivation inclut l'efficacité du rendu d'objet et environnement très complexe, la simplicité de l'algorithme du rendu et l'émergence des scanners 3D d'acquisition qui produisent un nuage très dense de point qu'on veut directement visualiser.

L'objectif du *rendu à base de point* est donc la visualisation de la représentation de point sur écran comme étant une surface continue. D'autre part, *la modélisation à base de point* dénote la représentation de surface à base de point y compris le traitement et l'édition, cette technique induit le besoin d'avoir un pipeline de traitement de la géométrie depuis l'acquisition à la visualisation (rendu). Enfin, *la représentation de surface à base de point* permet de traiter le nuage de point comme s'il était une surface sur laquelle on peut effectuer des modifications, opération de lissage et le lancer de rayon.

2.1- Attribut de point

Les attributs du point se résument ainsi:
La position (x, y, z), une normale, une couleur.

Si on assigne une **surface** au point échantillonné ça deviendra un élément de surface ou *surfel - surface element*. Aucune distinction entre le point échantillonné et surfel les deux termes sont interchangeables. A l'exception, surfel est utilisé dans le rendu à base de point et le « point » utilisé dans la modélisation à base de point. La surface assignée à un surfel peut être exprimé par un rayon dans le but de couvrir complètement la surface d'un objet et assurer une couverture des trous lors de la reconstruction.

2.2- Rendu à base de point

Le rendu à base de points peut être considéré en tant que reconstruction extérieure de ces échantillons de points. C'est-à-dire, étant donné un ensemble de point, l'algorithme de rendu dessine une surface douce représentée par l'ensemble de points. Cette reconstruction est une vue dépendante et doit être répétée pour chaque point de vue.

Plusieurs approches existent, elles utilisent directement l'ensemble des points pour le rendu et la plupart procèdent dans l'espace objet. Les primitives de points sont envoyées directement au pipeline et contribuent à la reconstruction de l'image finale.

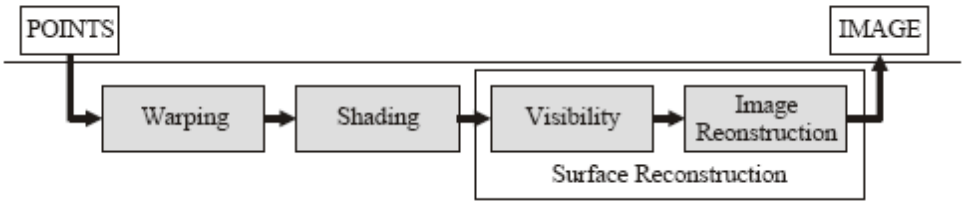


Figure 10. Pipeline du rendu à base de point.

Le schéma ci-dessus montre le pipeline du rendu à base de points. Il accepte des points sur son entrée et produit des images comme rendement. C'est le noyau de n'importe quel système de rendu basé point. La canalisation du tracé très simple est purement vers l'avant, elle est haute là ou il y a une localité de données (chaque point diffuse toute son information), aucune consultation de texture. Par conséquent le rendu de point est très efficace et favorable à l'exécution du matériel.

L'étape du Warping (déformation) projette chaque point pour examiner l'espace en utilisant la projection de perspective. Ceci peut être fait par le produit homogène matrice/vecteur. À l'étape shading (ombrage), l'ombrage par point est exécuté, n'importe quel modèle d'ombrage local est applicable. Le processus d'ombrage est souvent exécuté après visibilité (seulement des points évidents sont ombrés) ou après reconstruction de l'image en utilisant des attributs interpolés de points (par-Pixel ombré). Les deux prochaines étapes de reconstructions et de visibilité de l'image, forment ensemble une reconstruction extérieure dépendante de

la vue dans l'espace d'écran. Elles peuvent être séparées ou simultanées. La reconstruction extérieure inclut sur option le filtrage pour empêcher le crénelage.

Dans ce qui suit, us décrirons brièvement une technique récente du rendu à base de points.

2.3-Surfels

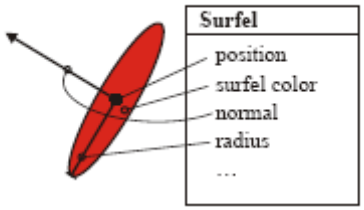


Figure 11. Attributs d'un surfel.

C'est une technique de rendu interactive d'environnements 3D complexes dans lesquels la primitive géométrique fondamentale n'est pas le triangle mais le "surfel" (surface element). Un surfel est tout simplement un point-échantillon 3D d'un objet, accompagné d'un "sprite" 2D correspondant à sa visualisation à l'écran. Le principe de cette technique a été proposé initialement par Grossman et Dally en 1998. Ensuite, trois extensions de cette technique ont été présentées indépendamment par Pfister et al. [PZvBG00], Rusinkiewicz et al. [RL00] et Tobor [TGS00]. La caractéristique principale de cette approche est que la manipulation et la visualisation des surfels se base entièrement sur les fonctionnalités offertes par les cartes graphiques 3D. Autrement dit, chacune des étapes de l'algorithme est décomposée en une série d'opérations élémentaires qui peuvent être effectuées par le hardware. Par rapport aux techniques existantes pour lesquelles une partie de l'algorithme est implémenté de manière logicielle, ce mode de fonctionnement permet de mélanger dans une même scène des objets définis par surfels et des objets définis par triangles tout en bénéficiant d'un rendu par le hardware pour l'ensemble de la scène. Les différents intérêts de la représentation par surfels sont:

– Chaque modèle géométrique (maillages polygonaux, carreaux splines, surfaces implicites, arbres CSG, …etc.) peuvent être facilement convertis en surfels. Ce processus, appelé surfelisation, est basiquement un processus de rastérisation 3D.

Deux algorithmes de surfelisation se présentent, l'un basé sur un processus logiciel de rastérisation 3D, l'autre sur un ensemble de photos de la scène par Z-buffer matériel.

– Différents niveaux de détails (LOD ou "level-of-detail") peuvent facilement être obtenus pour chaque objet, simplement en changeant la résolution de la grille utilisée pour la surfelisation. Ensuite, à chaque position de la caméra, le niveau de représentation le plus intéressant peut être chargé en mémoire.

– Une autre possibilité intéressante est que, pour une scène donnée, les objets définis par surfels peuvent être mélangés avec des objets définis par d'autres modèles géométriques (triangles, splines ou autres) tout en bénéficiant d'un rendu par le hardware. Ainsi, en définissant une scène par un classique mécanisme de graphe de scène [SC92], les grilles de surfels peuvent simplement être intégrées comme un nouveau type de nœud géométrique. Ainsi, on peut définir un mécanisme de LOD pour lequel on bascule d'une représentation en surfels vers une représentation en triangles ou en splines lorsque qu'une visualisation très détaillée est nécessaire. De ce point de vue, les objets définis par surfels peuvent être considérés comme un nouveau type d'imposteurs.

– Enfin, comme le rendu de chaque surfel est réalisé en temps constant, quelque soit sa position, son orientation ou ses caractéristiques optiques, un taux de rafraîchissement garanti peut être obtenu, simplement en fixant le nombre de surfels à visualiser par image, selon les capacités du matériel utilisé.

La représentation par surfels est en fait une grille 3D où des cellules stockent des surfels. Les données nécessaires pour le rendu d'un surfel sont :

– La position du surfel (donnée implicitement par des coordonnées de la cellule).

– L'orientation du surfel (un vecteur normal quantifié).

– Les propriétés optiques du surfel (une couleur ou un indice dans une table qui stocke de différents matériaux)

La surfelisation (conversion d'un modèle géométrique en sa représentation par surfels) est tout simplement un processus de rastérisation (ou échantillonnage) qui extrait des informations nécessaires et les stocke dans les cellules de la grille.

Le premier algorithme de surfelisation est purement logiciel qui met en œuvre un échantillonnage analytique avec une résolution donnée. Par exemple pour les maillages polygonaux on utilise la technique classique de parcours incrémental des voxels intersectés par le triangle, pour des surfaces implicites ou objets CSG on évalue la fonction in/out pour chaque cellule (à l'aide d'un algorithme de type marching cubes classiques).

Le second tente d'utiliser au maximum le hardware pour accélérer la surfelisation. Son principe est l'utilisation de Z-buffer matériel pour obtenir les informations dont on a besoin. Pour une position de caméra donnée on utilise le hardware et la projection orthogonale pour obtenir l'image de l'objet. Dans la terminologie OpenGL cette image est appelée *color buffer* et elle est basée le plus souvent sur le modèle d'illumination de Gouraud. Une autre possibilité, appelée matériel buffer, consiste à donner à chaque pixel une couleur constante permettant d'identifier le matériau (caractéristiques optiques) de l'objet concerné, matériau qui sera ensuite utilisé pour la phase de rendu. Avec l'algorithme de Z-buffer le hardware fournit aussi le depth buffer qui représente la distance entre la caméra et l'objet pour chaque pixel. On combine ces informations pour obtenir la position et le matériau de chaque point rendu. Ensuite, à chaque sommet du maillage on associe une pseudo-couleur obtenue en mappant le vecteur normal unitaire dans le domaine RGB. Ainsi, en appliquant un deuxième Z-buffer, nous obtenons une image appelée orientation buffer où chaque couleur de pixel représente le vecteur normal du point correspondant.

Ainsi pour chaque point-échantillon de la scène, on peut facilement obtenir ses coordonnées, son orientation et sa couleur ou son indice de matériau ce qui représente l'ensemble d'informations nécessaires au processus de surfelisation.

Bien évidemment cette technique ne génère que les surfels visibles à partir d'une position donnée de caméra. Pour obtenir la surfelisation complète de la scène un mécanisme de prise de vues multiples basées sur la subdivision régulière de la sphère peut être utilisé.

Conclusion :

Parmi les avantages de la représentation à base de points :

- La simplicité du concept (primitive simple).
- Pipeline facile à concevoir (algorithme de rendu simple).
- Complexité très réduite par rapport aux polygones (manipulation de facette vs points) très utile dans la représentation de scènes complexes ainsi que dans le contexte de niveau de détail.

Cette représentation présente aussi des inconvénients parmi lesquels, dont la qualité de l'image qui est moins appréciée.

En conclusion, nous avons décrit brièvement une technique bien populaire dans la communauté graphique et plusieurs autres techniques sont mentionnées dans [KJ03] pour plus de détail.

CHAPITRE III
ETAT DE L'ART

Techniques de Reconstruction de surface

III/ TECHNIQUES DE RECONSTRUCTION DE SURFACE

1. Introduction

Une image de synthèse réaliste est en fait générée à partir d'une scène 3D : une collection organisée et hiérarchisée d'objets 3D, lumières, matériaux, caméras, déformateurs physiques, etc. La scène décrit complètement l'espace tridimensionnel à partir duquel on synthétise l'image. Plus la scène est complexe et précise et plus l'image sera réaliste. En particulier, les objets 3D, qui décrivent les géométries de la scène, sont de plus en plus précis. Malgré la capacité grandissante du matériel graphique qui permet de générer des images de synthèse de plus en plus réalistes, avec un temps de rendu qui ne cesse de diminuer, la résolution des écrans, et donc des images générées, reste relativement stable. Du fait que les surfaces des objets générées sont converties en une collection de facettes contiguës « le plus souvent des triangles » et afin d’être visualisé efficacement, un triangle rendu peut correspondre à moins d'un pixel à l'écran, induisant ainsi de nombreux calculs inutiles.

Depuis quelques années, un nouveau paradigme de modélisation répond à cette problématique, les *surfaces à base de nuages de points*. Ce paradigme définit les objets 3D comme des nuages de points non structurés, et donc sans topologie explicite. Chaque point est doté d'une normale, éventuellement d'un rayon, et d'une liste de paramètres d'apparence (couleur, coordonnées de textures, etc.…). Visualiser de tels objets revient simplement à projeter le nuage sur le plan de l'écran. Pour pallier aux trous subsistant dans les zones insuffisamment denses, on dessine une « tâche » à l'endroit de la projection du point (splatting). Dès lors, le paradigme du point s'est imposé comme une bonne solution, réduisant l'espace mémoire des modèles, et négligeant la composante topologique explicite présente dans les surfaces polygonales. Cette absence de topologie explicite permet d'établir très facilement une structure de multi résolution sur un nuage de points (à l'aide d'un arbre BSP par exemple). Le fait d'avoir des objets sous formes de nuages de points non organisés permet également de

faire du streaming de points pour les applications en ligne, fournissant une visualisation progressive, un peu à la manière des images présentes sur le Web (format Jpeg progressif).

Cette technique basée image n'est pas toujours suffisante en terme de visualisation (qualité du rendu), et il est souvent nécessaire de reconstruire la surface dans l'espace de la scène, l'espace objet par exemple : pour gérer les phénomènes complexes d’illumination, ou bien pour mettre en œuvres des animations: on parle dès lors de *reconstruction de surface* [TB04].

Depuis une dizaine d’années, les progrès des technologies d’acquisition 3D ont favorisé l’essor de la modélisation géométrique d’objets réels à partir de données issues de leur numérisation, ce qui a permis d’envisager une nouvelle alternative par rapport à la modélisation classique. La capacité à créer, visualiser et manipuler des représentations numériques de la forme et de l’apparence d’objets physiques, joue un rôle important dans de multiples domaines d’application, comme le design industriel, l’imagerie médicale, ou les systèmes d’information géographique. Un autre domaine d’application important de la numérisation 3D concerne l’étude et la conservation du patrimoine historique et culturel, comprenant les collections des musées, les objets et les sites archéologiques [RA06].

Plusieurs projets de numérisation d’œuvres d’art de grande ampleur ont été menés ces dernières années, dont le populaire *Digital Michel Angelo Project* [LP00]. Ces projets ont contribué à la mise en place de chaînes de traitement permettant de passer des données comportant des millions, voire des milliards de points à des maillages manipulables sur des machines standard.

Dans ce mémoire, nous avons axé notre étude sur le rapprochement entre ces deux modèles de représentation géométrique afin d’obtenir une paramétrisation automatique pour le passage d’une représentation à une autre. L’une des motivations de cette étude est de bénéficier pleinement des avantages des deux représentations. Lorsqu’un niveau de détail est exigé il est bien évident que la représentation la plus

adéquate est celle des maillages polygonaux qui offre une meilleure qualité de rendu et bénéficie largement des capacités du pipeline graphique. Par contre, lorsque le niveau de détail est moins exigé, le temps consacré à la simplification des maillages est trop élevé et souvent inutile. Lorsque la scène est très complexe, un triangle peut se réduire à un simple pixel, et donc la solution convenable est d'utiliser le paradigme de point.

La modélisation à base de points présente de nombreux avantages : sélection aisée, travail en multi résolution, ou bien encore texturations des modèles. La place des surfaces de points aujourd'hui est claire : un excellent paradigme de modélisation, qui unifie les autres, grâce notamment aux techniques de reconstructions de surfaces à partir de nuages de points, permettant de passer d'une représentation par points (utilisée pour modéliser l'objet) à une autre (par exemple pour utiliser l'objet en visualisation ou en simulation). Dans ce qui suit, nous aborderons le sujet en faisant un bref explicatif sur l'acquisition des données d'un objet physique puis un tour d'horizon sur les techniques de reconstruction de surfaces et enfin nous abordons notre contribution en présentant un modèle qui permet de passer d'une représentation à une autre. Dans notre cas, nous étudions le sujet dans le sens ou à partir d'un nuage de point quels sont les éléments clé à prendre en compte pour passer à une représentation surfacique sur la base de ces points et vis versa.

2. Acquisition

D'une façon générale la figure 10 décrit de manière synthétique le processus standard de création d'un modèle d'un objet physique à l'aide d'un dispositif de type scanner laser. Ce processus comporte trois étapes fondamentales. La première étape d'acquisition consiste à effectuer des mesures de la géométrie de la surface de l'objet sous différents angles pour en tirer des données en trois dimensions, le plus souvent sous la forme d'une collection d'images de profondeur correspondant chacune à un point de vue d'acquisition. Chaque image de profondeur consiste en une grille régulière de pixels auxquels sont associées des coordonnées 3D calculées en fonction

de la distance qui sépare le capteur de la surface. L'étape de recalage consiste à aligner les images de profondeur dans un même repère pour ensuite les fusionner, de manière à obtenir un nuage de points répartis sur toute la surface numérisée de l'objet. La dernière étape fondamentale est celle qui consiste à reconstituer la surface de l'objet numérisé à partir de l'échantillon de points [RA06].

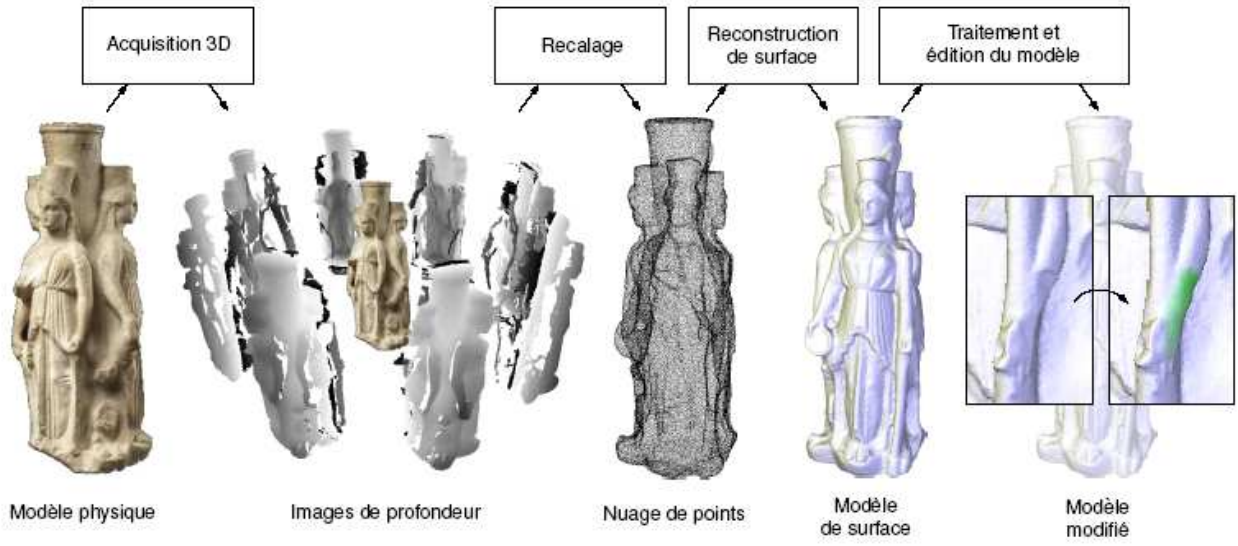


Figure 12. Processus d'acquisition [RA06]

3. Reconstruction de surface

Étant donné un ensemble de points P $\in \mathbb{R}^3$ échantillonnés sur une surface S, l'objectif de la reconstruction de surface est de calculer un modèle continu de S à partir de P.

Ce modèle est appelé reconstruction de S à partir de P.

La reconstruction de surface intervient dans un grand nombre d'applications et fait l'objet de travaux de recherche depuis plus de vingt ans dans les communautés de l'informatique graphique, de la modélisation géométrique et de la géométrie algorithmique. Ce domaine a connu un essor important ces dernières années, avec le développement des technologies d'acquisition 3D.

3.1-Classification

Les techniques de reconstruction de surface peuvent être classifiées selon deux approches principales [RA06] : *l'approche combinatoire et l'approche par ajustement d'un modèle prédéfini.* Un grand nombre de méthodes combinatoires ont pour principe d'établir des relations d'adjacence entre les points d'un échantillon. Elles ont pour la plupart été développées dans la communauté de la géométrie algorithmique, qui a apporté des contributions théoriques importantes au problème de la reconstruction de surface, notamment avec les notions d'échantillon et de triangulation de Delaunay restreinte à une surface.

La seconde approche repose sur l'idée d'approximer la surface échantillonnée à l'aide de modèles prédéfinis, sur la base d'hypothèses globales ou locales concernant la forme à reconstruire. Depuis 2001, on assiste à un fort engouement de la communauté graphique pour des approches implicites locales.

a- Approche combinatoire

Les méthodes combinatoires ont pour but d'établir des relations d'adjacence (ou de connectivité) entre les points voisins sur la surface échantillonnée. Pour atteindre cet objectif en l'absence de connaissances sur l'organisation des données, l'information topologique peut-être déduite de relations de proximité entre les points sous certaines hypothèses concernant la densité de l'échantillon.

Les méthodes combinatoires exploitent ainsi des structures géométriques permettant de mesurer la proximité des points dans l'espace. Parmi ces structures, le diagramme de Voronoï et son graphe dual, la triangulation de Delaunay, sont particulièrement populaires en géométrie algorithmique. Elles présentent en effet des propriétés intéressantes pour résoudre le problème de la reconstruction. Les relations de connectivité établies par la triangulation de Delaunay sont globales dans le sens où elles impliquent tous les points de l'échantillon. D'autres algorithmes combinatoires adoptent une approche plus locale en considérant la relation du plus proche voisin au sens de la distance euclidienne. Ces algorithmes ont recours à des structures de

localisation spatiale adaptées pour obtenir de manière efficace les points les plus proches d’un point donné.

b- Approche par un modèle d’ajustement prédéfini :

Les méthodes de reconstruction de surface par ajustement de modèles cherchent à contraindre un modèle mathématique de surface global ou local déterminé a priori pour minimiser l’écart entre le modèle et les données. Le processus de reconstruction se ramène alors principalement à un problème d’optimisation.

La surface peut être contrainte à passer par les points de données, on parle d’interpolation, ou bien à proximité, on parle alors d’approximation.

Le modèle peut être défini comme une fonction dépendant d’un nombre fini de paramètres. L’objectif est alors de trouver les valeurs des paramètres pour que le modèle interpole ou approxime les données en entrée, selon une approche globale ou locale ; on parlera dans ce cas d’ajustement de paramètres. Plus généralement, un modèle peut être représenté par une fonction de manière paramétrique ou implicite.

Une Autre classification que nous présentant ici est celle de [TB04], celle-ci classe les techniques de reconstruction en deux familles : *Reconstruction implicite ou explicite.*

c- Reconstruction implicite :

Il existe une première approche dans ce cadre qui consiste à utiliser des fonctions à base radiale (RBF) pour interpoler un nuage de points [TO02]. Initialement limitée à des nuages de points de petite taille, cette approche a par la suite été étendue aux nuages de points de grande taille via le principe de la partition de l'unité [TRS04], technique qui a également été utilisée sur d'autres types de surfaces implicites.

Une seconde approche possible est d'utiliser les surfaces MLS [ABCO+01] sur un nuage de points, à l'aide d'un opérateur de projection. Le point fort des méthodes de

reconstruction implicite est d'être très robuste lorsque le nuage de points n'est pas uniforme. Leur point faible, est que la surface résultante n'est pas directement visualisable par le pipeline de rendu matériel et nécessite une étape supplémentaire d'extraction de maillage de la surface implicite de type Marching Cubes [Blo88, LC87].

d- Reconstruction Explicite :

Là encore, il existe globalement deux approches pour générer une reconstruction polygonale du nuage de points. D'une part une approche dynamique, dérivée des contours actifs [BI98, KWT87] issues de la 2D, où l'on essaie de minimiser une énergie [Alg95] définie par le nuage de points, en tout point d'une surface graine, et ce en déformant cette surface ; on peut citer notamment les Intelligent Balloons [DQ01] pour leur flexibilité topologique. Ces techniques sont très lentes, et peu adaptées aux grands nuages de points. D'autre part, une approche statique, basée sur une triangulation, habituellement celle de Delaunay [BC00, CSD02] et son dual topologique, le diagramme de Voronoï. La technique du *Crust* d'amenta et al. [ACK01] s'appuie sur ce diagramme, et extrait une surface à partir d'une analyse de l'axe médian ; cet algorithme fut optimisé par le Cocoone de Dey et al. [DGH01]. Pour gérer des nuages de points plus importants, des approches par projections locales furent proposées par Gopi et al. [GKS00, GK00]. Outre l'ajout de qualité à la surface en post-traitement, la subdivision de surface [ZS00] est en général utilisée en reconstruction après la génération d'un maillage de faible définition à partir du nuage de points. L'application de passes de subdivision successives, en minimisant l'erreur avec le nuage de points à chaque étape converge vers une surface approximant le nuage de points.

Il est évident qu’un consensus sur les critères de classification des techniques de reconstruction de surface issue de points n’est pas encore atteint. Or, ces deux familles présentées ici ont davantage des points en communs :

- Le partitionnement du nuage en générale par octree pour un traitement local des points.
- La triangulation de Delaunay et son dual Voronoï pour établir des relations d'adjacence entre les points.
- L'utilisation d'une fonction implicite pour approximer ou interpoler les points à la surface.
- L'utilisation d'un modèle de surface mathématique pour minimiser l'écart entre le modèle et les données.
- Le recouvrement dans le cas d'un traitement local.
- Le résultat obtenu est soit un ensemble de facette triangulaire qui représente l'objet ou l'objet lui-même est généré sous forme de surface implicite après traitement bien sûr.
- Subdivision de maillage (lissage).
- Extraction de maillage pour le rendu.

D'une manière générale, la majorité des techniques s'inspirent de ces critères pour la reconstruction de surface issue de points, mais ce qui est remarquable, c'est qu'il y'a des techniques qui utilise tout le nuage à la fois et d'autre utilise un traitement local sur chacune des partitions du nuage. Cette remarque à fait l'objet d'une critique de M. Alexa dans [MG07, HP07] pour distinguer entre *la représentation de surface et reconstruction de surface*.

Pour bien éclaircir le sujet, nous proposons par la suite un exemple de technique de reconstruction qui s'attache à une grande partie des critères mentionnés auparavant.

3.2-Exemple de technique de reconstruction de surface

L’approche de reconstruction de surface par surface de subdivision [TB04] prend en entrée un nuage de points non structurés, où les points sont équipés d'une normale, et fournit en sortie des surfaces visuellement continue.

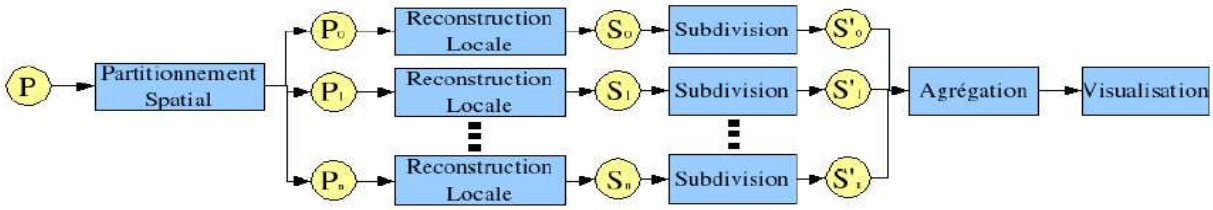


Figure 13. Principe général de l’algorithme de reconstruction [TB04].

Le principe général est le suivant :

a. Partitionnement en octree du nuage de points.

b. Reconstruction locale de chaque partition, en l'augmentant momentanément des points de son voisinage. L'objectif étant d'éliminer les trous en créant des recouvrements entre surfaces voisines.

c. Application du principe de subdivision à l'agrégat de surfaces obtenu, afin d'augmenter la qualité visuelle de l'ensemble.

Cet algorithme doit avoir un certain nombre de propriétés :

- Prendre en entrée un petit nuage de points non organisés P_i et fournir en sortie un morceau de surface S_i ;
- La surface reconstruite S_i doit interpoler, ou à défaut fournir une bonne approximation de P_i ;
- S_i sera soumis à la subdivision, et doit donc le plus possible se rapprocher des critères de qualité de subdivision (valence des sommets).

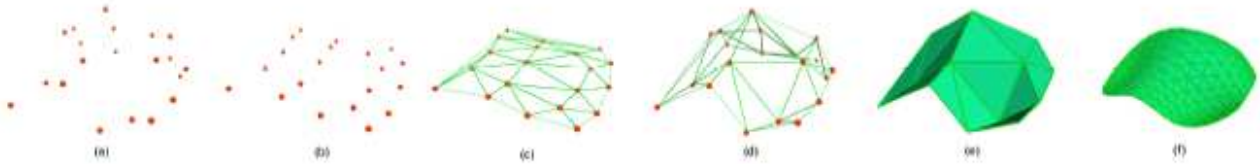


Figure 14. Reconstruction surfacique locale. (a) Partition initiale P_i (b) Projection sur le plan minimisant l'énergie pour P_i. (c) Reconstruction en 2D par triangulation de Delaunay. (d) Reprojection 3D de l'ensemble des triangles obtenus. (e) Surface reconstruite S_i avant subdivision. (f) Surface lisse obtenue par subdivision de Si [TB04].

Le traitement approprié pour une partition est de la considéré comme étant une carte d'élévation, ce qui veut dire qu'elle doit vérifier le prédicat suivant :

$$\forall k \in [0, N_i[,\ \overrightarrow{n_{i,k}}.\overrightarrow{n_i} > \delta_a,\ \delta_a \in [0,1] \text{ et } ||\overrightarrow{p_{i,k}, c_i}|| < \delta_d,\ \delta_d \in [0, s_i] \Rightarrow \kappa(P_i) \text{ est vrai}$$

Le paramètre δ_a est lié à l'angle maximal que peut former la normale en un point avec la normale moyenne d'une surface d'élévation. Les tests montrent que δ_a = 0.1 donne de bons résultats en général. Le paramètre δ_d représente un écart-type maximal autorisé pour la distribution des points autour de P_i, on choisit $\delta_d = \frac{S_i}{4}$ ce qui donne de bons résultats (en l'absence de critère sur la densité). Ce paramètre est assez intuitif pour être modifiée interactivement et limite les dégénérescences dues au partitionnement, et quand la partition vérifie ce prédicat alors l'algorithme pour chaque cellule C_i est le suivant :

1. Calculer le plan tangent moyen π_i (qui va minimiser l'énergie dans C_i) ayant pour normale n_i et passant par c_i ;
2. Projeter *Pi* sur le plan π_i ;
3. Calculer la triangulation de Delaunay en 2D sur les points projetés sur π_i ;
4. Reprojeter les triangles obtenus dans l'espace 3D par transformation inverse de l'étape 2.

La figure 14 montre les différentes phases de cet algorithme. Cette solution interpole P_i dans chaque cellule C_i, et fournit une bonne distribution des triangles. On génère beaucoup moins de triangles qu'une triangulation 3D, et les calculs à effectuer

sont plus simples (géométrie du plan). Cette triangulation offre l'avantage de maximiser le plus petit angle formé par 2 arêtes. La valence aux sommets extraordinaires est donc faible, et l'application de plusieurs passes de subdivision ne verra pas se développer de singularités trop fortes en ces points. Bien entendu, l'étape de reprojection 3D occasionnera une déformation des triangles, mais les tests montrent que le maillage reste correct pour les densités courantes.

Afin d'assurer une visualisation correcte -sans trou entre morceaux de surface- on propose d'effectuer un recouvrement entre surfaces voisines, en appliquant l'opérateur de reconstruction locale sur des partitions momentanément élargies à leur voisinage (figure 15). Cette étape s'apparente à une restitution des caractéristiques globales de la topologie. On élargit le support de reconstruction, sans contrainte particulière, chaque cellule est élargie d'un facteur β, afin d'englober les points voisins. Cette technique provoque un recouvrement entre morceaux de surface, et élimine ainsi les trous. Déterminer β peut se faire interactivement, ou bien en tenant compte de la densité du nuage de points. On parle de -densité d'un nuage de points si toute boule centrée en un point du nuage, de rayon γ ne contient aucun autre point, et que γ est maximal. Dans ces conditions, il suffit d'élargir le support de $\beta = \frac{2\gamma}{S_i} + 1$ pour garantir un recouvrement suffisant.

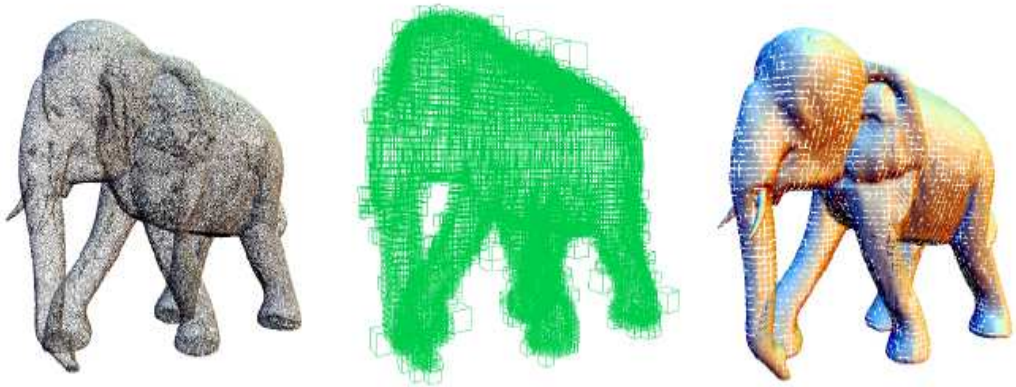


Figure 15. A gauche, le nuage de points P. Au centre, on visualise les cellules feuilles de l'octree partitionnant le nuage. A droite, un morceau de surface a été reconstruit indépendamment pour chaque partition [TB04].

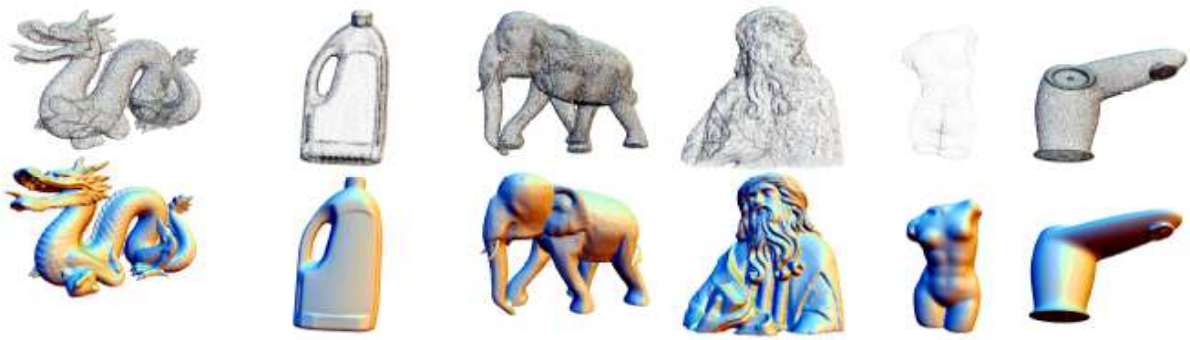


Figure 16. Reconstruction et visualisation de divers nuages de points [TB04].

La figure 16 montre le résultat de la reconstruction lors du rendu. Une bonne qualité sans oublier le temps d’exécution consacré à cet algorithme. Mais ce qui est remarquable c’est qu’aucune alternative n’est mentionné en cas ou le prédicat n’est pas vérifié, autre chose beaucoup de paramètre ont été introduit et prennent des valeurs estimées ou empiriques. Dans ce cas là, l’algorithme ne propose pas un modèle qui peut être modifié ou mis à jour pour une approche multi résolution cependant il reste un modèle figée.

4. Synthèse

1- Le nuage de points tel que fourni par le dispositif d’acquisition peut contenir des bruits, en général il est gaussien et donc nécessite un procédé de filtrage.

2- Mis à part des informations colorimétriques fournies avec le nuage de points, aucune information de connectivité entre les points et la normale n’est estimée que par un calcul approximatif (dans la majorité des cas l’ACP).

3- Divers technique de reconstruction de surface procèdent dans un cadre local ou global c'est-à-dire de façon à traiter tout le nuage à la fois ou bien localement dans un contexte réduit.

4- Ces techniques font recours parfois à des critères heuristiques ou des estimations d’erreurs expérimentales pour des situations de convergence.

5- Aucune méthode unique pour le test de comparaison entre le nuage de points et la reconstruction finale. Dans chaque technique, on procède en fonction de la nature des techniques utilisées (mathématique ou algorithmique).

6- Dans certaines techniques, il est très difficile de procéder à une mise à jour voir impossible de modifier le résultat, techniques qui délivrent un modèle figé.

7- L’aspect multi-résolution est moins traité dans plusieurs techniques ou parfois non détaillé.

8- Le passage interactif d’une représentation à une autre n’est pas inclus d’une façon implicite.

9- Il n’existe pas de techniques de reconstruction fiables, selon le domaine d’utilisation.

10- l’implémentation directe des techniques de reconstruction dans le pipeline graphique est très délicate.

5. Contributions

Le domaine de représentation de surface issue de points a largement focalisé l’intension de la communauté graphique durant ces trois dernières décennies, plusieurs techniques en découlent mais il n’existe aucune qui prend l’avantage d’être meilleure qu’une autre. Cela dépend de plusieurs facteurs dont le domaine d’application (médecine, archéologie, etc.…) est considéré comme un des critères pour lequel une technique est jugée intéressante par rapport à une autre.

En outre, un des critères les plus sollicités est que le résultat d’une technique puisse être exploité directement par le pipeline graphique, cela réduit énormément le temps du rendu et par conséquent, le mieux est de produire des surfaces polygonales afin de bénéficier d’un tel avantage.

Ce que nous proposons ici est de mener une étude sur ce que permet le rapprochement entre deux domaines de représentation. Pour atteindre ce but, nous avons développé une technique de reconstruction de surface à partir d’un nuage de points, ce qui nous permet d’avantage d’explorer profondément le sujet et voir de près les avantages d’un tel rapprochement. L’idée de base est simple, inspirée des techniques de contours et plans d’ajustement, on cherche à mettre en évidence les paramètres qui nous permettent le passage d’une représentation à une autre et afin d’améliorer la qualité du rendu nous tentons par la suite d’appliquer les outils implémentés dans le pipeline graphique tel que l’illumination locale et les textures.

Dans le chapitre suivant nous décrirons comment nous avons abordé le sujet et la stratégie nécessaire pour mettre en évidence cette conception.

CHAPITRE IV

Conception

Technique de reconstruction de surface

Points Vs Polygones

IV/ CONCEPTION D'UNE TECHNIQUE DE RECONSTRUCTION PvsP

1. Introduction

La reconstruction de surface issue de nuage de points est un domaine en cours de progression. Depuis plus de vingt ans, beaucoup de travaux de recherche ont été élaboré par les scientifiques de la communauté informatique graphique et cela dans l'objectif de répondre au besoin incitant de l'apport des scanners 3D d'acquisition de nuage de points d'un objet physique. Parmi les travaux de reconstruction de surface issue de points le MLS de [ABCO+01].

Malgré la diversité des techniques de reconstruction, plusieurs problèmes ont été constatés suite à une étude comparative de [RA06] évoquant les insuffisances dont souffrent ces techniques, parmi, des techniques qui ne sont pas robuste à un nuage insuffisamment échantillonné, à la densité des points dans le nuage (nuage éparse) et à la convergence de la technique appliqué sur différents modèles d'objets. D'autant plus que l'exploitation du pipeline graphique pour le rendu.

Notre objectif, est de fournir une technique qui dans le besoin d'un rapprochement entre deux domaines de représentation (à base de points et polygonales) qui exploite pleinement le pipeline graphique et offre des possibilités telle que le passage d'une représentation à une autre.

2. Reconstruction de surface par des Plans d'ajustement

2.1 Introduction

La reconstruction de surface à partir d'un nuage de point n'est pas triviale. Un nuage de points d'un objet physique échantillonné est naturellement une perte d'information en plus la redondance de point présent dans ce nuage fait que le résultat de la reconstruction n'est pas toujours cohérent.

L'idée est très simple, le nuage de points n'est pas totalement exploitable pour la reconstruction, seulement un sous ensemble de points qui ont une propriété qu'on détaillera par la suite qui vont être utilisé, donc le nuage de points sera simplifiés. Avant de simplifier le nuage, le processus doit passer par une phase préliminaire qui représente l'étape clé de la reconstruction et qui est celle du calcul des plans d'ajustement des points selon les trois axes X, Y, Z. Chaque point sera donc transformé et projeté sur trois plans calculés par regroupement de points selon une estimation. Les coordonnées d'un point sont donc transformées et ont des valeurs selon un plan X_i, un plan Y_j et un plan Z_k. Pour simplifié le nuage de point à partir des nouvelles valeurs des coordonnées de points il suffit de cherché les combinaisons possibles des 03 plans calculés, on forme ainsi le sous ensemble des points qui correspondent aux combinaisons de trois plans sélectionnés X_i', Y_j' et Z_k' et cela nous permettra donc de choisir parmi les points celui qui possède la plus grande distance par rapport à l'origine du repère. Selon l'axe choisit au départ, on détermine de cette manière l'ensemble des points (d'un plan) candidats à la reconstruction. Un plan voisin sera donc déterminé de la même manière et la recherche de polygones de quatre sommets (deux points voisins dans un plan et leurs correspondant dans l'autre) peut s'effectué.

Donc, le processus de reconstruction peut être résumé ainsi :

1-Calcul des plans selon les 03 axes X, Y et Z.

2-Simplification du nuage.

3-Formation des plans voisins selon un axe.

4-Recherche des polygones par balayage en partant par exemple du bas vers le haut.

La figure suivante illustre l'enchaînement que subit le nuage pour former des polygones à partir d'un nuage de points en entrés:

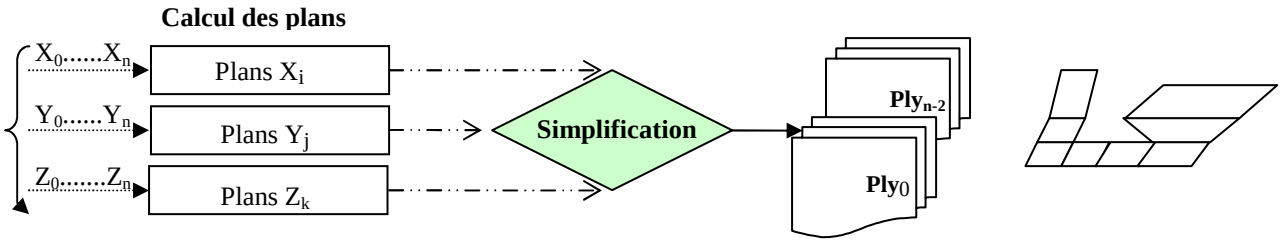


Figure 17. Pipe line de reconstruction de surface.

2.2 Calcul des Plans

La première étape de la reconstruction consiste à prendre les valeurs des coordonnées des points du nuage séparément selon chaque axe ensuite les triés par ordre croissant puis le calcul des plans X_i Y_j et Z_k s'effectuera dans chaque axe de la manière suivante:

a- Calculer la moyenne des valeurs, initialement les n valeurs

b- Centrer ces valeurs par rapport à la moyenne

c- Comparer la différence à une estimation donnée ξ.

d- Associer au sous-ensemble de valeurs ayant vérifié le test de comparaison au plan χ_i.

La figure suivante explique pour les 03 axes comment s'effectue le calcul des plans X_i, Y_j et Z_k.

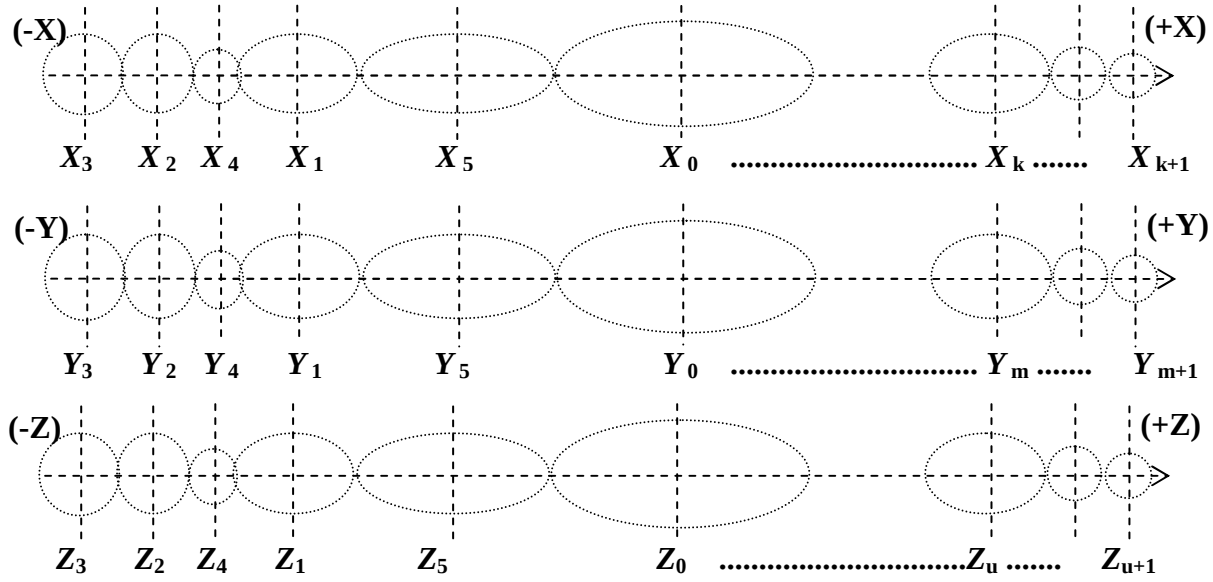


Figure 18. Calcul des plans d'ajustements des points.

a- Choix de l'estimation

Au départ nous avons opté pour un choix de l'estimation d'une façon empirique, ce choix est justifié par le besoin d'imposer l'écart entre les valeurs des points (selon un axe) et aussi pour tester la convergence de cet algorithme. Dans ce cas, le choix est dépendant de la mise en échelle des valeurs des points et pour une estimation très petite on aura un grand nombre de plan à calculer et en revanche si la valeur est grande le nombre de plan à calculer sera très réduit.

Les figures suivantes montrent la différence entre un choix de valeur de l'estimation (petit et grand).

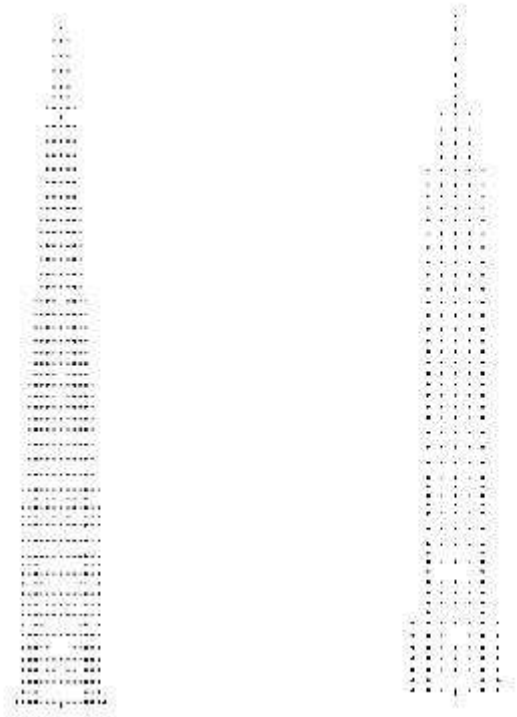


Figure 19. A gauche estimation petite- A droite estimation grande

La convergence de l'algorithme à partir d'un choix empirique de l'estimation n'est pas assuré lorsque celle ci est appliqué aux 03 axes c'est à dire qu'on utilise la même valeur pour le calcul des plans selon les 03 axe X, Y et Z. Pour remédier à ce problème, nous avons choisi de calculer pour chacun des axes l'estimation convenable issues des valeurs des points et donc on aura une estimation pour l'axe X une pour Y et une pour l'axe Z.

b- Calcul de l'estimation

La méthode de calcul de l'estimation selon un axe consiste à trié par ordre croissant les valeurs des points (ex : abscisses) et chercher ensuite le petit écart entre une valeur et les autres et prendre la plus grande valeur *min*, en d'autre terme le ***max*** des ***min*** (**Mdm**). On peut schématiser cela de la manière suivante :

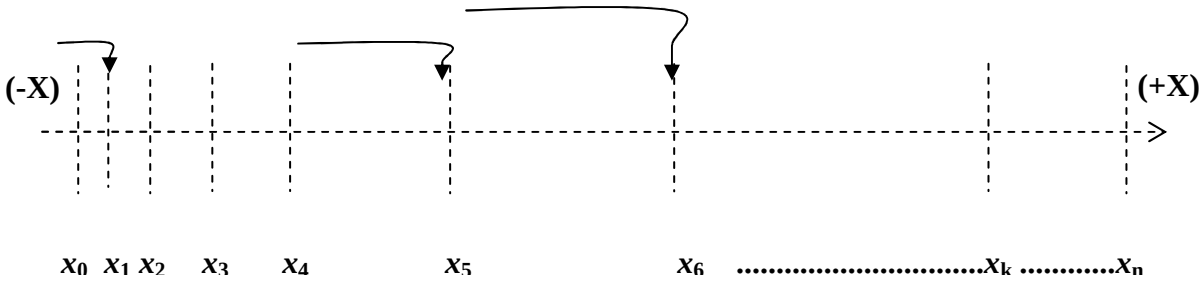


Figure 20. Choix de l'estimation à partir de l'écart de valeur.

Les x_i sont triés, donc l'écart E_i entre un x_i et x_{i+1} (le voisin de x_i) est l'écart minimum (dans le sens croissant) entre x_i et les autres, et pour chaque x_k on cherche de la même façon E_k, enfin de compte on prendra le maximum des E_k.

Soit :

$E_0 = \min\left(|X_0 - X_1|, |X_0 - X_2|, \dots, |X_0 - X_n|\right)$ et comme les X_i sont triés,

donc $E_0 = |X_0 - X_1|$.

De même pour :

$E_1 = \min\left(|X_1 - X_2|, |X_1 - X_3|, \dots, |X_1 - X_n|\right)$ et donc $E_1 = |X_1 - X_2|$.

.

.

et enfin $E_{n-1} = |X_{n-1} - X_n|$.

L'estimation selon l'axe des X sera donc $\xi_x = Max\ (E_0, E_1, ..., E_{n-1})$.

De la même manière on déterminera ξ_y et ξ_z.

- Calculer la moyenne des valeurs, initialement les n valeurs

  Prendre un intervalle de valeur initialement $[x_0, x_1, \ldots, x_n]$

$$moy_i = \sum_1^n xi / n$$

- Centrer ces valeurs par rapport à la moyenne

$$|x_0 - moy_i|, |x_1 - moy_i|, \ldots, |x_n - moy_i|$$

- Comparer la différence à une estimation donnée ξ.

$$|x_0 - moy_i| \leq \varepsilon_x, |x_1 - moy_i| \leq \varepsilon_x, \ldots, |x_n - moy_i| \leq \varepsilon_x$$

- Associer au sous-ensemble de valeurs ayant vérifié le test de comparaison au plan χ_i.

$$\{(x_{k-j}, x_{k-j+1}, x_{k-j+2}, \ldots, x_{k-j+l})\}$$

- Répétez ce même processus sur un autre intervalle soit vers la gauche ou vers la droite.

On peut donc résumé ceci par un algorithme illustrant les différentes étapes du processus de calcul des plans et la condition d'arrêt sera donc quand toutes les valeurs d'un axe auront leur affectation à un plan selon la moyenne et la comparaison à l'estimation.

Algorithme ***calculPlanAjust****(min ,max) ;*

Début

*Moy=****calculMoyenne****(min ,max) ;*

centrerValeur*(min, max ,moy) ;*

compareEstimAsso*(min, max, estim, Plan) ;*

Fin ;

Le résultat fourni par cet algorithme sera donc 03 ensembles de partition de Plan selon les axes X, Y et Z.

P/x=$\{Px_1, Px_2, \ldots, Px_k\}$;

P/y=$\{Py_1, Py_2, \ldots, Py_j\}$;

P/z=$\{Pz_1, Pz_2, \ldots, Pz_l\}$;

Avec k$\neq 0$; j$\neq 0$; l$\neq 0$

Convergence :

La convergence de cet algorithme est assurée de la façon suivante :

Rappelons le calcul de la moyenne des valeurs$x_i \in [x_1 .. x_n]$ est $\bar{x} = 1/n \sum_{1}^{n} x_i$; le centrage consistait à calculer l'écart $|x_i - \bar{x}|$ et la comparaison de cet écart à l'estimation doit donc aboutir à l'existence d'au moins un x_k qui satisfait l'inéquation.

Deux cas peuvent se présenter en fonction du calcul de la moyenne :

$$\begin{cases} \bar{x} = x_j \ \square\!\!\square \ x_j \in [x_1, \ldots, x_n] \\ x_j \leq \bar{x} \leq x_{j+1} \end{cases} \quad ou$$

Pour le premier cas si $\bar{x} = x_j$ alors $|x_j - \bar{x}| \leq \varepsilon_x \Rightarrow |x_j - x_j| \leq \varepsilon_x$ donc au moins un élément qui satisfait l'inéquation.

Le second cas, $|x_k - \bar{x}| \leq \varepsilon_x$ puisque $|x_k - x_{k+1}| \leq Max(|x_i - x_{i+1}|) = \varepsilon_x$ par définition au départ du calcul de l'estimation, donc $|x_k - \bar{x}| \leq \varepsilon_x$ est vrai sans oublier que les x_i sont ordonnés d'une façon croissante. Enfin de compte il existe au moins un élément qui satisfait l'inéquation.

Finalement, tout point du nuage possèdera en plus de ses coordonnés son appartenance à une classe unique par rapport à la partition des plans selon les 03 axes.

c- Alternative de calcul des plans

Une deuxième alternative que nous proposons pour le calcul des plans d'ajustement sans se soucier de la convergence de l'algorithme est de faire une subdivision adaptative de l'axe choisit en fonction toujours de l'estimation calculée.
Partant de l'abscisse le plus petit jusqu'au plus grand, la partition selon un axe devra satisfaire la condition suivante :

Un x_i appartient à une partition P_k s'il est inférieur à un accroissement δ dans l'ordre k.

Cet accroissement est mis à jour lorsqu'on atteint un x_j qui dépasse δ , a ce moment là on change δ par la valeur de x_j sommé avec l'estimation approprié de l'axe en cours de traitement et on incrémente aussi l'indice de la nouvelle partition.

$$x_i \in P_k \; si \; x_i \leq \delta j \quad \text{Et k=} \overline{1..m}$$

$x_i \geq x_j$

$$\text{Avec } \delta j = x_j + \xi_x$$

Donc on partitionne un axe avec un accroissement δ par rapport à une valeur de x jusqu'à avoir parcouru tout les écarts de l'axe en question. L'algorithme suivant illustre cette approche :

Algorithme PartitionneAxe ;
Début
Delta :=Axe[1].valeur + estim ; P_k :=1 ; Axe[i].$Part_k$:=P_k ;
Pour i :=2 **à** n **faire**
Si Axe[i].valeur $\leq$ delta **alors** Axe[i].$Part_k$:=P_k
Sinon
début
Delta :=Axe[i].valeur + estim ;
P_k := P_k +1 ;
Axe[i].$Part_k$:=P_k ;
fin ;
Fin ;

Cette approche est appliquée sur les 03 axes en utilisant les mêmes estimations calculées auparavant. Par rapport à la première technique, on n'a pas à se préoccupé de sa convergence car la seule contrainte à satisfaire est le parcours de tout les écarts selon l'axe à traité.

2.3 Simplification

La simplification du nuage de point consiste à chercher les combinaisons possibles des trois ensembles de plans Px, Py et Pz et chercher le point le plus significatif de cette combinaison. Il est évident qu'un point est représenté dans une base linéaire par 03 composantes selon les vecteurs unitaires ($\vec{i}$, $\vec{j}$, $\vec{k}$) qui forme une combinaison linéaire et une base canonique dans l'espace 3D orthonormé. Or, ici ce que nous contentons de chercher est l'ensemble des points qui appartiennent à la fois à un plan X fixe et Y fixe et Z fixe pour illustré cela on utilisera une structure arborescente de la façon suivante :

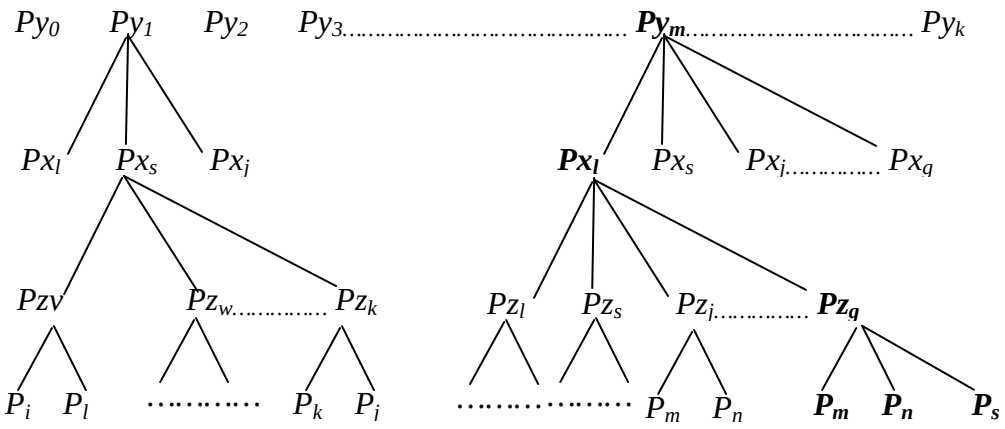


Figure 21. Partitionnement des points par rapport aux plans calculés.

Les nœuds représentent les plans et les feuilles les points associés au plan fixe X, plan Y, plan Z.

Donc le triplet par exemple (Py_m, Px_l, Pz_g) regroupe les points {P_m, P_n, P_s}.

Parmi ces points, on choisit celui qui sera candidat pour la reconstruction tout en fixant comme critère celui qui a le plus grand module ou en d'autre terme la plus grande distance (au sens de la distance euclidienne) par rapport à l'origine du repère (espace objet) :

$d_i = \mathbf{Max} \|\overrightarrow{OPi}\|$. *par rapport à la combinaisons Pxi Pyj et Pzk*

Le résultat de l'étape précédente est exploité de façon combinatoire selon les 03 axes et en fonction des plans d'ajustement calculé :

$P_x = \{px_1, px_2, \ldots, px_k\}$;

$P_y = \{py_1, py_2, \ldots, py_j\}$;

$P_z = \{pz_1, pz_2, \ldots, pz_l\}$;

Chaque point du nuage dispose des éléments qui notifient son appartenance à la fois à un plan Px_i un plan Py_j et un Plan Pz_k et cette appartenance est unique.

P_i : **(C_x, C_y, C_z)**, (Pl_x, Pl_y, Pl_z), (CT_x, CT_y, CT_z).

Donc le point P_i possède en plus de ses coordonnées de base son appartenance aux plans et ses coordonnées transformées.

Algorithme **Simplif_Point**(*PlanX*) ;
Début
Chercher_EnsembleY (*PlanX*, *PlanY*) ;
répéter
chercher_ensembleZ (*PlanX*, *PlanY*, *PlanZ*) ;
selection_Points_G_distance (*PlanX*, *PlanY*, *PlanZ***)**
jusqu'à TousPlanTraitéY ;
Fin ;

Bien évidement ce n'est pas le seul critère qui peut être appliqué pour la simplification, on peut aussi traiter le sujet d'une autre manière, par exemple calculer

un barycentre de ces points et sélectionné celui le plus loin ou le plus proche, ou encore de tester sur la profondeur (valeur de z). Donc, la simplification ressemble un peu à l'agrégation sans pour autant d'appliquer des opérations pour résumé l'information. Un seul point choisit parmi d'autre simplifie beaucoup le traitement dans la phase.

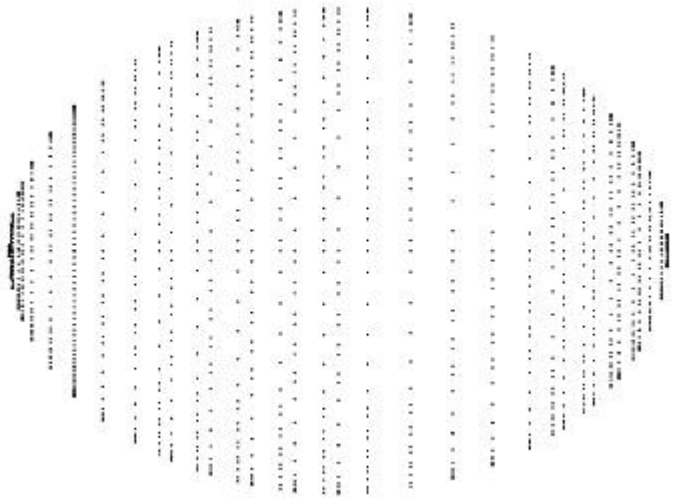


Figure 22. Nuage de Points : avant simplification

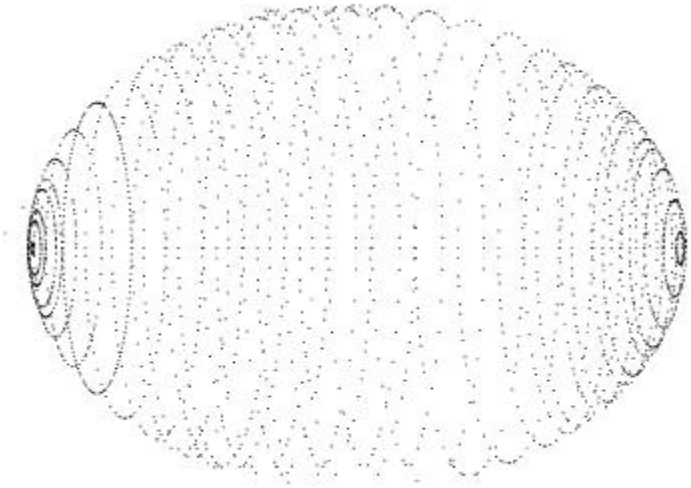


Figure 23. Nuage de Points : après simplification

2.4 Reconstruction à partir des plans formés

Le résultat de l'étape précédente est exploité afin de pouvoir reconstruire les surfaces issues de point. Pour cela, il existe différentes manières de procéder et dépendent de la façon et le sens qu'on veut entamer. Le résultat fourni identifie le choix à entreprendre. Cette étape consiste à faire un balayage selon le résultat dans un sens de parcours de gauche à droite Vs droite à gauche ou du bas vers le haut Vs haut vers le bas.

On part du principe de regrouper l'ensemble des points qui appartiennent au même plan d'ajustement selon un axe (X ou Y ou Z) après avoir subit la phase de simplification, Ce plan formé est utilisé dans le calcul de polygone avec un deuxième plan superposé a celui-ci.

Dans ces exemples on illustre bien la formation de plans superposés et dans le but de rechercher les polygones en part bien sûr selon le schéma du bas vers le haut ou l'inverse.

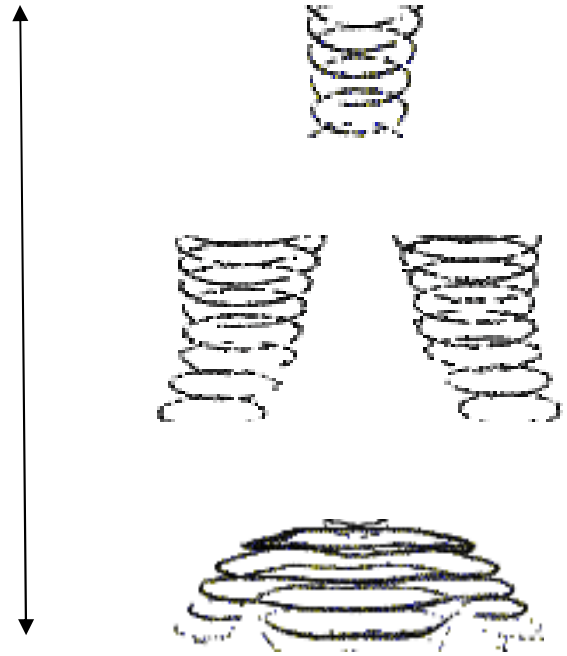


Figure 24. Sens du parcours à la recherche de polygones entre plans.

2.5 Extraction de polygones

Dans le but de chercher des polygones et selon le sens qu'on adopte pour le balayage des plans, on superpose les plans de manière à ce qu'ils soient voisins. Si on prend le sens du balayage celui du bas vers le haut, on commence par le premier plan Py0 avec le plan Py1.

Pour effectuer la recherche on a en entré les deux plans voisins Py0 et Py1 on procède de la manière suivante :

a- Chercher les points de Py0 et puis les points de Py1

b- Calcul les voisins des points dans Py0

c- Calcul les voisins des points dans Py1

d- Recherche des correspondances entre points des deux plans (deux passes)

Pour illustrer cela on considère un cas possible suivant :

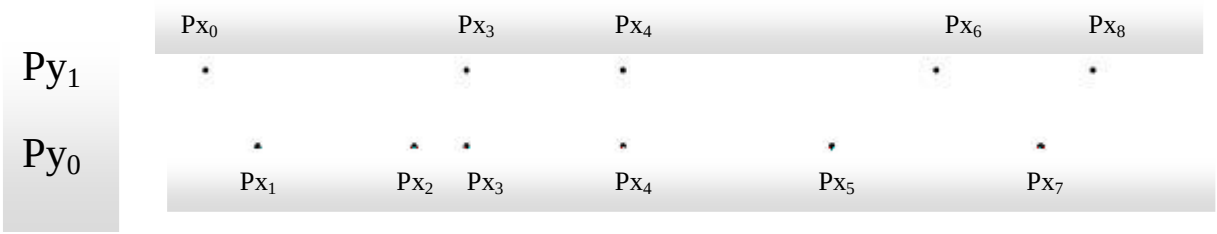


Figure 25. Deux plans voisins pour la recherche de polygone.

Pour tout point du plan Py_0 on cherche son correspondant (le plus proche selon la distance euclidienne) parmi les points du plan Py_1 et on prendra donc le $\min \|(P_i, P_j)\|$ Pour $i \neq j$ et j$\in [1..k]$ avec ***k*** le nombre de point du plan Py_1.

Dans l'exemple suivant nous n'avons présenté que quelques cas possibles parmi d'autres, bien sur il existe une multitude de combinaisons qu'il faudra traiter.

Le résultat de la recherche de polygone est illustré dans la figure suivante en tenant compte qu'on présente ici un algorithme à deux passes pour combler les insuffisances d'une seule passe.

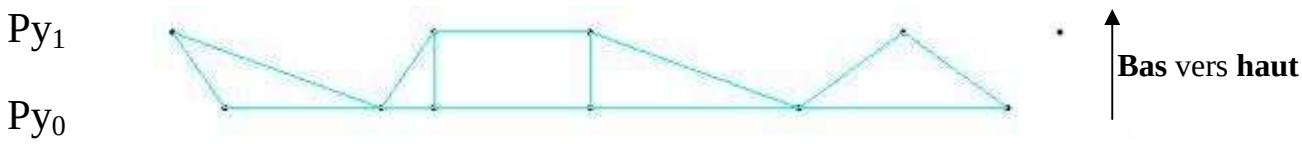


Figure 26. Première passe de recherche de polygones.

Ici le dernier sommet du plan Py_1 ne sera traité que dans la deuxième passe, du faite que le dernier du plan Py_0 n'a pas de voisin à sa droite.

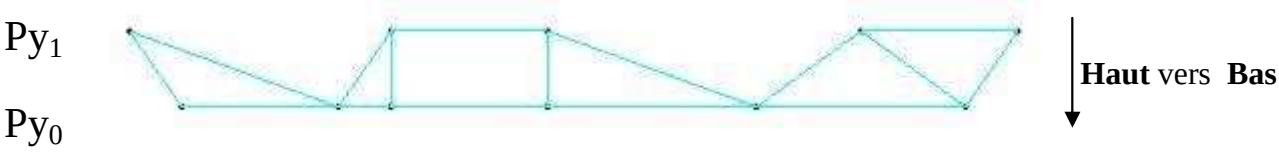


Figure 27. Deuxième passe de recherche de polygones.

Un autre exemple est présenté ici pour illustrer le résultat de la recherche de polygone à quatre sommets ou dans les pires des cas des triangles.

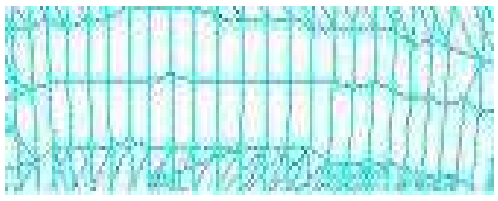


Figure 28. Schéma illustrant la reconstruction avec polygone 04 sommets ou triangle.

Donc, l'algorithme simplifié de recherche de polygones se résume ainsi :

Algorithme **chercherPolygone**(Py_1,Py_2) ;
Début
Répéter
    Si **a_voisin(***Py_1.P_i***)** alors
    Début
        S_1.P := Py_1.P_i ;
        S_4.P :=**Récupérervoisin (***Py_1, S_1.P***)** ;
        S_2.P :=Cherche Correspondant(Py_2) ;
        Si S_2.P.voisin <>0 **alors**
        S_3.P := **Récupérervoisin (***Py_2, S_2.P***)** ;
        Insère_Polygones (*S1,S2,S3,S4*) ;
    Fin ;
Jusqu'à traiteToutPoint(Py1) ;
Fin ;

Discussion :

Ce que nous avons présenté ici est l'aspect théorique de l'approche *point vs polygone* pour la reconstruction d'un nuage de point. Cette approche, basée sur la notion de plan d'ajustement parallèle dont l'objectif essentiel est de former des contours, le résultat espérer risque de ne pas se réaliser dans la pratique car si on le constate il n'y a pas de garantit pour que ces contours se forment tout au long des processus de partitionnement et de simplification. Face à cette problématique, nous sommes amenés à introduire la notion de *plan principale* et *plan complémentaire* que nous jugeons très utile pour remédier à cette insuffisance. Le plan principale désigne le plan de base choisit et les plans complémentaires désignent ceux qu'on estime compléter le plan principale afin de former un contour. Le nombre de plan complémentaire à joindre est déterminé d'une façon empirique ce qui conduit à une légère modification dans l'algorithme de recherche de polygone et au lieu de se basé

sur un plan et un autre voisin nous parlerons donc d'assemblage de plan de base et d'assemblage de plan voisin. Dans ce cadre là nous avons pensé à introduire un prédicat comme critère de teste pour assurer la formation du contour qui se résume en la vérification que *tout point est voisin et a un voisin*. Ce prédicat est bien une réponse mais la convergence n'est pas assurer car dans le cas d'un échantillonnage sous forme de spirale, présente un risque où on ne s'arrêtera pas à joindre des plans jusqu'au dernier.

Un autre problème qui mérite d'être évoqué est celui d'une partition qui présente un seul point. C'est une assurance de la convergence de l'algorithme de partitionnement mais présente une incohérence, par exemple un plan de base qui contient une dizaine de points et un plan voisin avec un seul point, évidement une non correspondance durant la phase de recherche de polygones. Dans notre cas on le considère comme un point aberrant. Par conséquent, ce plan est éliminé durant la sélection pour la reconstruction.

Le dernier cas que nous mentionnant est que la notion de voisinage (problème de continuité) entre les points est définie implicitement entre les points de plan voisin et explicitement dans un même plan. Ici, les points voisins (dans un même plan) sont pris par leur nature après la phase d'échantillonnage, puisqu'on prend la distance minimale comme critère de teste pour chercher le plus proche voisin d'un point. Par contre, dans le cas de plan voisin, le plan de base et le plan voisin est le suivant par le fait qu'on le calcul dans l'ordre des écarts.

Conclusion

Après avoir décrit la conception de notre technique il nous reste enfin que de la valider et voir les résultats dans le chapitre implémentation. On rappelle ici que nous partons toujours dans tous les cas d'un nuage de points sans connaissance de topologie ni d'adjacence entre ces points et l'information dont on a besoin c'est uniquement les coordonnées en 3D de ces points, en plus nous tenons à préciser que les problèmes qui ont été évoqué sont fortement lié à la façon dont on procède à l'échantillonnage de l'objet. Une numérisation horizontale, verticale ou hybride nous avons opté pour une hypothèse qu'un nuage de point présente suffisamment de régularité. Notons aussi que le choix de simplification s'avère stratégique où l'on opère sur un échantillon réduit ce qui facilite la tâche d'extraction de facette polygonales avec un nombre réduit de triangles ce qui accélère d'avantage le rendu.

L'une des caractéristiques essentielles de notre étude, c'est qu'en partant d'un nuage de points vers un ensemble de facettes (polygonales) sans calcul des normales en ces points et le résultat est exploitable directement dans le Pipeline graphique.

Pour illustrer cela, nous l'avons implémenté dans une application afin d'en tirer les avantages et les inconvénients de notre technique. Ce qui est décrit dans le chapitre suivant.

CHAPITRE V

Implémentation

Points Vs Polygones

1. Introduction

Pour mettre en évidence la technique que nous avons élaborée dans la partie conception, nous avons développé une application sous l'environnement Delphi afin de tester toute les phases de conception et cela nous a permis d'observer de près le résultat du concept et les insuffisances ayant été constatées après des exécutions sur différents nuage de points. Nous présentons ici les grandes lignes de l'application ainsi que la structure de données qui a servi au stockage du nuage de points.

2. Stockage des points

Nous avons en entrée un fichier de points qui contient les coordonnées selon les 03 axes dans l'espace objet. Afin d'introduire ces données dans le programme nous avons opté pour une structure statique. Ce choix est évidement insuffisant pour des nuages de plus de 1 millions de points et en plus on ne connait pas la taille réelle du nuage. Donc ce qu'on préconise est d'utiliser les structures dynamiques pour récupérer le nuage de points.

Nous rappelons que les informations qui concernent les points sont uniquement ses coordonnées, aucune autre information n'est livrée dans le fichier par exemple couleur ou information concernant la topologie entre ces points.

La structure de récupération de ces données est un tableau d'enregistrement à une dimension de taille fixé par une constante.

```
Const  NbrePoints= 500000 ;
    coord=record
        xcord, ycord, zcord: real; // les coordonnées d'origine des points
        xtransf, ytransf, ztransf : real; // coordonnées transformées
        Plx, Ply, Plz: Integer; // Plans d'appartenance
        Cle :Integer; // numérotation du point
        end;
    Var Tabpoints : array [1..NbrePoints] of coord;
```

Ici le nombre de points de la constante **NbrePoints** est fixé pour de faibles nuages de points. Ce tableau sert donc à stocker toutes les informations d'un point au

moment du chargement du fichier ainsi qu'au moment du traitement. Le processus que doit subir ce nuage de point est illustré dans la figure ci-dessus.

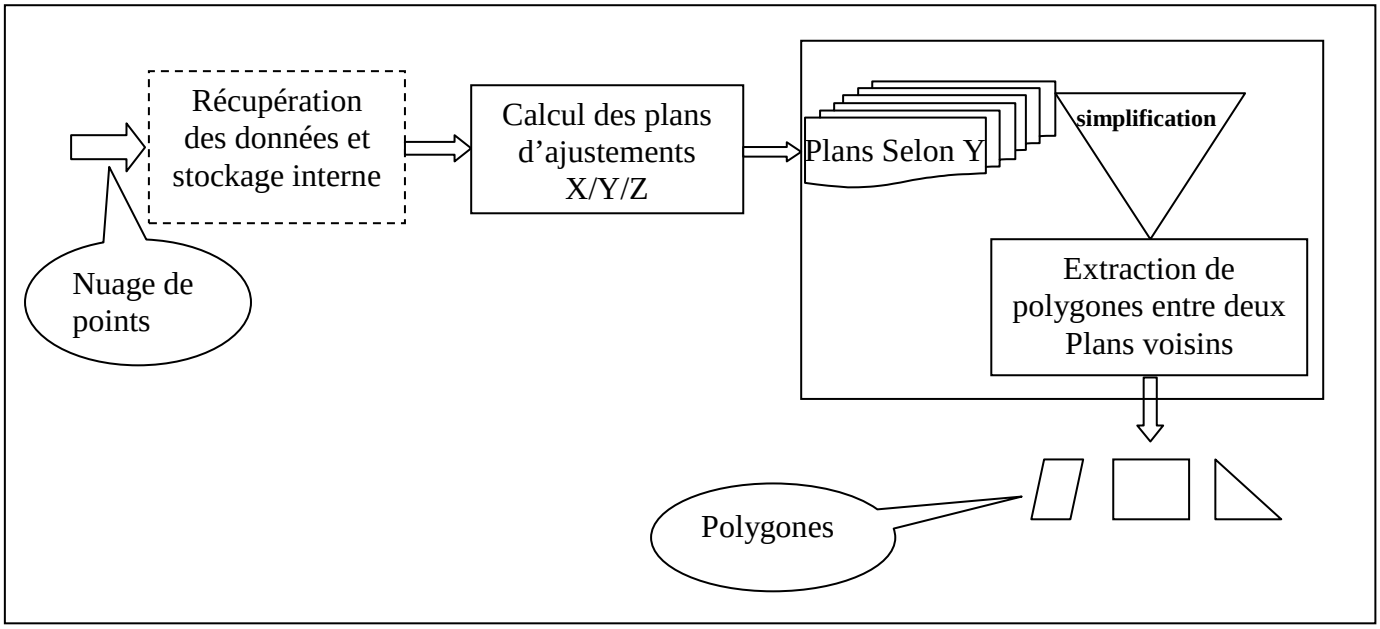


Figure 29.Processus de reconstruction d'un nuage de points.

3. Phases de la Reconstruction

3.1-Récupération des points

Dans cette étape les données sont stockées initialement dans un fichier texte et afin de récupérer ces données nous avons utilisé SQL server pour les importer ensuite le processus s'ensuit par une lecture depuis le fichier destination dans la base et puis stocker les coordonnées des points dans le tableau **TabPoints**. Ce qu'on récupère du fichier c'est uniquement les *xcord, ycord et zcord* des points tandis que la numérotation des points se fait selon l'ordre de lecture du fichier de base.

Finalement on obtient un tableau rempli de coordonnées des points et numéroté de 1 jusqu'à « N ».

3.2-Evaluation des estimations

Durant cette phase on s'est servie de 03 tableaux de structure réduite et qui ressemble au tableau **TabPoints** pour stocker dans l'ordre les x_i, y_j et z_k dans respectivement TabP*x* TabP*y* et TabP*z*. Comme les valeurs sont triées dans l'ordre croissant on procède alors à l'évaluation des estimations respectivement pour chaque axe et selon l'algorithme décrit dans la conception. Ces paramètres peuvent subir une variation d'une manière interactive selon le besoin de l'utilisateur c'est-à-dire lui permettre de voir le résultat tout en variant les estimations soit par pondération ou par réduction (réduire une estimation par division).

3.3-Calcul des plans d'ajustements

Les tableaux TabP*x* TabP*y* et TabP*z* sont utilisé pour le calcul des plans d'ajustements suivant l'algorithme cité dans le chapitre précédent (conception-Calcul des Plans) en prenons en compte les estimations calculées pour le partitionnement de points du nuage.

a- Partitionnement et structure de donnée

Après avoir calculé l'appartenance des x_i aux plans correspondants Px_i ainsi que les y_i aux plans Py_j et les z_k dans les Pz_k chaque point de **TabPoints s**aura son appartenance aux 03 plans de partitionnement ainsi que ses coordonnées transformés.

Le résultat fourni par le calcul des plans n'est pas cohérent vis-à-vis de l'ordre des plans d'où la nécessité de réorganisé les valeurs en numérotant les plans dans l'ordre croissant.

Dans cette étape il est impératif de se servir d'une autre structure de stockage adéquate pour entamer la prochaine étape. Avec le tableau **TabPoints** la recherche de

points qui appartiennent à un PX_i , Py_j et Pz_k aura des conséquences sur le temps de calcul et ralentira ainsi la vitesse de traitements. Donc on s'est servi d'une structure dynamique simple après avoir calculé les plans, les données du tableau **TabPoints** seront transférées dans cette nouvelle structure dynamique **ListeY** sous forme d'arborescence qui a montré son efficacité en minimisant le temps d'accès pour récupérer les valeurs des données.

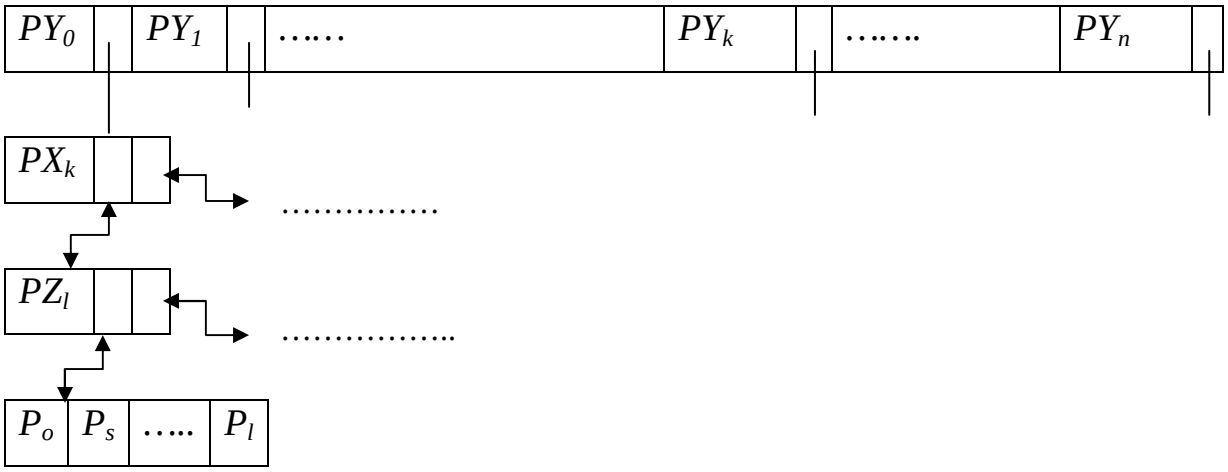


Figure 30.Structure de donnée illustrant le partitionnement des points.

b- Influence de l'estimation sur le calcul des Plans

Le partitionnement du nuage en plan est dépendant de l'estimation choisit pour chacun. Ce choix aura un impacte directe sur la reconstruction car plus l'estimation est grande moins est le nombre de plan à calculer, ce qui aboutit à une grande simplification dans la phase pré-reconstruction.

La figure 31 montre l'influence directe sur le calcul des plans.

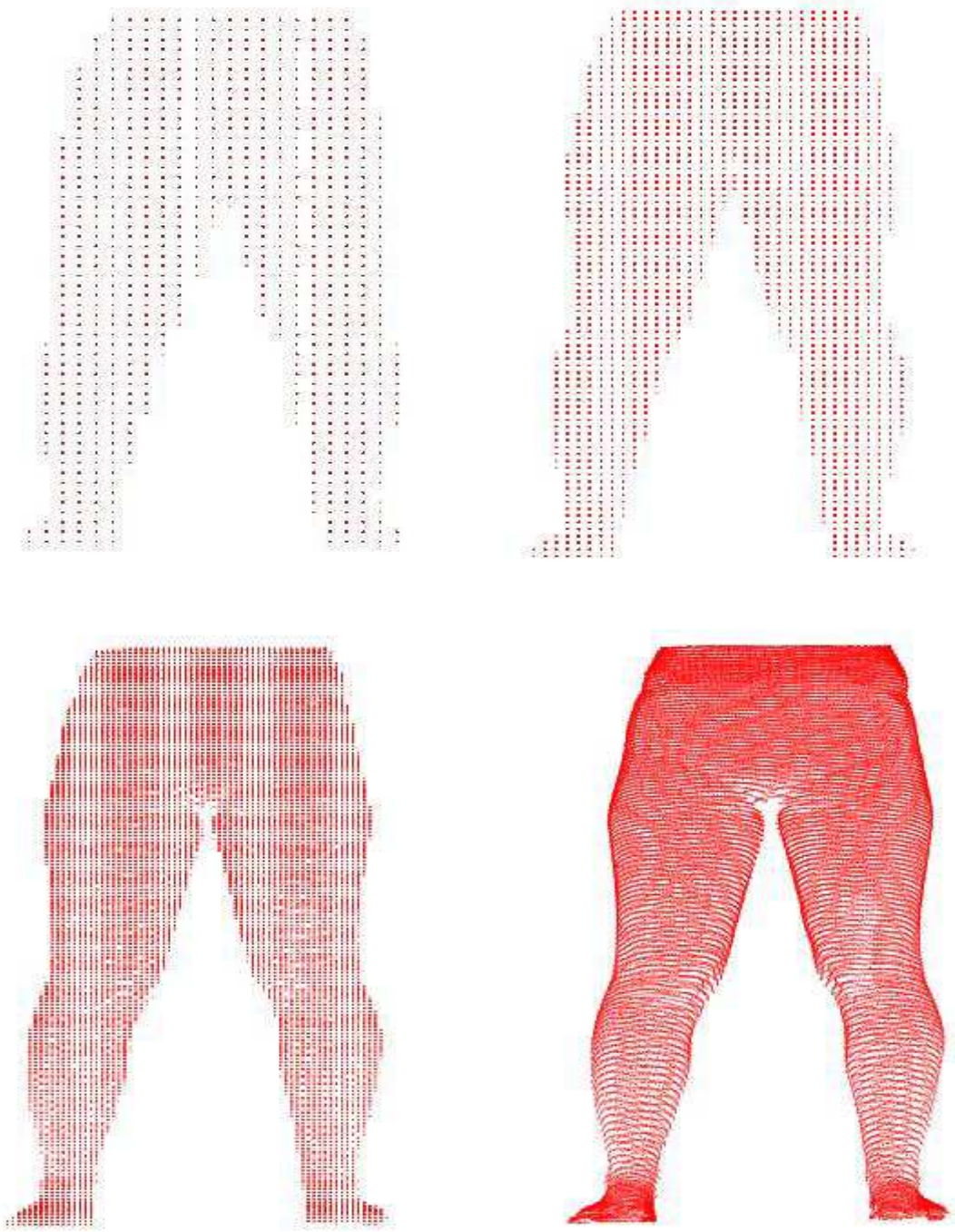


Figure 31.Influence du choix de l'estimation sur le calcul des plans d'ajustements

1 er Niveau A gauche Grande Estimation - A droite un peu moins

2eme Niveau A gauche Moyenne Estimation - A droite l'estimation des écarts Mdm.

	Estimation*30	Estimation*25	Estimation*10	Estimation
Nombre de Plan X	**23**	**28**	**69**	**660**
Nombre de Plan Y	**54**	**65**	**161**	**1530**
Nombre de Plan Z	**13**	**16**	**38**	**373**

Tableau 1. Le nombre de plan généré en fonction de l'estimation

Ce tableau résume parfaitement l'effet de pondération de l'estimation sur la génération des plans selon les 03 axes. La dernière colonne montre bien si on utilise uniquement les estimations calculées en fonction des écarts de valeurs le nombre de plans selon chaque axe est très grand mais ça induit une bonne approximation du nuage de point avec le moins d'effet escalier que dans les autres cas. On notera ici que les estimations pondérées sont respectivement pour chaque axe ξ_x, ξ_y, ξ_z.

$$\xi_x = 1.81 \ , \xi_y = 0.63 \ , \xi_z = 0.72$$

Chaque partitionnement selon un axe est traité indépendamment des autres, or il est parfaitement possible de modifier une estimation et garder les autres fixe. Par exemple dans le tableau ci-dessus on peut garder les estimations ξ_x et ξ_z fixes et modifier ξ_y.

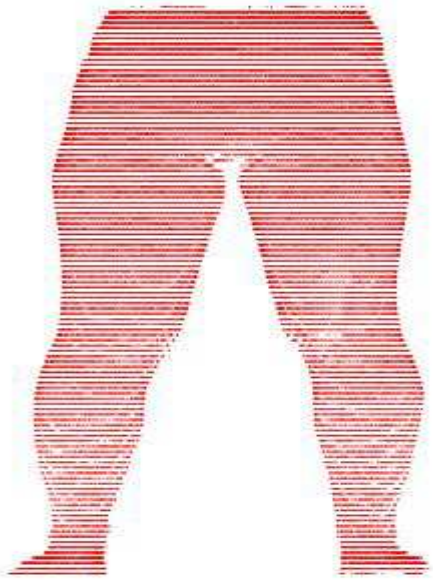


Figure 32. Estimation selon Y modifier

ξ_y *15, nombre de Plan Y= 180.

3.4- Reconstruction de surface

Une fois le calcul des plans d'ajustements est établi et le transfert des points vers la structure dynamique on peut maintenant procéder à la phase de reconstruction tout en regroupant la simplification et l'extraction de polygones. Ce que produit l'étape de simplification d'un plan Py_p est exploité par l'étape suivante.

a- Assemblage de plan

Dans le but de pallier à l'insuffisance du nombre de points dans un plan P_y et aussi la non correspondance directe entre deux plans voisin, nous avons opté pour une alternative d'assemblage de plan (*plan principale plan complémentaire*). Cet assemblage aura pour effet de maximiser le nombre de plan P_x dans un plan P_y et permettra ainsi de maximiser l'extraction de polygones. Ce paramètre est choisit d'une manière empirique est fortement lié à la densité du nuage dans une zone.

La figure 34 montre clairement l'effet d'assemblage de plan après la phase de reconstruction.

Figure 33. Paramètre d'assemblage, à gauche élevé, à droite faible.

b- Simplification

Soit un Plan initiale PY_0. La simplification consiste à parcourir toute les combinaisons des Plans X_1 et de même pour un autre plan X_2 appartenant à la branche

du plan *Y0* est exploré de la même manière suivant les plans *Z*. cette recherche abouti aux feuilles de l'arbre qui sont les points qui appartiennent aux plan PY_0 , PX_l et PZ_v.

Cette partition est chargée dans un tableau ensuite un calcul de distance par rapport à l'origine du repère est effectué, le point qui a la plus grande distance est sélectionné pour contribuer à la formation de polygones. De même toutes les sous branches des plans Z du plan PX_l du plan PY_0 sont explorés jusqu'au parcours total du plan PY_0.

Donc on obtient finalement l'ensemble des points du plan PY_0 candidats dans la recherche de polygones.

Dans cette étape on produit séquentiellement les plans PY_0 puis PY_1 et ainsi de suite jusqu'au plan limite selon Y. Une sous étape préparatrice pour la reconstruction est celle du calcul de voisinage des points du plan Y simplifié. Ici, il faut s'assuré que tout point possède un voisin et il est voisin ; le voisinage est calculé selon la distance euclidienne *min* entre les points (le plus proche voisin).

Donc, le plan PY_0 est bien préparé pour la prochaine étape et ainsi que le plan suivant PY_1.

c- Extraction des polygones

Le résultat de la simplification est exploité de la manière suivante :

Deux plans doivent être produis initialement, la recherche des polygones s'effectue selon l'algorithme décrit dans la conception, une fois que tout les points du premier plan sont exploré avec leur correspondant dans le plan suivant on inverse le processus (deuxième passe) pour combler les insuffisances d'une seule passe.

On obtient finalement, des polygones à 04 sommets ou des polygones et des triangles. Le deuxième plan qui a servit au premier va être considéré après la recherche comme premier support de recherche et l'étape de simplification d'un nouveau plan est sollicitée.

Deux structures statiques (tableau) sont utilisé pour faire la recherche l'un pour le premier plan et l'autre pour le second.

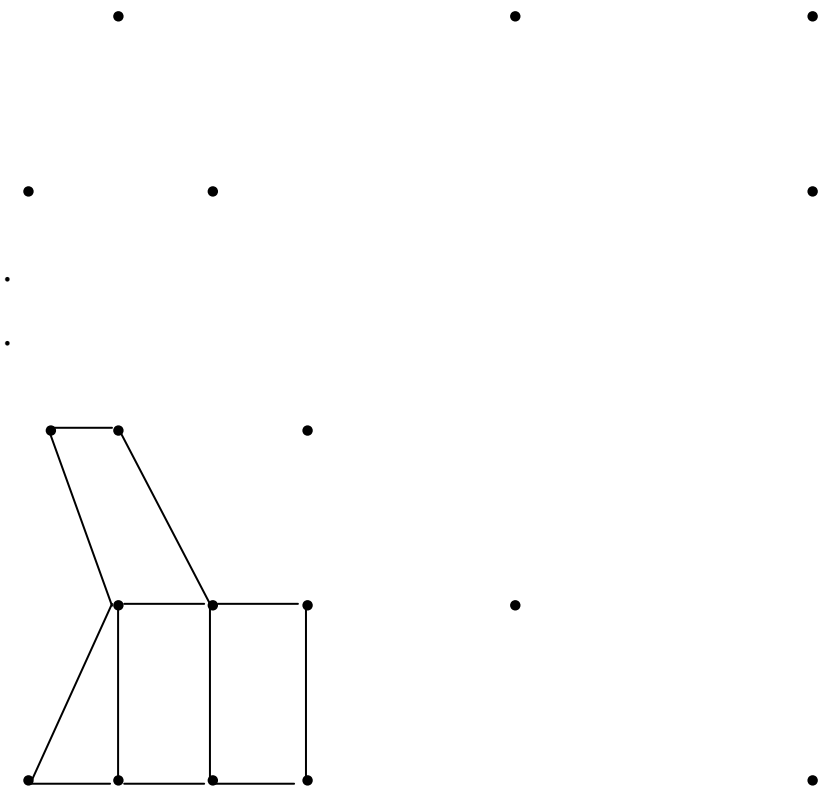


Figure 34, Schéma résumant le processus d’extraction de polygones.

4. Rendu

A la fin de l’étape d’extraction de polygones, l’étape finale est celle du rendu pour visualiser le résultat après l’application de cette technique. Nous avons travaillé dans l’espace scènes sans l’utilisation des projections (perspective ou autre), avec des outils simple d’OpenGL tel que GL_POINTS, GL_POLYGON et GL_TRIANGLES nous avons obtenus des résultats plus ou moins satisfaisants.

Les sommets des polygones sont stockés dans un tableau avec une indication si c’est un triangle ou un polygone à quatre sommets. Pour des raisons d’homogénéité durant la phase d’implémentation nous avons choisit de réduire les polygones à des triangles. Comme ça nous n’aurons plus besoin de distinguer entre triangles et polygones, donc une seule procédure de rendu suffira à parcourir le tableau de sommets qui sont fournis aux commandes OpenGL afin de visualiser le résultat de la reconstruction.

5. Résultats

Nous avons utilisé trois fichiers de nuage points téléchargé à partir d'un site Internet **www.korux.com,** ces trois nuages de points ont été exploités pour tester l'efficacité de notre approche.

Le résultat été très attendu malgré les insuffisances et la complexité du domaine sans connaître aussi les conditions d'acquisitions de ces nuages de points.

Figure 35. Image 1 constitué d'un nuage de points (bassin)

Nombre de points	Estimations	Nombre de plans Généré	Temps de reconstruction	Nombre de Triangles générés
41577	Ex = 1.810 Ey = 0.631 Ez = 0.721	Px = 236 Py = 1530 Pz = 373	11 s	71036

Tableau 2. Caractéristiques après reconstruction de l'image 1, assemblage P=10

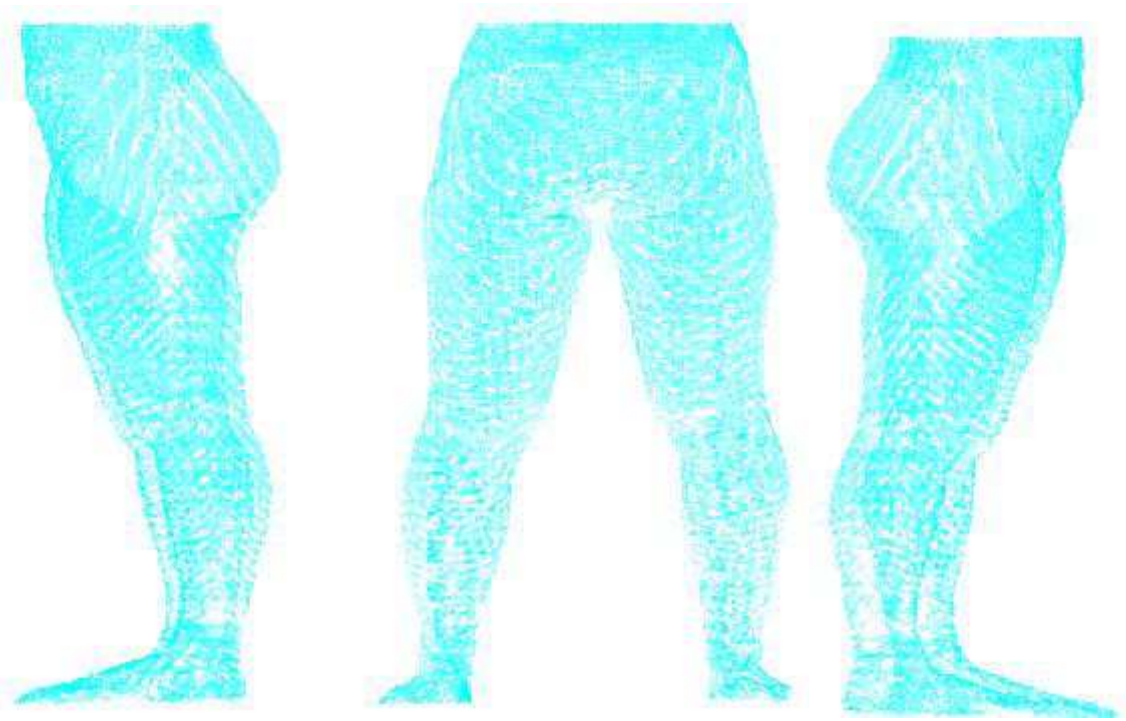


Figure 36. Résultat de la reconstruction de l’image 1, alpha Blend = 0.25

Figure 37. Résultat de la reconstruction de l’image 1, alpha Blend = 1

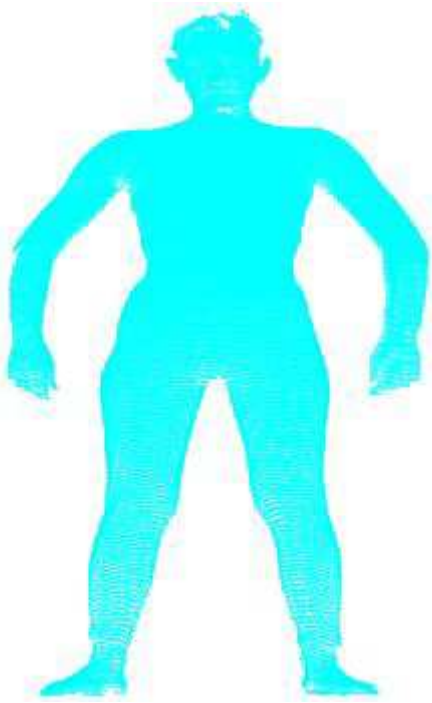


Figure 38. Image 2 constituée d'un nuage de points (Body2).

Nombre de points	Estimations	Nombre de plans Généré	Temps de reconstruction	Nombre de Triangles générés
107417	Ex = 0.395 Ey = 3.137 Ez = 1.565	Px = 1725 Py = 521 Pz = 172	05 mn :50 s	179860

Tableau 3. Caractéristiques après reconstruction de l'image 2, assemblage P=10

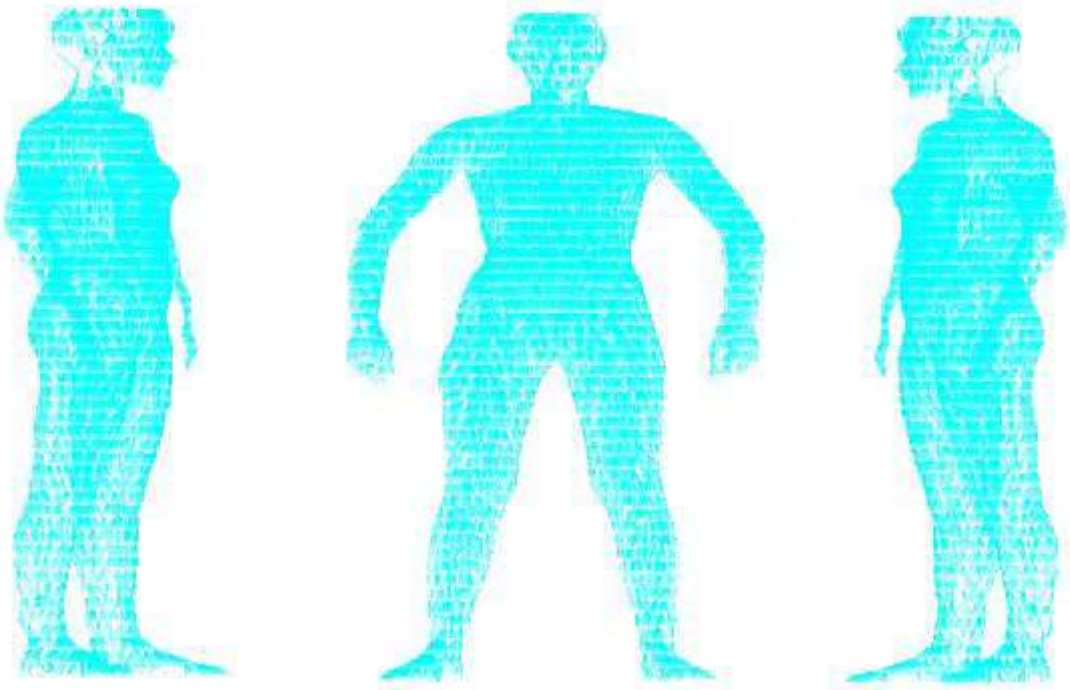


Figure 39. Résultat de la reconstruction de l'image 2, alpha Blend= 0.25

Figure 40. Résultat de la reconstruction de l’image 2, alpha Blend= 1.

Dans la figure 41 nous avons modifié le nombre de plan d’assemblage en le réduisant à 2. Le résultat est détaillé dans le tableau 3.

Le temps comme on le constate a considérablement diminuer, ceci est dû au nombre de points à traiter (simplification et calcul de voisinage dans le plan).

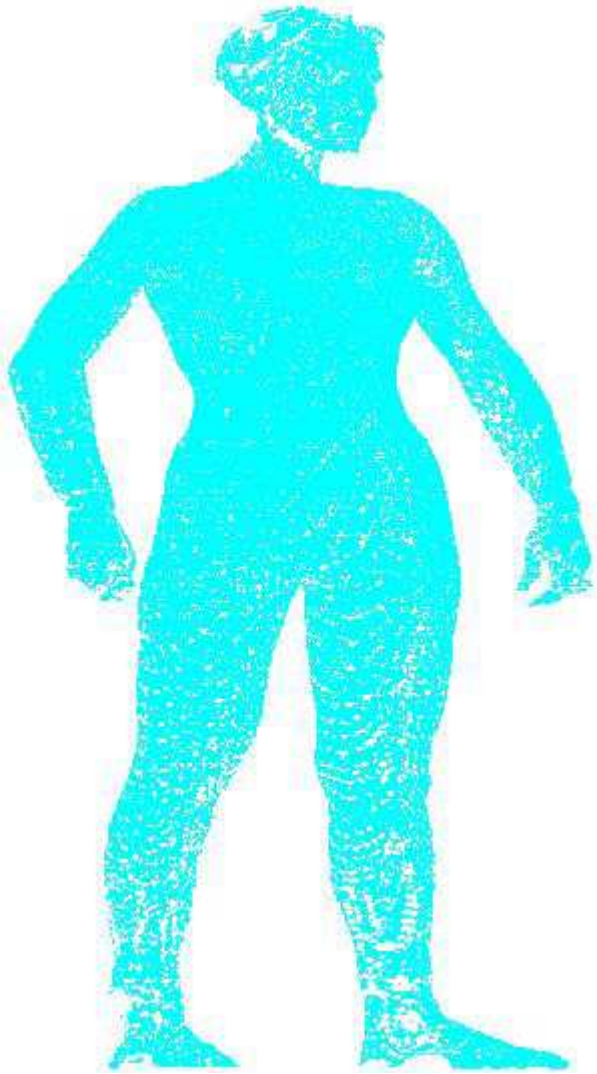


Figure 41, Image 2 reconstruite, Alpha Blend = 0.25, P=2.

Nombre de points	Estimations	Nombre de plans Généré	Temps de reconstruction	Nombre de Triangles générés
107417	Ex = 0.395 Ey = 3.137 Ez = 1.565	Px = 1725 Py = 521 Pz = 172	01 mn :36 s	186424

Tableau 4. Caractéristiques après reconstruction de l'image 2, assemblage P=2

Figure 42. Image 3 constituée d'un nuage de points (mannequin).

Nombre de points	Estimations	Nombre de plans Généré	Temps de reconstruction	Nombre de Triangles générés
214666	Ex = 0.912 Ey = 1.489 Ez = 0.865	Px = 673 Py = 1136 Pz = 441	11 mn :00 s	360417

Tableau 5. Caractéristiques après reconstruction de l'image 3, assemblage P=10

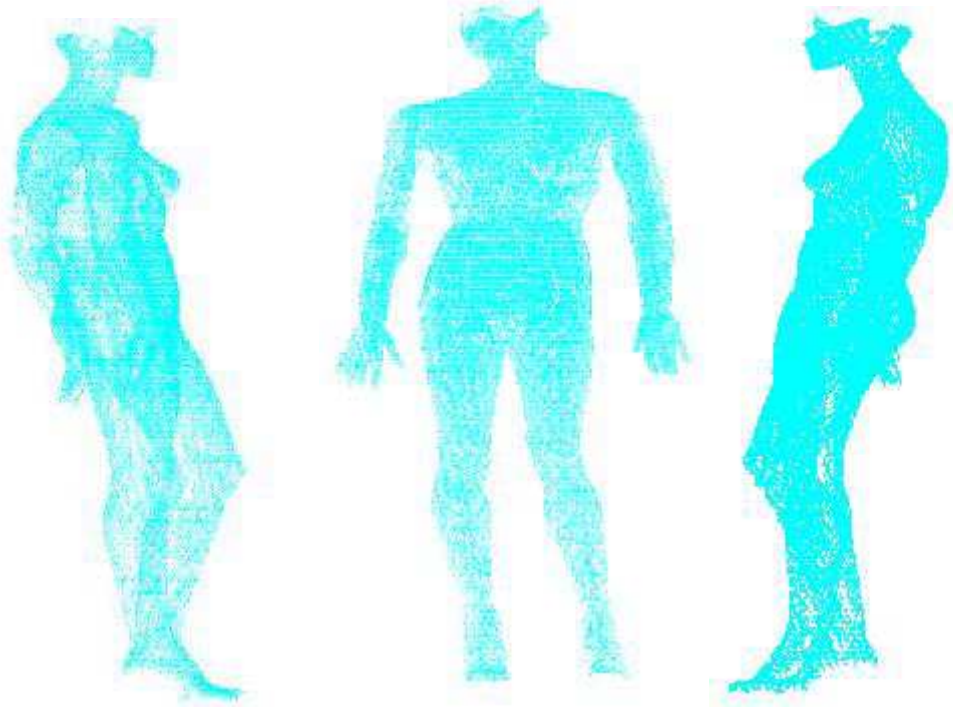


Figure 43, Image 3 reconstruite, A gauche, centre Alpha Blend = 0.25, à droite Alpha=1, P=10.

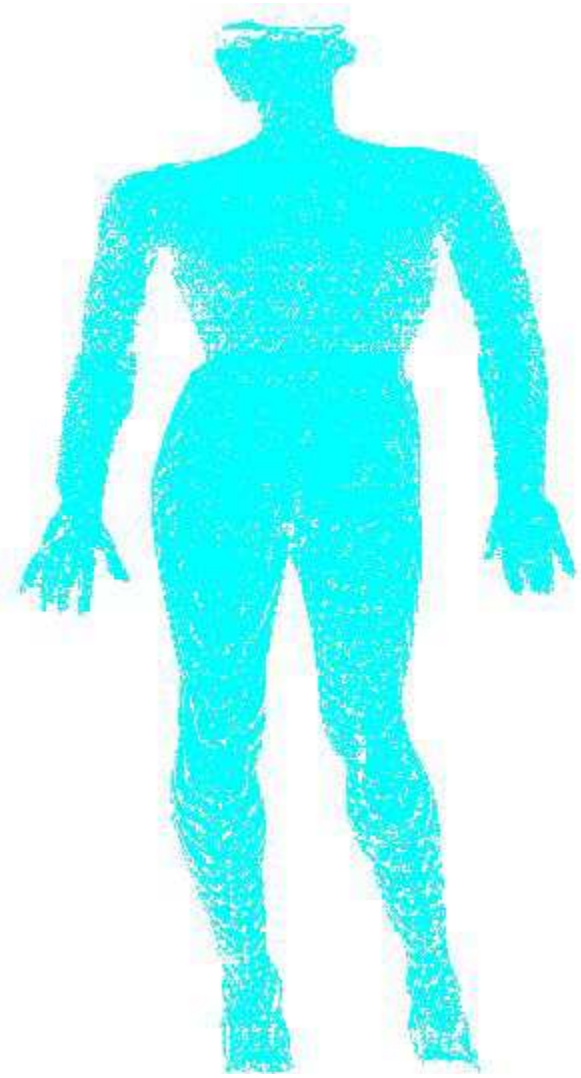


Figure 44, Image 3 reconstruite, Alpha Blend=1, P=5.

Nombre de points	Estimations	Nombre de plans Généré	Temps de reconstruction	Nombre de Triangles générés
214666	Ex = 0.912 Ey = 1.489 Ez = 0.865	Px = 673 Py = 1136 Pz = 441	06 mn :03 s	336715

Tableau 6. Caractéristiques après reconstruction de l'image 3, assemblage P=5

5.1- Placage de textures

Dans la réalité les textures ont toujours été utilisées pour l'amélioration de la qualité du rendu. En effet, leur apport en générale est d'affecter un certain réalisme sur les images produites dans le domaine de la synthèse d'image. Pour saisir cette

opportunité nous avons tenté d'appliquer une texture sur les images résultats de la reconstruction par notre technique. Pour réaliser cela, plusieurs images ont été téléchargées d'Internet (fig.45) et ont servis au placage de texture lors de la phase de rendu. Dans les exemples que nous allons présentés par la suite certaine d'entre eux sont affiché avec l'activation du tampon de profondeur (Z-Buffer).

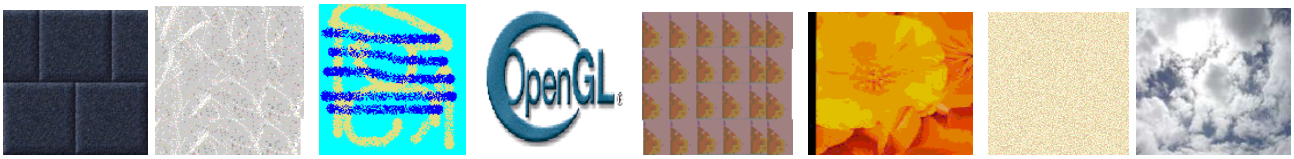


Figure 45, Des images pour le placage de texture respectivement
Im1, im2, im3, im4, im5, im6, im7, im8. (128x128 pixels)

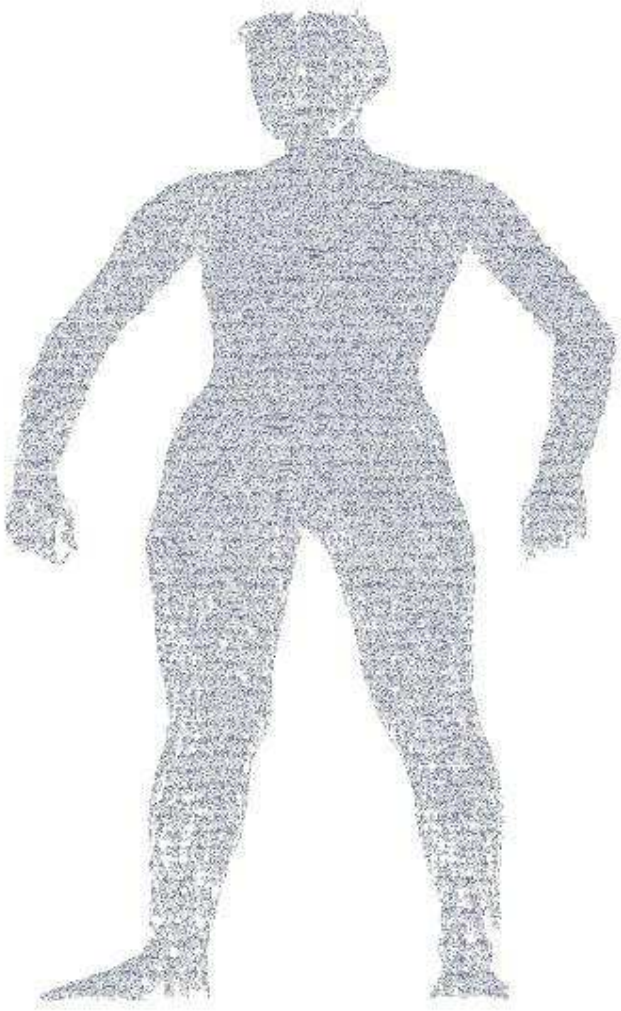


Figure 46, placage de texture im8 sur l'image 2.

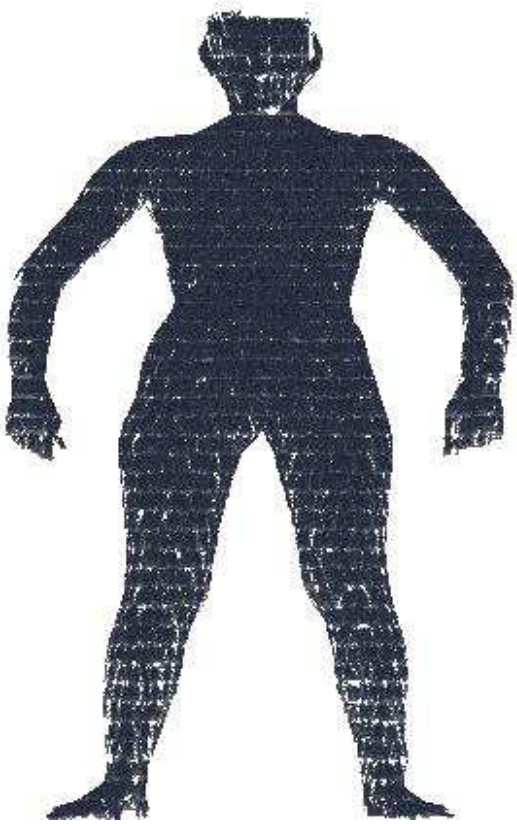


Figure 47, placage de texture im1 sur l’image 2.

Ici, le placage de texture est appliqué et chaque triangle prend toute la dimension de la texture (correspondance totale). Par la suite nous allons voir le placage par correspondance partielle c'est-à-dire que chaque triangle ne prendra que la proportion selon ses coordonnées par rapport aux coordonnées de la texture.

Nous mentionnant ici que dans le contexte du placage de texture, nous avons intégrer un traitement pour le calcul des correspondances entre les sommets et les coordonnées de la textures adéquates. Cette étape est divisée en deux parties :

- Pendant l‘extraction des triangles

  L’information qui concerne les deux plans qui ont servis à l’extraction est désignée par *segment*. Donc chaque sommet du triangle est affecté à un segment.

- Après l’extraction et avant le rendu

  Un parcours de la table qui contient les sommets en fonctions des segments pour affecter à chaque sommet ses coordonnées de texture adéquates avant le passage vers la phase du rendu.

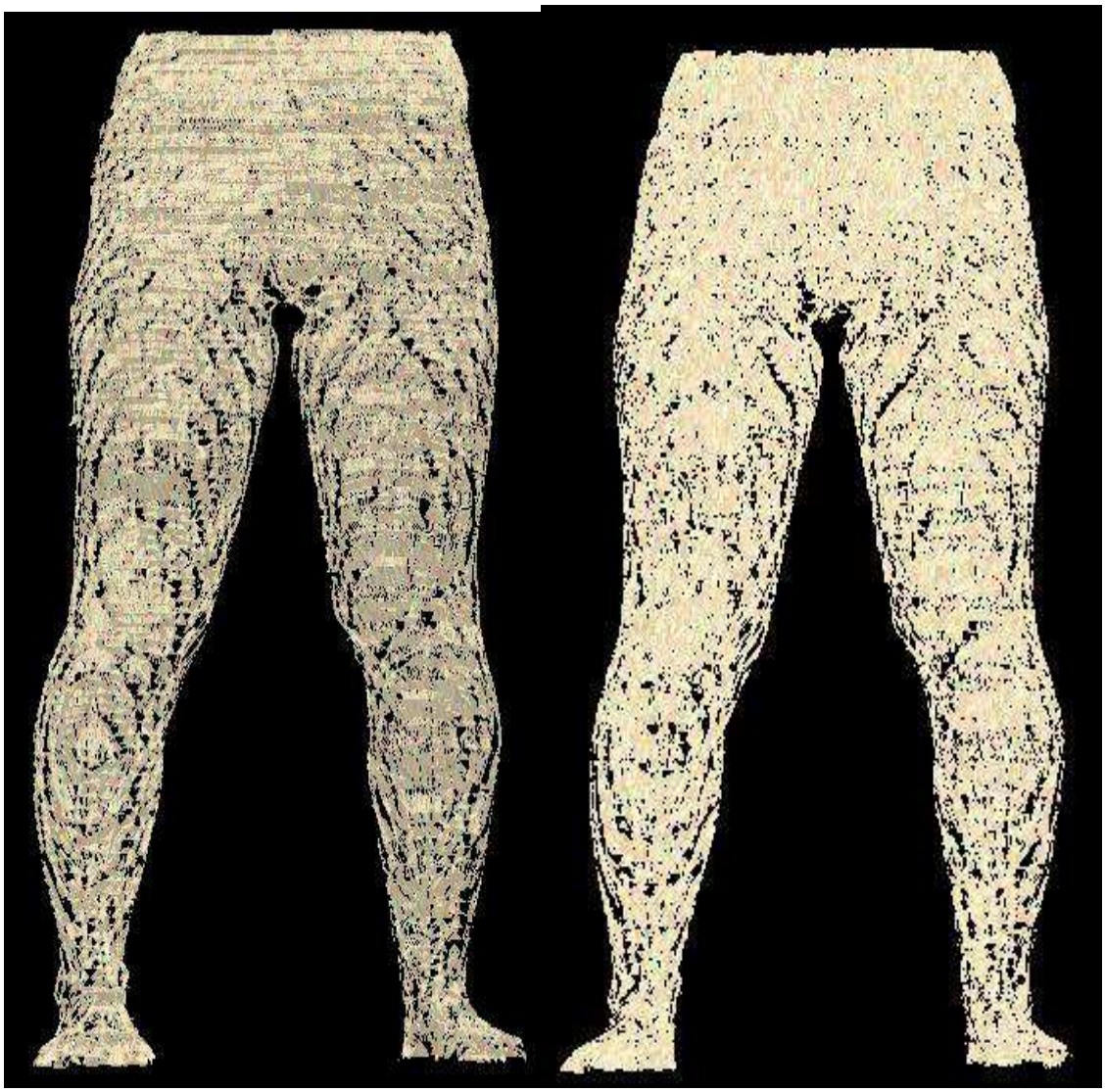


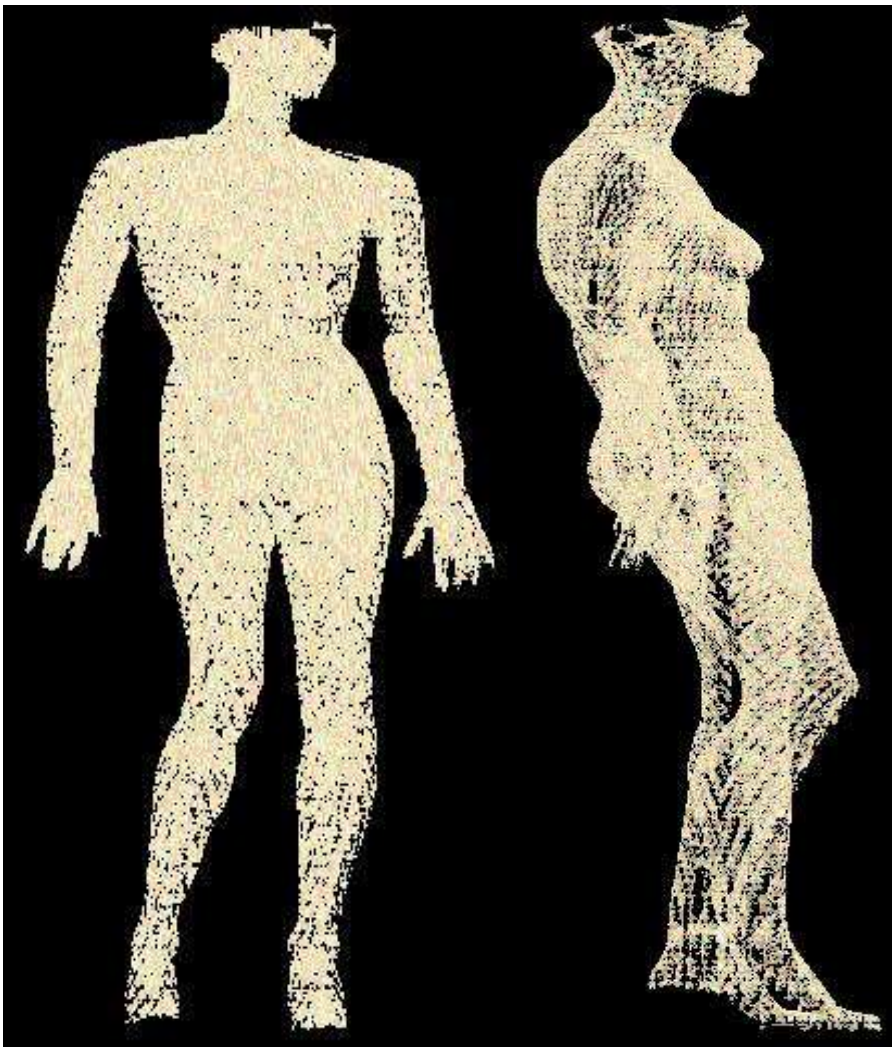
Figure 48 Placage de texture im7 sur l’image 1

A Gauche Alpha Blend =0.75, A droite Alpha Blend =1

Figure 49 Placage de texture Im7 et Im1 sur l'image 1

A gauche : avec activation du tampon de profondeur

A droite sans Activation du tampon de profondeur avec Alpha Blend = 0.25



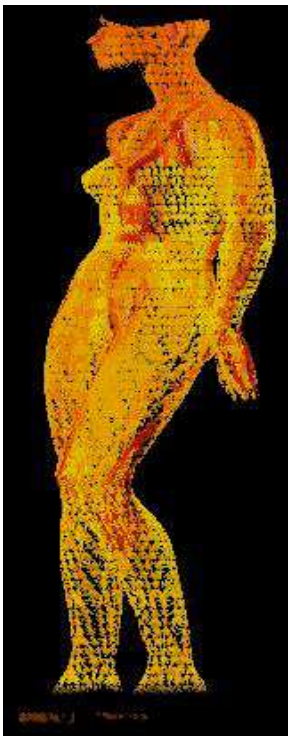


Figure 50 Placage de texture Im7 en haut sur l'image 3, droite (Z-Buffer activé) à gauche non activé
En bas Im6 sur l'image 3, Alpha Blend= 0.25

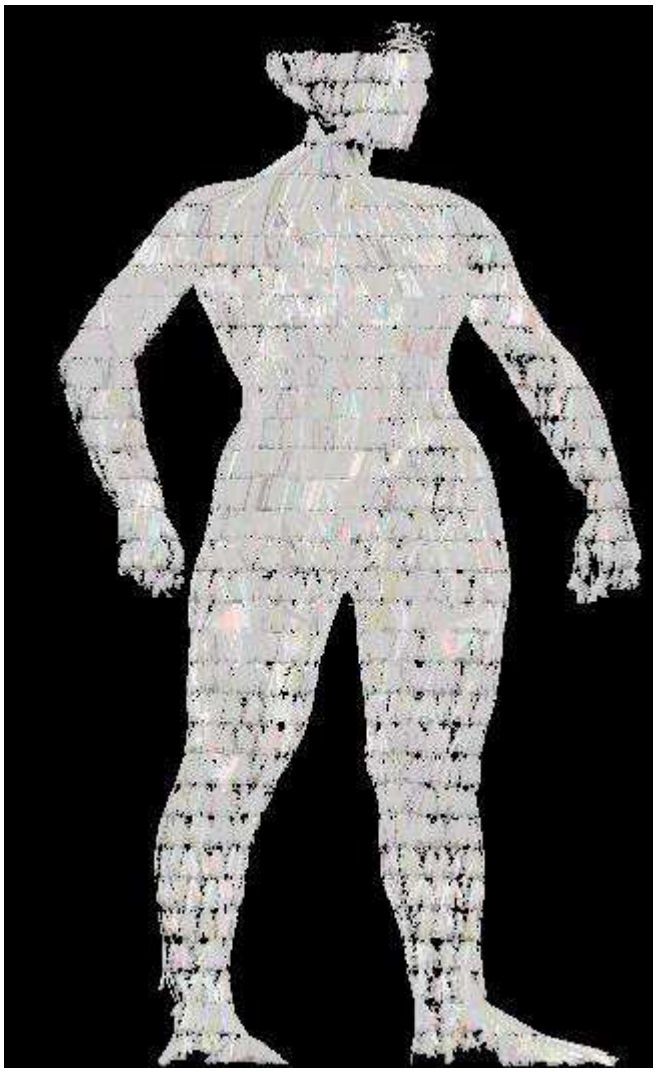


Figure 51, Placage de texture Im2 sur l’image 2.

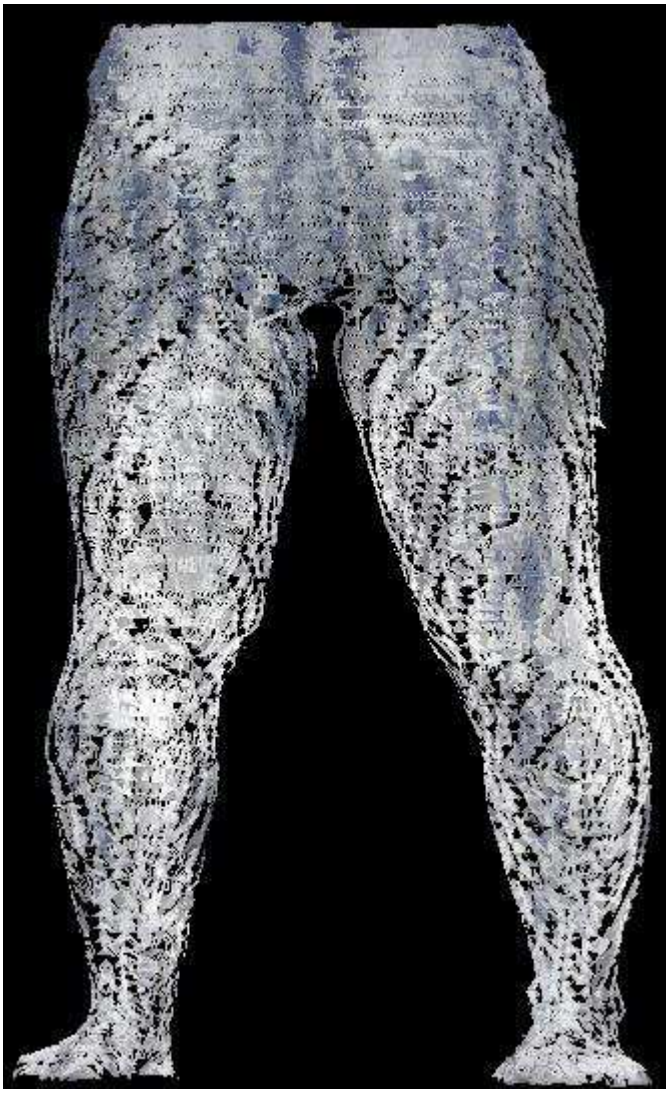


Figure 52, Placage de texture Im8 sur l’image 1.

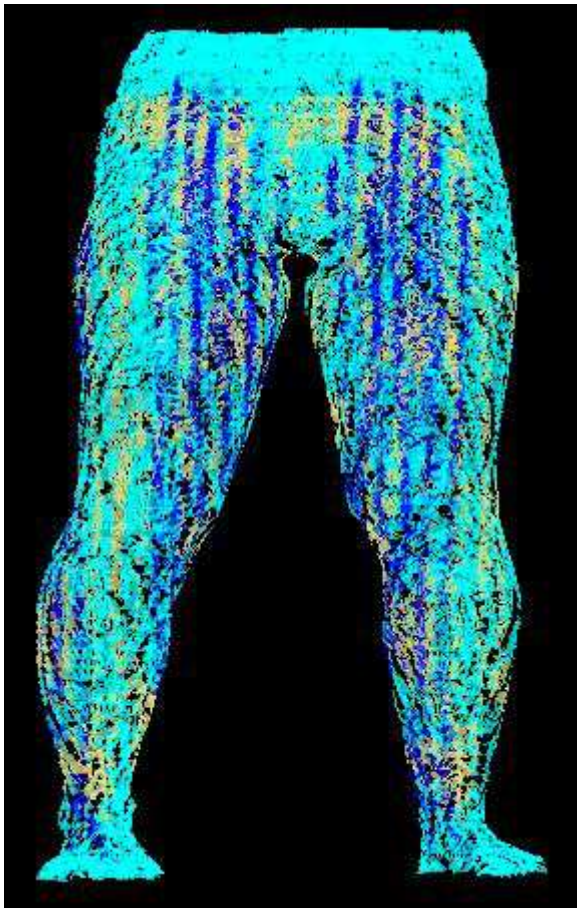


Figure 53, Placage de texture Im3 sur l’image 1.

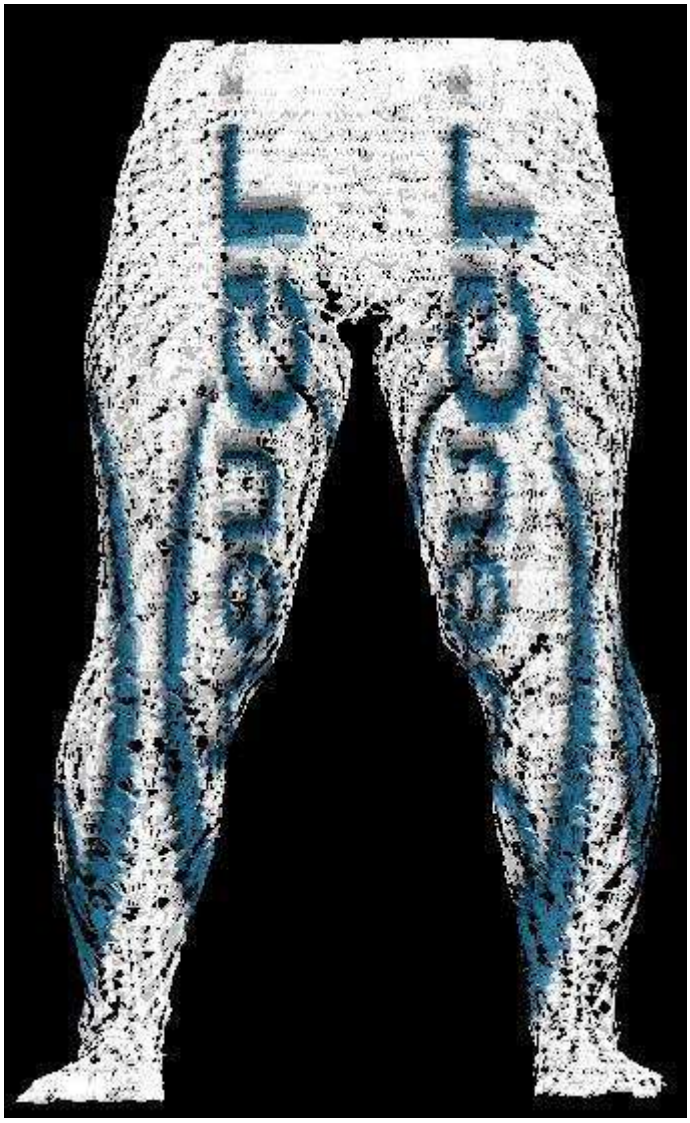


Figure 54, Placage de texture Im4 sur l’image 1.

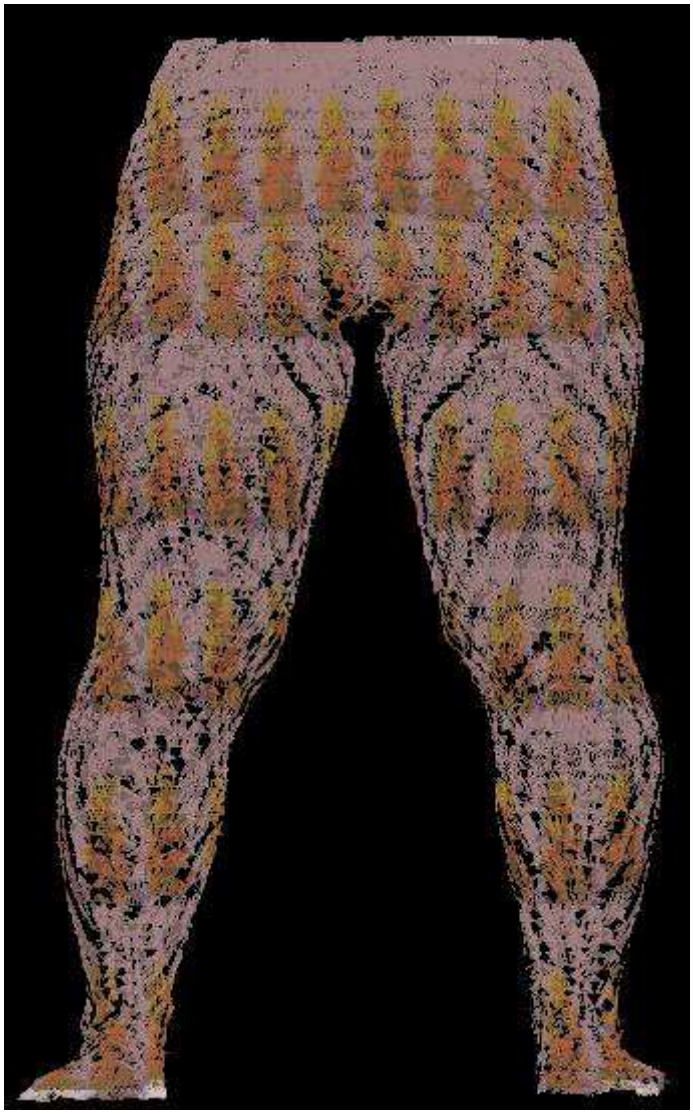


Figure 55, Placage de texture Im5 sur l’image 1.

5.2- Discussion

Comme tout autre technique celle-ci présente aussi des avantages et des inconvénients.

Nous commençons à citer les inconvénients d'après les résultats et aussi nos constatations durant tout le parcours de cette étude et réalisation.

Inconvénients :

- Cette technique est moyennement robuste aux zones sous-échantillonnées.
- La simplification génère des trous pendant la reconstruction
- Génère aussi des petits triangles dus à l'assemblage des plans et par conséquent on perd des détails très fins.
- L'algorithme d'extraction des triangles est encore loin d'être satisfaisants (ne traite pas tout les cas possibles).
- Les zones limite plate ne sont pas traitées à moins de changer le sens du balayage.
- Le temps de reconstruction dépend du nombre de points.

Avantage :

- Simple à implémenter et très efficace devant un échantillon de petite et moyenne taille.
- Possibilité de généré des variétés de reconstruction (modifier le choix dans l'estimation ou bien le choix des points dans la phase simplification).
- Possibilité d'interagir avec le résultat en modifiant les paramètres d'une zone.
- Possibilité de calculer la normal (après reconstruction) en fonction des sommets des polygones et aussi de plaquer une texture.
- De même on peut basculer d'une représentation à une autre en fonction du niveau de détail et la distance du point de vue.

Pour justifier les inconvénients décrits ci-dessus nous allons détailler les problèmes qui ont été à l'origine de ceux-ci.

Comme nous l'avons mentionné auparavant, après l'étape d'acquisition, le nuage résultant peut contenir un bruit, ce qui conditionne la validité de l'étape de partitionnement du nuage en plan selon les axes vu qu'on exprime les estimations en fonctions des écarts de valeurs. Quand à l'élimination de ce bruit on l'a supposé appliquée en post-acquisition.

Les images issues des fichiers de nuages de points présentent beaucoup de zones moins échantillonnées, ce qui conduit dans la phase simplification à augmenter l'écart de cette zone. De ce fait, l'algorithme implémenté considère une discontinuité et le voisinage à calculer d'un point se limitera à la distance minimale des points restitués par la simplification. Cela aura bien évidement des conséquences fatales durant le processus d'extraction des triangles.

Les zones non couvertes par des triangles ne sont pas en générale des trous, mais par le fait que nous entassons des plans (plan principale et plan complémentaire).

Cet assemblage permet de générer des triangles dont les points du premier plan et le plan voisin sont tellement proche que les triangles ne couvrent pas régulièrement une zone.

Figure 56, Zones avec des petits triangles.

Comme nous l'avons constaté, le temps de la reconstruction est dans des cas très importants (voir tableau 3). Ceci est dû au nombre de points traité par plan avec un assemblage suffisant pour regrouper un nombre très élevé de points, mais en revanche quand on a réduit ce nombre à 5 plans le temps a considérablement chuté presque de la moitié.

5.3- Perspectives

Pour remédier à ces problèmes, nous allons décrire des propositions et aussi des améliorations à cette technique. Vu la contrainte du temps, nous n'avons pas pu réaliser cela mais nous nous estimons très satisfaits des résultats obtenus.

Le partitionnement d'un nuage de points n'est pas du tout un nouveau concept, il a été utilisé dans plusieurs techniques de reconstruction, ce concept permet en générale d'agir dans une zone limitée du nuage au lieu de traiter tout l'ensemble.

Par contre ce que nous avons proposé dans notre technique, un partitionnement naturel des points selon un critère d'estimation. Ce partitionnement offre un avantage majeur pour le choix des points à l'extraction des polygones. Afin d'améliorer cette approche, nous proposons d'étudier le partitionnement par maximisation. En d'autres termes, *trouver les plans Py qui maximise les plans Px*, ça devient donc un problème d'optimisation qui consiste à déterminer des optimums entre les plans *Px*, *Py* et *Pz*. Cela évitera de faire recours à l'assemblage des plans et aussi à pondérer les estimations. Nous pensons que cette proposition aura de bonne conséquence sur la technique développée et représente selon notre point de vue la pièce maîtresse de la reconstruction.

Le passage d'une représentation à base de points à une représentation polygonale ouvre la porte vers d'autres perspectives. L'exploitation des polygones en tant que surface issue de point est un atout majeur surtout dans le besoin d'améliorer la qualité du rendu en appliquant la technique de rendu par le lancer de rayon ou le

placage de textures et aussi dans le cadre de représentation en multi résolution. Pour ce dernier cas, nous proposons d'enrichir la technique tout en intégrant un pipeline dédié au rendu de nuage de point quand l'objet est éloigné de l'observateur et aussi un algorithme de subdivision quand l'objet est très proche de l'observateur ce qui exige un bon niveau de détail et cela permettra d'éviter la simplification des triangles qui nécessite un temps très considérable.

CHAPITRE VI

CONCLUSION GENERALE

Conclusion

La modélisation de surfaces sur la base d'une représentation à base de points est un domaine en pleine expansion, l'un des facteurs qui a suscité son développement étant l'apparition de périphériques d'acquisitions qui permettent de numériser en quelques secondes un objet sous forme de nuage de points. Cet avantage permet donc d'économiser du temps par rapport à la modélisation géométrique, malgré que les techniques de rendu à base de points ne soient pas implémentées directement sur les GPU. En outre, la reconstruction de surfaces issues de points est le paradigme qui a le plus permis le rapprochement entre deux domaines, différents par leur concept et fortement liés à la primitive manipulée durant la phase de modélisation ou de rendu.

Cependant plusieurs technique ont été développé par les chercheurs et cela remonte à trois décennies ; classé ces techniques ou menée une étude comparative est presque impossible du fait que ces techniques dépendent de plusieurs facteurs (selon les champs d'application), et pour les critères de comparaisons nous ne retiendront que trois parmi d'autre: le premier est l'exploitation directe du résultat par le GPU, le second la qualité du rendu et en dernier le temps d'exécution. Cela démontre aussi l'intérêt des industrielles qui se focalisent essentiellement sur ces trois critères. Un autre critère qui n'a pas été très évoqué dans la bibliographie est celui de l'implémentation directe de la technique dans le GPU. Un sujet intéressant qui nous a motivés a développé une nouvelle technique comme un début dans ce parcours. Notre technique réalise un passage (par reconstruction) d'un nuage de points d'un objet à des surfaces polygonales (en particulier des triangles).

En premier lieu notre objectif s'est focaliser sur l'étude de ce que permet le rapprochement entre deux domaines, notre démarche est inspiré de la façon dont on procède pour numériser un objet c'est-à-dire suivre la trace de l'acquisition. Le principe de base de cette approche est de former des contours parallèles et d'extraire ensuite des polygones par correspondance des points entre plans. D'autres techniques

exploitent cette même démarche ce qui nous différencie c'est la manière dont nous procédons. Notre approche repose sur trois concepts fondamentaux : Le partitionnement et la simplification des points, le calcul du voisinage entre points d'un même plan et enfin l'extraction de polygones. Ces concepts sont fortement liés et peuvent être assimilé pour chacun à un étage dans un pipeline.

Pour le partitionnement des points, nous avons proposé deux alternatives basé toutes les deux sur l'écart naturel selon un axe entre les points avec une garantit de convergence ; une autre alternative que nous avons proposé est sujet de réflexion pour nos perspectives. La simplification après le partitionnement est la phase qui consiste à choisir un point parmi plusieurs selon un critère, cette étape a une influence importante sur l'approximation entre le nuage de points et l'objet reconstruit. En d'autre terme les choix que nous avons proposés sont valables selon le besoin d'allez plus en profondeur ou vers l'extérieur. Quand à l'étape du calcul de voisinage, nous avons choisi la distance euclidienne comme critère de mesure avec une condition c'est que cette distance ne doit pas dépasser le maximum des distances minimales entre les points d'un même plan. Cette condition est une garantit pour qu'un point n'ait pas un voisin au-delà des regroupements de point voisin. L'extraction des polygones et pour des raisons de simplification dans l'étape de rendu nous avons opté pour des triangles. Cette étape possède une forte dépendance avec les étapes qui la précède. Plus le partitionnement est très cohérent et plus l'algorithme d'extraction sera simple. Nous avons conçu un algorithme des correspondances entre points de plan parallèle qui traite plusieurs cas mais ça nécessite encore plus d'améliorations. C'est ainsi que nous avons conçu cette technique, pour la valider, nous avons traduit cela dans un programme développé dans l'environnement Delphi ce qui nous a permis de montrer que le résultat obtenu est exploitable directement par le GPU. L'une des caractéristiques majeures de notre approche est la simplification du nuage, un résultat qui se positionne à mi chemin de deux représentations et présente l'avantage d'effectuer un rendu en temps opportun et aussi l'amélioration de la qualité des images par placage de textures.

L'amélioration de la qualité des images par placage de textures n'a pas été traitée suffisamment par contrainte de temps et aussi nous nous sommes concentrés beaucoup plus sur l'étude de la faisabilité de notre approche, malgré cela nous avons présenté quelques exemples pour montrer que cela est possible.

Le choix de développer une nouvelle technique de reconstruction de surfaces était très bénéfique car cela nous a permis de nous confronter aux vrais problèmes du domaine qui reste encore en cours de développement et mérite aussi bien l'intention des chercheurs de la communauté graphique. Face à cette diversité de technique et la richesse du domaine nous avons mené une étude bibliographique très minutieuse, en revanche la rareté des échantillons de nuages de points issue de la numérisation ne fait que réduire la généralisation de notre technique cela est dû essentiellement à la non accessibilité à ces données, malgré cela nous sommes très satisfait des résultats obtenu lors du rendu. Par rapport à d'autres techniques nous avons l'avantage d'intégrer plusieurs alternatives dans chaque étape de reconstruction ceci est dû à un partage logique des différentes phases et offre un cadre idéale pour enrichir et améliorer cette technique.

Enfin, notre travail de magister nous ouvre la porte vers d'autres perspectives qui constituent une suite logique du domaine de la modélisation de surface issue de point que nous résumant par :

- L'amélioration du partitionnement par maximisation des plans,
- L'étude et l'application des méthodes de rendu avancé (lancer de rayon et radiosité)
- L'élargissement du champ d'application de cette technique dans divers domaine (médecine, archéologie).
- Une étude qui tente d'implémenter cette technique dans le GPU.

Pour conclure, nous estimons que notre travail fera l'objet de réflexion dans d'autre sujet de recherche telle que l'animation.

BIBLIOGRAPHIE

[ABCO+01] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. IEEE Visualization 2001, pages 21.28, October 2001. ISBN 0-7803-7200-x.

[ACK01] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In Proceedings of the sixth ACM symposium on Solid modeling and applications, pages 249.266. ACM Press, 2001.

[Alg95] Maria-Elena Algorri. Génération et simplification du maillage pour la reconstruction de surfaces de points non structurées. PhD thesis, ENST, 1995.

[Ama84] J. Amanatides: Ray tracing with cones. In Computer Graphics (SIGGRAPH'84), 18(3), 129-135, 1984.

[BC00] Jean-Daniel Boissonnat and Frédéric Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. In Proceedings of the sixteenth annual symposium on Computational geometry, pages 223.232. ACM Press, 2000.

[BI98] Andrew Blake and Michael Isard. Active Contours. Springer-Verlag, 1998.

[Blo88] Jules Bloomenthal. Polygonization of implicit surfaces. Computer Aided Geometric Design, 5 :341.355, 1988.

[Car84] L. Carpenter. The A-buffer, an Antialiased Hidden Surface Method. In Computer Graphics, volume 18 of SIGGRAPH ’84 Proceedings, pages 103–108. July 1984.

[Cat74] CATMULL, E. 1974. A Subdivision Algorithm for Computer Display of Curved Surfaces. PhD thesis, University of Utah.

[CSD02] David Cohen-Steiner and Frank Da. A greedy delaunay based surface reconstruction algorithm. Technical report, INRIA- Sophia Antipolis, 2002.

[DGH01] Tamal K. Dey, Joachim Giesen, and James Hudson. Delaunay based shape reconstruction from large data. In Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics, pages 19.27. IEEE Press, 2001.

[DQ01] Ye Duan and Hong Qin. Intelligent balloon: a subdivision-based deformable model for surface reconstruction of arbitrary topology. In Proceedings of the sixth ACM symposium on Solid modeling and applications, pages 47.58. ACM Press, 2001.

[GAC+89] A. S. Glassner, J. Arvo, R. L. Cook, E. Haines, P. Hanrahan, P. Heckbert, and D. B. Kirk. Introduction to Ray Tracing. Academic Press, 1989. 24

[GK93] Ned Greene, Michael Kass, Gavin Miller, "Hierarchical Z-buffer Visibility", Proc. SIGGRAPH '93 Proceedings, pp. 231-238

[GK00] M. Gopi and S. Krishnan. A fast and efficient projection based approach for surface reconstruction, 2000.

[GKS00] M. Gopi, S. Krishnan, and C.T. Silva. Surface reconstruction based on lower dimensional localized Delaunay triangulation. In Proceedings of Eurographics, volume 19, 2000.

[Gros06] Markus Gross, Getting to the Point…? Computer Graphics Laboratory ETH Zürich, Switzerland; appeared in "Graphically Speaking", IEEE Computer Graphics and Applications September/October 2006 (vol. 26 no. 5) pp. 96-99

[HA90] Paul Haeberli and Kurt Akeley. The accumulation buffer: Hardware support for high-quality rendering. In Forest Baskett, editor, Computer Graphics (SIGGRAPH '90 Proceedings), volume 24(4), pages 309–318, August 1990. (cited on page 59)

[HH84] P. Heckbert, P. Hanrahan: Beam tracing polygonal objects. Computer Graphics (SIGGRAPH'84), 18(3), pp. 119-127, 1984.

[Hop96] Hugues Hoppe. Progressive meshes. In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pages 99.108. ACM Press, 1996.

[Hop97] Hugues Hoppe. View-dependent refinement of progressive meshes. In Proceedings of the 24th annual conference on Computer graphics and interactive techniques, pages 189.198. ACM Press/Addison-Wesley Publishing Co., 1997.

[JW89] David Jevans and Brian Wyvill. Adaptive voxel subdivision for ray tracing. In Proceedings of Graphics Interface '89, pages 164–172, June 1989. (cited on page 40)

[KJ03] Jaroslav Krivanek. Representing and rendering surfaces with points. Technical Report DC-PSR-2003-03, Department of Computer Science and Engineering, Czech Technical University in Prague, 2003.

[KWT87] M. Kass, A. Witkin, and D. Terzopoulos. Snakes : Active contour models. Computer Vision, 1 :321.331, 1987.

[LC87] W.E. Lorensen and H.E. Cline. Marching cubes : a high resolution 3d surface construction algorithm. Computer Graphics, 21 :163.169, 1987.

[LW85] M. Levoy and T. Whitted. The use of points as a display primitive. Tech. Report 85-022,1985.

[PD04] Damien Porquet, Rendu en temps réel de scènes complexes, Thèse de doctorat Nov 2004 pages 9-15, université Limoges, France.

[PZvBG00] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In Proc. of Siggraph'2000, 2000.

[RA06] Contributions à l'introduction de flexibilité dans la reconstruction et l'édition de modèles 3D thèse de doctorats université Claude Bernard Lyon 1 Dec 2006.

[RL00] S. Rusinkiewicz and M. Levoy. Qsplat : A multiresolution point rendering system for large meshes. In Proc. of Siggraph'2000, 2000.

[SB87] John M. Snyder and Alan H. Barr. Ray tracing complex models containing surface tessellations. In Maureen C. Stone, editor, Computer Graphics (SIGGRAPH '87 Proceedings), volume 21(4), pages 119–128, July 1987. (cited on page 40)

[SC92] P. Strauss and R. Carey. An object-oriented 3d graphics toolkit. In Proc. of Siggraph'92, pages 341–349, 1992.

[SZL92] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In Edwin E. Catmull, editor, Proceedings of the 19th Annual ACM Conference on Computer Graphics and Interactive Techniques, pages 65.70, New York, NY, USA, July 1992. ACM Press

[TB04] Tamy Boubekeur ; Reconstruction de surface à l'aide de surfaces de subdivision. Master Recherche MM - Juin 2004 université Bordeaux.

[TGS00] I. Tobor, L. Grisoni, and C. Schlick. Rendering by surfels. In Proc. of Graphicon'2000, 2000.

[Thal03] D.Thalman Informatique graphique EPFL-VLAB, pages 28,29,30. 2003

[TO02] Greg Turk and James F. O'Brien. Implicit surfaces that interpolate. In SMI, 2002.

[TRS04] Ireneusz Tobor, Patrick Reuter, and Christophe Schlick. Efficient reconstruction of large scattered geometric datasets using the partition of unity and radial basis functions. In WSCG (Winter School of Computer Graphics), feb 2004.

[Whi80] Turner Whitted. An improved illumination model for shaded display. Communications

of the ACM, 23(6) :343.349, June 1980. 24, 35

[ZS00] Denis Zorin and Peter Schröder. Subdivision for modeling and animation. In SIGGRAPH Courses Notes, 2000.