Génération de surfaces adaptatives à partir de données volumiques binaires

R. Uribe Lobello[1] F. Denis[2] F. Dupont[2]

Université de Lyon , CNRS
[1]Université Lumière Lyon 2, LIRIS UMR 5205, France
[2]Université Claude Bernard Lyon 1, LIRIS UMR 5205, France
{Ricardo.Uribe-lobello, Florence.Denis, Florent.Dupont}@liris.cnrs.fr

Résumé

Nous présentons un algorithme pour la génération de surfaces multi-résolution adapté aux données volumiques. Ces dernières peuvent être soit composées d'un ensemble d'images segmentées soit obtenues par un échantillonnage régulier d'une fonction implicite. En effet, certaines applications, telles que les simulations par éléments finis, nécessitent une représentation explicite de la surface de l'objet. Par ailleurs, un échantillonnage régulier pouvant faire apparaître un certain crénelage sur la surface, notre solution utilise un octree afin de générer des surfaces adaptées à la courbure locale du volume. Une nouvelle méthode de localisation des sommets est également introduite afin de produire des surfaces lisses qui atténuent les artefacts dûs à l'échantillonnage régulier. Notre algorithme produit toujours des surfaces 2-variétés fermées. En comparaison avec les méthodes de l'état de l'art, notre solution offre un bon compromis entre la qualité du maillage et la qualité de l'approximation. Ceci est mis en évidence par les résultats expérimentaux.

Mots clefs

Reconstruction de Surface, Modèles Multi-résolution, Représentation d'Objets, Données Volumiques Binaires.

1 Introduction

Les données volumiques sont très présentes dans différents domaines, notamment l'imagerie médicale, la biologie... Une étape préalable de segmentation permet de bien identifier l'intérieur et l'extérieur des objets d'intérêt. Une fois labellisées, ces données volumiques peuvent être vues comme l'échantillonnage régulier d'un champ scalaire $F(x, y, z)$ où $F(x, y, z) > \omega$ dénote l'appartenance d'un voxel à l'intérieur d'un objet λ.
Néanmoins, une représentation volumique n'est pas la plus efficace pour des applications telles que la visualisation ou la simulation par éléments finis. Afin d'extraire des surfaces adéquates pour ce type d'applications, nous proposons une nouvelle technique pour la génération de surfaces. Notre méthode extrait des surfaces 2-variété fermées avec une meilleure adaptabilité aux caractéristiques des objets que les méthodes existantes, en associant une nouvelle localisation par composantes connexes permettant d'obtenir des surfaces lisses et d'atténuer fortement l'apparition de crénelage.

2 État de l'art

Il existe une grande quantité d'algorithmes pour la génération de maillages à partir des fonctions implicites discrètes (voir [1]). L'algorithme Marching cubes (MC) [2] extrait une surface d'un volume en utilisant une grille régulière. Malheureusement, MC génère trop de triangles et n'approxime pas bien les parties saillantes du volume. Pour remplacer la grille régulière, Wilhelms et Van Gelder [3] ont proposé l'utilisation d'un octree. Même si cette méthode améliore la vitesse du MC, elle n'améliore pas l'approximation des parties saillantes ni la qualité du maillage obtenu. La méthode de Kobbelt *et al.* [4] approxime mieux le volume en ajoutant des sommets supplémentaires à l'intérieur des cellules de l'octree et utilise des données (dites d'Hermite) pour bien les placer par rapport à la surface. Malgré cela, cette méthode génère trop de triangles et le maillage obtenu n'est pas adaptatif. Afin de rendre MC adaptatif, Varadhan *et al.* [5] ont proposé des algorithmes basés sur l'évaluation de la topologie locale de λ. Pourtant, ces algorithmes ont une adaptabilité qui est très limitée par la topologie de λ. Kazhdan *et al.* [6] ont proposé un algorithme qui génère des surfaces multi-résolution sur des octree arbitraires. Toutefois, cette méthode utilise MC à l'intérieur des cellules et n'approxime pas bien λ.
Pour construire des maillages adaptatifs, Ju *et al.* ont présenté un algorithme dual (DC) [7] qui, contrairement aux méthodes précédentes, ne place pas les sommets du maillage sur les arêtes des cellules mais crée un sommet (dual) à l'intérieur de chaque cellule. Cela permet d'obtenir des surfaces multi-résolution adaptées à la forme de λ. Dans certains cas d'ambiguïté topologique, les surfaces générées peuvent contenir des configurations non 2-variété ; ce problème est partiellement résolu par la méthode Dual Marching Cubes (DMC) [8] qui autorise plusieurs sommets dans une même cellule. Manifold Dual Contouring (MDC) [9] est une extension de DMC qui peut générer des surfaces adaptatives. Elle utilise un critère de simplification en fonction de la topologie de l'octree qui

garantit la conservation de la topologie du volume. Cet algorithme ne garantit pas l'absence d'auto-intersection de la surface.
Schaefer *et al.* [10] ont présenté une méthode qui génère une grille duale avec des cellules cubiques afin de pouvoir utiliser MC. Néanmoins, les surfaces générées peuvent contenir des artefacts topologiques parce que les cellules ne sont pas convexes. Récemment, Manson et Schaefer [11] ont proposé la génération d'une grille duale tétrahédrique combinée à un algorithme de Marching Tetrahedra pour extraire une surface sans artefact. Malheureusement, la surface générée contient beaucoup de triangles dégénérés.

Contributions

Dans cet article, nous proposons une extension de la méthode Dual Marching Cubes [8] adaptée aux données volumiques. Notre algorithme utilise une approche "top-bottom" afin de générer des surfaces multi-résolutions qui sont toujours des 2-variétés fermées. Nos trois principales contributions sont les suivantes :

– un algorithme pour extraire des surfaces 2-variété fermées en utilisant une extension de DMC ;
– un critère permettant de raffiner localement la surface tout en conservant les propriétés topologiques ;
– une nouvelle technique de localisation des sommets basée sur des composantes connexes et qui fixe le sommet du maillage à la surface de l'objet volumique.

3 Génération de surfaces multi-résolution

L'entrée de notre algorithme est un volume 3D binaire qui contient un objet d'intérêt λ. Nous considérons ce volume comme un domaine Ω et les valeurs de chaque voxel comme les valeurs d'une fonction indicatrice $\{F(x) \mid x \in \Omega\}$ tel que $\{F(x) = 1 \mid x \in \lambda\}$ et $F(x) = 0$ sinon. Notre solution se décompose en trois étapes principales : construction d'un octree adapté à la surface de λ, création des sommets et génération de la connectivité du maillage. Avant de les présenter, nous précisons quelques notations préliminaires.

Notations

Soit C une cellule considérée comme un graphe G comportant huit sommets ayant la connectivité d'un cube. C est alignée sur les axes principaux xyz et chaque sommet de C peut être à l'intérieur de λ (*solide*) ou dans son complément λ^c (*vide*). Nous notons $\partial\lambda$ la surface de λ.

Définition 1 *Soit G le graphe défini par les sommets d'une cellule C. Soient $S(C)$ et $V(C)$ les sous-ensembles des sommets $v_i \in G$ tels que $\{\forall v_i \in S(C), v_i \in \lambda\}$ et $\{\forall v_i \in V(C), v_i \in \lambda^c\}$. Soient $\mid S(C) \mid$ et $\mid V(C) \mid$ leur cardinal.*

Définition 2 *Soient $\phi_s(C)$ et $\phi_v(C)$ les composantes connexes de $S(C)$ et $V(C)$ respectivement.*

Une *facette ambigüe* d'une cellule C est une facette qui contient deux sommets solides non adjacents séparés par des sommets vides. Les sommets qui feront partie du maillage et qui seront créés à l'intérieur des cellules seront appelés *sommets duaux*. L'ensemble des sommets duaux dans une cellule C est noté $D(C)$ et son cardinal $\mid D(C) \mid$.

3.1 Construction de l'octree

La première étape de construction d'un octree se base sur celle proposée par Lewiner *et al.* [12]. Il s'agit d'un "octree" qui utilise une table de hachage indexée avec un code de Morton permettant de coder la position de chaque cellule dans la hiérarchie ainsi que sa position géométrique dans le domaine divisé. L'utilisation des algorithmes associés à cette structure de données fournit ainsi un accès très efficace au voisinage d'une cellule.
Pour construire l'octree, il faut détecter les cellules qui sont traversées par la surface. D'abord, nous observons les valeurs de F aux sommets de la cellule. Si certains sommets sont à l'intérieur de λ et d'autres à l'extérieur, alors $\partial\lambda$ traverse la cellule, celle-ci n'est pas homogène et sera donc divisée. Cependant, ce critère ne permet pas de déterminer toutes les intersections possibles avec $\partial\lambda$. Pour résoudre cela, nous utilisons un test qui détecte les cellules traversées par $\partial\lambda$ mais sans modification de la valeur de leurs sommets, elles sont appelées cellules complexes.

Définition 3 *Une cellule C est complexe si au moins une de ses facettes est complexe. Une facette est complexe si elle traverse λ et si tous ses sommets sont soit à l'intérieur, soit à l'extérieur de λ (voir figure 1). Une facette est aussi complexe si une de ses arêtes traverse plus d'une fois $\partial\lambda$.*

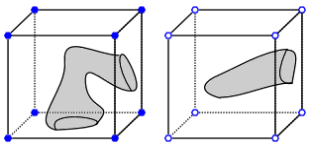


Figure 1 – *Cellule complexe avec un tunnel (à gauche), cellule complexe comportant une facette complexe (à droite).*

Les facettes complexes sont détectées par l'examen de leurs quatre sommets, puis s'ils sont tous solides ou tous vides, par l'extraction des composantes connexes sur la facette. Cela nécessite le parcours des voxels appartenant à la facette. Si il existe au moins une composante connexe appartenant à λ et au moins une à λ^c, alors la cellule est complexe.

Critère 1 *Soit C une cellule. C doit être divisée si et seulement si elle est complexe.*

Le critère 1 assure que l'octree sera raffiné jusqu'à ce qu'il n'existe plus de cellules complexes.

Adaptation de la surface à la courbure

La seule utilisation du critère 1 va produire des maillages adaptatifs. Mais, pour optimiser la triangulation par rapport à la géométrie de la surface, nous avons choisi d'utiliser un critère supplémentaire basé sur la courbure estimée de $\partial\lambda$ dans une cellule C. Si à l'intérieur de C, la courbure de $\partial\lambda$ est forte, C sera divisée. L'estimation de la courbure est réalisée en calculant les normales aux points d'intersection de $\partial\lambda$ avec les arêtes de la cellule. Pour cela, nous avons utilisé une approche surfacique qui utilise les normales unitaires (alignées sur les axes principaux) des surfels de la surface du volume autour du point d'intersection. Nous calculons la moyenne de ces normales dans un voisinage géodésique avec un rayon proportionnel à la taille de la cellule [13]. Le critère de courbure est formulé comme suit :

Critère 2 *Soient C une cellule, $I_{i=0\ldots n}$ ses points d'intersection avec $F(x)$ et $n_{i=0\ldots n}$ les normales associées. δ est un paramètre dont les valeurs sont comprises entre 0 et 1. Si Max$\{n_i \bullet n_j | i \neq j\} > \delta$, C doit être divisée.*

Il existe une relation de dépendance entre les critères 1 et 2. Il faut d'abord vérifier que la cellule ne soit pas complexe (critère 1) pour assurer une convergence dans le calcul de la courbure (critère 2). Les cellules qui ne sont pas divisées, que ce soit parce que la surface est très plane ou parce que l'intersection avec $\partial\lambda$ est simple, sont marquées comme *compactes*. L'algorithme complet pour la construction de l'octree est présenté dans 1.

Algorithm 1: Construction de l'octree

```
input  : Cellule c racine de l'octree. Niveau minimal minLevel et maximal
         maxLevel de subdivision de l'octree.
output: Octree régulier jusqu'au niveau minLevel et adaptatif entre minLevel
         et maxLevel.
Ajouter c à la liste toProcess;
while toProcess n'est pas vide do
    currentCell ← Premier (toProcess) ;
    n ← Niveau (currentCell);
    if (n>= minLevel) et (n <= maxLevel) then
        diviser ← non
        if Complexe (currentCell) then
            diviser ← oui
        else
            if (PasHomogene (currentCell) and (ForteCourbure
            (currentCell)) then
                diviser ← oui
            else
                Marquer currentCell comme une feuille compacte.
            end
        end
        if diviser then
            Commentaire : currentCell doit être divisée;
            S = {q1, q2, q3, ..., q8} ← Diviser (currentCell) ;
            foreach cellule qi en S do
                Ajouter qi dans toProcess
            end
        end
    end
end
```

Dans les zones de changement rapide de résolution (écart de niveau supérieur à 1 entre cellules voisines), l'algorithme 1 peut générer des configurations non 2-variétés comme le montre la figure 2a où le sommet de la cellule de gauche est relié à de trop nombreuses cellules voisines.

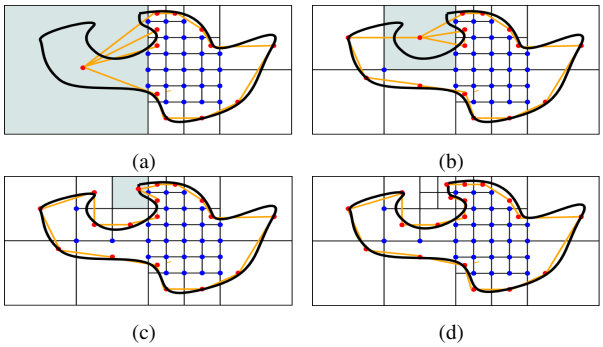


Figure 2 – *Raffinement progressif des cellules pour l'objet délimité en noir. Les sommets bleus sont à l'intérieur de l'objet, les sommets rouges sont les sommets duaux du maillage. L'algorithme 2 divise récursivement les cellules (en gris) qui contiennent une arête complexe, (d) il n'y a plus d'arête complexe et un maillage 2-variété est généré.*

Lorsqu'un changement de résolution important est présent localement, les facettes partagées par la cellule *compacte* et les cellules voisines sont analysées. Tant qu'il existe des facettes complexes, les cellules sont divisées récursivement comme illustre la figure 2. L'algorithme 2 décrit ce processus.

Algorithm 2: Algorithme de réparation locale.

```
input  : Ensemble T de cellules compactes c.
         Niveau maximal de l'octree maxLevel.
output: Soit A la liste des cellules à traiter.
Initialisation : A ← T
foreach cellule c ∈ A do
    foreach facette ∈ c do
        if facette est complexe then
            RaffinementOctree (c, Niveau (c), Niveau (c) +1)
            Ajouter les sous-cellules de c dans A ;
            Retirer c de A ;
        end
    end
end
```

3.2 Création et localisation des sommets

Afin de déterminer les sommets duaux qui seront créés dans une cellule, nous calculons les composantes connexes dans le graphe G, comme proposé par Wang *et al.*[14]. Comme une cellule peut contenir au plus quatre composantes connexes par rapport à ses sommets, notre algorithme peut insérer au plus quatre sommets duaux à l'intérieur d'une cellule. Les éléments de surface générés par cette règle sont illustrés dans la figure 3.
Ces éléments de surface sont équivalents à ceux produits par l'algorithme DMC [8]. Toutefois, quand deux cellules des cas 17 et 20 partagent une facette ambigüe, une arête non variété peut être créée. Pour résoudre ce problème, nous proposons un algorithme qui inverse la connectivité extraite afin de générer des éléments de surface qui soient 2-variété (voir algorithme 3).

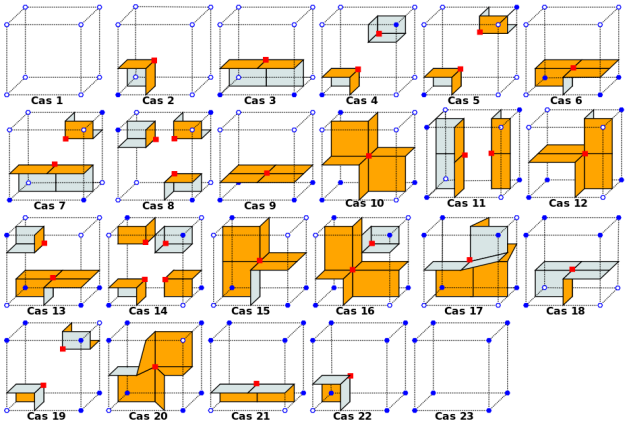
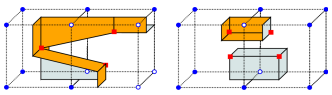


Figure 3 – *Éléments de surface générés par notre algorithme. L'intérieur de la surface est gris et l'extérieur est orange. Les sommets duaux sont les carrés rouges.*

Algorithm 3: Algorithme de génération des sommets.

input : Cellule C telle que $\{C \cap \lambda \neq \emptyset\}$
output: Cellule C avec des sommets duaux.
Initialement, les cellules ne sont pas marquées.
if `PasMarquee` (C) **then**
  Commentaire : CC Composantes connexes d'une cellule
  $CC \leftarrow$ `GetComposantsConnexeSolide` (C)
  Marque $\leftarrow$ Solide
  if $\mid CC \mid > 5$ **then**
    $CC \leftarrow$ `GetComposantsConnexeVide` (C)
    Marque $\leftarrow$ Vide
  end
  `GenereSommetsDuaux` (CC, Marque)
  $F_a \leftarrow$ `GetFacettesAmbigues` (C)
  foreach $f_i \in F_a$ **do**
    $c_a \leftarrow$ `GetAdjacentCellByFace` (f_i)
    if `PasMarquee` (c_a) **then**
      `MarquerComposante` (c_a, Marque)
    end
  end
end
else
  `GenereSommetsDuaux` (CC, Marque)
end

L'algorithme 3 assure que toutes les cellules qui partagent une facette ambigüe, utilisent les mêmes composantes (solide ou vide) pour générer ses morceaux de surface. Les éléments de surface produits pour les cas des cellules 17 et 20 sont illustrés dans les figures 4 et 5.

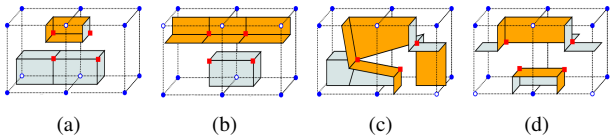


(a) (b)

Figure 4 – *Éléments de surface générés pour deux cellules adjacentes avec 6 sommets solides. (a) composants solides, (b) composants vides. Les sommets duaux sont en rouge.*

Localisation des sommets

La méthode de localisation proposée se base sur les barycentres des composantes connexes de λ à l'intérieur d'une cellule. Soit C une cellule telle que $\{C \cap \lambda \neq \emptyset\}$. Connaissant les dimensions de C, nous calculons sa masse M et son barycentre b_c. Un parcours du composant $S = C \cap \lambda$, permet de calculer son barycentre b_s et sa masse m_s, puis d'en déduire, en utilisant l'équation $Mb_c = m_s b_s + m_e b_e$, la masse m_e et le barycentre b_e du complémentaire de S .
Les barycentres b_s et b_e et les masses auxquelles ils correspondent sont utilisés pour estimer la position initiale du sommet dual et le vecteur normal à la surface. Finalement, le sommet dual est déplacé itérativement dans la direction de ce vecteur pour mieux approcher la surface (voir figure 6).



(a) (b) (c) (d)

Figure 5 – *Deux cellules qui partagent une facette ambigüe. (a,b) composants vides, (c,d) composants solides. Les sommets duaux sont notés en rouge.*

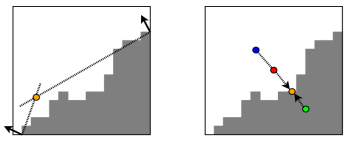


Figure 6 – *(Left) Dual node localisation avec QEF. (Right) Notre localisation. Barycentre de la cellule (rouge), barycentre de $F(x) \cap C$ (vert), barycentre de $(F(x) \cap C)^c$ (bleu), localisation estimée du sommet dual (orange).*

Nous avons considéré des méthodes telles que les fonctions d'erreur quadratique (QEF) [15]. Néanmoins, elles n'assurent pas que les sommets seront localisés sur la surface de l'objet volumique favorisant alors l'apparition d'autointersection en présence de plusieurs sommets dans une même cellule.

3.3 Génération de la connectivité

Pour construire la connectivité de la surface, nous utilisons la procédure proposeé par Ju *et al.* [7], consistant à parcourir récursivement les arêtes de l'octree à l'aide de trois méthodes : *CellProc*, *FaceProc* et *EdgeProc*.
La méthode *CellProc* considère une cellule c et parcourt toutes les cellules de l'octree qui traversent $\partial\lambda$. Elle fait huit appels aux sous-cellules de c, douze appels à la méthode *FaceProc* avec les sous-cellules qui partagent une facette commune et six appels à *EdgeProc* avec les sous-cellules qui partagent une demi-arête (voir figure 7).

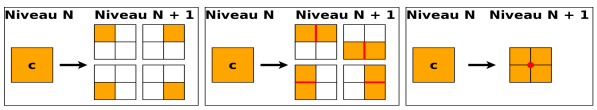
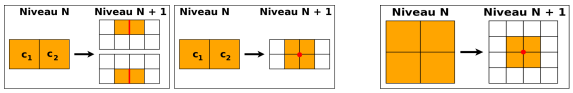


Figure 7 – *Méthode CellProc : Appels à CellProc (gauche), FaceProc (centre) et EdgeProc (droite) en 2D. Les facettes et sommets partagés sont en rouge.*

La méthode *FaceProc* considère deux cellules c_1 et c_2

partageant une facette f_c et fait quatre appels à *FaceProc* avec les sous-cellules partageant une sous-facette de f_c et quatre appels à *EdgeProc* avec les cellules qui partagent une arête contenue dans f_c (voir figure 8a).



(a) FaceProc. (b) EdgeProc.

Figure 8 – *(a) FaceProc : appel récursif à FaceProc (left), appel à EdgeProc (right). (b) EdgeProc : Appel recursif à EdgeProc. Les facettes et sommets partagés sont en rouge.*

Finalement, *EdgeProc* reçoit quatre cellules avec une arête en commun a_c et fait deux appels à elle-même avec les quatre sous-cellules qui partagent une demi-arête contenue dans a_c (voir figure 8b).
Seule la méthode EdgeProc génère des polygones en connectant les sommets duaux liés à une arête qui traverse $\partial\lambda$. EdgeProc crée des quadrangles lorsque les cellules ont la même résolution et des triangles dans le cas contraire.

4 Résultats et comparaison

Afin de tester notre algorithme, nous avons utilisé des données volumiques issues de la discrétisation de maillages surfaciques de très bonne qualité topologique et d'approximation par rapport aux surfaces originales. Les mesures de distance ont été calculées avec l'application Metro [16].

Résultats

Notre algorithme génère des surfaces multirésolution. La figure 9 montre une surface 2-variété fermée et adaptative, construite à partir d'un volume de taille $512 \times 512 \times 256$.

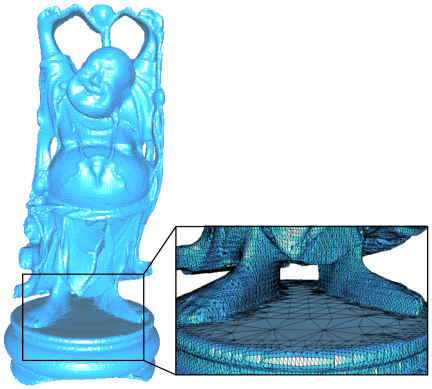


Figure 9 – *Surface adaptative avec 161004 facettes générée à partir d'un volume* $512 \times 512 \times 256$ *voxels.*

Il est possible d'obtenir des modèles simplifiés en utilisant différents seuils pour la courbure maximale dans le critère 2. Pour le modèle Armadillo, Le tableau 1 présente les écarts entre la surface initiale et les surfaces reconstruites à partir d'un volume discrétisé de taille 512^3, pour des valeurs de courbure allant de 0.1 jusqu'à 0.9. La figure 10 montre les surfaces obtenues pour les valeurs 0.1 et 0.9.

Courbure	# Triangles	Hausdorff	RMS	Time (secs)
0.1	89094	0.0043	0.15	9.1
0.3	42822	0.0091	0.21	8.3
0.5	27274	0.012	0.33	7.3
0.7	20740	0.014	0.41	6.5
0.9	16922	0.014	0.45	6.3

Tableau 1 – *Statistiques de la surface obtenue à partir du volume Armadillo (*512^3 *voxels) avec différentes valeurs de courbure. Les surfaces sont toujours des variétés fermées.*

Les temps d'exécution affichés incluent les temps de construction de l'octree, mais la plus grande partie du temps d'exécution est consacrée à l'évaluation des normales et des critères 1 et 2 directement sur les données volumiques.

Figure 10 – *Surfaces simplifiées par rapport à la courbure à partir d'un volume* 512^3 *voxels. A gauche, seuil de courbure* 0.1*, à droite, seuil de courbure* 0.9.

Un avantage important de l'algorithme 2 est de permettre un raffinement local ou progressif d'une surface en utilisant l'octree. Sur la figure 11, une surface simplifiée est progressivement raffinée afin d'obtenir une meilleure approximation dans la partie supérieure du modèle.

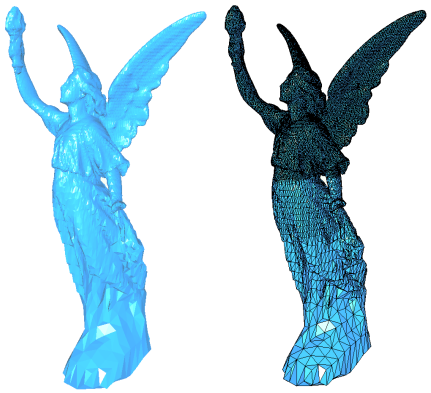
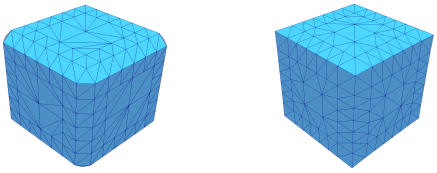


Figure 11 – *Raffinement progressif d'un maillage extrait à partir d'un volume* 512^3.

Comparaison

Pour évaluer notre algorithme, nous l'avons comparé avec l'algorithme DMC Adaptatif (ADMC) de Schaefer *et al* [10] et le Marching Tetrahedra Adaptif (AMT) appliqué

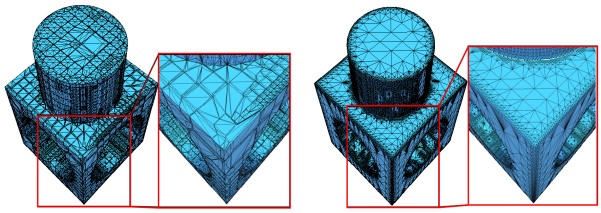
sur une grille duale, proposé par Manson *et al.* [11].
Par rapport à ADMC, notre algorithme est capable de générer des maillages adaptatifs qui approximent mieux les arêtes vives présentes dans les objets manufacturés (voir figure 12). En effet, même si ADMC génère une grille duale alignée sur les caractéristiques de $\partial\lambda$, il applique un algorithme de type MC qui localise les sommets du maillage sur les arêtes des cellules.



(a) (b)

Figure 12 – *Comparaison entre ADMC [10] (a) et notre algorithme (b) sur un objet "cube". L'utilisation des sommets duaux permet de mieux approximer les arêtes vives de $\partial\lambda$.*

Quant à l'algorithme AMT, il génère des maillages qui approximent très bien la surface mais la qualité des triangles est loin d'être optimale. De plus, les algorithmes basés sur MT ont l'inconvénient de générer trop de triangles. La figure 13 montre que la qualité des triangles construits par notre algorithme est meilleure, sans qu'il soit besoin d'appliquer un post-traitement.



(a) Reconstruction AMT. (b) Notre reconstruction.

Figure 13 – *Comparaison sur un modèle "block" : (a) AMT [11] (133512 facettes, angle minimal/maximal moyen de 16.16/100.19) et (b) notre algorithme (105060 facettes, angle minimal/maximal moyen de 31.65/90.88).*

Par rapport aux algorithmes de l'état de l'art, notre méthode offre un bon compromis entre la quantité de triangles, leur facteur de forme et la qualité de l'approximation obtenue.

5 Conclusions et perspectives

Nous avons présenté un algorithme pour extraire des surfaces 2-variété adaptées à la courbure des objets. Par rapport aux méthodes pré-existantes, notre approche est bien adaptée aux données volumiques et permet d'obtenir une bonne approximation sans aucune information préalable sur les caractéristiques de l'objet à modéliser.
Une des perspectives de nos travaux sera d'étendre notre algorithme pour générer des maillages tétraédriques ou hexaédriques nécessaires pour le calcul scientifique. De plus, l'utilisation d'un approche duale nous permet d'envisager des applications dans l'extraction simultanée de multiples surfaces multi-résolutions à partir d'un volume labellisé.

Remerciements

Cette recherche a été partiellement financée par le projet national PEPS INS2I CNRS ImageEar3D et le projet ANR Collaviz ANR-08-COSI-003. Nous remercions Stanford CGL et GeorgiaTech pour les modèles utilisés dans cet article.

Références

[1] J. Bloomenthal. Polygonization of implicit surfaces. *Comput. Aided Geom. Des.*, 5(4) :341–355, 1988.

[2] W.E. Lorensen et H.E. Cline. Marching cubes : A high resolution 3D surface construction algorithm. Dans *SIGGRAPH*, volume 87, pages 163–169, 1987.

[3] J. Wilhelms et A. Van Gelder. Octrees for faster isosurface generation. *ACM Trans. Graph.*, 11 :201–227, July 1992.

[4] L.P. Kobbelt, M. Botsch, U. Schwanecke, et H-P. Seidel. Feature sensitive surface extraction from volume data. SIGGRAPH, pages 57–66, 2001.

[5] G. Varadhan, S. Krishnan, T. Sriram, et D. Manocha. Topology preserving surface extraction using adaptive subdivision. *SGP*, 2004.

[6] M. Kazhdan, A. Klein, K. Dalal, et H. Hoppe. Unconstrained isosurface extraction on arbitrary octrees. SGP, pages 125–133, 2007.

[7] T. Ju, F. Losasso, S. Schaefer, et J. Warren. Dual contouring of hermite data. SIGGRAPH, pages 339–346, 2002.

[8] G.M. Nielson. Dual marching cubes. Dans *Visualization*, pages 489–496, 2004.

[9] S. Schaefer, T. Ju, et J. Warren. Manifold dual contouring. *IEEE Transactions on Visualization and Computer Graphics*, 13 :610–619, May 2007.

[10] S. Schaefer et J.D. Warren. Dual marching cubes : Primal contouring of dual grids. *Computer Graphics Forum*, pages 195–201, 2005.

[11] J. Manson et S. Schaefer. Isosurfaces over simplicial partitions of multiresolution grids. *Computer Graphics Forum*, 29(2) :377–385, 2010.

[12] T. Lewiner, V. Mello, A. Peixoto, S. Pesco, et H. Lopes. Fast generation of pointerless octree duals. Dans *Computer Graphics Forum*, volume 29, pages 1661–1669, july 2010.

[13] F. Flin. Adaptative estimation of normals and surface area for discrete 3d objects. *IEEE Transactions on Image Processing*, 14(5) :585–596, 2005.

[14] C.C.L. Wang et Y. Chen. Layered depth-normal images for complex geometries-part two : manifold-preserved adaptive contouring. *ASME IDETC/CIE Conference*, 2008.

[15] M. Garland et P.S. Heckbert. Surface simplification using quadric error metrics. SIGGRAPH, pages 209–216, 1997.

[16] P. Cignoni, C. Rocchini, et R. Scopigno. Metro : measuring error on simplified surfaces. Dans *Computer Graphics Forum*, volume 17, pages 167–174, 1998.