Université de Liège
Faculté des Sciences Appliquées
Département d'Aérospatiale et Mécanique

Génération de maillages surfaciques pour la création de modèles biomécaniques du cerveau

Vinciane d'Otreppe de Bouvette

Travail de fin d'études présenté en vue de l'obtention
du grade d'Ingénieur Civil Électromécanicien (orientation aérospatiale)

Promoteur : Prof. JP. Ponthot

Année académique 2007 - 2008

Remerciements

Je voudrais avant toute chose, remercier les personnes sans qui le présent travail n'aurait jamais vu le jour.

Je remercie tout d'abord le Professeur JP. Ponthot, mon promoteur, pour son enthousiasme et sa disponibilité malgré son emploi du temps chargé.

Je tiens ensuite à exprimer ma plus sincère reconnaissance à Romain Boman pour son aide constante, ses nombreux conseils ainsi que pour sa relecture minutieuse du présent document.

Je ne voudrais pas oublier Lara Vigneron qui m'aidée à réaliser la dernière partie de ce travail.

D'avance, j'adresse mes remerciements aux membres du jury pour le temps et l'intérêt qu'ils porteront à la lecture et l'évaluation de mon travail.

Je terminerais en remerciant mes parents qui m'ont patiemment soutenu tout au long de mes études.

Résumé

Le planning préopératoire en neurochirurgie est réalisé à partir d'images structurelles et fonctionnelles du patient. Cependant l'intervention chirurgicale réalisée à partir de ce planning se fonde sur l'hypothèse que les structures anatomiques ne bougent pas pendant l'opération. En réalité, pendant l'opération, le cerveau se déforme de sorte que les images préopératoires, sur lesquelles se base le neurochirurgien, ne correspondent plus à la réalité anatomique du patient. Un approche envisagée pour résoudre ce problème est de modéliser, par la méthode des éléments finis, le comportement mécanique du cerveau pendant l'opération. Ceci permettrait de fournir au chirurgien, tout au long de l'intervention, des images mises à jour de la même qualité que les images préopératoires. La création du maillage du modèle biomécanique du cerveau à partir d'images IRM préopératoires fait l'objet de ce travail.

La première et majeure partie du travail est consacrée à l'élaboration d'un mailleur surfacique. L'extraction de la surface, définie soit à l'aide d'une fonction implicite, soit à l'aide d'une image tridimensionnelle, se fait au moyen de la méthode Marching Tetrahedra. Le maillage ainsi obtenu étant inadapté au calcul éléments finis, il est ensuite simplifié et amélioré de manière à obtenir un maillage de bonne qualité. Afin de découpler la taille des mailles générées de la distance inter-slices, une méthode d'interpolation entre images a également été implémentée.

La seconde partie est l'intégration du mailleur surfacique créé dans l'embryon du système de neuronavigation en cours de développement à l'ULg. Par simplicité, le cerveau est considéré comme un milieu homogène et seul le phénomène du *brain shift* est modélisé. De plus, les images IRM sont supposées avoir été préalablement corrigées, segmentées et recalées. La figure 3 montre le résultat du calcul éléments finis du *brain shift*, réalisé dans Metafor.

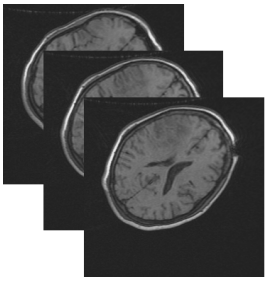


Fig. 1 – *Coupes IRM*

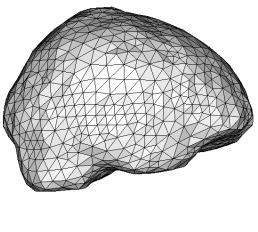


Fig. 2 – *Maillage surfacique*

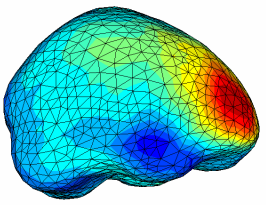


Fig. 3 – *Calcul éléments finis*

Table des matières

Introduction

Ce travail s'inscrit dans un projet de recherche qui vise à modéliser, par la méthode des éléments finis, le comportement mécanique du cerveau lors d'opérations neurochirurgicales. Dans ce cadre, la génération automatique d'un maillage à partir d'images IRM du corps à étudier est une tâche difficile mais indispensable.

Contexte

Une localisation précise de la tumeur est essentielle lors de l'enlèvement de celle-ci, à la fois pour éviter de toucher les tissus sains et pour pratiquer une résection tumorale complète. Dans ce cadre, l'imagerie médicale joue un rôle important. Tout d'abord, le neurochirurgien planifie son intervention sur base d'images structurelles et fonctionnelles (principalement du type IRM). Ensuite, pendant l'intervention, les systèmes de chirurgie guidée par l'image permettent de faire le lien entre les images préopératoires et le patient, en salle d'opération. C'est-à-dire que le neurochirurgien connaît, à partir d'un point désigné sur le patient par un outil, le point correspondant dans ses images préopératoires.

Explication du problème

Le problème est que lors de l'intervention, ces images préopératoires deviennent rapidement obsolètes et ne correspondent plus à la réalité anatomique du patient. En effet, lors de l'ouverture de la boîte crânienne le liquide céphalo-rachidien s'écoule et le cerveau à tendance à s'affaisser sous l'effet de la gravité (*brain shift*). Cette déformation est encore accentuée lorsque le neurochirurgien incise le cerveau pour créer une voie d'accès à la zone cible et résèque les tissus tumoraux.

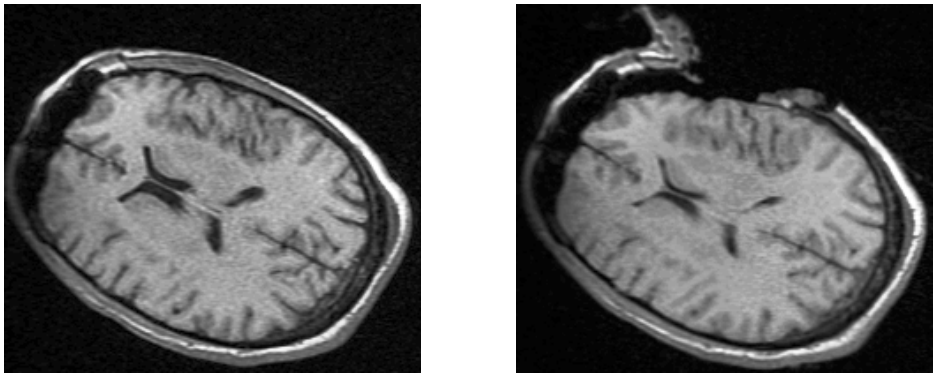


Fig. 4 – *Déformations du cerveau induites par une intervention chirurgicale [21].*

Les erreurs de navigation dues à la déformation du cerveau au cours de l'intervention pourraient être réduites si l'on pouvait acquérir, tout au long de l'opération, des images mises à jour de la même qualité que les images préopératoires. Ceci explique le développement depuis quelques années d'appareils à imagerie par résonance magnétique utilisables durant l'opération (IRMi). L'IRM a fait son entrée en salle d'opération en 1994 au Brigham and Women's Hospital de Boston. A l'heure actuelle, il n'en existe que deux exemplaires en Belgique : à l'hôpital Erasme (ULB) et, depuis début 2007, au CHU de Liège (fig. 5). L'IRM interventionnelle représente un progrès important dans la prise en charge des pathologies cérébrales car elle offre au neurochirurgien la possibilité de compléter et de réactualiser les informations préopératoires pendant son intervention. Cependant, parce que le champ magnétique permanent délivré par de tels appareils est environ dix fois plus faible que celui de l'IRM conventionnelle, les images fournies sont de résolution et de contraste inférieurs que les images préopératoires (fig. 6).

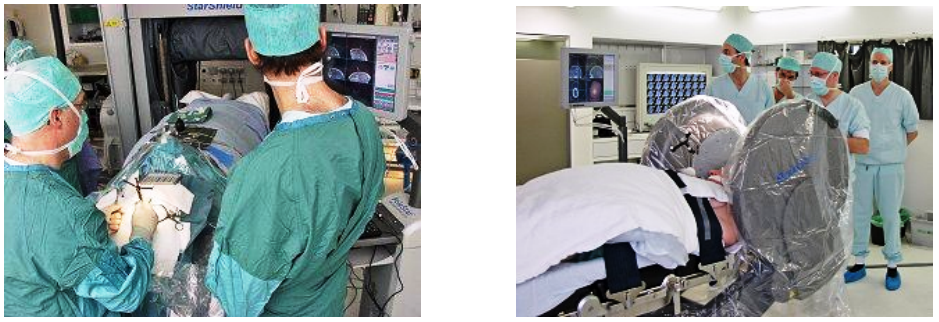
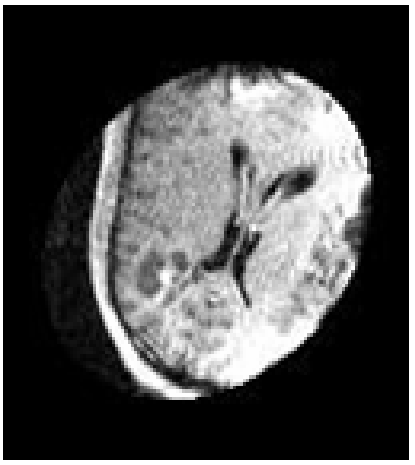


Fig. 5 – *IRM interventionnelle installée au CHU de Liège : système à bas champ magnétique (0.15 Tesla) Odin (Medtronics) PoleStar N20.*

(a) Image préopératoire



(b) Image intraopératoire (PoleStar (0.15 Tesla), CHU Liège)

Fig. 6 – *Les images intraopératoires sont de moins bonne résolution que les images préopératoires.*

Méthode utilisée pour résoudre le problème

L'objectif est de pouvoir fournir au chirurgien, tout au long de son opération, des images de la même qualité et de la même modalité que les images préopératoires. L'idée donc est de mettre à jour, de déformer, les images préopératoires de bonne qualité à l'aide d'informations (champ de déformation) recueillies sur les images intraopératoires de qualité limitée. On parle de *recalage non rigide d'images.*

La méthode de recalage utilisée correspond à un modèle physique déformable. Un modèle biomécanique du cerveau est tout d'abord créé à partir des images préopératoires de bonne qualité. Ce modèle consiste en un maillage volumique tétraédrique associé d'une ou plusieurs lois de comportement et des conditions aux limites appropriées. En cours d'opération, des images intraopératoires sont acquises. On en déduit un champ de déplacement par la méthode des surfaces actives [21, 33, 65]. Ce champ est alors appliqué sur le maillage du modèle biomécanique. Un calcul éléments finis permet déterminer le champ de déplacement en tout point du cerveau. La connaissance de ce-dernier permet de déformer les images préopératoires et de fournir au chirurgien des images intraopératoires de haute qualité pour guider son positionnement dans le crâne du patient et augmenter la qualité de ses interventions.

Un autre intérêt de la méthode est de déformer des images obtenues par d'autres modalités, qui ne sont pas disponibles pendant l'opération (l'IRM fonctionnelle par exemple). Les images préopératoires de différentes modalités sont fusionnées, c'est à dire que les informations pertinentes sont extraites, et le résultat est déformé pour correspondre à l'état réel du cerveau du patient pendant l'opération.

Énoncé du travail de fin d'études

Ce mémoire porte sur la création du maillage du modèle biomécanique du cerveau à partir d'images IRM et son intégration dans l'embryon du système de neuronavigation en cours de développement à l'ULg.

Hypothèses

Pour se simplifier la tâche, nous supposerons que les images IRM tridimensionnelles constituées de voxels plus ou moins gris ont été préalablement filtrées et corrigées par des algorithmes de traitement d'images développés par le Laboratoire d'Exploitation des Signaux et des Images (INTELSIG, ULg) du Professeur Verly. Nous supposerons en outre que la segmentation des images a déjà été effectuée et que les images intraopératoires ont été recalées sur les images préopératoires.

De plus, le cerveau sera considéré comme un milieu homogène. Bien entendu, le cerveau réel est très éloigné de cette situation simpliste. Il est principalement constitué de la matière grise qui renferme les neurones, de la matière blanche et du liquide céphalo-rachidien. Il contient également de nombreux vaisseaux et des parois méningées. Chacun de ces composants possède des propriétés mécaniques différentes qu'il faudrait inclure pour que le modèle créé puisse reproduire les déformations réelles.

Étapes de l'élaboration du mailleur

A partir des images segmentées, un maillage surfacique sera construit par la méthode Marching Tetrahedra. Le maillage ainsi obtenu est inadapté pour effectuer un calcul éléments finis et devra donc être simplifié et amélioré de manière à obtenir un maillage de qualité. Afin de découpler la taille des mailles générées de la distance inter-slices, une méthode d'interpolation entre images sera également implémentée.

En pratique, le logiciel *Isosurf* [61] permet déjà de réaliser un maillage surfacique au départ d'images segmentées. Cependant, le code source d'*Isosurf* n'étant pas disponible, il est difficile de comprendre les cas ne permettant pas d'aboutir à un maillage correct et il est impossible de l'améliorer. De plus, bien que géométriquement corrects, les maillages générés ne sont parfois pas optimaux pour leur future utilisation par la méthode des éléments finis. En particulier, *Isosurf* ne permet pas de contrôler la taille des mailles en fonction de la géométrie locale du cerveau. Enfin rappelons que l'objectif futur est de créer un modèle du cerveau prenant en compte les différentes substances de propriétés mécaniques différentes (matière blanche, matière grise, liquide céphalo-rachidien,...). Or *Isosurf* ne permet pas de générer, dans chacune de ces régions, un maillage compatible avec les régions voisines. Notons également d'autres problèmes annexes : le format RAW utilisé par Isosurf est un format incomplet : la taille des données, le nombres de coupes et leurs dimensions ne sont pas contenus dans la définition de l'image et doivent être donc retenus à part ; le format de sortie (OFF) n'est pas pratique car il nécessite l'utilisation Geomview (vielle interface, non disponible pour PC/Windows) et il n'y a pas de gestion d'erreurs. Toutes ces limitations justifient l'implémentation d'un mailleur surfacique au sein de l'ULg. L'objectif de ce travail de fin d'études n'est pas de faire mieux qu'*Isosurf* mais de construire une base solide d'un mailleur surfacique qui permettra de nombreuses

améliorations futures.

Ce mailleur surfacique sera implémenté en Python, par facilité, mais devra, dans un travail futur être retranscrit en C++ afin d'accroître sa vitesse d'exécution. La visualisation et la manipulation des images sera faite à l'aide de VTK [52] et de Qt.

Application biomédicale

Le mailleur surfacique développé dans la première partie du travail sera ensuite utilisé pour générer le maillage du modèle éléments finis du cerveau. A partir du maillage de la surface frontière du cerveau, un maillage volumique tétraédrique sera obtenu en utilisant un des logiciels TetGen ou Gmsh. Un calcul éléments finis pourra alors être réalisé à l'aide de Metafor [45], le logiciel éléments finis en grandes déformations développé au LTAS-MN^{2}L.

Structure du travail de fin d'études

Ce travail est subdivisé en trois parties, présentées suivant l'ordre dans lequel elles ont été réalisées.

Dans la **première partie**, nous laissons les images médicales de côté pour se concentrer directement sur l'obtention d'un maillage de surface de bonne qualité.

Nous commencerons par définir, à l'aide de fonctions implicites $f(x, y, z) = 0$, différents objets qui nous permettrons de valider chaque étape de l'algorithme (chap. 2).

Dans le chapitre 3, une approximation de la surface frontière du solide sous forme d'une triangulation sera construite au moyen de la méthode d'extraction de la surface Marching Tetrahedra.

Les méthodes d'extraction de la surface conduisent généralement à des modèles polygonaux complexes, comportant un nombre de triangles prohibitif pour une utilisation dans un code de calcul. C'est pourquoi des techniques de simplification ou de décimation des maillages de surfaces devront être appliqués dans le chapitre 4.

Dans le contexte des simulations numériques basées sur la méthode éléments finis, la qualité en forme et en taille des éléments ont beaucoup d'importance car elles influent sur la précision des résultats. Dans le chapitre 5, nous améliorons la topologie du maillage en supprimant les noeuds n'ayant que 3 ou 4 triangles voisins. La qualité du maillage est encore améliorée dans le chapitre 6 par un déplacement des noeuds du maillage sur la surface.

Les images médicales seront introduites dans la **seconde partie**. Nous commencerons par montrer comment la connaissance de la fonction implicite peut être remplacée par celle de la carte de distance de l'image tridimensionnelle médicale. Le calcul de la carte de distance d'un image binaire sera tout d'abord introduit en considérant une image 3D à voxels cubiques (chap. 8). Nous traiterons ensuite le cas d'une image médicale où la résolution est inférieure dans le sens perpendiculaire aux tranches de telle sorte qu'une méthode d'interpolation entre coupes est nécessaire (chap. 9).

Dans la **dernière partie** du travail de fin d'études, nous partirons de coupes IRM d'un cerveau pour aboutir à une simulation éléments finis dans Metafor, en passant par chacun des maillons de la chaîne de la figure 7. Notons que chacune des étapes illustrées à la figure 7 pourrait constituer un travail de fin d'études en soi. L'objectif de celui-ci est réaliser l'entièreté de cette chaîne, c'est-à-dire, d'obtenir un maillage tétraédrique de bonne qualité au départ d'images segmentées, en sachant que chacun des maillons de la chaîne sera amélioré dans un travail ultérieur.

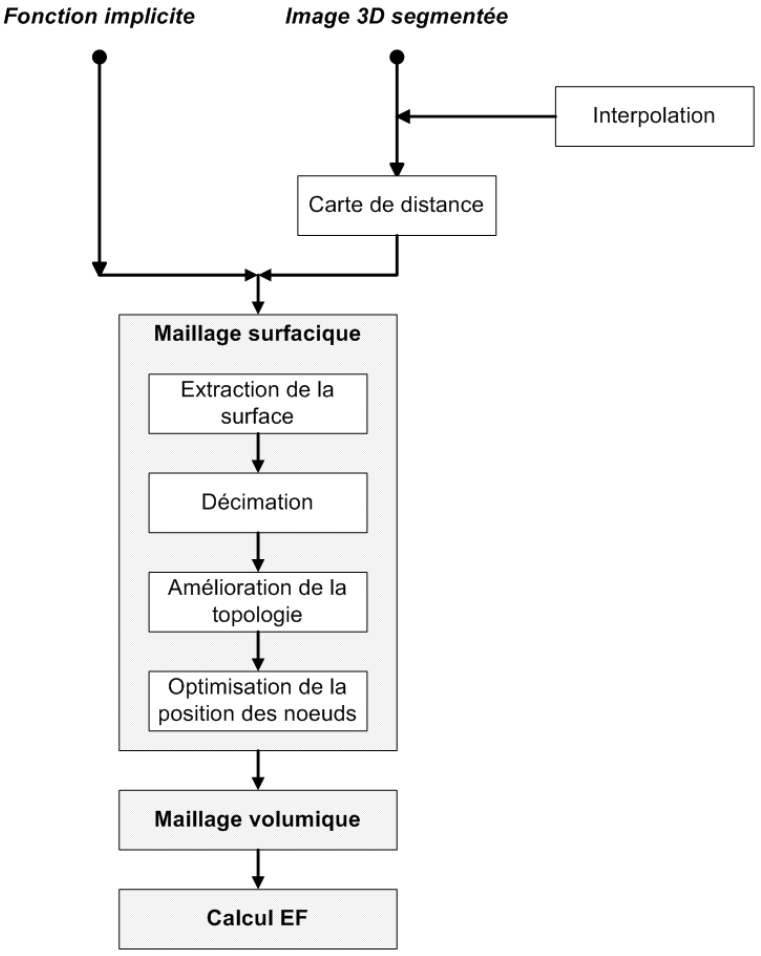


FIG. 7 – *Organigramme du travail de fin d'études*

Première partie

Triangulation d'une surface implicite

CHAPITRE 1

Introduction

La littérature sur la génération de maillages surfaciques à partir de fonctions implicites est abondante. Cependant, les algorithmes présentés dans la littérature sont généralement développés pour la visualisation scientifique et non le calcul éléments finis. Or, les critères de qualités imposés sur un maillage dépendent étroitement du contexte d'application. Ainsi, dans le cas de la méthode des éléments finis, la qualité du maillage est un critère beaucoup plus important que dans le contexte de la visualisation.

L'extraction de la surface ne permet pas de générer directement un maillage de la qualité requise (chap. 3). Un travail important a donc été réalisé afin d'obtenir un maillage de bonne qualité. En pratique, le mailleur surfacique implémenté contient quatre parties distinctes :

1. L'extraction de la surface à partir de sa définition sous forme d'une fonction implicite (chap. 3) ;
2. Un algorithme permettant de réduire le nombre de mailles (chap. 4) ;
3. Un algorithme permettant d'améliorer la topologie du maillage (chap. 5) ;
4. Un algorithme permettant d'améliorer la qualité des triangles de surface en déplacant les noeuds du maillage le long de celle-ci (chap. 6).

CHAPITRE 2

Surfaces implicites

Dans ce chapitre, nous définissons les surfaces "tests" qui seront utilisées tout au long de ce travail.

2.1 Intérêt et définition

Les surfaces et volumes implicites sont fréquemment utilisés pour la modélisation dans diverses applications scientifiques (par exemple, dans les systèmes de CAO). Le problème posé est de déduire une représentation du solide par sa frontière, ce qui revient, par exemple, à obtenir une triangulation de la surface définissant la frontière. Outre la visualisation, l'intérêt de la triangulation des surfaces implicites vient de la modélisation par éléments finis.

Dans le contexte biomédical, les systèmes d'imagerie médicale telles l'IRM fournissent des données volumétriques. Dans ces dernières, les voxels représentant une structure anatomique particulière ou un tissu biologique ont tous, approximativement, la même valeur si on considère que ces images ont été correctement filtrées (problème complexe supposé résolu dans le cadre de ce travail de fin d'études). Le problème de la triangulation d'une isosurface peut alors être rapproché du problème de la triangulation d'une surface implicite.

Une **surface implicite** réelle peut être formellement définie comme $f(x, y, z) = 0$. Une **isosurface** est un ensemble de points semblable pour lesquels $f(x, y, z) = c$ où c est une iso-valeur de la surface. La fonction f partitionne l'espace en deux sous-domaines, intérieur et extérieur, selon le signe de la fonction.

2.2 Présentation des surfaces implicites utilisées

Afin de tester le mailleur surfacique décrit aux chapitres 3 à 6, différentes surfaces implicites ont été maillées[1]. Nous verrons dans les chapitres suivants les problèmes liés à la triangulation de chacune de ces géométries.

La sphère (fig. 2.1)

$$f(x,y,z) = 20^2 - x^2 - y^2 - z^2 = 0$$

avec $(x,y,z) \in ([-25,25],[-25,25],[-25,25])$

Le cône arrondi (fig. 2.2)

$$f(x,y,z) = 0.5\,\sin(0.1\,x) + \sqrt{(0.1\,x)^2 + (0.1\,y)^2} - 1.5 + 0.01((0.1\,x)^6 + (0.1\,y)^6 + (0.1\,z)^6 - 1) = 0$$

avec $(x,y,z) \in ([-30,30],[-30,30],[-30,30])$

Le parallélépipède (fig. 2.3)

$$f(x,y,z) = \left(\frac{x}{30}\right)^6 + \left(\frac{x}{30} + \frac{y}{10}\right)^6 + \left(\frac{z}{30}\right)^6 - 1 = 0$$

avec $(x,y,z) \in ([-35,35],[-35,35],[-35,35])$

Le maximum entre un cube et une sphère (fig. 2.4)

$$f(x,y,z) = \max\left\{\left(\frac{x}{3} - 7\right)^2 + \left(\frac{y}{3} - 7\right)^2 + \left(\frac{z}{3} - 7\right)^2 - 225,\ \left(\frac{x}{3}\right)^6 + \left(\frac{y}{3}\right)^6 + \left(\frac{z}{3}\right)^6 - 9^6\right\} = 0$$

avec $(x,y,z) \in ([-32,32],[-32,32],[-32,32])$

Le minimum entre un cube et une sphère (fig. 2.5)

$$f(x,y,z) = \min\left\{\left(\frac{x}{2} - 7\right)^2 + \left(\frac{y}{2} - 7\right)^2 + \left(\frac{z}{2} - 7\right)^2 - 144,\ \left(\frac{x}{2}\right)^6 + \left(\frac{y}{2}\right)^6 + \left(\frac{z}{2}\right)^6 - 9^6\right\} = 0$$

avec $(x,y,z) \in ([-24,42],[-24,42],[-24,42])$

Le tore (fig. 2.6)

$$f(x,y,z) = \left(\left(\frac{x}{10}\right)^2 + \left(\frac{y}{10}\right)^2 + \left(\frac{z}{10}\right)^2 + 1.6^2 - 0.4^4\right)^2 - 4\,1.6^2\left(\left(\frac{x}{10}\right)^2 + \left(\frac{y}{10}\right)^2\right) = 0$$

avec $(x,y,z) \in ([-30,30],[-30,30],[-30,30])$

La multiplication de deux sphères (fig. 2.7)

$$f(x,y,z) = \left(\left(\frac{x}{15}\right)^2 + \left(\frac{y}{15}\right)^2 + \left(\frac{z}{15}\right)^2 - 1\right) \times \left(\left(\frac{x}{15}\right)^2 + (\left(\frac{y}{15}\right) - 1.15)^2 + \left(\frac{z}{15}\right)^2 - 0.5\right) - 0.015 = 0$$

avec $(x,y,z) \in ([-30,30],[-20,40],[-30,30])$

[1] La plupart des fonctions implicites présentées sont inspirées de [56]

La géométrie basée sur le nombre d'or (fig. 2.8)

$$f(x,y,z) = -2 - \cos\left(\left(\frac{x}{4}\right) + T\left(\frac{y}{4}\right)\right) - \cos\left(\left(\frac{x}{4}\right) - T\left(\frac{y}{4}\right)\right) - \cos\left(\left(\frac{y}{4}\right) + T\left(\frac{y}{4}\right)\right) -$$

$$\cos\left(\left(\frac{y}{4}\right) - T\left(\frac{z}{4}\right)\right) - \cos\left(\left(\frac{z}{4}\right) - T\left(\frac{z}{4}\right)\right) - \cos\left(\left(\frac{z}{4}\right) + T\left(\frac{x}{4}\right)\right) = 0$$

où $T = \frac{1+\sqrt{5}}{2}$ est le nombre d'or et avec $(x,y,z) \in ([-20,20],[-20,20],[-20,20])$

Pour créer *le maximum entre un cube et une sphère*, *le minimum entre un cube et une sphère* et *la multiplication de deux sphères*, nous avons fait appel à la géométrie constructive des solides (CSG), qui permet de créer des formes complexes en combinant différents solides. Dans ce cadre, prendre le maximum équivaut à une *intersection*, prendre le minimum réalise une *fusion*, ce qui est différent de l'*union* que est effectuée au moyen d'une multiplication. Nous verrons en effet que *la multiplication de deux sphères* présente une surface interne, contrairement au *minimum entre un cube et une sphère*.

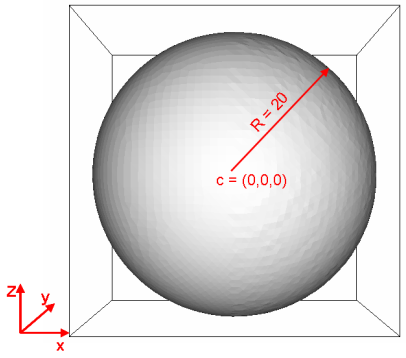


FIG. 2.1 – *La sphère*

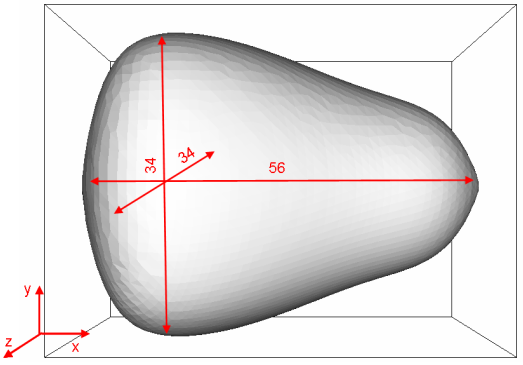


FIG. 2.2 – *Le cône arrondi*

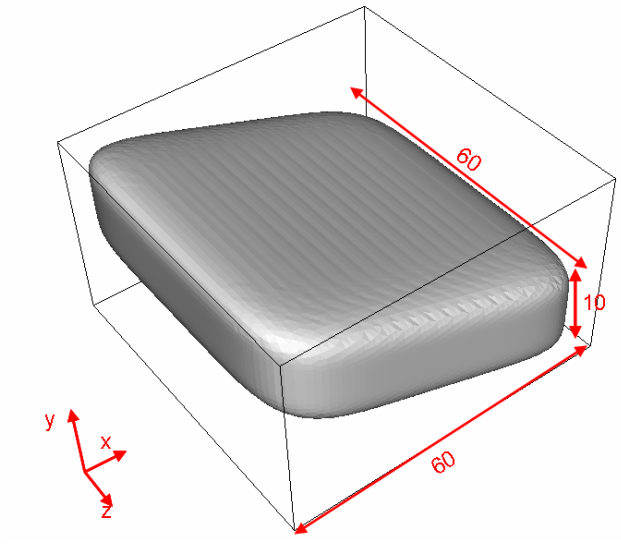


FIG. 2.3 – *Le parallélépipède*

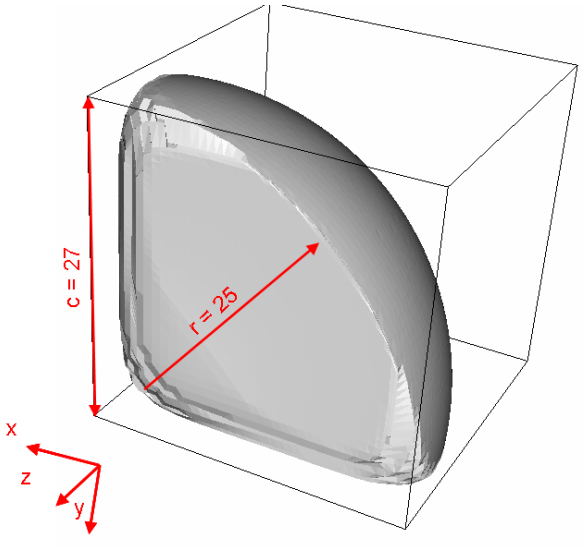


FIG. 2.4 – *Le maximum entre un cube et une sphère*

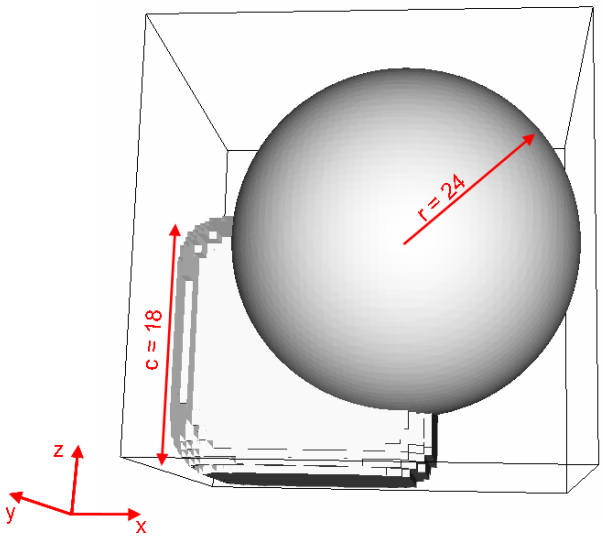


FIG. 2.5 – *Le minimum entre un cube et une sphère*

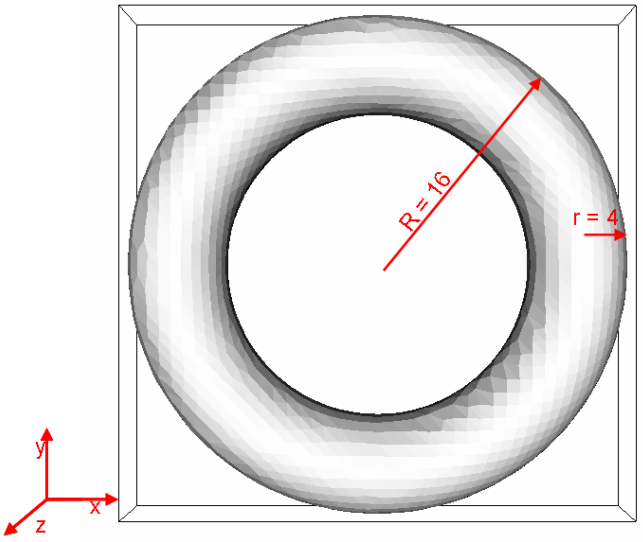


FIG. 2.6 – *Le tore*

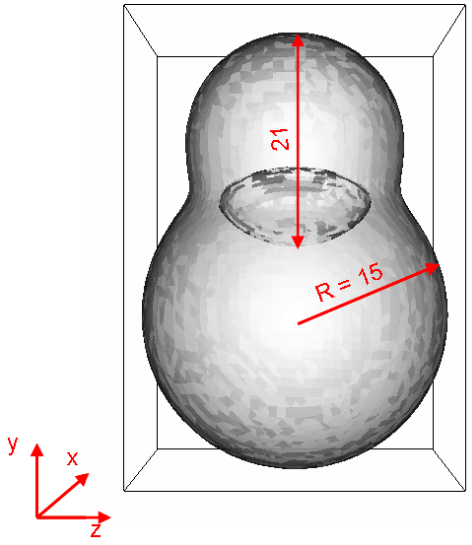


FIG. 2.7 – *La multiplication de deux sphères (la surface interne est visible par transparence)*

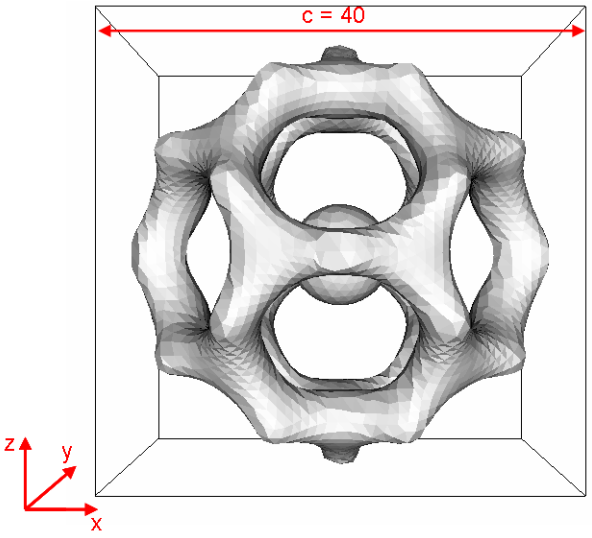


FIG. 2.8 – *La géométrie basée sur le nombre d'or*

Ces surfaces ont été choisies avec soin, chacune d'elles apportant une difficulté supplémentaire que notre algorithme devra traiter. Elles ont été classées par ordre de difficulté croissante. *La sphère* est la géométrie la plus simple, *le cône arrondi* est à peine plus complexe. Avec *le parallélépipède*, des angles (arrondis) sont introduits. *Le maximum entre un cube et une sphère* est plus complexe car il présente un angle vif. La frontière entre le cube et la sphère du *minimum entre un cube et une sphère* peut également poser problème. La spécificité du *tore* est qu'il s'agit d'un objet présentant un trou. *La multiplication de deux sphères* a la particularité de posséder une surface interne. Enfin, *la géométrie basée sur le nombre d'or* est complexe par sa forme et par le fait qu'elle est composée de deux objets distincts.

CHAPITRE 3

Extraction de la surface

Le problème posé dans ce chapitre est de représenter le plus fidèlement possible un volume implicite, grâce à une triangulation de sa surface frontière.

3.1 Etat de l'art

Les algorithmes de triangulation de surfaces implicites ou d'isosurfaces proposés dans la littérature construisent une approximation de la surface au moyen d'un recouvrement de l'espace en cellules. L'intersection de la surface avec chaque cellule est ensuite approchée par des triangles. Aucune information sur la topologie (nombres de trous, nombre de solides distincts, ...) n'est requise.

Les différentes méthodes se distinguent par la manière dont l'espace est divisé.

Dans l'algorithme du **Marching Cubes**, inventé par Bill Lorensen et Harvey Cline [11, 12, 14, 18, 28, 37, 40, 66], l'espace est partitionné en hexaèdres (idéalement des cubes). Le principe est de calculer les différentes configurations que peut prendre l'isosurface dans le cube élémentaire, selon la répartition des intersections de l'isosurface sur les arrêtes du cube. Un cube ayant 8 sommets et chaque sommet pouvant prendre deux états (suivant qu'il se situe à l'intérieur ou à l'extérieur de l'isosurface), il y'a $2^8 = 256$ configurations possibles, mais la présence de nombreux cas symétriques permet de se ramener à 16 configurations de base (fig. 3.1). Chaque configuration correspond à un ensemble de polygones tracés à l'intérieur du volume. A la figure 3.1, nous voyons que ces polygones ont entre 3 (triangle) et 6 côtés (hexagone). Ceux-ci doivent ensuite être triangulés afin d'obtenir un maillage constitué exclusivement de triangles. (fig. 3.2 (a)).

La difficulté majeure de la méthode du Marching Cubes est qu'une ambiguïté topologique apparaît lorsqu'une cellule de partitionnement contient une face ambiguë, dans laquelle les sommets de signes contraires sont diagonalement opposés deux à deux (fig. 3.2 (b)). De nombreux articles proposent des solutions à ce problème d'ambiguïté [11, 14, 37, 40]. Par contre, la méthode du Marching Cubes a l'avantage de générer moins de triangles que les méthodes décrites ci-après, elle est donc plus rapide et utilise moins de ressources machine.

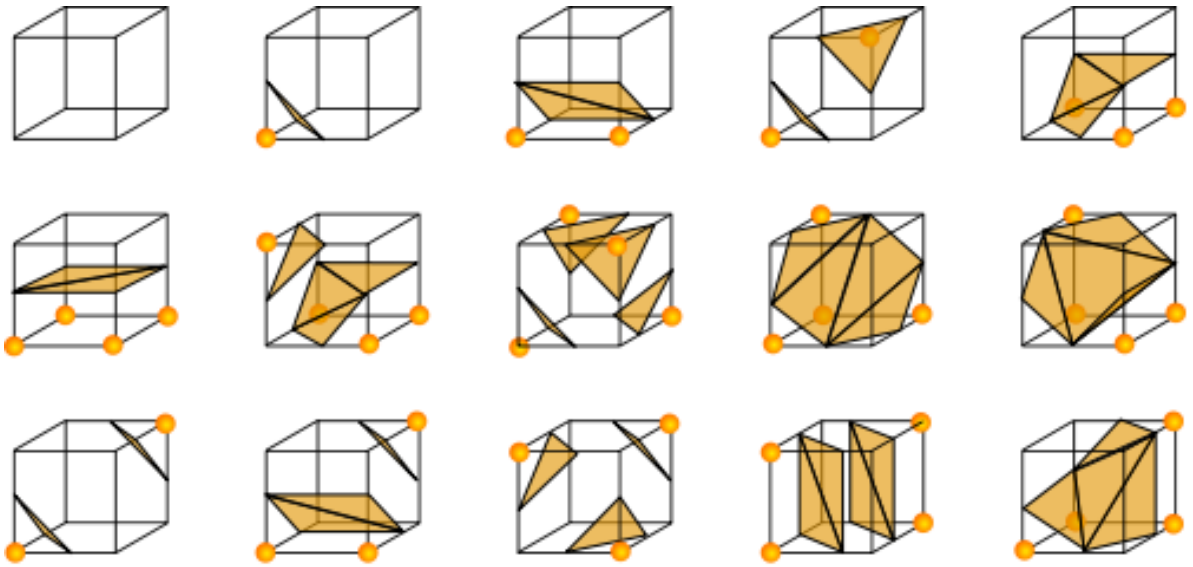
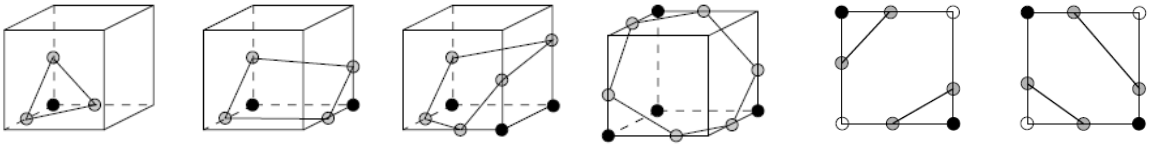


Fig. 3.1 – *Les 16 configurations de base d'intersection entre l'isosurface et le cube élémentaire de l'algorithme du Marching Cubes [66].*



(a) Quelques polygones créés (b) Ambiguïté topologique

Fig. 3.2 – *Algorithme du Marching Cubes [24].*

La méthode du **Marching Tetrahedra** [4, 28, 42, 43, 66] résout le problème d'ambiguïté topologique du Marching Cubes en subdivisant chaque cellule hexaédrique en 5, 6 ou 12 tétraèdres suivant l'algorithme choisi. Un autre avantage pour l'implémentation de la méthode du Marching Tetrahedra par rapport à celle du Marching Cubes est que les polygones créés par intersection avec la surface ont au maximum 4 côtés (quadrangle). Or on ne peut trianguler un quadrangle que de deux manières différentes, contrairement à l'hexaèdre pour lequel if fallait considérer 14 triangulations possibles. L'algorithme est donc plus simple à implémenter. Le choix de la subdivision de l'hexaèdre du Marching Cubes en 5 [29], 6 [43] ou 12 [24] tétraèdres dépend de facteurs tels le nombre de triangles générés, la difficulté d'implémentation, l'approximation de la surface désirée, l'efficacité en termes de vitesse, etc.

D'autres auteurs proposent un partitionnement du domaine d'étude en un réseau cubique centré [55, 61]. Les tétraèdres ainsi obtenus sont plus symétriques et plus réguliers. L'utilisation d'un réseau cubique faces centrées est moins courante [9, 59].

Une dernière possibilité est de traquer la surface par propagation [1, 17]. Une cellule initiale intersectant la surface est trouvée et de nouvelles cellules s'ajoutent à partir des cellules existantes intersectées par la surface. Cependant, la localisation de la cellule initiale peut s'avérer coûteuse.

3.2 Choix de l'algorithme du Marching Tetrahedra

Des algorithmes présentés ci-dessus, l'algorithme du Marching Cubes et celui qui génère le moins de triangles, qui est le plus rapide et qui donne les meilleurs résultats visuels. Pourtant, notre choix s'est porté sur l'algorithme du Marching Tetrahedra, et ce, pour les raisons suivantes :

- L'algorithme du Marching Tetrahedra est plus facile à implémenter que l'algorithme du Marching Cubes, d'une part parce qu'il ne présente pas le problème des cas ambigus, d'autre part parce que le nombre de configurations possibles d'intersection entre le volume et l'isosurface est plus faible si ce volume est un tétraèdre ($2^4 = 16$, ce qui donne 8 configurations en tenant compte des cas symétriques) au lieu d'être un cube ($2^8 = 256$, 16 configurations en tenant compte des cas symétriques).
- L'algorithme du Marching Tetrahedra est celui qui est utilisé dans le mailleur surfacique *Isosurf*, qui donne de très bons résultats.
- Le code étant plus simple, il sera plus facile à modifier dans le cadre de futurs projets de recherche. En particulier, l'algorithme du Marching Tetrahedra a l'avantage de pouvoir être couplé à d'autres méthodes telles le Vertex Clustering. C'est d'ailleurs ce qui est réalisé dans *Isosurf*.

Des différentes manières possibles de subdivision d'un cube en tétraèdres, nous avons choisi la subdivision de la figure 3.3, où chaque cube est divisé en 6 tétraèdres identiques. Selon, [43], il s'agit de la subdivision qui donne les meilleurs résultats.

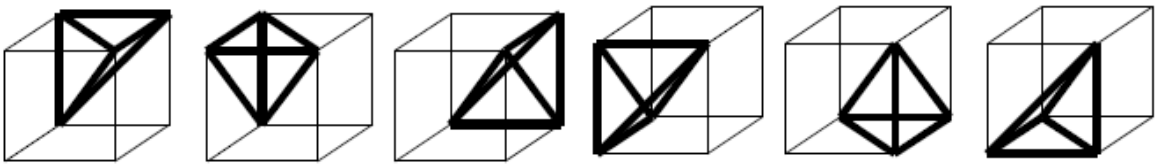


Fig. 3.3 – *Décomposition en 6 tétraèdres utilisée [43].*

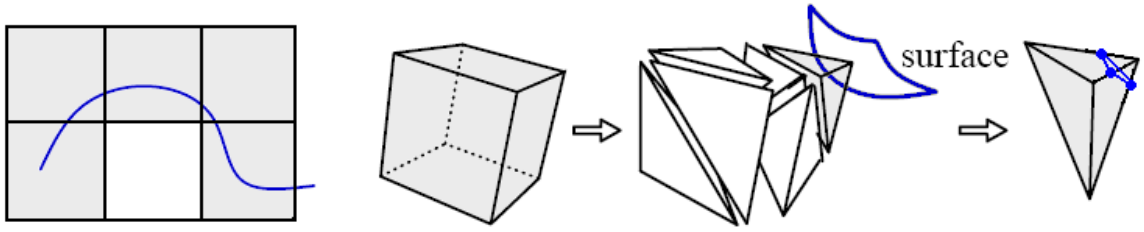


Fig. 3.4 – *Méthode du Marching Tetrahedra.*

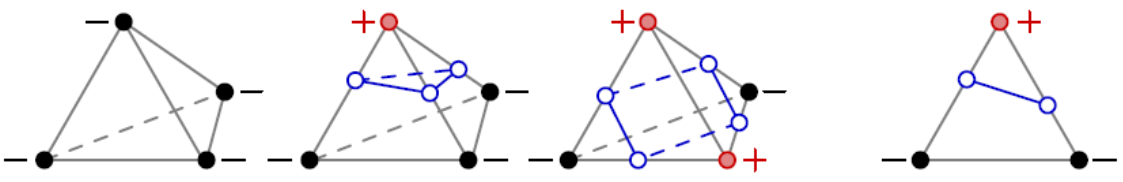


Fig. 3.5 – *Algorithme du Marching Tetrahedra : Polygonisation du tétraèdre [4].*

3.3 Implémentation

3.3.1 Algorithme

L'algorithme implémenté peut se résumer comme suit :

1. On définit une grille régulière, partionnant le domaine d'étude en cellules. Cette grille est définie par son étendue et son nombre de subdivisions. On en déduit les dimensions du cube élémentaire.
2. **Pour chaque cube :**
   (a) On regarde si le cube est intersecté par la surface implicite ; sachant qu'un cube est intersecté si on observe un changement de signe de la valeur de la fonction implicite en parcourant les 8 sommets du cube.
   (b) **Si le cube est intersecté :**
      i. On forme les 6 tétraèdres schématisés à la figure 3.3
      ii. **Pour chaque tétraèdre :**
         A. On parcourt les arrêtes du tétraèdre et on regarde si elles sont intersectées par la surface, c'est-à-dire, si les valeurs de la fonction implicite sur les deux sommets sont de signes opposés. Pour les arrêtes intersectées, on calcule, par interpolation linéaire, les coordonnées du point d'intersection avec la surface implicite.
         B. Si le tétraèdre est intersecté, on crée un index binaire variant de 0 à 2^4 en stockant l'état binaire de chaque sommet dans un bit (bit à 1 si la valeur de la fonction implicite sur le sommet est positive, 0 sinon). Ainsi, si seul le premier sommet est intersecté, l'index binaire vaudra 0001 (fig. 3.6).
         C. Une liste de correspondance fournit alors pour chaque index binaire, une liste ordonnée des arrêtes intersectées. Ceux-ci forment soit un, soit deux triangles (fig. 3.5). Ces triangles constituent l'approximation de la surface implicite dans le tétraèdre.
   (c) **Si le cube n'est pas intersecté :**
      i. On passe directement au cube suivant.

Ensuite, l'isosurface générée est visualisée à l'aide de VTK.

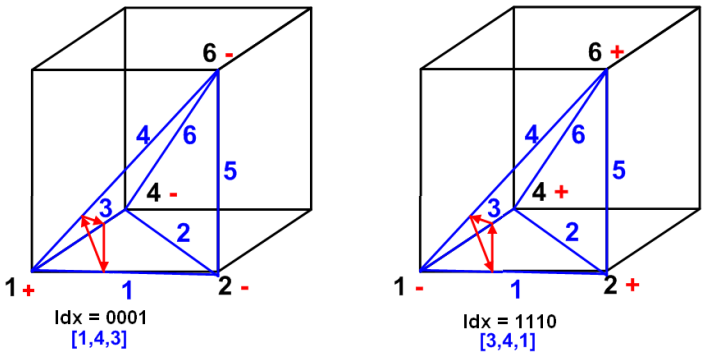


Fig. 3.6 – *Construction de l'isosurface sur un tétraèdre. Les numéros des sommets du cube élémentaire sont en noir, ceux des arrêtes du tétraèdre sont en bleu. L'index binaire (noté Idx) est créé en parcourant les sommets du tétraèdre dans l'ordre. Cet index permet de déterminer le vecteur (où les deux vecteurs dans le cas où deux sommets sont positifs, les deux autres étant négatifs) contenant les arrêtes intersectées par la surface (ce vecteur est indiqué en bleu). Ceci permet de donner une orientation au triangle créé.*

3.3.2 Structures

Pour réaliser plus facilement l'ensemble des fonctions du Marching Tetrahedra, ainsi que celles des algorithmes présentés dans les chapitres suivants, nous avons créé différentes classes :

Le noeud (Node) : caractérisé par son numéro, sa position, la valeur de la fonction implicite, ses arrêtes.

L'arrête (Edge) : un ensemble de deux *noeuds*.

La cellule (Cell) : un ensemble de *noeuds* et d'*arrêtes*, avec une intersection possible.

Le tétraèdre (Tetra) : une *cellule* à 4 *noeuds*.

Le triangle (Triangle) : une *cellule* à 3 *noeuds*.

La grille régulière (RegularGrid) : caractérisée par une fonction implicite, des *noeuds*, des *triangles*, des *arrêtes* et les dimensions de la grille (étendue, nombre de divisions et espacements).

Le maillage (Mesh) : un ensemble de *noeuds* et de *triangles*.

3.3.3 Difficultés et particularités de l'implémentation

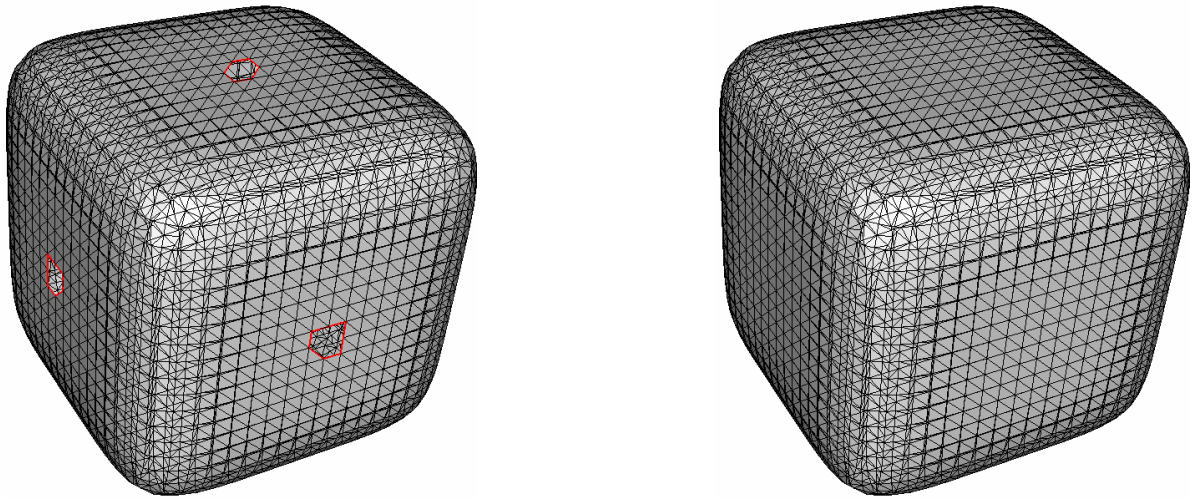
Pour accroître la vitesse de l'algorithme, l'implémentation a été réalisée de manière à ne pas garder en mémoire toute la grille. Ainsi, la création des *noeuds* de la grille ne se fait pas au début de l'algorithme mais lorsqu'ils sont rencontrés pour la première fois. De même, lorsqu'un *noeud* a été parcouru 8 fois, il est supprimé de la mémoire en même temps que ses *arrêtes*. De cette façon, la structure *grille régulière* ne contient pas l'ensemble des *noeuds* et des *arrêtes* de la grille mais seulement ceux qui sont utiles pour l'algorithme au moment considéré.

L'orientation des triangles du maillage surfacique a de l'importance : la normale aux triangles doit être dirigée vers l'extérieur de la surface. Dans le programme, *un triangle* est défini comme étant une suite ordonnée de trois noeuds. Les noeuds d'un *tétraèdre* sont également ordonnés. Pour le calcul de l'index binaire, ceux-ci sont parcourus dans l'ordre. A partir de l'index binaire, la liste de correspondance fournit, un ou deux vecteurs (orientés) contenant le numéro des arrêtes intersectées. Ces vecteurs définissent le ou les deux triangles, orientés, qui constituent l'approximation de l'isosurface dans le tétraèdre. La procédure est illustrée à la figure 3.6. Nous vérifions que dans les deux cas la normale aux triangles est dirigée vers le côté où la fonction est positive (dans la figure 3.6, la définition de la fonction implicite est telle qu'elle est négative à l'intérieur de la surface et positive à l'extérieur).

Une dernière difficulté a été la gestion du cas où la valeur de la fonction implicite sur un noeud de la grille est exactement nulle. Sans traitement spécifique, des trous peuvent apparaître dans le triangulation 3.7(a). En effet, si, lors de la construction de l'isosurface sur le tétraèdre, un ou deux sommets du tétraèdre sont nuls, l'isosurface générée dégénère en un point ou une droite et aucun triangle n'est créé (fig. 3.8). Le problème a été résolu en ajoutant une condition à la définition de la fonction implicite. Par exemple, pour la définition du cube de la figure 3.7 :

```
def func(x):
    x = x/10
    val = pow(x[0],6) + pow(x[1],6) + pow(x[2],6) - 1
    if val == 0.0:        # condition ajoutée pour éviter les trous
        val = 10e-5
    return val
```

Ainsi, le cas d'une fonction implicite nulle sur un sommet du tétraèdre n'apparaît plus et les triangulations générées ne comportent plus de trous. Notons qu'idéalement, cette correction devrait être faite à l'intérieur de l'algorithme.



(a) Cube avec trous (b) Cube sans trous

Fig. 3.7 – *Le problème des trous dans les maillages a été réglé en ajoutant une condition à la définition de la fonction implicite.*

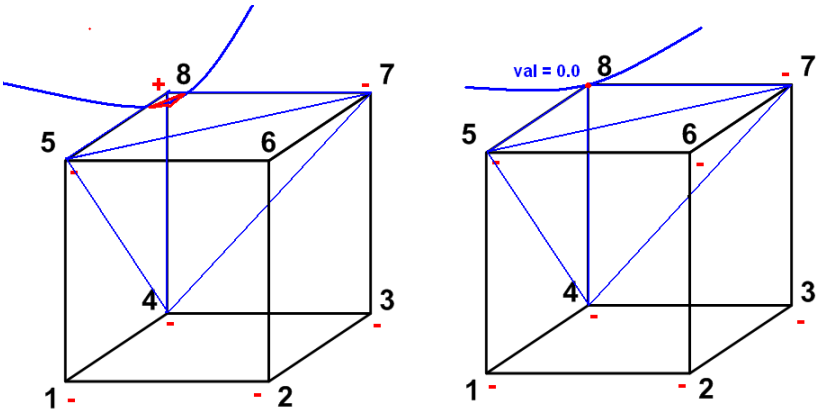


Fig. 3.8 – *Si la fonction implicite est quasi nulle sur un sommet du tétraèdre, le triangle créé est très petit (gauche). Si la fonction est exactement nulle sur le sommet du tétraèdre, aucun triangle n'est généré et un trou apparaît dans la triangulation.*

3.4 Présentation et analyse des résultats

3.4.1 Qualité des triangles générés

Mesure de la qualité d'un triangle

Dans le cas de la méthode des éléments finis, la qualité du maillage surfacique est un critère important, notamment en raison de son effet sur :

- la précision des solutions numériques et la convergence du calcul,
- la qualité du maillage tridimensionnel, liée inévitablement à la qualité du plus mauvais élément de la triangulation de la surface du domaine considéré.

La qualité Q_k d'un triangle k est une valeur numérique qui donne une mesure de sa forme géométrique. Par convention, celle-ci varie entre 0 (triangle plat) et 1 (triangle équilatéral). Des différentes mesures possibles de la qualité d'un triangle, nous en envisagerons deux.

Qualité d'un triangle selon Semenova [53] :

$$Q_K = \alpha_1 \frac{S(K)}{l_1^2 + l_2^2 + l_3^2} \tag{3.1}$$

avec

- $\alpha_1 = 4\sqrt{3}$, le coefficient de normalisation assurant une valeur de qualite Q_K unité dans le cas d'un triangle équilatéral ;
- $S(K)$,la surface du triangle K ;
- l_1, l_2, l_3, les longueurs des côtés des triangles.

Qualité d'un triangle selon Frey [23] :

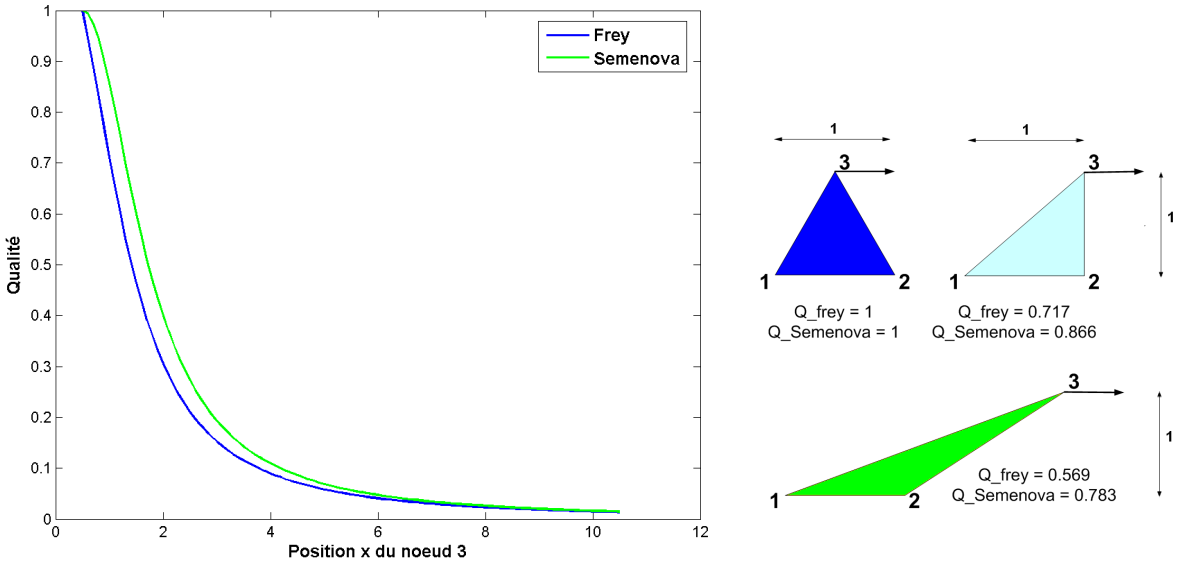
$$Q_K = \alpha_2 \frac{\rho_K}{h_{max}} = \alpha_2 \frac{S(K)}{p_K\, h_{max}} \tag{3.2}$$

avec

- $\alpha_2 = \frac{6}{\sqrt{3}}$, le coefficient de normalisation assurant une valeur de qualite Q_K unité dans le cas d'un triangle équilatéral ;
- h_{max}, le diamètre de K, c.à.d., la longueur de son plus grand côté ;
- ρ_K, le rayon du cercle inscrit à K, calculé comme le rapport de la surface $S(K)$ par le demi-périmètre p_K.

Bien que les deux mesures de qualité ci-dessus donnent une valeur de 1 pour un triangle équilatéral et de 0 pour un triangle plat, leur variation entre ces deux extrémités est différente. Ceci est illustré à la figure 3.9. Partant d'un triangle équilatéral, nous avons déplacé horizontalement un des noeuds afin d'obtenir un triangle de plus en plus allongé. Le graphe donne les valeurs de qualité obtenues (selon Semenova et Frey) en fonction du déplacement du noeud. Nous remarquons que la mesure de qualité utilisée par Semenova est plus optimiste que celle conseillée par Frey.

Dans ce qui suit, la mesure de qualité des triangles utilisée est celle conseillée par Frey. En effet, celle-ci étant plus pessimiste que la mesure de qualité de Semenova, elle nous permettra de mieux détecter les triangles de mauvaise qualité.



(a) (b)

Fig. 3.9 – *Mesures de qualité d'un triangle : Comparaison entre la mesure de Frey [23] et celle de Semenova [53]. Pour comparer ces deux mesures, nous déplaçons le noeud 3 du triangle représenté en (b) et calculons les qualités des triangles ainsi formés.*

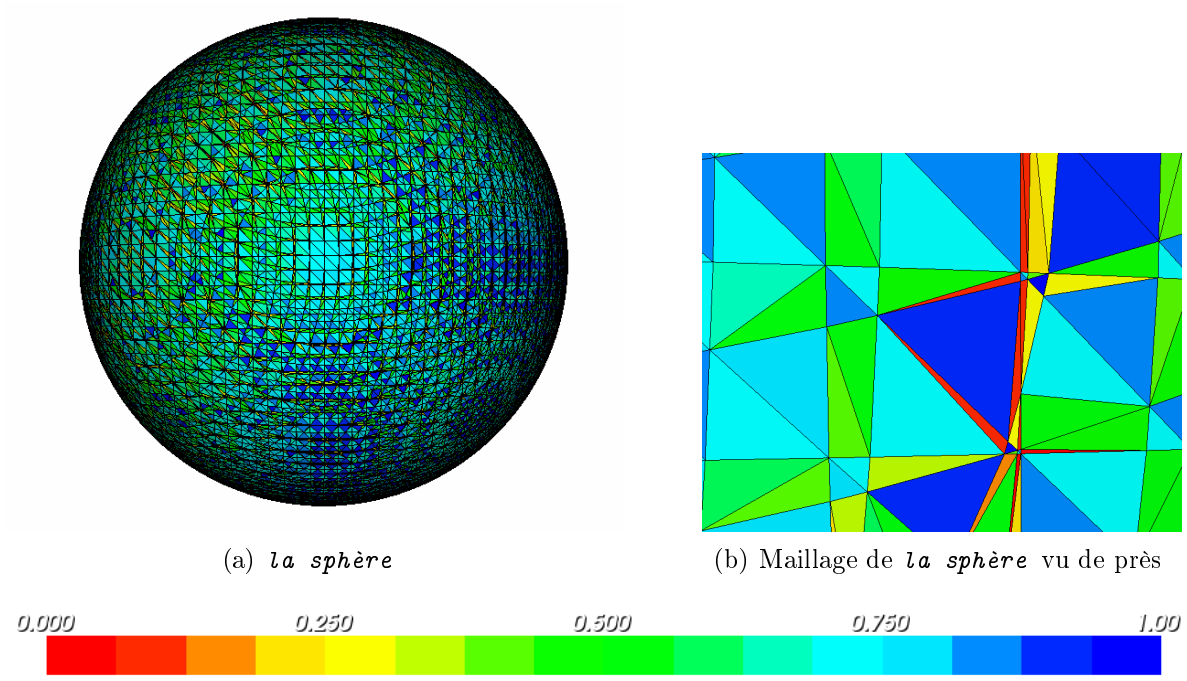
Marching Tetrahedra									**moyenne**
nombre de noeuds	22478	19678	40878	27486	46620	10788	18302	21954	26023
nombre de triangles	44952	39352	81752	54968	93236	21576	36596	43944	52047
Longueurs :									
Minimum	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Maximum	1,66	1,68	1,67	2,09	1,73	1,71	1,71	1,64	1,74
Moyenne	0,57	0,55	0,64	0,84	0,61	0,58	0,56	0,58	0,61
Aires :									
Minimum	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Maximum	0,59	0,57	0,66	1,03	0,71	0,63	0,63	0,65	0,68
Moyenne	0,09	0,09	0,09	0,02	0,05	0,08	0,09	0,09	0,08
Qualité selon Semenova									
Pire triangle	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Meilleur triangle	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
Moyenne	0,74	0,74	0,69	0,69	0,73	0,74	0,74	0,74	0,73
Qualité selon Frey									
Pire triangle	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Meilleur triangle	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
Moyenne	0,62	0,61	0,57	0,57	0,60	0,61	0,61	0,61	0,60
Temps de calcul [s]	22	27	47	27	52	23	26	15	30
Vitesse [triangles / s]	2032	1465	1755	2044	1789	951	1400	2881	1745

Tab. 3.1 – *Algorithme du Marching Tetrahedra : Résultats numériques obtenus pour les 8 géométries tests, dont les maillages sont présentés aux figures 3.18 à 3.25.*

Qualité des maillages des 8 géométries tests

Nous avons mesuré et visualisé la qualité des maillages obtenus pour les 8 géométries *tests*. Les résultats numériques sont repris dans le tableau 3.1. Dans ce tableau, nous lisons que pour chaque maillage, la valeur minimale de la qualité vaut 0, ce qui signifie que l'algorithme génère des triangles plats. Notons aussi qu'en moyenne et selon la mesure de qualité conseillée par Frey, la qualité d'un triangle est de 0.6. Les triangles du maillage de la figure 3.10 ont été coloriés en fonction de leur qualité. Nous remarquons que des triangles adjacents peuvent être de taille et de qualité très différentes.

En conclusion, la qualité des maillages obtenus est très mauvaise. Ceci n'est pas étonnant puisque aucune mesure n'est prise dans l'algorithme du Marching Tetrahedra pour obtenir des triangles de qualité. C'est pourquoi nous avons implémenté deux méthodes permettant d'améliorer la qualité du maillage en sortie de l'algorithme du Marching Tetrahedra. La première modifie la topologie du maillage (chap. 5), la seconde joue sur la position des noeuds (chap. 6).



(a) *la sphère*

(b) Maillage de *la sphère* vu de près

(c) Échelle utilisée pour la coloration des triangles en fonction de leur qualité

Fig. 3.10 – *Qualité du maillage surfacique généré par l'algorithme du Marching Tetrahedra. Exemple de la sphère. La mesure de qualité utilisée est celle conseillée par Frey.*

3.4.2 Quantité de triangles générés

Les triangulations obtenues conduisent à un nombre important de triangles (tab. 3.1), généralement trop important pour permettre un calcul éléments finis. Ceci est inévitable : le nombre de triangles générés résulte directement de la finesse du partitionnement du domaine spatial, finesse qui est indispensable pour préserver une certaine qualité de l'approximation géométrique de la surface (fig. 3.11). Ce choix de la taille des cellules ne se fait pas automatiquement, l'utilisateur doit ajuster ce paramètre avec soin, en fonction de la géométrie et en

tenant compte du compromis approximation géométrique - quantité de triangles (temps de calcul, espace mémoire). En sortie de l'algorithme du Marching Tetrahedra, il faudra trouver une méthode pour réduire le nombre de triangles tout en préservant la qualité de l'approximation géométrique (chap. 4).

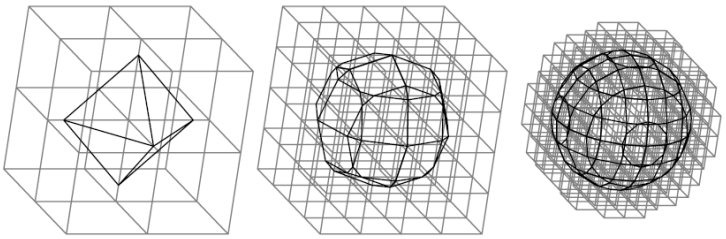


Fig. 3.11 – *Influence du partitionnement de l'espace en cellules sur l'approximation géométrique de la sphère [4].*

3.4.3 Temps de calcul en fonction du nombre de triangles générés

Les temps de calculs nécessaires à l'extraction des 8 surfaces implicites étudiées sont repris dans le tableau 3.1. Le graphe de la figure 3.12 indique que le temps de calcul varie un peu plus que linéairement avec le nombre de facettes générées. Ce graphe a été établi pour `la sphère`, en faisant varier la taille du cube élémentaire entre 1 et 12.

Les temps nécessaires à l'extraction des surfaces *tests* sont indiqués dans le tableau 3.1. En moyenne, la vitesse est de 1745 triangles/secondes[1].

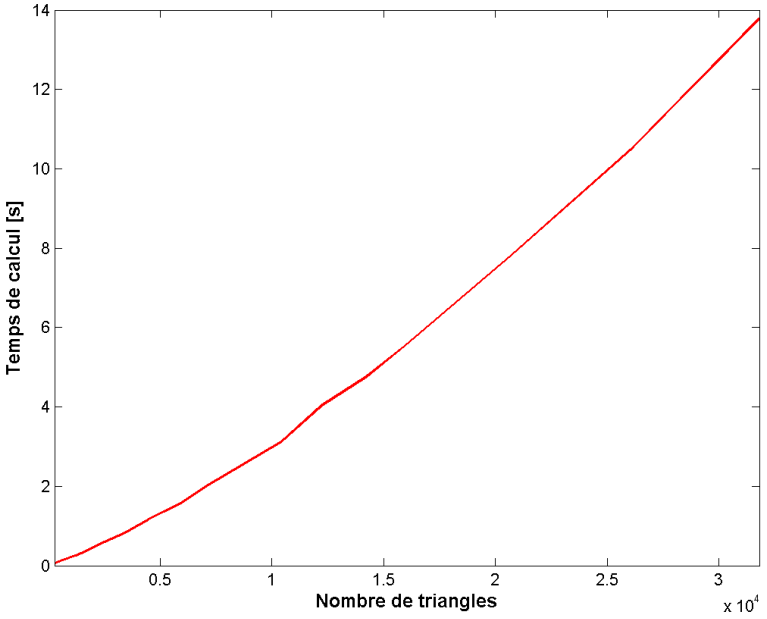


Fig. 3.12 – *Algorithme du Marching Tetrahedra : Temps de calcul en fonction du nombre de triangles générés. Résultats obtenus dans le cas de* `la sphère`*, en faisant varier la taille du cube élémentaire.*

[1] sur PC : Intel(R) Celeron(R), processeur 1.4 GHZ, 448 Mb RAM

3.4.4 Comparaison avec l'algorithme du Marching Cubes

La classe VTK `vtkMarchingContourFilter` [52] utilise l'algorithme du Marching Cubes de B. Lorensen et H. Cline pour générer des isosurfaces à partir de la définition d'une fonction implicite. L'utilisation de cette fonction va nous permettre de comparer les deux algorithmes Marching Cubes et Marching Tetrahedra.

Les résultats obtenus dans le cas de *la sphère* sont montrés à la figure 3.14. Nous vérifions que l'algorithme du Marching Cubes génère moins de triangles que l'algorithme implémenté : nous pouvons donc supposer que l'algorithme est moins gourmand en ressources (temps de calcul et mémoire). Cependant, parce qu'il génère plus de facettes, l'algorithme du Marching Tetrahedra approche mieux la surface implicite réelle.

La figure 3.13 permet de voir, pour les deux algorithmes, l'évolution du nombre de triangles générés en fonction de la taille du cube élémentaire dans le cas de *la sphère*. L'algorithme du Marching Tetrahedra génère environ 3.3 fois plus de triangles que l'algorithme du Marching Cubes.

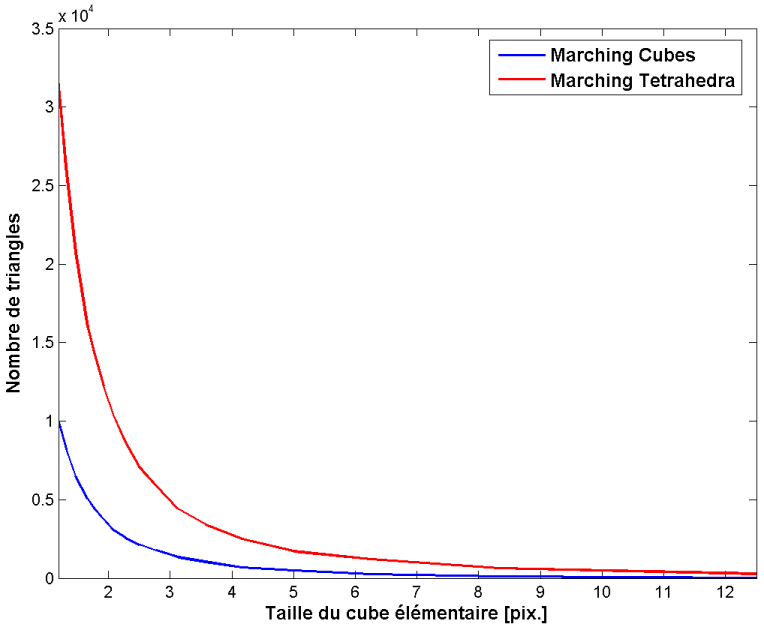
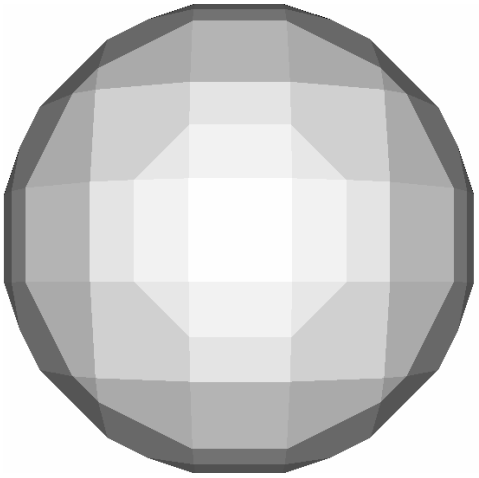
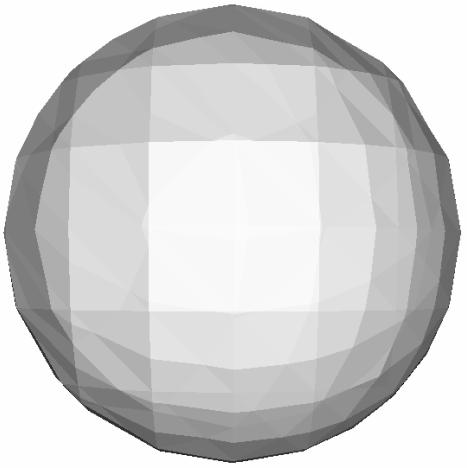


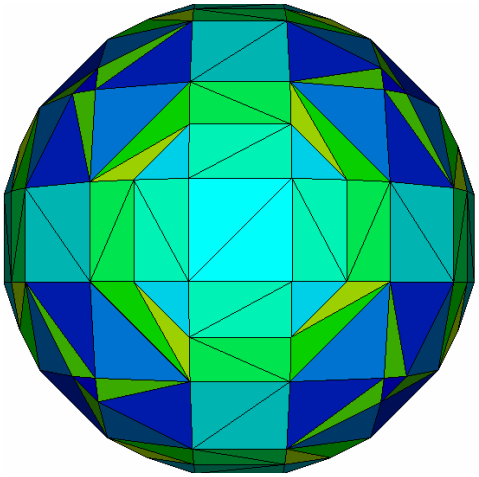
Fig. 3.13 – *Graphe d'évolution du nombre de triangles de surface en fonction de la taille du cube élémentaire. Ce graphe a été établi pour* *la sphère*, *en faisant varier la taille du cube élémentaire entre 1 et 12.*



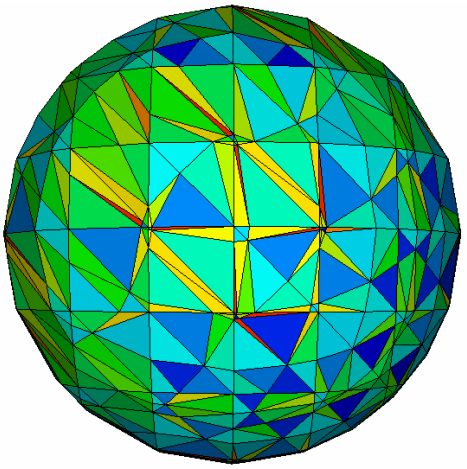
(a) Marching Cubes : visualisation



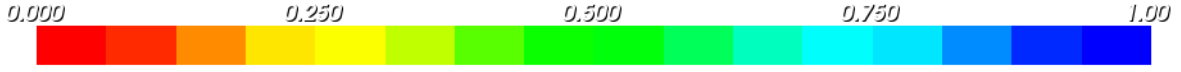
(b) Marching Tetrahedra : visualisation



(c) Marching Cubes : maillage



(d) Marching Tetrahedra, maillage



(e) Échelle de qualité (selon Frey)

Fig. 3.14 – *Comparaison entre l'algorithme du Marching Cubes et l'algorithme du Marching Tetrahedra implémenté.*

3.5 Visualisation des maillages obtenus pour les 8 géométries *tests*

L'algorithme du Marching Tetrahedra a été appliqué avec succès aux 8 géométries *tests* (fig. 3.18 à 3.25). La dimension du domaine d'étude est propre à chaque géométrie et a été donnée lors de la présentation des surfaces implicites (chap. 2). La grille utilisée par l'algorithme du Marching Tetrahedra est une grille régulière partitionnant le domaine d'étude en cubes de longueur unité. Rappelons que le nombre de triangles générés dépend directement de la finesse de la grille.

Les géométries définies en prenant *le maximum* et le *le minimum entre un cube et une sphère* montrent que l'algorithme du Marching Tetrahedra traite relativement bien les angles vifs, l'intersection étant décrite par une série continue d'arrêtes (fig. 3.15). A priori, rien n'implique, dans l'algorithme du Marching Tetrahedra, que ce soit toujours le cas, mais nous n'avons pu trouver un contre-exemple.

La géométrie du nombre d'or indique qu'il peut y avoir plusieurs objets dans le domaine étudié et que ces objets peuvent être troués.

Enfin, la figure 3.16 montre que, pour un espacement de la grille de 1, la surface intérieure de *la multiplication entre deux sphères* est détachée de la surface extérieure. Il n'en est pas de même si nous prenons une grille plus grossière (fig. 3.17). En effet, pour que les surfaces intérieures et extérieures soient détachées en sortie il faut que les surfaces en vis-à-vis soient situées dans des cellules différentes de la grille utilisée par l'algorithme du Marching Tetrahedra.

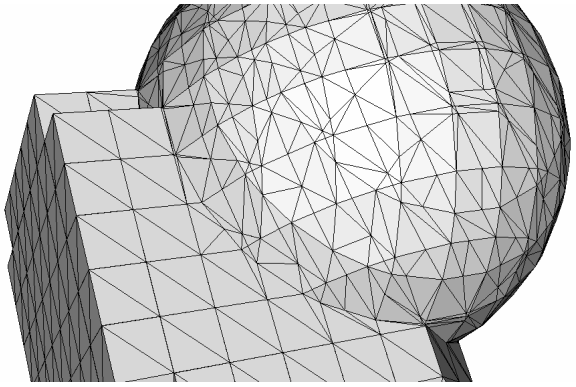


Fig. 3.15 – *La frontière entre le cube et la sphère est décrite par une série continue d'arrêtes (l'espacement de la grille utilisée par l'algorithme du Marching Tetrahedra a été imposée à 5).*

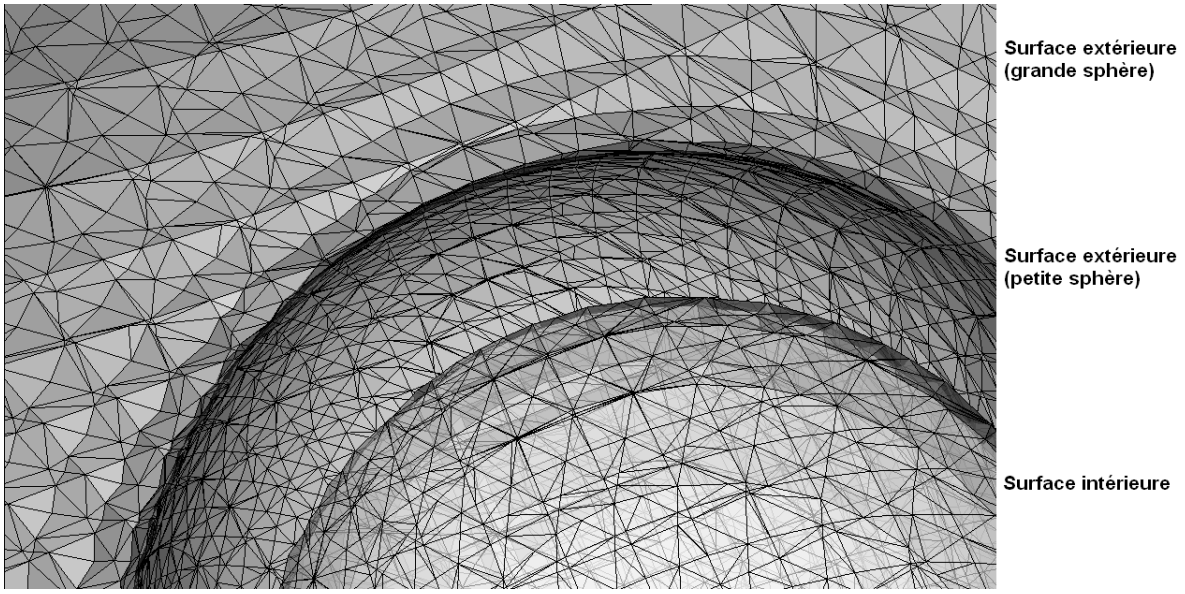


Fig. 3.16 – *La surface intérieure de* `la multiplication de deux sphères` *est détachée de la surface extérieure pour une grille régulière d'espacement 1 (qui est l'espacement par défaut utilisée par l'algorithme du Marching Tetrahedra implémenté).*

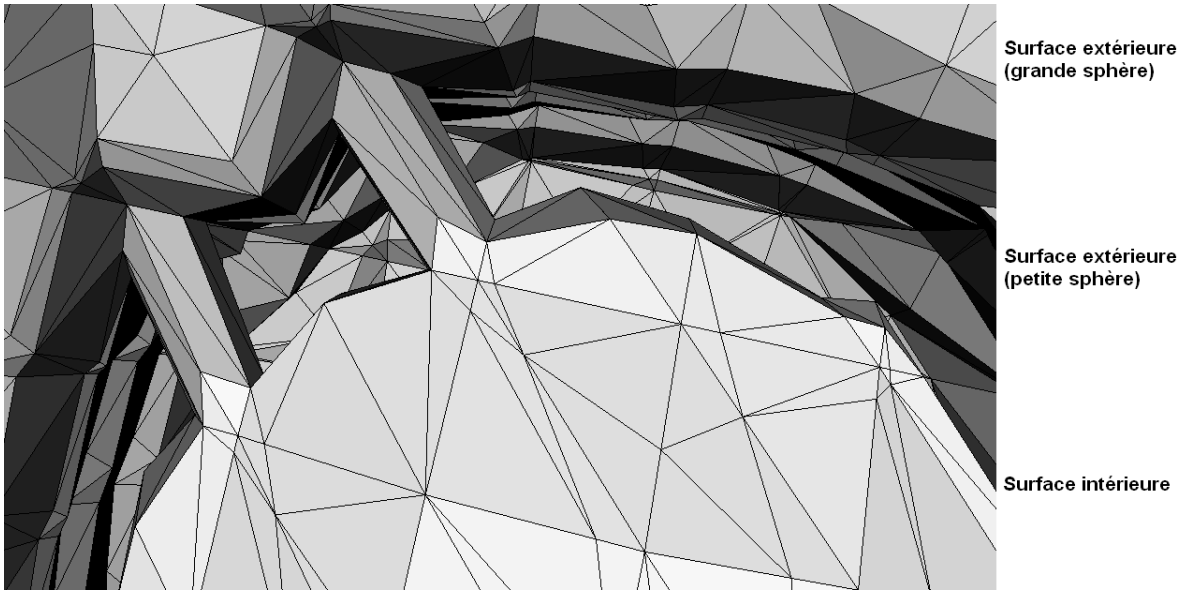


Fig. 3.17 – *La surface intérieure de* `la multiplication de deux sphères` *n'est pas détachée de la surface extérieure lorsque les cellules de la grille utilisée par l'algorithme sont de dimensions* $3 \times 3 \times 3$.

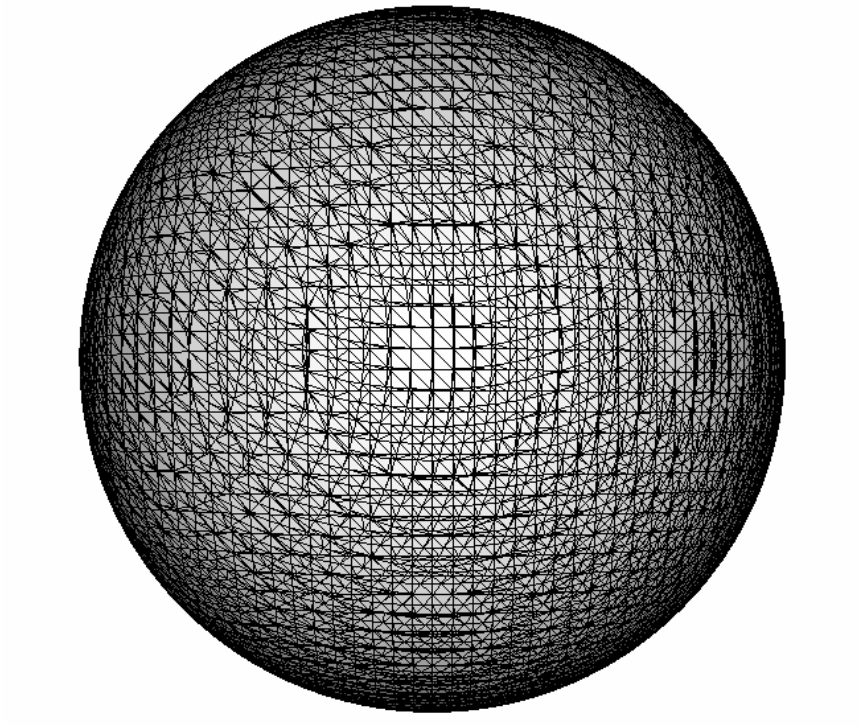


Fig. 3.18 – *Algorithme du Marching Tetrahedra appliqué sur* `la sphère` *: 44952 triangles générés.*

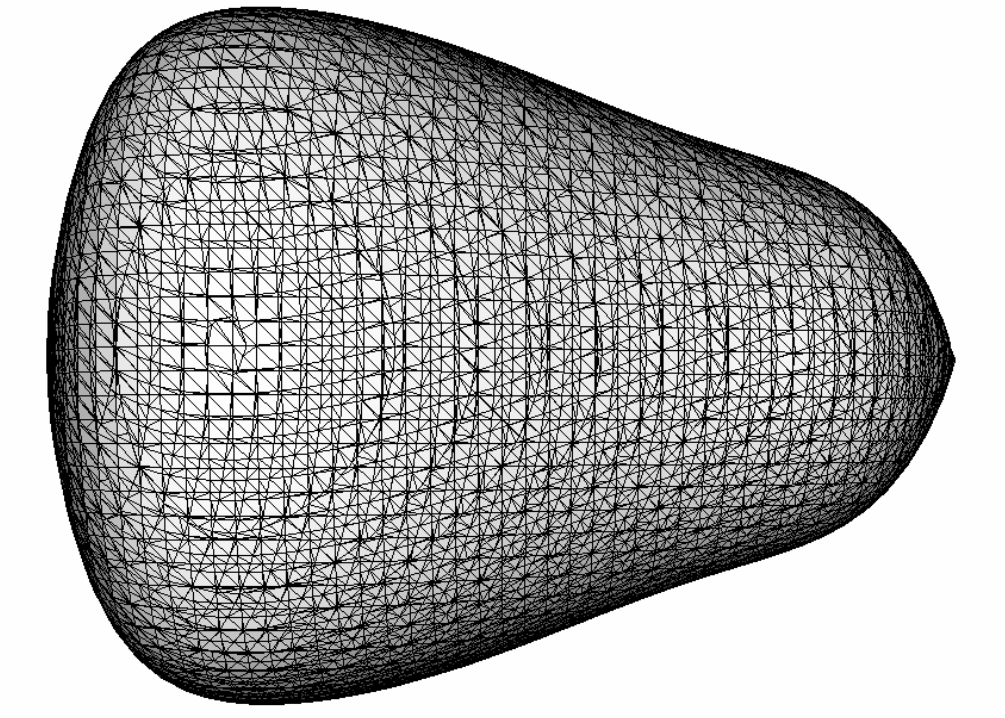


Fig. 3.19 – *Algorithme du Marching Tetrahedra appliqué sur* `le cône arrondi` *: 39352 triangles générés.*

Fig. 3.20 – *Algorithme du Marching Tetrahedra appliqué sur* `le parallélépipède` *: 81752 triangles générés.*

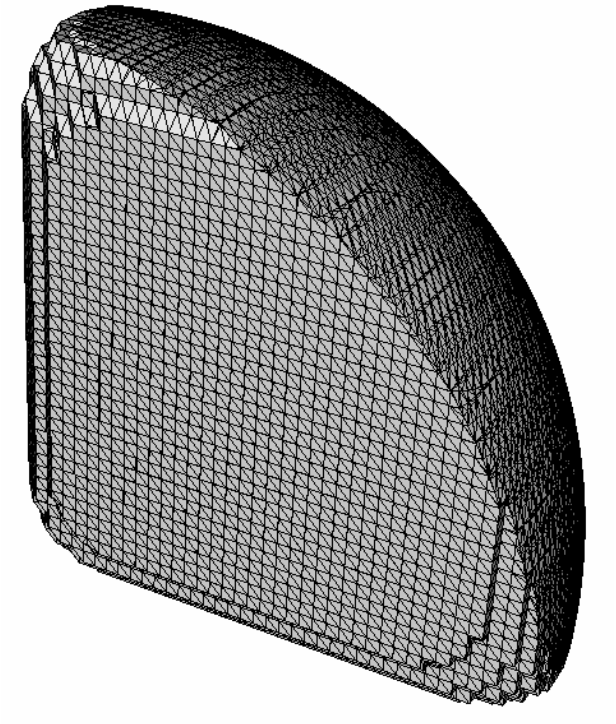


Fig. 3.21 – *Algorithme du Marching Tetrahedra appliqué sur* `le maximum entre un cube et une sphère` *: 54968 triangles générés.*

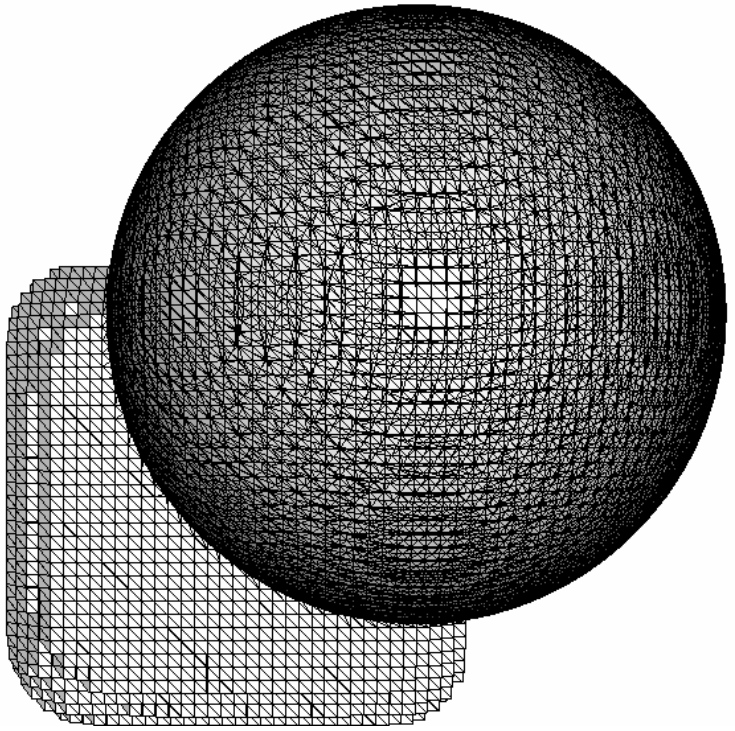


Fig. 3.22 – *Algorithme du Marching Tetrahedra appliqué sur* `le minimum entre un cube et une sphère` *: 93236 triangles générés.*

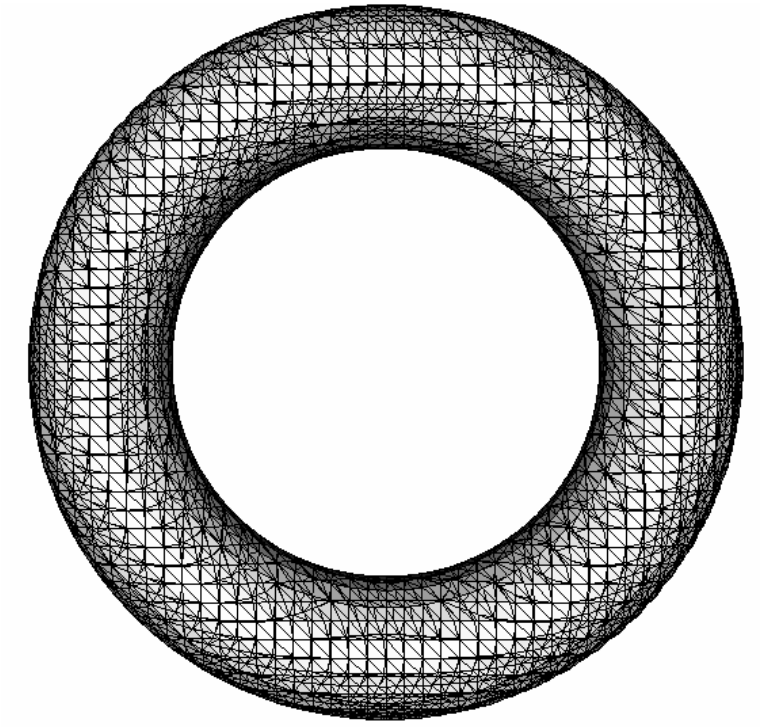


Fig. 3.23 – *Algorithme du Marching Tetrahedra appliqué sur* `le tore` *: 21576 triangles générés.*

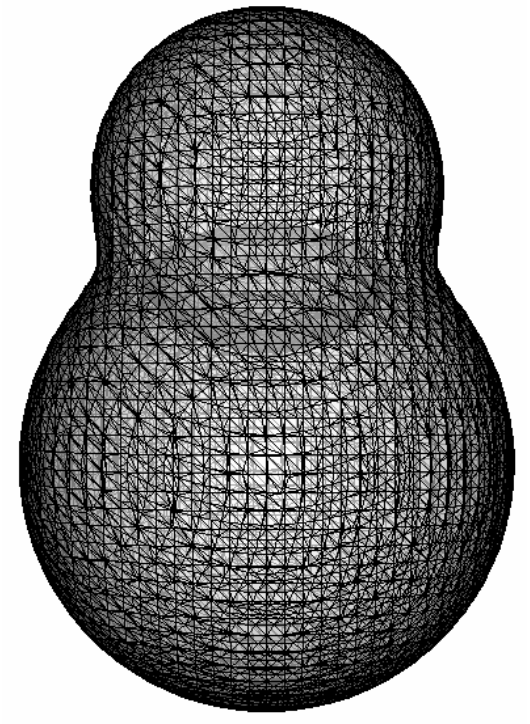


Fig. 3.24 – *Algorithme du Marching Tetrahedra appliqué sur* `la multiplication de deux sphères` *: 36596 triangles générés (la surface interne est visible par transparence).*

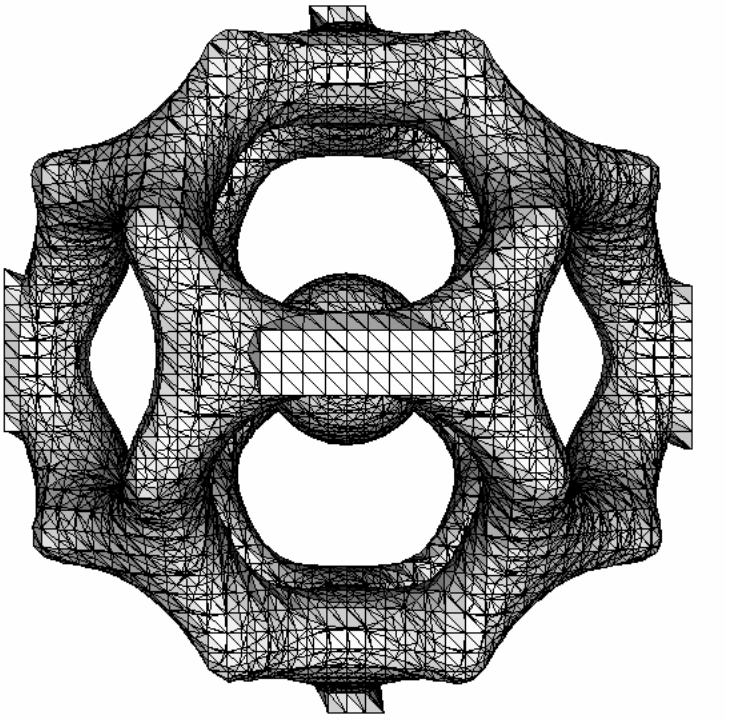


Fig. 3.25 – *Algorithme du Marching Tetrahedra appliqué sur* `la géométrie du nombre d'or` *: 43944 triangles générés.*

CHAPITRE 4

Simplification du maillage

Le maillage surfacique que l'on voudrait obtenir est un maillage où la taille des mailles est constante et peut être fiablement définie par l'utilisateur. De ce point de vue, le maillage fourni par l'algorithme du Marching Tetrahedra contient un nombre trop important de triangles, de qualités inégales. L'objectif de ce chapitre est de réduire le nombre de triangles du maillage tout en préservant le degré de précision géométrique de celui-ci.

4.1 État de l'art

Les différents algorithmes de simplification de maillages surfaciques présentés dans la littérature utilisent un ou plusieurs des mécanismes suivants :

Echantillonnage (sampling) : on commence par ré-échantillonner le modèle soit en choisissant des points sur sa surface, soit en superimposant une grille 3D de voxels. La qualité de sortie de l'algorithme est contrôlée par la densité d'échantillonnage [13, 30].

Subdivision adaptative : on établit d'abord un modèle fortement simplifié, puis on divise itérativement les polygones jusqu'à ce que l'on se rapproche du modèle original selon un critère choisi [13, 30].

Décimation : chaque étape de l'algorithme retire un sommet ou un polygone du modèle. Les trous obtenus sont retriangulés à chaque itération [13, 30, 52].

Fusion de sommets (vertex merging) : cette méthode fusionne deux sommets voisins en un seul, et ce récursivement. On utilise classiquement la méthode dite du *edge collapse* qui ne fusionne que deux sommets possédant une arête en commun [2, 29, 47].

Fusion de faces (coplanar surface merging) : des surfaces adjacentes coplanaires (satisfaisant un test de coplanarité) sont rassemblées pour n'en former plus qu'une. On retriangule ensuite le polygone formé. Une généralisation de la méthode est la méthode des supersurfaces [34].

Bien entendu, chaque méthode a ses avantages et inconvénients. Ceux-ci étant très bien résumés dans [30] et [13] nous ne rentrerons pas plus dans les détails dans le cadre de ce travail.

4.2 Méthode implémentée, le Vertex Clustering

4.2.1 Algorithme original

La méthode de simplification de maillages implémentée dans ce travail est basée sur l'article de Rossignac et Borrel [50]. Les principaux critères de choix de cette méthode ont été sa robustesse, sa simplicité d'implémentation, sa rapidité et le fait qu'il s'agit d'une des seules méthodes qui n'impose pas de restrictions sur la géométrie de la surface à mailler.

Par contre la méthode du Vertex Clustering comporte les inconvénients suivants :
- L'approximation de la surface est correcte, mais les détails ne sont pas bien conservés.
- La topologie du maillage initial n'est pas conservée, ce qui signifie que les relations entre les noeuds du maillage (la connexité du maillage) peuvent changer.
- Le résultat dépend de l'orientation du modèle dans l'espace.
- Il est impossible de prévoir à priori le nombre de triangles en sortie.
- Il est impossible d'imposer une borne sur l'approximation géométrique.
- Le domaine d'étude peut contenir plusieurs objets, mais si ceux-ci sont trop proches, il seront rassemblés.

Schématiquement, l'algorithme de Rossignac et Borrel se déroule comme suit (fig. 4.1) :

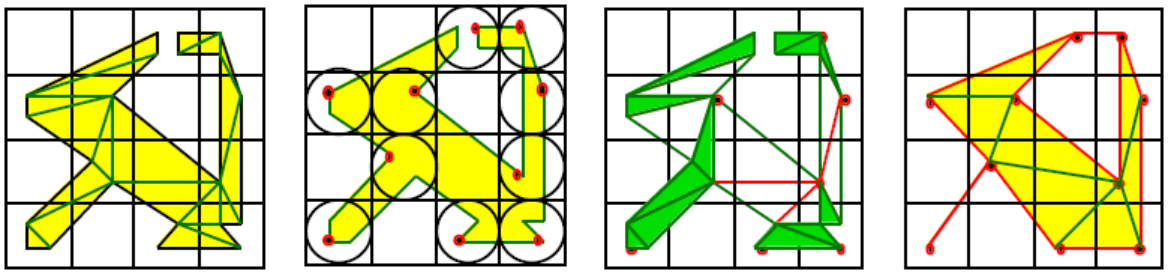


Fig. 4.1 – *Algorithme du Vertex Clustering : Résultats numériques [49].*

1. On assigne un *poids* à chaque noeud, en se basant par exemple sur la courbure locale et la surface des triangles avoisinants.
2. On superpose une grille tridimensionnelle. Les noeuds appartenant à un même cube unité forment un *cluster*.
3. On élit un noeud par cube unité (appelé *noeud représentatif* ci-dessous), en comparant les poids des concurrents.
4. Les autres noeuds sont placés sur le *noeud représentatif*. De cette manière, les triangles situés entièrement dans un *cluster* dégènerent en un point et les triangles ayant deux noeuds dans un même *cluster* dégénèrent en une arrête. Seuls les triangles ayant trois noeuds dans des *clusters* différents subsistent.
5. A la figure 4.1, nous voyons que deux solides (ou parties de solides) proches peuvent se souder et que des triangles peuvent dégénérer en une arrête dans le maillage final.

4.2.2 Modifications apportées à l'algorithme original

L'algorithme implémenté dans ce travail est fort semblable à celui de Rossignac et Borrel. La différence majeure entre les deux algorithmes est le choix du *noeud représentatif*. Au lieu

d'assigner un *poids* à chaque noeud du maillage, nous prenons comme *noeud représentatif* un nouveau noeud dont la position vaut la position *moyenne* des noeuds contenus dans le cube unité. Bien-entendu, rien n'assure que le noeud ainsi créé appartienne à la surface de l'objet. Il est donc indispensable, afin de limiter l'erreur d'approximation géométrique, de le projeter sur la surface réelle. Rappelons que la surface de l'objet à mailler correspond au zéro de la fonction implicite. Projeter le noeud sur la surface équivaut donc à une recherche de zéro. Celle-ci est réalisée suivant la direction du gradient, par une méthode de dichotomie.

Un critère a également été ajouté de manière à laisser intacts les noeuds situés sur la frontière du domaine d'étude.

4.3 Présentation et analyse des résultats

Les différents maillages obtenus par l'algorithme du Marching Tetrahedra ont été simplifiés grâce à l'algorithme du Vertex Clustering. Les résultats sont présentés en fin de chapitre, aux figures 4.9 à 4.16.

Le tableau 4.1 donne les résultats numériques obtenus, ceux-ci sont analysés et discutés dans ce qui suit.

Vertex Clustering									**moyenne**
nombre de noeuds	5942	5177	10494	6913	12293	2936	4824	6018	6825
nombre de triangles	11880	10350	20984	13822	24582	5872	9640	12072	13650
Longueurs									
Minimum	0,61	0,50	0,51	0,05	0,04	0,46	0,42	0,10	0,34
Maximum	1,97	2,05	2,01	4,58	3,70	2,05	2,09	2,30	2,59
Moyenne	0,99	0,98	1,02	1,28	0,99	0,99	0,99	0,98	1,03
Aires									
Minimum	0,22	0,17	0,20	0,01	0,01	0,16	0,16	0,02	0,12
Maximum	0,79	0,85	0,86	2,77	2,28	0,86	0,92	0,90	1,28
Moyenne	0,40	0,41	0,50	0,78	0,44	0,42	0,39	0,40	0,47
Qualité selon Semenova									
Pire triangle	0,53	0,49	0,51	0,00	0,01	0,47	0,46	0,16	0,33
Meilleur triangle	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
Moyenne	0,88	0,88	0,85	0,83	0,86	0,88	0,88	0,87	0,87
Qualité selon Frey									
Pire triangle	0,43	0,38	0,40	0,00	0,01	0,37	0,37	0,15	0,26
Meilleur triangle	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
Moyenne	0,76	0,76	0,71	0,68	0,71	0,75	0,76	0,74	0,73
Temps de calcul [s]	5	4	10	25	18	2	4	7	9
Vitesse [triangles/s]	2509	2760	2191	548	1359	2357	2429	1834	1468

Tab. 4.1 – *Algorithme du Vertex Clustering : Résultats numériques obtenus pour les 8 géométries tests, dont les maillages sont présentés aux figures 4.9 à 4.16.*

4.3.1 Réduction du nombre de triangles

Un inconvénient de la méthode du Vertex Clustering est que le nombre de triangles composant le maillage final ne peut pas être imposé. Cependant, ceci ne signifie pas qu'il est impossible de moduler le nombre de noeuds du maillage issu de notre algorithme. En effet, la dimension de la grille tridimensionnelle du Vertex Clustering peut être ajustée ; or celle-ci a un impact direct sur la taille des *clusters* et, à posteriori, sur le nombre de noeuds du maillage final. Nous avons illustré ceci en simplifiant plus ou moins le maillage de la sphère obtenu en sortie du Marching Tetrahedra. Le résultat est présenté à la figure 4.2 où λ désigne le rapport entre le taille de la grille du Vertex Clustering et celle du Marching Tetrahedra.

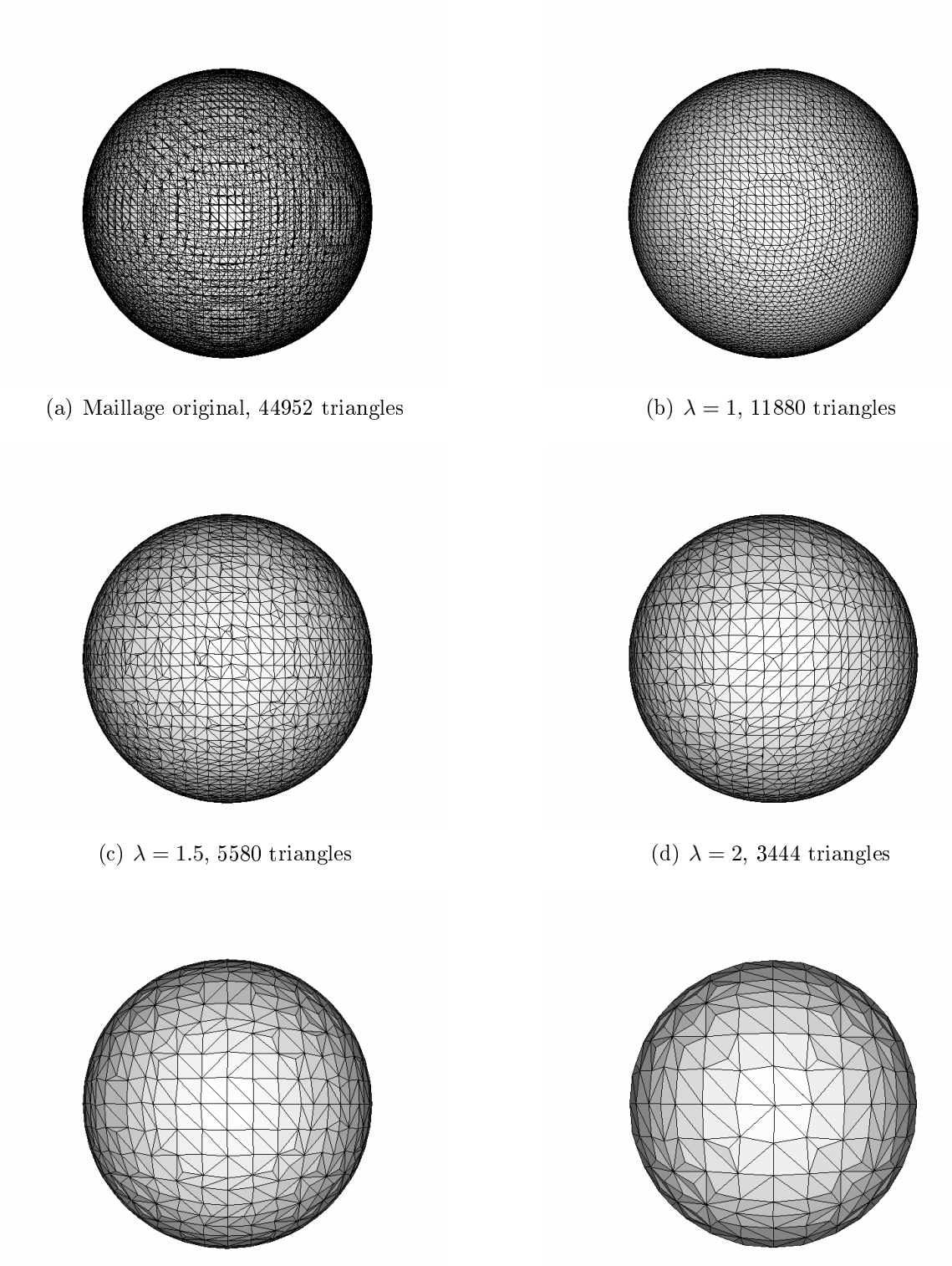
Dans les résultats présentés aux figures 4.9 à 4.16, la taille de la grille du Vertex Clustering a été choisie identique à celle du Marching Tetrahedra ($\lambda = 1$). C'est ce qui est réalisé par défaut par notre programme. Cependant, nous avons décalé les deux grilles. En effet, les noeuds du maillage issu de l'algorithme du Marching Tetrahedra se situent majoritairement sur les arrêtes des cubes formant la grille tridimensionnelle. Dans l'algorithme du Vertex Clustering, si la même grille est utilisée, ces noeuds pourraient se situent une fois dans l'un, une fois dans l'autre *cluster* (cube). A priori, ceci ne pose aucun problème puisque l'algorithme est tel qu'un noeud appartient toujours à un et un seul *cluster*. Cependant, les résultats obtenus sur deux ordinateurs séparés pourraient légèrement différer à cause des erreurs d'arrondis et c'est la raison pour laquelle nous avons décalé les deux grilles.

Le tableau 4.2 reprend, pour chacune des géométries, la réduction du nombre de triangles effectuée par l'algorithme du Vertex Clustering. Nous remarquons que, en moyenne et dans le cas où les tailles des grilles sont égales ($\lambda = 1$), l'algorithme de Vertex Clustering divise le nombre de triangles initial par 3.81. On retombe donc à un nombre de mailles proche de celui généré par l'algorithme du Marching Cubes (nous avions obtenu un rapport de 3.3 (sec. 3.4)).

Nombre de triangles									**moyenne**
Marching Tetrahedra	44952	39352	81752	54968	93236	21576	36596	43944	52047
Vertex Clustering	11880	10350	20984	13822	24582	5872	9640	12072	13650
facteur de réduction	3,78	3,80	3,90	3,98	3,79	3,67	3,80	3,64	3,81

Tab. 4.2 – *Réduction du nombre de triangles réalisée par l'algorithme du Vertex Clustering, pour les différentes géométries tests.*

Cette diminution du nombre du triangles se traduit par une augmentation de la longueur des arrêtes des triangles composant le maillage issu du Vertex Clustering. Alors que l'algorithme du Marching Tetrahedra génère des arrêtes de longueur quasi nulle, la longueur minimale observée en sortie du Vertex Clustering vaut 0.10 (observée pour *la géométrie du nombre d'or*). La figure 4.3 donne une idée de la variation des longueurs des arrêtes sur le maillage de *la sphère* ainsi que de l'augmentation de ces longueurs engendrée par l'algorithme du Vertex Clustering.



(a) Maillage original, 44952 triangles

(b) $\lambda = 1$, 11880 triangles

(c) $\lambda = 1.5$, 5580 triangles

(d) $\lambda = 2$, 3444 triangles

(e) $\lambda = 3$, 1428 triangles

(f) $\lambda = 4$, 768 triangles

Fig. 4.2 – *Simplification du maillage de* `la sphère` *issu du Marching Tetrahedra au moyen de l'algorithme du Vertex Clustering.*

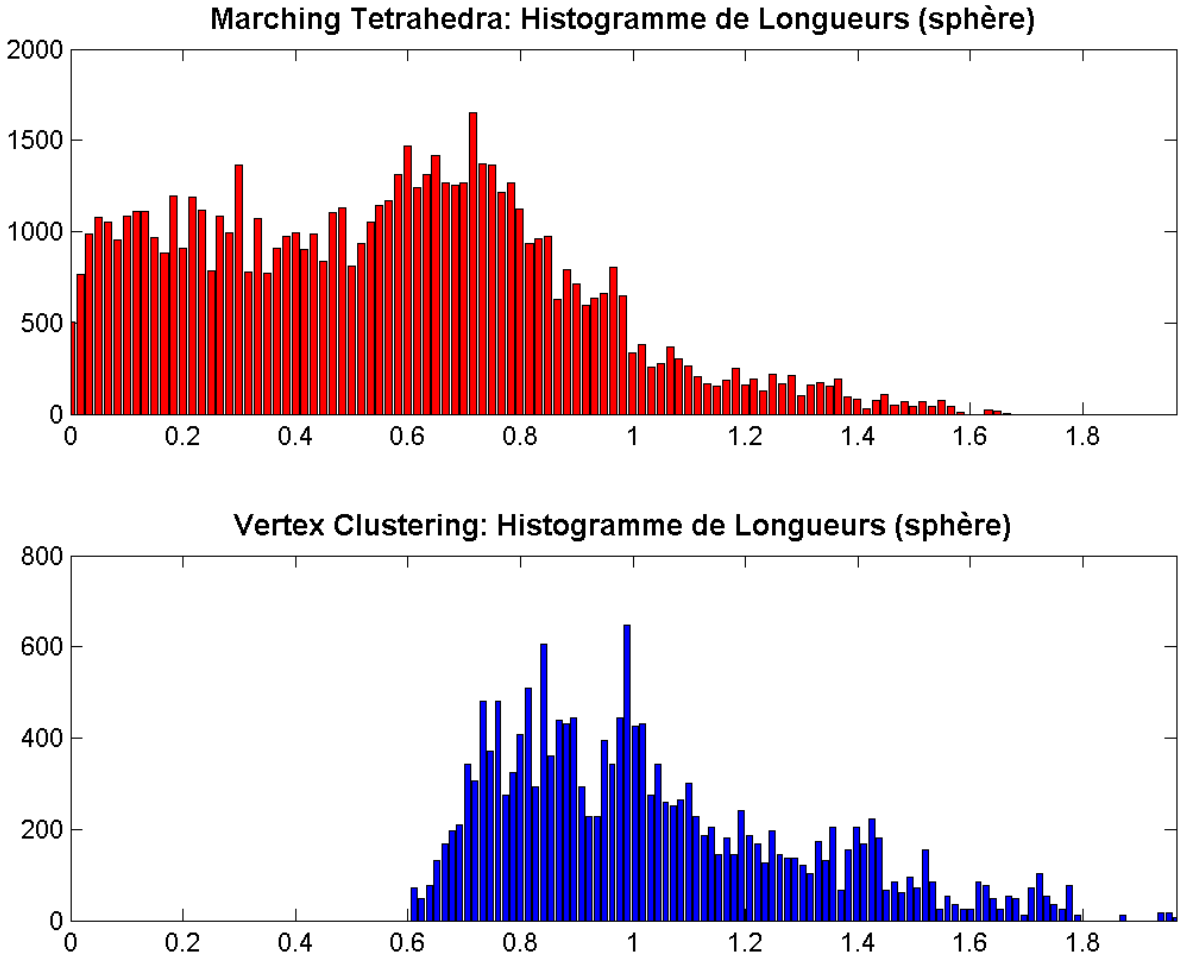


Fig. 4.3 – *Histogramme de la longueur des arrêtes du maillage de* `la sphère` *(fig. 4.9).*

4.3.2 Qualité du maillage obtenu

Le but premier de l'algorithme du Vertex Clustering est de réduire le nombre de mailles, et non d'améliorer la qualité de la triangulation. Malgré que l'algorithme ne tienne pas compte de la qualité des éléments formés, la réduction du nombre de triangles entraîne une légère amélioration de la qualité du maillage : en moyenne, la qualité[1] du pire élément du maillage passe de 0 (triangle (quasi) plat) à 0.26 et la qualité moyenne passe de 0.6 à 0.73. La répartition de la qualité des triangles est représentée, dans le cas de `la sphère`, à la figure 4.4.

[1]La mesure de qualité utilisée est celle conseillée par Frey (chap. 3).

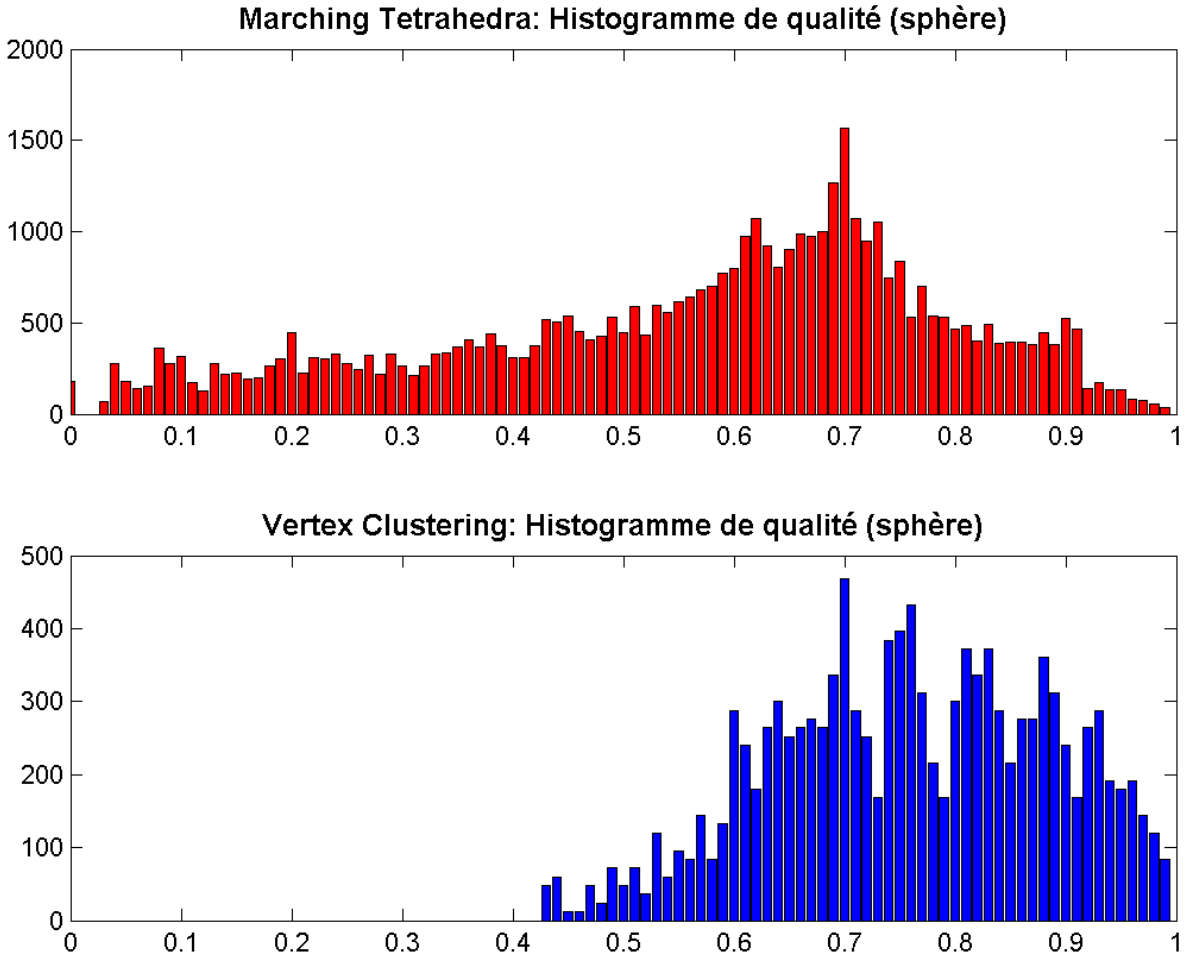


FIG. 4.4 – *Histogramme de la qualité des triangles du maillage de* `la sphère` *(fig. 4.9).*

4.3.3 Problème rencontré au niveau de la projection

En observant les résultats obtenus pour les différentes géométries *tests*, nous remarquons que les maillages *`du maximum entre un cube et une sphère`* et du *`minimum entre un cube et une sphère`* sont de qualité bien inférieure aux autres. Nous remarquons de plus que les maillages correspondants obtenus sans projection du *noeud représentatif* sur la surface sont bien plus beaux 4.5 car les triangles sont de meilleurs qualité. Ainsi, la projection du *noeud représentatif* sur la surface, indispensable pour avoir une bonne approximation de la géométrie, peut générer de mauvaises mailles.

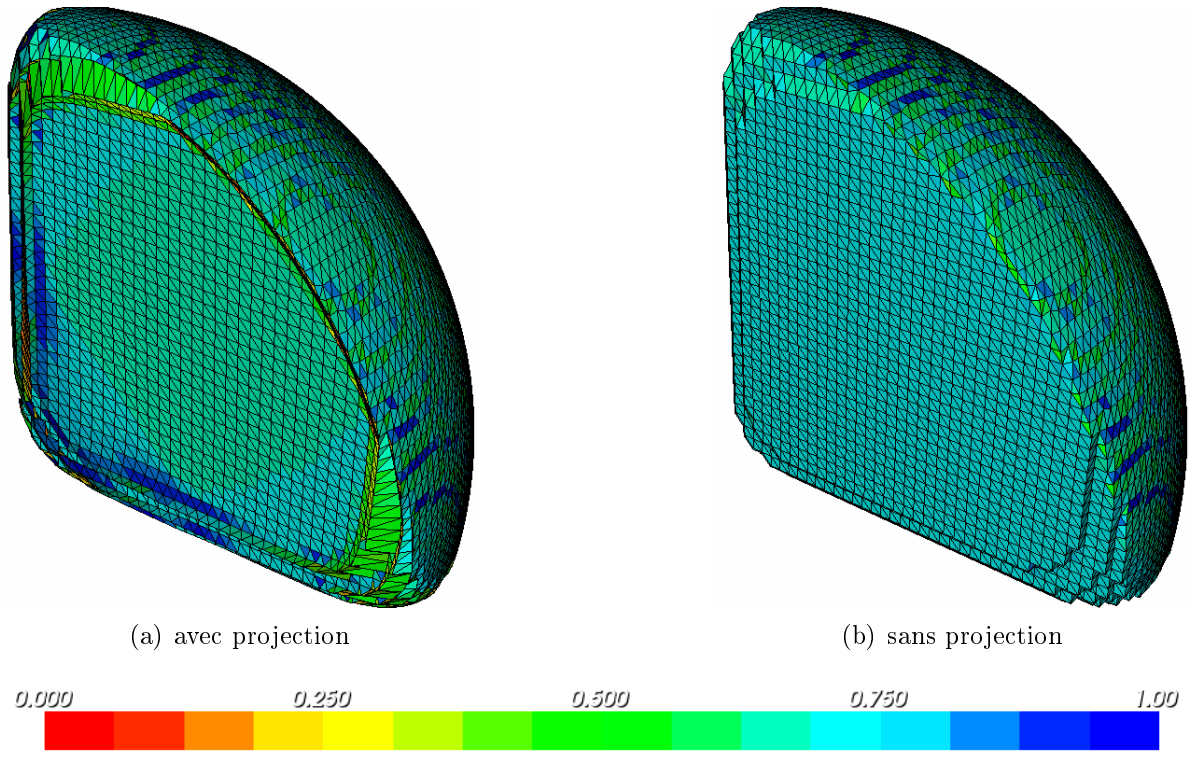
Dans le cas du *`maximum entre un cube et une sphère`*, l'erreur se situe au niveau de l'intersection entre le cube et la sphère. A cet endroit, la géométrie présente un angle vif et la dérivée de la fonction implicite est discontinue. Or, la direction de projection du noeud sur la surface est déterminée par un calcul du gradient de la fonction au point initial. Ainsi, deux points initialement proches pourraient être projetés dans des directions totalement différentes. Dans ce cas, une autre direction de projection aurait sans doute été préférable. Comme nous le verrons dans le chapitre 8, ce problème n'apparaîtra plus lors de la génération d'un maillage à partir d'une image 3D ; parce que dans ce cas, la dérivée sera déduite de l'image originale et sera toujours continue sur le domaine.

Dans le cas du *minimum entre un cube et une sphère*, les triangles de mauvaise qualité sont localisés au niveau de l'arrête du cube. A nouveau, la figure 4.6 montre que c'est la projection des noeuds sur la surface qui pose problème. Par contre, la frontière entre le cube et la sphère est bien traitée par l'algorithme.

Il existe plusieurs solutions possibles pour résoudre le problème :

1. Ajouter un critère lors de la projection qui permettrait de détecter les endroits ou le gradient est discontinu ; corriger ensuite la direction de projection (par exemple, en prenant une moyenne des normales aux triangles avoisinants).
2. Au lieu prendre pour le *noeud représentatif* une position moyenne des position des noeuds du même *cluster* ; choisir un des noeuds préexistants, comme dans la méthode de Rossignac et Borrel. De cette manière, aucune projection n'est nécessaire.

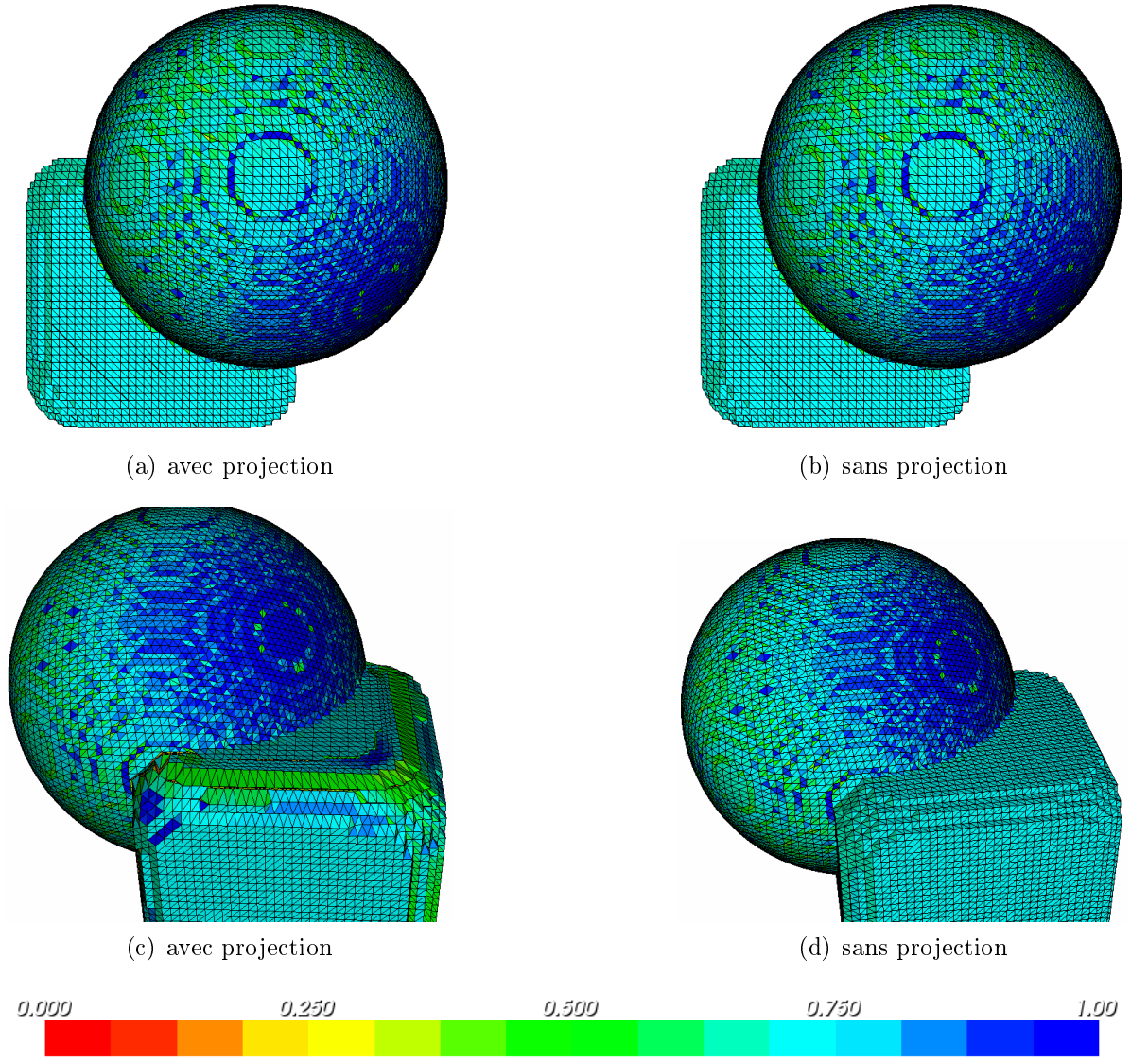
Les solutions proposées n'ont pas encore été implémentées, par manque de temps et parce que ces problèmes n'apparaissent plus lorsque la surface est définie par une image tridimensionnelle.



(a) avec projection (b) sans projection

(c) échelle de qualité

Fig. 4.5 – *Algorithme du Vertex Clustering appliqué sur le maillage du* ***maximum entre un cube et une sphère****. Effets indésirables engendrés par la projection des noeuds dans la direction du gradient initial.*



(a) avec projection

(b) sans projection

(c) avec projection

(d) sans projection

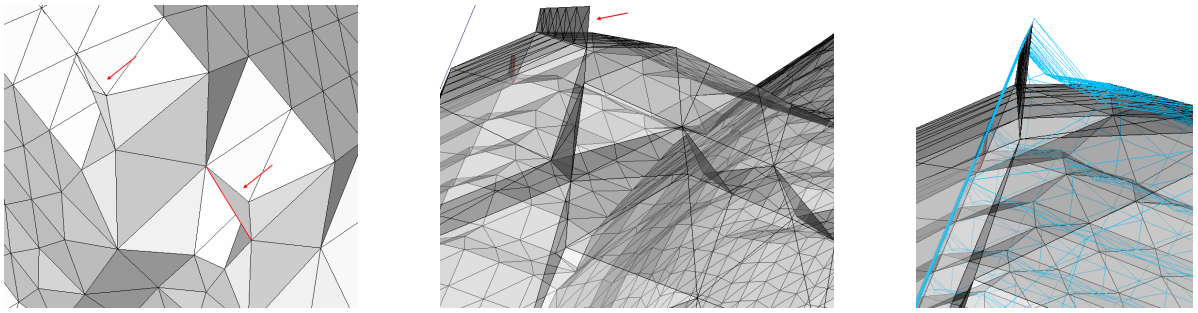
(e) échelle de qualité

Fig. 4.6 – *Algorithme du Vertex Clustering appliqué sur le maillage du* `minimum entre un cube et une sphère`*, avec et sans projection des noeuds sur la surface réelle (2 vues représentées).*

4.3.4 Problème rencontré lorsque les surfaces sont très irrégulières

Un inconvénient de la méthode du Vertex Clustering apparaît lorsqu'on tente de l'appliquer sur des surfaces très irrégulières. Ce problème a déjà été illustré lors de l'explication de la méthode, à la figure 4.1 : les trois triangles du coin inférieur gauche dégénèrent en une droite suite à l'application de l'algorithme. En pratique, ce problème est apparu lors de la génération d'un maillage surfacique au départ d'une image tridimensionnelle (fig. 4.7). Nous verrons dans le chapitre 9 que, dans ce cas, les surfaces frontières peuvent être très irrégulières. Ainsi, pour deux des huit géométries *tests* définies à l'aide d'une image 3D, des triangles dégénérés sont apparus.

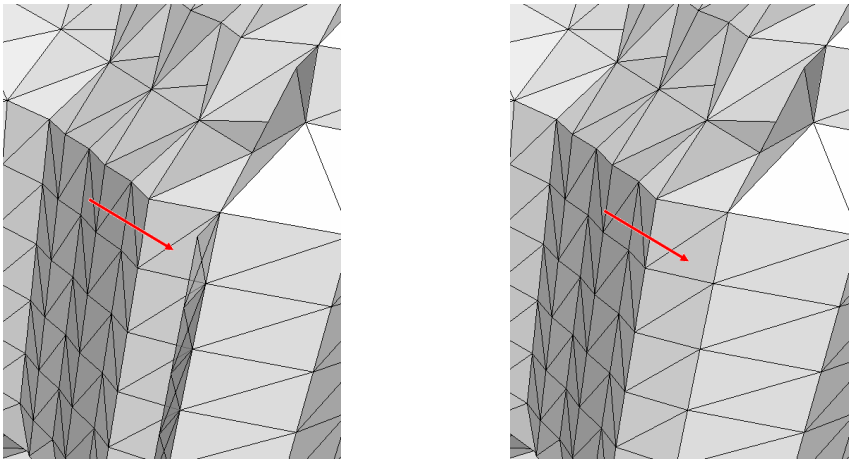
Dans l'explication de la méthode, nous avions également noté que l'algorithme du Vertex Clustering soude les objets qui sont proches. Le cas peut apparaître lorsque la distance entre les deux objets est inférieure l'espacement de la grille. A nouveau, ce problème n'apparaît pas pour les surfaces définies à l'aide d'une fonction implicite mais bien lorsque l'objet à mailler est contenu dans une image tridimensionnelle (fig. 4.7 (b,c)).



(a) (b) (c)

Fig. 4.7 – *Erreurs générées par l'algorithme du Vertex Clustering lorsque les surfaces sont fort irrégulières. L'arrête rouge signifie qu'elle ne possède qu'un seul triangle voisin. A la figure (c), le maillage bleu est le maillage issu de l'algorithme d'extraction de la surface, celui-ci a été superposé au maillage obtenu après application du Vertex Clustering.*

Des méthodes ont été implémentées pour tenter de corriger les erreurs. En particulier, le problème de la figure 4.8 est détecté en remarquant que deux triangles voisins ont des normales exactement opposées. Ces deux triangles sont ensuite supprimés du maillage. C'est en appliquant cette technique de manière itérative sur le maillage de la figure 4.8(a), que nous avons pu éliminer totalement la tranche de triangles collés 4.8(b). Cependant, cette méthode est assez coûteuse en temps de calcul puisqu'il faut calculer, pour chaque triangle, les normales des triangles voisins. Le problème présenté à la figure 4.7(a) n'a pas encore été résolu.



(a) avant correction (b) après correction

Fig. 4.8 – *Suppression des triangles collés, générés par l'algorithme du Vertex Clustering.*

Le mailleur surfacique *Isosurf* utilise également l'algorithme du Vertex Clustering. Afin d'éviter les problèmes ci-dessus, il effectue un lissage[2] des images définissant les surfaces avant de les traiter. Le lissage peut être désactivé, mais dans ce cas, *Isosurf* ne garantit plus que les maillages générés soient corrects. Nous avons vérifié qu'avec notre algorithme, un lissage préalable des images permet de supprimer totalement les erreurs décrites ci-dessus.

4.4 Visualisation des maillages obtenus pour les 8 géométries *tests*

Les figures 4.9 à 4.16 montrent les maillages obtenus après simplification des maillages issus de l'algorithme du Marching Tetrahedra. Les triangles ont été coloriés en fonction de leur qualité. La mesure de qualité utilisée est celle de Frey (eq. 3.2). L'échelle de couleurs utilisée celle de la figure 4.6(c). Cette échelle est la même pour tous les graphes. Comme expliqué ci-avant, *le maximum entre un cube et une sphère* et *le minimum entre un cube et une sphère* présentent des triangles de mauvaise qualité. La qualité des autres maillages est assez bonne, bien qu'elle pourrait être améliorée (chap. 5 et 6). Dans le maillage du *cône arrondi*, des triangles de qualité inférieure sont observés près des noeuds du maillage ne possédant que 4 triangles voisins. La suppression de tels noeuds fait l'objet du chapitre suivant (chap. 5). D'autre part, le maillage du *parallélépipède* pourrait être amélioré en déplaçant quelque peu ses noeuds. L'optimisation de la position des noeuds fait l'objet du chapitre 6. Enfin, en comparant les maillages à ceux du chapitre précédent (fig. 3.18 à 3.25), nous voyons que l'aire des triangles varie beaucoup moins d'un point à l'autre du maillage. Le Vertex Clustering nous a permit de rassembler en un seul les plus petits triangles générés par l'algorithme d'extraction de la surface.

[2] Ce lissage est réalisé au moyen d'une opération morphologique : par défaut il s'agit d'une ouverture suivie d'une fermeture.

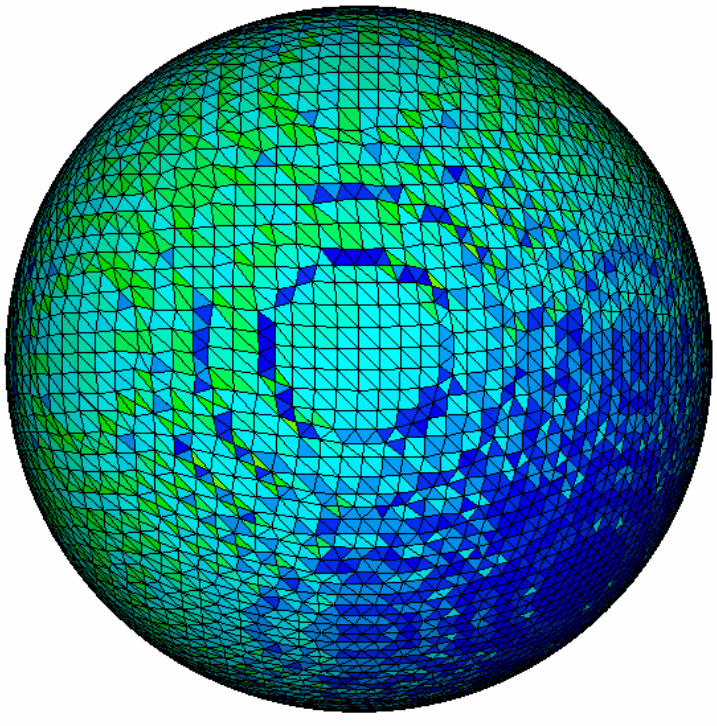


Fig. 4.9 – *Maillage surfacique de* `la sphère` *obtenu après application du Vertex Clustering.*

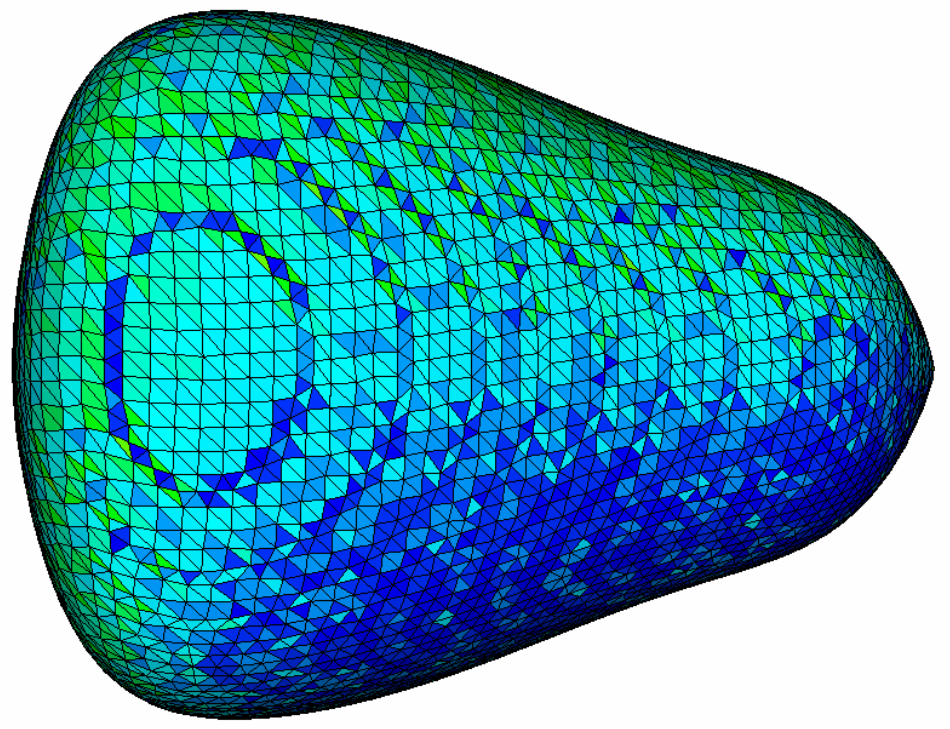


Fig. 4.10 – *Maillage surfacique* `du cône arrondi` *obtenu après application du Vertex Clustering.*

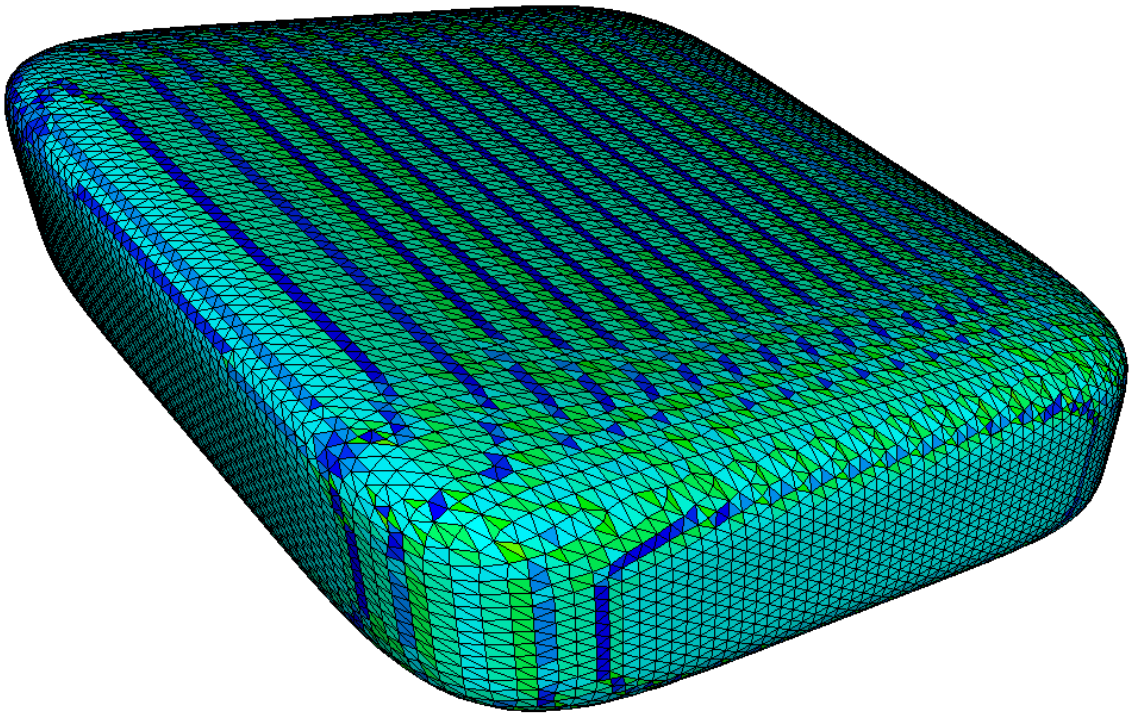


Fig. 4.11 – *Maillage surfacique* `du parallélépipède` *obtenu après application du Vertex Clustering.*

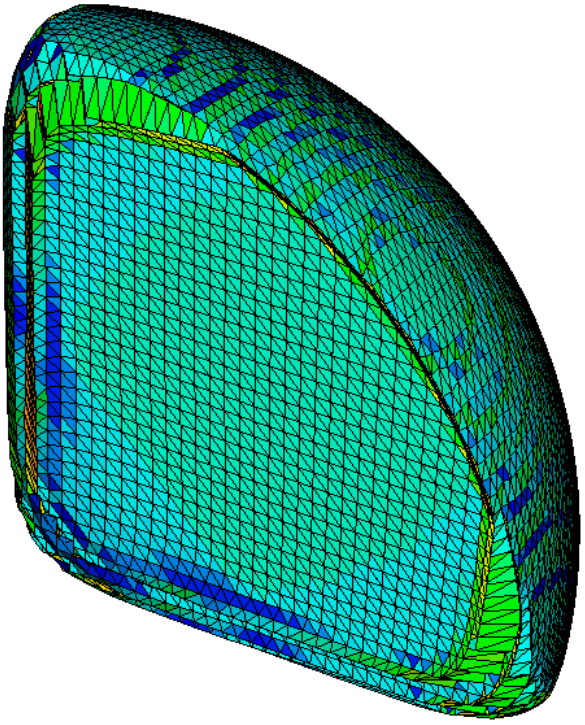


Fig. 4.12 – *Maillage surfacique* `du maximum entre un cube et une sphère` *obtenu après application du Vertex Clustering.*

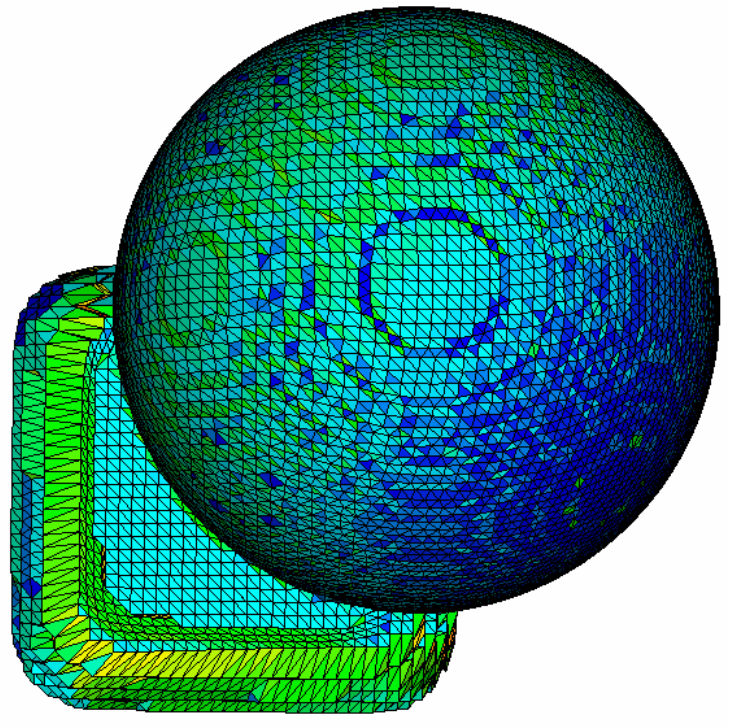


FIG. 4.13 – *Maillage surfacique* `du minimum entre un cube et une sphère` *obtenu après application du Vertex Clustering.*

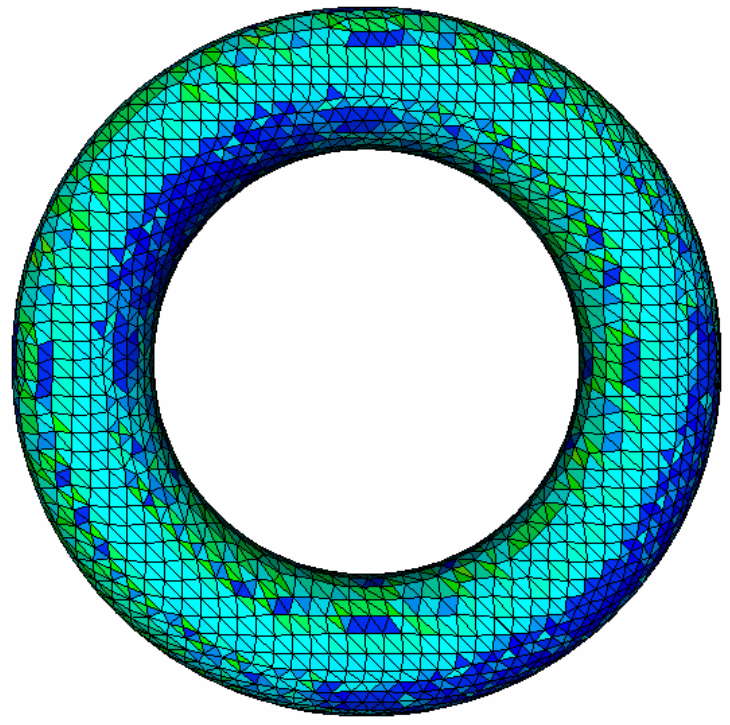


FIG. 4.14 – *Maillage surfacique* `du tore` *obtenu après application du Vertex Clustering.*

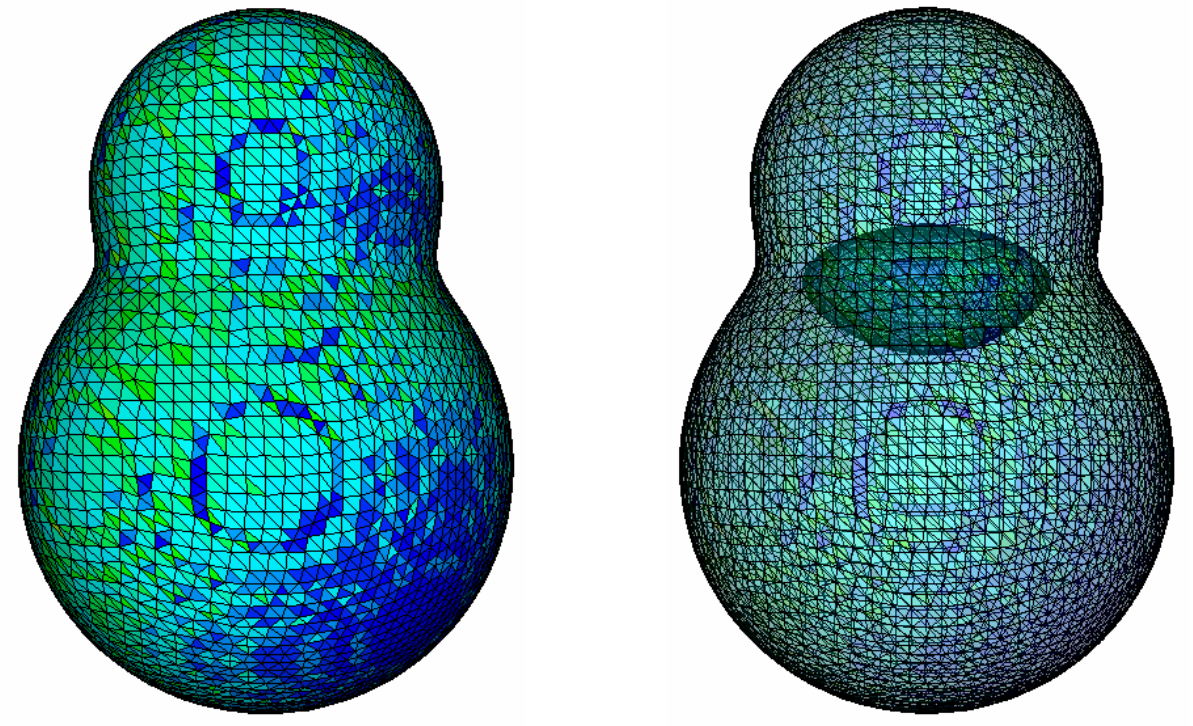


FIG. 4.15 – *Maillage surfacique de* `la multiplication de deux sphères` *obtenu après application du Vertex Clustering (le maillage de droite permet de visualiser la surface interne par transparence).*

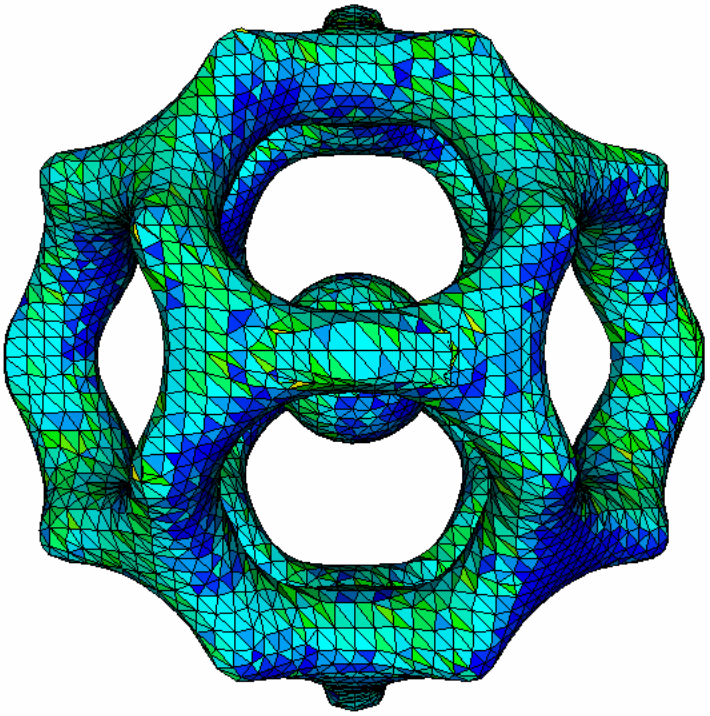


FIG. 4.16 – *Maillage surfacique de* `la géométrie basée sur le nombre d'or` *obtenu après application du Vertex Clustering.*

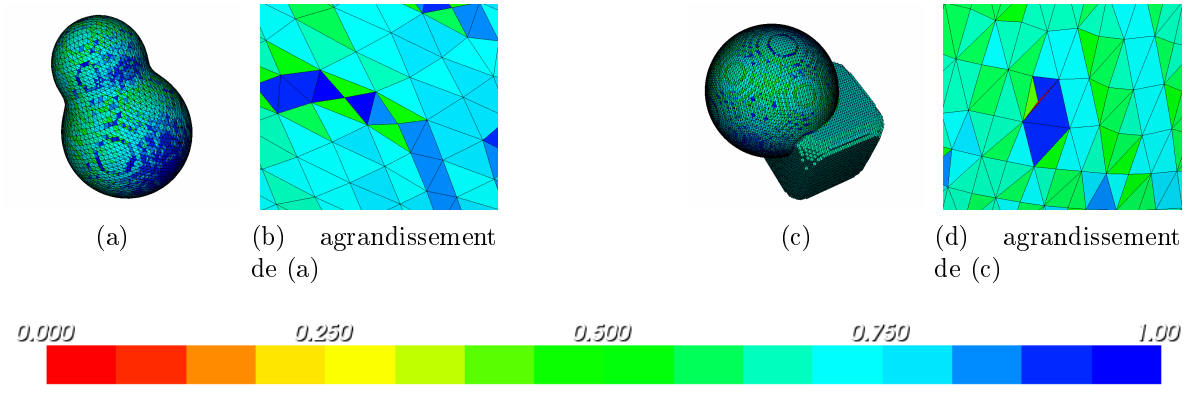
CHAPITRE 5

Amélioration de la topologie

L'algorithme créé dans ce chapitre permet d'améliorer la topologie, c'est-à-dire les relations entre les noeuds, du maillage obtenu en sortie de l'algorithme du Vertex Clustering.

5.1 Analyse de la topologie dans le maillage issu du Vertex Clustering

Si la surface était plane, une topologie parfaite impliquerait que chaque noeud ait 6 triangles voisins. Pour un partitionnement de l'espace suffisamment fin et des surfaces régulières, chaque noeud devrait donc être entouré par 6 triangles. Or si en moyenne il y a bien 6 triangles adjacents à chaque noeud, le nombre de voisins peut varier entre 3 et 10 triangles. Ces erreurs de topologie sont à l'origine de distorsions dans le maillage issu du Vertex Clustering et induisent des triangles de mauvaise qualité, qui ne pourront jamais devenir équilatéraux dans une dernière phase de lissage (chap. 6). Ceci est illustré à la figure 5.1. L'objet de ce chapitre est d'améliorer la topologie du maillage en éliminant les noeuds possédant soit 3, soit 4 triangles voisins.



(a) (b) agrandissement de (a) (c) (d) agrandissement de (c)

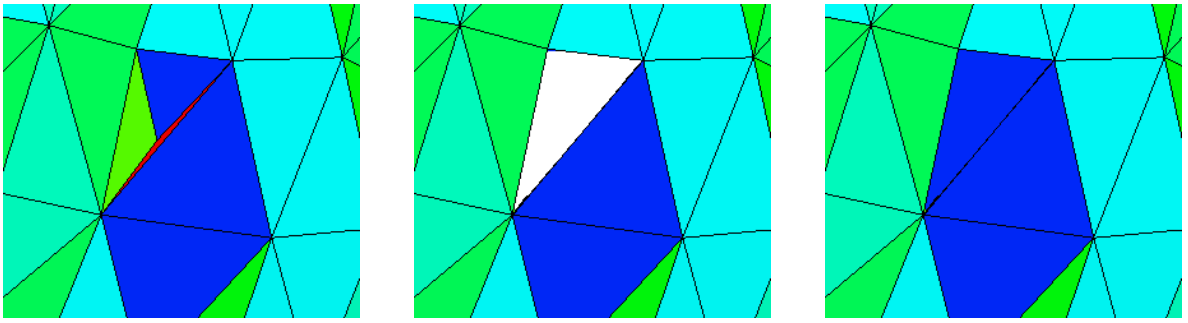
(e) échelle de couleurs pour la visualisation de la qualité des mailles

Fig. 5.1 – *Problèmes de topologie observés dans les maillages issus de l'algorithme du Vertex Clustering.*

5.2 Algorithme implémenté

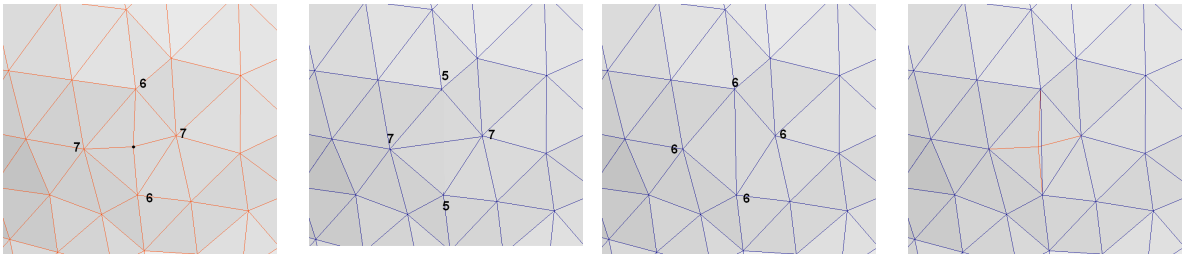
La première étape de l'algorithme consiste à parcourir les noeuds du maillage et marquer ceux qui n'ont que 3 ou 4 triangles voisins. Ces noeuds sont ensuite supprimés, de même que les triangles adjacents. Nous nous retrouvons donc avec un maillage comportant un trou à l'endroit où les triangles ont été supprimés. La suite de l'algorithme dépend de la forme du trou formé :

- Dans le cas où le noeud supprimé ne possédait que 3 triangles voisins, le trou formé a la forme d'un triangle et il suffit de l'ajouter au maillage (fig. 5.2).
- Par contre, si le noeud éliminé possédait 4 triangles, le trou formé est un quadrangle qu'il convient de retrianguler. Un quadrangle peut être triangulé de deux manières différentes, en choisissant l'une ou l'autre diagonale. La configuration choisie est celle qui permettra d'obtenir le maillage de la meilleure qualité après optimisation de la position des noeuds (chap. 6). La procédure est illustrée à la figure 5.3. La situation (c) donnera lieu à un meilleur maillage que la configuration (b), étant donnée qu'elle se rapproche plus de notre définition de topologie idéale. C'est donc en comptant le nombre de triangles associés aux noeuds voisins que nous trancherons entre les deux manières possibles de trianguler un quadrangle.



(a) avant correction (b) après correction (c) échelle de couleurs pour la visualisation de la qualité des mailles

Fig. 5.2 – *Suppression du noeud ne comportant que 3 triangles et formation d'un nouveau triangle.*



(a) maillage initial (b) retriangulation 1 : mauvais choix (c) retriangulation 2 : bon choix (d) superposition du maillage (a) et (c)

Fig. 5.3 – *Deux configurations possibles pour la retriangulation d'un quadrangle.*

5.3 Analyse et visualisation des résultats

L'algorithme implémenté fonctionne avec succès : il supprime sans exception tous les noeuds ne possédant que 3 ou 4 triangles. L'effet de l'amélioration de la topologie sur les différentes géométries est illustrée aux figures 5.4 à 5.11, où l'on a superposé le maillage ayant subit la correction de la topologie (en bleu) au maillage initial, issu du Vertex Clustering (en rouge). Les deux maillages sont quasi-identiques, seuls certains noeuds sont supprimés par l'algorithme d'amélioration de la topologie. Cette étape permet d'améliorer grandement l'efficacité du repositionnement des noeuds, traité dans le chapitre suivant.

Topologie									**moyenne**
nombre de noeuds	5312	4685	10150	9627	11474	2656	4337	5274	6689
nombre de triangles	10620	9366	20296	19250	22944	5312	8666	10584	13380
nombre de noeuds	5312	4685	10150	6515	11474	2656	4337	5274	6300
nombre de triangles	10620	9366	20296	13026	22944	5312	8666	10584	12602
nombre de noeuds	5312	4685	10150	6515	11474	2656	4337	5274	6300
nombre de triangles	10620	9366	20296	13026	22944	5312	8666	10584	12602
Longueurs									
Minimum	0,63	0,58	0,62	0,05	0,04	0,61	0,42	0,39	0,42
Maximum	2,67	2,73	2,75	4,58	4,12	2,91	2,69	2,68	3,14
Moyenne	0,99	1,01	1,03	1,32	0,98	1,00	1,01	1,01	1,04
Aires									
Minimum	0,22	0,21	0,22	0,01	0,01	0,22	0,18	0,02	0,13
Maximum	0,98	1,08	1,04	2,97	2,55	1,85	1,02	1,04	1,57
Moyenne	0,43	0,45	0,51	0,78	0,49	0,46	0,44	0,46	0,50
Qualité selon Semenova									
Pire triangle	0,52	0,43	0,43	0,00	0,01	0,41	0,43	0,06	0,29
Meilleur triangle	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
Moyenne	0,90	0,89	0,85	0,83	0,86	0,89	0,90	0,89	0,88
Qualité selon Frey									
Pire triangle	0,40	0,33	0,33	0,00	0,01	0,31	0,33	0,04	0,22
Meilleur triangle	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
Moyenne	0,78	0,77	0,71	0,68	0,71	0,76	0,78	0,77	0,75
Temps de calcul [s]	4	3	6	5	17	1	3	5	6
Vitesse [triangles/s]	2857	3454	3356	2467	1338	3583	3117	2098	2280

Tab. 5.1 – *Algorithme permettant d'améliorer la topologie : Résultats numériques obtenus pour les 8 géométries tests, dont les maillages sont présentés aux figures 5.4 à 5.11.*

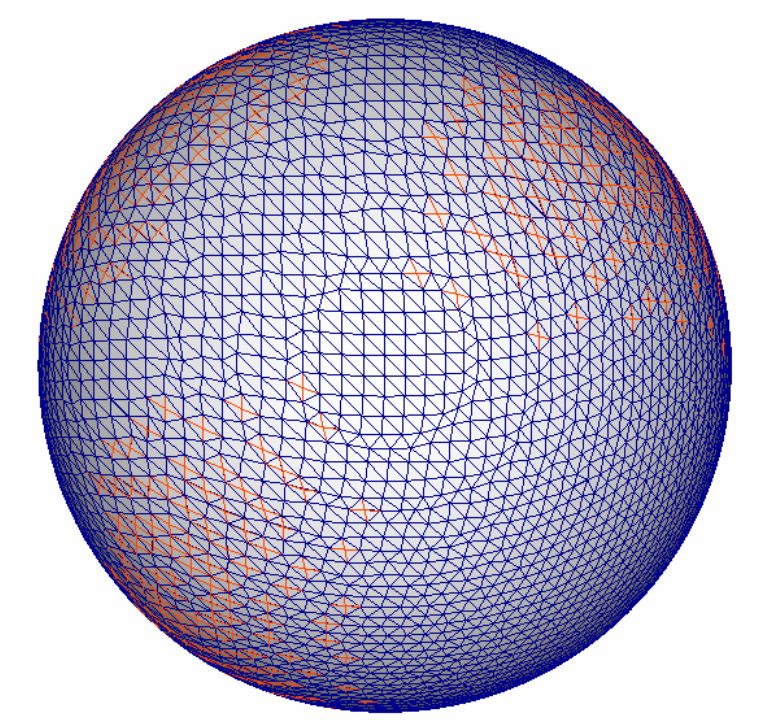


Fig. 5.4 – *Amélioration de la topologie du maillage de* `la sphère`.

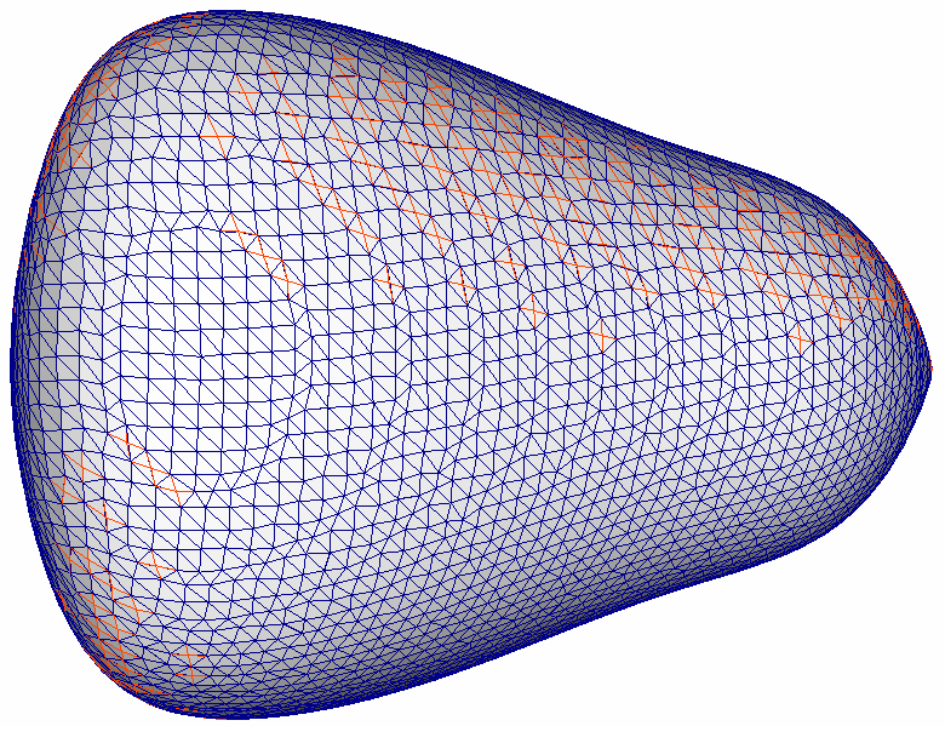


Fig. 5.5 – *Amélioration de la topologie du maillage* `du cône arrondi`.

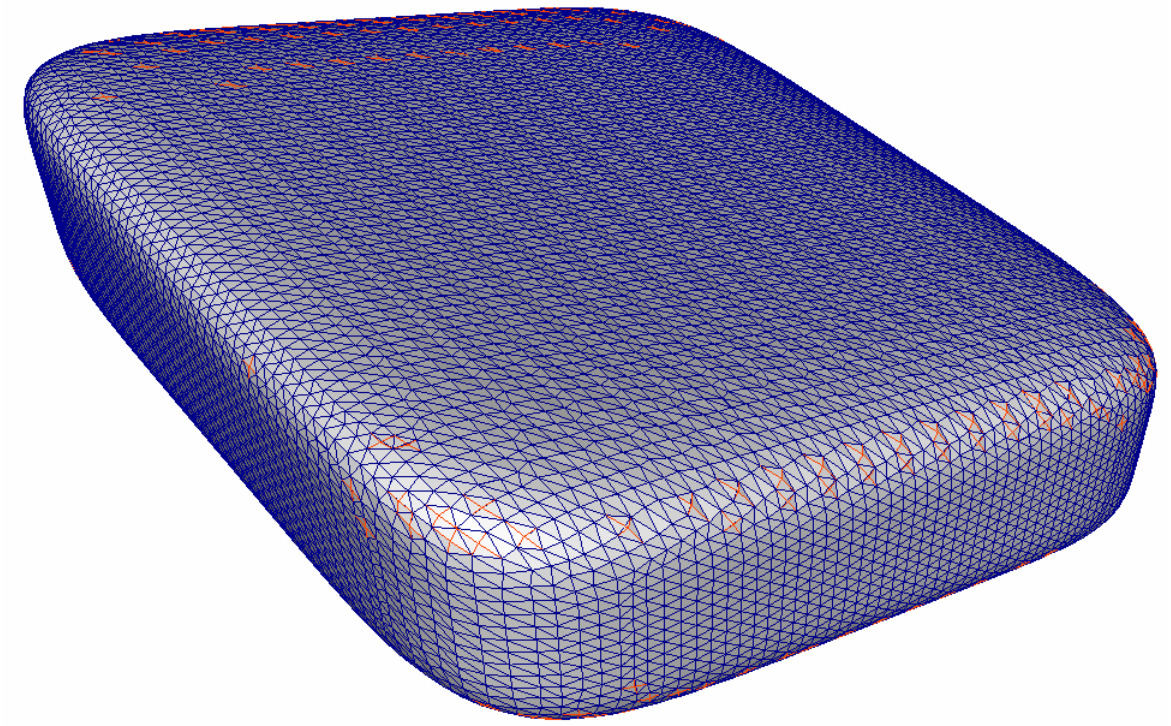


FIG. 5.6 – *Amélioration de la topologie du maillage* `du parallélépipède`.

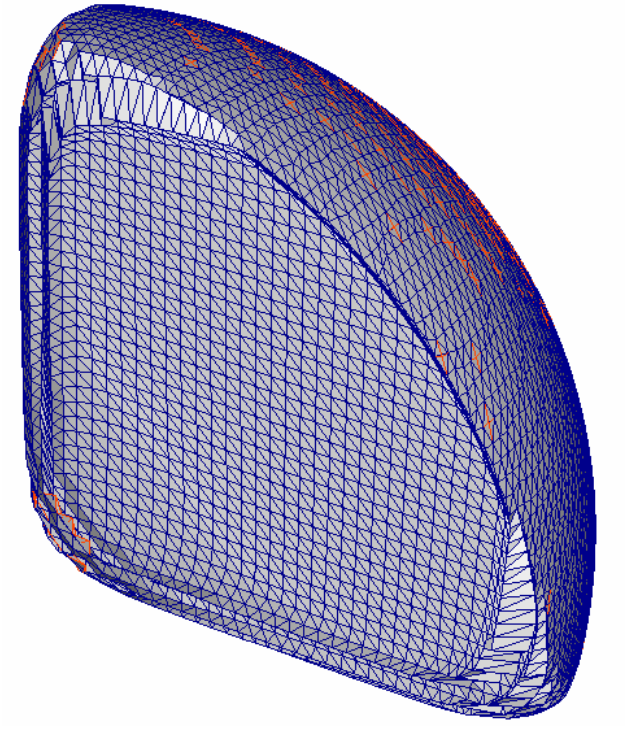


FIG. 5.7 – *Amélioration de la topologie du maillage* `du maximum entre un cube et une sphère`.

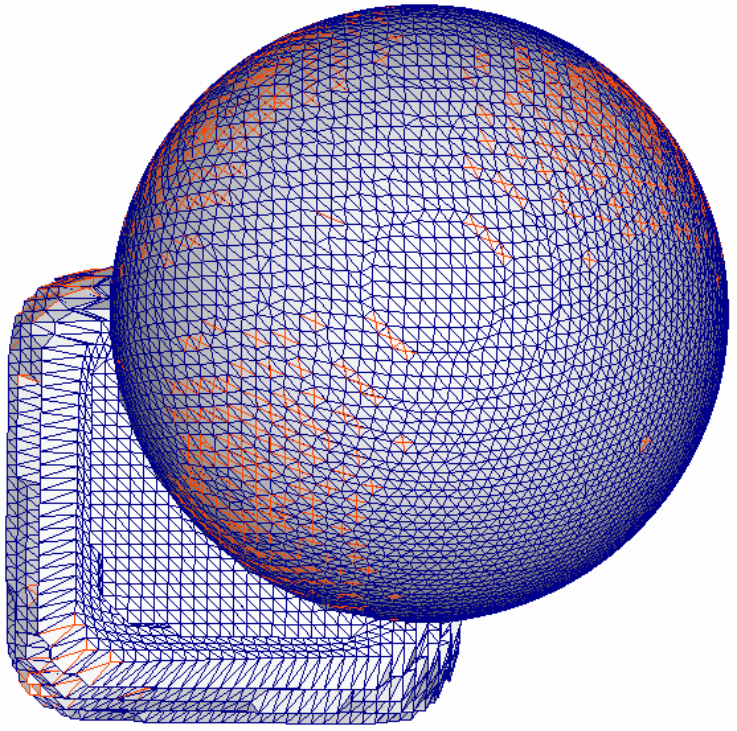


FIG. 5.8 – *Amélioration de la topologie du maillage* `du minimum entre un cube et une sphère`.

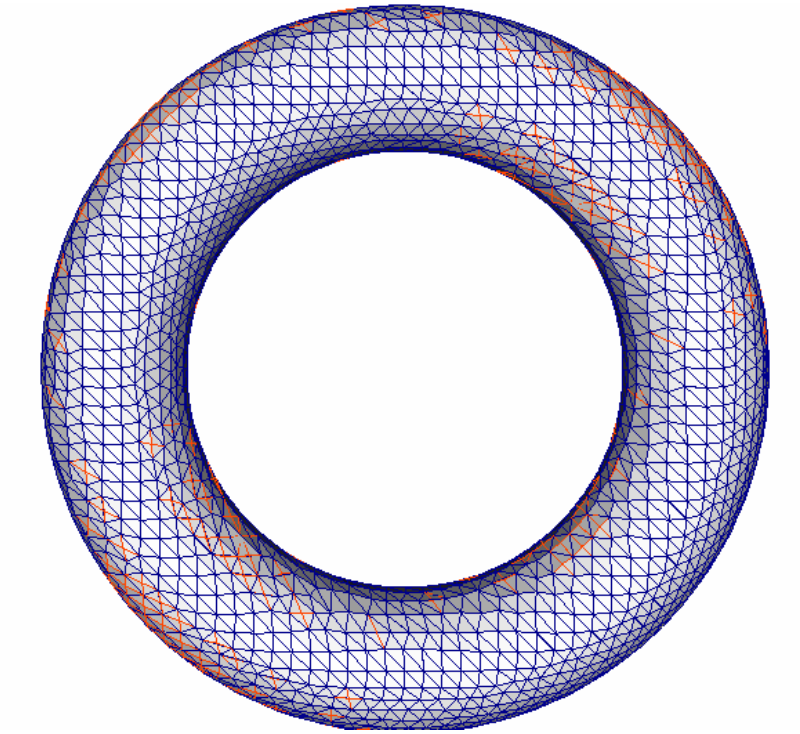


FIG. 5.9 – *Amélioration de la topologie du maillage* `du tore`.

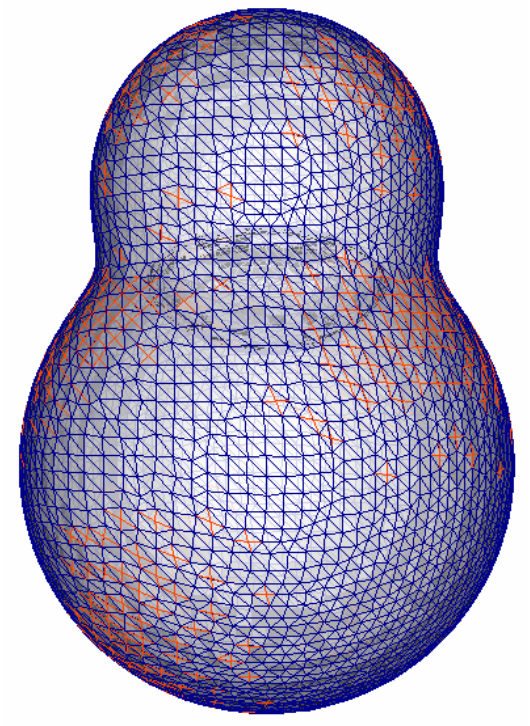


FIG. 5.10 – *Amélioration de la topologie du maillage de* `la multiplication de deux sphères`.

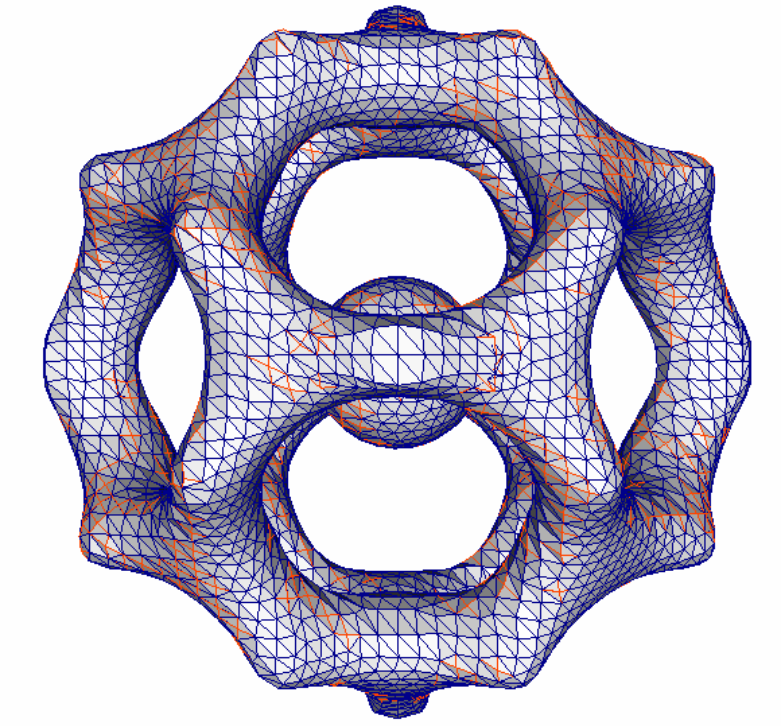


FIG. 5.11 – *Amélioration de la topologie du maillage de* `la géométrie basée sur le nombre d'or`.

CHAPITRE 6

Amélioration de la qualité des mailles

La qualité du maillage a beaucoup d'importance dans le cadre des simulations éléments finis. Dans ce chapitre, nous créons un algorithme permettant d'améliorer le maillage en déplaçant de manière itérative les noeuds le constituant. Le nombre de mailles et la topologie restent inchangés.

6.1 Aperçu des méthodes proposées dans la littérature

Les méthodes généralement utilisées et trouvées dans la littérature consistent à utiliser des techniques d'optimisation non linéaire pour guider le repositionnement des noeuds et atteindre un maillage de qualité. La plupart du temps, la condition d'application de la méthode est que le maillage initial soit valide. Pour que la maillage puisse être optimisé, il faut que la fonction objective possède un minimum, ce qui est le cas si les conditions suivantes sont remplies :

1. Le gradient de la fonction objective doit être nul au minimum.
2. Le Hessien doit être semi-défini positif au minimum.

L'approche est soit locale, soit globale.

Approche globale : Dans une approche globale, une seule fonction objective est utilisée pour l'entièreté du maillage initial. Sa minimisation permet de déterminer l'ensemble des nouvelles positions nodales simultanément. L'inconvénient des méthodes globales est qu'elles nécessitent la résolution de systèmes d'équations à grand nombre d'inconnues [38, 63].

Approche locale : Les méthodes d'optimisations locales résolvent le problème d'optimisation dans le voisinage d'un noeud. Les noeuds sont parcourus un à un et une méthode d'optimisation est utilisée pour améliorer la qualité des quantités géométriques du noeud. Différentes fonctions objectives sont proposées et comparées dans [35] et [36]. Selon ces articles, il est possible d'obtenir des éléments de qualité sur l'entièreté du maillage en n'étudiant que des fonctions objectives locales.
Dans [19], la surface formée du noeud courant et des triangles adjacents est projetée sur un plan, dans lequel la position du noeud central est optimisée. L'orientation du plan peut également résulter d'une optimisation.

Dans [26], Knupp et Garimella définissent un espace paramétrique local au noeud courant, dans lequel le noeud est repositionné.

6.2 Méthode implémentée

La méthode implémentée est une version simplifiée de [19]. L'algorithme implémenté se déroule comme suit :

1. Le maillage est parcouru de noeud en noeud.
2. Le *patch* formé du noeud courant et des triangles avoisinants est projeté sur un plan tangent à la surface au noeud central. La direction de projection est calculée en prenant le gradient de la fonction implicite au noeud courant.
3. Dans le plan de projection, le noeud central est repositionné au centre de ses noeuds adjacents.
4. Le noeud central, déplacé en 2D, est projeté sur la surface tridimensionnelle initiale. La direction de projection utilisée est la même qu'à l'étape 2 mais son sens est inversé.
5. Lorsque tout le maillage a été parcouru, une nouvelle itération sur l'ensemble des noeuds du maillage commence.
6. L'algorithme s'arrête lorsqu'une des conditions suivantes est remplie :
   (a) Le déplacement maximal réalisé sur l'ensemble du maillage est inférieur au dixième de la diagonale du cube élémentaire de la grille utilisée pour le Vertex Clustering.
   (b) La proportion de noeuds ayant subit un déplacement non négligeable est inférieur à 1 pourcent.
   (c) Le nombre d'itérations est supérieur à 10.

Cette méthode peut être vue comme un lissage du maillage, réalisé de manière à ce que les noeuds du maillage restent sur la surface réelle.

6.3 Présentation et analyse des résultats

A nouveau, les résultats obtenus pour les différentes géométries tests sont présentées en fin de chapitre (fig. 6.5 à 6.12). Le tableau ci-dessous reprend les résultats numériques obtenus.

Lissage sous contraintes									moyenne
nombre de noeuds	5312	4685	10150	6515	11474	2656	4337	5274	6300
nombre de triangles	10620	9366	20296	13026	22944	5312	8666	10584	12602
Longueurs									
Minimum	0,63	0,68	0,72	0,08	0,12	0,70	0,58	0,47	0,50
Maximum	2,69	2,69	2,28	5,11	2,74	2,48	2,76	2,88	2,95
Moyenne	1,00	1,00	1,05	1,34	0,99	1,00	0,99	1,02	1,05
Aires									
Minimum	0,25	0,25	0,29	0,02	0,01	0,28	0,20	0,15	0,18
Maximum	1,16	1,23	1,09	2,76	1,58	1,33	1,35	1,54	1,51
Moyenne	0,45	0,46	0,52	0,76	0,45	0,45	0,44	0,44	0,50
Qualité selon Semenova									
Pire triangle	0,39	0,50	0,48	0,02	0,08	0,56	0,48	0,36	0,36
Meilleur triangle	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
Moyenne	0,92	0,90	0,86	0,86	0,90	0,91	0,92	0,93	0,90
Qualité selon Frey									
Pire triangle	0,31	0,39	0,37	0,01	0,07	0,44	0,37	0,28	0,28
Meilleur triangle	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
Moyenne	0,80	0,79	0,71	0,72	0,77	0,79	0,81	0,81	0,78
Temps de calcul [s]	47	41	87	202	348	23	50	110	114
Vitesse [triangles/s]	228	231	233	64	66	229	173	96	111

Tab. 6.1 – *Algorithme d'optimisation de la position des noeuds : Résultats numériques obtenus pour les 8 géométries tests, dont les maillages sont présentés aux figures 4.9 à 4.16.*

6.3.1 Qualité des triangles générés

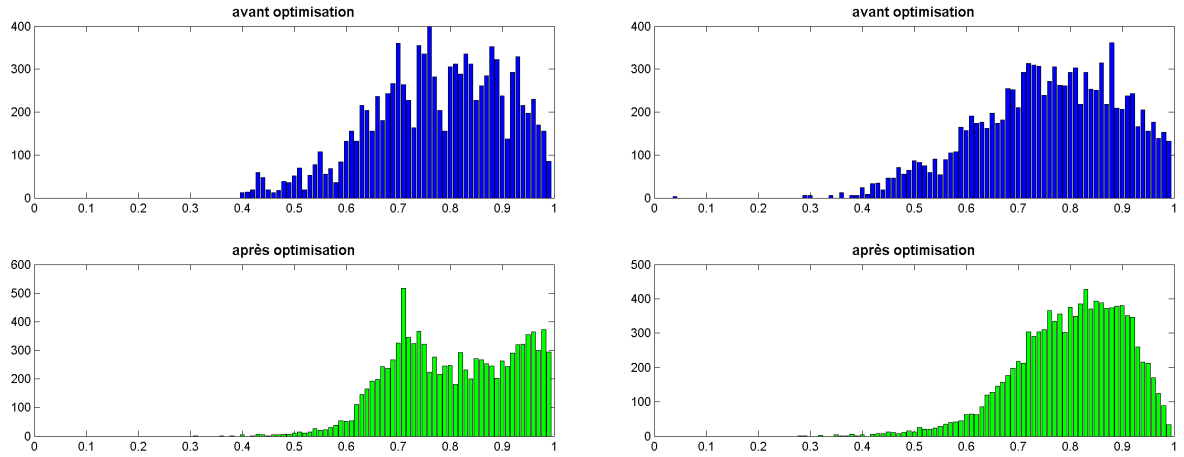
Pour chacune des géométries *tests*, nous avons mesuré la qualité du pire triangle, celle du meilleur triangle et la qualité moyenne sur le maillage. Ces mesures de qualités ont été prises avant et après l'exécution de l'algorithme d'optimisation. Le tableau 6.2 indique une augmentation moyenne de la qualité de 0.75 à 0.78.

Qualité selon Frey									moyenne
avant optimisation									
Pire triangle	0,40	0,33	0,33	0,00	0,01	0,31	0,33	0,04	0,22
Meilleur triangle	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
Moyenne	0,78	0,77	0,71	0,68	0,71	0,76	0,78	0,77	0,75
après optimisation									
Pire triangle	0,31	0,39	0,37	0,01	0,07	0,44	0,37	0,28	0,28
Meilleur triangle	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
Moyenne	0,80	0,79	0,71	0,72	0,77	0,79	0,81	0,81	0,78

Tab. 6.2 – *Mesures de qualité (selon Frey) effectuées avant et après l'algorithme d'optimisation.*

Il peut être intéressant de voir comment cette qualité se réparti sur les triangles du maillage et comment cette répartition est modifiée par l'algorithme. Cette répartition est montrée pour *la sphère* ainsi que pour *la géométrie basée sur le nombre d'or* à la figure 6.1. Nous remarquons que dans le cas de *la sphère*, le maillage surfacique obtenu en sortie de l'algorithme d'amélioration de la topologie était déjà de fort bonne qualité et que par conséquent, l'algorithme d'optimisation de la position des noeuds n'a pas une grande influence. Par contre,

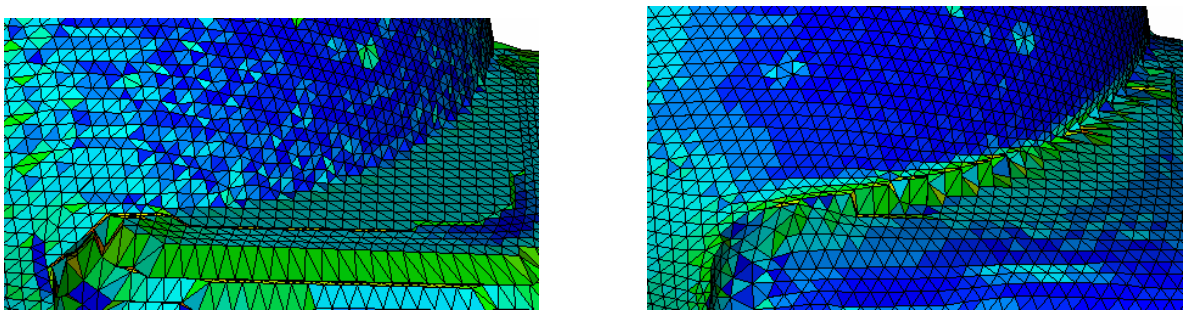
pour les surfaces plus complexes telles *la géométrie basée sur le nombre d'or*, un repositionnement des noeuds déplace le pic de qualité vers les qualités supérieures (fig. 6.1(b)).



(a) *la sphère* (b) *la géométrie basée sur le nombre d'or*

Fig. 6.1 – *Histogramme de qualité avant et après optimisation de la position des noeuds, cas de la sphère et de la géométrie basée sur le nombre d'or.*

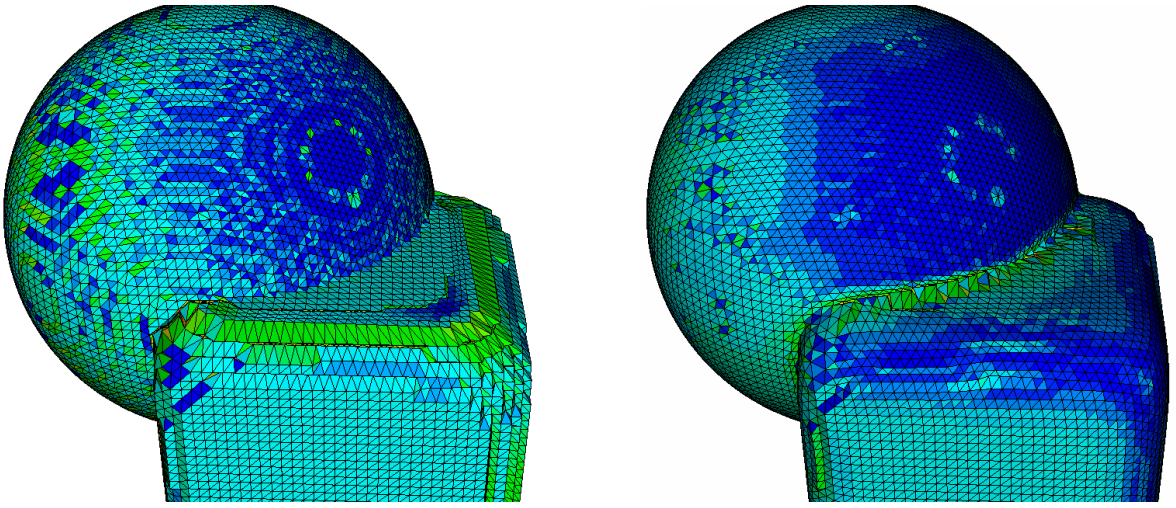
La projection suivant la direction du gradient de la fonction crée des problèmes aux endroits où ce gradient est discontinu. Ceci est observé pour *le minimum entre un cube et une sphère*, à l'intersection de cube et de la sphère (fig. 6.2). Une solution serait de projeter non pas selon le gradient de la fonction implicite au noeud central (calculé à l'aide de la dérivée de la fonction implicite), mais selon la moyenne des normales des triangles adjacents au noeud central.



(a) avant optimisation (b) après optimisation

Fig. 6.2 – *L'algorithme d'optimisation fournit des triangles de mauvaise qualité à l'intersection entre le cube et la sphère.*

Par contre, l'algorithme d'optimisation permet d'améliorer très fortement la qualité des triangles situés sur les côtés du cube du *minimum entre un cube et une sphère* (fig. 6.3).



(a) avant optimisation (b) après optimisation

Fig. 6.3 – *L'algorithme d'optimisation améliore fortement la qualité des triangles situés sur les bord du cubes.*

6.3.2 Temps de calcul

Le temps de calcul nécessaire à l'exécution de l'algorithme dépend directement du nombre d'itérations nécessaires. En pratique, ce nombre est inférieur à 4 pour les surfaces simples comme *la sphère*, *le cône arrondi* et *le tore*, et augmente pour les géométries plus compliquées telles *la multiplication de deux sphères* et *la géométrie basée sur nombre d'or* (6 à 7 itérations nécessaires). Dans le tableau 6.1, nous lisons un temps de calcul anormalement élevé pour *le maximum entre un cube et une sphère* et *le minimum entre un cube et une sphère*. Ceci est dû au problème de discontinuité du gradient de la fonction implicite au niveau de la frontière entre le cube et la sphère. La direction de projection change d'une itération à l'autre, de sorte que les noeuds proches de la frontière subissent, à chaque itération, un déplacement non négligeable. Puisque les deux premiers critères d'arrêt nous sont jamais atteints, le nombre d'itérations nécessaire à l'exécution de l'algorithme est égal au nombre maximal imposé (10 itérations) (sec. 6.2).

6.4 Visualisation des maillages obtenus pour les 8 géométries *tests*

L'algorithme d'optimisation de la position des noeuds constitue la dernière étape du mailleur surfacique créé. Ainsi, les maillages présentés aux figures 6.5 à 6.12 sont les maillages finaux, fournis en sortie du programme entier.

Les deux géométries obtenues par fusion et intersection entre un cube et une sphère posent toujours problème. Cependant, puisque la source d'erreur est la définition de la fonction implicite, ces problèmes n'apparaîtront plus lorsque nous appliquerons notre programme sur des images tridimensionnelles de type IRM (chap. 8). La qualité des autres maillages est suffisante pour permettre un calcul éléments finis. Des géométries complexes telles *la multiplication de deux sphères* et *la géométrie basée sur le nombre d'or* sont maillées avec succès par

notre mailleur surfacique. En particulier, la figure 6.4 montre que la surface intérieure de *la multiplication de deux sphères* est de bonne qualité.

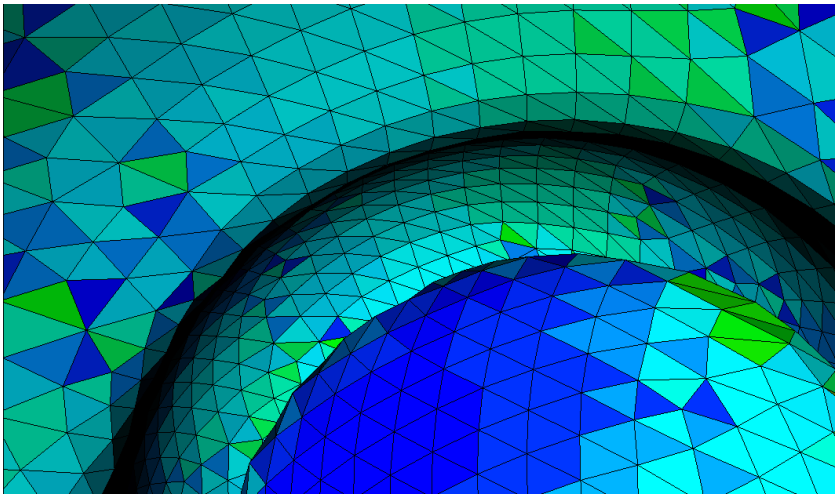


Fig. 6.4 – *La surface intérieure du* *maximum entre un cube et une sphère* *est maillée avec succès par notre mailleur surfacique.*

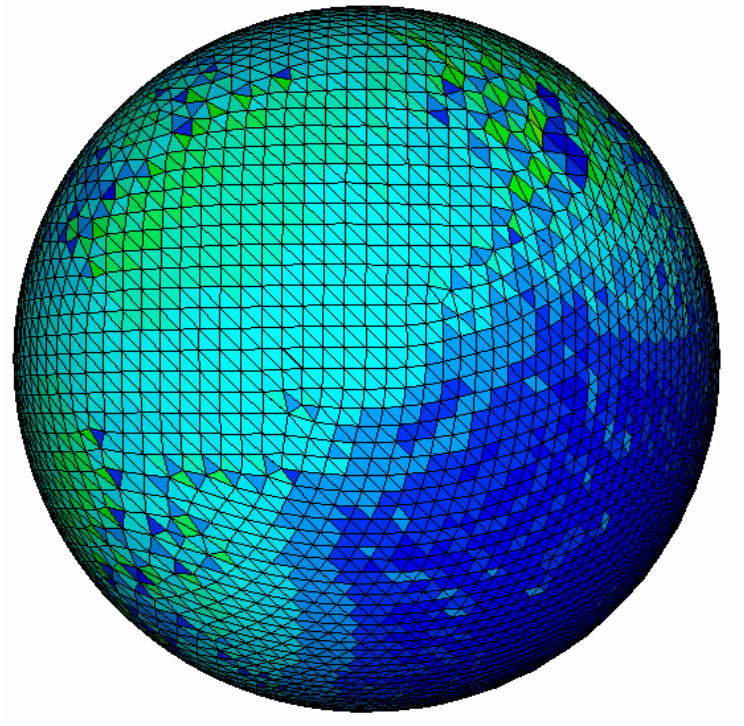


FIG. 6.5 – *Résultat du mailleur surfacique créé sur* `la sphère`.

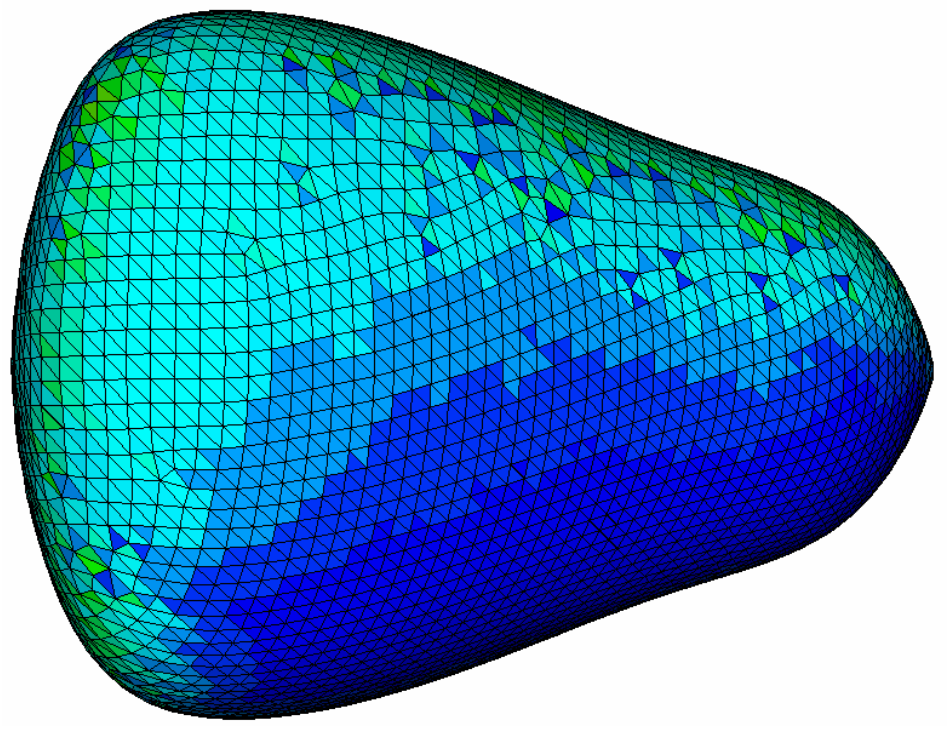


FIG. 6.6 – *Résultat du mailleur surfacique créé sur* `le cône arrondi`.

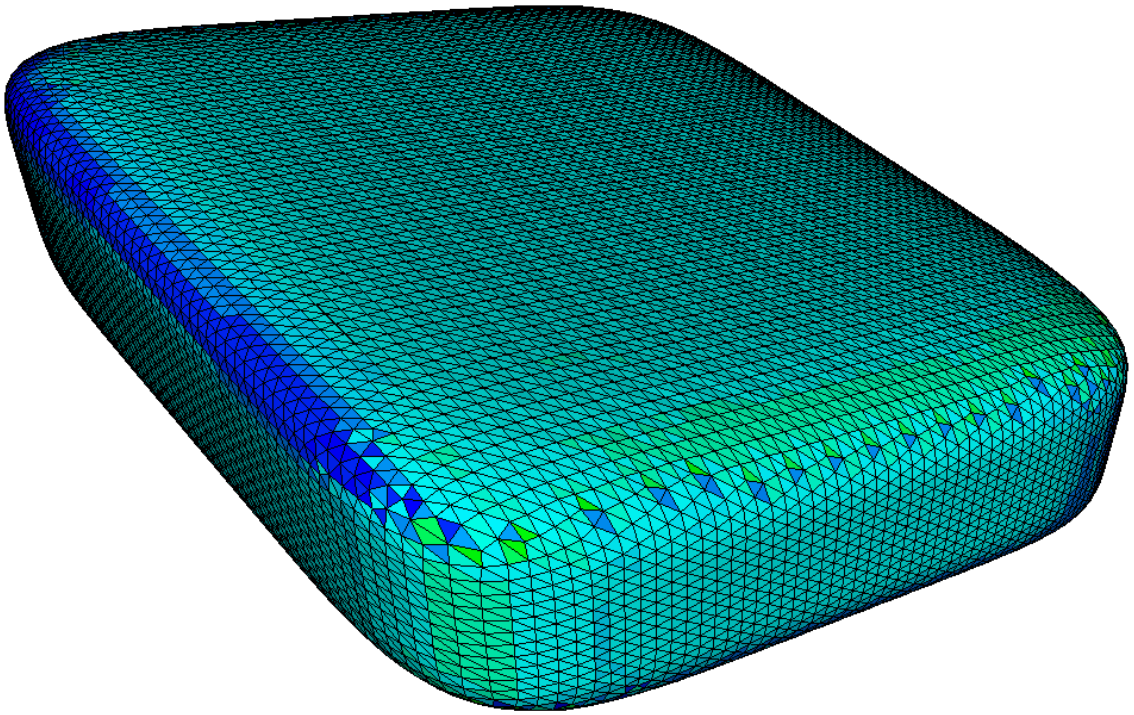


FIG. 6.7 – *Résultat du mailleur surfacique créé sur le parallélépipède.*

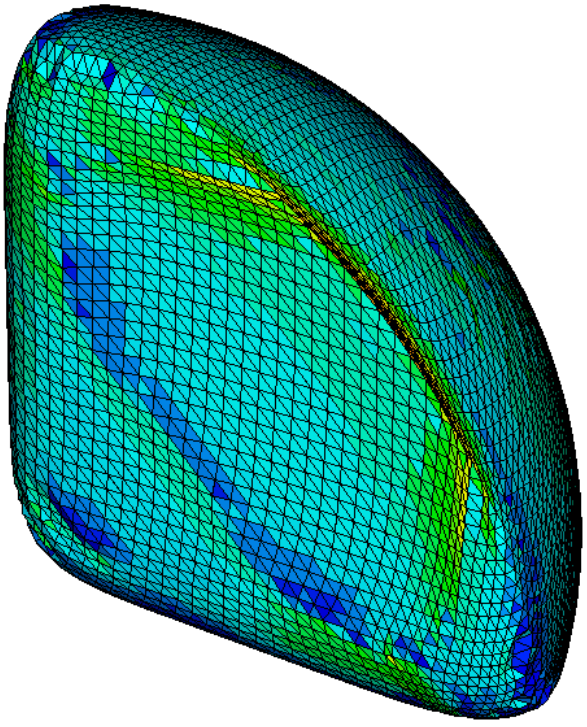


FIG. 6.8 – *Résultat du mailleur surfacique créé sur le maximum entre un cube et une sphère.*

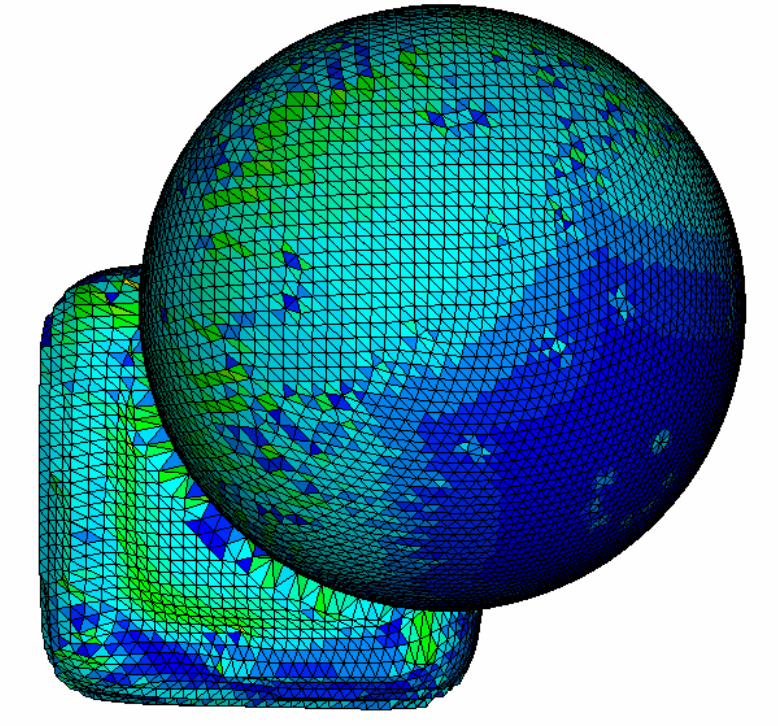


FIG. 6.9 – *Résultat du mailleur surfacique créé sur* `le minimum entre un cube et une sphère`.

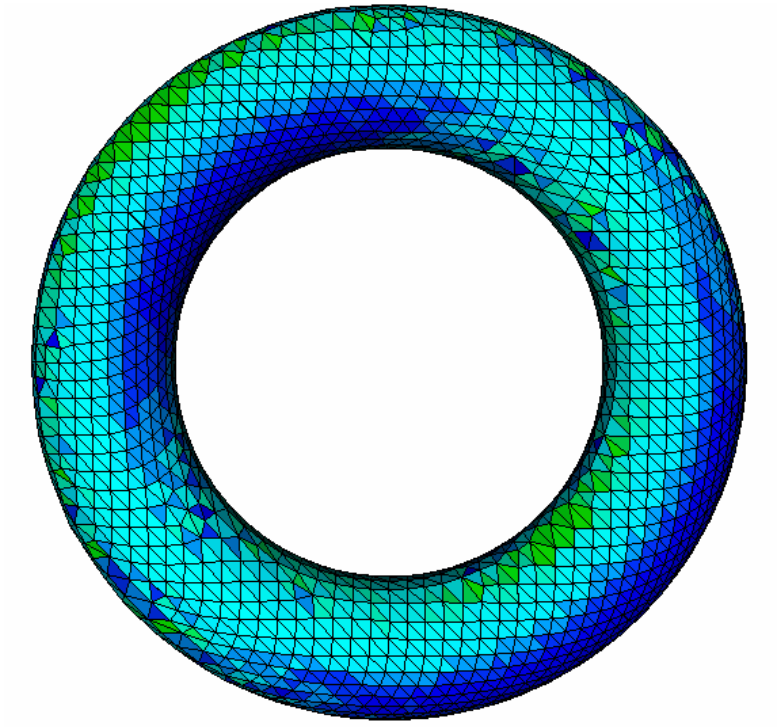


FIG. 6.10 – *Résultat du mailleur surfacique créé sur* `le tore`.

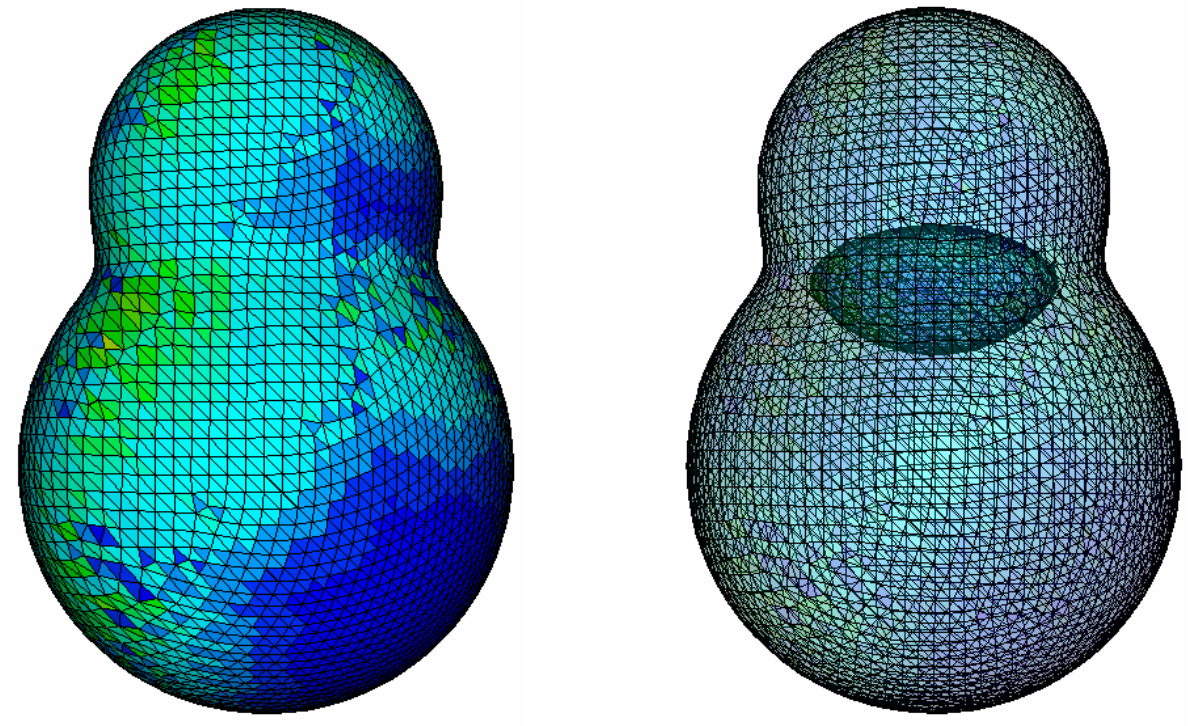


FIG. 6.11 – *Résultat du mailleur surfacique créé sur* `la multiplication de deux sphères` *(le maillage de droite permet de visualiser la surface interne par transparence).*

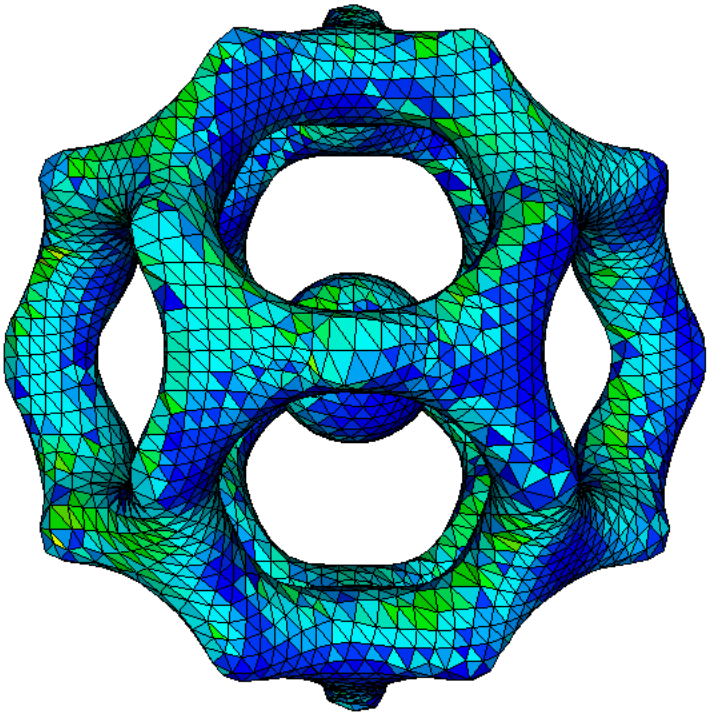


FIG. 6.12 – *Résultat du mailleur surfacique créé sur* `la géométrie basée sur nombre d'or`.

Deuxième partie

Application aux fonctions implicites provenant d'une segmentation d'image médicale

CHAPITRE 7

Introduction

L'objectif de cette seconde partie est de créer une triangulation de surface à partir d'images médicales 3D de type IRM. Ces images tridimensionnelles en niveaux de gris ont été préalablement filtrées et corrigées afin d'améliorer leur qualité. Elles ont ensuite été segmentées pour délimiter clairement les frontières. Pour se simplifier la tâche et se focaliser rapidement sur l'aspect mécanique du problème, ces deux premières opérations de traitement d'images sont supposées effectuées et ne seront pas abordées dans ce travail de fin d'études.

(a) Image 3D initiale (b) Image 3D segmentée

Fig. 7.1 – *Les images introduites dans notre mailleur surfacique ont été préalablement filtrées et segmentées.*

L'approche généralement utilisée consiste à définir une fonction de distance signée de l'image 3D segmentée et à en calculer le zéro. Cette fonction associe à chaque voxel de l'image, sa distance à la surface. Par convention, on donne une valeur positive aux voxels situées à l'intérieur de la surface et une valeur négative aux voxels extérieurs. La fonction de distance signée peut être vue comme une fonction implicite discrète dont la valeur est connue uniquement aux points d'échantillonnage (les voxels). Elle permet de se ramener au cas des chapitres précédents où la surface à trianguler est l'isosurface de niveau zéro.

Les images IRM se présentent sous forme de coupes successives. La résolution spatiale des images du type IRM est liée à l'intensité du champ magnétique (de nos jours, les appareils utilisent un champ de 1 à 3 teslas) et à la durée de l'acquisition (en général une dizaine de minutes). Dans tous les cas, l'épaisseur de coupe (typiquement 3mm) est supérieure à la distance entre deux pixels voisins dans la plan (de l'ordre du millimètre). Les axes sont choisis de telle manière à ce que l'axe x et l'axe y soient dans le plan de la coupe. L'axe z est la verticale si le patient est debout.

Nous supposerons dans un premier temps que la résolution est la même dans toutes les directions (c'est-à-dire que les voxels sont cubiques). Ceci nous permettra d'introduire le calcul de la carte des distances d'une image 2D ou 3D binaire (chap. 8). Nous considérerons ensuite le cas d'une trame anisotrope (chap. 9). Deux approches seront envisagées. La première consiste à se ramener aux cas de voxels cubiques grâce à l'une ou l'autre méthode d'interpolation. La deuxième approche est d'étendre les transformées de distance aux cas d'une trame anisotrope.

CHAPITRE 8

Calcul de la carte de distance d'une image 3D segmentée

Nous présentons ici le calcul de la carte de distance d'une image binaire. Après une description succincte des différentes approches possibles, nous étudierons de manière détaillée le comportement de la transformation du chanfrein. Nous choisirons ensuite la distance de chanfrein la plus adaptée à notre problème. Nous supposons dans tout ce chapitre que les voxels sont cubiques.

8.1 Définitions

Nous rappelons dans cette partie les notions de topologie discrète, de distance discrète et de carte de distance. Pour une explication plus détaillée, nous renvoyons vers la thèse de O. Cuisenaire [6].

8.1.1 Voisinages et connexité

Une *image binaire* I définit deux ensembles de points, l'un correspondant à un objet (noté O), l'autre à l'extérieur de cet objet (noté O') (fig. 8.1).

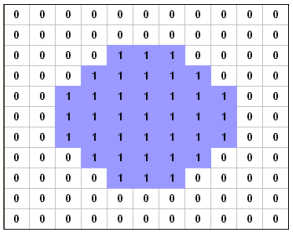
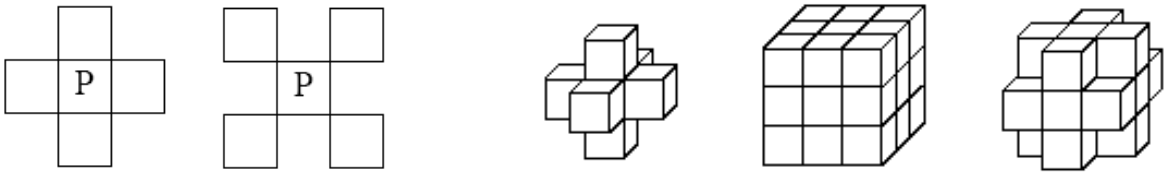


Fig. 8.1 – *Exemple d'une image binaire bidimensionnelle*

Dans le cas d'une image 2D, un pixel p a 4 voisins directs et 4 voisins indirects. On définit le *4-voisinage* et le *8-voisinage* d'une image bidimensionnelle, en considérant respectivement les voisins directs, ou directs et indirects. La notion se généralise au cas tridimensionnel où

l'on définit le *6-voisinage*, le *18-voisinage* et le *26-voisinage*. Ces voisinages sont présentés à la figure 8.2.



(a) 2D : voisins directs (gauche), voisins indirects (droite)

(b) 3D : 6-voisinage (gauche), 18-voisinage (centre), 26-voisinage (droite)

Fig. 8.2 – *Voisinage dans une image 2D et 3D*

Un *chemin* de p_0 à p_k et une suite de points voisins. Le chemin dépend du voisinage utilisé.

8.1.2 Distances

Définition 8.1 (Distance) *Une distance sur $\mathbb{Z}^n$ à valeurs dans $\mathbb{R}$ est une application d : $\mathbb{Z}^n \to \mathbb{R}$ vérifiant $\forall A, B, C \in \mathbb{Z}^n$:*

1. $d(A,B) \geq 0$, *avec* $(A,B) = 0 \Leftrightarrow A = B$ — *définie positive*
2. $d(A,B) = d(B,A)$ — *symétrie*
3. $d(A,B) \leq d(A,C) + d(C,B)$ — *inégalité triangulaire*

Les distances à valeurs dans $\mathbb{R}$, $\mathbb{Q}$ ou $\mathbb{Z}$ sont appelées respectivement distances *réelles*, *rationnelles* ou *discrètes*. Dans ce travail nous utiliserons principalement des distances *discrètes*, plus avantageuses du point de vue stockage et facilité de calcul.

La distance la plus naturellement utilisée est la *distance euclidienne*, définie pour deux points $A(x_a, y_a, z_a)$ et $B(x_b, y_b, z_b)$ de $\mathbb{R}$ par :

$$d_E(A,B) = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2 + (z_b - z_a)} \tag{8.1}$$

Pour obtenir une distance discrète, on prend généralement la distance euclidienne au carré d_E^2. Puisque les pixels de l'image sont localisés dans $\mathbb{Z}^2$, le calcul de d_E^2 ne requiert que des opération entières. Cependant, cette fonction n'est pas une distance au sens de la définition 8.1 car elle ne respecte pas l'inégalité triangulaire[1].

Les premières véritables distances discrètes à avoir été employées en analyse d'images sont, pour deux points $A(x_a, y_a, z_a)$ et $B(x_b, y_b, z_b)$:

- $d_4(A,B) = |x_b - x_a| + |y_b - y_a| + |z_b - z_a|$ — *city block, Manhattan*
- $d_8(A,B) = \max\{|x_b - x_a|, |y_b - y_a|, |z_b - z_a|\}$ — *chessboard, échiquier*

[1] soit A = (0,0), B = (1,0) et C = (2,0), trois points de $\mathbb{Z}^2$; on a $d_E^2(A,B) = 1$, $d_E^2(B,C) = 1$ et $d_E^2(A,C) = 4$, donc $d_E^2(A,C) \not\leq d_E^2(A,B) + d_E^2(B,C)$

8.1.3 Carte de distance

Définition 8.2 (Carte de distance) *Étant donné une image binaire I formée d'un objet O et d'un arrière-plan O' dans un espace métrique, on appelle carte de distance l'image notée D telle que la valeur de chaque pixel p vaut sa distance à l'arrière-plan O', c.-à-d. la plus courte des distance entre p et l'ensemble des pixels de O' :*

$$D(p) = \min \left\{ d(p,q), q \in O' \right\} \tag{8.2}$$

Définition 8.3 (Transformation de distance (DT)) *Une transformation de distance est un algorithme qui permet de calculer la carte des distances au départ de l'image binaire représentant l'objet.*

8.2 Problèmes liés à l'obtention d'une carte de distance euclidienne

Une transformation de distance tel que définie par l'équation 8.2 est une transformation *globale*. Dès lors, l'application directe de la définition nécessiterait de trouver, pour chaque pixel, le minimum de l'ensemble des distances calculées entre ce pixel et la totalité des pixels objets. Bien-entendu, ceci donne lieu à des temps de calculs prohibitifs. C'est pourquoi la plupart des algorithmes de transformations de distances présents dans la littérature tentent de *localiser* les transformations, c.-à-d. qu'ils utilisent des transformations pour lesquelles la valeur d'un pixel dépend uniquement de ses pixels voisins. Le principe à la base de ces algorithmes est le suivant : puisque les valeurs des pixels varient progressivement dans la carte des distances, il doit être possible de déduire la valeur d'un pixel à partir des pixels voisins. Cette constatation est vraie pour les distances d_4 et d_8 mais pas pour la distance euclidienne [15].

L'isotropie des propriétés et la fiabilité des mesures sont importantes en analyse d'images. Cela explique le volume des travaux autour de la distance euclidienne. Cependant, les algorithmes de transformation de distance euclidienne présents dans la littérature sont complexes et relativement coûteux. Ceci est dû à l'absence de définition *locale* de la distance euclidienne.

De nombreux algorithmes permettant d'approcher la distance euclidienne ont été proposés pour pallier à cet inconvénient. Parmi eux, la transformation du chanfrein est sans doute la plus utilisée du fait de son efficacité et de sa simplicité de mise en oeuvre.

8.3 Distance de chanfrein

Notre choix s'est porté sur les transformations de distance du chanfrein, car celles-ci offrent un bon compromis entre l'approximation de la distance euclidienne, la facilité d'utilisation et l'efficacité des schémas algorithmiques [5, 6, 15].

8.3.1 Distance et masque de chanfrein

Le principe à la base de la transformation du chanfrein est qu'il est possible de mesurer des distances *globales* dans une image par propagation d'une information *locale* entre les éléments voisins. L'information véhiculée est tout simplement la distance. Dans le cas d'un plan continu, cette information est suffisante pour calculer la carte de distance euclidienne exacte. Par contre, dans le cas d'une grille discrète, la distance euclidienne est approchée, on parle d'une *distance*

de chanfrein.

L'idée est la suivante. On commence par définir un certain nombre de *déplacements* autorisés et on leur associe à chacun un *coût* (ou *poids*). L'ensemble des couples *déplacement-poids*, appelés *pondérations*, constitue le *masque de chanfrein*. Pour calculer la distance entre deux points p et q on considère les *chemins* possibles de p à q, formés uniquement de déplacements autorisés. On calcule alors le coût de chacun de ces chemins comme la somme des coûts des déplacements élémentaires utilisés. La distance de chanfrein entre p et q est le minimum du coût des chemins possibles.

La distance euclidienne est plus ou moins bien approchée en fonction du choix des pondérations du masque de chanfrein. Dans ce qui suit, nous cherchons à minimiser l'écart entre la distance euclidienne et la distance de chanfrein.

8.3.2 Masques de chanfrein utilisés dans la littérature

Vocabulaire et Notations

Un *masque* de chanfrein M est constitué d'un ensemble fini de *pondérations*. Une *pondération* est un couple $(\vec{v}, \omega)$ formé d'un vecteur $\vec{v}$, appelé *déplacement*, et d'un scalaire ω, appelé *poids*. Dès lors, un masque se note comme suit :

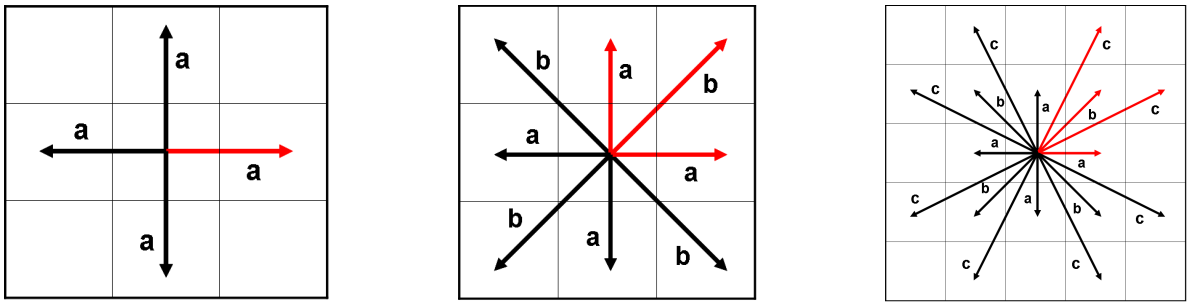
$$M = \{(\vec{v_i}, \omega_i)\}_{1 \leq i \leq m}$$

En pratique on se restreint à l'étude des vecteurs d'un quart de l'espace (en 2D) et on déduit les autres vecteurs par symétrie. Et donc, seuls les vecteurs dont les coordonnées vérifient $x, y \geq 0$ sont notés dans l'expression du masque.

De plus, on peut encore alléger l'écriture des petits masques en notant :

$$\begin{array}{lrcl}
\text{à deux dimensions :} & < a > & = & \{((1,0),a)\} \\
 & < a,b > & = & \{((1,0),a),((1,1),b)\} \\
 & < a,b,c > & = & \{((1,0),a),((1,1),b),((2,1),c)\} \\
\text{à trois dimensions :} & < a,b,c > & = & \{((0,0,1),a),((0,1,1),b),((1,1,1),c)\}
\end{array}$$

Les trois masques bidimensionnels sont présentées à la figure 8.3.



(a) masque $< a >$, taille 3×3

(b) masque $< a, b >$, taille 3×3

(c) masque $< a, b, c >$, taille 5×5

Fig. 8.3 – *Notation des trois petits masques bidimensionnels*

A deux dimensions

L'exemple le plus simple de masque de chanfrein est celui qui définit la distance d_4 (4-voisinage : *city block*). Le masque d_4 est composé de 4 vecteurs pondérés à 1 :

$$\{((1,0),1),((0,1),1),((-1,0),1),((0,-1),1)\}$$

En utilisant les notations définies ci-dessus, le masque d_4 s'écrit encore $< 1 >$. De même, la distance d_8 (8-voisinage : *chessboard*) fait partie des distances que la transformation du chanfrein permet d'approcher. Le masque d_8 est $< 1, 1 >$.

Montanari [41] conseille d'utiliser comme poids les normes euclidiennes des déplacements autorisés :

4-voisinage : $< 1 >$
8-voisinage : $< 1, \sqrt{2} >$
16-voisinage : $< 1, \sqrt{2}, \sqrt{5} >$

Afin d'éviter les calculs en virgules flottantes, Barrow [3] conseille d'utiliser l'approximation entière $< 2, 3 >$ et de diviser l'image résultante par 2.

Les valeurs proposées par Montanari donnent lieu à une surestimation de la distance euclidienne globale. En effet, le chemin de coût minimum joignant deux pixels peut être décomposé, pour le masque $< 1, \sqrt{2} >$, en une ligne droite de déplacements verticaux ou horizontaux et une ligne droite de déplacements diagonaux. Au raison de l'inégalité triangulaire, le chemin direct est nécessairement plus court.

En réalité, les valeurs de poids qui minimisent l'écart entre la distance de chanfrein et la distance euclidienne sont $< 1, 1.351 >$ [15]. A nouveau, pour accélérer les calculs, Borgefors [6] conseille d'utiliser l'approximation entière $< 3, 4 >$ et de diviser la carte de distance résultante par 3.

Une meilleure approximation de la distance euclidienne peut être obtenue en utilisant un voisinage plus grand. En particulier, l'utilisation d'un masque de taille 5×5 (16-voisinage) au lieu du classique 3×3 (8-voisinage) est intéressante. Dans ce cas, Borgefors conseille de prendre les valeurs $< 5, 7, 11 >$ [6].

A trois dimensions

A trois dimensions, les valeurs des poids qui donnent la meilleure approximation de la distance euclidienne sont : $< 1, 1.314, 1.628 >$ [15]. Dans son célèbre article [5], Borgefors conseille d'utiliser le masque $< 3, 4, 5 >$ puis de diviser l'image obtenue par 3.

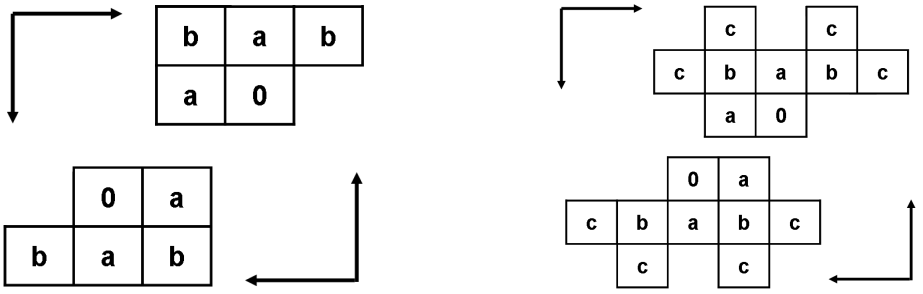
8.3.3 Implémentation de la transformation de distance du chanfrein

Nous présentons ici un algorithme séquentiel pour le calcul de la distance de chanfrein. La simplicité et la rapidité de cet algorithme proposé par Rosenfeld [48] a grandement participé au succès des transformations de distance du chanfrein.

Deux parcours de l'image 3D suffisent pour calculer la transformation de distance du chanfrein. Dans une première passe, *avant* ou *forward*, le masque est glissé sur l'image binaire I de gauche à droite, de haut en bas et d'avant en arrière. Dans un seconde passe, *arrière* ou

backward, le masque parcours l'image de droite à gauche, de bas en haut et d'arrière en avant.

Les masques avant arrière et arrière sont présentés aux figures 8.4 (cas 2D) et 8.5 (cas 3D). Ces masques sont symétriques par rapport à 0.



(a) <a,b> (b) <a,b,c>

Fig. 8.4 – *Masques séquentiels 2D : masques forward (dessus) et backward (dessous)*

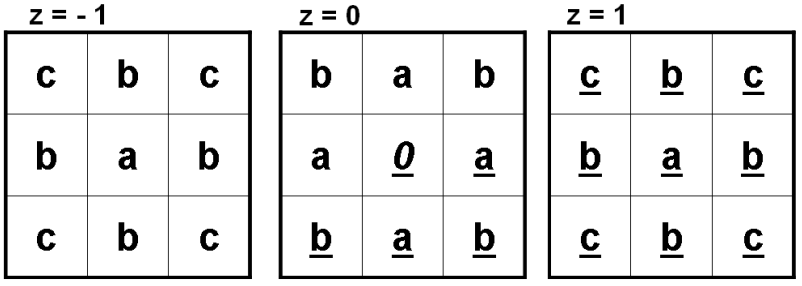


Fig. 8.5 – *Masques de chanfrein 3D : <a,b,c>. Les valeurs non soulignées sont utilisées lors de passe avant, les valeurs soulignées lors de la passe arrière.*

Nous pouvons résumer l'algorithme comme suit :

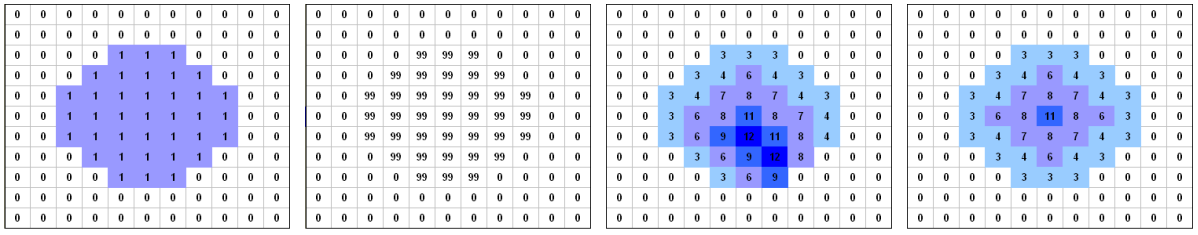
Algorithmne 8.1 – *Transformation de distance de chanfrein en deux passes*

```
# Initialisation :
# Donner une grande valeur aux pixels objets

# Passe avant :
x, y, z = 0, 0, 0
while z < n:
    while y < n:
        while x < n:
            I(x,y,z) = min(I(x+i,y+j,z+k)+w(i,j,k))
            x = x + 1
        y = y + 1
    z = z + 1

# Passe arrière :
x, y, z = n-1, n-1, n-1
```

```
while z >=0:
    while y >=0:
        while x >= 0:
            I(x,y,z) = min(I(x+i,y+j,z+k)+w(i,j,k))
            x = x - 1
        y = y - 1
    z = z - 1
```

Dans un premier temps, l'image est initialisée : on donne une valeur nulle aux pixels de l'arrière-plan et une grande valeur aux pixels objets (en pratique 10^6). Ensuite on effectue les deux passes, avant et arrière. On note les poids des masques $\omega(i,j,k)$ avec, $i = -1, 0, 1$, $j = -1, 0, 1$ et $k = -1, 0, 1$, les coordonnées locales x, y, z autour du pixel courant. Pour chaque position du masque sur l'image, on ajoute les poids des masques aux valeurs (distances) temporaires des voxels correspondants. La nouvelle valeur du pixel courant est le minimum de ces additions.

La figure 8.6 illustre les trois étapes de l'algorithme, dans le cas d'une image bidimensionnelle de taille 11×11 et avec le masque $< 3, 4 >$.



(a) Image binaire initiale (b) Initialisation (c) Première passe (d) Seconde passe

Fig. 8.6 – *Algorithme de transformation du chanfrein. Illustration des différentes étapes dans le cas du masque 2D <3,4>*

8.4 Distance de chanfrein par rapport à la frontière de l'objet

8.4.1 Modifications à apporter à l'algorithme de transformation du chanfrein

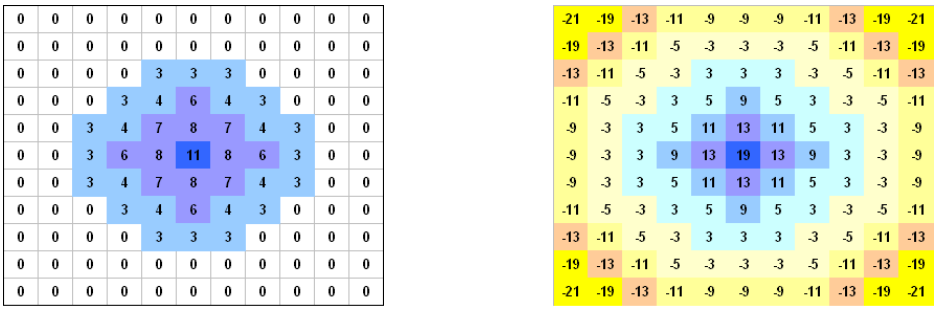
L'algorithme de transformation de distance de chanfrein tel que présenté dans la littérature calcule une carte des distances dans laquelle la valeur de chaque voxel vaut sa distance au voxel le plus proche de l'arrière-plan.

Dans notre cas, nous souhaitons nous ramener au cas des fonctions implicites (chap. 2). Ainsi, les pixels appartenant à l'objet doivent avoir une valeur positive et indiquer la distance par rapport à la frontière de l'objet. Les pixels de l'arrière-plan doivent être négatifs et indiquer la distance par rapport à la frontière de l'objet. Il y a donc deux modifications à effectuer à l'algorithme 8.1 :

1. Nous voulons calculer la carte de distance à la fois de l'intérieur et de l'extérieur de l'objet. Par conséquent, il faudrait appliquer deux fois l'algorithme 8.1. Cependant, dans

[31] Herman propose une solution pour calculer les deux ensembles de distances simultanément, dans le cas d'images bidimensionnelles.

2. Nous ne nous intéressons plus à la distance par rapport à l'objet mais à la distance par rapport au bord de l'objet.8.7 Herman conseille de doubler les poids des masques de sorte que la distance *frontière-pixel* soit entière.



(a) Algorithme de transformation du chanfrein original, masque <3,4>

(b) Algorithme de transformation du chanfrein adapté, masque <6,8>

Fig. 8.7 – *Adaptation de l'algorithme de transformation du chanfrein.*

Nous avons donc implémenté l'algorithme 8.1 en tenant compte des conseils d'Herman. Etant donné qu'Herman ne traite qu'un masque particulier pour les images 2D, nous avons généralisé les principes au cas de masques arbitraires et d'images tridimensionnelles. Les trois sous-sections décrivent les trois étapes de l'algorithme, à savoir, l'initialisation, la première passe et la seconde passe.

8.4.2 Étape 1 : Initialisation

Cas d'une image 2D et d'un masque de taille 3×3, noté $< a, b >$

Dans le cas d'un masque de taille 3×3 (*8-voisinage*), l'image binaire I est initialisée en imposant une grande valeur positive (10^6) aux pixels objets $\in O$ et une grande valeur négative (-10^6) aux pixels appartenant au complémentaire de O, noté O'. Il y'a cependant des exceptions :

- Si $p \in O$ est voisin *direct* avec un pixel $q \in O'$ on lui assigne déjà sa valeur finale, c.à.d. la moitié du premier poids (poids a) du masque. En effet, vu la définition du masque, deux pixels qui sont voisins directs sont séparés d'une distance a. La distance entre p et la frontière vaut donc $a/2$.
- De même, si $q \in O'$ est voisin *direct* avec un pixel $p \in O$, on l'initialise à $-a/2$.

Cas d'une image 2D et d'un masque de taille 5×5, noté $< a, b, c >$

L'initialisation d'une image binaire 2D dans le cas d'un masque de chanfrein de taille 5×5 est similaire mais on ajoute deux conditions :

- Si $p \in O$ est voisin *indirect* avec un pixel $q \in O'$ on lui assigne sa valeur finale, soit $b - a/2$.
- De même, si $q \in O'$ est voisin *indirect* avec un pixel $p \in O$, on l'initialise à $-b + a/2$.

Cas d'une image 3D et d'un masque de taille $3 \times 3 \times 3$, noté $< a, b, c >$

Dans le cas d'image tridimensionnelle et d'un masque $< a, b, c >$, on impose une grande valeur positive aux voxels objet $\in O$ et une grande valeur négative aux voxels $\in O'$. On impose les valeurs finales aux voxels ayant une face où une arrête sur la frontière :

- Si $p \in O$ a une *face* commune avec un voxel $q \in O'$ on lui assigne sa valeur finale, soit $a/2$.
- De même, si $q \in O'$ a une *face* commune avec un voxel $p \in O$ on l'initialise à $-a/2$.
- Si $p \in O$ a une *arrête* commune avec un voxel $q \in O'$ on lui assigne sa valeur finale, soit $b - a/2$.
- De même, si $q \in O'$ une *arrête* commune avec un voxel $p \in O$ on l'initialise à $-b + a/2$.

-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99
-99	-99	-99	-99	-3	-3	-3	-99	-99	-99	-99
-99	-99	-99	-3	3	3	3	-3	-99	-99	-99
-99	-99	-3	3	99	99	99	3	-3	-99	-99
-99	-3	3	99	99	99	99	99	3	-3	-99
-99	-3	3	99	99	99	99	99	3	-3	-99
-99	-3	3	99	99	99	99	99	3	-3	-99
-99	-99	-3	3	99	99	99	3	-3	-99	-99
-99	-99	-99	-3	3	3	3	-3	-99	-99	-99
-99	-99	-99	-99	-3	-3	-3	-99	-99	-99	-99
-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99

(a) masque 2D du type $< a, b >=< 6, 8 >$

-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99
-99	-99	-99	-9	-5	-5	-5	-9	-99	-99	-99
-99	-99	-9	-5	5	5	5	-5	-9	-99	-99
-99	-9	-5	5	9	99	9	5	-5	-9	-99
-99	-5	5	9	99	99	99	9	5	-5	-99
-99	-5	5	99	99	99	99	99	5	-5	-99
-99	-5	5	9	99	99	99	9	5	-5	-99
-99	-9	-5	5	9	99	9	5	-5	-9	-99
-99	-99	-9	-5	5	5	5	-5	-9	-99	-99
-99	-99	-99	-9	-5	-5	-5	-9	-99	-99	-99
-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99

(b) masque 2D du type $< a, b, c >=< 10, 14, 22 >$

-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99
-99	-99	-99	-5	-3	-3	-3	-5	-99	-99	-99
-99	-99	-5	-3	3	3	3	-3	-5	-99	-99
-99	-5	-3	3	5	5	5	3	-3	-5	-99
-99	-3	3	5	99	99	99	5	3	-3	-99
-99	-3	3	5	99	99	99	5	3	-3	-99
-99	-3	3	5	99	99	99	5	3	-3	-99
-99	-5	-3	3	5	5	5	3	-3	-5	-99
-99	-99	-5	-3	3	3	3	-3	-5	-99	-99
-99	-99	-99	-5	-3	-3	-3	-5	-99	-99	-99
-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99

(c) masque 3D du type $< a, b, c >=< 5, 7, 11 >$

Fig. 8.8 – *Adaptation de l'algorithme de transformation du chanfrein : Initialisation. Les "grandes valeurs" sont représentées par* 99.

8.4.3 Étape 2 : Première passe

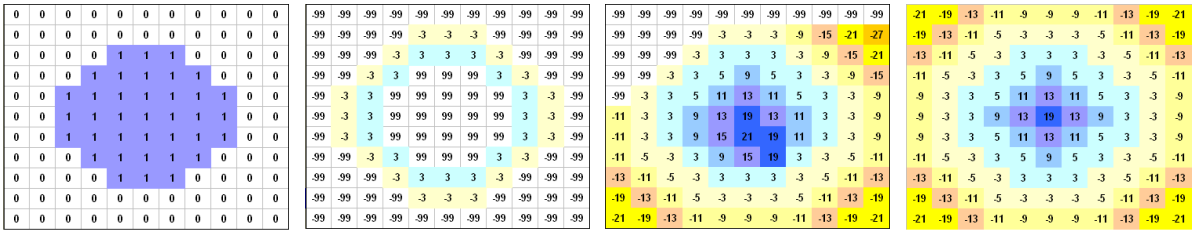
L'image est parcourue de gauche à droite, de haut en bas et d'avant en arrière. Le masque séquetiel *forward* (fig. 8.4 (cas 2D) ou fig. 8.5 (cas 3D)) est placé de manière à ce que le 0 soit situé sur le voxel courant. Si le voxel courant est un voxel qui a été initialisé à $\pm a/2$ ou à $\pm(b - a/2)$, on passe directement au voxel suivant en laissant inchangée la valeur du voxel. Sinon, la valeur du voxel est modifiée comme suit :

Si le voxel courant est positif (c.à.d qu'il appartient à l'objet) : on ajoute les poids du masque aux voxels correspondants. La valeur assignée au voxel courant est la plus petite des sommes calculées.

Si le voxel courant est négatif (c.à.d qu'il n'appartient pas à l'objet) : on soustrait les poids du masque aux voxels correspondants. La valeur assignée au voxel courant est la plus grande des soustractions calculées.

8.4.4 Étape 3 : Seconde passe

La seconde passe est appliquée sur l'image obtenue à l'issue de la première passe, en parcourant l'image de droite à gauche, de bas en haut et d'arrière en avant. La procédure est la même que pour la première passe mais le masque utilisé différent(fig. 8.4 et 8.5).



(a) Image binaire initiale (b) Initialisation (c) Première passe (d) Seconde passe

Fig. 8.9 – *Algorithme de transformation du chanfrein adapté. Illustration des différentes étapes dans le cas du masque 2D <3,4>*

8.5 Analyse des résultats et choix du masque de chanfrein

8.5.1 masques étudiés et critères de comparaison

Dans ce qui suit, nous présenterons les résultats et comparerons quatre masques de chanfrein *bidimensionnels* :

1. le masque $3 \times 3 < 1, 1.351 >$,optimal dans le sens où il minimise l'écart avec la distance euclidienne ;
2. le masque $3 \times 3 < 5, 7 >$, qui est celui utilisé par Herman dans [31] ;
3. le masque $3 \times 3 < 3, 4 >$, conseillé par Borgefors dans [6] ;
4. le masque $5 \times 5 < 5, 7, 11 >$, conseillé par Borgefors dans [5].

Nous analyserons également les deux masques *tridimensionnels* suivants :

1. le masque $3 \times 3 \times 3 < 1, 1.314, 1.628 >$, qui minimise l'écart avec la distance euclidienne.
2. le masque $3 \times 3 \times 3 < 3, 4, 5 >$, approximation entière du précédent et conseillé par Borgefors dans [5].

Nous avons implémenté les cinq masques de chanfrein ci-dessus en adaptant l'algorithme de la transformation de distance du chanfrein comme expliqué à la section 8.4.

La librairie VTK [52] fournit également une classe `vtkImageEuclideanDistance` permettant de calculer la carte de distance 3D euclidienne d'une image binaire.

Nous recherchons le masque de chanfrein qui approxime au mieux la distance euclidienne, en des temps de calculs raisonnables. Au lieu de rechercher à approximer la distance euclidienne sur la totalité de l'image, seul importe, dans notre cas, de bien l'approximer au voisinage de la surface frontière. En effet, dans les calculs ultérieurs d'extraction de la surface, de simplification et d'amélioration du maillage, les seules valeurs de voxels pris en compte sont ceux situés dans le voisinage de la surface.

Dans cette section, nous répondrons aux questions suivantes :
- Quelle est la transformation de chanfrein 2D qui approxime au mieux la distance euclidienne, en un temps de calcul raisonnable ?
- Quelle est la transformation de chanfrein 3D qui approxime au mieux la distance euclidienne, en un temps de calcul raisonnable ?

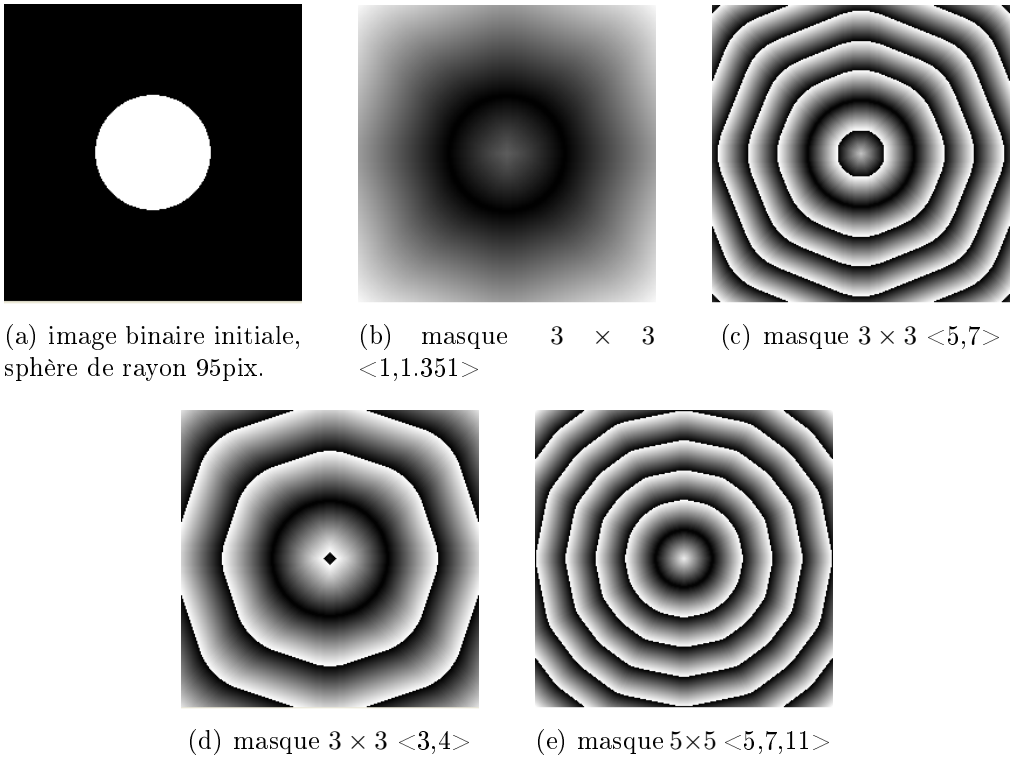
- Que fournit la classe `vtkImageEuclideanDistance` ? Comment se positionne la carte de distance fournie par rapport aux cartes de distance chanfrein implémentées ?
- Dans le cas où la résolution inter-slice est la même que la résolution dans le plan, vaut-il mieux prendre la carte de distance coupe par coupe (calcul 2D) ou utiliser une carte de distance tridimensionnelle ?
- Faut-il normer les cartes de distances de chanfrein obtenues ou peut-on garder des images à valeurs entières ?

Nous appliquerons ensuite les algorithmes d'extraction de la surface, de simplification et d'amélioration du maillage afin d'analyser et de visualiser l'effet des différents méthodes de transformation de distance sur le maillage final.

Pour rappel, dans ce chapitre, la résolution entre les coupes IRM est supposée être la même que la résolution dans le plan de la coupe, de sorte que les voxels sont cubiques. Le cas d'une trame anisotrope sera traité dans le chapitre 9.

8.5.2 Choix du masque de chanfrein bidimensionnel

L'image binaire utilisée pour comparer les différentes transformations de distance de chanfrein bidimensionnelles est une image de taille 256×256 pixels ($x \in 0, 255, y \in 0, 255$) contenant une cercle de diamètre 95 pix. centrée dans l'image.



(a) image binaire initiale, sphère de rayon 95pix.

(b) masque 3×3 <1,1.351>

(c) masque 3×3 <5,7>

(d) masque 3×3 <3,4>

(e) masque 5×5 <5,7,11>

Fig. 8.10 – *Représentation des cartes de distance de chanfrein bidimensionnelles à l'aide d'une palette circulaire de niveaux de gris. Les distances n'ont pas été normées et sont indiquées en valeur absolue. Taille des images :* 256×256 *pixels.*

Les cartes de distance obtenues avec les différents masques bidimensionnels sont illustrées

à la figure 8.10. Elles sont représentées en niveaux gris et renferment donc au maximum 256 teintes de gris. Ceci explique l'apparence de *courbes de niveau*, d'autant plus nombreuses que les poids du masque sont élevés (puisque les cartes de distances représentées n'ont pas été normées). A la figure 8.10, nous remarquons que le cercle est approximé par un octogone pour les masques de taille $3 \times 3 < 3,4 >$ et $< 5,7 >$ et par un hexadécagone pour le masque $5 \times 5 < 5,7,11 >$. Autrement dit, enrichir le masque en tenant compte d'un plus grand voisinage permet d'augmenter le nombre de côtés du polygone et donc de rendre la représentation moins dépendante d'une rotation de l'objet.

La figure 8.11 donne une idée de la variation des valeurs numériques dans les cartes de distance. Nous pouvons y lire le poids accordé à chaque déplacement élémentaire dans l'image et ainsi reconnaître le masque de chanfrein qui a été utilisé.

(a) masque <1,1.351> (b) masque <5,7> (c) masque <3,4> (d) masque <5,7,11>

Fig. 8.11 – *Cartes de distance de chanfrein bidimensionnelles. L'image binaire initiale est représentée à la figure 8.9.*

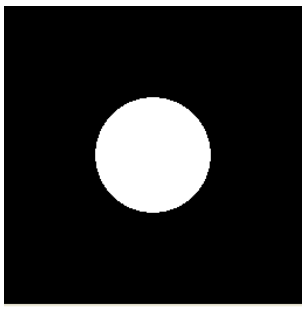
Afin de visualiser l'erreur absolue par rapport à la distance euclidienne en chaque pixel, nous avons pris la différence entre les carte de distances de la figure 8.10 et la carte de distance euclidienne exacte, que nous avons facilement pu calculer dans le cas du cercle. Bien entendu pour pouvoir les comparer à la carte de distance euclidienne, nous avons tout d'abord normé[2] les masques de la figure 8.10 de manière à accorder un poids unitaire au déplacement entre deux voisins directs. Le résultat est visible à la figure 8.12.

Les erreurs absolues maximales et moyennes par rapport à la carte de distance euclidienne sont indiquées dans le tableau 8.1 où nous avons également indiqué les temps de calculs nécessaires à la création des différentes carte de distance de la figure 8.10.

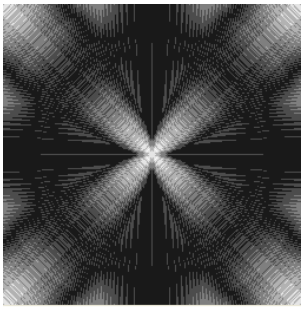
Nous remarquons directement la supériorité du masque $5 \times 5 < 5,7,11 >$ par rapport aux masques 3×3. Celui-ci donne lieu, dans le cas du cercle de rayon 95 pix. et d'une image de taille 256×256, à écart un maximal avec la distance euclidienne de 1.7 pix., situé au centre de l'image. Dans le cas d'un masque 5×5, 16 pixels sont évalués pour l'évaluation de la valeur du pixel courant[3] au lieu de 8 évaluations dans le cas d'un masque 3×3. C'est pourquoi le temps de calcul est supérieur pour les plus grands masques. Pour des grandes images, nous pourrions vouloir limiter le temps de calcul. Le masque que nous conseillons alors est le masque 3×3 <5,7>.

[2] en divisant la carte de distance par le premier poids du masque

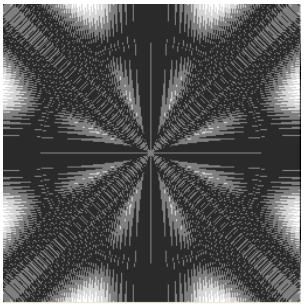
[3] 16 évalutions par pixel et par passe, étant donné que l'algorithme effectue trois passes sur l'image, cela donne 48 évaluations par pixel.



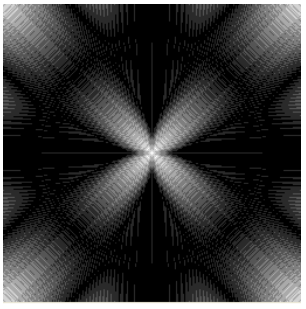
(a) image binaire 3D initiale



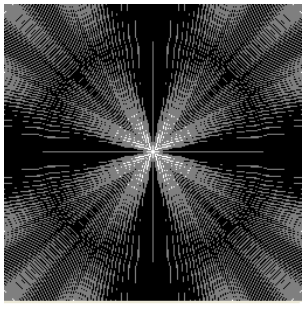
(b) masque 3×3 <1,1.351>, $\epsilon_{max} = 4.1$pix.



(c) masque 3×3 <5,7>, $\epsilon_{max} = 3.0$pix.



(d) masque 3×3 <3,4>, $\epsilon_{max} = 5.3$pix.



(e) masque 5×5 <5,7,11>, $\epsilon_{max} = 1.7$pix.

Fig. 8.12 – *Erreurs absolues entre les cartes de distance de chanfrein 2D étudiées et la carte de distance euclidienne 2D exacte (dans chaque image, la valeur de pixel la plus élevée (en blanc) correspond à l'erreur maximale indiquée sous la figure).*

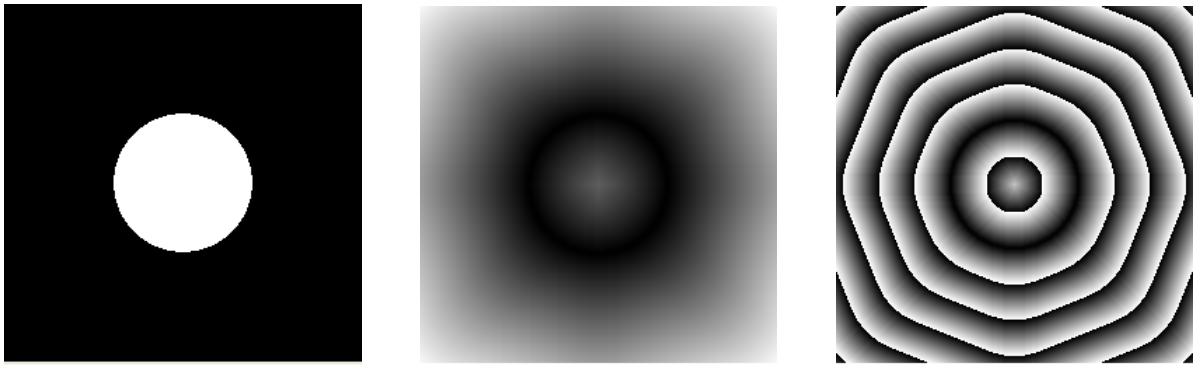
	temps de calcul [s]		erreur absolue [pix]	
	sans normaliser	en normalisant	maximale	moyenne
masque 3×3 : <1,1.351>	3,552	3,696	4,108	0,843
masque 3×3 : <5,7>	3,514	3,501	2,986	0,507
masque 3×3 : <3,4>	3,516	3,645	5,264	1,058
masque 5×5 : <5,7,11>	4,115	4,213	1,700	0,447

Tab. 8.1 – *Temps de calcul et écart par rapport à la distance euclidienne pour les 4 masques bidimensionnels testés sur l'image 2D de taille* 256×256 *de la figure 8.10(a).*

8.5.3 Choix du masque de chanfrein tridimensionnel

L'image binaire utilisée pour comparer les différentes transformations de distance de chanfrein tridimensionnelles est une image 3D de taille $256 \times 256 \times 256$ pixels ($x \in 0,255, y \in 0,255, z \in 0,255$) contenant une sphère de rayon 100 centrée dans l'image.

Les cartes de distance obtenues avec les différents masques 3D sont illustrées à la figure 8.13. Nous avons choisi de représenter la quarantième coupe car celle-ci correspond à un cercle de diamètre 95 pix., tout comme dans le paragraphe précédent.



(a) image binaire initiale (b) masque $< 1, 1.314, 1.628 >$ (c) masque <3,4,5>

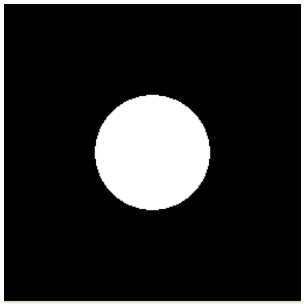
Fig. 8.13 – *Représentation des cartes de distance de chanfrein tridimensionnelles à l'aide d'une palette circulaire de niveaux de gris. Les distances n'ont pas été normées et sont indiquées en valeur absolue. La coupe représentée ici est la coupe 40 du volume d'image de taille* $256 \times 256 \times 256$ *contenant une sphère de rayon 100 en son centre.*

Afin de visualiser l'écart par rapport à la distance euclidienne, nous avons soustrait la carte de distance euclidienne tridimensionnelle exacte aux carte de distance 3D obtenues. A la figure 8.14, nous avons représenté deux coupes du volume d'image, la coupe 40 et la coupe passant par le centre de la sphère.

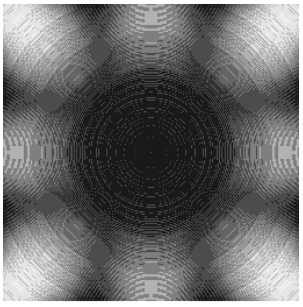
	temps de calcul [s]		erreur absolue [pix]	
	sans normaliser	en normalisant	maximale	moyenne
<1,1.314,1.628>	1911	2074	5	1
<3,4,5>	1810	1959	3	0

Tab. 8.2 – *Temps de calcul et écart par rapport à la distance euclidienne pour les 3 masques tridimensionnels testés sur l'image 3D de taille* $256 \times 256 \times 256$ *de la figure 8.13(a).*

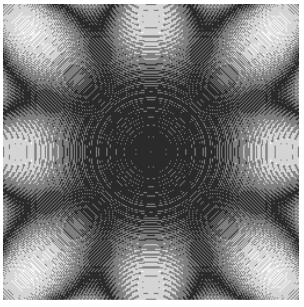
Les résultats obtenus avec les deux masques de chanfrein tridimensionnels $< 1, 1.314, 1.628 >$ et $< 3, 4, 5 >$ sont équivalents. Notre choix se porte donc sur le masque $< 3, 4, 5 >$, qui a l'avantage d'être entier.



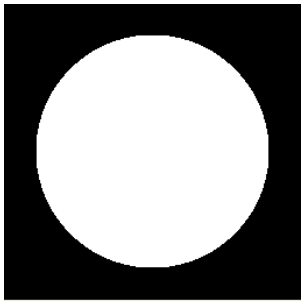
(a) image binaire 3D initiale, coupe 40



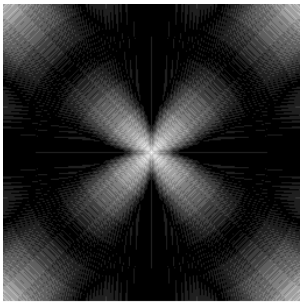
(b) masque $3 \times 3 \times 3$ <1,1.314,1.628>, coupe 40



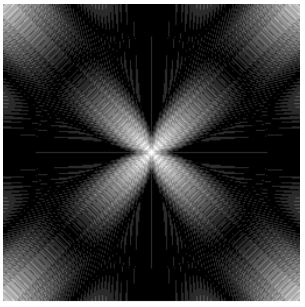
(c) masque $3 \times 3 \times 3$ <3,4,5>, coupe 40



(d) image binaire 3D initiale, coupe 127



(e) masque $3 \times 3 \times 3$ <1,1.314,1.628>, coupe 127



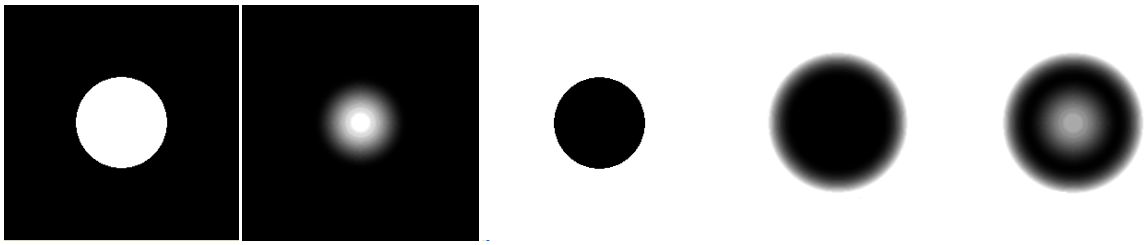
(f) masque $3 \times 3 \times 3$ <3,4,5>, coupe 127

Fig. 8.14 – *Erreurs absolues entre les transformations de chanfrein 3D étudiés et la carte de distance euclidienne exacte également calculée en 3D. Représentation pour deux coupes de dimension* 126×126 *: la coupe du milieu, passant par le centre de la sphère (coupe 127) et une coupe arbitraire (coupe 40).*

8.5.4 Carte de distance obtenue à l'aide de VTK

La classe `vtkImageEuclideanDistance` de VTK permet de calculer la carte de distance 3D euclidienne d'une image binaire. Le calcul se fait sur base de l'algorithme de Saito [51]. La carte des distance fournie contient les distances euclidiennes au carré. La documentation de VTK mentionne que l'algorithme de Saito fonctionne bien pour les petites images, mais qu'une autre méthode (non encore implémentée) devrait être employée pour les images de plus grandes tailles.

Afin d'obtenir la carte des distances de l'intérieur et de l'extérieur de l'objet, nous avons appliqué la fonction deux fois : sur l'image binaire initiale et sur son complémentaire. Puis, nous avons additionné les cartes des distances (fig. 8.15). Par contre, la valeur d'un voxel de l'objet (resp. du fond) est sa distance par rapport au plus proche voxel de l'arrière-plan (resp. de l'objet) et non sa distance par rapport à la surface frontière (il y'a une différence d'un demi voxel). Ainsi, la carte de distance obtenue à l'aide de VTK contiendra toujours des erreurs.



(a) image initiale (b) carte de distance de (a) (c) négatif (d) carte de distance de (b) (e) résultat final

Fig. 8.15 – *Calcul d'une carte de distance à l'aide de VTK (`vtkImageEuclideanDistance`). Les valeurs supérieures à 255 ont été fixées à 255 et les valeurs inférieures à 0 ont été limitées à 0. Les distances n'ont pas été normées et représentent donc le carré de la distance euclidienne. La coupe représentée ici est la coupe 40 du volume d'image de taille* $256 \times 256 \times 256$ *contenant une sphère de rayon 100 en son centre.*

8.5.5 Comparaisons des transformation de chanfrein 2D et 3D

Lorsque l'image tridimensionnelle segmentée est telle que les voxels sont cubiques, il est préférable d'utiliser une transformation de distance tridimensionnelle au lieu de calculer la carte de distance 2D coupe par coupe. En effet, une série de cartes de distance bidimensionnelles ne permet pas d'approximer la carte de distance euclidienne exacte tridimensionnelle. Dans le graphe de la figure 8.16, nous voyons que l'écart maximal par rapport à la distance euclidienne augmente, dans le cas bidimensionnel, de coupe en coupe lorsqu'on s'éloigne de la coupe passant par le centre de la sphère. Ceci se vérifie à la figure 8.17, où l'on a pris la différence des cartes de distance 2D et 3D avec la carte de distance euclidienne tridimensionnelle. Nous voyons que l'erreur est grande dans la quarantième coupe de la carte de distance bidimensionnelle. Cet effet est encore plus marqué si la surface n'est pas sphérique mais possède des parties non convexes par exemple.

Ainsi, malgré la légère augmentation en temps de calcul, nous utiliserons toujours une transformation de distance tridimensionnelle pour le calcul de la carte de distance d'une image à voxels cubiques.

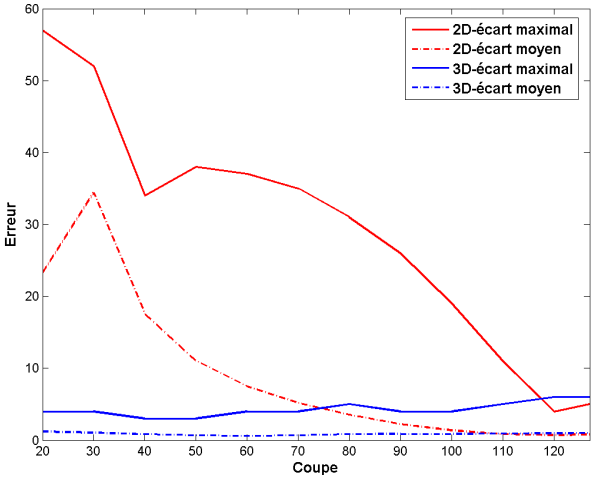
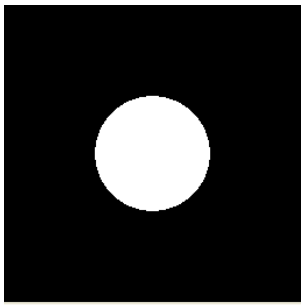
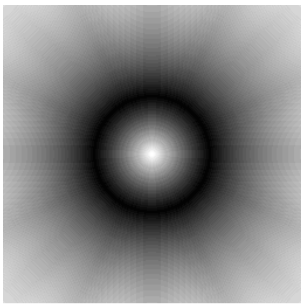


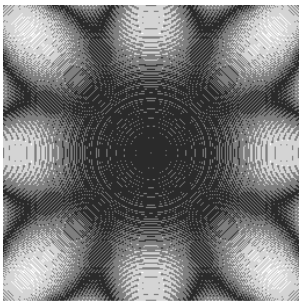
Fig. 8.16 – *Comparaison entre les cartes de distance 2D et 3D : Écart par rapport à la carte de distance euclidienne tridimensionnelle. L'image binaire initiale est une image de taille* 256×256 *contenant une sphère de rayon 100.*



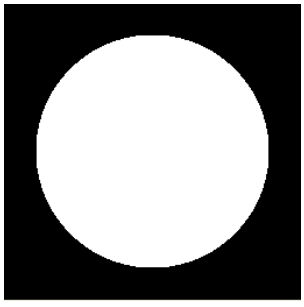
(a) image binaire 3D initiale, coupe 40



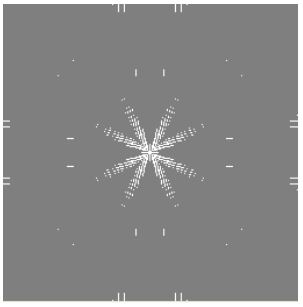
(b) masque 5×5 <5,7,11>, coupe 40, $\epsilon_{max} = 36$pix.



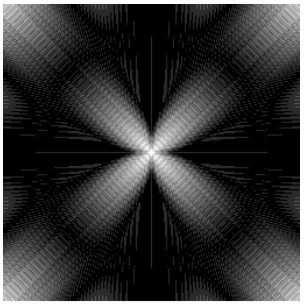
(c) masque $3 \times 3 \times 3$ <3,4,5>, coupe 40, $\epsilon_{max} = 3$pix.



(d) image binaire 3D initiale, coupe 127



(e) masque 5×5 <5,7,11>, coupe 127, $\epsilon_{max} = 1$pix.



(f) masque $3 \times 3 \times 3$ <3,4,5>, coupe 127, $\epsilon_{max} = 6$pix.

Fig. 8.17 – *Comparaison de la transformation de chanfrein 2D* 5×5 *<5,7,11> et la transformation de chanfrein 3D* $3 \times 3 \times 3$ *<3,4,5>. Pour la coupe 127 passant par le centre de la sphère, l'écart par rapport à la carte de distance euclidienne est similaire. Par contre, dès que l'on prend des coupes éloignés du centre de la sphère (coupe 40) : la transformation de distance bidimensionnelle approxime mal la distance euclidienne.*

8.5.6 Faut-il normer les cartes de distances obtenues ?

Il n'est pas nécessaire de normer les cartes de distances obtenues pour la suite du calcul puisque cela n'influence pas la définition de la fonction implicite. Ainsi, nous garderons les cartes de distance à valeurs dans $\mathbb{Z}$ afin de minimiser le coût des calculs ultérieurs.

8.6 Application de l'algorithme d'extraction de la surface

Nous allons maintenant utiliser les algorithmes d'extraction de la surface, de simplification et d'amélioration du maillage développés dans la première partie du travail pour mailler surfaciquement les objets contenus dans des images tridimensionnelles segmentées. Pour l'instant, les voxels sont toujours supposés cubiques.

Dans la carte de distance, la valeur de chaque voxel correspond à sa distance à la surface frontière de l'objet d'intérêt. Par convention, les voxels situés à l'intérieur de la surface ont une valeur positive tandis que les voxels extérieurs sont négatifs. Tout comme dans le cas des fonctions implicites, la surface à extraire est l'isosurface de niveau 0. Il suffit donc pour pouvoir appliquer les algorithmes développés dans la première partie de ce travail de remplacer la fonction implicite par la carte de distance tridimensionnelle.

Pour pouvoir comparer les maillages de surface obtenus au départ d'une fonction implicite et ceux obtenus à partir d'une image 3D segmentée composée de voxels cubiques, nous avons créé les images 3D binaires correspondantes aux géométries présentées dans la première partie du travail (ch.2). Nous avons ensuite calculé leur carte de distance à l'aide de la transformation de distance de chanfrein $3 \times 3 \times 3 < 3, 4, 5 >$. Ces cartes de distance ont ensuite été appliquées en entrée de l'algorithme d'extraction de la surface du Marching Tetrahedra, tout comme l'était la fonction implicite. Nous avons alors successivement appliqué :

1. L'algorithme de simplification du maillage : le Vertex Clustering.
2. L'algorithme permettant d'améliorer la topologie en s'assurant que chaque noeud ait au minimum 5 triangles.
3. L'algorithme permettant d'améliorer la qualité de la triangulation par des déplacements de noeuds.

Les résultats sont présentés aux figures 8.18 à 8.25. Ces maillages sont à comparer aux maillages des surfaces implicites correspondantes (figures 6.5 à 6.12). Le tableau 8.3 reprend les résultats numériques obtenus.

L'ombrage des géométries fait apparaître des *marches d'escaliers* sur celles-ci, dues au fait que le maillage est généré non plus à partir d'une fonction implicite mais à partir d'une image 3D discrète.

Remarquons également la symétrie 3D de ces *marches d'escaliers* sur la sphère. Cette symétrie est présente grâce au fait que nous avons utilisé un masque de chanfrein tridimensionnel pour calculer la carte de distance de l'image binaire initiale.

La taille de la grille régulière de l'algorithme du Marching Tetrahedra est directement liée à l'image : chaque voxel de l'image 3D correspond à un noeud de la grille. Dans l'algorithme

du Marching Tetrahedra, seule la valeur aux noeuds de la grille (correspondant aux valeurs des voxels de la carte de distance) est nécessaire. Pour les trois algorithmes suivants, une interpolation tri-linéaire est réalisée pour pouvoir se ramener au cas ou d'une fonction connue en tout point du domaine. Par défaut et pour les surfaces présentées aux figures 6.5 à 6.12, la grille utilisée dans l'algorithme du Vertex Clustering est identique mais décalée d'un demi-voxel pour les raisons expliquées dans le chapitre 4. Le nombre de mailles générées dépend de la finesse de ces deux grilles. Il existe donc deux techniques pour réduire le nombres de mailles générées :

1. diminuer la taille de l'image 3D initiale ;
2. diminuer le nombre de subdivisions de la grille du Vertex Clustering, en gardant tous les voxels de l'image 3D initiale.

Image 3D									**moyenne**
Image initiale									
nombre de voxels	125000	216000	343000	262144	287496	216000	216000	64000	216205
Marching Tet.									
nombre de noeuds	22394	19678	40878	40120	46620	10788	18302	21798	27572
nombre de triangles	44784	39352	81752	80236	93236	21576	36596	43632	55146
Maillage final									
nombre de noeuds	5675	4969	10231	10020	11899	2705	4681	5515	6962
nombre de triangles	11346	9934	20458	20036	23794	5410	9354	11068	13925
Qualité selon Frey									
Pire triangle	0,46	0,31	0,34	0,32	0,26	0,27	0,20	0,07	0
Meilleur triangle	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	1
Moyenne	0,89	0,76	0,72	0,72	0,74	0,80	0,77	0,79	1
Temps de calcul [s]									
Carte de distance	7	12	18	13	15	12	12	6	12
Marching Tetrahedra	19	27	46	37	41	21	24	14	29
Vertex Clustering	7	7	13	14	16	3	6	9	9
Topologie	7	5	13	8	19	2	4	7	8
Optimisation	72	68	331	159	309	84	147	193	170
Total	111	119	421	230	400	122	193	230	228
Vitesse [triangles/s]	102	83	49	87	60	45	49	48	61

Tab. 8.3 – *Génération d'un maillage surfacique au départ d'une image 3D à voxels cubiques (distance de chanfein 3D : $< 3, 4, 5 >$: Résultats numériques obtenus pour les 8 géométries tests, dont les maillages sont présentés aux figures 8.18 à 8.25.*

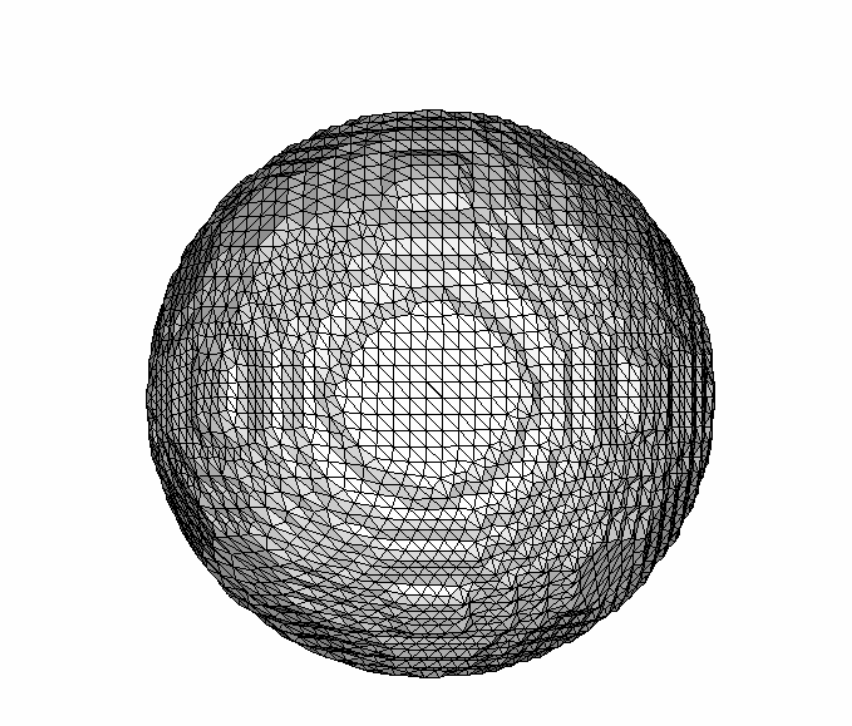


Fig. 8.18 – *Génération d'un maillage surfacique au départ d'une image 3D à voxels cubiques contenant* `la sphère`.

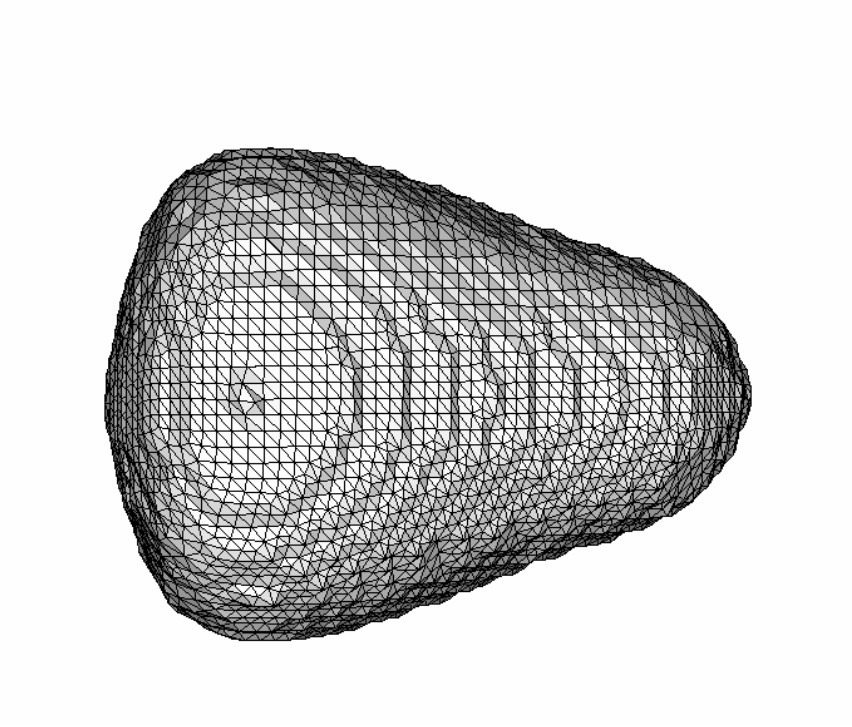


Fig. 8.19 – *Génération d'un maillage surfacique au départ d'une image 3D à voxels cubiques contenant* `le cône arrondi`.

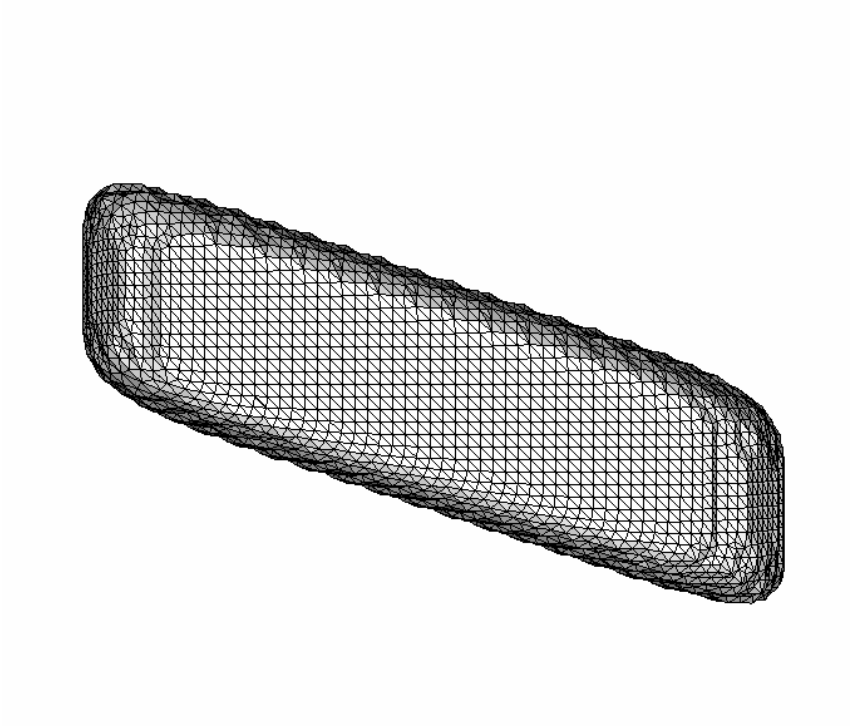


Fig. 8.20 – *Génération d'un maillage surfacique au départ d'une image 3D à voxels cubiques contenant le parallélépipède.*

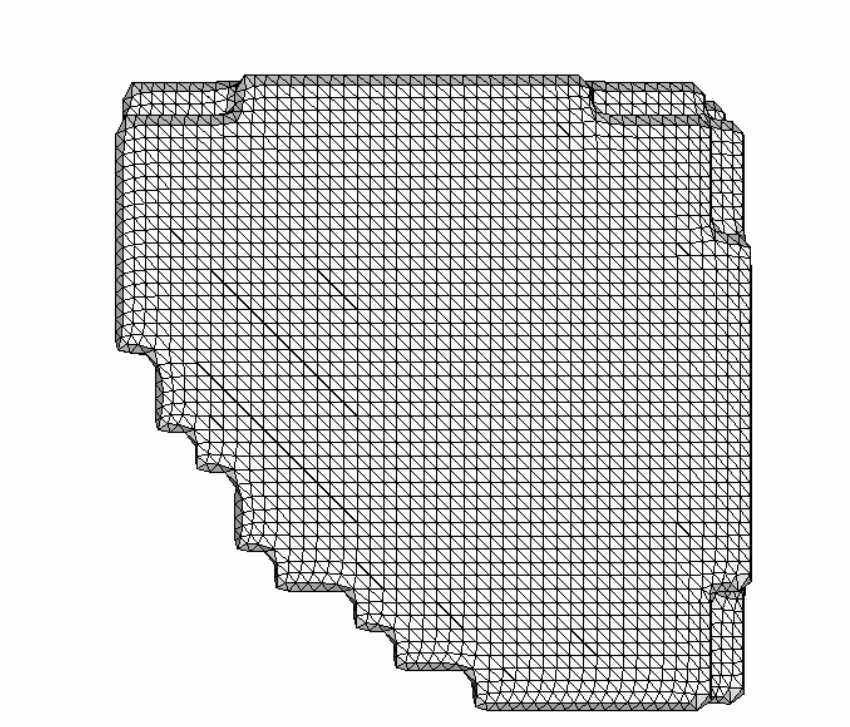


Fig. 8.21 – *Génération d'un maillage surfacique au départ d'une image 3D à voxels cubiques contenant le maximum entre un cube et une sphère.*

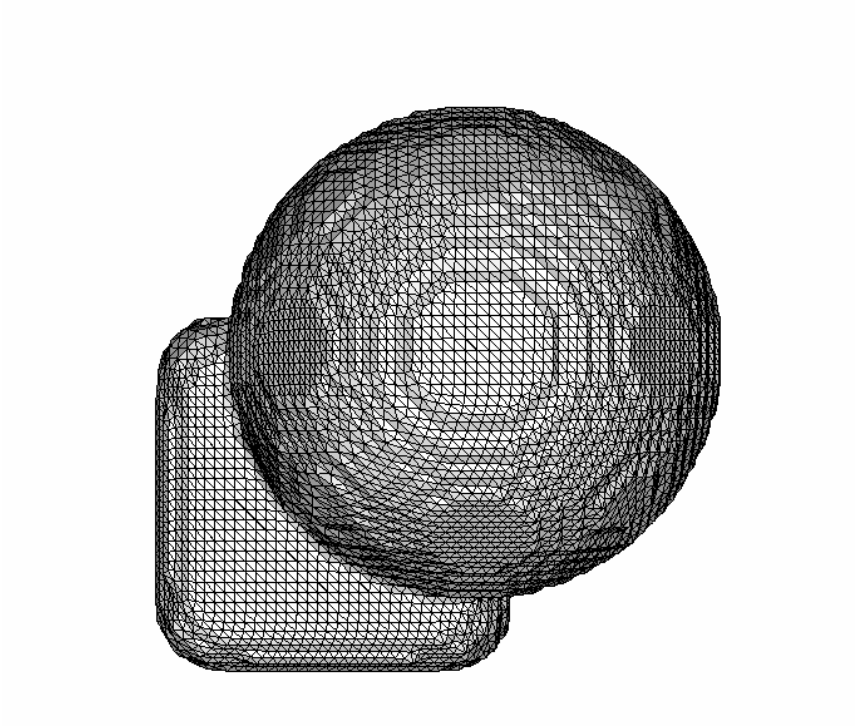


Fig. 8.22 – *Génération d'un maillage surfacique au départ d'une image 3D à voxels cubiques contenant le minimum entre un cube et une sphère.*

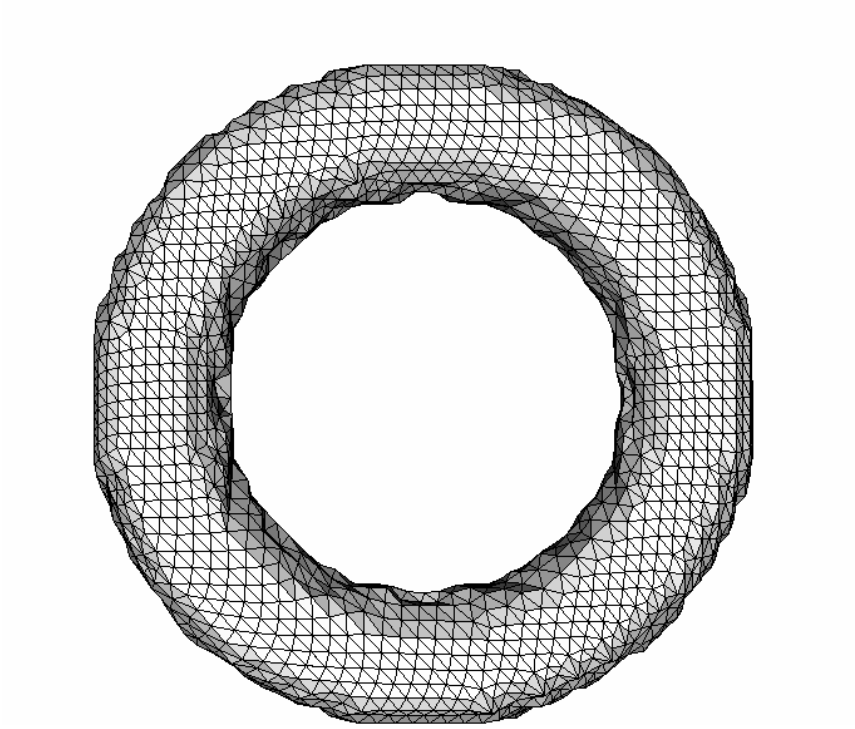


Fig. 8.23 – *Génération d'un maillage surfacique au départ d'une image 3D à voxels cubiques contenant le tore.*

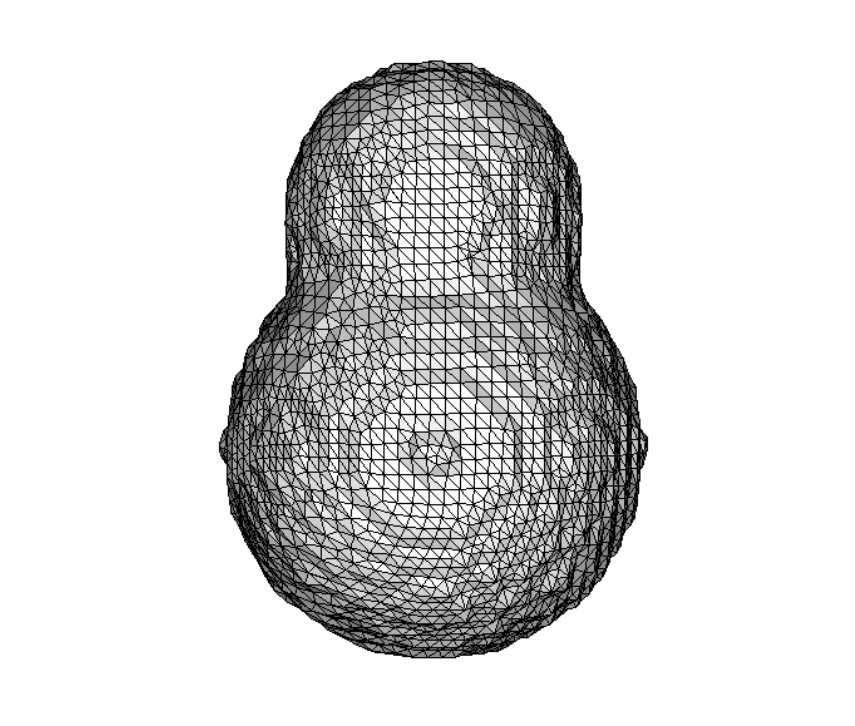


Fig. 8.24 – *Génération d'un maillage surfacique au départ d'une image 3D à voxels cubiques contenant* `la multiplication de deux sphères`.

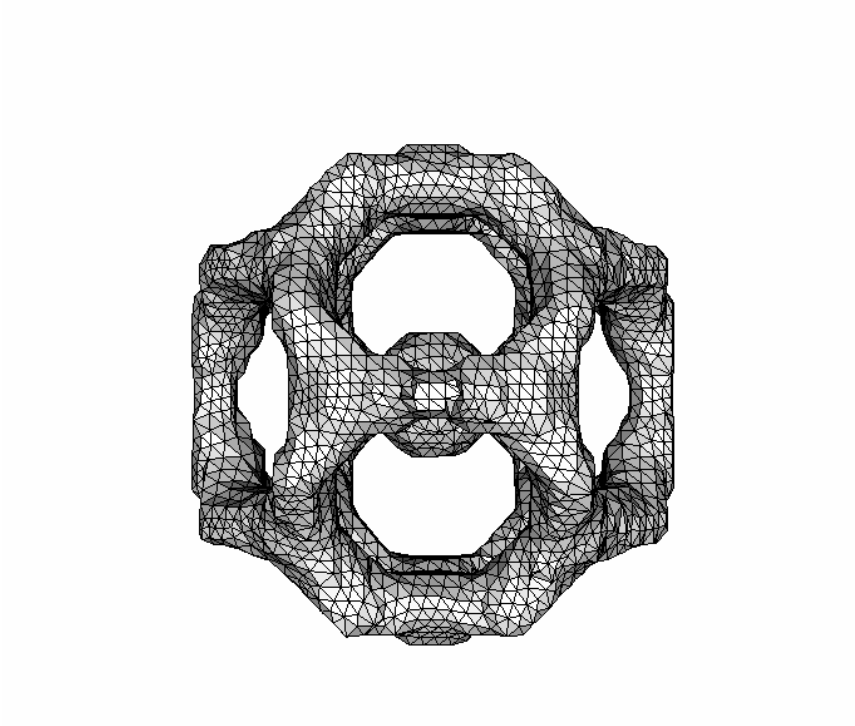


Fig. 8.25 – *Génération d'un maillage surfacique au départ d'une image 3D à voxels cubiques contenant* `la géométrie du nombre d'or`.

CHAPITRE 9

Trame anisotrope : carte de distance et interpolation

Dans ce chapitre, nous traitons le cas plus réaliste d'une image médicale dont la résolution est inférieure dans la direction perpendiculaire aux tranches que dans le plan de la tranche.

9.1 Deux approches possibles

En imagerie médicale, les images tridimensionnelles générées ont généralement une résolution inférieure dans la direction perpendiculaire aux coupes. Au lieu d'être cubiques comme nous l'avions supposé dans le chapitre précédent, les voxels ont pratiquement toujours un côté plus long que les deux autres.

Nous envisagerons deux méthodes pour résoudre ce problème. La première solution est d'étendre la théorie des transformations de distance de chanfrein tridimensionnelles au cas d'une trame anisotrope, c'est la méthode proposée par Sintorn et Borgefors dans [58]. L'avantage de la méthode proposée par Sintorn et Borgefors est qu'elle permet de traiter des voxels allongés sans nécessiter d'interpolation. Ceci permet d'éviter de générer une carte de distance plus volumineuse que l'image binaire initiale. Par conséquent, le coût des opérations ultérieures telles l'extraction de la surface et la simplification du maillage est plus faible que si l'on avait effectué une interpolation. Cependant, nous verrons que les résultats se dégradent lorsque l'épaisseur de coupe augmente. De plus, les mailles générées seront allongées selon z. Par conséquent, une interpolation entre coupes est indispensable lorsque les voxels deviennent trop allongés. Des méthodes d'interpolation entre coupes seront présentés à la section 9.3.

En conclusion, pour des voxels peu allongés nous utiliserons la méthode de Sintorn et Borgefors et pour des voxels plus allongés nous utiliserons une méthode d'interpolation entre coupes pour se ramener aux cas de voxels cubiques. L'analyse des résultats à la section 9.4 permettra de sélectionner la méthode la plus appropriée en fonction de l'épaisseur de coupe.

9.2 Extension des transformations de distance de chanfrein au cas d'une trame anisotrope

9.2.1 Explication de la méthode

La méthode de Sintorn et Borgefors permet de calculer la carte de distance tridimensionnelle d'une image binaire composée de voxels allongés [58]. Les voxels sont supposés avoir une hauteur et une largeur égales à 1 (directions x et y) et une longueur de Λ (direction z).

Dans le cas de voxels allongés et d'une transformation de distance de chanfrein du type $3 \times 3 \times 3$, il existe 5 types de déplacements (au lieu de 3 pour les voxels cubiques). Ceux-ci sont représentés à la figure 9.1.

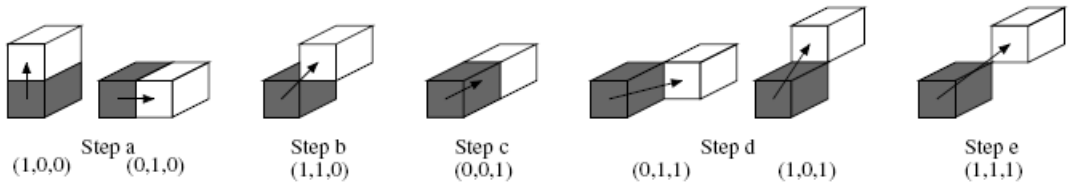


Fig. 9.1 – *Déplacements élémentaires dans le cas d'une transformation de distance de chanfrein tridimensionnelle* $3 \times 3 \times 3$. *La longueur d'un voxel vaut* Λ *tandis que sa hauteur et sa largeur valent* 1 *[58].*

Le masque de chanfrein possède donc 5 poids différents, notés de a à e. Celui-ci est schématisé à la figure 9.2.

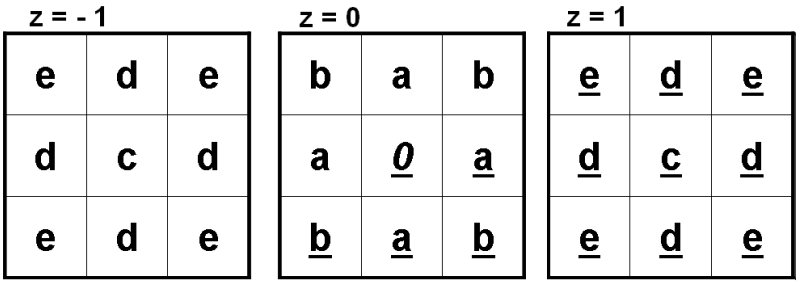


Fig. 9.2 – *Masques de chanfrein 3D utilisé dans le cas de voxels allongés :* <*a,b,c,d,e*>. *Les valeurs non soulignées sont utilisées lors de passe avant, les valeurs soulignées lors de la passe arrière.*

Dans [58], Sitorn et Borgefors calculent, pour différentes valeurs de Λ, les poids a, b, c, d, e qui minimisent l'erreur maximale entre la carte des distances de chanfrein générée et la carte de distance euclidienne dans une image 3D de taille $\Lambda M \times \Lambda M \times M$ (tab. 9.1).

On trouve des approximations entières des poids optimaux en les multipliant par un facteur d'échelle (qui peut être non entier) et en les arrondissant à l'entier le plus proche. Les approximations entières considérées dans le cadre de ce travail sont indiquées dans le tableau 9.1. L'écart maximal par rapport à la distance euclidienne est noté *maxdiff*.

	facteur d'échelle	a	b	c	d	e	maxdiff
$\Lambda = 1,5$							
valeurs optimales		0,9085	1,3227	1,3628	1,6656	1,9243	0,1370
approx. entière	8,8470	8	12	12	15	17	0,144
$\Lambda = 2,58$							
valeurs optimales		0,8750	1,2892	2,2575	2,4445	2,6197	0,3230
approx. entière	6,8820	6	9	16	17	18	0,331
$\Lambda = 5$							
valeurs optimales		0,8322	1,2464	4,1609	4,2599	4,3571	0,8390
approx. entière	9,6250	8	12	41	41	42	0,844

Tab. 9.1 – *Valeurs optimales des poids pour différentes longueurs de voxels Λ [58].*

L'article de Sintorn et Borgefors fournit également un graphe donnant l'augmentation de l'erreur *maxdiff* avec l'épaisseur de coupe Λ (figure 9.3). Nous voyons que l'erreur maximale entre la carte de distance de chanfrein calculée et la carte distance euclidienne augmente rapidement avec Λ. Ceci signifie que la carte des distances est de plus en plus influencée par une rotation de l'objet dans l'image. Il existe donc un certaine épaisseur de coupe, notée Λ_c, à partir de laquelle l'erreur devient inacceptable. Pour les Λ supérieurs, une interpolation entre coupes est indispensable.

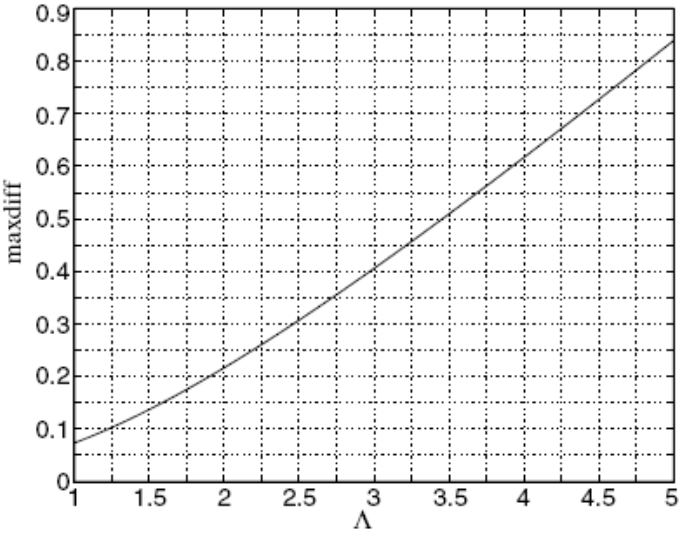


Fig. 9.3 – *Augmentation de l'erreur maximale entre la carte de distance de chanfrein et la distance euclidienne* `maxdiff` *avec l'épaisseur de coupe Λ (quelle que soit l'image) [58].*

9.3 Interpolation entre coupes

9.3.1 Etat de l'art

Dans les images médicales la distance qui sépare les différentes coupes est généralement plus grande que la dimension du pixel dans le plan. L'interpolation entre coupes permet alors de calculer des coupes intermédiaires (c.à.d. des nouvelles coupes) afin obtenir des voxels cubiques

(et donc obtenir des mailles de meilleures qualités que pour des voxels allongés).

L'approche classique consiste à partir de l'image non segmentée et à interpoler les valeurs grisées. Cette méthode est déconseillée [31] pour les deux raisons suivantes :

1. La segmentation d'images médicales n'est la plupart du temps que semi-automatique : le médecin doit indiquer la zone d'intérêt sur l'écran. Ainsi, l'interpolation des images en niveaux de gris augmenterait la charge de travail de l'utilisateur.
2. L'interpolation des images en niveaux de gris donne de mauvais résultats lorsque l'objet d'intérêt est fortement déplacé d'une image à l'autre [31]. Ceci est illustré à la figure 9.4. Les coupes intermédiaires ont été calculées par interpolation linéaire des valeurs et l'objet est obtenue par seuillage à 136. A la figure 9.4, nous observons un phénomène de marches d'escaliers dans la frontière de l'objet.

199	**189**	**145**	**83**	**33**	**11**	**11**	**14**	**14**	**12**
192	182	149	99	61	46	45	39	25	12
185	176	153	115	89	82	80	64	36	13
178	169	156	132	118	117	114	89	47	13
171	163	160	148	146	153	149	114	58	14
164	**156**	**164**	**164**	**174**	**188**	**183**	**139**	**69**	**14**

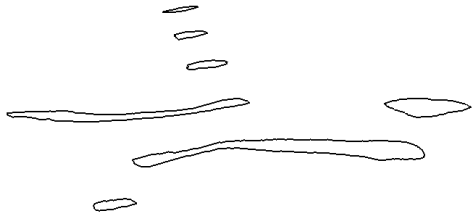
Fig. 9.4 – *Illustration de l'artefact de l'escalier. Les valeurs grisées représentent le tissus (valeur < 136) tandis que l'os est en blanc (valeur > 136). Nous voyons que l'interpolation linéaire des valeurs des coupes extérieures pour obtenir les 4 coupes intermédiaires donne lieu à une changement brusque de la position de la frontière tissus-os [58].*

Pour pallier à ces inconvénients, Raya et Udupa proposent une méthode d'interpolation fondée sur la forme (*shape-based interpolation*) [46]. Raya et Udupa partent de l'image tridimensionnelle segmentée (binaire) et estiment la position de l'objet (des objets) dans les coupes intermédiaires (non existantes). Dans un premier temps, ils calculent la carte des distances 2D de chacune des coupes. De cette manière, ils obtiennent un ensemble de coupes en niveaux de gris où la valeur de chaque pixel représente sa distance à la frontière de l'objet dans la coupe[1]. Les nouvelles coupes sont alors obtenues par interpolation des valeurs grisées (distances). Raya et Udupa conseillent d'utiliser une interpolation par spline cubique en démontrant que celle-ci donne de meilleurs résultats que l'interpolation linéaire.

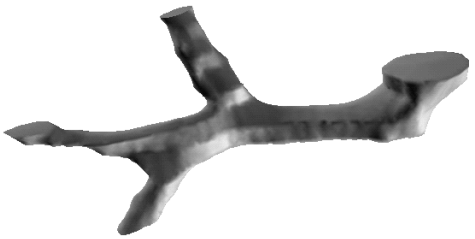
La méthode d'interpolation fondée sur la forme proposée par Raya et Udupa et améliorée par Herman dans [31] est simple et rapide, mais donne de mauvais résultats lorsque les coupes successives sont fort différentes. En particulier, si l'objet subit une forte translation entre deux coupes successives, de sorte qu'il n'y plus de partie commune, une interpolation dans le sens perpendiculaire aux coupes ne permettra pas de lier les deux objets (cfr. fig. 9.5(c) et section 9.4). Des méthodes ont été proposées pour résoudre le problème, comme par exemple une in-

[1]Le principe est le même qu'à la section 8.4, sauf que la méthode présentée à la section 8.4 provient de Herman [31] et est une amélioration de celle proposée par Raya et Udupa [46].

terpolation le long de la ligne liant les centres de gravité dans chaque coupe [32].

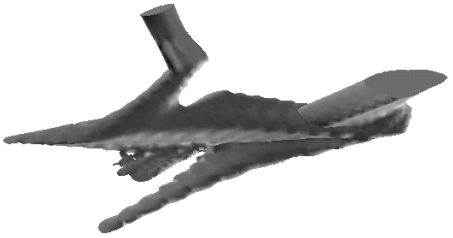


(a) Coupes intiales



(b) Surface réelle

(c) Shape-based interpolation [46] (avec une interpolation linéaire)

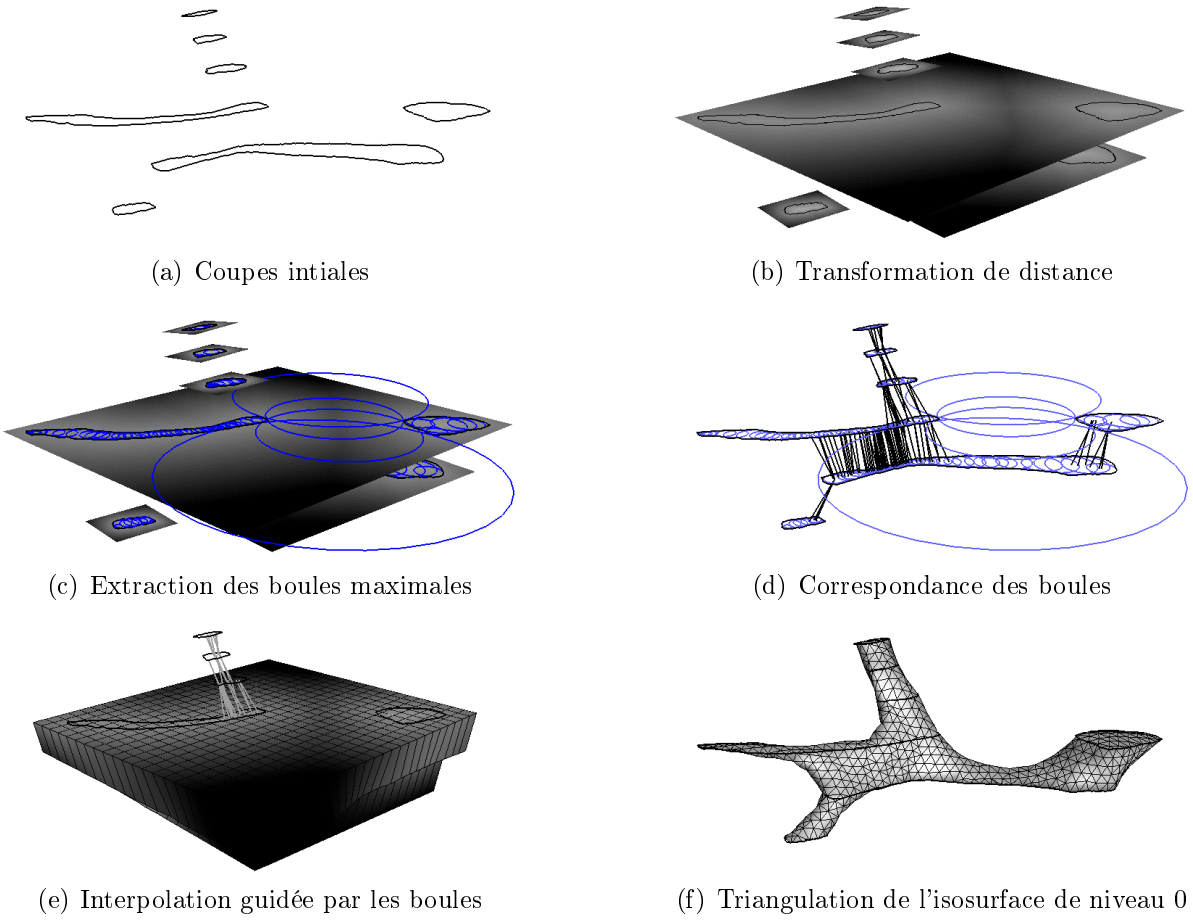


(d) Interpolation suivant la ligne liant les centres de gravité [32]

Fig. 9.5 – *Différentes méthodes d'interpolation illustrées sur une partie du système hépatique [60].*

Une méthode plus sophistiquée est le *maximal disc-guided interpolation* proposé par Treece et Prager [62],[60]. Le principe de la méthode est illustré à la figure 9.6. L'intérieur et l'extérieur de l'objet sont représentés par une série de boules maximales[2] (fig. 9.6(c)). Ces boules maximales sont déterminées dans chaque coupe à l'aide de la carte de distance de la coupe. Ensuite, une correspondance entre les boules des coupes adjacentes est calculée (fig. 9.6(d)). On détermine ainsi une direction d'interpolation différente en chaque en point (fig. 9.6(e)). Cette méthode permet d'obtenir de bons résultats, même lorsque la forme de l'objet est très différente d'une coupe à l'autre (fig. 9.6(f)).

[2]Une boule est dite maximale dans la forme si elle n'est incluse dans aucune autre boule



(a) Coupes intiales (b) Transformation de distance

(c) Extraction des boules maximales (d) Correspondance des boules

(e) Interpolation guidée par les boules (f) Triangulation de l'isosurface de niveau 0

Fig. 9.6 – *Illustration de l'algorithme du* ***Maximal disc-guided interpolation*** *sur le système hépatique de la figure 9.5 [60].*

9.3.2 Implémentation

Notre choix s'est porté sur la méthode d'interpolation fondée sur la forme de Raya et Udupa [46]. L'algorithme implémenté se résume comme suit :

1. L'image de départ est une image tridimensionnelle segmentée. L'épaisseur de coupe est arbitraire.
2. La carte de distance est calculée pour chaque coupe par une transformation de distance de chanfrein bidimensionnelle.
3. Les coupes intermédiaires sont calculées par interpolation des valeurs de distance obtenues à l'étape précédente. Le nombre et la position de ces nouvelles coupes sont déterminés de manière à obtenir des voxels cubiques.
4. L'isosurface de niveau 0 de l'image tridimensionnelle créée correspond à la frontière de l'objet d'intérêt.

Au lieu d'utiliser une transformation de distance du type *city-block* comme Raya et Udupa, nous avons utilisé la transformation de distance de chanfrein du type $5 \times 5 < 5, 7, 11 >$. En effet, dans l'analyse différentes transformations de chanfrein bidimensionnelles (section 8.5), nous avons montré que la distance de chanfrein $< 5, 7, 11 >$ donne de meilleurs résultats que la distance *city-block* car elle approxime mieux la distance euclidienne.

En plus de l'interpolation linéaire, nous avons utilisé l'interpolation par spline cubique basée sur la courbe spline de Catmull-Rom (fig. 9.7)[10]. L'interpolation par spline cubique estime la valeur d'un point situé entre deux points x_1, x_2 grâce à la connaissance de la valeur en x_1 et x_2 et de la valeur aux deux points situés de part et d'autre de x_1 et x_2, notés x_0 et x_3. Si $t \in [0,1]$ indique la position du point d'intérêt entre x_1 et x_2, la valeur recherchée vaut :

$$q(t) = \frac{1}{2}\left((2x_1) + (-x_0 + x_2)\,t + (2x_0 - 5x_1 + 4x_2 - x_3)\,t^2 + (-x_0 + 3x_1 - 3x_2 + x_3)\,t^3\right) \tag{9.1}$$

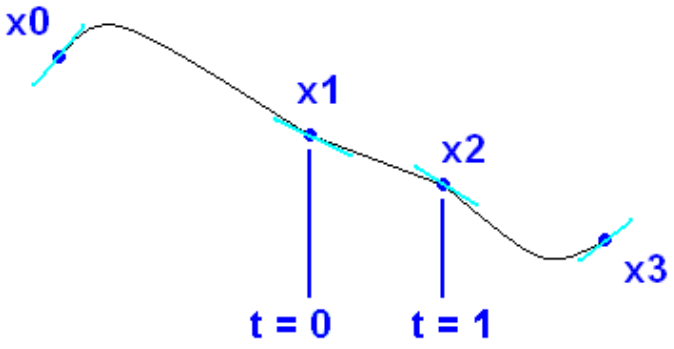


Fig. 9.7 – *Spline de Catmull-Rom*

La courbe spline de Catmull-Rom possède les caractéristiques intéressantes suivantes :
- Elle passe à travers les quatre points de contrôles, x_0,x_1,x_2 et x_3.
- Elle est continu et sa dérivée est continue (continuité C^1).

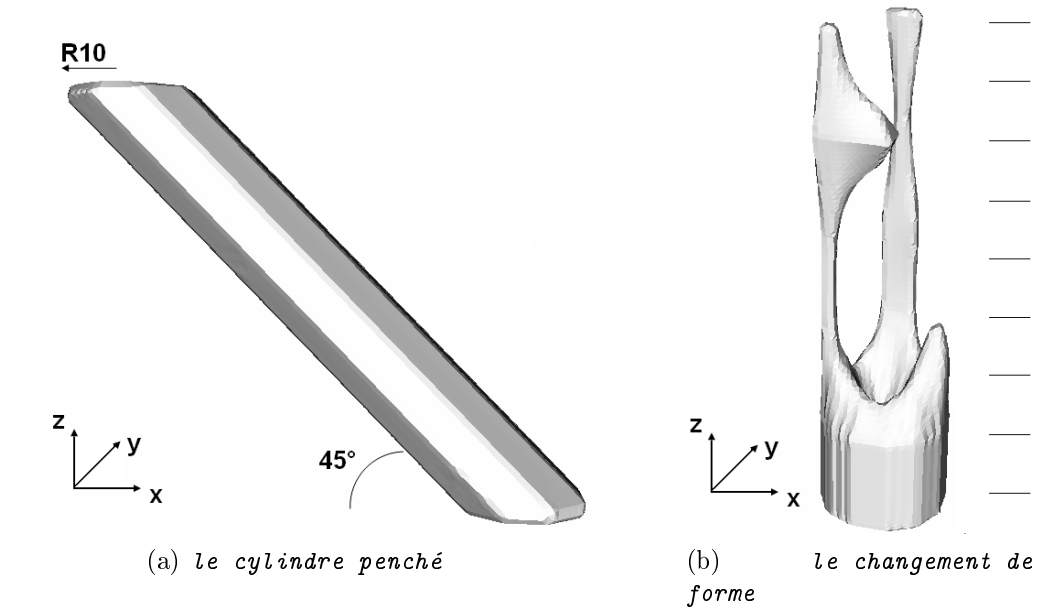
L'interpolation par spline cubique n'est pas possible entre la première et la deuxième coupe (x_0 non défini) et entre l'avant-dernière et la dernière coupe (x_3 non défini). Une interpolation linéaire sera alors utilisée.

9.4 Présentation et analyse des résultats

Nous allons comparer les quatre méthodes d'interpolation suivantes :
- La méthode de Sintorn et Borgefors [58], qui étend le calcul de la transformation de chanfrein tridimensionnelle au cas d'une trame anisotrope (section 9.2). Cette méthode ne sera que utilisée dans les cas où la distance inter-slice vaut $\Lambda = 1.5$, $\Lambda = 2.58$ et $\Lambda = 5$, distances pour lesquelles les poids optimaux du masque de chanfrein sont connus (tab.9.1).
- La méthode d'interpolation fondée sur la forme de Raya et Udupa [46] avec une interpolation linéaire entre coupes.
- La méthode d'interpolation fondée sur la forme de Raya et Udupa [46] avec l'interpolation cubique de Catmull-Rom [10].
- La méthode du *maximal disc-guided interpolation* via le mailleur surfacique *Isosurf* [60].

Pour comparer ces différentes méthodes, nous avons créé deux nouvelles géométries :
- Le *cylindre penché* 9.8(a) qui permet de voir l'influence d'une translation de l'objet d'une coupe à l'autre, sans changement de forme.

– Le *changement de forme* 9.8(b) dans laquelle le nombre d'objets et la taille de ceux-ci varie d'une coupe à l'autre.



(a) *le cylindre penché* (b) *le changement de forme*

Fig. 9.8 – *Nouveaux objets créés afin de comparer les méthodes d'interpolation entre elles.*

De plus, les 4 méthodes ci-dessus ont été appliquées aux géométries tests habituelles, pour lesquelles nous avons construit, sous format VTK, des images binaires d'épaisseurs de coupes $\Lambda = 1.5, 2.58, 5$.

9.4.1 Comparaison entre la méthode de Sintorn et Borgefors et la méthode d'interpolation fondée sur la forme

La géométrie de *la multiplication de deux sphères* permet de comparer les résultats des deux approches envisagées. Les maillages obtenus après extraction de la surface et pour différentes épaisseurs de coupe sont présentés à la figure 9.9. Les algorithmes du Vertex Clustering, d'amélioration de la topologie et repositionnement des noeuds ont ensuite été appliqués sur ces maillages (figure 9.10).

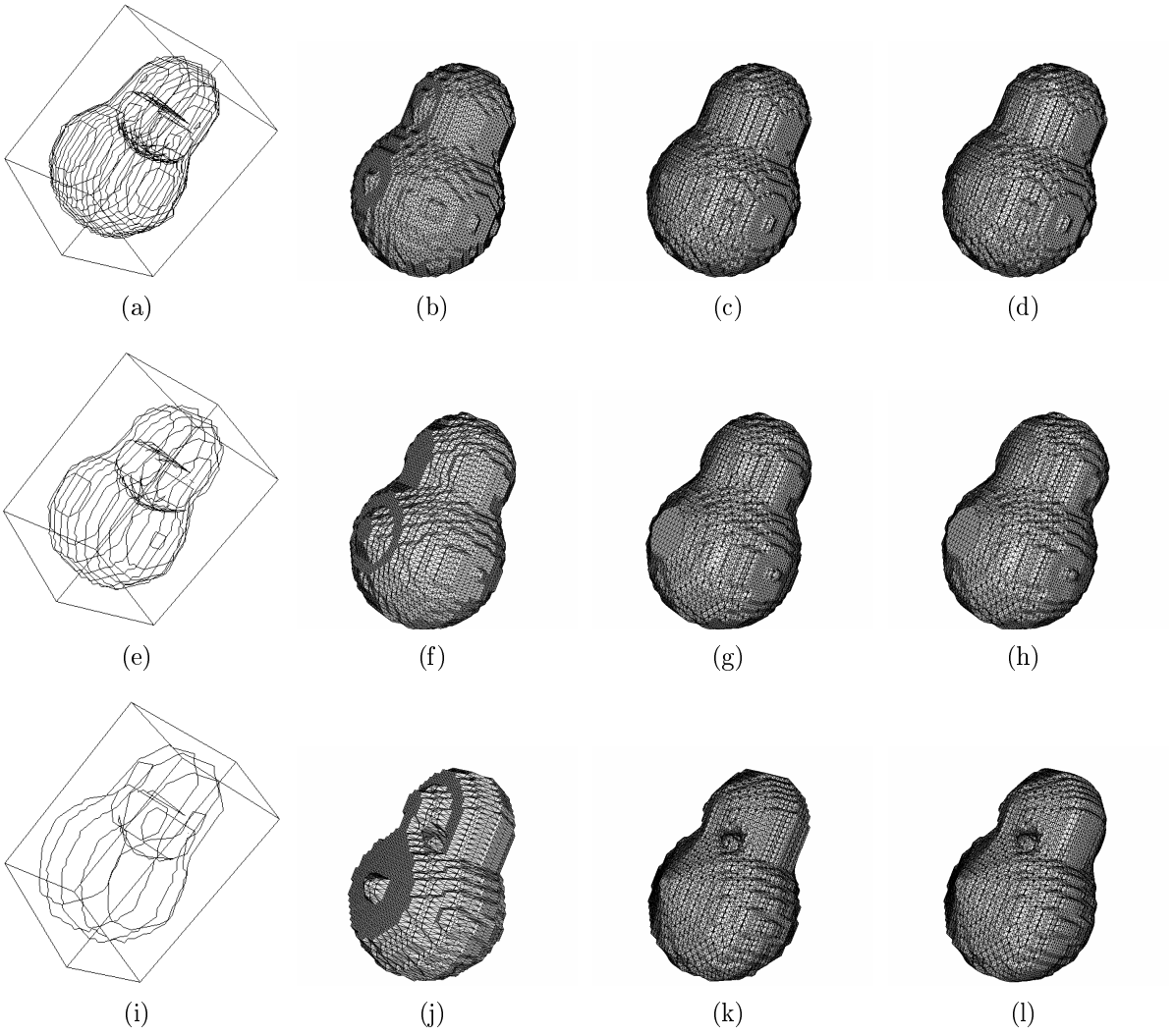


Fig. 9.9 – *La multiplication de deux sphères. Maillages obtenus après extraction de la surface. Comparaison de différentes méthodes d'interpolation : l'approche de Sintorn et Borgefors (colonne 2), la méthode d'interpolation fondée par la forme avec interpolation linéaire (colonne 3), la méthode d'interpolation fondée par la forme avec interpolation par spline cubique (colonne 4). La comparaison est réalisée pour différentes épaisseurs de coupe :* $\Lambda = 1.5$ *(ligne 1),* $\Lambda = 2.58$ *(ligne 2),* $\Lambda = 5$ *(ligne 3).*

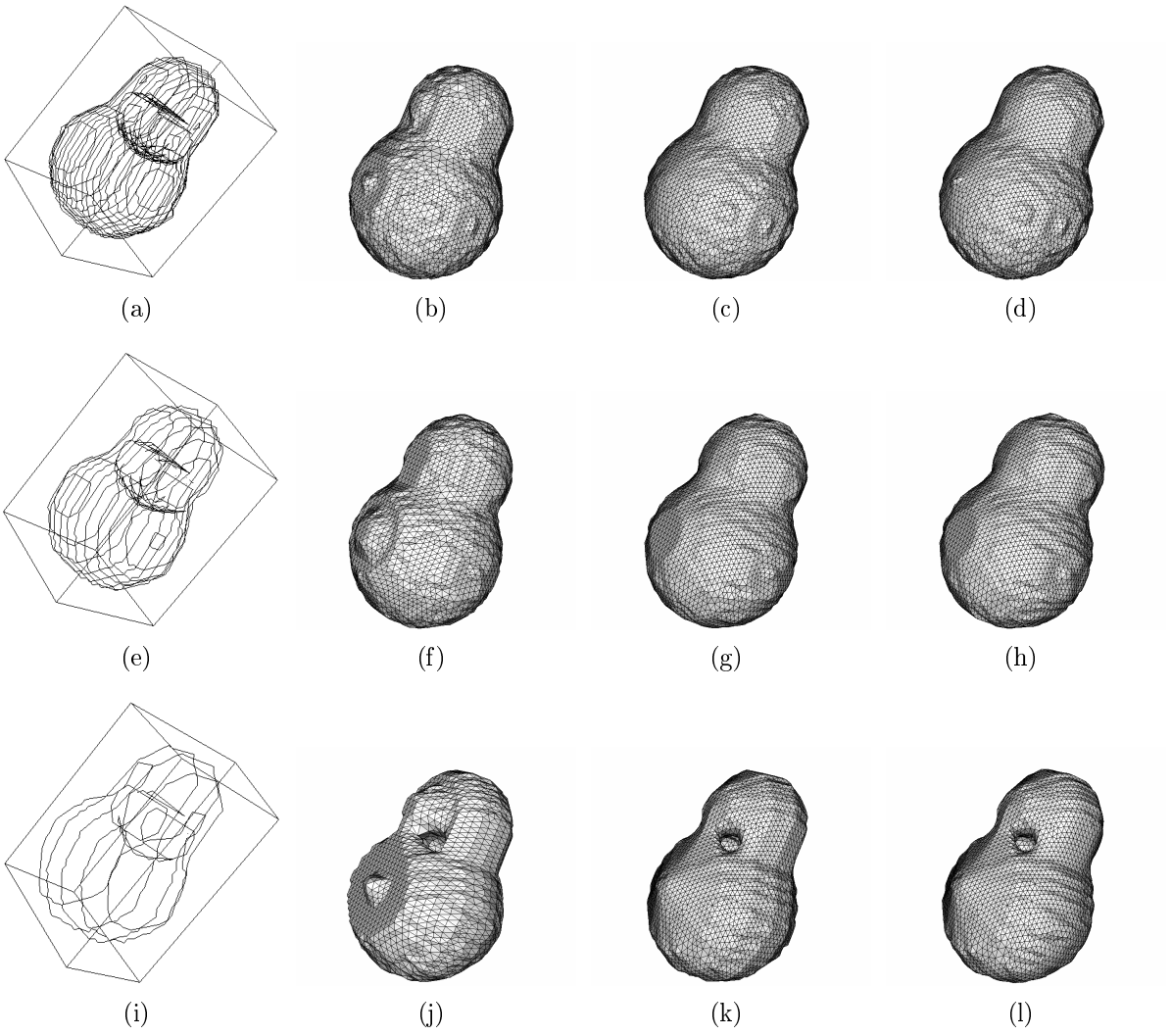


Fig. 9.10 – *La multiplication de deux sphères. Maillages obtenus après extraction de la surface, Vertex Clustering, amélioration de la topologie et repositionnement des noeuds. Comparaison de différentes méthodes d'interpolation : l'approche de Sintorn et Borgefors (colonne 2), la méthode d'interpolation fondée par la forme avec interpolation linéaire (colonne 3), la méthode d'interpolation fondée par la forme avec interpolation par spline cubique (colonne 4). La comparaison est réalisée pour différentes épaisseurs de coupe :* $\Lambda = 1.5$ *(ligne 1),* $\Lambda = 2.58$ *(ligne 2),* $\Lambda = 5$ *(ligne 3).*

Dans le cas de la méthode de Sintorn et Borgefors, le nombre de voxels à traiter dans les étapes ultérieures telles l'extraction de la surface est donc moindre, de même que le nombre de triangles générés (ce qui se vérifie aux figures 9.9 et 9.10). Le graphe de la figure 9.11 confirme que le temps nécessaire à la création du maillage surfacique est diminué par rapport à la méthode d'interpolation fondée sur la forme de Raya et Udupa. Cependant, aux figures 9.9 et 9.10, nous remarquons que la géométrie est moins bien approchée lorsqu'on utilise l'approche de Sintorn et Borgefors. Nous remarquons également que cette erreur croît avec la distance inter-slice Λ. Ceci est en accord avec le graphe de la figure 9.3 qui montrait que la carte de distance de chanfrein obtenue par la méthode de Sintorn et Borgefors n'est proche de la carte de distance euclidienne que pour de faible valeurs de Λ.

Un autre inconvénient de l'approche de Sintorn et Borgefors est le suivant. Dans l'algorithme du Marching Tetrahedra, l'espace est subdivisé en cubes, chaque cube étant ensuite subdivisé en 6 tétraèdres. Si, au lieu de cubes, nous partitionnons le domaine d'étude en parallélépipèdes afin de tenir compte de toute l'information comprise dans l'image, nous obtenons la situation de la figure 9.9 (j) où la majorité des triangles sont allongés dans la direction z. Cependant, l'application de l'algorithme de repositionnement des noeuds corrige tout à fait la situation.

En conclusion, l'approche de Sintorn et Borgefors permet de gagner en temps de calcul tout en donnant de bons résultats pour une épaisseur de coupe égale à $\Lambda = 1.5$. Au delà, nous utiliserons la méthode d'interpolation fondée sur la forme de Raya et Udupa (linéaire ou par spline cubique).

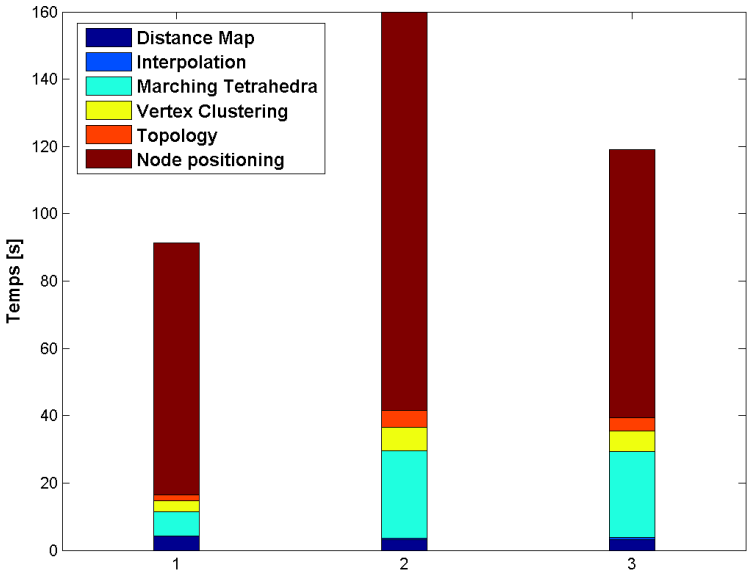


Fig. 9.11 – *Temps de calculs pour la création du maillage surfacique en fonction de l'approche utilisée. 1 : Extension de la transformation de chanfrien proprosée par Sintorn et Borgefors. 2 : Méthode d'interpolation fondée sur la forme avec interpolation linéaire. 3 : Méthode d'interpolation fondée sur la forme avec interpolation par spline cubique.*

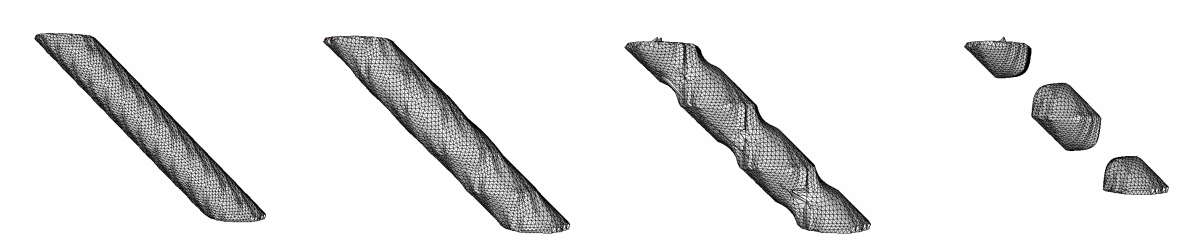
9.4.2 Le cylindre penché : effet d'une translation entre coupes successives

Les tests sur *le cylindre penché* permettent de visualiser la limitation majeure de la méthode d'interpolation fondée sur la forme de Raya et Udupa : sa réaction à une translation d'une section de l'objet d'une coupe à l'autre.

Dans l'image tridimensionnelle initiale du cylindre penché (fig.9.8(a)), à voxels cubiques et de taille $100 * 100 * 101$, nous avons prélevé des coupes espacées de $\Lambda = 5$, $\Lambda = 10$, $\Lambda = 15$ et $\Lambda = 20$, ce qui correspond respectivement à 21, 11, 7 et 3 coupes régulièrement espacées.

Les différentes méthodes d'interpolation proposées ont été appliquées sur ces 4 images 3D binaires créées. Les résultats obtenus par la méthode d'interpolation fondée sur la forme (avec l'interpolation par spline cubique de Catmull Rom) sont présentés à la figure 9.12. Nous observons que le cylindre est de moins en moins bien approché lorsque l'épaisseur de coupe

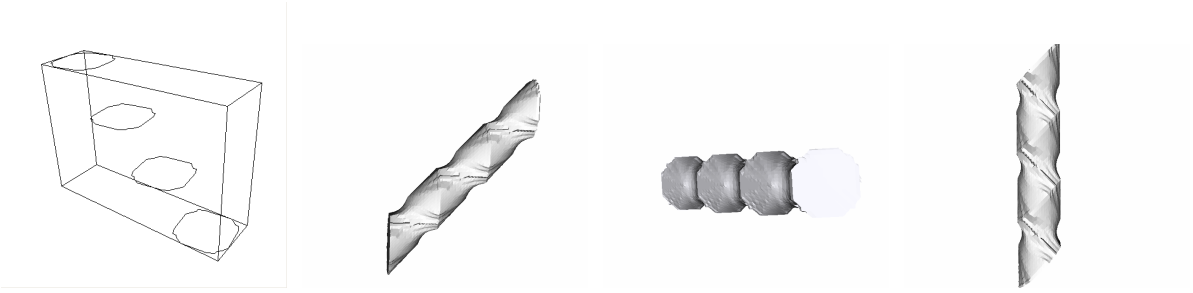
augmente. A un moment donné ($\Lambda = 20$), la translation entre coupes successives devient telle qu'une interpolation selon z ne permet plus de reconstruire le cylindre.



(a) $\Lambda = 5$ (b) $\Lambda = 10$ (c) $\Lambda = 15$ (d) $\Lambda = 20$

Fig. 9.12 – *Étude de la méthode d'interpolation fondée sur la forme (avec interpolation par spline cubique) sur le* `le cylindre penché`*. Les maillages présentés sont ceux obtenus après extraction de la surface, simplification et amélioration de la qualité du maillage.*

La figure 9.13 présente les différentes vues du cylindre obtenu pour $\Lambda = 15$. La vue de face montre bien la concavité du cylindre obtenu par la méthode d'interpolation fondée sur la forme. Cette concavité est due au fait que l'objet interpolé ne peut se situer qu'entre les frontières des coupes originales (perpendiculaires à la direction d'interpolation). Ceci se vérifie dans la vue de face. Notons que ce problème ne serait pas apparu si le cylindre avait été scanné perpendiculairement à son axe.

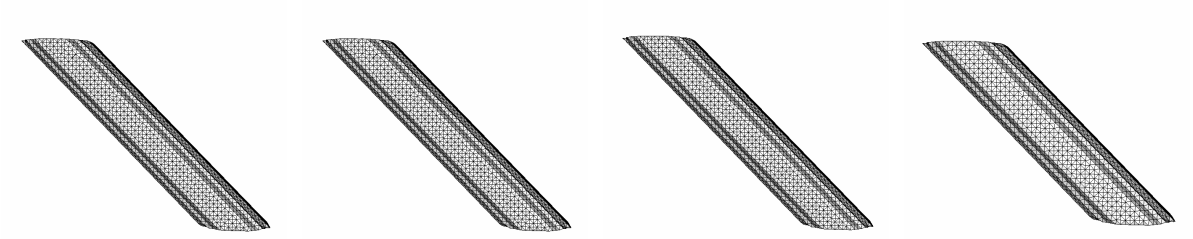


(a) coupes initiales (b) côté (c) dessus (d) de face

Fig. 9.13 – *Différentes vues* `du cylindre penché` *obtenu par la méthode d'interpolation fondée sur la forme.* $\Lambda = 20$.

L'origine du problème est que dans la méthode d'interpolation fondée sur la forme, la direction d'interpolation est fixe (perpendiculairement aux coupes) et ne tient pas compte de l'orientation de l'objet dans l'image 3D. Dans le cas du cylindre penché, il vaudrait mieux interpoler selon l'axe du cylindre que perpendiculairement aux coupes. Dans ce cas la méthode de Higgins et al. [32], dans laquelle l'interpolation se fait selon la ligne joignant les centres de gravité, permettrait d'obtenir de meilleurs résultats. Il en est de même pour la méthode du *maximal disc-guided interpolation* de Treece et Prager dans laquelle la direction d'interpolation varie d'un point à l'autre de l'image et est calculée de manière à lier d'une coupe à l'autre les régions qui se ressemblent [60, 62]. La figure 9.14 montre les résultats obtenus par la méthode du *maximal disc-guided interpolation* utilisée par *Isosurf*. Le cylindre est très bien approché et le maillage obtenu est pratiquement le même pour les différentes épaisseurs de coupe.

Ainsi, lorsque l'objet subit une translation d'une coupe à l'autre, la méthode du *maximal disc-guided interpolation* donne de meilleurs résultats que la méthode d'interpolation fondée sur la forme implémentée dans le cadre de ce travail.



(a) $\Lambda = 5$ (b) $\Lambda = 10$ (c) $\Lambda = 15$ (d) $\Lambda = 20$

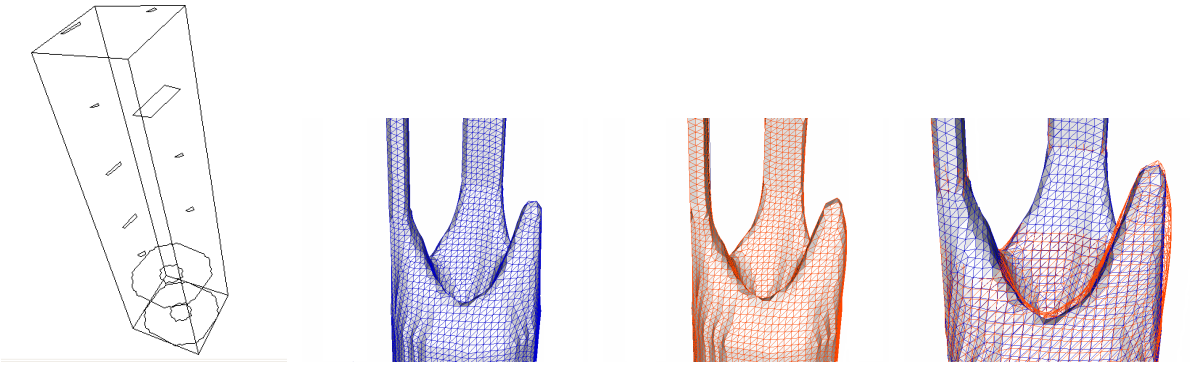
Fig. 9.14 – *Maillages* `du cylindre penché` *obtenus à l'aide d'Isosurf.*

9.4.3 Effet d'un changement de forme entre coupes successives

La géométrie du `changement de forme` permet de voir comment les différentes méthodes d'interpolation réagissent à changement de la forme de l'objet entre deux coupes successives (fig. 9.8).

Les résultats sont présentés à la figure 9.15. Nous remarquons que le fait que le nombre d'objets varie d'une coupe à l'autre ne pose pas de problème particulier. Par contre la méthode d'interpolation influe sur la forme de l'objet final. La figure 9.15(d) montre la différence entre l'interpolation linéaire (en blue) et l'interpolation par spline cubique (en rouge).

L'interpolation par spline cubique est celle conseillée par Raya et Udupa [46] et Herman [31] car elle donne généralement des meilleurs résultats visuels que l'interpolation linéaire pour des temps de calculs similaires. Celle donc elle que nous utiliserons dans la suite du travail.



(a) Coupes initiales (b) linéaire (c) cubique (d) Superposition

Fig. 9.15 – *La géométrie du* `changement de forme` *: Effet de la méthode d'interpolation utilisée.*

Troisième partie

Utilisation du mailleur surfacique créé pour le développement de modèles biomécaniques du cerveau

CHAPITRE 10

Introduction

Dans les chapitres précédents, nous avons testé le mailleur surfacique developpé sur des objets réguliers, définis par des formes analytiques. Le maillage de formes naturelles telles le cerveau est plus complexe car les surfaces des objets naturels sont irrégulières.

L'objectif de cette troisième partie est de tester notre programme sur des objets naturels en illustrant deux applications possibles du mailleur surfacique :

1. La première concerne la simulation numérique par la méthode des éléments finis d'organes soumis à des sollicitations externes (chap. 11). Nous prenons l'exemple de l'affaissement d'un cerveau soumis à la gravité. Un maillage surfacique est tout d'abord généré au départ d'images segmentées. Ensuite nous utilisons le mailleur volumique TetGen pour générer un maillage volumique tétraédrique. Celui-ci est alors introduit dans le logiciel de calcul éléments finis Metafor. Dans Metafor, un système de mise en charge est appliqué et les déformations sont calculées.
2. La seconde application est une illustration du projet multidisciplinaire réalisé à l'Université de Liège visant à améliorer la planification et le guidage d'opérations neurochirurgicales (neurochirurgie assistée par ordinateur).

Dans les deux cas, le cerveau est modélisé comme un matériau élastique linéaire isotrope. Ce modèle permet de bien approcher les déformations réelles lorsque celles-ci restent faibles. Le comportement élastique linéaire dépend de deux paramètres : le module d'Young E et le coefficient de poisson ν. Bien-entendu, chaque constituant du cerveau possède ses propres caractéristiques E et ν qu'il faudrait inclure dans notre modèle pour approcher au mieux les déformations réelles du cerveau. Cependant, par simplicité, nous considérons le cerveau comme étant un milieu homogène. Matière grise, matière blanche et liquide céphalo-rachidien sont donc considérés comme un seul matériau caractérisé par les valeurs moyennes $E = 3000\,Pa$ et $\nu = 0.45$. Ces valeurs sont celles utilisées par M. Ferrant [20, 21] ainsi que par J. Verly et L. Vigneron [64, 65].

CHAPITRE 11

Application 1 : Développement de modèles éléments finis du cerveau

Dans ce chapitre, nous montrons comment le mailleur surfacique développé dans les deux premières parties du travail peut être utilisé pour la simulation par la méthode des éléments finis des déformations subies par le cerveau sous un système de forces connues et définies explicitement.

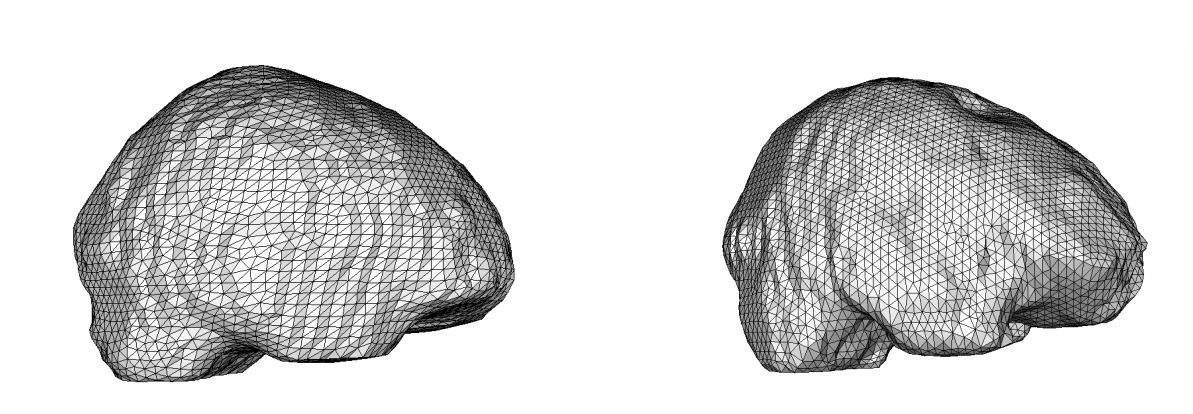
11.1 Génération d'un maillage surfacique du cerveau

Les images médicales contenant le cerveau ont été préalablement segmentées (manuellement). L'image tridimensionnelle traitée dans ce chapitre est constituée de 59 coupes de 255×255 pixels. L'espacement entre pixels dans le plan est de $0.9675\,mm$ et la distance entre deux coupes successives vaut $2.5\,mm$. Parmi les méthodes d'interpolation implémentées, nous avons choisi la méthode d'interpolation fondée sur la forme avec interpolation par spline cubique de Catmull-Rom puisque c'est elle qui donnait les meilleurs résultats (chap. 9).

(a) Image 3D initiale (b) Image 3D segmentée

Fig. 11.1 – *Les images introduites dans notre mailleur surfacique ont été préalablement segmentées.*

Le mailleur surfacique que nous avons développé dans ce travail permet de générer des maillages d'organes humains tels le cerveau. A la figure 11.2, nous présentons plusieurs maillages de cerveau, obtenus pour des patients différents. Le patient 1 est le patient qui sera étudié dans la suite, le patient 2 et celui qui correspond à l'image IRM 3D de la figure 11.1.



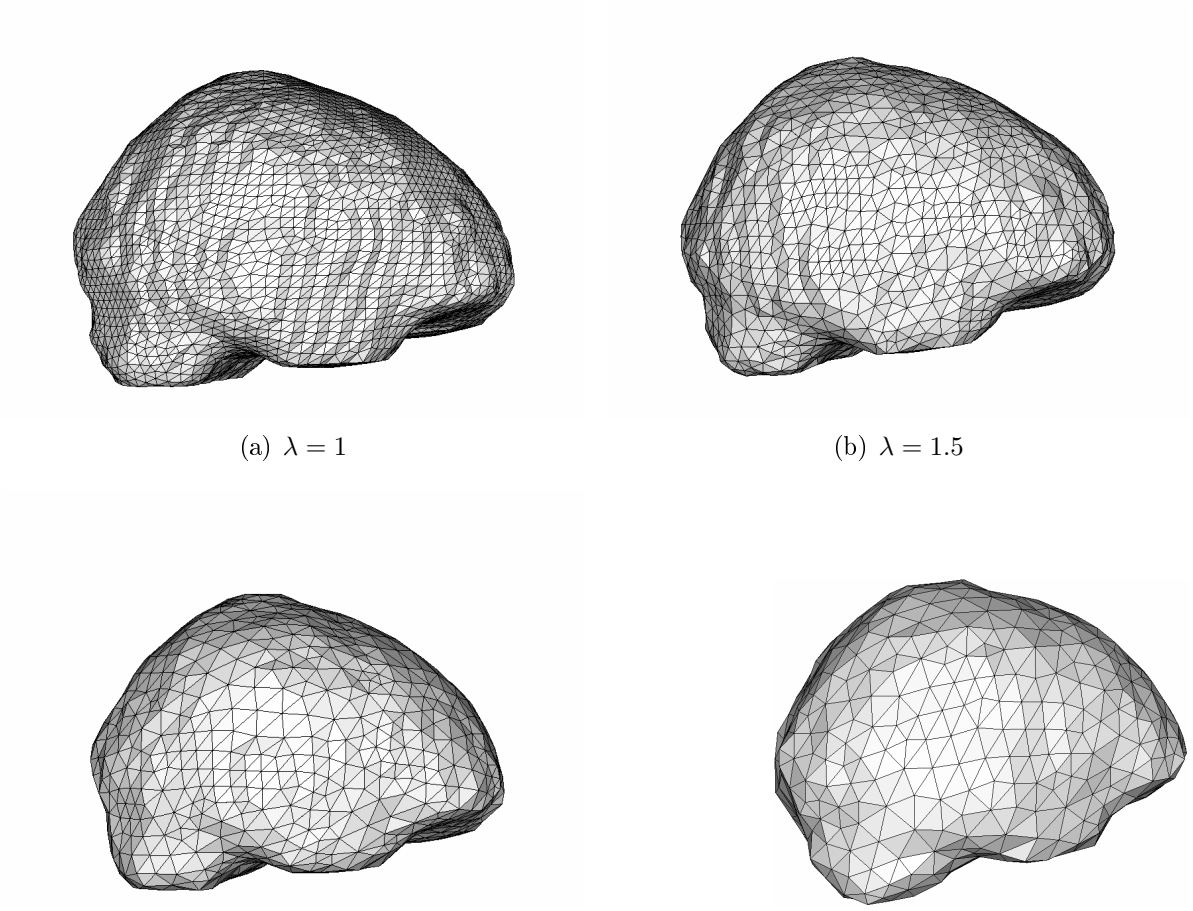
(a) Patient 1 (b) Patient 2

Fig. 11.2 – *Maillages surfaciques du cerveau obtenu à l'aide de notre programme.*

Dans le cadre de simulations numériques lors d'interventions neurochirurgicales, il peut être intéressant de réduire la finesse du maillage afin respecter la contrainte de temps réel. Ceci est tout à fait possible avec notre mailleur surfacique et peut se faire de deux manières :

1. La résolution de l'image tridimensionnelle peut être diminuée en entrée du programme, avant l'extraction de la surface. Les arrêtes auront ainsi une taille moyenne proche de la nouvelle résolution (distance inter-voxels).
2. L'algorithme de simplification du maillage, basée sur la méthode du Vertex Clustering, peut être réglé de manière à réduire plus ou moins le nombre de triangles. Ce réglage est réalisé en ajustant la finesse de la grille du Vertex Clustering, comme expliqué dans le chapitre 4. On agit ainsi directement sur la taille moyenne des arrêtes.

La première méthode est plus rapide, puisque l'extraction de la surface est réalisé sur un image réduite. La seconde méthode par contre, permet de réduire la finesse du maillage généré en tenant compte de l'entièreté de l'information contenue dans l'image originale. C'est donc cette seconde approche qu'il convient d'utiliser car elle fournit un maillage qui, pour un même nombre de mailles, est plus proche la surface réelle. Dans ce chapitre nous avons préalablement réduit la taille de l'image d'un facteur 4 ; ensuite différentes finesses de maillage ont été obtenus en ajustant la grille du Vertex Clustering. Ceux-ci sont montrés à la figure 11.3 où λ désigne le rapport entre le côté du cube élémentaire de la méthode du Vertex Clustering et celui du Marching Tetrahedra (qui correspond à la distance entre deux pixels dans une coupe de l'image initiale).



(a) $\lambda = 1$ (b) $\lambda = 1.5$

(c) $\lambda = 2$ (d) $\lambda = 2.5$

Fig. 11.3 – *La finesse du maillage généré peut être ajustée en réglant la finesse de la grille du Vertex Clustering.*

Le tableau 11.1 reprend quelques résultats numériques relatifs aux maillages de la figure 11.3.

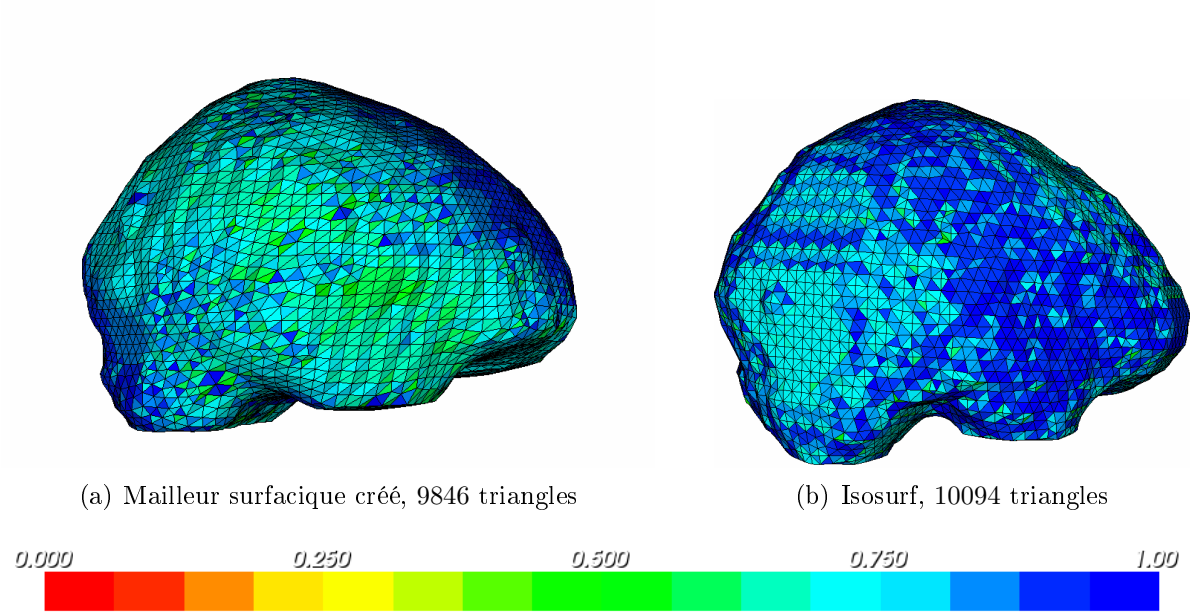
	$\lambda = 1$	$\lambda = 1.5$	$\lambda = 2$	$\lambda = 2.5$
Nombre de noeuds	4924	2087	1204	746
Nombre de triangles	9846	4170	2404	1492
Temps de génération du maillage [s]	93	48	38	27

Tab. 11.1 – *Génération de maillages surfaciques du cerveau. Résultats pour différentes finesses du maillage.*

Il est intéressant de comparer nos résultats avec ceux obtenus à l'aide du logiciel *Isosurf* [61]. A la figure 11.4, nous montrons deux maillages l'un obtenu par notre mailleur l'autre par *Isosurf*. Ces maillages surfaciques ont été obtenus à partir de la même image segmentée. Les paramètres introduits dans *Isosurf* sont les paramètres par défaut, hormis le lissage préalable de l'image, que nous avons désactivé par mesure de comparaison. Les triangles ont été coloriés en fonction de leur qualité[1]. Nous remarquons que bien que la qualité du maillage généré par

[1]La mesure de qualité utilisée est celle conseillée par Frey [23].

notre programme est relativement bonne, il reste encore des progrès à faire pour atteindre la qualité des maillages fournis par *Isosurf*. Dans ce sens, une méthode du type *edge swapping* pourrait être envisagée dans un travail futur. Cette méthode consiste à parcourir le maillage et inverser la diagonale des quadrilatères lorsqu'un critère, par exemple sur la qualité des deux triangles formant le quadralitère, est vérifié. Enfin, le tableau 11.2 indique que dans ce cas-ci notre programme génère un peu moins de triangles qu'*Isosurf*.



(a) Mailleur surfacique créé, 9846 triangles

(b) Isosurf, 10094 triangles

(c) Échelle de couleur pour la mesure de la qualité

Fig. 11.4 – *Comparaison des maillages surfaciques obtenus à l'aide de notre programme avec ceux obtenus à l'aide du logiciel Isosurf.*

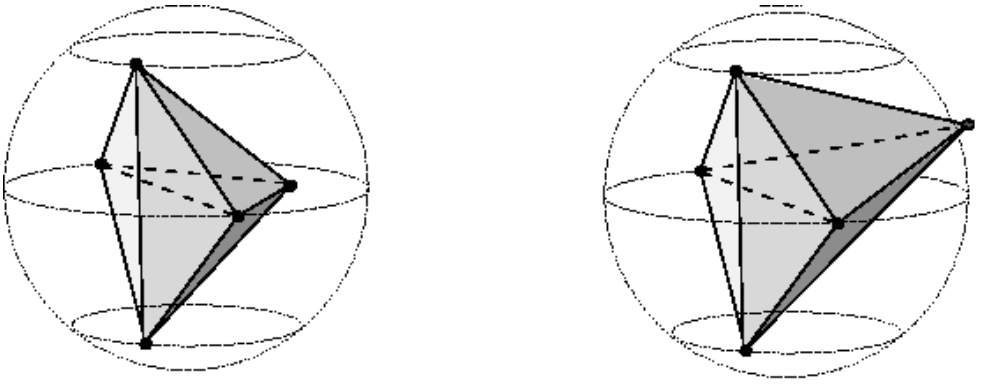
	Mailleur créé	Isosurf
nombre de noeuds	4924	9846
nombre de triangles	9846	5049
temps de calcul	130,296462	11

Tab. 11.2 – *Comparaison du mailleur développé dans ce travail avec Isosurf.*

11.2 Génération du maillage volumique à partir du maillage surfacique

Le maillage est ensuite étendu au domaine entier à l'intérieur de la surface fournie par notre mailleur surfacique. Le mailleur volumique utilisé est TetGen [54], qui est basé sur la méthode de Delaunay. Le maillage volumique fourni par TetGen est de bonne qualité et adapté au calcul éléments fins. En particulier, les tétraèdres vérifient le critère de Delaunay.

Définition 11.1 (Critère de Delaunay ou critère de la boule vide) *Soit V un ensemble de points et t un tétraèdre dont les sommets appartiennent à V. Le tétraèdre respecte le critère de Delaunay si la sphère circonscrite à t ne contient aucun point de V (fig. 11.5).*



(a) critère de Delaunay non respecté (b) critère de Delaunay respecté

Fig. 11.5 – *Illustration du critère de Delaunay [22].*

La qualité d'un tétraèdre peut être mesurée par le rapport

$$Q = \frac{R}{L}$$

où R est le rayon du cercle circonscrit au tétraèdre et L est la longueur de la plus courte arrête. Ce rapport est faible pour des tétraèdres de forme optimale et grandit lorsque la forme du tétraèdre devient mauvaise (fig. 11.6). TetGen assure que tous les tétraèdres générés ont un rapport R/L inférieur à 1.414.

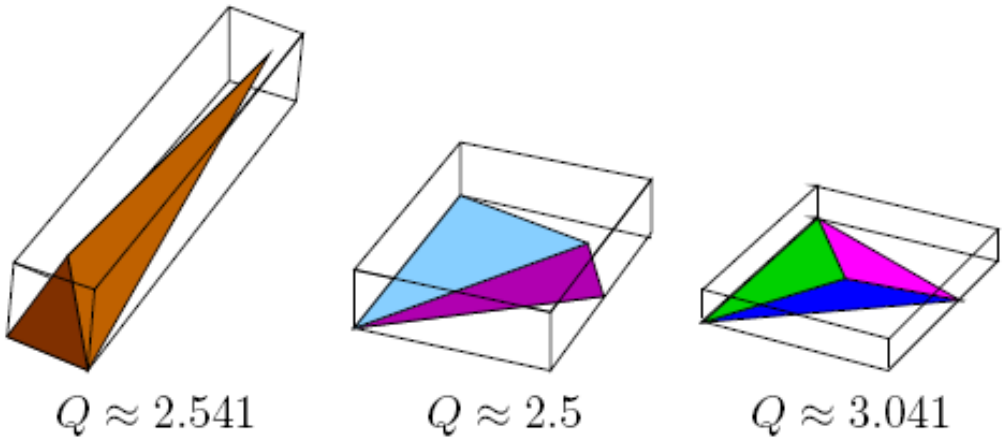
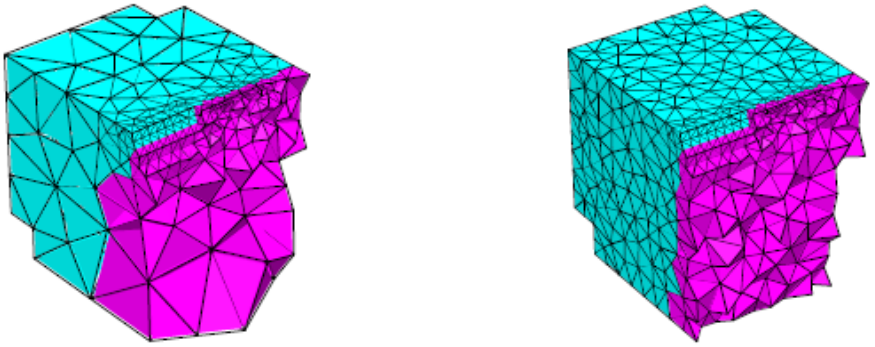


Fig. 11.6 – *Mesure de la qualité d'un tétraèdre.*

Deux paramètres permettent de régler la forme et la taille des tétraèdres fournis en sortie de TetGen :

1. q, une borne minimale sur la qualité, la valeur par défaut étant 2.0.
2. a, une borne maximale sur le volume des tétraèdres.

La manière dont ces paramètres influe sur les résultats obtenus est schématisé à la figure 11.7.



(a) Borne sur la qualité q (b) Borne sur le volume maximal a

Fig. 11.7 – *Réglage des paramètres dans TetGen.*

Dans le cas présent, les valeurs des paramères ont été ajustées à $q = 2$ et $a = 10$, ce qui donne le maillage volumique tétraédrique de la figure 11.8.

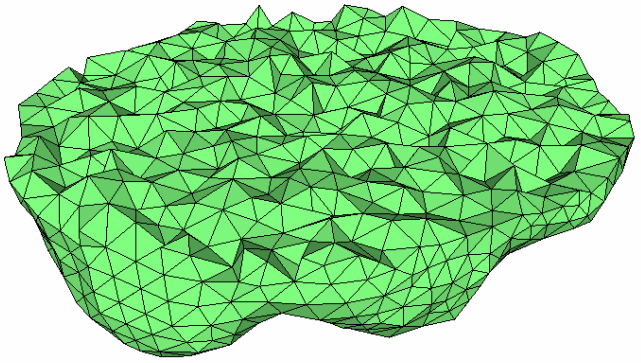


Fig. 11.8 – *Maillage volumique tétraédrique obtenu à l'aide du logiciel TetGen.*

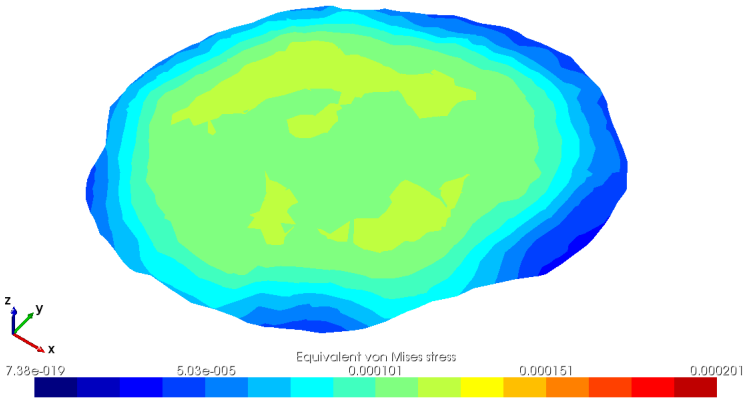
11.3 Calcul éléments finis dans Metafor

La qualité de maillage volumique tétraédrique obtenu à l'étape précédente est très bonne et donc un calcul éléments finis est possible. L'exemple choisi ici est de simuler l'affaissement d'un cerveau soumis à la gravité. Le contact avec la boite crânienne n'est pas modélisé.

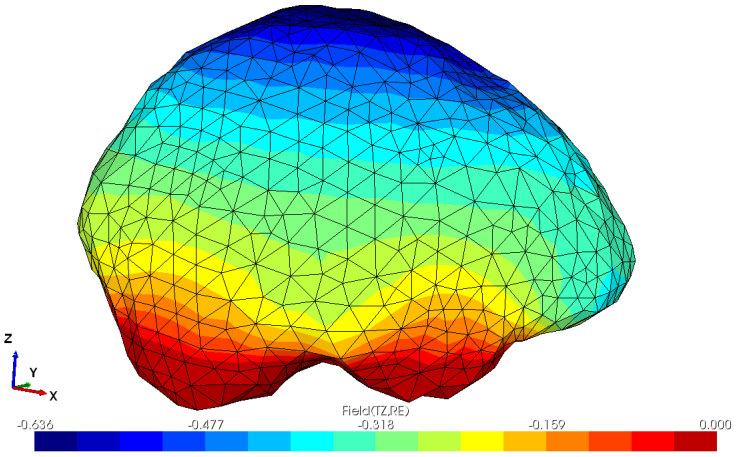
Tout d'abord, un fonction permettant de charger un maillage du type VTK a été créée dans Metafor. Le maillage volumique obtenu à l'étape précédente a donc pu être introduit dans le logiciel de calcul éléments finis. La mise en charge a ensuite été appliquée. Nous avons choisi de fixer les noeuds de la partie inférieure du cerveau et d'appliquer une force de gravité à l'ensemble des noeuds du maillage volumique. Metafor est un code non linéaire prenant en compte les grandes déformations et les rotations du cerveau sous l'effet de la gravité. Pour garantir la convergence, une mise en charge incrémentale (progressive) doit être mise en place. C'est pour cette raison qu'on applique la gravité au moyen d'une fonction rampe. Au lieu d'utiliser le solveur Skyline, le solveur par défaut de Metafor, nous avons pris le solveur direct Paradiso

(Direct Sparse Solver). Celui-ci a l'avantage d'être moins gourmand en mémoire que le solveur SKYLINE et d'être plus rapide lorsque la matrice de raideur possède une grande largeur de bande. Or, dans notre cas, la matrice de raideur est loin d'être diagonale puisque le maillage est isotrope et le cerveau est massif (même densité de mailles dans toutes les directions).

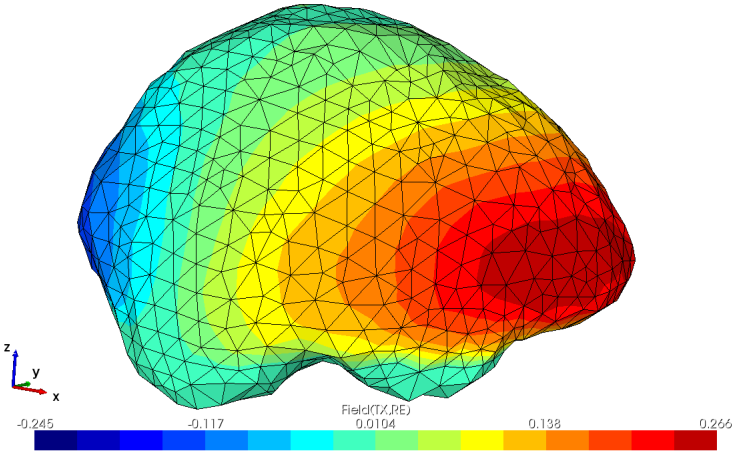
Les résultats obtenus sont présentés à la figure 11.9. Comme attendu, les contraintes sont maximales à l'intérieur du volume ; la surface étant libre. Le déplacement de la matière selon z est d'autant plus important que la matière se situe loin du points fixés (le bas du cerveau). La figure 11.9 (c) montrant le champ de déplacement selon x indique aussi que le cerveau s'écrase.



(a) Contrainte équivalente de Von Mises dans une coupe, MPa



(b) Déplacement selon z, mm



(c) Déplacement selon x, mm

Fig. 11.9 – *Simulation éléments finis d'un cerveau soumis à la gravité.*

CHAPITRE 12

Application 2 : Mise à jour des images neurochirurgicales pré-opératoires du cerveau par un modèle physique déformable

Ce chapitre montre comment ce travail de fin d'études s'insère dans le contexte de la recherche développée à l'Université de Liège concernant la mise à jour des images neurochirurgicales pré-opératoires au moyen de modèles physiques déformables. Dans ce but, le mailleur a été intégré dans une chaîne de calcul préexistante, remplaçant d'Isosurf.

12.1 Contexte

Les erreurs de navigation dues à la déformation du cerveau au cours de l'intervention pourraient être réduites si l'on pouvait acquérir, tout au long de l'opération, des images mises à jour de la même qualité que les images préopératoires. Malheureusement, vu les contraintes matérielles et spatiales lors de l'opération, les images acquises en salle d'opération n'ont pas une résolution suffisante pour permettre au chirurgien de se baser exclusivement sur elles en cours d'opération. La solution pour pouvoir fournir des images de qualité au chirurgien pendant son opération est de mettre à jour les images pré-opératoires à l'aide d'images intra-opératoires. On parle de recalage non rigide[1]. Une des méthodes pour effectuer ce recalage est de modéliser le comportement mécanique du cerveau lors d'interventions neurochirurgicales à l'aide d'un modèle numérique créé à partir des images pré-opératoires. Ce type de modélisation fait partie de l'ensemble des *modèles physiques déformables*.

Cependant, cette stratégie de prédiction des déformations subies par le cerveau en cours d'opération n'a pas encore été implémentée en pratique. C'est pour remédier à cet état de fait qu'un projet interdisciplinaire de recherche à long terme, impliquant médecins et ingénieurs, est né à l'Université de Liège. Les neurochirurgiens P. Robe et D. Martin apportent leur connaissance du cerveau. J. Verly et L. Vigneron, de l'unité de recherche en exploitation des signaux et images (Intelsig, Institut Montefiore) s'occupent de la partie traitement d'images. En plus de l'Institut Montefiore, ce projet de recherche implique les deux autres grands départements

[1] Il existe deux types de recalage d'images : le *recalage rigide* qui permet d'aligner différentes images d'un même objet par translation, rotation ou agrandissements et le *recalage non rigide* qui permet en plus de tenir compte du changement de forme de l'objet

de la faculté des Sciences appliquées. D'une part le département d'aérospatiale et mécanique dont le rôle est de simuler la déformation du cerveau au moyen d'un calcul éléments finis réalisé à l'aide de Metafor (JP Ponthot, R. Boman). D'autre part, le département d'architecture, géologie, environnement et constructions qui fournit les modèles élastiques et viscoélastiques utiles pour établir les lois de déformation du cerveau (S. Cescotto, A.M. Habraken). Sur le diagramme de collaboration de la figure 12.1, réalisé par R. Boman fin 2006, nous remarquons que personne ne s'occupe de la création du maillage à partir des images segmentées. C'est ainsi que ce travail de fin d'études est né.

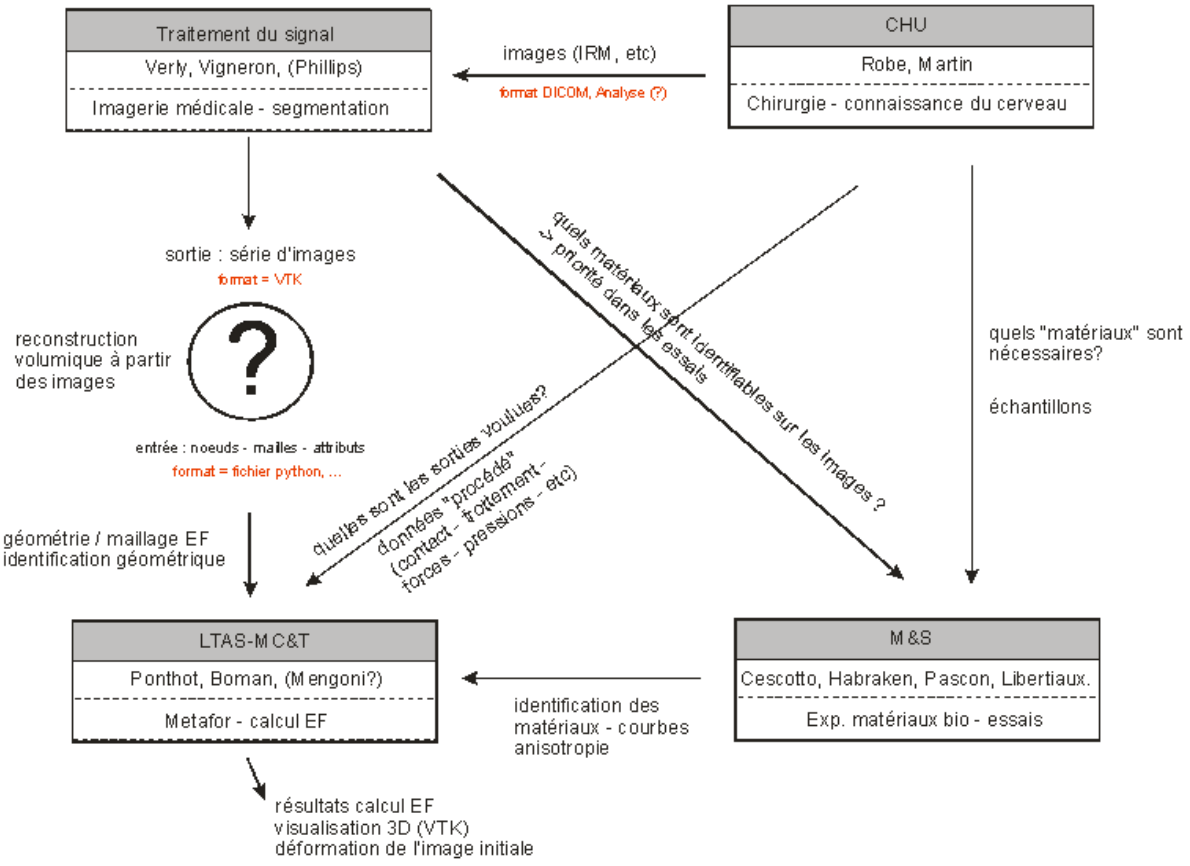


Fig. 12.1 – *Diagramme de collaboration du projet au sein de l'ULg (R.Boman, oct. 2006).*

L'approche utilisée pour simuler les déformations du cerveau provoquées par l'ouverture de la boîte crânienne, puis par la résection de la tumeur, est résumée à la figure 12.2.

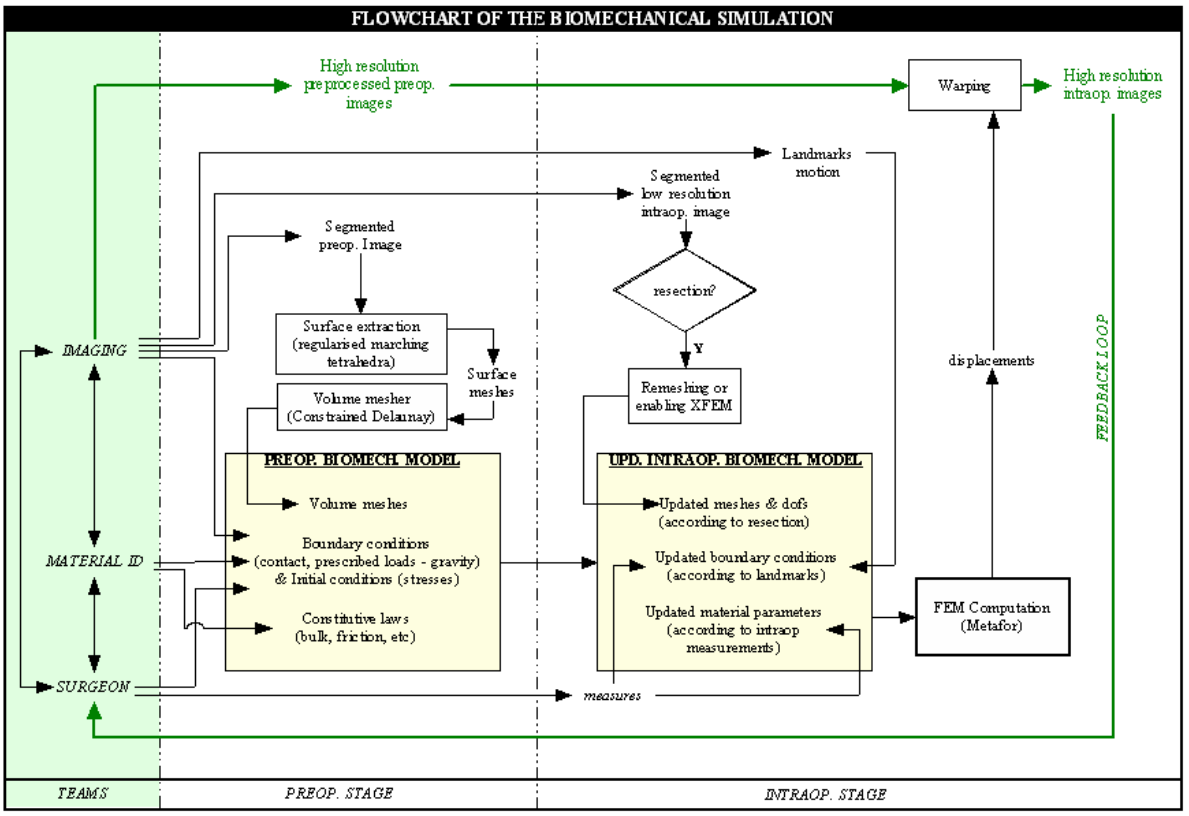


Fig. 12.2 – *Présentation du projet de recherche* `biomec` *developpé à l'Ulg (R.Boman, oct. 2006).*

L'objectif donc est de déformer les images pré-opératoires de bonne qualité, à l'aide des images intra-opératoires (de qualité limitée), pour fournir au chirurgien, tout au long de son opération, des images mises à jour de la même qualité que les images pré-opératoires (boucle extérieure de la figure 12.2). Pour ce faire, un modèle biomécanique de cerveau est tout d'abord créé à partir des images pré-opératoires. Ce modèle numérique est défini par :

1. **Un maillage volumique du cerveau**. L'image pré-opératoire est tout d'abord segmentée. La surface est alors extraite, par exemple au moyen du programme développé dans ce travail. Le maillage surfacique obtenu est alors introduit dans un mailleur volumique.
2. **Les conditions aux limites et conditions initiales**, donnant l'état de contrainte du cerveau du patient avant l'ouverture de son crâne.
3. **Les lois constitutives** des différents constituants du cerveau.

Pendant l'opération, des images intra-opératoires sont acquises. Ces nouvelles images réactualisées du cerveau du patient vont permettre de déduire les déplacements subis par le cortex. Ce champ de déplacement de la surface du cerveau est ensuite appliqué sur le modèle biomécanique créé. Un calcul éléments finis est alors réalisé pour déterminer le déplacement du cerveau en tout point. Ayant déterminé le champ de déplacement total, on peut déterminer le déplacement de chaque voxel appartenant au cerveau de l'image pré-opératoire et mettre à jour cette image.

Au lieu d'appliquer un champ de déplacement aux noeuds de surface du modèle biomécanique du cerveau, une autre approche aurait été de leur appliquer un système de forces. Cependant, ceci nécessiterait de modéliser chacune des forces intervenant lors de l'opération,

telles que celles appliquées par le chirurgien, celles dues à la perte du liquide céphalo-rachidien, ou encore celles dues à la présence du produit anesthésique. Vu le nombre de variables entrant en jeu, le modèle ainsi obtenu est beaucoup plus complexe que celui adopté.

12.2 Introduction du mailleur surfacique dans le projet de recherche

Jusqu'à présent le maillage surfacique nécessaire à la création du modèle numérique du cerveau est réalisé au moyen du logiciel *Isosurf*. Nous montrons dans ce chapitre les résultats qui seraient obtenus si l'on substitue *Isosurf* par le mailleur surfacique élaboré dans les chapitres précédents. La partie traitement d'images a été réalisée par L. Vigneron d'une manière similaire à ce qu'elle a fait dans sa thèse [65] et dans l'article [64].

La déformation principale du cerveau lors de l'enlèvement d'une tumeur est due à l'affaissement du cerveau sous l'effet de la gravité. On parle de *brain-shift*. Les autres types de déformations, provenant des actes chirurgicaux tels que l'incision, la rétraction et la résection, ne sont pas prises en compte dans la modélisation réalisée ici. Comme précédemment (chap. 11), le cerveau est considéré comme étant un milieu homogène, formé d'un seul matériau élastique linéaire isotrope de caractéristiques $E = 3000\,Pa$ et $\nu = 0.45$.

Ces hypothèses permettent de simplifier le schéma de la figure 12.2, de la manière indiquée à la figure 12.3. Les différentes étapes nécessaires à la mise à jour de l'image préopératoire sont la segmentation, la génération du maillage (surfacique puis volumique), la détermination du champ de déplacements initial, le calcul éléments finis et la déformation de l'image préopératoire. Chacune d'elle est illustrée dans ce qui suit.

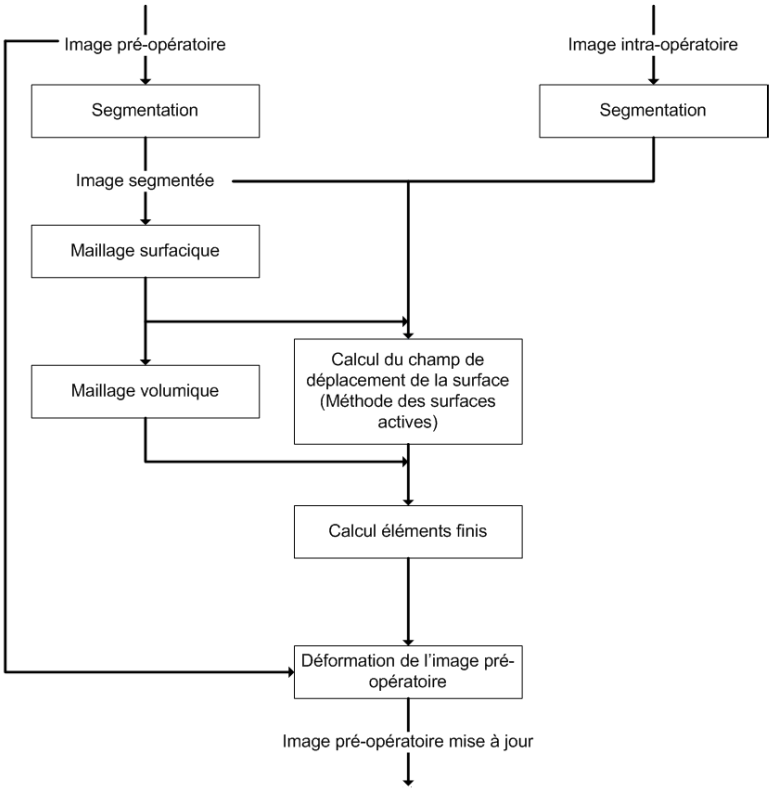
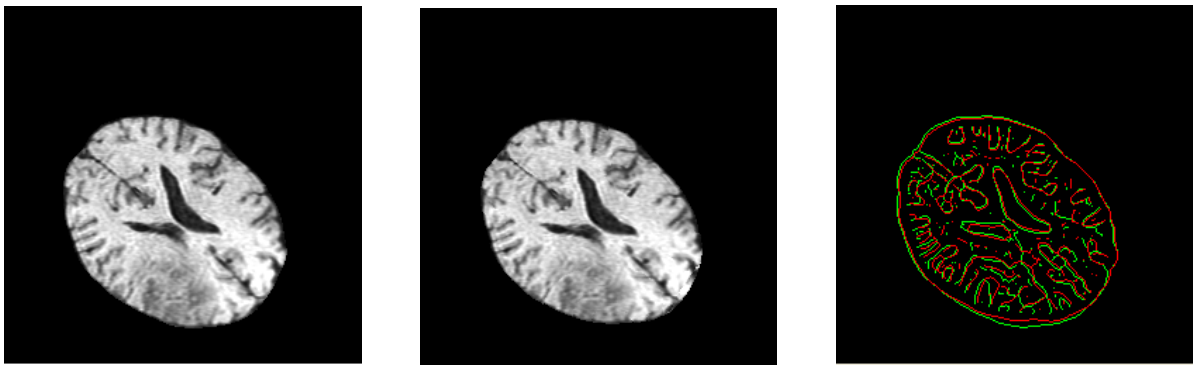


Fig. 12.3 – *Étapes de calculs nécessaires à la mise à jour de l'image pré-opératoire.*

12.2.1 Images pré- et intra-opératoires

La figure 12.4 montre les deux images initiales, acquises avant et pendant l'opération. Afin de montrer la déformation, une détection des contours a été réalisé au moyen de l'algorithme de Canny. Le résultat est présenté à la figure 12.4(c). Les contours du cerveau déformé sont en rouge et les contours correspondant au cerveau non déformé sont en vert.

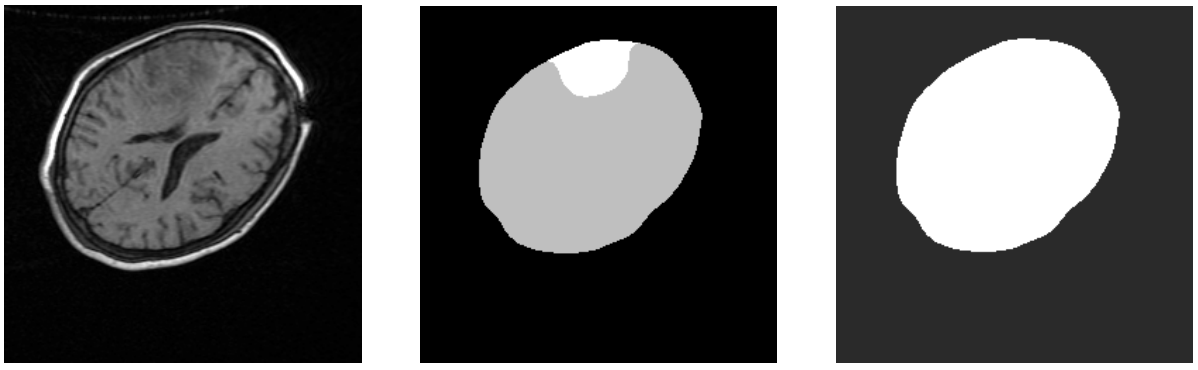


(a) Image pré-opératoire (b) Image intra-opératoire (c) Déformation subie

Fig. 12.4 – *Images pré- et intra-opératoires étudiées.*

12.2.2 Segmentation

La segmentation permet de repérer les zones d'intérêt dans les images pré- et intra-opératoires. Lors de l'enlèvement d'une tumeur par exemple, ce que nous intéresse est la position de la tumeur dans le cerveau. Prenons le cas de l'image 12.5(a). En donnant une valeur 0 à l'arrière-plan, 2 au cerveau et 4 à la tumeur, nous obtenons l'image de la figure 12.5(b). A ce jour, le mailleur surfacique créé ne permet pas encore de mailler séparément le cerveau et la tumeur, en générant un maillage qui soit compatible au niveau de leur interface. C'était d'ailleurs un des principaux inconvénients d'*Isosurf* et la raison pour laquelle l'implémentation d'un mailleur surfacique a été envisagée. En conclusion, l'image binaire utilisée dans le cadre de ce travail est celle de la figure 12.5(c).



(a) Coupe initiale

(b) Segmentation de la partie saine du cerveau et de la tumeur

(c) Segmentation de la totalité du cerveau

Fig. 12.5 – *Segmentation des images pré- et intra-opératoires.*

L'utilité de la segmentation, dans le cadre de ce projet, est double :

1. La segmentation de l'image pré-opératoire est nécessaire pour la génération du maillage surfacique.
2. La segmentation de l'image pré- et intra-opératoire est nécessaire pour la détermination du champ de déplacements à appliquer au maillage du modèle éléments finis.

12.2.3 Génération du maillage éléments finis

Génération du maillage surfacique

Le maillage surfacique est réalisé à partir de l'image pré-opératoire segmentée (fig. 12.5(c)) au moyen du mailleur surfacique créé dans ce travail. La figure 12.6 illustre le maillage obtenu.

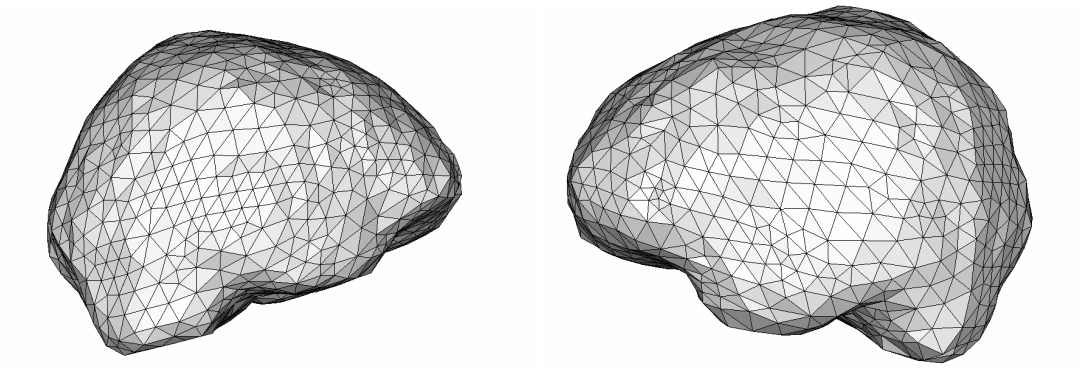


Fig. 12.6 – *Maillage surfacique du cerveau obtenu à l'aide de notre programme, à partir de l'image segmentée de la figure 12.5(c).*

Génération du maillage volumique

A partir du maillage surfacique, un maillage volumique est généré. Dans ce cas ci, le mailleur volumique qui a été utilisé est Gmsh [27]. Le maillage volumique obtenu est illustré à la figure 12.7. Ce maillage volumique est tel que chaque triangle du maillage surfacique correspond à une face d'un tétraèdre du maillage volumique.

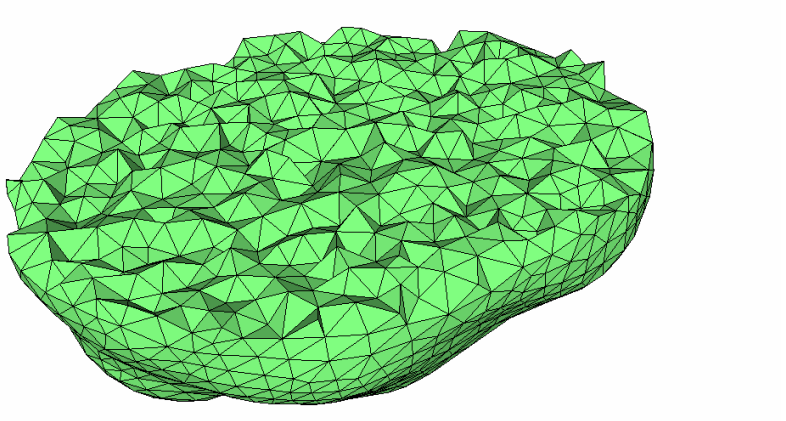


Fig. 12.7 – *Maillage volumique tétraédrique obtenu à l'aide du logiciel Gmsh.*

12.2.4 Détermination du champ de déplacement initial

le champ de déplacement initial, à appliquer dans le modèle éléments finis, est déduit d'une paire d'images pré- et intra-opératoires en observant le déplacement de la frontière du cerveau, le cortex cérébral (fig. 12.8).

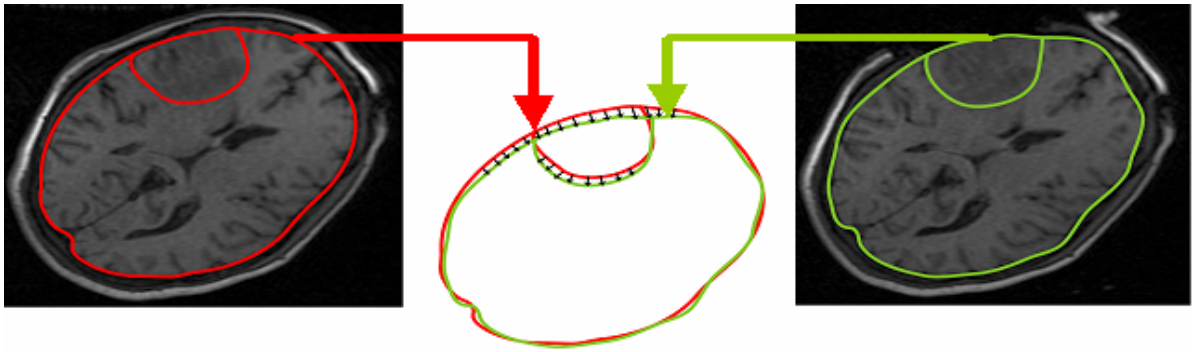


Fig. 12.8 – *Déduction des déplacements des frontières du modèle biomécanique (géométrie avant- en rouge- et après - en vert- ouverture de la boîte cranienne). Images fournies par le Brigham Women's Hospital (Boston USA).*

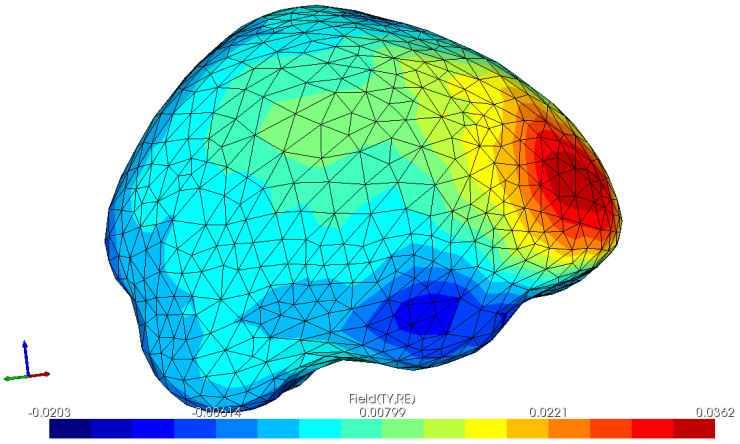
Le champ de déplacement initial surfacique est obtenu par la méthode des surfaces actives[21, 33, 65]. Celle-ci consiste à donner une rigidité au maillage surfacique et à le déformer progressivement par la méthode des éléments finis vers le cortex de l'image intraopératoire, à l'aide de forces mises à jour itérativement en fonction de la distance du maillage surfacique à la surface cible.

12.2.5 Calcul éléments finis

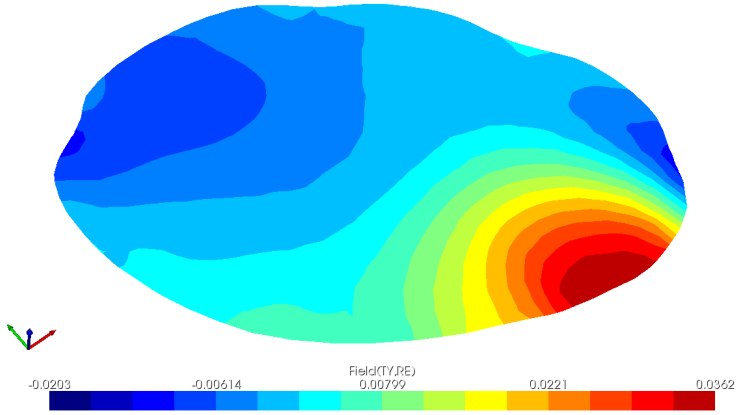
Le maillage éléments finis du cerveau permet de passer d'un champ continu de déplacements inconnus à un champ discret de déplacements inconnus en chaque noeud du maillage.

Le champ de déplacement du cortex déterminé à l'étape précédente est imposé aux noeuds de surface du maillage éléments finis. Un calcul éléments finis est ensuite réalisé dans Metafor. Le champ de déplacements fourni par le calcul est celui qui minimise l'énergie de déformation totale du cerveau, qui dépend de la qualité du maillage, du modèle de matériau utilisé et du chargement appliqué.

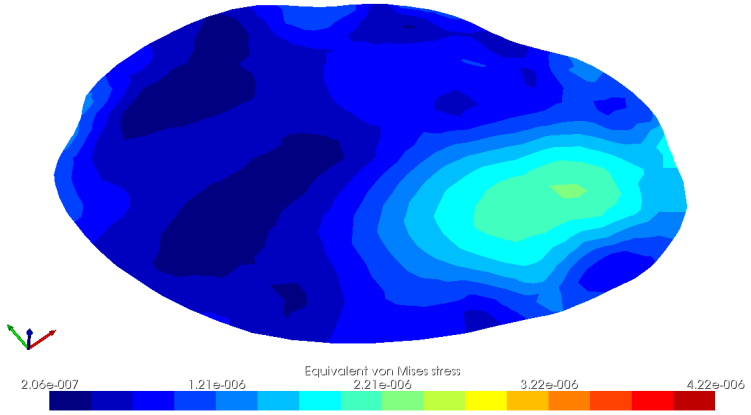
Les résultats obtenus sont présentés à la figure 12.9. Nous vérifions que le déplacement selon y correspond au déplacement entre les deux images étudiées, déplacement que nous avions montré au moyen d'une détection de contours à la figure 12.4. La figure 12.9(c) donne le champ de contraintes dans une coupe du cerveau. Nous voyons que contrairement à la simulation effectuée dans le chapitre précédent, la surface du cerveau n'est pas libre. Ceci est dû au fait qu'on impose des déplacements aux noeuds de la surface du maillage.



(a) Déplacement selon y, mm



(b) Déplacement selon y, mm (coupe)



(c) Contrainte équivalente de Von Mises, MPa (coupe)

Fig. 12.9 – *Évaluation du champ de déplacement en tout point à l'aide d'une simulation éléments finis.*

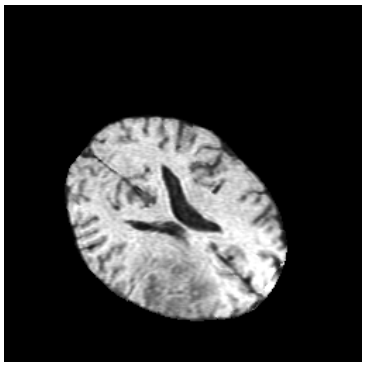
12.2.6 Déformation de l'image pré-opératoire

La dernière étape est la déformation de l'image pré-opératoire. Pour ce faire, le déplacement de chaque voxel de l'image doit être connu.

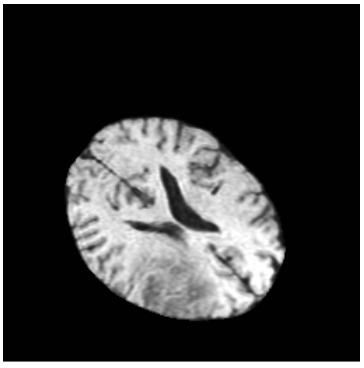
L'image pré-opératoire déformée est obtenue comme suit :

1. Une image pré-opératoire déformée vide de dimensions identique à l'image pré-opératoire est créée.
2. Cette image pré-opératoire déformée est parcourue voxel par voxel.
3. Pour chaque voxel, on détermine l'élément fini (le tétraèdre) du maillage volumique en configuration déformée auquel il appartient. Si le voxel n'appartient à aucun élément, une valeur nulle lui est attribuée et on passe au voxel suivant. Sinon :
   (a) On détermine l'élément fini (le tétraèdre) du maillage déformé auquel il appartient.
   (b) Les fonctions de forme, définies pour l'élément fini, permettent d'interpoler le déplacement des noeuds à l'intérieur de l'élément [44]. Un déplacement peut donc être associé au voxel.
   (c) On additionne le déplacement obtenu aux coordonnées du centre du voxel.
   (d) La valeur d'intensité correspondante est recherchée dans l'image pré-opératoire. Puisque le point recherché ne correspond généralement pas au centre d'un voxel d'un l'image pré-opératoire, une interpolation est réalisée.

La figure 12.10 illustre le résultat obtenu. La première image (fig. 12.10(a)) est identique à celle de la figure 12.4(b), il s'agit de l'image intra-opératoire, donnant la déformation du cerveau pendant l'opération. La second image (fig. 12.10(b)) est le résultat des opérations précédentes : c'est l'image pré-opératoire qui a été déformée pour correspondre à l'image intra-opératoire. Idéalement, ces deux images devraient donc être identiques hormis le fait que l'image pré-opératoire déformée est de même qualité que l'image pré-opératoire tandis que l'image intra-opératoire est de résolution inférieure. Pour le vérifier, les contours de deux images ont été extraits au moyen de l'algorithme de Canny (fig. 12.10(c)). L'image obtenue est à comparer avec celle de la figure 12.4(c) qui donne la différence entre l'image intra-opératoire intiale et l'image pré-opératoire. Nous constatons que l'image pré-opératoire a bel et bien été déformée et que l'image intra-opératoire déformée obtenue est proche de l'image intra-opératoire initiale.



(a) Image intra-opératoire initiale



(b) Image intra-opératoire calculée

(c) Différence entre les images (a) et (b)

Fig. 12.10 – *Résultat de la mise à jour de l'image pré-opératoire à l'aide de l'image intra-opératoire.*

Conclusion

Contributions du travail

Un mailleur surfacique a été créé. Son élaboration a été réalisée en deux étapes. Les algorithmes d'extraction de la surface, de simplification, d'amélioration de la topologie et de lissage ont été développés dans le cadre de la triangulation de surfaces implicites. Ce mailleur a enuite été étendu pour la génération de maillages surfaciques à partir d'images IRM segmentées. Afin de découpler la taille des mailles générées de la distance inter-slices, une méthode d'interpolation a été implémentée.

Ce mailleur a ensuite été utilisé pour générer des maillages surfaciques de cerveau. L'affaissement d'un cerveau soumis à la gravité a été simulé par un calcul éléments finis et le mailleur a été introduit avec succès dans une chaîne de calcul en cours de développement à l'ULg permettant la mise à jour des images neurochirurgicales préopératoires au moyen de modèles physiques déformables. Pour ces applications, nous avons considéré le cerveau comme étant un milieu homogène, formé d'un seul matériau élastique linéaire isotrope de caractéristiques moyennes $E = 3000\,Pa$ et $\nu = 0.45$.

Le mailleur surfacique développé dans ce travail sera utilisé et amélioré au sein de l'ULg dès l'année prochaine. D'une part, dans le cadre d'un doctorat pour simuler les déformations du cerveau lors d'une intervention chirurgicale (neurochirurgie assistée par ordinateur). D'autre part, dans le cadre d'un travail de fin d'études pour la création d'un modèle éléments finis de la mâchoire. De nombreuses autres applications sont possibles.

Améliorations possibles

Nous parcourons ici les différentes étapes du programme développé, en rappelant brièvement la méthode utilisée et en donnant des perspectives d'améliorations.

L'interpolation entre coupes successives est réalisée au moyen d'une méthode d'interpolation fondée sur la forme (*shape-based interpolation*) pour laquelle nous avons utilisé les courbes splines de Catmull-Rom. La méthode du *maximal disc-guided interpolation* proposée par Treece et Prager, plus sophistiquée, permettrait d'obtenir de meilleurs résultats lorsque la forme et la position de l'objet est très différente d'une coupe à l'autre.

L'extraction de la surface est réalisée par la méthode Marching Tetrahedra. L'algorithme fournit de bons résultats, quelque soit l'image tridimensionnelle segmentée initiale. La méthode du Vertex Clustering est ensuite utilisée pour réduire le nombre de mailles générées, trop important pour permettre un calcul éléments finis. A ce jour, ces deux algorithmes se succèdent mais une introduction de Vertex Clustering dans l'algorithme du Marching Tetrahedra est possible. Ceci permettrait de simplifier le maillage pendant de sa création pour générer directement un maillage comportant un nombre réduit de mailles.

La méthode du Vertex Clustering a été choisie pour sa simplicité et sa rapidité. Un inconvénient majeur de la méthode est qu'elle peut produire des triangles dégénérés ou soudés lorsque la surface est trop irrégulière. Ce problème a été rencontré lors de la génération de maillages à partir d'images tridimensionnelles non lissées. Il peut être résolu en lissant au préalable les images. Si ce lissage, entraînant nécessairement des pertes d'informations, n'est pas désiré, une autre méthode de simplification de maillages devra être utilisée.

Le qualité du maillage issu du Vertex Clustering est ensuite améliorée en deux étapes. La première consiste à corriger la topologie du maillage, en supprimant les noeuds qui ne possèdent que 3 ou 4 triangles adjacents. En effet, ces noeuds entraînent des distortions dans le maillage qui ne sont pas corrigibles par un simple lissage. La dernière étape consiste à optimiser la position des noeuds du maillage, c'est à dire, à lisser le maillage tout en gardant les noeuds sur la surface. Ces deux algorithmes fonctionnent bien en général et le maillage obtenu est de qualité suffisante pour permettre un calcul éléments finis. Une méthode de *edge swapping* pourrait être utilisée pour améliorer encore la qualité du maillage. Cette méthode consiste à inverser la diagonale d'un quadrilatère si un certain nombre de critères sont vérifiés (par exemple, si les deux nouveaux triangles sont de meilleure qualité).

Enfin, le code python devrait être retranscrit en C++ afin d'accroître la vitesse d'exécution du programme.

Bibliographie

[1] Samir Akkouche and Eric Galin. Adaptive implicit surface polygonization using marching triangles. *Computer Graphics forum*, 20(2) :67–80, 2001.

[2] Maria-Elena Algorri and Francis Schmitt. Mesh simplification. *Eurographics*, 15(3) :C–77 – C–86, 1996.

[3] H.G. Barrow, J.M. Tenenbaum, R.C. Bolles *et al.* Parametric correspondance and chamfer matching : two techniques for image matching. In *In Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 659–663, 1977.

[4] Jules Bloomenthal. *Skeletal Design of Natural Forms*. Thèse de Doctorat, University of Calgari, 1995. http ://www.unchainedgeometry.com/jbloom/dissertation.html.

[5] Gunilla Borgefors. Distance transformation in arbitrary dimensions. *Computer Vision, Graphics, and Image Processing*, 27 :321–145, 1984.

[6] Gunilla Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34 :344–371, 1986.

[7] Gunilla Borgefors. On digital distance transforms in three dimensions. *Computer vision and image understanding*, 64(3) :368–376, 1996.

[8] Gunilla Borgefors and Ingela Nyström. Efficient shape representation by minimizing the set of centers of maximal discs/spheres. *Pattern Recognition Letters*, 18 :465–472, 1997.

[9] Hamish Carr, Thomas Theußl, and Torsten Möller. Isosurfaces on optimal regular samples. *IEEE TCVG Symposium on Visualization*, 2003.

[10] Edwin Catmull and Raphael Rom. A class of local interpolating splines. *Computer aided geometric design*, pages 317–326, 1974.

[11] T. Chen, L. Serra, and H. Ng. Surface extraction : dividing voxels. *International Congress Series*, 1268 :225–230, 2004.

[12] P. Cignoni, P. Marino, C. Montani *et al.* Speeding up isosurface extraction using interval trees. *IEEE Transactions on Visualization and Computer Graphics*, 3(2) :158–170, 1997.

[13] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers and Graphics*, 22(1) :37–54, 1997.

[14] Shi Hua Cui and Jie Liu. Simplified patterns for extracting the isosurfaces of solid objects. *Image Vision Comput.*, 26(2) :174–186, 2007.

[15] Olivier Cuisenaire. *Distance transformations : Fast algorithms and applications to medical image processing.* Thèse de Doctorat, Université catholique de Louvain, Octobre 1999.

[16] P.E. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14 :227–248, 1980.

[17] Bruno Rodrigues de Araujo and Joaquim Armando Pires Jorge. Adaptive polygonization of implicit surfaces. *Computers and Graphics*, 29 :686–696, 2005.

[18] K.S. Delibasis, G.K. Matsopoulos, N.A. Mouravliansky *et al.* A novel and efficient implementation of the marching cubes algorithm. *Computerized Medical Imaging and Graphics*, 25 :343–352, 2001.

[19] J.M. Escobar, G. Montero, R. Montenegro *et al.* An algebraic method for smoothing surface triangulations on a local parmetric space. *International Journal For Numerical Methods in Engineering*, 66 :740–760, 2006.

[20] Matthieu Ferrant. *Physics-based Deformable Modeling of Volumes and Surfaces for Medical Image Registration.* Thèse de Doctorat, Université Catholique de Louvain, Belgique, 2001. http ://www.unchainedgeometry.com/jbloom/dissertation.html.

[21] Matthieu Ferrant, Simon K. Warfield, Arya Nabavi *et al.* Registration of 3d intraoperative MR images of the brain using a finite element biomechanical model. In *MICCAI*, pages 19–28, 2000.

[22] Peter Fleischmann. *Mesh Generation for Technology CAD in Three Dimensions.* Thèse de Doctorat, Université de Vienne, Autriche, 1999.

[23] Pascal J. Frey and Houman Borouchaki. Critères géométriques pour l'évaluation des triangulations de surfaces. Rapport technique, INRIA, Juillet 1996.

[24] Pascal J. Frey and Houman Borouchaki. Texel : triangulation de surfaces implicites. partie i : aspects théoriques. Rapport technique, INRIA, Décembre 1996.

[25] Pascal J. Frey and Houman Borouchaki. Texel : triangulation de surfaces implicites. partie ii : exemples d'application. Rapport technique, INRIA, Février 1997.

[26] R. Garimella, M. Shashkov, and P. Knupp. Optimization of surface mesh quality using local parameterization. In *In Proceedings of the Eleventh International Meshing Roundtable*, pages 41–52, September 2002.

[27] Christophe Geuzaine and Jean-François Remacle. Gmsh : a finite element mesh generator with built-in pre- and post-processing facilities, reference manual, 2008.

[28] Nicolas Grosdenier and Guillaume Forbin. Visualisation scientifique : Algorithmes du marching cube et du marching tetrahedra. http ://perso.orange.fr/kohonen/Marching/indexn.htm.

[29] André Guéziec and Robert Hummel. Exploiting triangulated surface extraction using tetrahedral decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 1(4) :328–342, 1995.

[30] Paul S. Heckbert and Michael Garland. Survey of polygonal surface simplification algorithms. Rapport technique, Carnegie Mellon University, May 1997.

[31] Gabor T. Herman, Jingsheng Zheng, and Carolyn A.Bucholtz. Shape-based interpolation. *IEEE Computer Graphics and Applications*, pages 69–79, May 1992.

[32] W.E. Higgins, C. Morice, and E.L. Ritman. Shape-based interpolation of tree-like structures in three-dimensional images. *IEEE transactions on Medical Imaging*, 12(3) :439–450, 1993.

[33] Yala Kaluma. Développement de modèles biomécaniques du cerveau adaptés au recalage non rigide d'images en neurochirurgie. Travail de fin d'études, Université de Liège, Belgique, 2005.

[34] Alan D. Kalvin and Russell H. Taylor. Supersurfaces : polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Applications*, 16(3), May 1996.

[35] Patrick M. Knupp. Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities. part i - a framework for surface mesh optimization. *International Journal For Numerical Methods in Engineering*, 48 :401–420, 2000.

[36] Patrick M. Knupp. Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities. part ii - a framework for volume mesh optimization and the condition number of the jacobian matrix. *International Journal For Numerical Methods in Engineering*, 48 :1165–1185, 2000.

[37] Chin-Feng Lin, Don-Lin Yang, and Yeh-Ching Chung. A marching voxels method for surface rendering of volume data. In *Computer Graphics International 2001. Proceedings*, pages 306–313, March - June 2001.

[38] Ligang Liu, Chiew-Lan Tai, Zhongping Ji *et al.* Non-iterative approach for global mesh optimization. *Computer-aided design*, 39 :772–789, 2007.

[39] Kok-Lim Low and Tiow-Seng Tan. Model simplification using vertex-clustering. In *SI3D '97 : Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 75–ff., New York, NY, USA, 1997. ACM.

[40] Sergey V. Matveyev. Approximation of isosurface in the marching cube : ambiguity problem. In *VIS '94 : Proceedings of the conference on Visualization '94*, pages 288–292, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[41] U. Montanari. A method for obtaining skeletons using a quasi-euclidean distance. *J. Assoc. Comp. Mach.*, 15(4) :600–624, 1968.

[42] Jan Patera and Václav Skala. A comparison of fundamental methods for iso surface extraction. *MG&V*, 13(4) :329–343, 2004.

[43] Jan Patera and Václav Skala. Centered cubic lattice method comparison. *Proceedings of Algoritmy*, pages 1–10, 2005.

[44] JP Ponthot. Méthode des éléments finis. Notes de cours.

[45] J.P. Ponthot. *Traitement unifié de la Mécanique des Milieux Continus solides en grandes transformations par la méthode des éléments finis.* Thèse de Doctorat, Université de Liège, 1995.

[46] S. Raya and J. Udupa. Shape-based interpolation of multi-dimensional objects. *IEEE transactions on Medical Imaging*, 9(1) :33–42, March 1990.

[47] Rémi Ronfard and Jarek Rossignac. Full-range approximation of triangulated polyhedra. *Eurographics Association*, 15(3) :C67–C76, 1996.

[48] Azriel Rosenfeld and John L. Pfaltz. Sequential operations in digital picture processing. *J. Assoc. Comp. Mach.*, 13(4) :471–494, 1966.

[49] J. Rossignac. Simplification and compression of 3d scenes. tutorial Eurographics '96.

[50] J. Rossignac and P. Borrel. Multi-resolution3d approximation for rendering complex scenes. *Geometric Modeling in Computer Graphics*, pages 455–465, 1993. Springer Verlag.

[51] T. Saito and J.I. Toriwaki. New algorithms for euclidean distance transformations of an n-dimessional digitised picture with applications. *Pattern Recognition Letters*, 27(11) :1551–1565, 1994.

[52] Will Schroeder, Kenneth M. Martin, and William E. Lorensen. *The visualization toolkit (2nd ed.) : an object-oriented approach to 3D graphics.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998.

[53] Irina B. Semenova, Vladimir V. Savchenko, and Ichiro Hagiwara. Improvement of triangular and quadrilateral surface meshes. In *Proceedings 14th International Conference on Computer Graphics and Vision (Graphicon 2004)*, pages 79–87, Sept. 2004.

[54] Hang Si. Tetgen : A quality tetrahedral mesh generator and three-dimensional delaunay triangulator, January 2006. Manuel de l'utilisateur.

[55] Chan S.L. and Purisima E.O. A new tetrahedral tesselation scheme for isosurface generation. *Computers and Graphics*, 22 :83–90(8), 25 February 1998.

[56] Douglas Summers Stay. Mathematical computer graphics : Introduction. http ://iat.ubalt.edu/summers/math/platsol.htm.

[57] Stina Svensson and Gunilla Borgefors. Distance transforms in 3d using four different weights. *Pattern Recognition Letters*, 23 :1407–1418, 2002.

[58] Stina Svensson and Gunilla Borgefors. Weighted distance transforms for volume images digitized in elongated voxel grids. *Pattern Recognition Letters*, 25 :571–580, 2004.

[59] Tomokazu Takahashi and Tatsuhiro Yonekura. Isosurface construction from a data set sampled on a face-centered-cubic lattice. In *Proc. of ICCVG*, 2, pages 754–763, September 2002.

[60] G. M. Treece, R. W. Prager, A. H. Gee *et al.* Surface interpolation for sparse cross sections using region correspondence. In *Medical Image Understanding and Analysis 1999*, 1999.

[61] G.M. Treece, R.W. Prager, and A.H. Gee. Regularised marching tetrahedra : improved iso-surface extraction. Rapport technique, University of Cambridge, September 1998. ftp ://svr-ftp.eng.cam.ac.uk/pub/reports/treece_tr333.pdf.

[62] G.M. Treece, R.W. Prager, A.H. Gee *et al.* Fast surface and volume estimation from non-parallel cross-sections, for freehand 3-d ultrasound. *Medical Image Analysis*, 3(2) :141–173, 1999.

[63] Paval Vachal, Rao V. Garimella, and Mikhail J. Shaskkov. Untangling of 2d meshes in ale simulations. *Journal of computational physics*, 196 :627–644, 2004.

[64] J. G. Verly, L. M. Vigneron, N. Petitjean *et al.* Human-visual-system-based fusion of multimodality 3D neuroimagery using brain-shift-compensating finite-element-based deformable models. In *Medical Imaging 2003 : Visualization, Image-Guided Procedures, and Display. Edited by Galloway, Robert L., Jr. Proceedings of the SPIE, Volume 5029, pp. 338-349 (2003).*, 2003.

[65] Lara Vigneron. *Mise à jour des images neurochirurgicales pré-opératoires du cerveau par un modèle physique déformable.* Thèse de Doctorat, Université de Liège, Belgique, 2003.

[66] Wikipedia. Encyclopédie libre. http ://fr.wikipedia.org.