DEA IARFA 2002-2003

Rapport de Stage

"Utilisation de cartes de profondeurs pour la reconstruction 3D"

ROMAIN DUPONT

Responsables de stage : Philippe Fuchs et Julian Aubourg
Dates du stage : du 7 Avril au 8 Septembre 2003

Remerciements

Je remercie tout d'abord Claude Laurgeau pour m'avoir accueilli au Centre de Robotique.

Je remercie particulièrement mes deux encadrants, Julian et Philippe, qui m'ont accordé leur confiance et prodigué leurs nombreux et précieux conseils.

Un grand merci aussi à François et à Domitile pour leur aide sur mes travaux de stage et les discussions autour de mon avenir professionel.

Je salue tous les thésards et stagiaires du laboratoire qui ont maintenu une bonne ambiance au sein du laboratoire, ainsi que les joueurs de guitare, Alain et Samuel, et de didjeridoo, Konaly, qui a réussi à obtenir la vibration !

Introduction

Contexte général

On constate un intérêt croissant pour la digitalisation de notre patrimoine culturel, non seulement pour prévenir sa détérioration inéluctable, mais également pour permettre son accès au public et aux chercheurs. Plusieurs techniques existent à l'heure actuelle mais restent couteuses et difficiles à mettre en place.

Le projet GANTOM [1] vise à développer une technique de digitalisation des objets muséographiques par vision passive. Celle-ci est peu couteuse et peu contraignante, notamment parce qu'elle s'appuie sur l'utilisation de photographies numériques dont la qualité et la résolution progresse de jour en jour.

Une chaîne complète de traitements a été développé au Centre de Robotique de l'Ecole des Mines de Paris, dans le cadre du projet GANTOM. Elle couvre l'acquisition photographique de l'objet, la reconstruction géométrique et la texturation,réalisée par le partenaire SIMTEAM.

Problématique

C'est la reconstruction géométrique de l'objet qui est au centre de ce stage. Elle est en soi un problème difficile car paradoxal : on ne peut reconstruire exactement l'objet que si sa reconstruction exacte est disponible ! Les ambiguïtés topologiques et géométriques ne manquent pas et il est impossible d'atteindre une qualité parfaite sans assistance humaine.

La reconstruction géométrique s'opère en 3 étapes :
- création de cartes de profondeurs par stéréovision
- calcul du modèle volumique formé de voxels binaires grâce à un nouvel algorithme de fusion de cartes de profondeurs, le Shape From Depth [1]
- extraction d'une surface triangulée.

Le modèle surfacique obtenu souffre d'un défaut majeur : la surface est bosselée.

Démarche

Pour trouver l'origine du problème, la chaîne de traitements a été étudiée maillon par maillon. Les données d'entrée de la chaîne sont les cartes de profondeurs. Celles-ci sont bruitées et comportent des erreurs locales. Afin d'étudier l'influence de leurs défauts sur le restant de la chaîne, un générateur de cartes synthétiques a été développé. En présence de données synthétiques parfaites, la surface extraite du modèle volumique restait bosselée. L'algorithme d'extraction de

[1] Génération Automatique de Numérisation et de Texturation d'Objets Muséographiques

[1] Un voxel est un cube avec une densité comprise entre 0 et 1. Si le voxel est binaire, sa densité vaut 0 ou 1 ; il est ainsi soit plein, soit vide.

surface gérait mal les données binaires fournies par le Shape From Depth et créait ainsi un effet d'aliasing[2].

La première étape fut de retravailler en profondeur l'algorithme du Shape From Depth. Il s'est en effet avéré que son implémentation comportait de nombreux bugs et que la précision du modèle volumique était inférieure à celle des cartes de profondeurs. La seconde étape fut de développer un système d'anti-aliasing. Plusieurs techniques de lissage pour réduire l'effet de discrétisation ont été développées. La dernière étape fut d'améliorer les techniques de triangulation de la surface, celles-ci ne respectant pas toujours la topologie de l'objet. Nous verrons qu'il n'y a pas une technique de triangulation meilleure qu'une autre et que tout dépend de la nature de l'objet.

Plan du rapport

Le chapitre 1 expose le cadre du stage, le projet GANTOM, et présente la problématique.

Le plan du rapport suit alors la chaîne de traitements.

- Le chapitre 2 présente la technique de lissage des cartes de profondeurs.
- Le chapitre 3 présente le Shape From Depth et les améliorations apportées.
- Le chapitre 4 présente les différentes techniques d'anti-aliasing qui corrige l'effet d'escalier.
- Le chapitre 5 s'intéresse à l'extraction de la surface du modèle volumique et expose différentes techniques de triangulation, ainsi que leur cadre d'utilisation.

Le dernier chapitre dressera la synthèse et les perspectives.

[2]L'aliasing, ou crênelage, est un effet d'escalier disgracieux dû à la nature discrète des données.

Table des matières

Chapitre 1

Cadre du stage et problématique

1.1 Présentation du projet GANTOM

Le projet GANTOM [4] a pour objectif la reconstruction géométrique texturée d'objets muséographiques de petites tailles[1]. Une chaîne complète de numérisation a été développée au sein du Centre de Robotique de l'Ecole des Mines de Paris et en partenariat avec l'entreprise SIMTEAM. Elle va des prises de vues numériques à la texturation de l'objet 3D reconstruit.

La première partie de la chaîne (acquisition photographique et extraction de la profondeur) est détaillée en annexe A. La texturation ne sera pas présentée dans ce rapport.

1.2 Reconstruction géométrique

Ce stage est centré sur la deuxième partie de la chaîne de traitements : la reconstruction géométrique. La figure 1.1 en illustre les différents maillons.

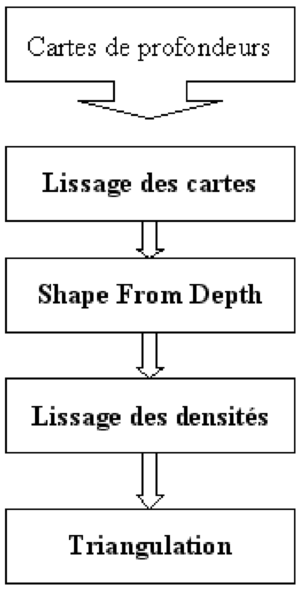


FIG. 1.1 – Deuxième partie de la chaîne de traitements du projet GANTOM (hors texturation)

[1]qui rentre dans un cube de 20/30 centimètres d'arête en général

1.2.1 Cartes de profondeurs

Les cartes de profondeurs sont les données d'entrée de la chaîne de reconstruction géométrique (figure 1.2).

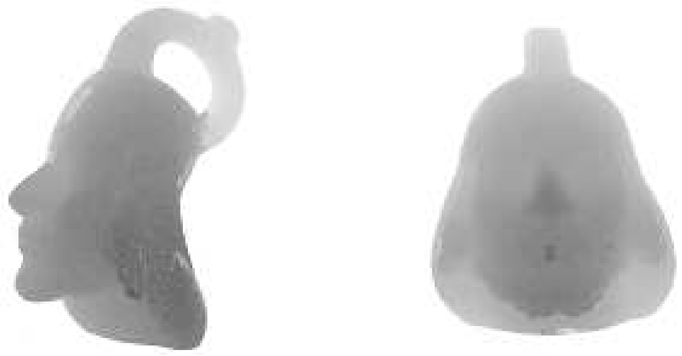


FIG. 1.2 – Cartes de profondeurs (une couleur claire indique une profondeur élévée)

Dans le cadre du projet GANTOM, celles-ci sont calculées par stéréoscopie mais pourraient aussi être obtenues par d'autres techniques, comme la télémétrie laser.

1.2.2 Shape From Depth et notion de densité

L'algorithme « Shape From Depth » [4] construit un modèle volumique de l'objet en fusionnant les cartes de profondeurs[1]. Le modèle obtenu est un ensemble de voxels binaires.

Un voxel est un cube qui a une densité comprise entre 0 et 1. Les voxels qui ont une densité inférieure à un seuil, généralement fixé à 0.5, sont à l'intérieur de l'objet, les autres à l'extérieur. Si le voxel est binaire, sa densité vaut 0 ou 1 [2]. Nous verrons que la notion de densité des voxels sera au coeur des techniques d'extraction de surface.

1.2.3 Passage du modèle volumétrique au modèle surfacique

L'extraction de la surface s'effectue par l'algorithme « Marching Cube » [16]. Il triangule la surface du modèle volumique en faisant appel à une table précalculée (détails au chapitre 5).

1.3 Les problèmes

Les techniques de reconstruction volumétrique utilisées ne donnent pas de résultats satisfaisants. La figure 1.3 illustre le problème : la surface de l'objet est bosselée.

[1]cet algorithme sera, par la suite, appelé OSFD (Octree-based Shape From Depth)

[2]On rencontre généralement des densités de voxels continues dans le domaine médical : les données fournies par les scanners sont des voxels dont la densité est liée aux propriétés phyisques des tissus organiques.

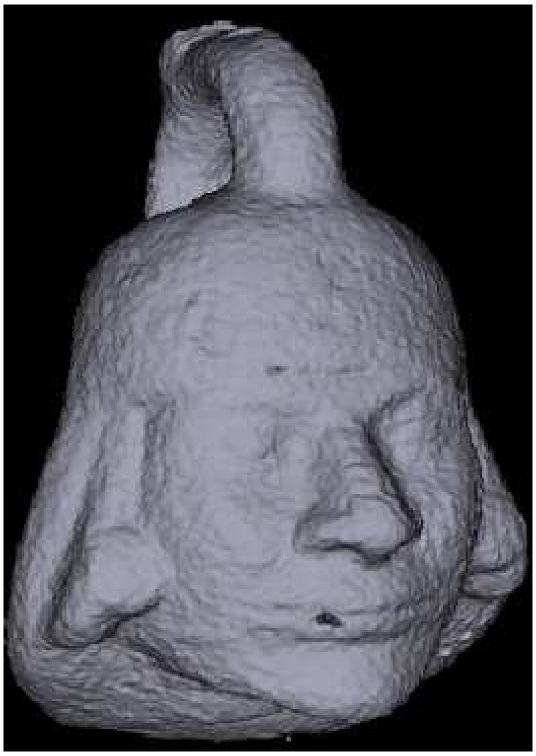


FIG. 1.3 – Masque reconstruit : la surface est bosselée

Plusieurs maillons de la chaîne peuvent en être à l'origine :
- les cartes de profondeurs
- le Shape From Depth
- l'extraction de la surface par Marching Cube

1.3.1 Cartes de profondeurs

Les cartes de profondeurs sont obtenues par stéréovision ; elles sont bruitées et comportent de nombreux artefacts. Pour mieux identifier l'origine du problème et analyser l'influence du bruit, un générateur de cartes de profondeurs synthétiques a été développé (détails à l'annexe D).

Le travail s'est alors porté sur des cartes synthétiques, représentant un cube avec une sphère, une pyramide et des détails divers (3 petits cubes par exemple). Si on reconstruit à partir de ces données non bruitées, on obtient un effet bosselé atténué (figure 1.4). Le bruit des cartes réelles n'est donc pas le seul problème. Notons qu'un système de lissage a tout de même été développé. Il est présenté au chapitre 2.

1.3.2 Problème du Shape From Depth

Le modèle volumique généré par le Shape From Depth ne comporte pas d'artefact particulier mais il est formé de voxels binaires. Nous verrons dans la sous-section 1.3.3 que c'est l'extraction de la surface qui est à l'origine du problème. Celle-ci n'est pas adaptée aux informations binaires fournies par l'OSFD.

Cependant, il s'est rapidement avéré que l'implémentation du Shape from Depth était in-

FIG. 1.4 – Surface bosselée (la triangulation est effectuée par le Marching Cube)

consistante. Sa gestion des voxels de taille sub-pixelliques était mauvaise ; le modèle volumique obtenu se décomposait (cf. figure 1.5). De surcroît, la précision du modèle n'atteignait pas celle des cartes de profondeurs. Le Shape From Depth a donc été retravaillé pour le rendre fonctionnel, et surtout, plus précis. Le chapitre 3 présentera les améliorations apportées.

1.3.3 Aliasing

Ce que l'on observe sur la figure 1.4 n'est en fait rien d'autre qu'un effet d'aliasing. Les données fournies par l'OSFD sont binaires. La discrétisation se traduit par une perte d'information ; on observe alors un effet de crénélage dit aliasing[1]. La figure 1.6 illustre ce phénomène en 2D.

On retrouve le même problème en 3D, en partie corrigée par le Marching Cube. Malheureusement, seules les zones plates sont correctement triangulées, les zones courbes sont quant à elles, bosselées. Le chapitre 4 présentera les techniques de lissage développées pour supprimer cet aliasing.

1.3.4 Triangulation de la surface

Notre attention s'est ensuite portée sur un autre problème : le Marching Cube introduit des artefacts géométriques. Il se révèle incapable de bien trianguler les bords tranchants de l'objet, ceux-ci étant systématiquement coupés en biseau. La figure 1.7 illustre ce phénomène.

Le chapitre 5 exposera en détail les causes et des solutions apportées pour y rémédier.

[1]phénomène bien connu en traitement du signal qui survient lorsqu'on discrétise un signal continu

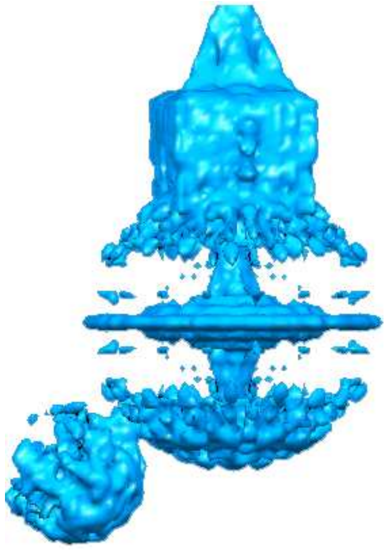


FIG. 1.5 – Lorsque la taille des voxels projetés sur les cartes de profondeurs avoisine la taille d'un pixel, le volume se décompose

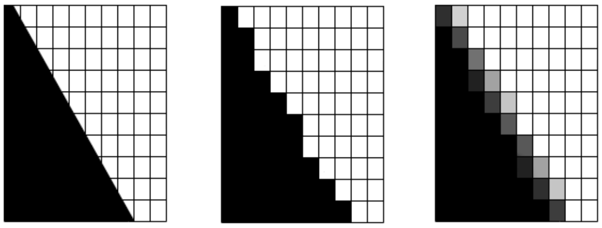


FIG. 1.6 – A gauche : pente initiale. Au milieu, un exemple d'aliasing après binarisation. A droite : un exemple d'anti-aliasing qui atténue l'effet escalier

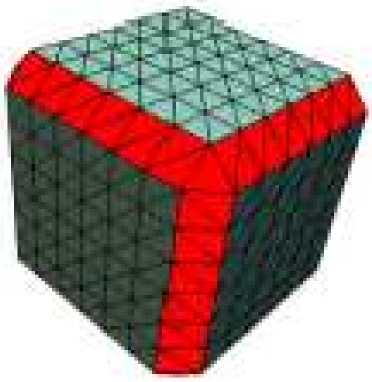
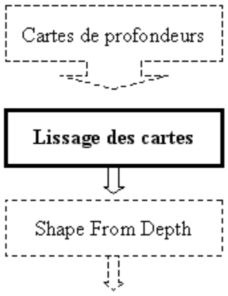


FIG. 1.7 – Arêtes d'un cube coupés en biseau

Chapitre 2

Lissage des cartes de profondeurs



Nous avons vu en sous-section 1.2.1 que les cartes de profondeurs sont bruitées. Un lissage gaussien peut être appliqué mais celui-ci estompe les détails. Ce chapitre présente une nouvelle technique : le lissage anisotropique.

2.1 Principe général du lissage anisotropique

Lorsqu'on filtre une image avec un lissage gaussien, le bruit est estompé mais les détails disparaissent par la même occasion : les bords deviennent flous. La solution serait alors de limiter le lissage aux zones qui comportent peu de détails. Le lissage anisotropique [21] répond justement à cette problématique : c'est un lissage non uniforme, son intensité dépendant d'un critère local de complexité. On pourra se référer à l'annexe F pour la bibliographie commentée faite sur ce sujet.

En 2D, c'est l'information de gradient qui est généralement utilisée pour définir les zones à lisser. En effet, les zones avec beaucoup de détails (les bords par exemple) ont un gradient assez important (en norme) tandis que les zones globalement uniformes ont un faible gradient. Une fonction décroissante avec le gradient régule alors l'intensité du lissage : lorsque le gradient est élevé, le lissage est stoppé et inversement.

La fonction de blocage peut être :

- une fonction seuil (si le gradient est supérieur à une valeur S, le lissage est bloqué),
- une fonction décroissante, comme par exemple :

$$f(x) = 1 - \exp(-\frac{\beta}{\|x\|})$$

ou

$$f(x) = \exp(-\left(\frac{\|x\|}{\beta}\right)^2)$$

ou

$$f(x) = \frac{1}{1 + \left(\frac{\|x\|}{\beta}\right)^2}$$

avec β un paramètre de blocage défini par l'utilisateur et x, la valeur du gradient (les 2 dernières fonctions sont proposées par P. Perona et J. Malik [21])

Une technique améliorée de lissage anisotropique 2D (conçue par B. Migeon et V.Serfaty [18]) a été implémentée. Cette technique offre des contours moins crénélés par rapport à un lissage anisotropique classique (cf. l'article [18] pour les comparatifs). Les 3 images ci-dessous montrent les effets du lissage anisotropique sur une image 2D où l'on voit que le bruit a été éliminé sans que les détails et les bordures disparaissent.

Image originale (bruitée)

Après un lissage isotropique (gaussien)

Après un lissage anisotropique

Cette technique de lissage anisotropique a été utilisée pour lisser les cartes de profondeurs et ainsi, réduire le bruit issu de la stéréovision.

2.2 Lissage des cartes de profondeurs

La figure 2.1 illustre les modifications apportées par le lissage anisotropique sur les cartes de profondeurs. On génère un modèle volumique avec les nouvelles cartes. La figure 2.2 permet de comparer les résultats obtenus avec et sans lissage. La surface est beaucoup plus régulière, les gros défauts disparaissent (cf. les escaliers sur la joue gauche). Certains détails comme la fissure du front commencent à s'estomper, d'autres restent bien visibles (bouton sur l'oreille gauche).

FIG. 2.1 – A gauche : carte de profondeur non lissée. A droite : la même carte, mais lissée anisotropiquement

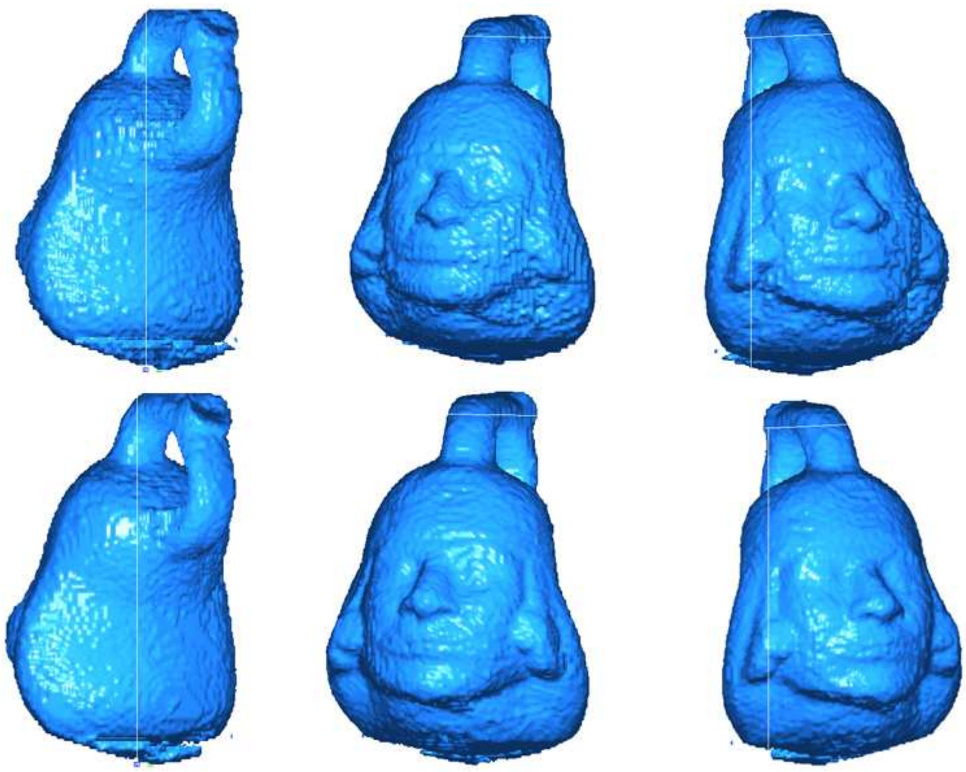
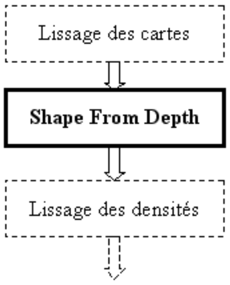


FIG. 2.2 – En haut, diverses vues du masque reconstruit avec les cartes brutes. En bas, la même chose mais les cartes ont été lissées anisotropiquement (les rayures blanches que l'on aperçoit sont des repères utilisées lors de l'analyse des résultats)

Chapitre 3

Le Shape From Depth



3.1 Fonctionnement

Le principe de fonctionnement du Shape-From-Depth [4] s'inspire de celui du "Shape From Silhouette", qui est présenté ci-dessous.

3.1.1 Shape From Silhouette

Le Shape-From-Silhouette est une technique simple de numérisation 3D d'objets à partir de leurs silhouettes. Ces silhouettes sont obtenues à partir d'acquisitions photographiques qui sont binarisées (intérieur de l'objet $\leftrightarrow$ extérieur de l'objet). Pour chacune de ces silhouettes, on calcule le cône dont la base est la silhouette et le sommet est le centre optique de la caméra. L'intersection de tous ces cônes donne le modèle volumique.

Ses deux principaux défauts que l'on peut retenir :

- le modèle volumique qu'on obtient est l'enveloppe convexe de l'objet : certaines parties concaves ne sont pas correctement supprimées lors de l'intersection des cônes.
- la qualité de la numérisation dépend entièrement du nombre de silhouettes en entrée.

3.1.2 Shape From Depth

La technique Shape From Depth (ou OSFD) reprend le principe du Shape From Silhouette mais ne s'appuie non pas sur des silhouettes, mais sur des cartes de profondeurs. Les cônes obtenus sont les mêmes qu'avec les silhouettes, sauf que la partie entre le centre optique de la caméra et la surface de l'objet est supprimée (partie blanche des cônes sur la figure 3.1). La figure 3.1 montre les principales étapes de la sculpture par OSFD. L'OSFD permet ainsi de sculpter

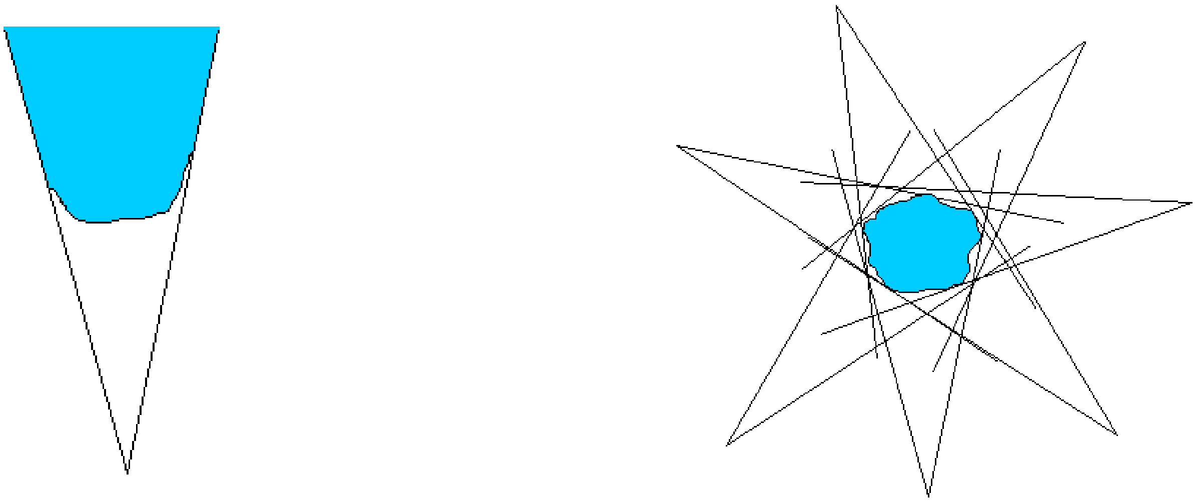


FIG. 3.1 – A gauche : volume obtenu à partir de la 1ère carte de profondeur. A droite : résultat de la fusion de toutes les cartes.

toutes les concavités des objets. Le Shape From Silhouette est en fait un cas particulier de l'OSFD où les cartes de profondeurs seraient binaires : profondeurs à l'infini hors de la silhouette, à 0 dans la silhouette.

3.2 Amélioration de la précision

La précision du modèle volumique obtenu était inférieure à celle des cartes de profondeurs. Dès que la taille des voxels projetés atteignait la précision des cartes de profondeurs, la version initiale du Shape From Depth considérait qu'il n'y avait plus d'information exploitable. Or ceci est faux car les voxels ne sont pratiquement jamais alignés sur les pixels des cartes de profondeurs. Ils sont en général à cheval sur plusieurs pixels (figure 3.2) et il serait intéressant de resubdiviser les voxels pour approcher au mieux les cartes de profondeurs.

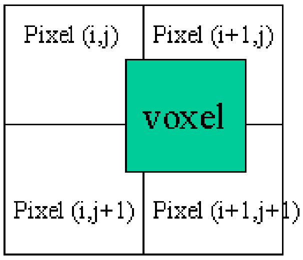


FIG. 3.2 – Le voxel, projeté sur la carte de profondeur, est à cheval sur plusieurs pixels.

3.3 Interpolation des cartes de profondeurs

Les cartes de profondeurs sont définies au pixel près. On a une information discontinue de la profondeur de l'objet (figure 3.3) qui se traduit par un effet escalier sur le modèle volumique. Les valeurs de profondeurs sont donc interpolées linéairement entre chaque pixel afin d'avoir

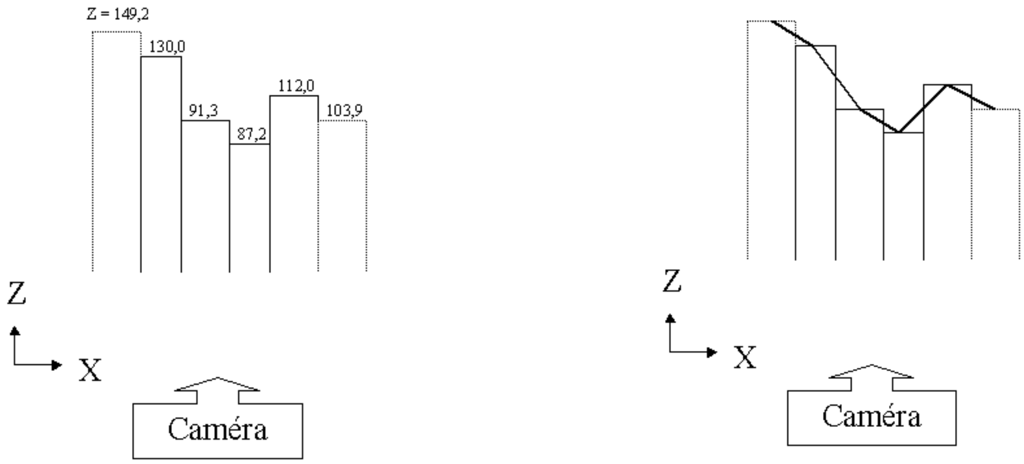


FIG. 3.3 – A gauche : effet d'escalier des cartes de profondeurs. A droite : interpolation linéaire selon l'axe des X.

une continuité par morceaux (figure 3.3). On peut utiliser d'autres types d'interpolation pour améliorer la continuité et le lissage (interpolation d'ordre 2 ou 3 par exemple) (figure 3.4) mais au prix d'une explosion du temps de calculs.

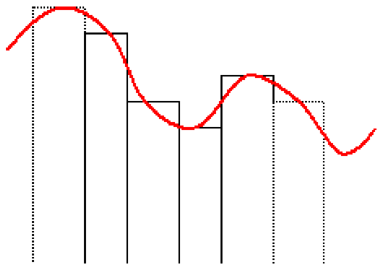
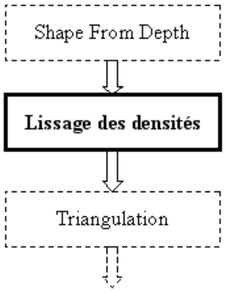


FIG. 3.4 – Interpolation d'ordre supérieur des cartes de profondeurs

Chapitre 4

Anti-aliasing : Lissage des densités



Pour rémédier au problème d'aliasing vu en section 1.3.3, plusieurs solutions existent :
- lisser les densités des voxels à la surface de l'objet.
- sur-échantilloner les données puis réduire la résolution par moyennage. Il existe plusieurs techniques de moyennage : moyenne gaussienne ou moyenne par réduction d'Octree. Cette dernière sera présentée à la section 4.2. Le sur-échantillonnage se réalise ici en effectuant plus finement la sculpture par l'OSFD.

Notons que les techniques d'anti-aliasing développées filtrent par la même occasion le bruit présent dans le modèle volumique.

4.1 Diffusion isotropique des densités

La première technique d'anti-aliasing qui a été mise en place est la diffusion isotropique des densités. C'est un simple lissage type Gaussien, uniforme qui, pour chaque voxel, calcule une moyenne des densités des voxels voisins. Au fur et à mesure des itérations, on obtient une information de densité plus continue.

A la manière d'un lissage Gaussien, cette technique de calcul des densités réduit l'aliasing, la surface est uniformément lisse. En contrepartie, les détails de l'objet disparaissent, les bords tranchants sont arrondis. Cette technique, visuellement acceptable à grande échelle, ne l'est plus si l'on s'intéresse aux détails. On perd de l'information, comme on peut le constater sur la figure 4.1 où la fissure du front disparait.

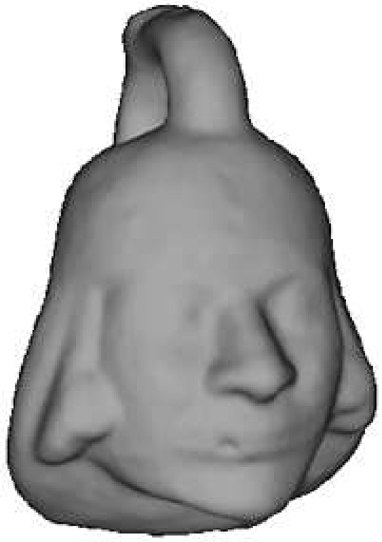


FIG. 4.1 – Le masque lissé isotropiquement : les détails sont estompés

4.2 Lissage de densité par réduction de niveau de l'Octree

La technique présentée ici, s'apparente à la technique d'anti-aliasing dite «over-sampling». Elle consiste à échantilloner les données à une résolution supérieure pour ensuite réduire la résolution en effectuant un moyennage.

4.2.1 Principe général

Les voxels du modèle volumique sont stockés dans un Octree. Cette structure est utilisée pour lisser les densités des voxels [1]. Le principe du lissage développé est de réduire le nombre de niveaux de l'Octree (on obtient alors des cubes de taille supérieure) tout en récupérant une information de densité selon le principe suivant : lorsqu'on passe du niveau N au niveau N-1, les cubes doublent en taille mais au lieu d'être binaires, ils acquièrent l'information de densité des 8 cubes fils par moyennage (figure 4.2).

[1] L'idée d'utiliser la structure en Octree des voxels n'est pas nouvelle : l'article [24] propose une décimation qui s'appuie sur l'Octree

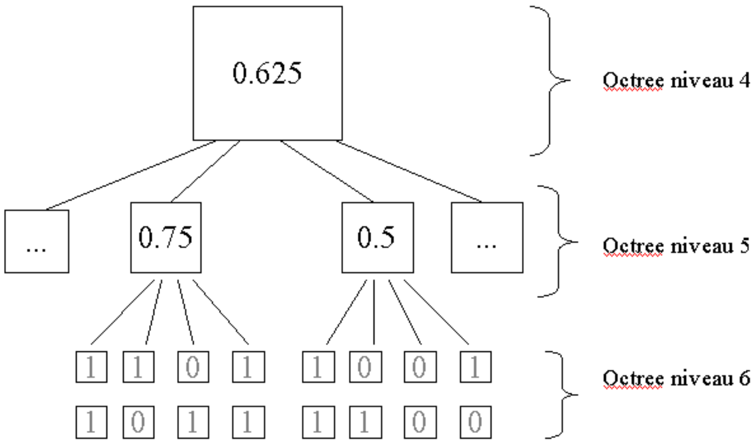


FIG. 4.2 – Calcul de densité (seuls 4 fils sont représentés)

La quantité d'information par voxel est multipliée par 8 à chaque réduction de niveau. On peut ainsi voir ce processus de lissage comme une conversion d'information : on passe de voxels binaires en voxels avec information de densité. Cette information sera directement mis à profit par le Marching Cube.

4.2.2 Résultats

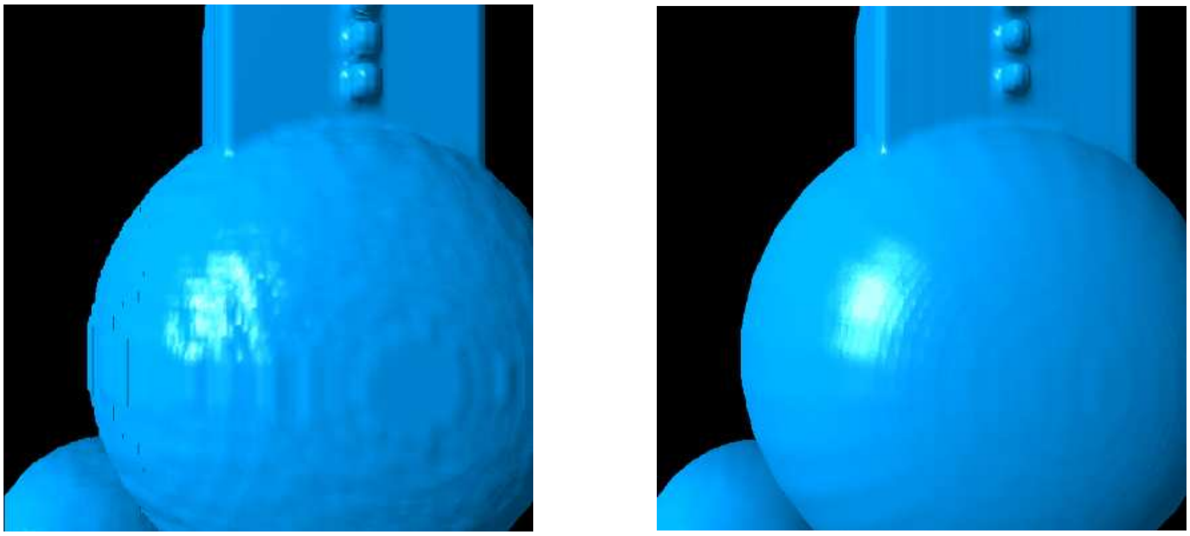


FIG. 4.3 – A gauche : le modèle volumique initial (aliasing). A droite : anti-aliasing

Sur les données synthétiques, cette technique d'anti-aliasing améliore nettement les résultats (figure 4.3). Cette technique a été appliquée sur les cartes de profondeurs réelles. On voit déjà une amélioration (figure 4.4) par rapport au modèle initial (figure 1.3) : la fissure du front reste visible, même après un lissage isotropique qui a été appliqué pour parfaire le lissage.

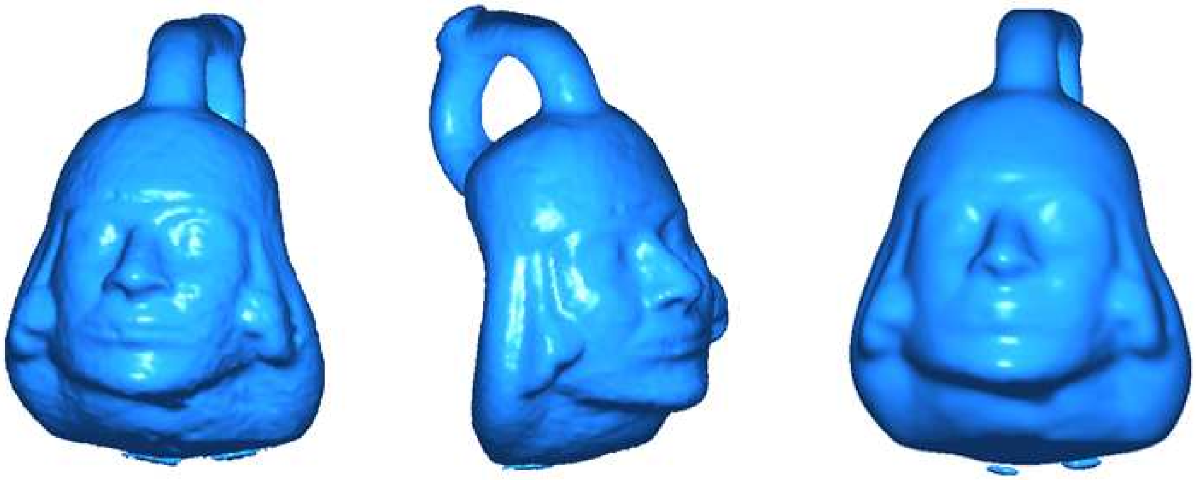


FIG. 4.4 – A gauche et au milieu : résultat de l'anti-aliasing (limité ici). A droite, on a rajouté 2 itérations de lissage isotropique pour parfaire le lissage de la surface

Exploitant la structure de l'Octree, cette technique est très rapide (quasi instantanée) et économe en mémoire-vive [1]. Cependant, l'OSFD doit améliorer la précision du modèle volumique, ce qui se traduit par un surcoût en temps de calculs.

4.3 Lissage anisotropique 3D des densités

En chapitre 2, une technique de lissage qui s'appuie sur une diffusion anisotropique a été présentée. Cette idée de lissage adaptatif a été transposée en 3D : le but est en effet de lisser les zones bruitées tout en conservant les zones tranchantes. Plusieurs articles ont été publiés à ce sujet, on pourra se référer à la bibliographie commentée (annexe F).

Lisser revient à faire une moyenne pondérée du voisinage. La façon dont ces coefficients de pondération sont calculées va être au coeur du principe. Si ces coefficients sont faibles, on lisse peu et vice-versa. La technique de lissage adaptatif développée s'appuie sur le principe suivant : l'intensité du lissage va dépendre d'un indice de complexité locale de l'objet. Cet indice sera maximal sur les bords tranchants et minimal sur les zones plates. Nous verrons un peu plus loin que différentes mesures de cet indice ont été étudiées. La figure 4.5 en donne un aperçu où l'on peut remarquer qu'une forte complexité locale peut s'exprimer par une forte variation des normales.

[1]Les modèles volumiques atteignent en effet plusieurs centaines de Méga-Octet voire le Giga-Octet

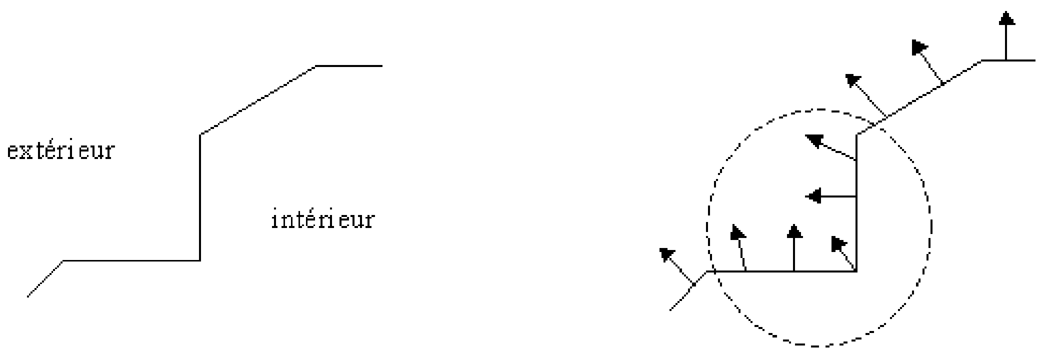


Fig. 4.5 – A gauche : la surface de l'objet (vue latérale). A droite : les normales (forte variation au niveau du cercle pointillé)

4.3.1 L'algorithme

```
REPETER x FOIS (x est le nombre d'itérations défini par l'utilisateur)

   Calculer les normales de tous les voxels à partir des densités

   POUR chaque voxel V FAIRE
       - Calculer l'indice de complexité locale à partir du voisinage de V
       - Modifier la densité du voxel V en effectuant une moyenne des densités des
       voxels voisins, plus ou moins pondérée selon l'indice de complexité locale.

   FIN POUR
FIN REPETER
```

4.3.2 Indices de complexité locale

Divers estimateurs de la complexité locale ont été étudiés.

la norme du gradient Ψ des densités estimé localement

On transpose ici en 3D une technique généralement utilisée en 2D : l'indice de complexité locale s'appuie sur la norme du gradient (qui est en fait ici la normale). Celui-ci peut être estimé par différence centrale ou en utilisant une technique plus robuste proposée en annexe E. Mais cette technique n'est pas du tout adaptée si les voxels sont binaires ou faiblement échantillonés. Il arrive en effet fréquemment que les voxels appartenant à la surface passe brutalement d'une densité 1 à une densité 0 (au milieu d'une face bien alignée sur la grille par exemple, donc « non complexe »), ce qui se traduit ici par un fort gradient. Et cette situation est systématique si les voxels sont binaires.

la moyenne des angles entre la normale centrale et ses voisines

On peut s'appuyer non pas sur la norme du gradient (ou normale), mais sur l'angle entre les normales. La figure 4.5 illustre ce principe. Une zone complexe peut ainsi se définir comme étant une zone où les normales varient beaucoup localement en angle. Un estimateur de complexité locale peut ainsi être la moyenne des angles entre la normale du voxel central et les normales des voxels voisins. On calcule l'angle entre 2 vecteurs très simplement avec leur produit scalaire. Cette technique ne s'avère efficace que si les données sont lissées et s'il n'y a pas de bords tranchants. Malheureusement, ce n'est pas le cas. On a vu en début de chapitre que les données fournies par l'OSFD sont binaires.

Fig. 4.6 – A gauche : bord d'un objet, les voxels pleins son représentés en gris. A droite : la somme des différences des angles avec le voisinage immédiat

Les valeurs sur la figure 4.6 sont obtenues en additionnant les différences d'angles entre la normale centrale (celle du voxel considéré) et les normales des voxels voisins :
- la valeur 90° est obtenue en additionnant 45° (angle avec le voxel de gauche) et 45° (angle avec le voxel du dessous)
- la valeur 135° (en haut à gauche) est obtenue en additionnant 45° (angle avec le voxel de droite) et 90° (angle avec le voxel en bas à droite, dans la diagonale)
- la valeur 0° est obtenue lorsqu'il n'y a pas de variation d'angles avec le voisinage.

Ces valeurs montrent que ce n'est pas le voxel de la pointe qui est considéré comme étant le plus "complexe" mais ses voxels voisins qui sont pourtant plus éloignés du bord. Il faut donc adjoindre un système qui détecte ces fortes irrégularités pour en réadapter le calcul.

l'angle maximal ou minimal de ces angles

Plutôt que la moyenne, selon les besoins du programme, on peut utiliser le minimum ou le maximum. Cette idée est utilisée dans [14] pour déterminer les zones tranchantes et adapter la technique de triangulation à ces zones (le minimum des angles sert à détecter les bords tranchants et le maximum des angles, les pointes de l'objet).

la courbure de la surface

La complexité locale peut s'exprimer à travers la notion de courbure de la surface ([22] , [26] et [5]). La courbure d'une surface est une fonction continue et n'a de sens que si les données sont continues ou suffisament échantillonées (hypothèse admise par les articles citées plus haut). Dans notre cas, cette hyptohèse de régularité n'est pas toujours vérifiée.

gradient « 2D 1/2 »

L'estimateur proposé ici consiste pour chaque voxel, à
- extraire la carte de profondeur locale, en regardant l'objet dans l'axe de la normale du voxel. Cette carte locale peut être considérée comme une image 2D (de taille 5*5 par exemple), où la valeur des pixels indique la profondeur relative des voxels voisins. Le pixel central est évidemment à profondeur 0.

– puis on calcule le gradient du pixel central : ce sera l'estimateur.

la variance des normales

Le dernier estimateur de complexité présenté ici s'appuie sur la variance (au sens statistique) des normales du même voisinage. Il s'est révélé robuste face aux données fournies par l'OSFD, notamment sur les bords tranchants. La pointe du bord possède la complexité la plus élevée et celle-ci va en décroissant lorsqu'on s'en éloigne. D'autre part, les zones plates possèdent une complexité nulle. On détaille ci-dessous le calcul de la variance.

Soient N la taille du voisinage du voxel central, $\overrightarrow{Nc}$ la normale du voxel, central, $\overrightarrow{Ni}$ la normale du i-ème voxel appartenant au voisinage du voxel central et $\overrightarrow{Nmoy}$ la normale moyenne de toutes les normales du voisinage. La variance pour chaque axe (X, Y, Z) est :

$$\sigma_x = \frac{1}{N-1} \sum^{Ni \in Voisinage} (Ni_x - Nmoy_x)^2$$

idem pour σ_y et σ_z.

L'indice de complexité locale sera la norme du vecteur variance :

$$\|\sigma\| = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2}$$

Cet estimateur a été retenu pour le lissage anisotropique.

4.3.3 Paramètres du lissage anisotropique

Les principaux paramètres sont présentés et étudiés ci-dessous.

- le nombre d'itérations : c'est un facteur d'échelle. Plus il est élevé, plus le lissage est important. Sur des données binaires, les meilleurs résultats sont obtenus avec 5 itérations.
- le seuil de bloquage β : plus celui-ci est élévé, plus la complexité locale doit être faible pour déclencher le lissage. Il est fixé au cas par cas selon les données volumiques et la nature de l'objet. Il varie de 0.001 à 20 dans les cas rencontrés.
- taille du voisinage : plus celui-ci est élévé, plus les normales calculées sont lissées. De façon générale, pour bien contrôler les autres paramètres, celui-ci est fixé à 3 (voisinage 3*3*3). Augmenter la taille du voisinage multiplie les artefacts dûs aux données binaires et le temps de calcul.

4.3.4 Résultats obtenus

Premier cas d'étude

Premier cas d'étude, un cube avec une pointe (figure 4.7). Les paramètres sont un β fixé à 10 et un voisinage 3*3*3. Dès la première itération, et pour les 2 techniques de lissage, la pointe n'est plus visible. Avec le lissage isotropique, le cube perd sa forme originale, il devient progressivement une sphère (cf. itération 4). Il y a un changement topologique. Quant au lissage anisotropique, celui-ci conserve la géométrie initiale, tout en lissant la pointe.

Second cas d'étude

La seconde étude porte sur un modèle volumique généré à partir de cartes de profondeurs synthétiques (figure 4.8). Les paramètres sont les mêmes que ci-dessus. Après un lissage anisotropique (à droite), la sphère est devenue lisse et on peut toujours très bien distinguer les 3 petits

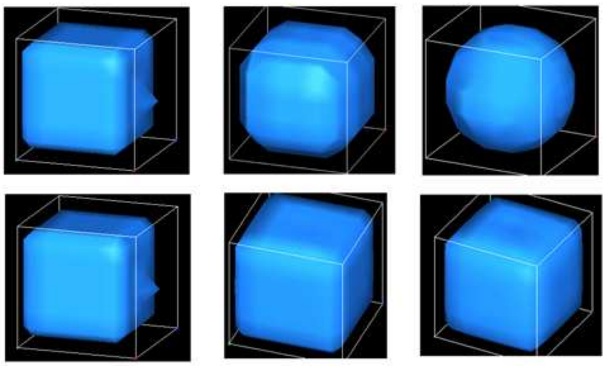


FIG. 4.7 – En haut, lissage isotropique d'un cube comportant une bosse. En bas, même cube avec le lissage anisotropique. A gauche, état initial, au milieu après 1 itération, à droite après 4 itérations

cubes qui sont sur le gros cube. L'effet d'escalier de la pyramide supérieure a disparu, sauf sur les arêtes (ce qui est normal, compte tenu de l'importance de cet effet sur le modèle initial).

On note cependant une légère instabilité dans les zones anguleuses qui se traduit par un léger creux (difficilement visible sur les captures d'écran). Cette instabilité est due à la première itération, où les données sont encore binaires. Si on a une zone similaire à celle qui est à gauche sur la figure 4.6 page 21, la complexité locale est certes bien calculée en utilisant la variance des normales, mais le lissage des voxels qui sont proches de la pointe prendra toujours en considération les voxels voisins qui sont de l'autre côté du bord.

Troisième cas d'étude

La troisième étude porte sur un modèle volumique généré à partir de cartes de profondeurs réelles (celles du masque) (figure 4.9). Les paramètres utilisés sont 5 itérations, un β à 15 et un voisinage de 3*3*3. Le modèle volumique initial (à gauche sur figure 4.9) comporte beaucoup de bosses, mais celles-ci proviennent non seulement de l'aliasing mais aussi des erreurs des cartes de profondeurs. Le lissage isotropique estompe l'aliasing et le bruit : on obtient une surface lisse mais sans aucun détail. Le lissage anisotropique estompe l'aliasing (la surface est lisse) mais certaines bosses demeurent, notamment au niveau du cou. A ce stade, il est difficile de savoir si ces bosses proviennent du bruit ou si c'est le modèle volumique qui est bosselé par nature. Pour supprimer ces bosses, il y a 2 possibilités : soit faire appel à une correction manuelle du modèle volumique, soit effectuer un lissage anisotropique des cartes de profondeurs.

Quatrième cas d'étude

L'application d'un lissage anisotropique sur un modèle volumique plus fin que le précédent (celui de la figure 1.3) a été effectuée. Ce modèle présente des détails difficilement distinguable du bruit, comme par exemple la paupière de l'oeil gauche ou la fissure en bas du front.

Si l'homme est capable de les distinguer, l'algorithme de lissage anisotropique développé ici ne l'a pas été ; son effet est resté similaire à celui du lissage isotropique. La bosse de la paupière a presque la même épaisseur que les bosses dues au bruit. L'homme arrive à les distinguer car ces fissures sont longues et régulières. Dans la conclusion de ce rapport, une piste est proposée pour détecter de telles fissures.

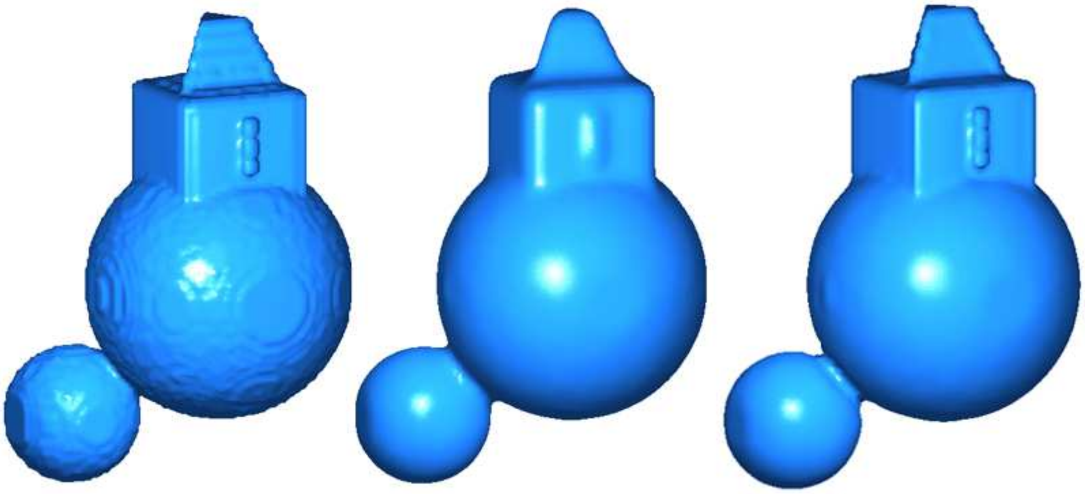


FIG. 4.8 – A gauche, le modèle volumique initial. Au milieu, après un lissage isotropique (5 itérations). A droite, après un lissage anisotropique (5 itérations)

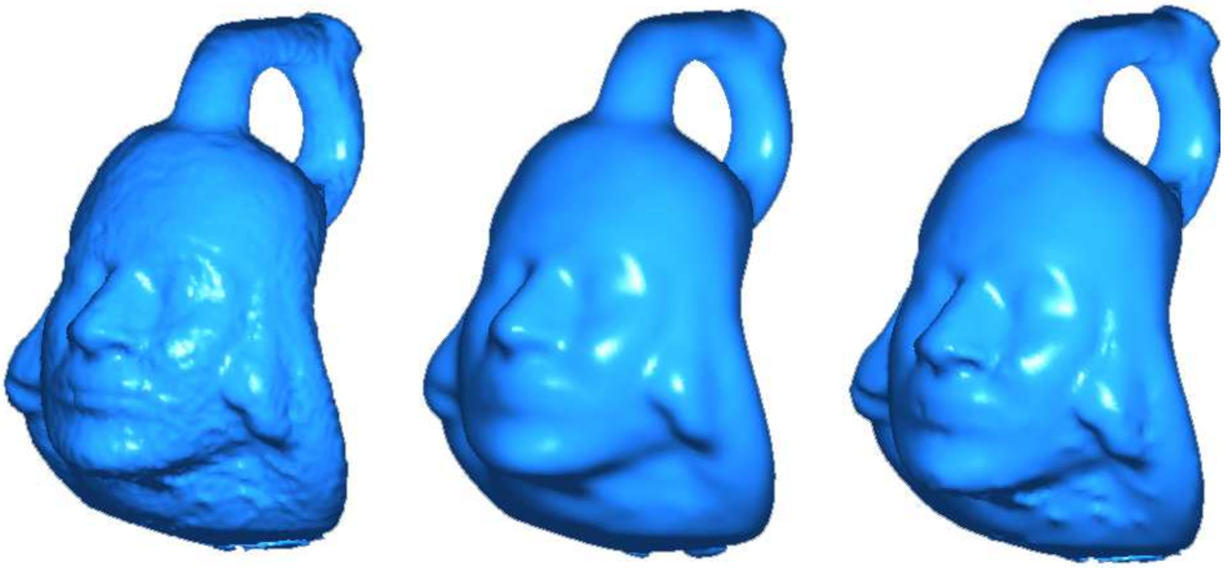
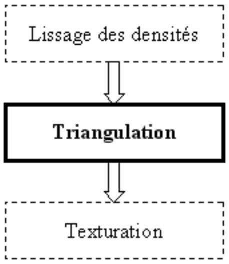


FIG. 4.9 – A gauche, le modèle initial (lissé en réduisant l'Octree sur 1 niveau). Au milieu, après 5 itérations de lissage isotropique sur le modèle à gauche. A droite, après 5 itérations de lissage anisotropique.

Chapitre 5

Extraction de la surface



5.1 Triangulation par Marching Cube

Celui-ci fait partie des algorithmes de triangulations les plus employés. Il est rapide, simple à mettre en oeuvre et donne des résultats visuels satisfaisants si les données sont lisses.

5.1.1 Principe général

Cet algorithme est conçu pour trianguler une surface définie implicitement. Cette surface est définie par un ensemble de voxels qui ont une densité (comprise entre 0 et 1) : un voxel de densité supérieure à un seuil est considéré comme étant à l'extérieur de la surface, les autres comme étant à l'intérieur de la surface [1]. Le Marching Cube va calculer les sommets des triangles par interpolation entre les densités des voxels à partir de tables de triangulation précalculées (Pour les détails de fonctionnement, on pourra se référer à l'article de Paul Bourke [7]).

5.1.2 Ses défauts

Parmi ses défauts bien connus et résolus par la communauté scientifique, citons notamment les problèmes d'ambiguïtés rencontrés lors du choix de la triangulation pour une configuration donnée. Ces défauts n'ont pas été rencontrés dans le cadre du stage et n'ont donc pas été corrigés. Par contre, les données dont on dispose à l'issue de l'OSFD soulèvent de nouveaux problèmes généralement ignorés par la plupart des chercheurs qui travaillent dans la reconstruction volumique.

[1]le seuil est fixé à 0.5 en général mais il peut être adapté au problème et aux données

Alignement des sommets des triangles sur la grille

Dans la mesure où les sommets des triangles générés sont nécessairement alignés sur une grille, il en résulte une imprécision et certaines propriétés géométriques de l'objet peuvent être impossible à reconstruire (effet notamment visible dans les zones où il y a une discontinuité de la surface). Plusieurs solutions paliant en partie cet effet existent :
- réduire le pas de la grille : plus le pas de la grille est petit, plus la position des sommets se rapprochent de l'objet réel [14]. Mais aussi petit que soit le pas de la grille, il subsistera toujours une imprécision.
- s'affranchir de la contrainte de l'alignement sur la grille en rajoutant des sommets dans les zones les plus tranchantes de l'objet (idée proposée par [14]).

Bords tranchants coupés en biseaux

Le Marching Cube coupe en biseau les bords du modèle volumique. Pour y rémédier, une solution est proposée par l'article [14] qui consiste à détecter les zones tranchantes à l'aide des normales pour ensuite les trianguler correctement. Les 2 figures ci-dessous (extraites de [14]) montrent la triangulation d'un ensemble de voxels qui forment initialement un cube : à gauche la triangulation par Marching Cube (où l'on voit les bords coupés en biseau) et à droite, la solution proposée où l'on voit en rouge et en vert les sommets rajoutés.

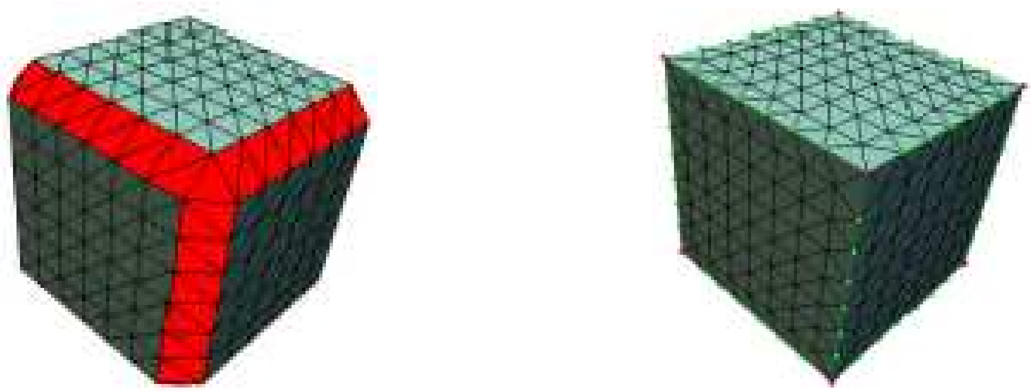


FIG. 5.1 – A gauche : la triangulation par Marching Cube (bords en biseaux). A droite : on rajoute des sommets (en vert et en rouge) et des triangles dans les zones biseautées pour améliorer le résultat

Cette solution n'a pas été mise en place dans le cadre du projet GANTOM, car elle nécessite le calcul des distances directionnelles qui n'ést pas trivial. La sous-section 5.1.3 présente une autre technique inspirée du même article [14].

5.1.3 Adaptation du Marching Cube aux zones anguleuses

L'article [14] explique bien les défauts du Marching Cube classique : chaque sommet possède une distance à la surface mais le Marching Cube classique n'utilise pas la notion de distance directionnelle. La figure 5.2 (extraite de l'article [14]) montre la différence entre l'utilisation d'une distance non directionnelle et une distance directionnelle.

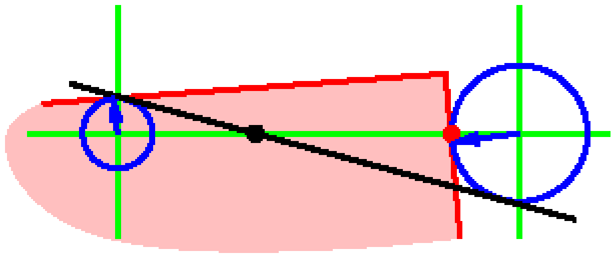


FIG. 5.2 – Si l'on considère 2 points (en vert) de la grille à proximité d'un bord (en rouge), l'utilisation d'une distance non directionnelle (en bleu) engendre une erreur lors du calcul du point d'intersection (en noir) par interpolation linéaire

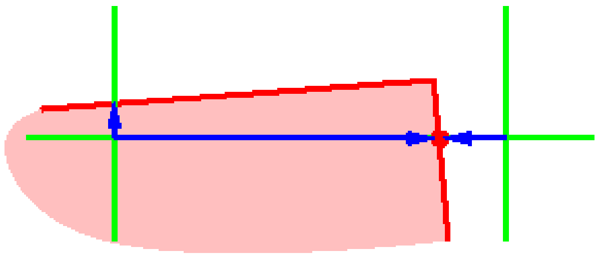


FIG. 5.3 – Si l'on utilise les distances directionnelles (en bleu), on réduit la marge d'erreur lors du calcul du point d'intersection (ici en rouge)

Les données dont on dispose permettent certes de calculer les distances directionnelles mais de façon non triviale : il s'est avéré plus simple d'utiliser les normales et la distance à la surface pour calculer les points d'intersection entre la surface et les arêtes du cube. l'annexe C détaille ce calcul.

Pour calculer un meilleur point d'intersection sur les arêtes de la grille, on calcule l'intersection entre le plan P et l'arête, le plan P approximant localement la surface. L'équation de ce plan est calculée de la façon suivante :

Soit $\vec{N_a}$ la normale en A, D_a la densité du cube A, $\vec{N_b}$ la normale en B, D_b la densité du cube B, $\vec{N_{moy}}$ la normale moyenne de $\vec{N_a}$ et $\vec{N_b}$

Si l'on considère que le point A est à l'intérieur de la surface et que B est à l'extérieur, on calcule le point M par lequel passe le plan P de la façon suivante :

$$M_x = \frac{(A_x + N_{a_x} \times D_a) + (B_x - N_{b_x} \times D_b)}{2}$$

Idem pour les M_y et M_z.

On calcule alors le point d'intersection T entre ce plan P et l'arête AB.

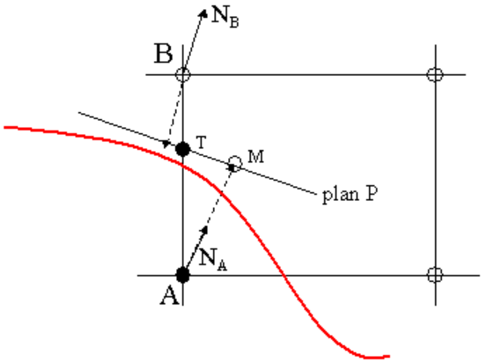


FIG. 5.4 – Calcul du point d'intersection T entre le plan P approximant la surface et l'arête AB

On prend les 2 points A et B simultanément, afin de rendre symétrique le calcul et de rester cohérent avec la notion de potentiel, sur laquelle s'appuie le Marching Cube. La figure 5.5 illustre les résultats obtenus. La différence est faible mais on note que le Marching Cube adapté ne fait finalement qu'accentuer les défauts et les discontinuités de la surface. Une étude sur le masque a révélé le même phénomène : le nez est par exemple devenu trop pointu.

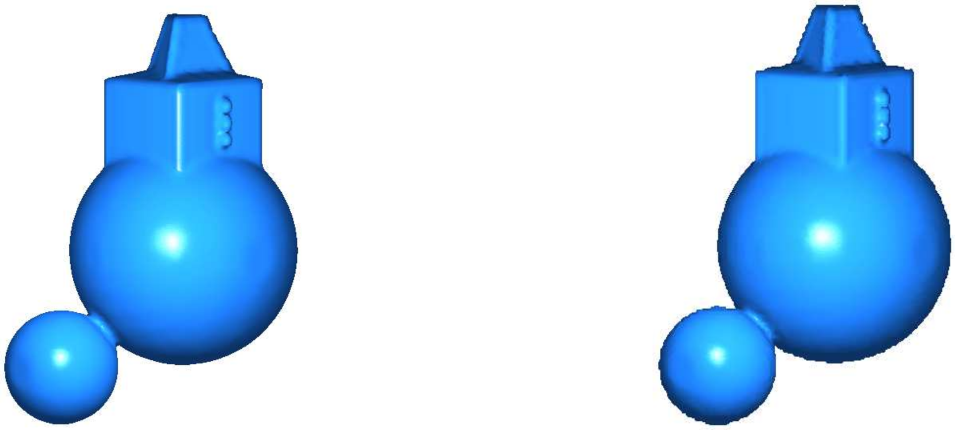


FIG. 5.5 – A gauche, extraction de la surface par le Marching Cube classique. A droite, extraction de la surface par le Marching Cube adapté

5.2 Triangulation à partir d'un nuage de points

5.2.1 Construction du nuage de points

Une autre technique pour extraire la surface est de construire un nuage de points (un voxel surfacique [1] = un point) puis de trianguler ce nuage de points avec des techniques de triangulation adaptées. La bibliographie commentée (cf. annexe F) en présente quelques-unes parmi les plus connues et performantes.

Pour chaque voxel surfacique, on estime le point par lequel passe la surface. La densité du voxel et sa normale sont utilisées pour estimer la position de ce point. L'annexe C présente le détail de ce calcul (qui n'est pas évident).

[1] voxel qui appartient à la surface de l'objet

5.2.2 Triangulation de ce nuage de points

La performance de cette technique dépend étroitement de la technique de triangulation du nuage de points. Les techniques présentées dans la bibliographie commentée permettent seulement d'obtenir une surface sans discontinuité. Or les données volumiques dont on dispose ne sont pas toujours régulières et la surface de l'objet peut présenter des discontinuités.

Une nouvelle technique, qui gère les irrégularités des données, a été développée. Pour chaque voxel, on connaît son point associé (celui qui appartient à la surface). Ce point est ensuite relié aux points associés des voxels voisins (dans le voisinage 3*3*3 maximum) par une triangulation en étoile qui s'appuie sur la topologie locale de l'objet. Cette technique est presque terminée à ce jour, quelques mises au point demeurent.

5.3 Quelle technique de triangulation adopter ?

Cette section présente une réflexion sur la technique de triangulation à adopter pour différents cas de figure : si l'objet réel est lisse, s'il est anguleux ou les deux. Les différentes techniques utilisées (lissage, triangulation) ont en effet chacune leurs spécificités propres, concernant le type de données qu'elles admettent et le type de résultat qu'elles donnent. Ce n'est pas une question de format de données (les densités sont toujours en « flottant ») mais de cohérence générale. Différents cas de figure sont présentés ci-dessous, ainsi que les techniques de reconstruction adaptées.

Si l'objet réel est lisse

Si l'objet réel est lisse, l'utilisation du Marching Cube classique précédé d'un lissage isotropique ou anisotropique 3D est recommandé. L'exemple du masque est révélateur : si on utilise le Marching Cube modifié, celui qui triangule mieux les bords tranchants (sous-section 5.1.3), le nez devient très pointu. Trop pointu. Le Marching Cube classique donne de meilleurs résultats visuels.

Si l'objet réel est très anguleux

Si l'objet réel est très anguleux, il vaut mieux faire appel à une technique de lissage autre que le Marching Cube classique. On a vu que celui-ci coupe les bords en biseau. Il a été proposé au chapitre 5 une autre technique permettant de mieux trianguler les zones tranchantes. Concernant le lissage et l'anti-aliasing, une diffusion anisotropique (présentée au chapitre 4) permet de lisser les zones courbes sans toucher aux zones anguleuses.

Si l'objet réel contient des zones lisses et des zones tranchantes

Il est souvent très difficile de savoir, sans connaissance à priori sur l'objet réel, quelles sont les parties lisses et les parties anguleuses. Entre 2 voxels qui sont chacun d'un côté d'un bord, on ignore si ce bord est droit ou coupé en biseau. Il peut être intéressant de faire appel à un opérateur humain qui indique les parties du modèle volumique qui doivent rester lisses ou, au contraire, tranchantes. Le programme de reconstruction géométrique doit alors choisir les techniques de lissage et de triangulation adaptées.

Chapitre 6

Conclusion

Nous avons vu à travers ce rapport, comment le problème de la surface bosselée a été abordé et atténué. Nous avons d'abord filtré les cartes de profondeurs avec un lissage adaptatif qui conserve les détails et enlève le bruit ; on a ainsi obtenu un modèle volumique moins bosselé. L'algorithme de fusion de données, le Shape From Depth, a ensuite été amélioré, ce qui nous a permis d'obtenir un modèle volumique plus précis.

Différentes techniques d'anti-aliasing ont été développées et l'influence de leurs paramètres a été étudiée. Le lissage anisotropique, précédé d'un lissage par réduction de l'Octree, a donné des résultats satisfaisants : l'aliasing a complètement disparu et les détails restent bien visibles.

Le test sur les données réelles a cependant révélé que certains types de bruits sont difficilement distinguables du relief réel de l'objet et les techniques de lissage adaptatif peinent parfois à faire la différence. On atteint là les limites imposées la technique d'acquisition des données : l'imprécision des mesures et de leurs traitements est de l'ordre des variations du relief de la surface de l'objet. Les fissures du masque ont par exemple une profondeur similaire aux bosses issues du bruit. Ce qui nous permet de les distinguer, c'est leur forme différente : les fissures sont plus longues que les bosses. Une classification s'appuyant sur les volumes des creux et des bosses permettrait de les distinguer [1].

Enfin, dans le dernier chapitre, nous avons vu comment adapter la triangulation de la surface pour la rendre géométriquement plus exacte. Il est notamment nécessaire d'apporter aux algorithmes des connaissances sur l'objet réel, afin que ceux-ci prennent les meilleures décisions face aux ambiguïtés du modèle volumique. L'assistance humaine dans le choix des méthodes et des paramètres de reconstruction 3D restera toujours nécessaire pour obtenir une reconstruction très performante.

[1] On peut s'inspirer de la technique de la Ligne des Partages des Eaux (LPE) pour mesurer les volumes des creux et en extraire les fissures. On remplit progressivement les creux jusqu'à une certaine hauteur avec de l'eau puis on mesure les volumes des bassins d'eau.

Bibliographie

[1] S. Acton. Multigrid anisotropic diffusion, 1998.

[2] N. Amenta, S. Choi, and R. Kolluri. The power crust,, unions of balls, and the medial axis transform. *Computational Geometry : Theory and Applications*, page pages 127 à 153, 2001.

[3] T. Apel. Anisotropic finite elements : Local estimates and applications, 1999.

[4] J. Aubourg, G. Moreau, and P. Fuchs. Accurate 3d reconstruction of museographic objects using multiple images, 2003.

[5] Chandrajit L. Bajaj and Qiu Wu. Level set based volumetric anisotropic diffusion for image filtering, 2002.

[6] M.J. Black, G. Sapiro, D. Marimont, and D. Heeger. Robust anisotropic diffusion. *IEEE Trans. on Image Processing*, 7(3) :421–432, 1998.

[7] Paul Bourke. Polygonising a scalar field. (site web : http ://astronomy.swin.edu.au/~pbourke).

[8] D. Cohen-Steiner and Frank Da. Greedy delaunay based surface reconstruction algorithm. *ECG Technical Report ECG-TR-124202-01*, 2000. (URL pour le programme : http ://cgal.inria.fr/Reconstruction).

[9] David Eberly. Level set extraction from gridded 2d and 3d data, 2002. (code source disponible à l adresse www.magic-software.com).

[10] Shachar Fleishman, Iddo Drori, and Daniel Cohen-Or. Bilateral mesh denoising. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003)*, 22(3) :950–953, 2003.

[11] J.-M. Geusebroek, A. WM. Smeulders, and J. van de Weijer. Fast anisotropic gauss filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 99–112, 2002.

[12] S. F. F Gibson. Using distance maps for accurate surface representation in sampled volumes. *IEEE Symposium on Volume Visualization*, pages pages 23–28, 1998.

[13] Thouis R. Jones, Fredo Durand, and Mathieu Desbrun. Non-iterative, feature-preserving mesh smoothing. *ACM Transactions on Graphics*, pages 943–949, 2003.

[14] Leif Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel. Feature sensitive surface extraction from volume data. *SIGGRAPH 2001 proceedings*, 2001.

[15] M. N. Kolountzakis and K. N. Kutulakos. Fast computation of the euclidean distance map for binary images, 1992.

[16] W. Lorensen and H. Cline. Marching cubes : A high resolution 3-d surface construction algorithm. *Computer Graphics*, 21 : pages 163 à 169, 1987.

[17] Matrox. Le bump-mapping. URL : www.matrox.com/mga/francais/products/tech_info/pdf/embm.pdf.

[18] B. Migeon and V. Serfaty. Filtrage adaptatif par diffusion anisotropique multiéchelle basé sur le principe du maximum d'entropie. *INRIA*, 1994.

[19] L. Neumann, B. Csébfalvi, A. König, and E. Gröller. Gradient estimation in volume data using 4d linear regression, 1997.

[20] Wolfgang Niem. Error analysis for silhouette-based 3d shape estimation from multiple views. *Proceedings on International Workshop on Synthetic - Natural Hybrid Coding and Three Dimensional Imaging (IWSNHC3DI 97)*, 1997.

[21] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, 1990.

[22] B. Rieger, F.J. Timmermans, and L.J. van Vliet. Estimation of curvature on surfaces in 3d grey-value images. *Proc. ASCI 2002, 8th Annual Conf. of the Advanced School for Computing and Imaging (Lochem, NL, June 19-21), ASCI, Delft*, 2001.

[23] C. Segall and S. Acton. Morphological anisotropic diffusion.

[24] Raj Shekhar, Elias Fayyad, Roni Yagel, and J. Fredrick Cornhill. Octree-based decimation of marching cubes surfaces. In *Proceedings of the conference on Visualization '96*, pages 335–ff. IEEE Computer Society Press, 1996.

[25] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher. Geometric surface smoothing via anisotropic diffusion of normals, 2003.

[26] U.Clarenz, U.Diewald, and M.Rumpf. Anisotropic geometric diffusion in surface processing. *Proceedings of IEEE Visualization*, 2000.

[27] S. Venegas-Martinez, J. M. Rendon, and G. Stamon. Anisotropic diffusion by a recursive linear convolving method : Application to space-time segmentation and pattern recognition, 2002.

[28] J. Weickert and B. Benhamouda. Why perona-malik filter works, 1997.

Annexe A

Extraction des cartes de profondeurs

La chaîne de traitements du projet GANTOM peut se diviser en 2 parties :
- l'acquisition des photographies suivi de l'extraction des cartes de profondeurs
- la reconstruction géométrique

Cet annexe présente la première partie de la chaîne, illustrée par la figure A.1.

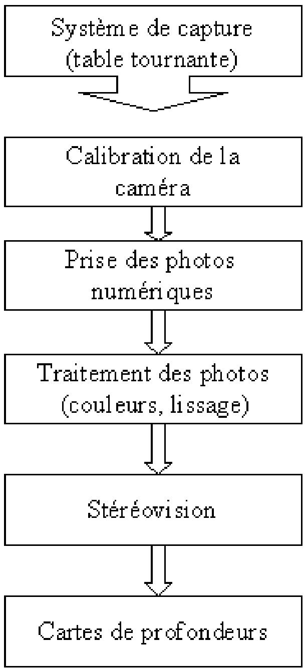


FIG. A.1 – Première partie de la chaîne de traitements : calibration, prises des vues et stéréovision

A.1 Acquisition des photos numériques et calibration

Le système de numérisation est une simple table tournante (figure A.2).

FIG. A.2 – La table tournante (en noir la table, en blanc les panneaux diffusants et au milieu l'appareil photo)

La calibration de la caméra s'effectue quant à elle en utilisant une cible sur la table tournante. Les matrices intrinsèques et extrinsèques ainsi que la déformation radiale sont alors calculées. La précision théorique de la calibration est de l'ordre du 1/10 ème de millimètre.

A.2 Extraction des cartes de profondeurs

La construction des cartes de profondeurs s'effectue par appariemments stéréoscopiques multi-échelles des prises de vues numériques. Ces cartes de profondeurs sont bruitées et comportent des erreurs ponctuelles.

Ci-dessous un exemple de cartes de profondeurs (fig. A.3).

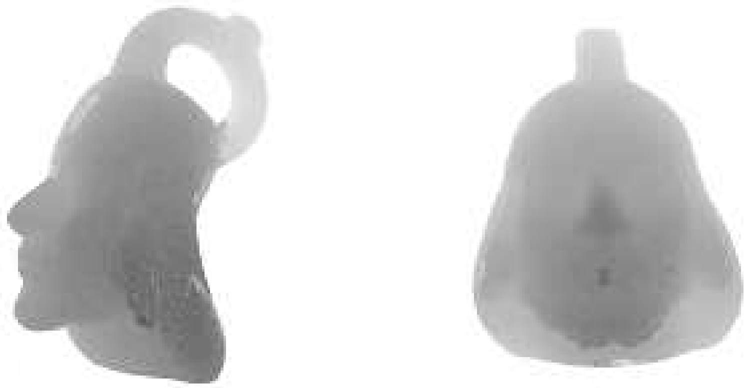


FIG. A.3 – Cartes de profondeurs (plus c'est foncé, plus c'est proche de la caméra)

Annexe B

Shape From Depth : algorithme détaillé

B.1 Algorithme général

L'algorithme exploite la récursité et la structure en Octree des voxels (cf. la section B.5 pour les détails sur cette structure). L'utilisation de la structure d'Octree permet de compresser les données volumiques au fur et à mesure de la sculpture de l'objet et de définir le niveau de détail, paramétré par l'utilisateur. Ci-dessous l'algorithme général de l'OSFD

```
Mettre tous les voxels du gros cube initial à << INSIDE >>

POUR chaque voxel V qui est << INSIDE >> FAIRE
    POUR chaque carte C de profondeur FAIRE

        Tester la position du cube par rapport à C : 3 cas sont possibles

        SI << INSIDE >> (le voxel est derrière la carte C) ALORS
            Ne rien faire

        SI << OUTSIDE >> (le voxel est devant la carte C) ALORS
            Augmenter le compteur du voxel V

            SI le compteur atteint la valeur C_MAX ALORS
             Mettre le voxel V à OUTSIDE et stopper la boucle POUR courante
            FIN SI

        SI << INTERSECT >> (le voxel est à cheval sur la carte C) ALORS
            Soit on resubdivise le voxel V pour gagner en précision
            Soit on le considère comme étant INSIDE (par défaut)

     FIN DES SI
    FIN POUR
FIN POUR
```

Notons qu'un voxel ne voit pas son statut passer à OUTSIDE tout de suite lorsqu'il est devant une carte de profondeurs. Un système de vote a été mis en place.

B.2 Système de vote

Dans la plupart des cas, les informations de profondeurs (notamment celles qui sont construites à partir de la stérévision) sont erronées : le bruit des photographies numériques et les mauvais appariemments stéréoscopiques peuvent engendrer beaucoup d'erreurs. Cependant, on a à notre disposition beaucoup d'information de profondeurs. Cette profusion de données (plusieurs millions de points de profondeurs) est mise à profit lors du Shape From Depth par le biais d'un système vote dont la règle est la suivante :

"Un voxel est considéré comme étant "Vide" si cette information est confirmée par plusieurs cartes de profondeurs simultanément"

Généralement 10% des cartes de profondeurs sont nécessaires pour confirmer la suppression d'un cube. Cette simple technique de fusion de données se révèle performante et donne des résultats graphiques très intéressant même en présence de mauvaises cartes de profondeurs. Pour s'en convaincre, il suffit de regarder les cartes de profondeurs estimées par stéréovision et le modèle 3D obtenu avec l'OSFD.

B.3 Projection des cubes, "rastérisation" des triangles

Chaque voxel est projeté sur toutes les cartes de profondeurs afin de connaître sa position relative à ces cartes (devant , derrière ou intersectant). Pour ce faire, le voxel est décomposé en 12 triangles (sa surface) et on teste la position de chaque triangle par rapport aux cartes de profondeurs. Il est nécessaire d'effectuer une rastérisation complète de chaque triangle afin de tester la position de chaque point intérieur aux triangles par rapport aux cartes de profondeurs. Chaque triangle fait l'objet d'un balayage vertical (selon l'axe des Y) puis d'un balayage horizontal (dans l'axe des X) d'un bord à l'autre du triangle. Dans la mesure où l'on atteint rapidement une résolution proche ou inférieure au pixel, on effectue le balayage horizontal pixel par pixel.

B.4 Précision du Shape From Depth

Précision du Shape From Silhouette

W. Niem fait une première étude de la précision du Shape From Silhouette [20]. Il étudie notamment l'influence

- de la calibration imprécise de la caméra
- du nombre de silhouettes
- et de la discrétisation en voxel

sur l'exactitude de la sculpture. Ce calcul est différent avec le Shape-From-Depth car toutes les concavités sont prises en compte. On pourra se référer à l'article de Niem, notamment si l'on s'intéresse à l'influence de l'imprécision de la calibration.

Précision du Shape From Depth

On considère, temporairement, que la précision maximale est atteinte lorsque les voxels projetés sur les cartes de profondeurs ont une taille inférieure à celle du pixel. La précision atteinte par le Shape From Depth dépend étroitement :

- de la matrice de projection de la caméra (focale + "zoom" horizontal et vertical)
- de la distance moyenne (ou maximale dans le pire des cas) entre la caméra et l'objet
- du paramètre sizeCube (taille du plus petit voxel) défini par l'utilisateur.

Ainsi, si

- la carte de profondeur fait 2000 * 2000 pixels,
- si l'objet est à une profondeur moyenne de 500 mm par rapport à la caméra
- et si SizeCube = 0.2 mm,

alors la taille des voxels projetés sera approximativement de l'ordre du pixel (0.2 * 500 / 2000 = 0.8). Si l'on considère que les cartes de profondeurs couvrent une surface de 300 mm * 300 mm, un pixel représente une distance de 1/7 mm soit 0.14 mm. Ainsi, la précision moyenne atteinte est de l'ordre du 1/10 de millimètre. Notons que les voxels ne se projettent pas toujours à l'intérieur des pixels et chevauchent plusieurs pixels simultanément (détail section 3.2). Le Shape From Depth prend en compte ce phénomène et redivise les voxels qui sont à cheval sur plusieurs pixels en plus petits voxels. C'est ainsi que le paramètre SizeCube doit être défini en dessous de la précision théorique maximale.

Nb : les valeurs choisies ci-dessus correspondent aux valeurs habituelles rencontrées lors de la numérisation d'objet muséographiques de taille moyenne (environ 30 cm * 30 cm * 30 cm).

B.5 Structure de l'Octree

L'octree est un arbre dont chaque noeud possède jusqu'à 8 feuilles et une variable « filled » définie ci-après. Un noeud est considéré comme étant une feuille si ses 8 fils sont nuls. La variable « filled » indique si cette feuille est pleine ou vide.

La structure d'Octree est utilisée ici pour 3 objectifs :

- optimisation de la mémoire utilisée lors du OSFD
- paramétrisation de la taille minimale des voxels à atteindre (dépend de la hauteur de l'arbre). La figure B.1 illustre les différentes résolutions disponibles selon la hauteur de l'arbre considérée.
- lissage des densités des voxels par réduction de la hauteur de l'arbre (cf. section 4.2)

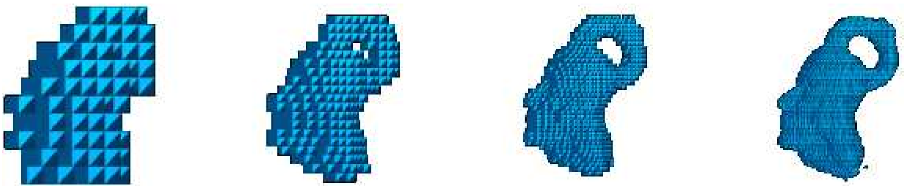


FIG. B.1 – Les différents niveaux de détails implicitement offerts par la structure d'Octree

Annexe C

Conversion Densité $\longleftrightarrow$ Distance à la surface

La notion de densité des voxels ne correspond pas forcément à la notion de potentiel comme il est usuellement admis lorsqu'on utilise le Marching Cube. On peut aussi voir la densité comme une information de distance entre le voxel et la surface. Plusieurs études ont été faite à ce sujet ; citons les articles [15] et [12] qui présentent chacun une technique pour calculer la distance entre un voxel et la surface de l'objet. La densité sera assimilée ici à une information volumique. Connaissant la densité du cube et sa normale, on peut en déduire la surface (figure C.1).

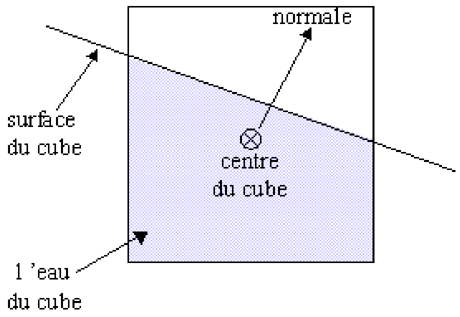


Fig. C.1 – le cube et sa surface (vue latérale)

Pour mieux visualiser le problème, on peut imaginer que l'on remplit le cube avec de l'eau (le volume d'eau est déterminé par la densité) puis que l'on incline le cube selon l'orientation de la normale. La normale du cube indique l'orientation du plan, il ne reste plus, avec la densité, qu'à calculer par quel point passe ce plan. La surface du cube sera définie par sa normale et la distance qui sépare le centre du cube et le plan. Toute la difficulté est alors de calculer l'équation du plan de la surface. Il n'existe pas d'équation simple pour passer de la densité à l'équation du plan. 2 idées ont été envisagées pour calculer cette équation.

C.1 Technique 1 : Calcul approximatif de l'équation du plan

Lorsque la densité vaut 0.5, la surface passe toujours par le centre du cube (cf. plus loin) et lorsqu'elle vaut 1, le plan passe par le dernier sommet atteint lorsqu'on remplit complètement le cube d'eau dans la direction de la normale (voir figure C.2).

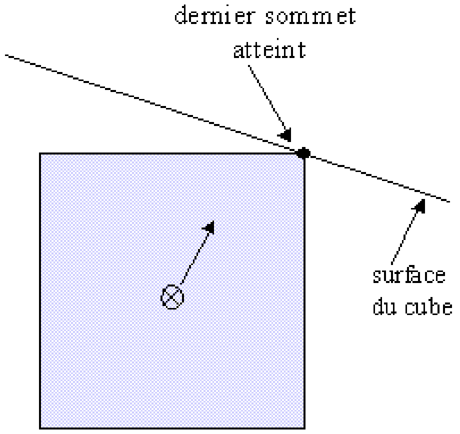


FIG. C.2 – Dernier sommet atteint (cube avec une densité de 1)

On détermine ensuite la distance entre la surface et le centre du cube de la façon suivante : soient $\vec{N}$ la normale du cube, et *densite* sa densité ; P_{max} le plan défini par $\vec{N}$ et passant par le dernier sommet atteint si l'on remplit le cube complètement et D_{max} = distance entre le centre du cube et le plan P_{max} La distance D entre le centre du cube et la surface du cube est : $D = \frac{D_{max}}{densite}$

Cette technique est simple et rapide mais la précision reste assez grossière : il n'y a pas de proportionnalité entre la densité du cube et la distance entre son centre et sa surface. Elle peut servir de point d'initialisation pour la seconde technique, présentée ci-après.

C.2 Technique 2 : Approche dichotomique

On calcule l'équation du plan par dichotomie : on effectue une première estimation de l'équation (à partir de la méthode ci-dessus par exemple) puis on calcule le volume délimité par cette surface. On sait alors si l'équation calculée est trop haute ou trop basse par rapport au niveau de l'eau réel. On ré-estime alors l'équation du plan. Ce processus est réitéré jusqu'à la convergence ou dès que la précision souhaitée est atteinte. L'algorithme est détaillé ci-après.

```
Initialisation :
   1) Faire une première estimation de la distance (D)
      entre le centre du cube et la surface (par exemple D = 0)
   2) borneMin = -2, borneMax = 2

TANT QUE la précision n'est pas atteinte, FAIRE

      Calculer le volume (V) délimité par la surface estimée

      SI ( V < densité du cube ) ALORS
          D = (borneMax-borneMin)/2
          borneMin = (borneMax-borneMin)/2
      SI ( V > densité du cube ) ALORS
          D = (borneMax-borneMin)/2
          borneMax = (borneMax-borneMin)/2
      SINON
          Précision atteinte
      FIN SI
FIN TANT QUE
```

A chaque itération, on calcule le volume délimité par la surface. Cette opération peut s'effectuer de plusieurs façons, en voici un : on calcule le centre de gravité (G) de ce volume (équibarycentre des 8 sommets), puis on estime le volume à partir des 6 pyramides partant de G ayant pour base les 6 faces. La deuxième méthode est plus précise, non linéaire, mais nécessite plus de calculs (le calcul du volume à chaque itération n'est pas immédiat).

C.3 Une densité de 0.5 et centre de gravité

Un cube de densité 0.5 verra sa surface passer par le centre du cube :

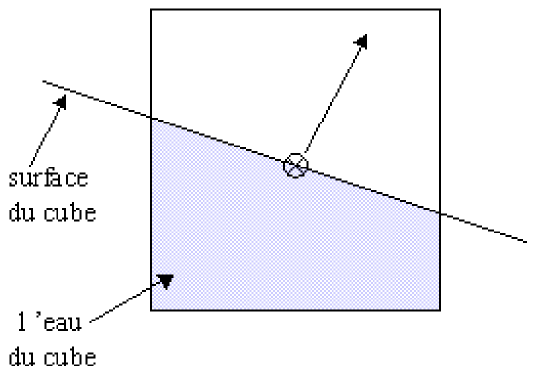


Fig. C.3 – Tout plan qui passe par le centre de gravité coupe le volume en 2 parties égales

En effet, le centre du cube se superpose au centre de gravité (du cube plein) et ce dernier est l'équibarycentre des masses ponctuels de l'ensemble du cube. Si l'on suppose le cube plein, tout plan passant par le centre de gravité coupe le volume du cube en 2 parties égales en masse donc en volume (la densité est constante à l'intérieur du cube).

Annexe D

Générateur de cartes de profondeurs synthétiques

Afin de tester l'algorithme du Shape from Depth du projet GANTOM, le programme développé ici permet de générer automatiquement des cartes de profondeurs précises au format GANTOM. Le principe général du programme est d'utiliser OpenGL pour construire un objet 3D pour récupérer le z-buffer calculé par la carte graphique. Ceci permet d'utiliser les fonctions avancées d'OpenGL pour générer des objets complexes, précis et visualisables, de s'affranchir des difficultés du rendu (notamment le z-buffer et autres optimisations) et accessoirement d'utiliser l'accélération HardWare des cartes graphiques pour le rendu. Les difficultés rencontrées sont la faible précision du z-buffer (8 bits) qui nécessitent plusieurs passes et le passage du repère OpenGL (repère direct) au repère Gantom (repère indirect). Notons que la plupart des cartes graphiques gèrent le z-buffer en 16 ou 24 voire 32 bits mais la lecture de ce buffer, du moins en utilisant les fonctions standards OpenGL, reste limitée à 8 bits.

D.1 Fonctionnement général du programme

La caméra OpenGL tourne autour de l'origine où se trouve l'objet 3D à rendre. On effectue à chaque rotation de la caméra une prise de vue (un rendu) et on récupère la carte de profondeurs (z-buffer). La caméra peut aussi bien tourner horizontalement autour de l'axe Y que verticalement.

On calcule les matrices intrinsèques et extrinsèques utilisées pour projeter cet objet à chaque position de la caméra au format compatible GANTOM.

Les matrices et la carte de profondeurs sont alors stockées dans plusieurs fichiers : i.a, i.mat et i.data (avec i, le numero de la carte de profondeur). Un quatrième fichier, i.tif ou i.raw, stocke la carte de profondeur sous forme d'une image pour visualiser rapidement les résultats.

D.2 Récupération du z-buffer généré par OpenGL

On récupère le z-buffer en utilisant la commande :

```
glReadPixels(0,0,256,256,GL_DEPTH_COMPONENT ,0,carteProfondeur)
```

La qualité de cette carte de profondeur dépend des paramètres zNear et zFar qui indiquent la profondeur minimale et maximale à prendre en compte. Plus ces valeurs sont proches l'une de l'autre, meilleure sera la précision. zNear est ici égal à 300 et zFar à 900 (on travaille en millimètre). Ces valeurs sont données à la fonction gluPerspective(...) .

La carte de profondeur calculée par OpenGL est de faible précision puisqu'en 8 bits. 2 solutions sont envisageables : utiliser une extension OpenGL qui gère des cartes de profondeurs de 16 ou 24 bits (mais qui nécessite une carte graphique appropriée) ou effectuer plusieurs passes, ie. plusieurs rendus successifs en changeant à chaque fois les valeurs zNear et zFar. Seule cette deuxième solution est retenue, l'écart entre zNear et zFar est alors fixé à 50 mm. Ainsi 8 bits codent 50 mm, on a une précision inférieure à 1/5 de mm donc inférieure également à la précision théorique du système GANTOM actuel (environ 0,3 mm). On effectue une quinzaine de passes par carte de profondeurs.

D.3 Calcul des matrices intrinsèques et extrinsèques compatibles Gantom

Calcul de la matrice extrinsèque GANTOM

Les repères OpenGL et Gantom ne sont pas orientés de la même façon, le premier est direct, le second indirect. La méthodologie employée pour générer des cartes de profondeur compatibles GANTOM et pour calculer les matrices MAT (extrinsèques) est la suivante : Pour la première prise de vue, les caméras OpenGL et GANTOM sont placées à Z = 600mm, la caméra OpenGL regarde vers les Z négatifs (position par défaut) et celle de GANTOM vers les Z positifs. On adapte alors la matrice MAT de sorte que les axes oX, oY et oZ GANTOM collent avec les axes OpenGL.

La matrice MAT initiale est ainsi égale à :

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 600 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Comme les repères ne sont pas orientés de la même façon, si l'axe Z est conservé, il faut alors renverser les axes oX et oY, d'où les -1 sur la diagonale de la matrice. A chaque prise de vue, on fait tourner la caméra OpenGL autour de l'axe Y d'un angle positif, et la caméra GANTOM d'un angle négatif.

Calcul de la matrice intrinsèque GANTOM

Cette matrice (aussi appelée matrice A) est obtenue en récupérant la matrice de projection d'OpenGL :

```
glGetFloatv(GL_PROJECTION_MATRIX, matriceA);
```

Cette matrice de projection donnent des valeurs normalisées (comprises entre -1 et 1). OpenGL calcule les coordonnées des points projetés à l'écran avec une dernière matrice, la " ViewPort Matrix " qui prend en compte les dimensions de la fenêtre d'affichage : elle multiplie les valeurs projetées par la moitié de la taille de la fenêtre auquelle elle rajoute la position du centre de l'écran. La matrice A calculée est ainsi (pour une fenêtre d'affichage 256 * 256) :

$$\begin{bmatrix} 274.5 & 0 & 128 \\ 0 & 274.5 & 128 \\ 0 & 0 & 0 \end{bmatrix}$$

D.4 Autres fonctionnalités

Outre la possibilité de faire tourner la caméra dans tous les sens (en restant à la même distance du point 0), citons les autres fonctionnalités :

Prise en charge du format VRML

Le programme peut prendre en paramètre de ligne de commande un nom de fichier de type VRML (donc qui stocke un objet VRML) pour ensuite l'afficher à la place des 2 sphères. Ceci permettra d'utiliser facilement des objets graphiques plus complexes.

Génération du fichier Conf pour le shape-from-depth

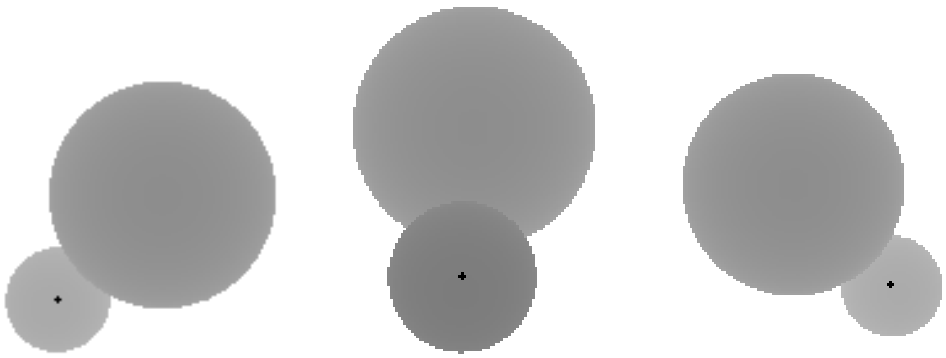
Le programme peut générer le fichier de configuration utilisé par le shape-from-depth qui stocke l'ensemble des matrices A et MAT ainsi que l'emplacement des cartes de profondeurs.

Génération du bruit

Le programme peut rajouter du bruit aux cartes : il peut être uniforme (bruit aléatoire) et ponctuel (rajout d'informations fausses disposées aléatoirement dans la carte de profondeurs).

D.5 Test du programme

Le test du programme est effectué en calculant les cartes de profondeur de 2 sphères, une grosse et une petite. L'exactitude des matrices GANTOM est vérifiée en les utilisant pour projeter le centre de la petite sphère en superposition sur la carte de profondeurs générée par OpenGL. On vérifie bien la corrélation entre les deux systèmes (les croix indiquent la position du centre de la petite sphère reprojetée) .



0° 112,5° 236,25°

Annexe E

Calcul des normales

E.1 L'importance de la normale sur le rendu

La figure E.1 illustre l'importance d'avoir une normale bien estimée en chaque voxel.

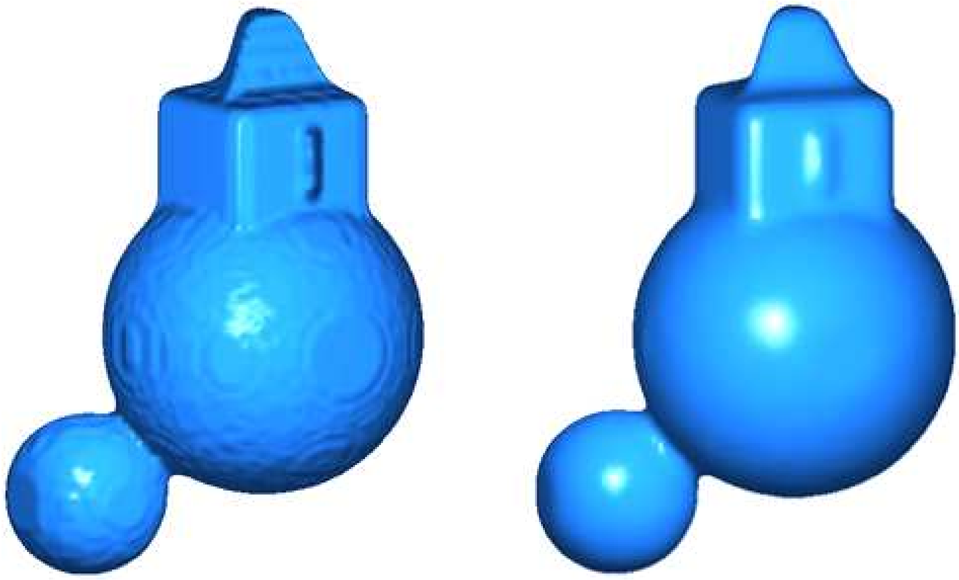


FIG. E.1 – A gauche, les voxels sont lissés mais pas les normales. A droite, les mêmes voxels avec des normales lissées (en fait, recalculées à partir des voxels lissés)

Nous voyons que la façon dont les normales sont calculées et lissées joue un rôle important dans le processus de lissage et de rendu [1]. D'où toute l'attention qui y est portée.

E.2 Calcul de la normale

La technique retenue est celle proposée par l'article [19] qui s'appuie sur une technique de régression linéaire 4D. Sachant que la normale d'un point P définit l'équation du plan de la

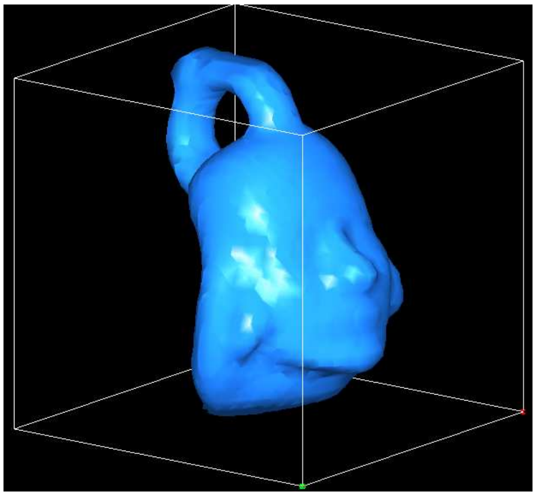
[1]Le bump-mapping s'appuie sur ce principe : c'est une technique qui permet de créer une surface, par exemple rugueuse, en lissant de façon inhabituelle les normales. On pourra se référer à [17] pour les détails

surface en P, on cherche à approximer les coefficients de ce plan à partir du voisinage de P. Cette technique a été utilisée en raison de sa simplicité et de son efficacité, même en présence de données binaires.

Pour calculer la normale d'un voxel, on s'appuie donc sur les densités des voxels voisins. Dans la plupart des cas, seul le voisinage immédiat (3*3*3) est considéré.

E.3 Interpolation des normales

Lorsqu'on utilise le Marching Cube, les sommets qui forment les triangles ne sont pas placés au niveau de voxels (du moins rarement). La surface apparaîtra plus lisse si on ré-estime les normales par interpolation à partir des normales des voxels voisins. On fait ainsi appel aux 8 voxels voisins. La figure E.2 illustre l'amélioration.



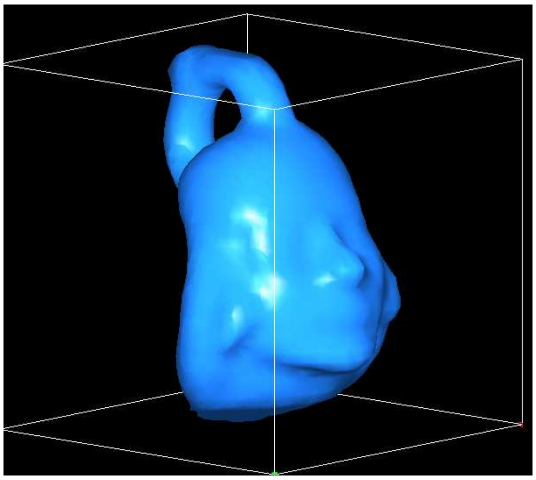


FIG. E.2 – A gauche : les normales non interpolées. A droite : les normales interpolées à partir du voisinage par moyennage pondérée

Annexe F

Bibliographie commentée

Cette section retrace ici les travaux qui ont déjà été réalisés dans le domaine du lissage et de la triangulation.

F.1 Techniques de lissage anisotropique

Le lissage anisotropique 2D a été développé pour la première fois par P. Perona et J. Malik en 1990 ([21] et [28]). De nombreux articles ont été publiés, certains proposant des améliorations en terme de qualité ([6], [23]), d'autres en terme de rapidité d'éxécution ([11], [1], [27]). Citons l'article [3] qui propose un lissage anisotrope s'appuyant sur la théorie des éléments finis.

Dans le domaine 3D, on peut citer les récents travaux :
- de Clarenz et al. [26],
- de T. Tasdizen et al. [25]
- de C.L. Bajaj et al. [5]
- de S. Fleishman et al. [10]
- et de T.R. Jones [13]

L'idée de lisser les densités des voxels, de façon « anisotropique », est récente. Quelques-un des articles cités ci-dessus ([13], [10] et [25]) ont été publiés en 2003 et pendant le stage.

F.2 Techniques de triangulation

On peut classer les techniques de triangulation en 2 catégories :
- celles qui s'appuient sur une surface définie implicitement
- celles qui s'appuient sur un nuage de points représentant la surface de l'objet.

A l'issue du Shape From Depth, on obtient des voxels de densités binaires (ou quasi-continues si on lisse) à partir duquel on estime implicitement la surface que l'on triangule. On fait ainsi appel à la 1ère catégorie de techniques, résumée ci-dessous.

Triangulation des surfaces implicites

La technique la plus utilisée est le Marching Cube [16]. Il s'appuie sur une table de triangulation précalculée qui stocke, pour toutes les situations que l'on peut rencontrer, la triangulation la mieux adaptée.

L'article [14] propose d'utiliser les distances directionnelles des voxels à la surface. Avec les objets anguleux, la qualité de la triangulation est améliorée. Mais si l'objet réel est globalement lisse, la qualité est dégradée, l'objet reconstruit étant plus anguleux que dans la réalité (cf. section 5.3 page 29).

D'autres techniques qui s'inspirent du principe mais sans utiliser de tables précalculées ont été mises en oeuvre. Citons notamment les travaux de David Eberly [9].

Triangulation d'un nuage de points

A partir de chaque voxel de la surface, on peut associer un point dans l'espace qui appartient à la surface de l'objet. On obtient alors un nuage de points représentant l'ensemble de la surface de l'objet. On fait alors appel à une technique de triangulations d'un nuage de points dont les plus populaires sont :

- la tétrahédrisation de Delawnay (contrainte ou non) suivi d'une extraction des triangles appartenant à la surface.
- le « PowerCrust », développé par N. Amenta et al. [2] (le code source est disponible sur la page perso de N. Amenta)
- l'algorithme développé par l'INRIA [8] sur la base de la bibliothèque CGAL[1]

[1]Disponible à l'adresse http ://www.cgal.org/