Réparation et optimisation de maillages 3D pour l'impression 3D

Célestin Lanterne

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

par **Célestin Lanterne**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Réparation et optimisation de maillages 3D pour l'impression 3D

Date de soutenance : 19 Décembre 2019

Devant la commission d'examen composée de :

Bertrand KERAUTRET ...	Professeur, Université Lumière Lyon 2	Rapporteur
Pascal BALLET	Maître de conférences HDR, Université de Bretagne Occidentale	Rapporteur
Jean-Philippe DOMENGER	Professeur, Université de Bordeaux	Président
Nicolas PARISEY	Ingénieur de Recherche, INRA	Examinateur
Pascal DESBARATS	Professeur, Université de Bordeaux	Directeur
Stefka GUEORGUIEVA ...	Maître de conférences, Université de Bordeaux	Co-encadrante

– 2019 –

Résumé Les imprimantes 3D utilisent des modèles 3D sous la forme de maillages pour définir la géométrie et l'apparence des objets à imprimer. Un maillage 3D doit posséder certaines propriétés topologiques pour que la géométrie qu'il représente soit imprimable, et la géométrie elle même doit respecter certaines conditions pour être imprimable. Ces propriétés et conditions peuvent varier selon la technologie d'impression 3D utilisée.

De nombreux maillages 3D utilisés pour l'impression n'ont dans un premier temps pas été conçus pour cette application. La principale utilisation première de ces maillages est la visualisation, qui ne nécessite pas les mêmes propriétés topologiques et conditions géométriques. Le sujet de cette thèse est la réparation de ces maillages afin de les rendre imprimables.

Une chaîne de réparation comprenant plusieurs étapes a été conçue dans ce but. Les conditions de non-variété sont réparées en réalisant une extraction de composantes connexes (surfaces). Les bords des surfaces sont détectés et classés en fonction de la meilleure réparation à appliquer sur chaque. Les bords des surfaces sont réparés suivant leurs classement soit par une méthode de remplissage soit par une méthode d'épaississement. La fragilité de la géométrie est détectée et contrôlée.

Title Repair and optimization of 3D meshes for 3D printing

Abstract 3D printers use 3D models in the form of meshes to define the geometry and the appearance of objects to be printed. A 3D mesh must have some topological properties so that the geometry it represents could be printable and the geometry itself must respect certain conditions to be printable. These properties and conditions may vary depending on the 3D printing technologies in use.

Many 3D meshes used for printing were not initially designed for this purpose application. The main primary use of these meshes is visualization, which does not require the same topological properties and geometric conditions. The subject of this thesis is the repair of these meshes to make them printable.

A repair chain including several steps was designed for this purpose. Non-manifold conditions are repaired by extracting related components (surfaces). The boundaries of surfaces are detected and classified according to the best repair to be applied on each. The boundaries of surfaces are repaired according to their classification either by a filling method or by an offset method. The weakness of the geometry is detected and controlled.

Keywords Categories and Subject Descriptors according to ACM CCS:
I.3: COMPUTER GRAPHICS
I.3.5: Computational Geometry and Object Modeling
Polyhedral surfaces, 3D mesh repair, 3D printing

Mots-clés Informatique Graphique
Géométrie Algorithmique et Modélisation Géométrique
Surfaces polyédriques, Réparation de maillages 3D, Impression 3D

Laboratoire d'accueil LaBRI, Bât A30, Université de Bordeaux, 451 cours de la libération, 33405 Talence Cedex

Remerciements

Je voudrais remercier en premier Pascal DESBARATS mon directeur de thèse, pour les nombreux cafés bienvenus après mes nuits à travailler afin d'être "prêt" pour nos réunions du matin. Merci aussi pour ses conseils de lecture de fantasy et de science-fiction qui vont m'occuper dans cette période post-thèse.

Je voudrais remercier également Stefka GUEORGUIEVA ma co-encadrante, pour m'avoir souvent sortie la tête de mon code pour la ramener sur un plan plus théorique.

Et surtout un merci commun à tous les deux pour votre investissement dans ma thèse et pour avoir continué de m'accompagner pendant les années supplémentaires qu'elle a duré.

Merci à Bertrand KERAUTRET et Pascal BALLET pour avoir accepté d'être les rapporteurs de ma thèse, pour l'avoir lu attentivement et pour vos retours.

Merci aussi aux deux autres membres du jury de ma soutenance Jean-Philippe DOMENGER et Nicolas PARISEY, pour m'avoir écouté et pour vos questions pertinentes.

Je voudrais remercier Florent PITOUN et Matthieu SAINT DENIS les dirigeants de FabZat, pour m'avoir proposé de faire cette thèse alors que mon stage de fin d'études venait à peine de commencer. Merci aussi de m'avoir laissé une grande liberté dans mon travail de recherche et pour l'avoir financé.

Merci à David CHEVALIER dont nos nombreuses discussions ont toujours amené de nouvelles idées. Et merci pour avoir été à mes côtés un soir dans Bordeaux pendant 42km.

Merci à Kevin MAYS, Vincent DANDREAU, toutes les personnes et tous les stagiaires qui sont passés par FabZat pendant ces 5 années, pour la super ambiance qu'il y a toujours eu.

Enfin je voudrais aussi remercier mes parents Odile ERHARD et Philippe LANTERNE, pour votre soutient et vos encouragements pendant toutes ces années d'études. Merci aussi pour tous les moments de pause passés à vos côtés qui m'ont permis de me ressourcer et de me changer les idées.

Merci à mon coach Robert VERGNIEUX, pour nos rendez-vous hebdomadaires pendant cette dernière année de rédaction qui m'ont permis de garder un rythme de travail. Merci aussi pour tous tes précieux conseils et tes encouragements.

Merci à ma grande sœur Adeline LANTERNE, pour avoir partagé son expérience de sa propre thèse et pour m'avoir toujours dit quel que soit mon problème que c'était normal. Merci aussi pour tous tes messages d'encouragement que je recevais au réveil.

Merci à ma petite sœur Eloïse LANTERNE, pour m'avoir montré depuis son adolescence ce que voulait dire travailler studieusement ce qui m'a motivé à en faire un peu plus les journées moins productives. Merci aussi pour tous tes appels d'encouragement durant tes trajets avant et après tes gardes à la maternité.

Un dernier merci à toute ma famille et tous mes amis dont plusieurs ont déjà été cités, pour m'avoir permis de profiter de vos appels et de vos messages afin de réorganiser mes pensées et de faire mon planning des prochains jours.

Table des matières

Introduction

Ces dernières années, l'impression 3D a révolutionné le monde de l'industrie en permettant la fabrication d'objets de formes et de matériaux multiples en en baissant les coûts de production, en particulier dans les domaines du médical et du spatial.

Il est à noter que le terme d'impression 3D s'est généralisé pour regrouper toutes les technologies de fabrication additive c'est à dire la production d'objet par agglomération ou fusion de matière.

La fabrication additive a dans un premier temps été utilisée pour faire du prototypage rapide. Un des avantages de cette technique de fabrication est la possibilité de fabriquer des objets de formes voire de matières différentes à chaque utilisation, sans modification de la machine de production. Cette polyvalence permet de tester plusieurs formes possibles d'un objet à concevoir avant de le produire.

Les technologies de fabrication additive se sont aujourd'hui améliorées en termes de rapidité de production et de précision. Ces améliorations ont permis leur utilisation industrielle comme outil de production. Elles sont utiles pour produire des objets uniques ou en petite série. Elles permettent également de réaliser des formes qui seraient plus complexes à fabriquer avec d'autres procédés de fabrication.

Cette thèse a été réalisée en convention industrielle de formation par la recherche (CIFRE) avec l'entreprise FabZat. FabZat est une entreprise de production et de vente d'objets imprimés en 3D à partir d'un modèle 3D fourni par le client.

Ces modèles proviennent soit d'une boutique en ligne située à l'intérieur de jeux vidéo sous forme de *plug-in* soit directement des clients (demande de prototypage, maquette d'architecture, modèles issus de scanner 3D,...). Les modèles de cette boutique en ligne sont issus d'éléments 3D d'un jeu vidéo (des personnages, des véhicules, des animaux,...). Les modèles que FabZat reçoit ont donc des origines diverses (jeux vidéo, scanner 3D, logiciel CAD,...) et n'ont pas été spécialement conçus pour l'impression 3D, mais souvent pour être seulement visualisés.

Par exemple la cape d'un personnage est souvent modélisée par une surface sans volume, alors qu'un modèle 3D imprimable doit définir un volume.

De plus, le volume défini doit être adapté à notre monde physique. En effet, si le modèle 3D comprend une partie trop fragile, celui-ci risque de casser à l'impression.

Il existe des solutions partielles pour réparer ces modèles afin de les rendre imprimables, par exemple des solutions pour remplir les trous d'une surface, mais il n'existe pas de solution complète répondant à l'ensemble de nos problèmes. Pour palier ce manque, nous proposons une chaîne de réparation semi-automatique complète qui traite ces problèmes dans un ordre permettant à chaque étape de travailler sur un modèle ayant des caractéristiques topologiques et géométriques garanties par les étapes précédentes.

La thèse est présentée en trois parties. La première partie, **Domaine et problématique**, présente les problèmes des modèles non-imprimable (chap.1), l'état de l'art des réparation de ces problèmes et les résultats de logiciels de réparations existants sur dix modèles de test (Chap.2).

La notion de variété topologique est souvent utilisée dans cette thèse et plus particulièrement la notion de 2D-Variété. Une 2D-Variété est une surface dans E^3 qui en chaque point a un voisinage homéomorphe à E^2.

La deuxième partie, **Chaîne de réparation**, présente la chaîne de réparation (Chap.3) et chacune de ses étapes : l'import du modèle et la réparation des conditions de non-variétés (Chap.4), la détection et la classification des contours (Chap.5), la réparation par remplissage (Chap.6), la réparation par épaississement (Chap.7) et la détection de la fragilité (Chap.8).

La troisième partie, **Implémentation et résultats**, présente le logiciel qui implémente la chaine de réparation (Chap.9) et discute des résultats du logiciel sur dix modèles de test (Chap.10).

Enfin, en annexe sont présente une présentation des différentes technologies d'impression 3D (Annexe.A) et la documentation du code du logiciel (Annexe.B).

Il est à noter que la méthode de détection de fragilité sur un modèle avant impression a été publiée sous forme d'un article dans les actes de la conférence internationale "Computer Graphics, Visualization, Computer Vision and Image Processing" (CGVCVIP) (Lanterne *et al.* [2016]).

De plus, cette chaîne complète de réparation à été implémentée sous la forme d'un logiciel pour l'entreprise FabZat. Ce logiciel a été utilisé dans la chaîne de production pour aider à réparer de nombreux modèles 3D. Il est aussi intégré dans le *plug-in* de la boutique en ligne, principalement pour prévenir des risques de fragilité.

Première partie

Domaine et problématique

Chapitre 1

Domaine et problématique

1.1 Contexte industriel

Il y a plusieurs moyens de produire un objet physique. Les moyens utilisés aujourd'hui dans les chaînes de production peuvent être regroupés en trois grandes familles : la première famille comprend la fabrication par moulage où à partir du moule de l'objet, on introduit une matière qui va en prendre la forme et ainsi constituer l'objet. La deuxième famille est basée sur la fabrication par découpage où l'on part d'un bloc de matière dans lequel on va tailler l'objet. Par exemple, les machines de découpe laser et les tours d'usinage. La troisième famille est celle des technologies de fabrication additive. Elles sont basées sur le principe inverse de la découpe. Il s'agit de construire un objet en ajoutant de la matière par couches superposées.

Le cadre industriel de notre travail est défini par l'entreprise FabZat [1] qui utilise une chaîne de production appartenant à la troisième famille. FabZat vend des objets imprimés en 3D à partir d'un modèle 3D fourni par le client. Une partie de ces modèles 3D sont des éléments virtuels issus de jeux vidéo (personnages, véhicules, décors). FabZat fournit également à ces clients une boutique en ligne à intégrer dans une application mobile (jeux vidéo).

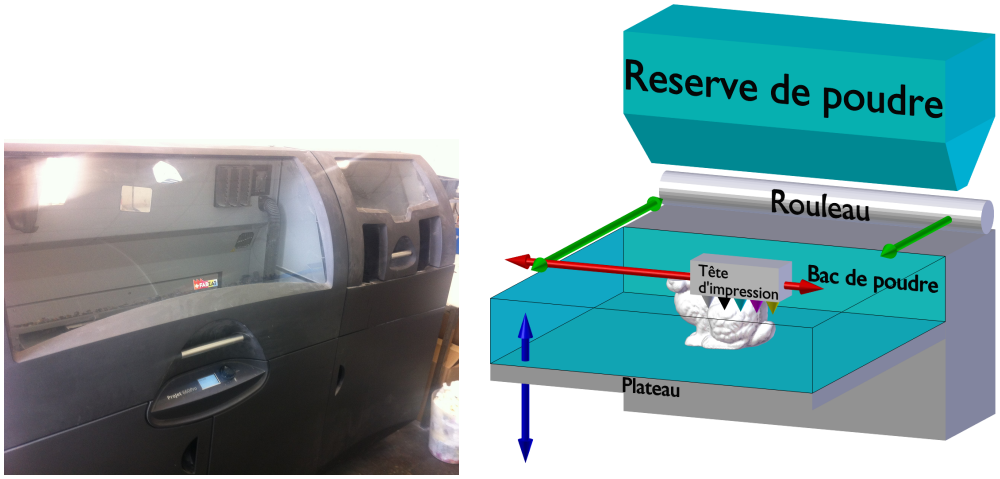
La majorité des modèles 3D reçus par FabZat, et en particulier les modèles 3D issus de jeux vidéo n'ont pas été conçus en vue d'être imprimés. La plupart ont comme but initial de servir au rendu d'une image 2D. Par conséquent ils ne respectent pas forcément les contraintes liées à une impression 3D qui est principalement de définir un volume. Mais aussi les contraintes liées aux lois de la physique une fois l'objet imprimé comme la fragilité ou l'équilibre.

La technologie exploitée par FabZat est une impression à poudre (Fig.1.1). L'agglomération de la poudre se fait par un liant liquide qui va coller les grains de poudre entre eux. Ce liant est déposé à la surface d'impression par jets de façon identique à une imprimante 2D à jet d'encre. La tête d'impression se

1. http ://www.fabzat.com/

déplace avec un mouvement de balayage sur la surface de la couche de poudre. Une fois imprimé, l'objet est sorti de la poudre puis trempé dans un liant pour le solidifier. Cette technologie permet en plus d'imprimer des objets en couleurs. Pour ce faire, on rajoute à côté de la tête d'impression déposant le liant, quatre autres têtes d'impression pour déposer des encres de couleurs, le cyan, le magenta, le jaune et le noir. Les couleurs sont déposées uniquement sur quelques millimètres le long du contour de la tranche agglomérée de l'objet. En effet, il est inutile de colorer l'intérieur de l'objet.

Pour plus d'information sur les différentes technologies d'impression 3D voir (Annexe.A).



(a) (b)

FIGURE 1.1 – Imprimante 3D à poudre : (a) Photo de l'imprimante *Zcorp 650* de FabZat, (b) Schéma de la technologie d'impression.

L'avantage principal est la possibilité d'imprimer des objets en couleurs. Par contre, il est important de s'assurer que les objets produits ne présentent pas des fragilités, car la poudre agglomérée est cassante.

L'objectif principal de la thèse est de corriger les modèles 3D, quelsqu'ils soient, pour les rendre imprimables sans trop dénaturer le modèle original à l'aide d'une série d'algorithmes et de méthodes. Un modèle 3D imprimable est un modèle 3D dont le maillage définit un volume (problème structurel) et dont la géométrie est viable une fois l'objet physique imprimé (problème de faiblesse).

1.2 Problématique

Les modèles 3D sont avant impression interprétés par un logiciel lié à l'imprimante appelé *Slicer*. Le *Slicer* prend en entrée un modèle 3D et en calcule

les différentes tranches que l'imprimante va solidifier.

Les *Slicer* sont différents d'une imprimante à une autre et peuvent réagir différemment aux erreurs des modèles 3D. Dans le cadre de la thèse, on se basera sur l'étude par expérience du *Slicer* de la *Zcorp 650* dont le code n'est pas *open source* et sur l'étude du code du *Slicer Miracle Grue* des imprimantes à fil plastique *MakerBot*.

Les problèmes d'imprimabilité des modèles numériques sont classés en deux groupes : les problèmes structurels et les problèmes de faiblesse.

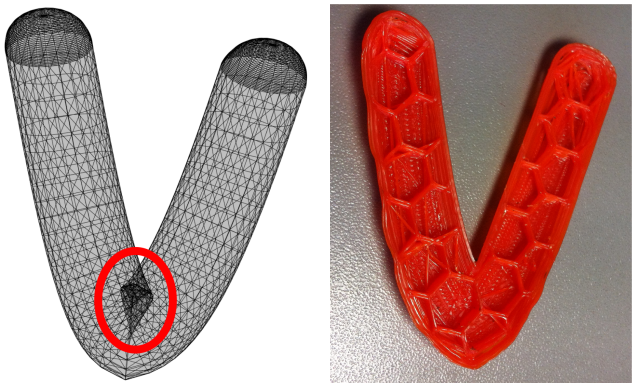
1.2.1 Problèmes structurels

Les problèmes structurels sont liés à la structure topologique du modèle. Ils comprennent :

— Éléments dégénérés : Les sommets et les arêtes isolés ne posent pas de problème au *Slicer* car celui-ci travaille uniquement sur les faces.

Les faces qui ont une aire nulle sont traitées par le *Slicer* comme les autres, le *Slicer* traite les faces une par une sans utiliser d'information de connexité entre elles. Lors du calcul d'une tranche, une telle face n'ajoutera qu'un point au contour de la zone de l'objet. Ce point étant aussi ajouté par d'autres faces, la tranche calculée est identique avec ou sans cette face.

— Intersection et auto-intersection : Elles sont traitées par le *Slicer* (Fig.1.2). Nous travaillons sur des modèles 3D composés de plusieurs volumes intersectés entre eux. Mais l'auto-intersection peut induire d'autres problèmes sur l'orientation d'une surface par exemple.

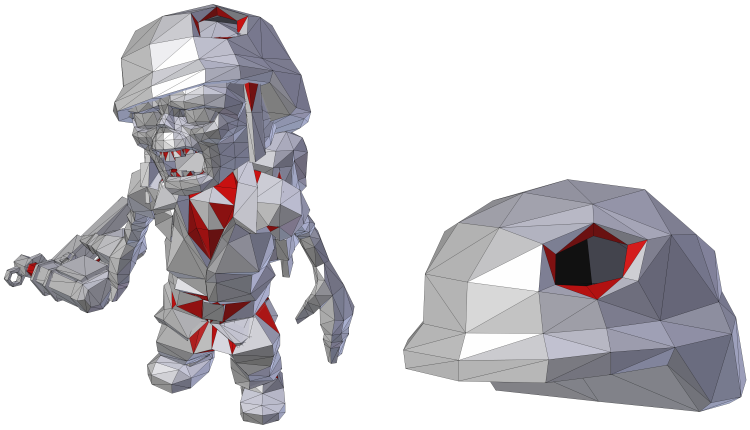


(a) (b)

FIGURE 1.2 – Modèle comprenant une auto-intersection : (a) Maillage 3D avec l'auto-intersection (entourée en Rouge), (b) Coupe du modèle imprimé, la structure interne n'est pas altérée par l'auto-intersection.

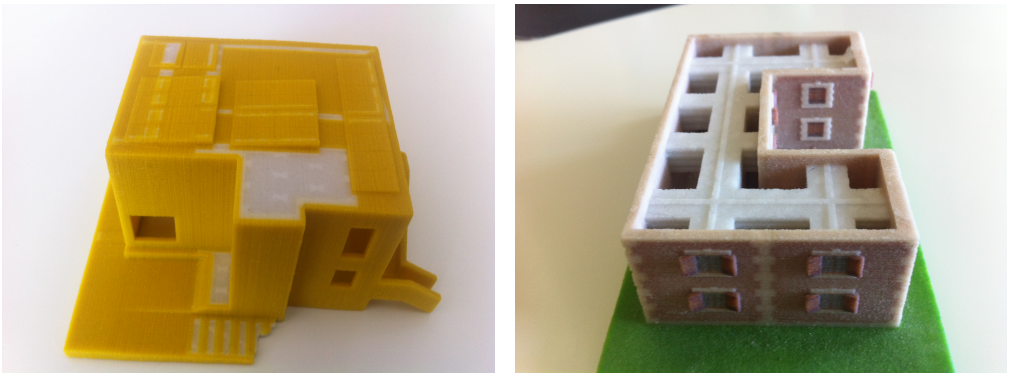
— Recouvrement d'éléments : Provoque une incertitude sur le choix de la couleur qui sera utilisée.

— Surfaces avec des bords ou trous dans la surface (Fig.1.3) : Provoque des structures internes visibles en surface de l'objet, on appellera ces structures des projections (Fig.1.4,(a)) et de l'ajout de matière si des trous ce fond face (Fig.1.4,(b)).



(a) (b)

FIGURE 1.3 – Modèle comprenant des surfaces avec des bords : (a) Modèle "Zombie", (b) Zoom sur le casque.



(a) (b)

FIGURE 1.4 – Modèles imprimés comprenant des surfaces avec des bords : (a) Modèle "Maison" avec des projections à plusieurs endroits, (b) Modèle "Châteaux" avec impression de structure non présente sur le modèle reliant les bords qui se font face.

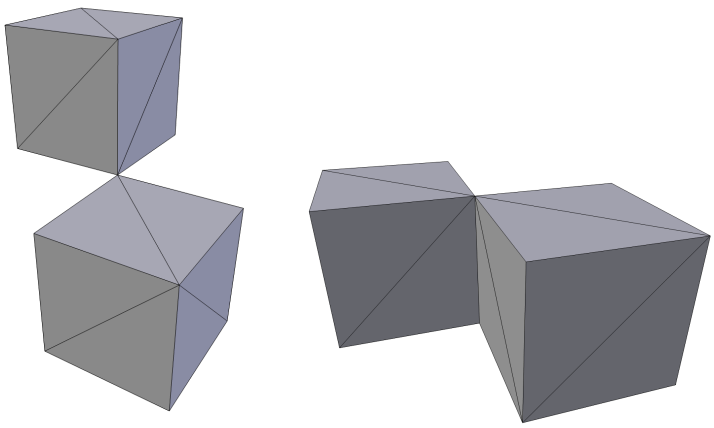
— Orientation non cohérente de la surface : Provoque les mêmes erreurs qu'une surface avec un bord.

— Orientation cohérente de la surface mais inversée : Provoque un vide et une suppression des autres volumes intersectés par ce vide (Fig.1.5).

(a) (b)

FIGURE 1.5 – Modèle imprimé comprenant une orientation cohérente mais inversée d'une de ses surfaces : Modèle "Mini-moi" imprimé avec les cheveux manquants, (a) Face, (b) Profil.

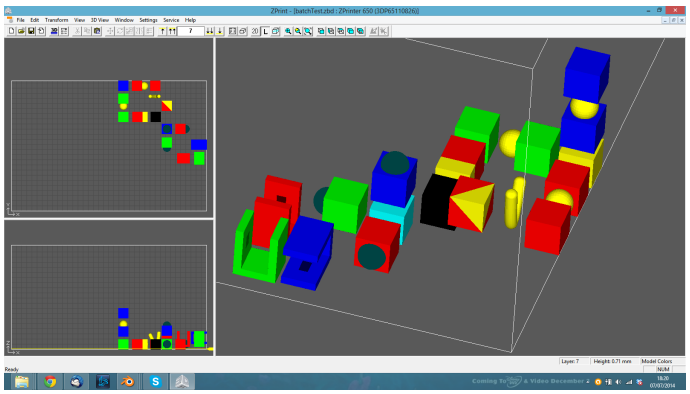
— Surface non-variété (*non-manifold*)(Fig.1.6) : Ne pose pas de problèmes au *Slicer* car celui-ci travaille sur les faces de façon indépendante et n'a pas de notion de surface. Mais une surface non-variété peut induire d'autres problèmes comme des bords ou des mauvaises orientations de normales.



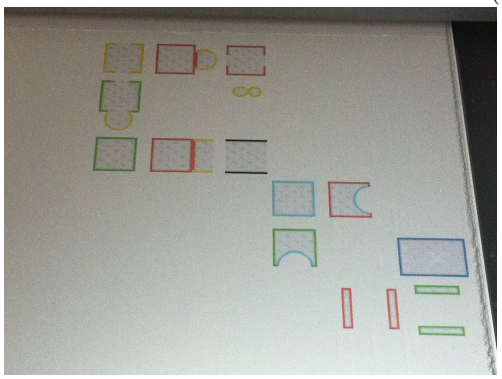
(a) (b)

FIGURE 1.6 – Modèles comprenant des conditions de non-variété : (a) sur un sommet, (b) sur une arête.

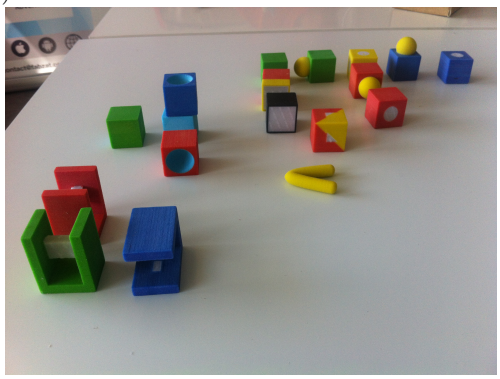
Certaines imprimantes 3D permettent d'ajouter plusieurs modèles 3D dans un *Batch* d'impression pour les imprimer en même temps (Fig.1.7,(a)). Le *Slicer* traite l'ensemble d'un coup comme un seul modèle composé de nombreux volumes intersectés ou non. Mais alors une mauvaise définition de volume (bords, orientation de normale) sur un seul modèle 3D peut provoquer des erreurs sur tous les modèles présents dans le batch. Ce qui augmente l'importance de la vérification et la correction de ces problèmes avant de lancer une impression.



(a)



(b)



(c)

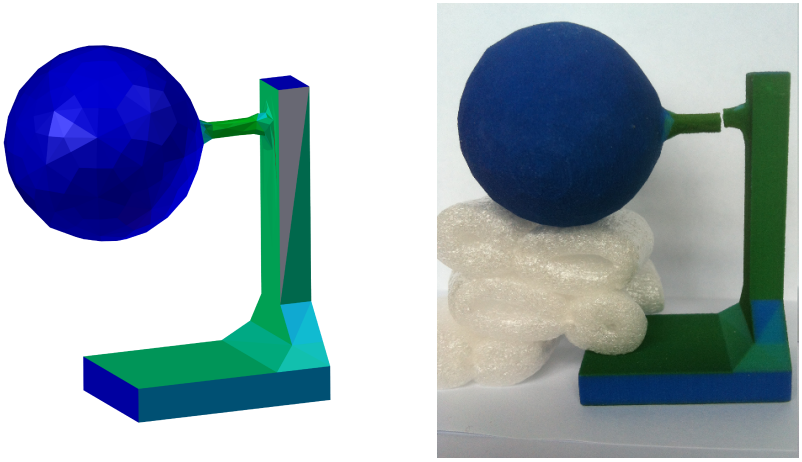
FIGURE 1.7 – Impression d'un batch contenant des modèles de test comprenant différents problèmes structurels : trous, mauvaise orientation de normale, face interne, auto-intersection. (a) Visualisation du batch, (b) Tranche durant l'impression, (c) Impression des modèles tests.

1.2.2 Problèmes de faiblesse

Les problèmes de faiblesse sont liés à la géométrie de l'objet, mais aussi aux caractéristiques de la technologie utilisée.

— Fragilité : Risque de cassure sur l'objet imprimé, qui dépend de la géométrie du modèle 3D et de la matière utilisée (Fig.1.8).

— Équilibre : L'objet n'est pas stable dans la position voulue initialement.
— Support : Les technologies utilisant un support lors de l'impression limitent les formes géométriques possibles à imprimer, pour plus de détails voir (Annexe.A).



(a) (b)

FIGURE 1.8 – Modèle comprenant un problème de faiblesse : (a) Modèle numérique "Hanging-Ball" issu de (Stava *et al.* [2012]), (b) Modèle imprimé avec une cassure due au poids de la boule.

1.3 Domaine

Notre travail s'inscrit dans le domaine de la modélisation géométrique et en particulier de la réparation de maillages 3D. Il existe un grand nombre d'études et de logiciels qui traitent cette problématique. Néanmoins, l'imprimabilité reste un sujet de recherche. En particulier nous nous intéressons à plusieurs points :

— Modèles 3D avec couleur et texture.
— Modèles 3D composés de plusieurs surfaces définissant plusieurs volumes intersectés.
— Reproductivité : Il s'agit d'assurer que l'ordre des éléments décrit dans un format 3D (OBJ,STL,OFF,..) ne fait pas varier le résultat de nos algorithmes.
— Correction des surfaces non-variété : Il s'agit de changer chaque surface du modèle 3D composé en surface variété avec ou sans bords, voir (Chap.4).
— Détection et classification des bords : Il s'agit de déterminer si ces bords sont des contours définissant des trous où des contours définissant des surfaces fines voir (Chap.5).

— Réparation de bords : Les bords définissant des trous sont comblés par l'ajout de face voir (Chap.6), les bords définissant des surfaces fines sont réparés par épaississement de surface voir (Chap.7).

— Calcul d'épaisseur : Il s'agit de calculer l'épaisseur de la géométrie d'un modèle 3D pour en détecter les fragilités, voir (Chap.8).

Chapitre 2

État de l'art

2.1 Format de fichier 3D

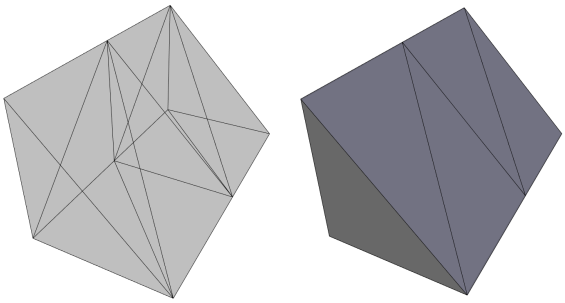
En impression 3D, les caractéristiques du modèle 3D à imprimer sont décrites dans un fichier au format 3D. Les formats de fichier 3D peuvent être propriétaires (Spécification appartenant à une entité privé) ou ouverts (Spécification publique). Ces formats décrivent :

— La géométrie : description de la forme ou du volume à imprimer, soit à l'aide d'un maillage 3D représentant une surface frontière entre l'intérieur et l'extérieur du volume (Surfacique), soit par une construction de solides simples (cube,sphère,tore,etc.) appelée CSG pour *Constructive Solid Geometry* (Volumique).

— L'apparence : couleur,texture et matériau.

Le maillage 3D d'un format surfacique est représenté par une liste de faces planaires, chacune étant définie par la séquence de ses sommets, les sommets étant stockés par leurs coordonnées (x ; y ; z), par exemple le format de fichier Wavefront OBJect (Fig.2.1).

```
# Blender v2.74 (sub 0) OBJ File: ''
# www.blender.org
mtllib Face2CC.mtl
o Face2CC-ss123-132
v -0.000000 -0.000000 0.000000
v -0.000000 2.000000 -0.000000
v 0.000000 0.000000 2.000000
v -2.000000 2.000000 -0.000000
v -2.000000 -0.000000 0.000000
v -2.000000 0.000000 2.000000
v 2.000000 -0.000000 0.000000
v 2.000000 -0.000000 2.000000
v 2.000000 2.000000 -0.000000
usemtl None
s 1
f 4 5 6
f 3 6 1
f 6 5 1
f 3 2 6
f 4 2 5
f 2 4 6
f 2 1 5
f 7 8 3
f 7 9 8
f 8 9 3
f 3 9 2
f 9 7 1
f 9 1 2
f 3 1 7
```



(a) (b) (c)

FIGURE 2.1 – Représentation du maillage (a) Format OBJ Wavefront, (b)(c) Importation et visualisation avec Blender V2.76.

Il existe une évolution continue des formats de données pour la fabrication additive. À l'heure actuelle, parmi la douzaine de formats cités en 1998 par Marsan et al. (Marsan *et al.* [1998]), seul le format STL reste populaire dans les systèmes d'impression 3D actuels.

2.1.1 Formats actuels

(D.Chakravorty [[Online][Accessed : 14-November-2019]]), présente une vue d'ensemble de ces formats, qui peuvent être résumés ainsi :

1. **AMF** (pour Additive Manufacturing File) est un format publié par le National Resource Center of Materials Technology Education, Technician Education in Additive Manufacturing (projet TEAM), avec la subvention de la National Science Foundation (États-Unis) : DUE #1003530, pour plus de détails voir (H.Lipson [2014]). AMF est un fichier XML avec une hiérarchie de cinq éléments : objet, matériau, texture, constellation et métadonnées. Introduit en tant que remplacement du format STL, le format AMF est une norme ISO depuis 2013 [1]. Il décrit les surfaces des objets avec des maillages triangulaires, permet des faces non planaires et la manipulation de surfaces incurvées.

   Avantages : Le format AMF permet de représenter la couleur et les matériaux. Il gère les treillis et les constellations, deux éléments créés pour l'impression 3D. Ce format est simple, évolutif et compatible avec le format STL.

1. https ://www.astm.org/Standards/F2915.htm

Inconvénients : Le format est peu utilisé.

2. **3MF** (pour Manufacturing Format) est un format de fichier pour la 3D apparu dans un contexte industriel. Il a été publié par le consortium 3MF [2] dont font partie Microsoft et Dassault Systemes, pour plus de détails voir (3MFConsortium [[Online][Accessed : 14-November-2019]]). 3MF est un format basé sur XML. Il dispose d'une représentation géométrique similaire au format STL (maillages triangulaires), mais bien plus compacte. Le format 3MF garantit que les surfaces décrites sont des variétés sans bord et n'ont pas de triangles superposés (*overlap*). La géométrie et l'apparence du modèle sont enregistrées dans une archive unique.

   Avantages : Le format 3MF contient les informations complètes sur le modèle dans une "archive unique". Le format est simple, ouvert et extensible.

   Inconvénients : Le format est toujours à ses débuts (non mature). L'évolution de l'accès open source est imprévisible.

3. **DAE, Collada** (pour Digital Asset Exchange, Collaborative Design Activity) est un format publié par Khronos Groupe [3], pour plus de détails voir (KhronosGroup [[Online][Accessed : 14-November-2019]]). DAE est un format neutre basé sur le langage XML conçu pour être un format d'échange entre applications 3D. Le format DAE est une norme ISO depuis 2012 [4]. Le format COLLADA prend en charge la géométrie, les propriétés liées à l'apparence comme la couleur, le matériau, la texture et l'animation.

   Avantages : Le format DAE permet de représenter la couleur et les textures. Le format est ouvert.

   Inconvénients : Les spécifications du format sont complexes.

4. **FBX** (pour FilmBoX), est le format de fichier développé par Kaydara pour ses services de capture de mouvement. Ce format est détenu par Autodesk depuis 2006, pour plus de détails voir (Autodesk [[Online][Accessed : 14-November-2019]]).

   FBX est un format propriétaire qui permet l'interopérabilité entre applications de création de contenu numérique telles que 3DS Max, Maya, MotionBuilder, Mudbox, Rhino et d'autres logiciels propriétaires ou tiers. Il supporte la géométrie (maillage, squelette, morphes), les matériaux, les textures, l'animation, les déformations et autres caractéristiques. FBX est actuellement le choix le plus populaire pour l'animation.

   Avantages : Le format FBX permet de représenter la couleur et les

2. https ://3mf.io/
3. https ://www.khronos.org/
4. https : //www.iso.org/standard/59902.html

textures. C'est un format très utilisé pour l'animation notamment dans l'industrie du jeu vidéo.

Inconvénients : Le format est propriétaire utilisé par Autodesk et sa suite logicielle.

5. **OBJ** (pour OBJect) est le format de fichier créé par Wavefront Technologies, pour plus de détails voir (FileFormatInfo [[Online][Accessed : 14-November-2019]]).

   La variante OBJ Ascii est un format neutre, tandis que la variante OBJ Binary est un format propriétaire. Les fichiers OBJ peuvent coder la géométrie d'un modèle 3D avec des informations sur la couleur, les matériaux et les textures. Les informations sur la couleur et la texture sont stockées dans un format de fichier compagnon appelé le format MTL (pour Material Template Library). Le fichier OBJ, quand jumelé avec le fichier MTL correspondant, peut générer un modèle multicolore et texturé. Le format de fichier OBJ est un format de données ouvert qui représente seulement la géométrie 3D - à savoir la position de chaque sommet, les coordonnées du sommet dans un système de coordonnées donné. Les coordonnées OBJ n'ont pas d'unités, mais des fichiers OBJ peuvent contenir des informations d'échelle dans une ligne de commentaire lisible. Comme ce format est largement utilisé par les programmes de modélisation 3D, il peut être transféré entre programmes et directement interprété par certains *Slicer* sans avoir besoin d'être converti en format STL.

   Avantages : Le format OBJ permet de représenter la couleur, les matériaux et les textures. Le format est simple et ouvert.

   Inconvénients : Le format est permissif à des erreurs topologiques (doublons de faces, faces dégénérées). C'est un format non standardisé.

6. **STL** (pour STereoLithography), est un format de fichier créé par 3D Systems (1987, Chuck Hull). Il est également connu sous le nom de Standart Tesselation Language. C'est le format 3D standard, pour plus de détails voir (Burns [[Online][Accessed : 14-November-2019]]).

   STL est un format neutre. Le fichier STL contient la liste des faces triangulaires planaires, chaque triangle étant défini par trois sommets et une orientation de la normale dirigée depuis l'intérieur vers l'extérieur de l'objet. Puisque seuls les sommets sont stockés explicitement, les arêtes et la connectivité des faces sont représentées implicitement.

   Avantages : Le format est simple, ouvert et supporté par beaucoup d'imprimante 3D.

   Inconvénients : Il ne stocke pas d'informations sur la connectivité des faces triangulaires couvrant la surface. Sans connectivité, le fichier représente essentiellement un groupe de triangles flottant dans l'espace, également appelé "soupe de polygones". La génération d'informations

topologiques à partir de telles "soupe de polygones" est un problème largement étudié. Voir par exemple (Rock et Wozny [1992]) pour un début d'information. Le format est permissif à des erreurs topologiques (doublons de faces, faces dégénérées). Les informations sur l'apparence de l'objet ne sont pas présentes. La précision vient au détriment de la taille dans le fichier.

7. **X3D** (pour eXtensible 3D), format de fichier publié par Web 3D Consortium, pour plus de détails voir (Web3D [[Online][Accessed : 14-November-2019]]). X3D est un standard ISO depuis 2005 [5]. Le format est libre d'utilisation et succède au langage VRML. X3D est un format neutre basé sur XML. Il supporte plusieurs géométries et types d'apparence (forme, matière, texture, éclairage, shaders), l'animation, l'interaction utilisateur tactile, les scripts, les méthodes de rendu de volume, la physique des corps rigides, la structure d'assemblage CAO et les métadonnées. Il est robuste aussi bien pour la modélisation et l'interprétation des données de l'objet que pour la visualisation des objets.

   Avantages : Le format X3D est très utilisé pour le contenu Web. Le format est ouvert, modulaire et extensible.

   Inconvénients : La spécification du format est complexe.

2.1.2 Conclusion

Dans le cadre de la thèse nous avons besoin d'un format 3D avec les caractéristiques suivantes :

— Gestion de la couleur et de la texture. Les technologies utilisées par FabZat imprimant des objets en couleur, l'information doit être définie dans le fichier 3D.

— Format ouvert, pour modifier un modèle 3D afin de le rendre imprimable, il faut pouvoir lire et écrire les informations du modèle 3D.

Le format OBJ remplit ces deux conditions mais en plus :

— Le format OBJ est utilisé par de nombreux logiciels de modélisation. Cela facilite son utilisation par les clients de FabZat. Il existe aussi de nombreux convertisseurs permettant de passer d'un autre format vers le format OBJ.

— Le format OBJ contient les informations de connectivité des faces.

L'ensemble de ces points nous a fait choisir le format OBJ comme format principal dans le traitement d'un modèle 3D pour le rendre imprimable.

5. http : //www.web3d.org/standards/number/19775-1

2.2 Imprimabilité

Un maillage 3D n'est pas toujours conçu comme frontière d'un objet solide (d'un volume), c'est-à-dire comme une 2D-variété sans bord, orientable et orientée, plongée dans $\mathbb{R}^3$. En pratique, un maillage 3D peut être une "soupe de polygones", un ensemble de polygones non organisés sans information de topologie explicite, avec ou sans normales, avec des défauts géométriques comme des intersections de polygones, des trous et des connectivités multiples de faces autour d'une arête.

L'imprimabilité dépend en partie de la technologie d'impression. Aucune impression n'est possible si le maillage n'enferme pas un volume fini avec des normales orientées de manière cohérente. Par contre, l'auto-intersection peut être analysée par certains *Slicers* d'imprimantes 3D qui produisent un volume englobant les faces internes dues à l'auto-intersection.

Cette partie vise à présenter les différents algorithmes de l'état de l'art, utilisés pour convertir les maillages 3D en maillages imprimables, qui ont inspirés nos recherches ainsi que notre chaîne de réparation pour rendre imprimable un maillage 3D. Ces articles ont été comparés dans un état de l'art présenté par (Attene *et al.* [2013]).

2.2.1 Conditions de non-variété

Quelques définitions sont rappelées ci-dessous (Hoffmann [1989]).

Une n-variété dans E^m, l'espace Euclidien en m-dimensions, avec $m \geq n$, est un sous-espace qui est localement homéomorphe à E^n. C'est à dire qu'en tout point il existe un voisinage qui est homéomorphe à E^n.

Une n-variété avec bord est un sous-espace dont les points de bord ont un voisinage qui est homéomorphe au demi-espace positif E^{n+} et dont les points intérieurs ont un voisinage homéomorphe à E^n.

$$E^{n+} = \{(x_1, \ldots, x_n) \in E^n | x_i \geq 0 | 1 \leq i \leq n\} \tag{2.1}$$

Un solide S est une 3D-variété, connectée avec bord, plongée dans E^3. La frontière ∂S de S est une 2D-variété, sans bord, orientable et orientée, plongée dans E^3.

Étant donné la représentation de la frontière de S, pour qu'un modèle 3D soit imprimable il faut que sa surface frontière représentée par le maillage soit une 2D-variété, sans bord, orientable et orientée, plongée dans E^3.

Guéziec et al. (Guéziec *et al.* [2001]) proposent deux opérations, la coupe (*cutting*) et la couture (*stitching*), pour supprimer les singularités topologiques des données d'entrée et convertir la surface non-variété en une surface variété. L'algorithme comprend une procédure en cinq étapes.

— En premier, les "arêtes de bord", incidentes avec une seule face, et les arêtes singulières, incidentes avec au moins trois faces ("arête condition de non-variété"), sont marquées (Fig.2.2,(a)).

— Ensuite, les sommets sur un pincement (*isolated singular vertex*) sont identifiés.

— Ensuite, l'opération de coupe est appliquée sur les arêtes et les sommets marqués (Fig.2.2,(b)).

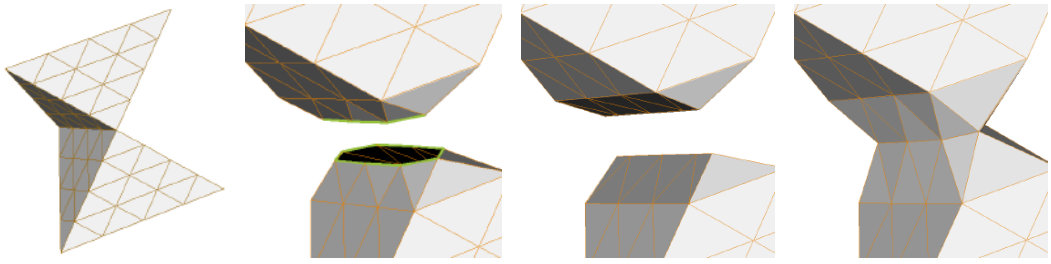
Un arbre couvrant de faces orientées de manière cohérente est construit. Deux faces adjacentes ne forment un arc de l'arbre que si une orientation cohérente peut être trouvée sur toutes les arêtes partagées, éventuellement après inversion de l'orientation de l'une des faces. L'orientation de chaque face de l'arbre correspond donc à l'orientation de la face racine.

— L'opération de coupe est effectuée le long des arêtes partagées où l'orientation est incohérente. L'arbre couvrant définit le bord d'une variété orientée.

— Enfin, les arêtes de bord crées par les coupures sont cousues par "pincement" (*pinching*)(Fig.2.2,(c)), une arête de bord issue d'une même coupe, ou par "accrochage" (*snapping*)(Fig.2.2,(d)), pour relier des arêtes de bord issues de plusieurs coupes.

Avantages : La méthode concerne les surfaces polyédriques. Elle est utiles pour la simplification et la compression.

Inconvénients : La méthode déforme en partie la surface originale.



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

FIGURE 2.2 – (a) Modèle avec des arêtes condition de non-variété, (b) Opération de coupe (*cutting*), (c) Couture par pincement (*pinching*), (d) Couture par accrochage (*snapping*). D'après Fig.1 de (Guéziec *et al.* [2001]).

Rossignac et Cardoze (Rossignac et Cardoze [1999])

En général, on peut associer à chaque modèle non-variété un ensemble de modèles variétés par duplications des sommets et des arêtes condition de non-variété, une ou plusieurs fois. Il est également possible de définir des conventions pour interpréter de telles incohérences. Rossignac et Cardoze introduisent *Matchmaker* sur la base de perturbations "infiniment petites" des sommets et

des arêtes pour produire une 2D-variété. À cette fin, les auto-intersections sont classées comme suit :

— "*Osculating contacts*" lorsque deux arêtes ou plus s'intersectent sur un point ou un sommet qui est inclus dans une face ou une arête.

— Arêtes et sommets coïncidents.

— Croisement des arêtes.

— Intersections de faces lorsque deux faces se chevauchent ou se croisent dans leur intérieur.

Pour les deux premiers types, une interprétation valide peut être définie en déplaçant certains sommets d'une quantité infiniment petite ou en pliant certaines arêtes d'une quantité infiniment petite.

Les deux autres types d'auto-intersections ne sont pas traités.

Avantages : La méthode évite la plupart des duplications de sommets inutiles, peut inclure des sommets et des arêtes coïncidentes.

Inconvénients : La méthode ne traite pas les croisement d'arêtes et les intersections de faces.

2.2.2 Trous

Un trou dans une surface est une zone sans faces entourée par un contour. Un contour étant une suite d'arêtes de bord formant une boucle. Le but est de remplir les trous pour construire une surface "étanche" (*watertight*), ce qui va corriger les arêtes de bord de la surface. La difficulté est de préserver la topologie et la géométrie là où elles existent et de construire des formes "plausibles" dans les régions non existantes.

Roth et Wibowo (Roth et Wibowoo [1997]) proposent une procédure en deux étapes :

On recherche les contours des trous, puis un plan est ajusté à travers les sommets 3D de chaque contour. Ces sommets sont projetés sur ce plan et un test d'auto-intersection du contour est effectué.

S'il n'y a pas d'auto-intersection, le contour du trou projeté est triangulé dans ce plan. Cette même séquence de triangles est ensuite utilisée pour trianguler le contour du trou 3D.

Avantages : La méthode est basée sur une triangulation 2D qui garantit une triangulation valide sur un contour 2D sans auto-intersection.

Inconvénients : Moins le contour est plan plus il y a un risque d'auto-intersection lors de la projection. Le cas arrive généralement quand le contour est grand.

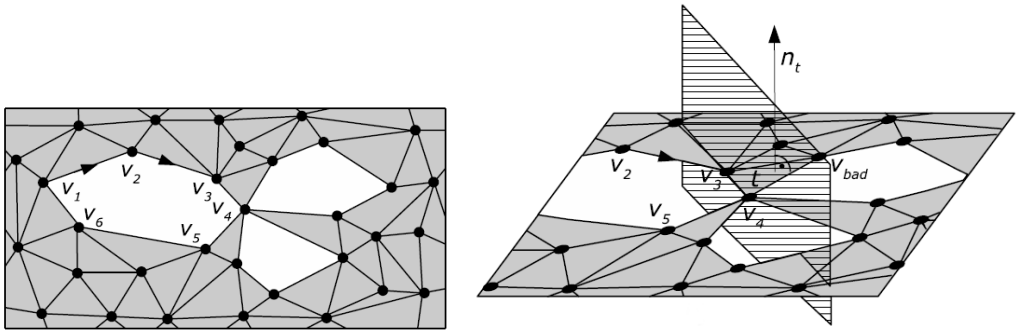
Varnuska et al. (Varnuska *et al.* [2005]) propose une méthode pour combler les petits trous, subsistants après une reconstruction de surface à partir d'un ensemble de points, dans une procédure en deux étapes :

— Le contour du trou est tracé pour être aussi petit que possible. Dans le cas où le contour arrive sur un sommet qui a plus de deux arêtes de bord adjacentes, le choix de l'arête suivante du contour dépend d'une analyse locale. Étant donné l'arête de bord actuelle et la normale de son triangle adjacent, un plan est construit séparant l'espace le long de ce bord en deux demi-espaces. L'arête suivante du contour est celle avec l'angle le plus petit qui permet au tracé de rester dans le même demi-espace (Fig.2.3).

— Les petits trous sont comblés à l'aide d'une triangulation 2D par l'algorithme *Earclipping*. Une limite heuristique de la taille du trou, (le nombre d'arêtes de délimitation), détermine quels trous sont suffisamment petits pour pouvoir être traités. Pour l'évaluation des oreilles (*Ear*), plusieurs stratégies sont expérimentées : le plus petit angle, la plus petite longueur des arêtes et le plus petit angle voisin.

Avantages : La méthode a une approche simple qui utilise une triangulation 2D.

Inconvénients : Limité aux petits trous détectés après une reconstruction de surface. Pour les ensembles de points avec du bruit, les résultats sont mauvais en raison de la mauvaise qualité de la reconstruction de la surface et des problèmes de recouvrement et d'intersection des triangles.



(a) (b)

FIGURE 2.3 – Tracé d'un contour : (a) Début du tracé du contour $V_1V_2V_3V_4$, (b) Un plan est construit en utilisant l'arête V_3V_4 et la normale n_t de la face t. Le contour se poursuit sur V_5 qui est le sommet avec l'angle le plus petit dans le même demi-espace que le reste du contour. D'après Fig.3.1 de (Varnuska *et al.* [2005]).

Bohn et Wozny (Bohn et Wozny [1992]) décrivent une méthode de fermeture de surface. L'orientation des normales et les conditions de non-variété sont également interprétées. La méthode suit une stratégie en trois étapes.

En premier lieu, les arêtes ayant une seule face adjacente, appelées "arêtes de bord" (*lamina edges*), sont identifiées. Une arête de bord existe pour chaque arête d'une face qui ne relie pas cette face avec une autre. Plusieurs arêtes de bord coïncidentes correspondent à des bords de la surface qui se chevauchent.

En deuxième étape, les arêtes de bord sont combinés en boucles fermées et non auto-intersectées, appelées "courbes de Jordan". Chaque arête de bord est considérée comme un arc directionnel. L'ensemble des arêtes de bord est transformé en un graphe directionnel en fusionnant des arcs adjacents de manière itérative. Cinq cas de fusion sont identifiés : combinaison simple, combinaison linéaire, cyclique avec rendement direct, cyclique avec plusieurs rendements directs et cyclique avec rendement indirect (Fig.2.4). Chaque cas génère un graphe de complexité variable et résout les ambiguïtés dues à la structure non-variété.

Enfin, les courbes de Jordan sont remplies pour corriger les arêtes de bord. Pour chaque courbe de Jordan un algorithme itératif est appliqué en éliminant un sommet à la fois et en ajoutant une face qui utilise le sommet éliminé et ses deux sommets voisins. Le choix du sommet à supprimer suit un algorithme heuristique : à chaque étape, le sommet qui forme le plus petit angle avec ses deux voisins est sélectionné.

Avantages : La méthode est une solution complète pour la fermeture des surfaces représentées par un maillage, assure une organisation topologique des faces en 2D-variétés.

Inconvénients : Calculer les angles entre les sommets n'est pas toujours possible, ainsi l'algorithme heuristique peut être indéterminé. Une surface non-orientable peut être créée si la surface avec bord est non-orientable.

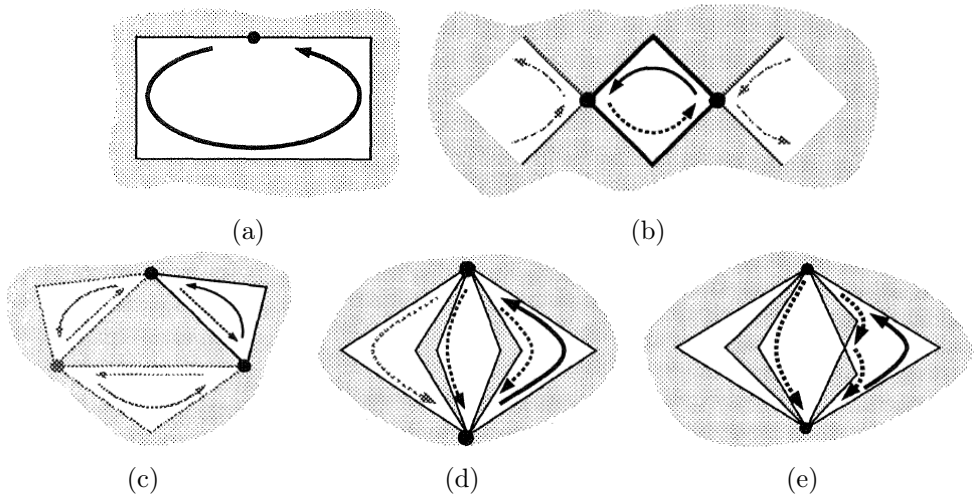


FIGURE 2.4 – Cas de fusion : (a) Combinaison simple, (b) Combinaison linéaire, (c) Cyclique avec rendement direct, (d) Cyclique avec plusieurs rendements directs, (e) Cyclique avec rendement indirect. D'après Fig.4 de (Bohn et Wozny [1992]).

2.2.3 Gaps

Gaps (fissures minces) sont des trous dans la surface mais dont l'espace entre les éléments du contour (sommet,arête de bord) est faible. Deux techniques existent pour les corriger, un remplissage (*filling*) adapté à ce type de trou, une fusion des éléments du contour (*stitching*). Un *gap* peut aussi exister entre plusieurs contours.

Patel et al. (Patel *et al.* [2005]) présentent un algorithme basé sur des opérations de contraction de paire de sommets (*stitching*) et d'expansion (*filling*) pour remplir de petits espaces / chevauchements et pour combler les grands trous (Fig.2.5).

Premièrement, les sommets et les arêtes de bord sont détectés et insérés dans un octree pour une recherche efficace. Ensuite, les sommets de bords sont associés par paire. La liste de paires est triée à l'aide d'une fonction de coût dépendant de la distance entre les sommets de chaque paire. Elle est mise à jour de manière itérative après la contraction ou l'expansion d'une paire de sommets.

Une paire de sommets est contractée si le coût de la paire est inférieur à la tolérance de fusion, sinon une expansion est effectuée. Dans les deux cas, les informations de connectivité sont mises à jour. Cette procédure est basée sur une hypothèse locale des problèmes géométriques et topologiques.

Avantages : La méthode prend en charge les topologies variété et non-variété.

Inconvénients : La méthode ne choisit pas automatiquement le seuil de fusion (une interaction de l'utilisateur est requise).

2.2.4 Orientation cohérente de la surface

Une surface 2D-variété, sans bord, orientable et orientée, plongée dans E^3 est la frontière d'un volume. L'orientation de cette surface indique de quel côté de la surface se trouve l'intérieur et l'extérieur du volume. L'orientation d'une surface est basée sur l'orientation de ses faces représentées par leur normale. La normale d'une face est orientée vers l'extérieur du volume. Une orientation cohérente d'une surface signifie que les normales de toutes ses faces sont orientées vers un même côté de la surface.

Roth et Wibowo (Roth et Wibowoo [1997]) utilisent une approche volumique pour calculer les normales lors de la construction d'un maillage triangulaire à partir d'une images 3D ou d'un ensemble de points. La grille de voxels est considérée dans les six axes de directions (+/- x, +/- y, +/- z). En regardant de l'infini vers le bas sur chaque axe dans la grille de voxels, les voxels qui sont visibles doivent avoir leurs normales orientées vers l'observa-

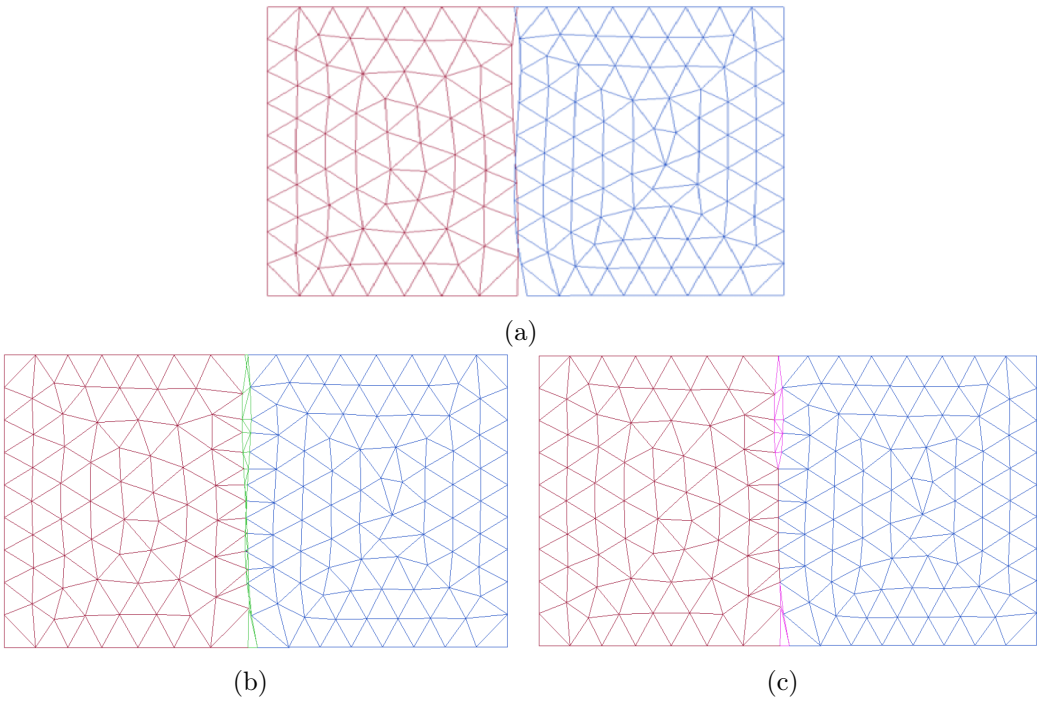


FIGURE 2.5 – (a) Maillage original, (b) Après l'opération d'expansion (*filling*), (c) Après l'opération de contraction (*stitching*). D'après Fig.3, Fig.4 et Fig.5 de (Patel *et al.* [2005]).

teur. Ainsi, l'orientation des normales est fixée pour les faces des voxels visible, puis l'orientation de la normale est propagée aux voxels voisins.

Avantages : La méthode est une approche complète pour la construction d'une surface 2D-variété, sans bord, orientable et orientée, plongée dans E^3 à partir d'une images 3D ou d'un ensemble de points.

Inconvénients : La méthode nécessite des prétraitements pour définir la taille d'un voxel, construire la grille de voxel et éliminer les points parasites (points inexistants à la surface de l'objet). L'heuristique fonctionne uniquement si la grille de voxels représente un volume fini.

2.2.5 Intersection et auto-intersection de surfaces

Les intersections entre surfaces ou les auto-intersections sur une surface sont des conditions de non-variété. Certains *Slicers* peuvent imprimer un modèle 3D comportant des intersections et des auto-intersections en les traitant par couches 2D, mais elles peuvent induire d'autres problèmes comme une surface non-orientable. Le traitement des auto-intersections consiste à trouver une interprétation topologique autour de ces conditions de non-variété.

Nooruddin et Turk (Nooruddin et Turk [2003]) proposent deux méthodes de conversion des polygones en représentation voxel permettant d'éliminer des éléments indésirables tels que des petits trous, des volumes minces ou des auto-intersections.

Les deux méthodes consistent à plonger le maillage dans une grille régulière de voxels. Chaque voxel est ensuite classé comme "intérieur" ou "extérieur".

Pour corriger les erreurs, deux opérations morphologiques, érosion et dilatation, sont effectuées pour l'"ouverture" (une érosion suivie d'une dilatation) ou pour la "fermeture" (dilatation suivie d'une érosion) de petits trous et pour joindre des parties de surface non connectées.

Enfin, une extraction d'isosurface est effectuée par une extension de l'algorithme *Marching Cube* qui prend en compte une interpolation de couleur pour les triangles générés.

La différence entre les deux méthodes vient du classement des voxels. La première méthode "*Parity Count*" consiste à compter le nombre d'intersections entre les polygones et un rayon dont l'origine est située au centre du voxel. Un nombre impair d'intersections signifie que le voxel est à l'intérieur du modèle et un nombre pair signifie qu'il est extérieur. Pour que la méthode fonctionne avec des modèles comportant des trous et des condition de non-variété, un nombre k de rayon est lancé dans k direction et le voxel est classé dans la catégorie majoritaire.

La deuxième méthode "*Ray Stabbing*" consiste à lancer un rayon de l'extérieur du modèle et à ne retenir que la première et la dernière intersection avec un polygone du maillage. Les voxels qui se situent entre ces deux intersections sont classés comme intérieur par le rayon. Plusieurs rayons sont lancés dans différentes directions, un voxel est classé comme intérieur si tous les rayons le classe comme tel. Les autres voxels sont classés comme extérieur.

Avantages : La méthode permet une réparation complète, d'un maillage vers une 2D-variété. Les opérations morphologiques de volumes utilisées réparent les petits trous, les auto-intersections et les détails intérieurs. Les attributs de couleur sont préservés.

Inconvénients : La topologie n'est pas préservée, si une érosion ou une ouverture est effectuée sur un volume mince, l'érosion pourrait détruire la surface.

2.2.6 Faces internes

Les faces internes sont la conséquence des intersections entre plusieurs surfaces ou des auto-intersections, une partie des faces du maillage se trouve à l'intérieur du volume de l'objet. Les *Slicers* qui traitent les intersections et les auto-intersections traitent aussi les faces internes. Une coupe d'un modèle en

cours d'impression montre que ni la géométrie ni la couleur des faces internes ne sont imprimées, elles sont remplacées par la structure interne de l'objet imprimé. Néanmoins avoir une surface 2D-variété unique sans auto-intersection facilite la plupart des algorithmes notamment pour traiter les problèmes de faiblesses. De plus cela évite de laisser le *Slicer* qui dépend de l'imprimante utilisée faire cette correction.

Campen et Kobbelt (Campen et Kobbelt [2010]) proposent une modification locale des (auto-)intersections des surfaces en utilisant un arbre de partition binaire de l'espace (BSP pour *binary space partitioning*). L'arbre BSP est utilisé comme une représentation du modèle 3D, où chaque plan séparant l'espace en deux est le plan d'une face du maillage. Cette méthode de représentation du modèle 3D est reprise de **Thibault et Naylor** (Thibault et Naylor [1987]).

L'arbre permet de définir si une partie de l'espace est à l'intérieur ou à l'extérieur du modèle 3D. La surface est reconstruite en utilisant les plans frontières entre intérieur et extérieur. Les sommets et les arêtes de la surface sont reconstruits à partir des intersections de ces plans (Fig.2.6).

Avantages : La représentation de la géométrie basée sur les plans d'un arbre BSP assure de ne pas déformer les surfaces originales après la reconstruction.

Inconvénients : La méthode ne gère pas les vides internes.

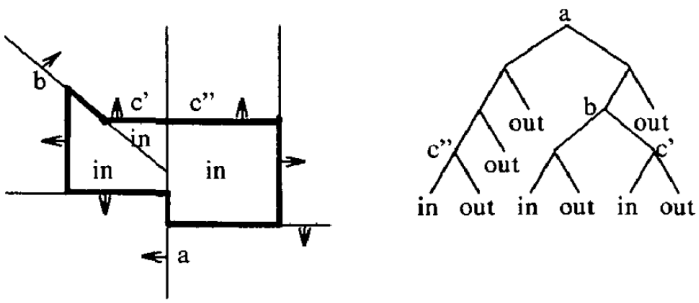


FIGURE 2.6 – Arbre BSP utilisé pour définir l'intérieur et l'extérieur d'un polygone. D'après Fig.3 de (Thibault et Naylor [1987]).

2.2.7 Conclusion

Les méthodes présentées dans l'état de l'art ne permettent pas de répondre entièrement à notre problématique. Plusieurs idées seront néanmoins en partie reprises dans les différentes étapes de notre chaîne de réparation :

— Réparer dans un premier temps les conditions de non-variété et dans un deuxième temps les arêtes de bord (Bohn et Wozny [1992]).

— Dupliquer les éléments du maillages 3D pour corriger les conditions de non-variété (Rossignac et Cardoze [1999]).

— Orientation des normales en utilisant un lancer de rayon depuis l'extérieur du modèle 3D (Roth et Wibowoo [1997]).

— Adapter la réparation des arêtes de bord en fonction de leur contexte (Patel *et al.* [2005]).

— Projection des contours sur un plan pour vérifier les auto-intersections (Roth et Wibowoo [1997]).

— Remplir les trous en utilisant une triangulation 2D par *Earclipping* (Varnuska *et al.* [2005]).

— La détection de fragilité utilise une méthode de lancer de rayon proche de celles utilisées par (Nooruddin et Turk [2003]).

La méthode présentée par (Attene [2016],Attene [2018]) est celle qui se rapproche le plus des objectifs de cette thèse. Elle a la particularité de prendre en entrée des maillages quelconques ("soupe de polygones") et d'avoir pour objectif l'impression 3D de ce maillage.

2.3 Logiciels Existants

De nombreux logiciels, proposant des réparations sur des modèles 3D en vue d'une impression 3D, existent. Une liste représentative de ces logiciels a été établie pour les comparer et les tester. Cette liste est basée sur les recommandations de plusieurs sites internet spécialisés [6] [7] et par connaissance de certains logiciels expérimentés dans le cadre industriel.

Les logiciels de réparation sont comparés dans un premier temps sur trois critères principaux (Tab.2.1) :

— Les plateformes supportées : Le portage sur plusieurs plateformes facilite le déploiement en production.

— Les formats 3D utilisés : Les formats STL et OBJ sont actuellement les deux formats standards, il était donc impératif que les logiciels retenus pour les tests les supportent tous les deux, autant en entrée qu'en sortie.

— La licence d'utilisation : Les logiciels sous licence gratuite nous assurent la possibilité de pouvoir les tester et de pouvoir comparer les performances des différentes méthodes et des différents outils de réparation qu'ils proposent.

6. https ://all3dp.com/1/best-free-3d-printing-software-3d-printer-program/
7. http ://meshrepair.org/

Logiciel	OS				Format			Licence
	Linux	Mac OS	Windows	Web	STL	OBJ	Autres	Gratuite
3D Tools				✓	✓	✓	✓	✓
3DprinterOS				✓	✓	✓		
Ansys SpaceClaim	✓		✓		✓	✓	✓	
Autodesk Meshmixer		✓	✓		✓	✓	✓	✓
Autodesk Netfab			✓				✓	
Blender	✓	✓	✓		✓	✓	✓	✓
Emendo		✓	✓		✓			
FreeCAD	✓	✓	✓		✓	✓	✓	✓
Graphite	✓		✓		✓	✓	✓	✓
LimitState :FIX			✓		✓			
MakePrintable				✓	✓	✓	✓	
Materialise Cloud				✓	✓	✓	✓	
MeshFix			✓		✓			✓
MeshLab	✓	✓	✓		✓	✓	✓	✓
MeshWorks			✓			✓	✓	✓
Open3mod			✓		✓	✓	✓	✓
OpenFlipper	✓	✓	✓		✓	✓	✓	✓
PolyMender	✓		✓		✓		✓	✓
RameshCleaner								
ReMESH	✓		✓			✓	✓	✓
TrIMM			✓		✓	✓	✓	✓
Trinckle				✓	✓		✓	✓

TABLE 2.1 – Classification des logiciels en fonction de la plateforme, des formats et de la licence. Logiciels testés dans la (Section.2.3.2)(en Bleu).

On distingue plusieurs familles de logiciel inter-connectées en fonction de la provenance du logiciel et de ses fonctionnalités :

Logiciels commerciaux/Logiciels supportées par des communautés : Les logiciels commerciaux sont développés par des entreprises privées, que ce soit par des grands groupes du domaine (**Autodesk Meshmixer**) ou par des entreprises plus modestes (**MakePrintable**). Les logiciels supportés par des communautés peuvent l'être par une communauté formée autour d'un projet *open source* (**Blender**) ou par une communauté universitaire attachée à des organismes de recherche (**MeshLab**).

Logiciels de modélisation/Logiciels de réparation : Les logiciels conçus pour la modélisation géométrique intègrent des fonctionnalités de réparation en vue de l'impression 3D. Le but est de garantir après la modélisation d'un modèle 3D un export imprimable. On peut distinguer deux sous-catégories de logiciels de modélisation, les logiciels de modélisation surfacique (**Blender**) et les logiciels de modélisation CAO (pour conception assistée par ordinateur)(**FreeCad**).

Les logiciels de réparation peuvent être répartis en deux sous-catégories, les logiciels de réparation topologique et géométrique généralistes (**MeshLab**) et les logiciels de réparation spécialisés pour l'impression 3D (**3D Tools**).

Réparation automatique/Réparation manuelle : Les réparations automatiques prennent en entrée un modèle 3D et, sans intervention hormis quelques paramètres, retournent une version imprimable du modèle 3D (**MakePrintable**).

Les réparations manuelles sont faites en utilisant différentes fonctionnalités séparées sur le modèle 3D comme le remplissage de trous, l'épaississement, ou encore le traitement des conditions de non-variété. Ces fonctions sont laissées au libre choix de l'utilisateur (**ReMesh**).

Méthode surfacique/Méthode volumique : Les méthodes surfaciques s'effectuent sur les éléments du maillage (sommets, arêtes, faces). Cette approche permet de conserver les éléments originaux du modèle 3D (3DTools).

Les méthodes volumiques comprennent le plongement du maillage original dans une grille spatiale. En sortie, le maillage est alors reconstitué à partir des informations contenues dans les cellules de la grille. Les éléments de la surface originale ne sont pas réutilisés. Cette procédure modifie forcément le modèle 3D (**MakePrintable**).

Six logiciels sont sélectionnés pour être testés sur dix modèles de test. La sélection a été faite pour couvrir le plus possible de familles.

— **3D Tools :** Logiciel de réparation pour l'impression 3D, intégré à un site

internet gratuit, est développé par une entreprise privée (Microsoft). La réparation est automatique mais la méthode utilisée n'est pas connue, la surface originale semble modifiée par endroit et conservée à d'autres.

— **Blender :** Logiciel de modélisation géométrique surfacique développé par une communauté *open source*. Il propose des réparations manuelles adaptées à l'impression 3D. Les réparations proposées sont des méthodes surfaciques.

— **MakePrintable :** Logiciel de réparation pour l'impression 3D avec une licence payante, développé par une entreprise privée. Il propose une réparation automatique avec une méthode volumique.

— **MeshLab :** Logiciel de réparation et traitement généraliste développé par une communauté universitaire. Il propose des réparations manuelles avec des méthodes surfaciques et volumiques. Seules des méthodes surfaciques seront testées.

— **MeshWorks :** Logiciel de réparation et traitement généraliste développé par Charlie Wang pour ses recherches universitaires. Il propose des réparations manuelles avec des méthodes surfaciques ou utilisant un arbre BSP (Wang et Manocha [2013]), mais aussi une méthode d'épaississement de surface adaptée pour l'impression 3D (Wang et Chen [2013]).

— **ReMesh :** Logiciel de réparation et traitement généraliste développé Marco Attene pour ses recherches universitaires. Il propose des réparations manuelles avec des méthodes surfaciques (Attene [2014]).

2.3.1 Présentation des modèles de test

Les dix modèles de test sont répartis en trois catégories en fonction du type principal de réparation qu'ils testent :

Les quatre modèles "vertex1CC", "vertex2CC", "edge1CC" et "edgeMultiCC" sont conçus pour tester principalement les réparations des conditions de non-variété.

— "vertex1CC" : 1 sommet condition de non-variété (Fig.2.7,(a)).

— "vertex2CC" : 1 sommet condition de non-variété qui sépare deux volumes (Fig.2.7,(b)).

— "edge1CC" : 1 arête condition de non-variété (Fig.2.7,(c)).

— "edgeMultiCC" : 1 arête condition de non-variété qui sépare deux volumes et un plan, 3 arêtes de bord (Fig.2.7,(d)).

Les quatre modèles "cubeHole", "cubeU", "Casquette" et "FlorentHead" ont été choisis pour tester principalement les réparations des arêtes de bord.

— "cubeHole" : 25 arêtes de bord, 1 contour sur un seul plan (Fig.2.8,(a)).

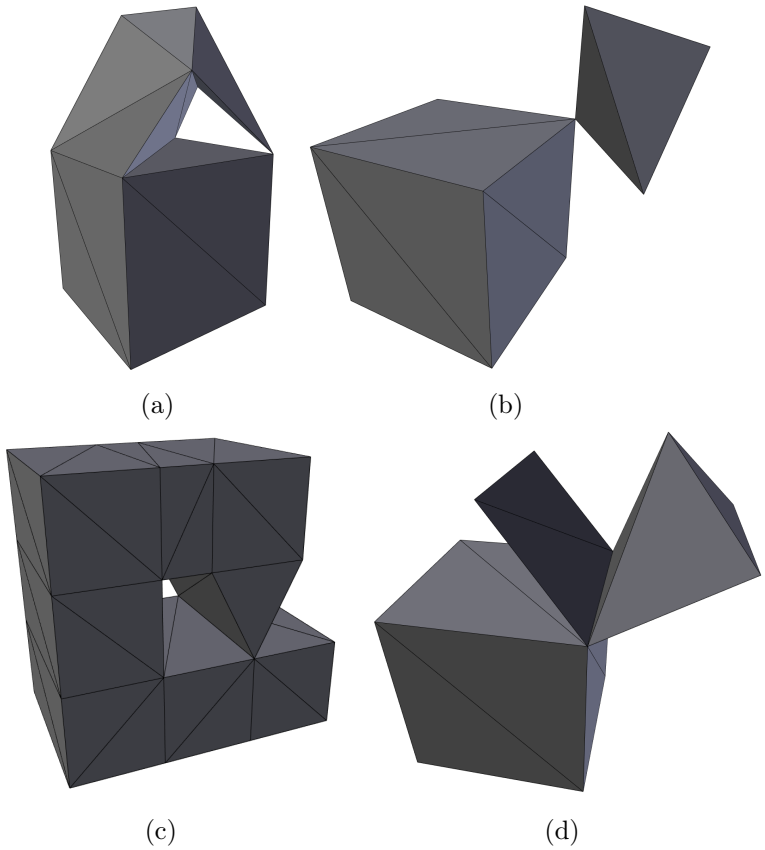


FIGURE 2.7 – Modèles de test des conditions de non-variété : (a) "vertex1CC", (b) "vertex2CC", (c) "edge1CC", (d) "edgeMultiCC".

— "cubeU" : 203 arêtes de bord, 1 contour sur deux plans (Fig.2.8,(b)).

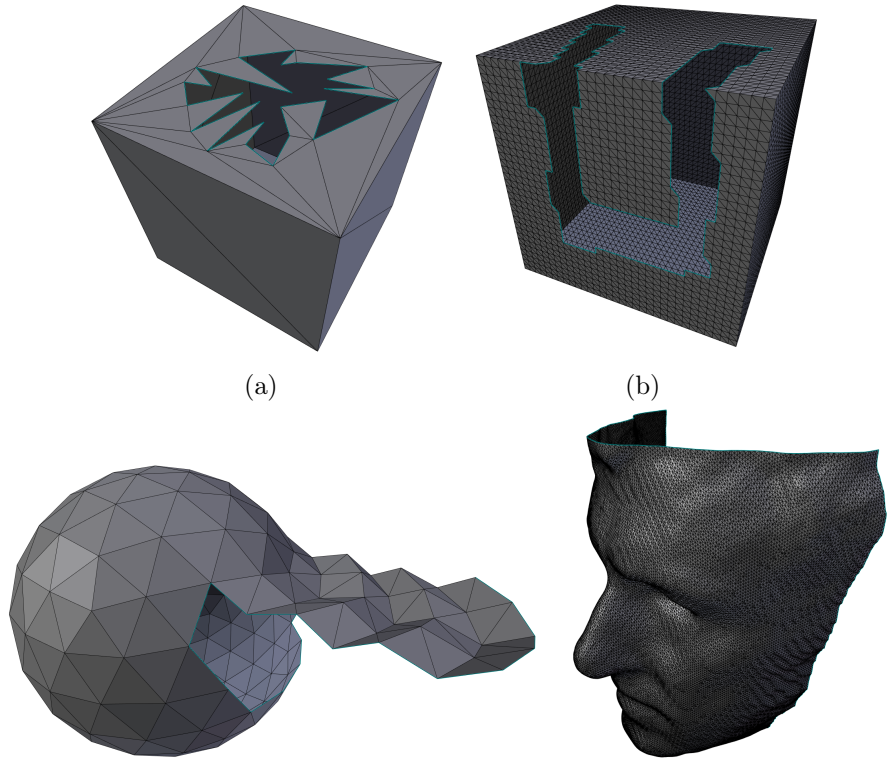
— "Casquette" : 26 arêtes de bord, 1 contour 3D (Fig.2.8,(c)).

— "FlorentHead" : 639 arêtes de bord, 1 contour sur deux plans, modèle industriel (Fig.2.8,(d)).

Les deux modèles "Kate" et "WOW" sont des modèles industriels complexes pour tester une réparation complète.

— "Kate" : 6455 arêtes de bord (pour 58339 arêtes sur le modèle), mauvaises orientations de surfaces, nombreux volumes intersectés, nombreuses surfaces planes sans volume (Fig.2.9,(a)).

— "WOW" : 0 connectivité entre les faces, arêtes et sommets conditions de non-variété après ajout d'une connectivité, nombreux volumes intersectés, surfaces planes sans volume (Fig.2.9,(b)).



(a) (b)

(c) (d)

FIGURE 2.8 – Modèles de test des réparations de bords : (a) "cubeHole", (b) "cubeU", (c) "Casquette", (d) "FlorentHead".

(a) (b)

FIGURE 2.9 – Modèles de test d'une réparation complète : (a) "Kate", (b) "WOW".

2.3.2 Résultat des logiciels existants sur les modèles de test

Les six logiciels ont des fonctionnements différents, les tests de réparations ont été réalisés par un processus différent sur chacun :

— **3D Tools** : La réparation est automatique.

— **Blender** : Les réparations sont faites avec l'add-ons *3D Print Toolbox*. On utilise dans cet outil la fonction *Make Manifold*. C'est une réparation automatique de type surfacique.

— **MakePrintable** : La réparation est automatique.

— **MeshLab** : Pour corriger les modèles de test de condition de non-variété, on utilise la fonction *Split Vertexes Incident on Non Manifold Faces*. Pour réparer les modèles de test des réparations de bords, on utilise la fonction *Close Hole*.

— **MeshWorks** : Les réparations sont faites sur tous les modèles avec la fonction *Mesh Hole-filling* ou la fonction *Mesh Thickening*.

— **ReMesh** : Les réparations sont faites sur tout les modèles avec la fonction *Fill holes*.

Pour évaluer une réparation il faut en sortie vérifier que les problèmes du modèle original ont été réparés mais aussi que de nouvelles erreurs n'ont pas été créées par la réparation. Ces erreurs sont du même type que celles qu'on cherche à corriger. Par exemple, des nouvelles arêtes de bord créées lors d'une réparation alors que le modèle original n'en avait pas. Ou encore une mauvaise orientation des normales alors que la surface était correctement orientée avant la réparation.

On évalue aussi le respect de la géométrie et de l'apparence du modèle original. La réparation ne doit pas déformer le volume du modèle 3D ni changer sa couleur ou sa texture. Que ce soit pour la forme ou la couleur de l'objet, si une réparation nécessite la création de nouvelle face, il faut qu'elle respecte au mieux le modèle original. L'évaluation de ces erreurs est visuelle et soumise à interprétation. La tolérance au changement dépend de l'utilisation de l'objet imprimé. Une pièce mécanique sera peu tolérante sur le changement de forme, alors qu'un objet décoratif sera peu tolérant sur le changement de ses couleurs et de ses textures.

Les résultats sont présentés sous forme de tableaux, un tableau par catégorie de modèles testées (Tab.2.2)(Tab.2.3)(Tab.2.4). Les tableaux sont remplis par des abréviations des erreurs présentes en sortie :

— ✓= Imprimable

— blanc = Modèle non testé pour ce logiciel (Licence payante)

— / = séparation entre deux réparations sur le même modèle

— AB = Arête de bord

— DV = Déformation du volume

— FI = Face isolée

— FNP = Face non planaire

— NT = Problème non traité par le logiciel, entre parenthèse les problèmes du modèle d'origine non traité

— NV = non-variété

— MON = Mauvaise orientation de normale

— II = Import Impossible du modèle par le logiciel

— OVL = Recouvrement de la surface (*Overlap*)

— SNO = Surface non orientable

— ZVN = Zone de volume nul

Modèles de test des conditions de non-variété

Sur les modèles de test des conditions de non-variété (Tab.2.2), le problème le plus courant est que les conditions de non-variété ne sont pas traitées par le logiciel de réparation. La condition de non-variété est toujours présente en sortie (Fig.2.10,(a)).

Le logiciel Blender est celui dont les réparations sur les conditions de non-variété ont le plus de problèmes en sortie. Sur le modèle "edge1CC" on note : une forte déformation du volume il ne reste plus que la partie supérieure du volume (DV), une face avec quatre sommets qui ne sont pas sur un même plan (FNP) et une face isolée (FI) qui n'a par conséquent pas de volume (ZNV) et dont toutes les arêtes sont des arêtes de bord (AB)(Fig.2.10,(b)).

Certains logiciels lèvent correctement les conditions de non-variété, par exemple MeshLab mais uniquement sur les sommets (Fig.2.10,(c)). Le logiciel ReMesh lève les conditions de non-variété sur les arêtes mais sur le modèle "edgeMultiCC" le plan est réparé par le recouvrement (OVL) des deux faces du plan par deux autres faces, ce qui crée une zone de volume nul (ZVN)(Fig.2.10,(d)).

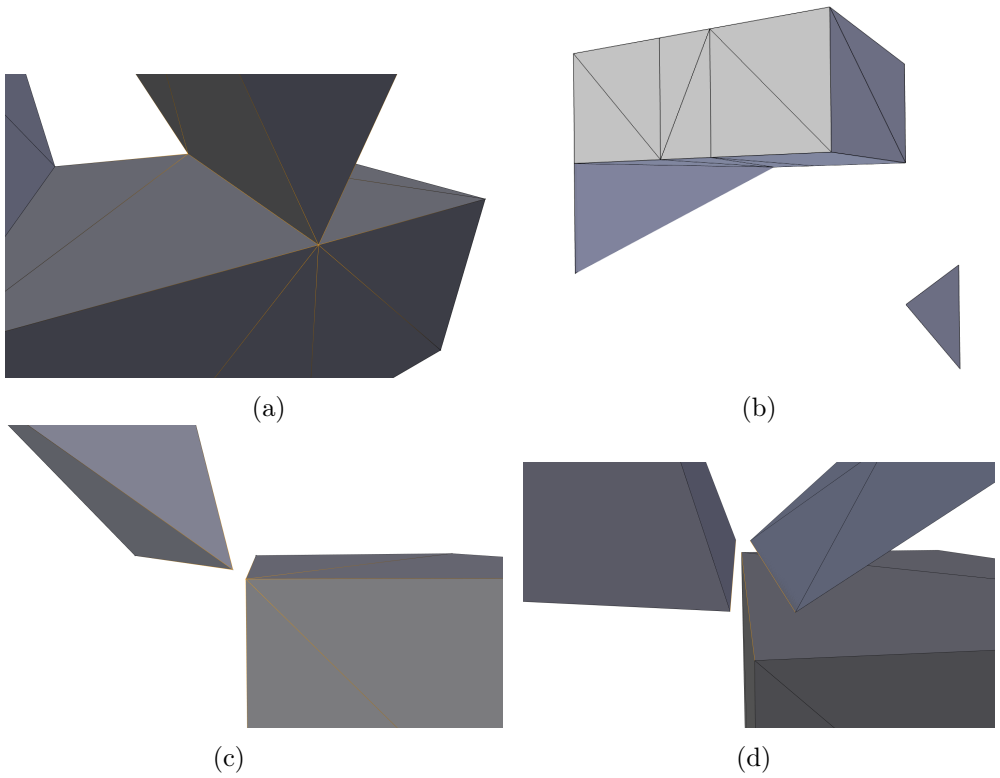


FIGURE 2.10 – Exemples de réparations sur les conditions de non-variété : (a) "edge1CC" par Mesh Works, (b) "vertex2CC" par MeshLab, (c) "edge1CC" par Blender, (d) "edgeMultiCC" par ReMesh.

Logiciel	vertex1CC.obj	vertex2CC.obj	edge1CC.obj	edgeMultiCC.obj
3D Tools	NT(NV)	NT(NV)	NT(NV)	NT(NV),DV
Blender	DV	DV,FI	DV,FI,FNP	DV
Make Printable				✓
MeshLab	✓	✓	NT(NV)	NT(NV,AB)
MeshWorks	NT(NV)	NT(NV)	NT(NV)	NT(NV,AB)
ReMesh	✓	✓	✓	OVL

TABLE 2.2 – Résultat sur les modèles de test des conditions de non-variété : résultats illustrés (en Rouge).

Modèles de test des réparations de bords

Sur les modèles de test des réparations de bords (Tab.2.3), le problème le plus courant sont des auto-intersections créées par la réparation qui provoquent

des surfaces non-orientable (SNO) et des zones de volume nul (ZVN)(Fig.2.11,(a)). La réparation essaye de remplir le trou dans la surface mais la forme 3D du contour entourant le trou provoque ces erreurs.

Le remplissage d'un trou peut ne pas finir entièrement, dans ce cas il restera sur le modèle des arêtes de bord (AB)(Fig.2.11,(b)). Si le remplissage se termine correctement alors il n'y a plus d'arête de bord sur la surface (Fig.2.11,(c)).

Une autre méthode pour réparer des bords c'est l'épaississement de la surface. Le Logiciel MakePrintable utilise une méthode d'épaississement volumique avec reconstruction de la surface sur le modèle "cubeU" (Fig.2.11,(d)). Dans ce cas, même si le modèle est imprimable en sortie on remarque que la réparation n'est pas celle attendue. De plus, la faible épaisseur du modèle risque de casser à l'impression ou demandera une taille d'impression très grande pour être solide.

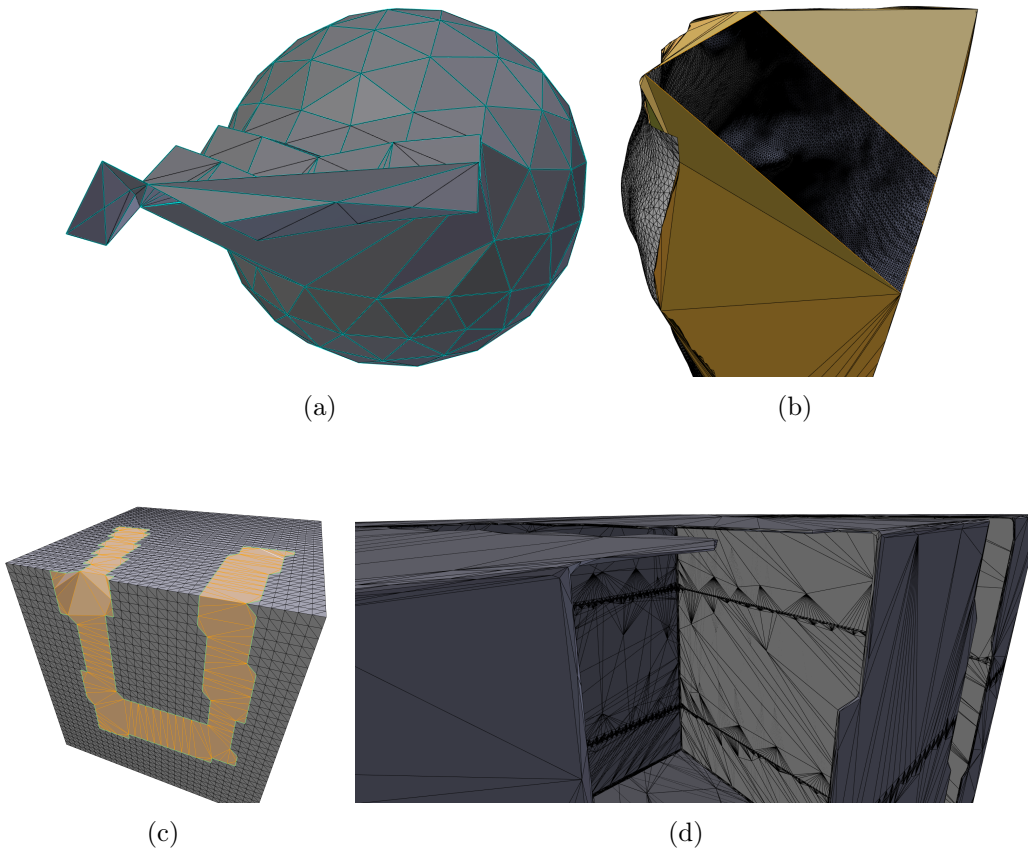


FIGURE 2.11 – Exemples de réparations sur les bords : (a) "Casquette" par 3D Tools, (b) "FlorentHead" par MeshLab, (c) "cubeU" par Blender, (d) "cubeU" par MakePrintable.

Logiciel	Casquette.obj	cubeHole.obj	cubeU.obj	FlorentHead.obj
3D Tools	SNO,ZVN	✓	✓	SNO,ZVN
Blender	SNO,ZVN	✓	✓	SNO,ZVN
Make Printable	✓		✓	
MeshLab	AB,SNO,ZVN	AB,OVL	OVL	AB,ZVN
MeshWorks	SNO,ZVN / ✓	✓	✓	SNO,ZVN
ReMesh	SNO,ZVN	✓	OVL	II

TABLE 2.3 – Résultat sur les modèles de test des réparations de bords : résultats illustrés (en Rouge).

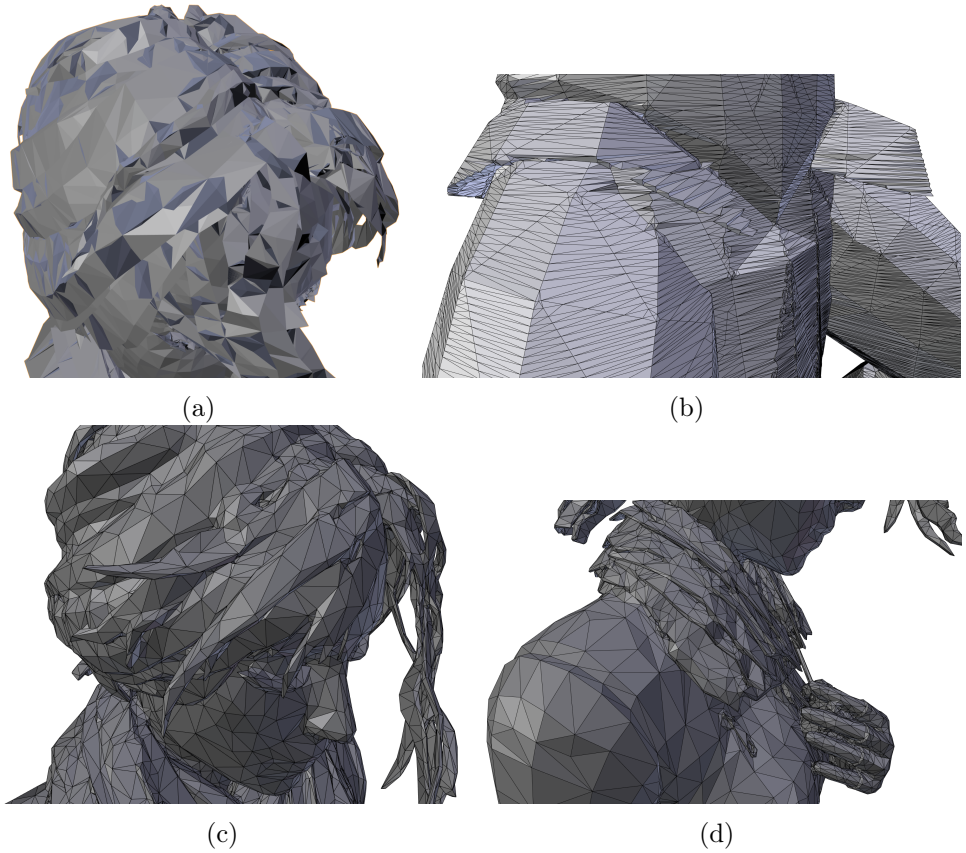
Modèles de test de réparations complètes

Sur les modèles de test de réparations complètes (Tab.2.4), seul le logiciel MakePrintable arrive à un résultat imprimable ou proche. Les autres logiciels n'arrivent principalement pas à corriger les surfaces planes présentes dans les modèles originaux, leurs corrections reviennent souvent à créer des zones de volume nul (ZVN).

D'autres erreurs peuvent être observées, par exemple les cheveux du modèle réparé par 3D Tools ne sont pas correctement orientés, l'arrière de nombreuses faces sont visibles (MON). De plus, quelque soit l'orientation choisie pour les surfaces de ces cheveux, il y aura toujours des faces avec la mauvaise orientation. On dit que la surface est non-orientable (SNO)(Fig.2.12,(a)).

Sur le modèle "WOW" on peut noter quelques défauts dans la réparation de MakePrintable, une déformation légère de la surface avec la formation de zone crénelée et un volume déconnecté du reste du modèle (Fig.2.12,(b)). La réparation de ce modèle par les outils de la thèse permet de conserver la surface originale (Fig.10.20,(b)(c)).

Sur le modèle "Kate" la réparation de MakePrintable est parfaitement imprimable. La méthode volumique change toujours la surface originale.



(a) (b)

(c) (d)

FIGURE 2.12 – Exemples de réparations complètes : (a) "Kate" par 3D Tools, (b) "WOW" par MakePrintable, (c)(d) "Kate" par MakePrintable.

Logiciel	kate.obj	wow.obj
3D Tools	NV,SNO,ZVN	ZVN
Blender	DV,FI,MON,SNO,ZVN	DV,FI,AB,ZVN
Make Printable	✓	AB,DV
MeshLab	II	NT(AB)
MeshWorks	ZVN	ZVN
ReMesh	II	II

TABLE 2.4 – Résultat sur les modèles de test de réparations complètes : résultats illustrés (en Rouge).

Traitement de la couleur par les logiciels testés

Seuls les logiciels Blender, Meshlab et Makeprintable conservent la couleur après réparation. Blender et Meshlab conservent les informations de couleurs

originales sur les faces qui ne sont pas modifiées. Pour les nouvelles faces Blender leur donne une couleur blanche et MeshLab une couleur aléatoire.

Makeprintable refait une surface lors de sa réparation, il conserve la couleur en créant une texture à partir de rendus du modèle 3D avant la réparation. Après la réparation il applique cette texture sur le modèle réparé. On note une déformation importante de la couleur. Il y a aussi une perte de qualité (flou, pixelisation)(Fig.2.13). La réparation de ce modèle par les outils de la thèse permet de conserver la texture originale (Fig.10.21,(c)).

(a) (b)

FIGURE 2.13 – Couleurs du modèles "WOW" après une réparation avec le logiciel MakePrintable.

2.3.3 Conclusion

Le logiciel Makeprintable avec son approche volumique a les meilleurs résultats sur les modèles de test. Ses plus gros défauts sont une déformation de la surface et donc du volume mais aussi la perte de qualité sur les couleurs et les textures. La reconstruction entière de la surface rend difficile la conservation des couleurs et des textures originales. Dans le contexte de notre travail et de l'impression 3D couleur, la perte et la déformation de la couleur originale sont des défauts éliminatoires.

2.4 Conclusion

À l'heure actuelle et à ma connaissance, il n'existe pas d'outils de réparation des problèmes structurels (conditions de non-variété, arêtes de bord) et des problèmes de faiblesse (Fragilité) qui garantissent la préservation de la géométrie et de l'aspect (couleur, texture) du maillage original.

Le travail de cette thèse propose des outils de réparation surfacique préservant la géométrie et l'apparence. Les principaux outils proposés sont :

1. Une méthode de correction des conditions de non-variété, sans suppression ou déplacement d'élément du maillage, sans interprétation en cas

de choix multiples et sans prise en compte de l'orientation originale des faces.

2. Une méthode de correction des arêtes de bord, en différenciant celles bordant un trou de celles bordant une surface fine pour leurs appliquer une réparation adaptée.
3. Une méthode de détection de la fragilité.
4. Une chaîne de réparation regroupant ces méthodes dans un ordre améliorant la réparation complète.

Deuxième partie

Chaîne de réparation

Chapitre 3

Chaîne de réparation

L'objectif est de passer d'un modèle 3D quelconque à un modèle 3D imprimable.

Le modèle 3D en entrée est décrit dans un format STL ou OBJ. Il n'y a aucune condition d'entrée sur le maillage ("soupe de polygones"). Le modèle peut contenir des informations de couleurs ou de texture.

Le modèle 3D imprimable en sortie doit être visuellement proche du modèle d'entrée. La géométrie de la surface externe doit être conservée ou similaire, elle ne peut être modifiée que pour corriger des problèmes de faiblesse. Les couleurs et les textures présentes sur la surface externe doivent être conservées ou similaires. La surface externe correspond à la surface entourant le volume global du modèle 3D (Fig.3.6).

Remarque : l'impression est soumise à une limite de précision et peut lisser ou perdre des détails de la géométrie, des couleurs ou des textures. L'importance de ces pertes dépend de la taille d'impression du modèle. Il n'est alors pas nécessaire que le modèle 3D de sortie conserve ces détails.

Pour être imprimable, le modèle 3D doit être composé d'une ou plusieurs surfaces intersectées qui chacune définissent un volume. Une surface définit un volume dont elle est la frontière, si elle est une 2D-variété sans bord. Le volume du modèle 3D ne doit pas comporter de faiblesse physique.

La chaîne de réparation effectue des corrections par étapes. En effectuant leurs corrections, chaque étape valide des propriétés sur le modèle 3D, ce qui permet aux étapes suivantes de ne pas travailler sur une "soupe de polygones" mais sur un maillage structuré.

L'ordre des étapes est choisi pour que ces validations soient utiles aux étapes suivantes. Mais l'ordre prend en compte aussi le risque qu'une étape puisse créer des erreurs, soit pour que l'erreur soit traitée lors d'une étape suivante soit pour que l'erreur ne puisse pas empêcher le déroulement des étapes suivantes.

La chaîne est constituée des étapes suivantes (Fig.3.1) :

— La première étape de la chaîne consiste à passer d'un maillage "soupe de polygone" décrit au format STL ou OBJ à des surfaces variétés avec ou sans bord. Ces surfaces sont appelées composantes connexes.

— La deuxième étape de la chaîne consiste à regrouper les arêtes de bord en contours. Puis de classer les contours en deux catégories, entourant un trou, entourant une surface fine.

— La troisième étape de la chaîne consiste à réparer les contours classés comme entourant un trou avec un algorithme de remplissage. La réparation va transformer les arêtes de bord de ces contours en arêtes ayant deux faces voisines. Il ne reste plus dans les composantes connexes que des arêtes de bord appartenant à des contours entourant des surfaces fines.

— La quatrième étape de la chaîne consiste à épaissir les surfaces fines. Une surface est définie comme fine si elle possède un contour classé comme entourant une surface fine. Les arêtes de bord appartenant à ces contours sont transformées en arêtes ayant deux faces voisines. Chaque composante connexe est une surface 2D-variété sans bord et définit donc un volume.

— Une fois la garantie faite que le maillage définisse un volume ou plusieurs volumes intersectés, on peut modifier sa géométrie pour enlever les faiblesses physiques. La cinquième étape consiste à détecter les fragilités physiques.

— En fin de chaîne, on retourne simplement le maillage modifié en l'écrivant au format OBJ.

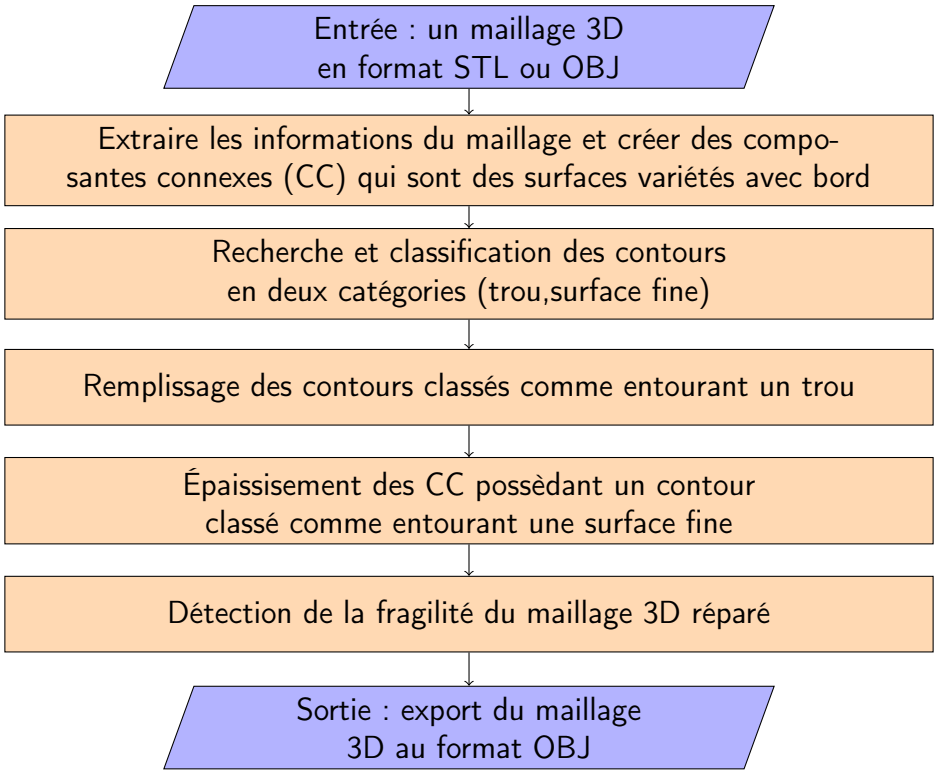


FIGURE 3.1 – Diagramme de la chaîne de réparation.

3.1 Importation, composante connexe et non-variété

Répartir les faces en composantes connexes permet de passer d'une "soupe de polygones" à des surfaces. Pour que le maillage définisse un volume il faut que chaque surface qui le compose définisse un volume. Les étapes suivantes de la chaîne vont pouvoir traiter chaque surface indépendamment. Des informations de connexité entre les éléments (sommet,arête,face) sont définies lors de cette répartition. Certaines étapes comme la recherche de contour ont besoin des informations de connexité.

La correction des conditions de non-variété a pour but de passer des surfaces à des surfaces variété avec ou sans bord. Des informations de connexité sont modifiées en levant des ambiguïtés qui n'auront pas à être traitées par les étapes suivantes. Les étapes suivantes qui modifient le maillage ne devront pas créer de nouvelles conditions de non-variété.

L'orientation des surfaces est ignorée jusqu'ici notamment dans les choix de connexité. Ces choix étant finis on peut orienter d'une façon cohérente les composantes connexes. Elles deviennent des surfaces variété avec ou sans bord.

La structure *Halfedge* est une structure de maillage 3D qui ne peut décrire

que des surfaces variété avec ou sans bord. Décrire le maillage 3D dans cette structure à la fin de l'étape d'importation permet de garantir les propriétés amenées par cette étape (Fig.3.2).

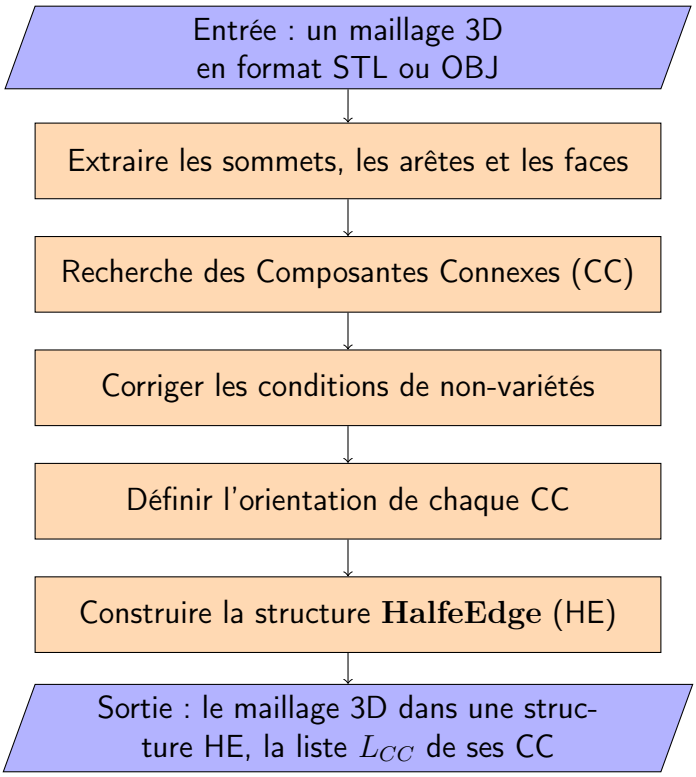


FIGURE 3.2 – Diagramme de l'importation des données brutes vers une structure *Halfedge*.

3.2 Détection et classification des bords

Les composantes connexes sont traitées indépendamment. Un contour est une boucle d'arêtes de bord consécutives. En sortie toutes les arêtes de bord appartiennent à un et seulement un contour.

Chaque contour est classé entre deux catégories. Ces catégories déterminent la réparation à appliquer pour transformer les arêtes de bord du contour en arêtes ayant deux faces voisines. Première catégorie, le contour entoure un trou dans la surface, on répare avec une méthode de remplissage sur le trou. Deuxième catégorie, le contour entoure la surface. La surface de ce contour devient une surface fine, on répare avec une méthode d'épaississement de la surface fine. En sortie, tous les contours sont classés dans une des deux catégories (Fig.3.3).

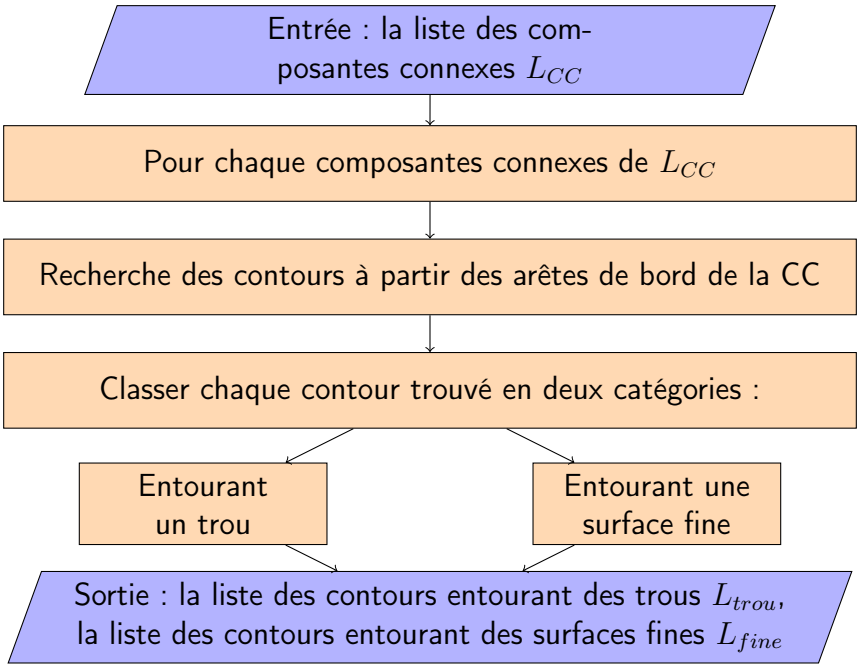


FIGURE 3.3 – Diagramme de la recherche et classification de contours.

3.3 Réparation des contours interprétés comme des trous dans la surface

Pour remplir un trou, on utilise une méthode de triangulation en deux dimensions (*earclipping*). La triangulation 2D ne peut être appliquée que sur un contour dont les éléments (sommets,arêtes) sont sur un même plan ou proche de l'être.

Chaque contour est décomposé en segments planaires. Chaque segment planaire est fermé par une arête de fermeture. La triangulation est appliquée sur chaque segment planaire fermé. Les arêtes de fermeture deviennent des arêtes de bord qui ne sont dans aucun contour. Une nouvelle recherche de contour permet de les inclure dans un ou plusieurs nouveaux contours. Il n'y a plus d'arêtes de bord lorsqu'un contour est décomposé en seulement un ou deux segments planaires (Fig.3.4).

En sortie toutes les arêtes de bord des contours classés comme des trous sont transformées en arêtes ayant deux faces voisines. De nouvelles arêtes sont créées et sont également des arêtes ayant deux faces voisines en sortie. Les arêtes de bord restantes appartiennent à des contours entourant des surfaces fines.

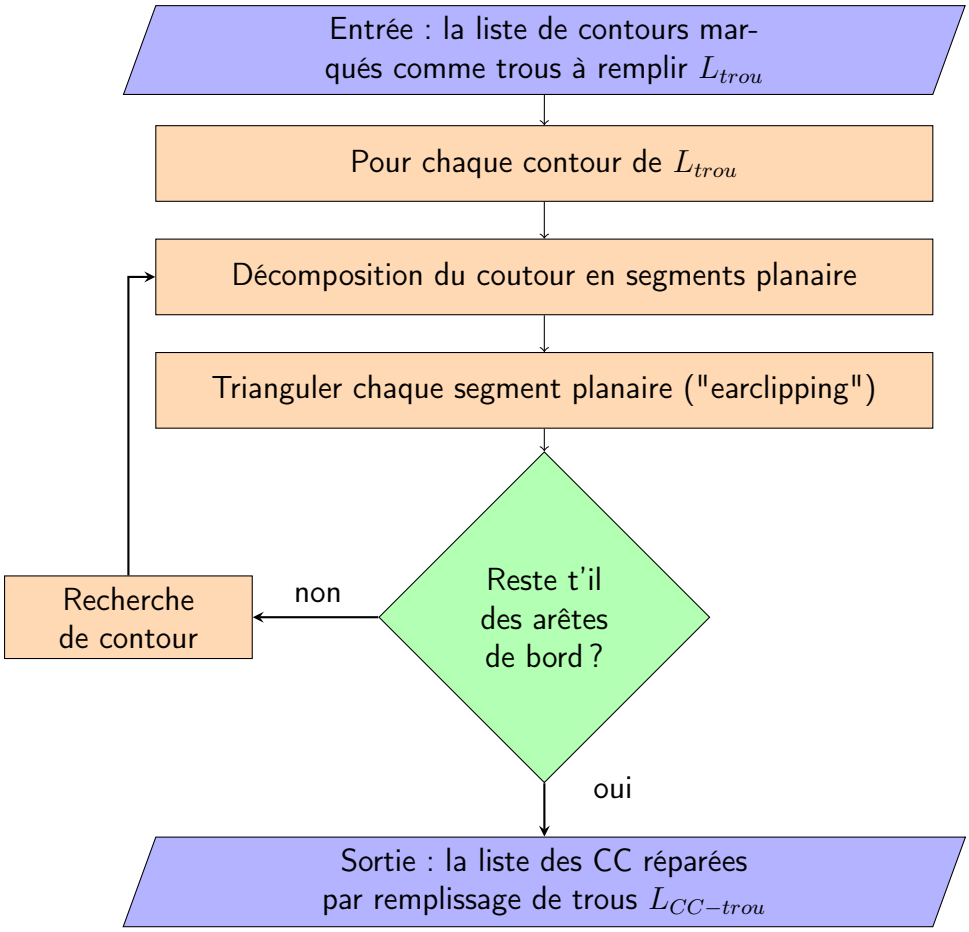


FIGURE 3.4 – Diagramme de la réparation de bords par remplissage.

3.4 Réparation des contours interprétés comme des frontières de surfaces fines

Chaque surface définie comme étant une surface fine est épaissie. La méthode d'épaississement consiste à créer une image de la surface fine et à décaler cette image d'une distance supérieure au seuil de fragilité. La surface fine et son image sont reliées en ajoutant des faces entre le contour entourant la surface fine et son équivalent sur la surface image. Ces faces transforment les arêtes de bords des deux contours en arêtes ayant deux faces voisines (Fig.3.5).

En sortie toutes les composantes connexes sont des surfaces 2D-variété sans bord qui définissent un volume. Le volume global du modèle 3D correspond à l'union de ces volumes. Éventuellement ces volumes peuvent ne pas s'intersecter, alors le modèle 3D sera imprimé en plusieurs objets.

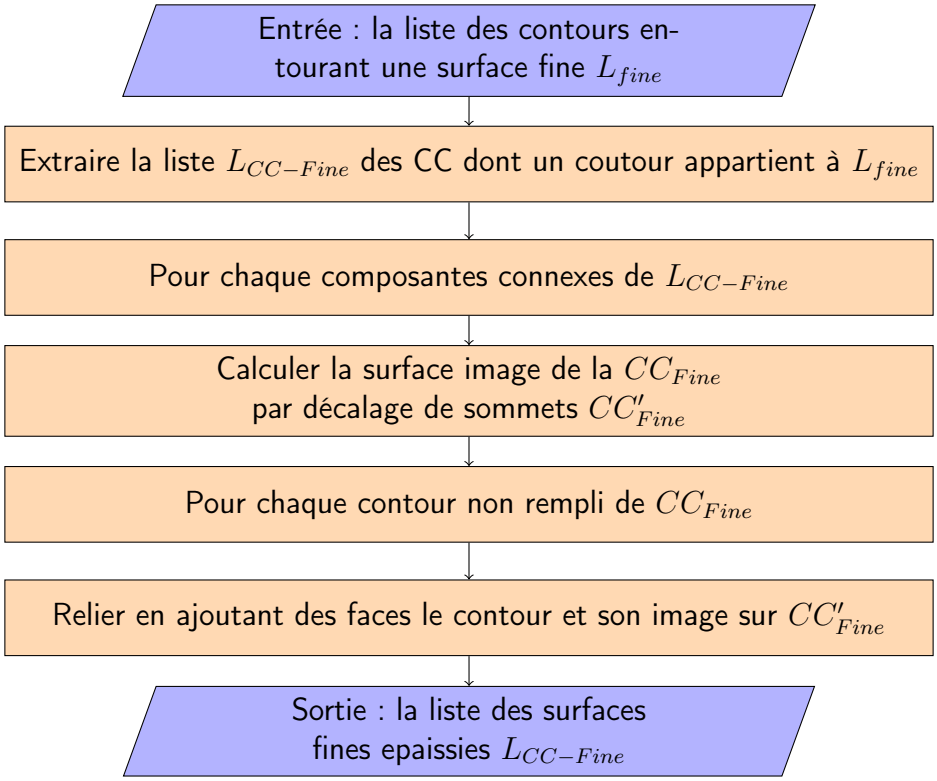


FIGURE 3.5 – Diagramme de la réparation de bords par épaississement.

3.5 Détection de la fragilité

La détection de la fragilité se fait en calculant l'épaisseur du volume du modèle 3D en différents points de sa surface externe. Rappel, la surface externe correspond à la surface entourant le volume global du modèle 3D. En pratique l'épaisseur est calculée pour le géocentre de chaque face.

L'épaisseur est la distance entre deux points de la surface externe, un point initial et son opposé. Soit une demi-droite partant du point initial vers l'intérieur du volume. La demi-droite est perpendiculaire au plan de la face du point initial. Le point opposé correspond au point d'intersection le plus proche du point initial entre la surface externe et cette demi-droite (Fig.3.6). L'épaisseur est comparée à un seuil de fragilité défini par la matière et la technologie de l'imprimante 3D.

En sortie, chaque face où une épaisseur inférieure au seuil a été calculée est marquée pour être signalée avant l'impression (Fig.3.7). D'autres étapes pourraient être rajoutées à la chaîne pour traiter des problèmes de faiblesse. Par exemple une correction automatique des fragilités détectées en modifiant le maillage pour augmenter l'épaisseur en ces points. Ou une étape pour détecter des problèmes d'équilibre du modèle une fois imprimé.

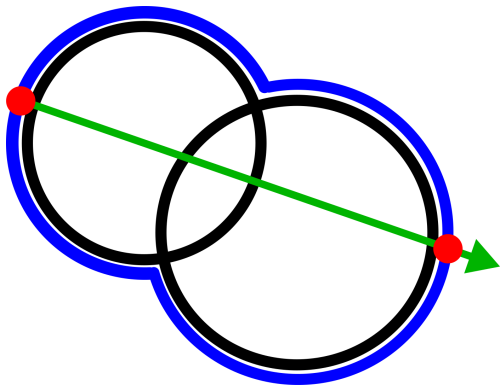


FIGURE 3.6 – Épaisseur pour un point de la surface externe : Surface des CC (en Noir), Surface externe (en Bleu), Point initial et son opposé (en Rouge), demi-droite (en Vert).

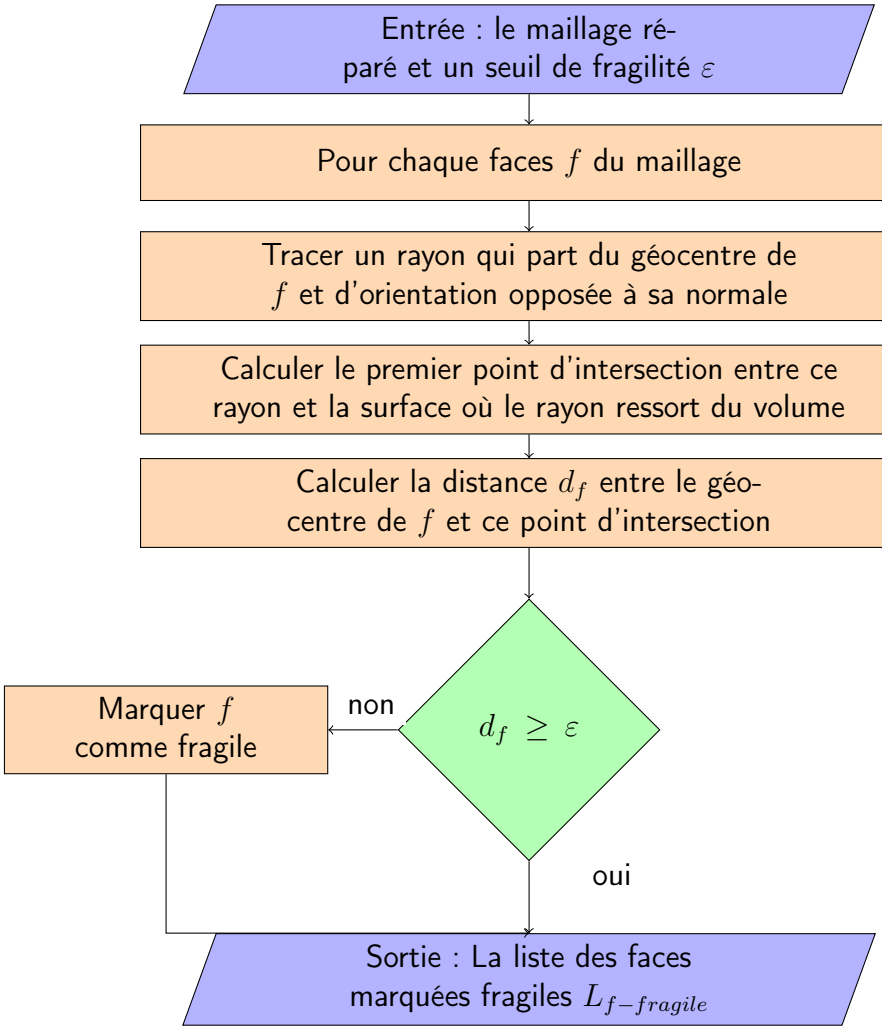


FIGURE 3.7 – Diagramme de détection de la fragilité.

Chapitre 4

Import

L'objectif de ce maillon de la chaîne est de passer d'un modèle 3D quelconque décrit dans le format OBJ à un modèle 3D ayant plusieurs caractéristiques :

— Les faces sont réparties en plusieurs surfaces appelées "composante connexe".

— Chaque composante connexe est une surface variété avec ou sans bord.

— Les faces d'une même composante connexe ont une orientation cohérente.

— Le modèle 3D est décrit dans une structure *Halfedge*.

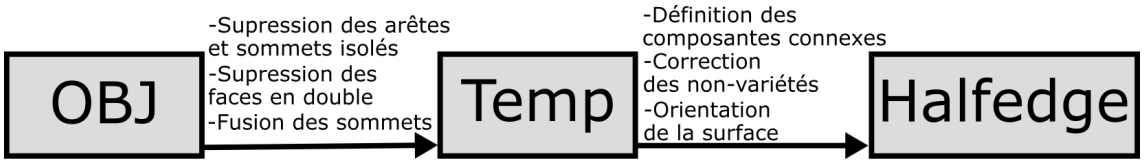


FIGURE 4.1 – Étapes de ce chapitre.

4.1 Extraction du format OBJ

Ce maillon de la chaîne a pour entrée un modèle 3D quelconque décrit dans le format (OBJ FileFormatInfo [[Online][Accessed : 14-November-2019]]). Un modèle au format STL peut également être utilisé après sa transformation en OBJ. Cette transformation est sans perte et sans interprétation, car les informations du format STL sont aussi présentent dans le format OBJ. Les informations du format OBJ nécessaire à l'import du modèle 3D dans notre chaîne sont les suivantes :

— Les sommets, leurs positions dans l'espace.

— Les faces, liste ordonnée de leurs sommets.

— Les informations de couleur et de texture attachées aux faces.

— L'orientation des faces.

D'autres informations présentes dans l'OBJ vont être ignorées et donc supprimées :

— Les sommets utilisés par aucune face.

— Les arêtes, elles seront extraites à partir des faces.

— Les faces en double sont supprimées, on conserve une occurrence.

— La connexité entre les faces. Une nouvelle connexité sera trouvée avec une étape de fusion des sommets ayant des coordonnées spatiales identiques.

Le format OBJ ne garantit pas que les sommets d'une face soient sur le même plan, s' ils sont plus de trois. Si une face n'a pas tous ses sommets sur le même plan alors elle n'est pas correctement définie. Les *Slicer* des imprimantes peuvent traiter ces cas en effectuant une triangulation. Mais le résultat d'une triangulation n'est pas unique, le choix fait par le *Slicer* n'est pas prévisible et peux être différents sur le *Slicer* d'une autre imprimante. La position de la surface au niveau d'une telle face change selon ce choix.

Effectuer une triangulation en amont de la réparation permet de définir la surface entre les sommets d'une telle face et de maîtriser ce choix. De plus, les étapes suivantes de la chaîne de réparation n'auront que des faces triangulaires à traiter, ce qui les simplifiera.

Une triangulation sur une face dont tous les sommets ne sont pas sur un même plan est une problématique complexe. En supprimant la face, on retrouve la même problématique que le remplissage d'un trou dans une surface, que l'on détaillera dans le (Chap.6).

Pour le *Slicer* la connexité entre les faces n'est pas utile, il traite chaque face indépendamment les unes des autres. Le format STL est le format le plus couramment accepté en entrée d'un *Slicer* et ne comporte pas d'information de connexité entre les faces.

Pour la chaîne de réparation, la connexité entre les faces est indispensable. Toutes les problématiques abordées au cours de la chaîne (variété, orientation, trou, épaississement) sont des problématiques de surface, or une surface est un ensemble de face connexe.

La connexité entre les faces est définie dans le format OBJ par l'utilisation d'un même identifiant d'un sommet. Cette connexité n'est pas gardée, car elle peut comporter des erreurs notamment sur les problèmes de non-variété. La connexité est créée par une fusion des sommets dont les coordonnées spatiales sont identiques, puis modifiées par les étapes de réparation des éléments de non-variété. Remarque : les modèles issus d'un format STL ont des sommets uniques pour chaque face, donc aucune face n'est connectée à une autre.

À la fin de ce maillon de la chaîne, on veut que les informations du modèle 3D soient enregistrées dans une structure 3D appelée *Halfedge*. Cette structure

va permettre pour les maillons suivants de faire des parcours sur des éléments plus rapidement. On détaillera la structure et ces parcours dans le (Chap.5). Le deuxième avantage de cette structure est qu'elle ne peut représenter que des surfaces qui sont des variétés avec ou sans bord. Or, pour qu'un modèle 3D puisse être imprimé, on a besoin que toutes ses surfaces soient des variétés sans bord. On va donc dans ce maillon faire des premières réparations pour que toutes les faces du modèle 3D d'entrée soient conservées dans cette structure.

4.2 Composante connexe

Les faces sont réparties par "composante connexe" (CC). Cette répartition permettra aux étapes suivantes de la chaîne de traiter chaque composante connexe séparément. Cette répartition va aussi permettre d'enlever certaines conditions de non-variété sur le modèle. Une composante connexe est définie comme suit :

— On définit deux faces comme "voisines" si elles ont deux sommets en commun et qu'il n'existe pas une ou plusieurs autres faces utilisant également ces deux sommets.

— On définit le "chemin entre deux faces voisines" (Fig.4.2, D et A) comme une suite de faces, voisines entre elles, permettant de passer de la première face (Fig.4.2, D) à la seconde (Fig.4.2, A).

— On définit une "composante connexe" comme un groupe de faces pour lesquels, quel que soit le couple de faces choisies, il existe un chemin entre elles.

L'orientation d'une face est définie par l'ordre de ses sommets. Dans le cas de la recherche des composantes connexes, l'orientation des faces n'est pas prise en compte pour exclure ou inclure une face dans une composante connexe. Donc l'ordre des deux sommets commun entre deux faces "voisines" n'a pas d'importance.

Chaque composante connexe doit être une surface variété avec ou sans bord. Une arête de la composante connexe doit être utilisée par uniquement une ou deux faces. Si en entrée une arête est utilisée par plus de deux faces (arête condition de non-variété), il est commun dans l'état de l'art d'essayer de regrouper ces faces en couples. L'orientation des normales de ces faces peut être utilisée pour le choix de ces couples (Bohn et Wozny [1992]).

Notre approche est d'interpréter le moins possible. L'orientation des normales n'est pas considérée comme fiable à ce stade de la chaîne. Si un chemin existe entre deux faces utilisant la même arête condition de non-variété, alors ces deux faces appartiennent à la même composante connexe. Par définition ces deux faces seront regroupées en couple à l'intérieur de la même composante connexe. L'arête qui les sépare sera utilisée par deux et uniquement deux faces.

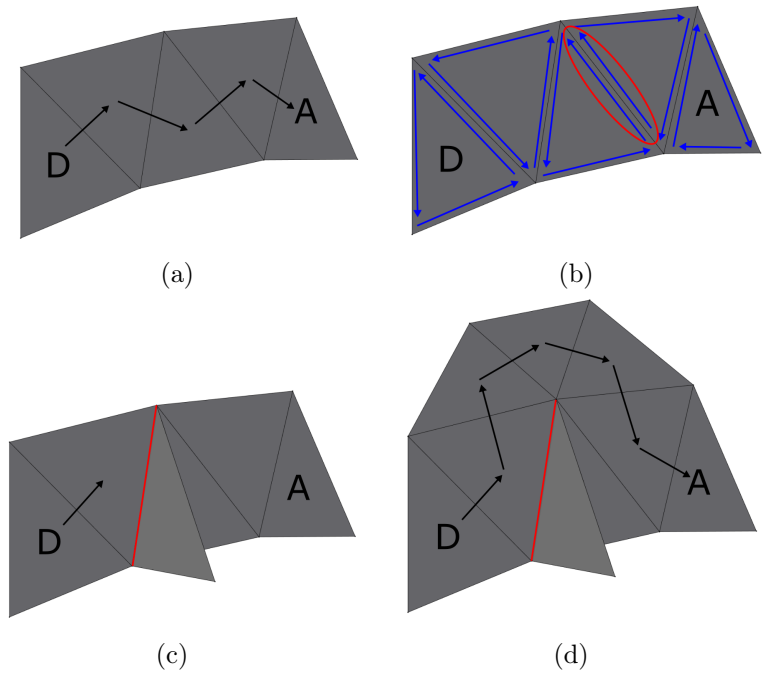


FIGURE 4.2 – (a) Chemin entre les faces D et A, (b) Le chemin existe quelque soit l'orientation des faces, (c) Le chemin n'existe pas, car une arête condition de non-variété (en rouge) bloque, (d) Chemin entre les faces D et A malgré l'arête condition de non-variété (en rouge).

Toutes les faces utilisant l'arête condition de non-variété qui n'ont pas un tel chemin sont séparées dans des composantes connexes différentes. Leurs arêtes deviennent une arête de bord.

4.2.1 Description de la structure temporaire

À partir du fichier au format OBJ on crée une structure temporaire qui contient :

- Une liste de faces, chaque face est définie par :
  - Les identifiants de ses trois sommets.
  - Ces identifiants sont ordonnés en fonction de l'orientation de la face.
  - La position de ses arêtes dans le tableau d'arêtes trié.
  - L'information sur sa couleur et/ou texture.
- Un tableau de demi-arêtes trié, chaque demi-arête est définie par :
  - La face dont elle provient.
  - Les identifiants de ses deux sommets extrémités.
  - Ces identifiants sont ordonnés par ordre croissant.

— Un booléen définissant l'orientation de l'arête.

— Une liste de sommets, chaque sommet est défini par :

— Sa position dans l'espace (x,y,z).

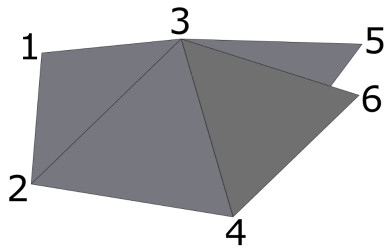
— Un identifiant unique.

— La liste des faces l'utilisant.

Les arêtes ne sont pas ici une frontière entre plusieurs faces, mais un élément d'une seule face (demi-arête). Il y aura dans la liste autant d'occurrences d'une arête sous la forme de demi-arêtes, que de faces l'utilisant. Les différentes occurrences ne sont pas identiques, la face est différente pour chacune, et l'orientation varie, mais les identifiants des deux sommets sont identiques.

Les identifiants de sommet sont ordonnés en fonction de leurs valeurs, le premier identifiant est celui qui à la plus petite valeur et le deuxième identifiant est celui qui à la plus grande. L'orientation de la demi-arête n'est donc pas donnée par l'ordre de ses sommets, comme c'est le cas pour la face. Un booléen conserve cette information, en indiquant si l'orientation de la demi-arête va du sommet ayant le premier identifiant vers le sommet ayant le deuxième identifiant ou l'inverse. Cette orientation est directement issue de l'orientation de la face dont elle provient.

Le tableau de demi-arêtes est trié (Fig.4.3). D'abord par ordre croissant de leurs premiers identifiants puis s'il est identique par ordre croissant de leurs deuxièmes identifiants. L'idée est que les demi-arêtes issues de la même arête soient placées à côté dans le tableau. Comme les identifiants sont ordonnés, c'est le cas, quelle que soit l'orientation de l'arête.



(a)

1	1	2	2	2	3	3	3	3	3	4	4
2	3	3	3	4	4	4	4	5	6	5	5

(b)

FIGURE 4.3 – Exemple d'un tableau trié des demi-arêtes : Le tableau contient deux demi-arêtes (2,3), l'arête (2,3) est donc une arête ayant deux faces voisines. Le tableau contient trois demi-arêtes (3,4), l'arête (3,4) est donc une arête condition de non-variété. Toutes les autres demi-arêtes n'ont qu'une occurrence dans le tableau, les autres arêtes sont donc des arêtes de bord.

4.2.2 Algorithme de parcours de recherche des composantes connexes

La recherche de composante connexe est effectuée en parcourant les faces du modèle 3D (Algo.1). Le parcours commence sur une face qu'on attribue à une première composante connexe. Puis pour chaque face de la composante connexe, on y ajoute ses faces voisines, sauf si elles sont déjà ajoutées. Lorsque toutes les faces de la composante connexe ont été traitées, on vérifie s'il reste des faces n'appartenant à aucune composante connexe. S'il en reste alors on refait un parcours à partir d'une de ces faces en l'attribuant à une nouvelle composante connexe. Le parcours s'arrête lorsque toutes les faces ont été attribuées à une composante connexe.

Algorithme 1 Recherche des composantes connexes et de l'orientation de leurs faces

Entrées: F, la liste des faces du modèle 3D

Sorties: F', la liste des faces étiquetées deux fois(par le numéro de la CC et par une orientation)

```
1: idCC = 0 //Numéro de la CC
2: tant que toute les faces de F ne sont pas étiquetée faire
3:    f = première face de F non étiquetées
4:    Étiqueter(f,idCC)
5:    Étiqueter(f,+1) //Une des deux orientations possible
6:    fCCPile = nouvellePile() //Pile de faces pour une CC
7:    empiler(fCCPile,f)
8:    tant que fCCPile n'est pas vide faire
9:       f = dépiler(fCCPile)
10:      voisines = rechercheFaceVoisines(f) //Voir Algo.2
11:      pour chaque faceVoisine de voisines faire
12:         si faceVoisine n'est pas étiquetée alors
13:            Étiqueter(faceVoisine,idCC)
14:            definirOrientation(f,faceVoisine) //Voir Algo.6
15:            empiler(fCCPile,faceVoisine)
16:         fin si
17:      fin pour
18:   fin tant que
19:   idCC++
20: fin tant que
```

La recherche des faces voisines d'une face se fait en cherchant pour chacune de ses demi-arêtes, s' il existe une demi-arête jumelle (Algo.2). On appelle jumelle deux demi-arêtes, si elles ont des identifiants de sommets identiques et s'il n'existe pas d'autre demi-arête respectant cette condition. Les faces

voisines sont les faces utilisant les demi-arêtes jumelles des demi-arêtes de la face initiale.

Algorithme 2 Recherche des faces voisines, rechercheFaceVoisines(f)

Entrées: A, tableau trié des demi-arêtes du modèle 3D ;
f, une face
Sorties: facesVoisines, les faces voisines de f

```
1: pour chaque demiArrête a de f faire
2:    demiAreteJumelle = rechercheDemiAreteJumelle(a,A) //Voir Algo.3
3:    si demiAreteJumelle existe alors
4:       faceVoisine = face(demiAreteJumelle)
5:       Ajout de faceVoisine dans facesVoisines
6:    fin si
7: fin pour
```

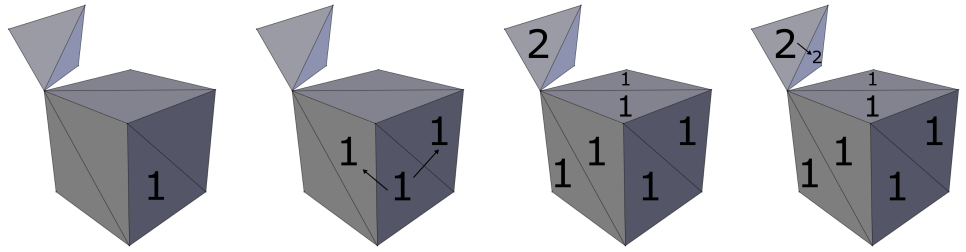
La recherche d'une demi-arête jumelle ce fait en utilisant le tableau des demi-arêtes trié (Algo.3). Les demi-arêtes ayant les mêmes identifiants de sommet se trouvent forcément dans des cases consécutives du tableau. La position dans le tableau de la demi-arête est donnée par la face dont on cherche les voisines. À partir de cette position, on parcourt le tableau dans les deux directions pour compter le nombre de demi-arêtes ayant les mêmes identifiants de sommet. S'il en existe qu'une autre alors celle-ci est définie comme l'arête jumelle et sa face comme une face voisine de la face l'initiale.

Algorithme 3 Recherche une arête identique si elle existe et est unique, rechercheDemiAreteJumelle(a,A)

Entrées: A, tableau trié de demi-arêtes ;
a, une demi-arête
Sorties: demiAreteJumelle, demi-arête jumelle à a si elle existe

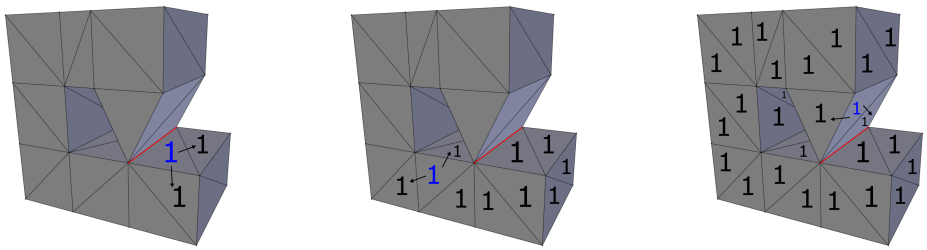
```
1: iAreteSuivante = indice(a) + 1 // indice(a), indice de a dans A
2: iAretePrecedente = indice(a) - 1
3: tant que A[iAreteSuivante] == a faire
4:     iDemiAreteJumelle = iAreteSuivante
5:     iAreteSuivante ++
6: fin tant que
7: tant que A[iAretePrecedente] == a faire
8:     iDemiAreteJumelle = iAretePrecedente
9:     iAretePrecedente ++
10: fin tant que
11: si iAreteSuivante - iAretePrecedente == 1 alors
12:     demiAreteJumelle = A[iDemiAreteJumelle] // Il n'existe qu'une demi-
        arête ayant les mêmes identifiants de sommet que a
13: sinon
14:     DemiAreteJumelle = null // Il n'existe aucune ou plus d'une demi-arête
        ayant les mêmes identifiants de sommet que a
15: fin si
```

La notion de face voisine étant réciproque, quelles que soient les faces choisies pour commencer une composante connexe, la répartition des faces sera identique. Deux exemples de recherche de composantes connexes sont illustrés sur les figures suivantes (Fig.4.4)(Fig.4.5).



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

Figure 4.4 – Parcours de recherche des composantes connexes : (a) On attribue la CC 1 à une face, (b) La face propage sa CC à ses voisines, (c) On attribue la CC 2 à une face restante après le premier parcours, (d) La face propage sa CC à ses voisines.



(a) (b) (c)

FIGURE 4.5 – Parcours de recherche des composantes connexes : (a) La face (en bleu) propage sa CC à ses voisines, mais pas aux faces connexes à l'arête de non-variété (en rouge), (b)(c) La face (en bleu) propage sa CC à ses voisines ce qui inclut une face connexe à l'arête de non-variété (en rouge).

4.3 Correction de conditions de non-variété

4.3.1 Définition d'une surface non-variété

Une surface est une 2D-variété sans bord si en tout point p de sa surface, il existe une distance d pour laquelle, tous les points de la surface éloignés d'une distance inférieure à d de p forme un disque. Une surface est une 2D-variété avec bord si en certains points p, ces points forment un demi-disque. Une surface est une non-variété si aucune des deux définitions précédentes ne lui correspond.

Un élément condition de non-variété est un élément de la surface (sommet,arête) pour lequel tous les points lui appartenant ne remplissent pas les conditions des deux premières définitions.

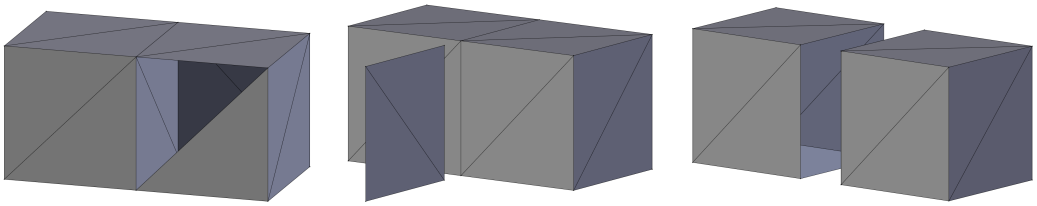
4.3.2 Pourquoi corriger les non-variétés ?

Dans le cadre d'une impression 3D, la correction des non-variétés n'est pas obligatoire. Le *Slicer* traite les faces séparément les unes des autres sans utiliser les informations de connexité entre les éléments, il n'a pas de notion de surface et donc de non-variétés. Mais pour que le *Slicer* interprète le bon volume à imprimer, il faut quand même que ce volume soit défini et donc entouré par une ou des surfaces variétés intersectées.

Par exemple une condition de non-variété sur une arête commune à quatre faces, peux être correctement interprété par le *Slicer*, si l'arête est l'intersection entre deux surfaces variété. À l'inverse sur une condition de non-variété sur une arête commune à trois faces, quelle que soit l'interprétation il y aura au moins une surface avec un bord et donc pas de volume défini (Fig.4.6).

Les informations de connexité sont utiles pour les différentes étapes de la chaîne de réparation (orientation de la surface, détection de contour, remplissage d'un trou, etc ...). Comme les non-variétés ajoutent des choix multiples

sujets à interprétation dans les connexions entre les éléments (Fig.4.6). Lever ces interprétations en corrigeant les non-variétés en début de chaîne, rendra les étapes suivantes plus simples.



(a) (b) (c)

FIGURE 4.6 – Face interne : (a) Avant réparation (une face est cachée pour voir la face interne), (b) Réparation possible : un parallélépipède et une surface plane, (c) Réparation possible : Deux cubes dont un a une face manquante, (b)(c) Déplacement des éléments pour la visualisation.

4.3.3 Comment corriger les non-variétés ?

La correction des non-variétés est faite en modifiant sur le maillage les connexités entre les éléments (sommets, arêtes et faces). Après correction toutes les arêtes du maillage sont utilisées par uniquement une ou deux faces.

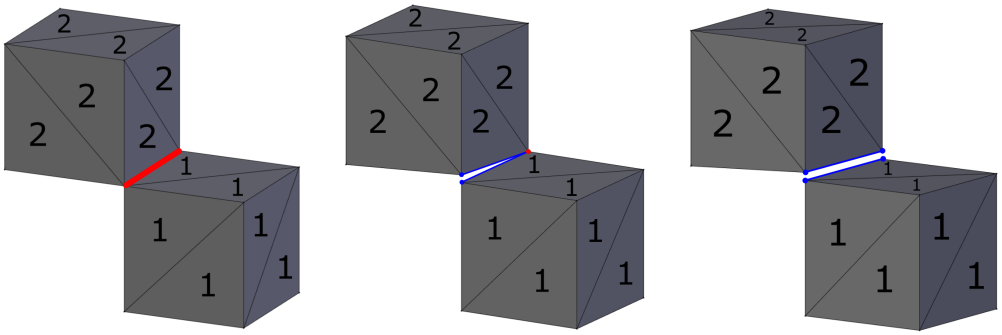
Un parcours du maillage en utilisant la connexité entre les éléments ne détectera pas d'éléments condition de non-variété. Mais comme aucun élément n'est déplacé, géométriquement le modèle 3D ne change pas et la surface reste une non-variété géométrique. Comme vu précédemment, ce n'est pas un problème pour le *Slicer* de l'imprimante.

Pour corriger une condition de non-variété (sommets, arêtes), il faut en créer plusieurs occurrences puis les répartir entre les différentes faces adjointes à l'élément condition de non-variété. La répartition ce fait de manière à qu'après répartition aucune des occurrences ne soit condition de non-variété.

Dans le cas d'une arête, créer une occurrence de ces sommets extrémités et attribuer cette paire à une ou plusieurs faces créera une occurrence de l'arête initiale (Fig.4.7). Pour corriger les éléments condition de non-variété, on ne créera que de nouveaux sommets, les arêtes seront créées implicitement par les faces utilisant ces nouveaux sommets.

4.3.4 Notre approche

Pour chaque sommet, les faces qui utilisent ce sommet sont divisées en groupe. Pour chaque groupe de faces on crée une occurrence du sommet.

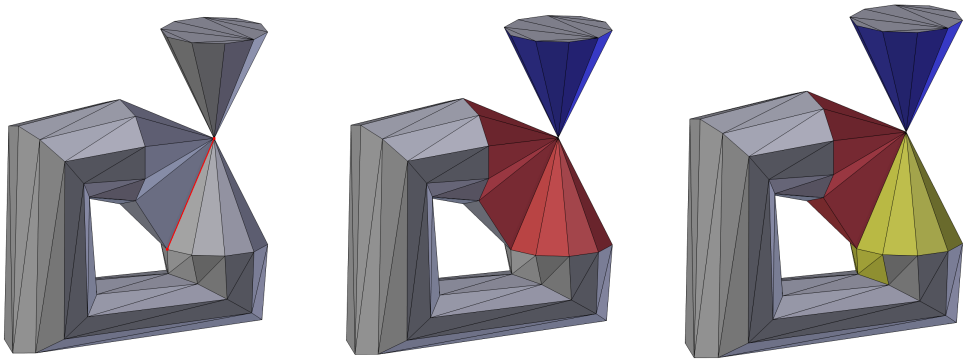


(a) (b) (c)

FIGURE 4.7 – Étape de dédoublement d'une arête : (a) Non-variété sur une arête, (b) Dédoublement d'un des sommets extrémités, avec répartition des doublons entre les faces de la composante connexe 1 et 2, (c) Dédoublement du deuxième sommet extrémité, (b)(c) Déplacement des sommets pour la visualisation.

Chaque groupe de face utilisera une occurrence différente du sommet. Le premier groupe utilisera le sommet original (Algo.4).

La répartition en groupe ce fait en deux étapes (Fig.4.8). La première est de diviser les faces en groupe par composante connexe. Pour la deuxième étape chaque groupe va être redivisé en groupe de faces formant un même cycle autour du sommet. Un cycle est une suite de faces voisines utilisant un même sommet, il peut être fermé (la première et la dernière face sont voisines) ou non.



(a) (b) (c)

FIGURE 4.8 – Étape de répartition des faces selon le sommet qu'elles utiliseront : (a) Modèle avec un sommet et une arête non-variété en rouge, (b) Répartition des faces après la subdivision par composante connexe, (c) Répartition des faces après la subdivision par cycle.

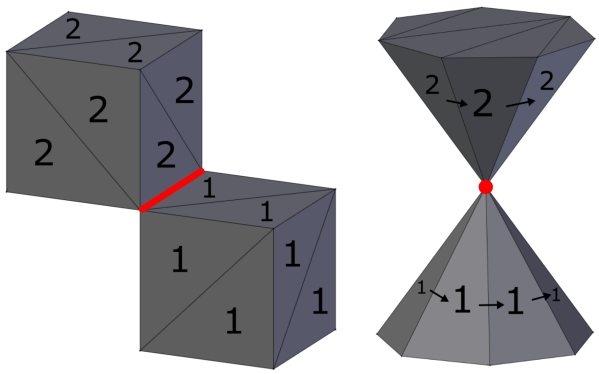
4.3.5 Première étape : Répartition par composantes connexes

La première étape de correction de non-variété découle directement de la répartition des faces en composantes connexes.

Si une arête est élément d'une variété alors il y a deux et seulement deux faces qui utilisent cette arête. Les deux faces sont voisines et appartiennent donc à la même composante connexe. Si une arête appartient à des faces de composantes connexes différentes, alors les faces sont forcément plus de deux et l'arête est donc élément de non-variété (Fig.4.9,(a)).

Si un sommet est une variété, alors l'ensemble des faces auquel il appartient forme un cycle de faces voisines. Si un sommet appartient à des faces de composantes connexes différentes, alors il y a au moins deux de ces faces pour lesquelles il n'existe pas de chemin de l'une à l'autre. Toutes les faces du sommet ne forment donc pas un unique cycle et le sommet est donc condition de non-variété.

On en conclut que si un sommet ou une arête appartient à deux composantes connexes ou plus alors ils sont condition d'une non-variété (Fig.4.9,(b)). C'est pour lever cette condition que les faces appartenant à des composantes connexes différentes et utilisant un même sommet sont placées dans des groupes différents (Algo.4).



(a) (b)

FIGURE 4.9 – Élément de non non-variété entre deux composantes connexes : (a) Sur une arête, (b) Sur un sommet.

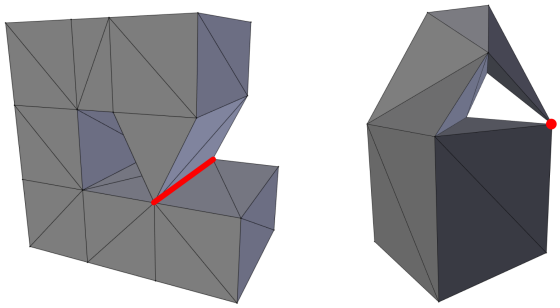
Algorithme 4 Répartition des faces en groupe utilisant la même occurrence d'un sommet

Entrées: un sommet s ;
fSommet, faces utilisant le sommet s
Sorties: groupesFacesFinale, groupes de faces utilisant la même occurrence du sommet s

1: **pour chaque** face f ∈ fSommet **faire**
2: Ajout de f dans le groupe de sa CC dans groupesFacesParCC
3: **fin pour**
4: **pour chaque** groupe de face facesCC ∈ groupesFacesParCC **faire**
5: **tant que** ∃ face ∈ facesCC tel que face ∉ facesCycle, ∀ facesCycle ∈ groupeFacesParCycle **faire**
6: f = une face de facesCC
7: facesCycle = rechercheCycle(s,f,facesCC) *//Voir Algo.5)*
8: Ajout du groupe de face facesCycle dans groupeFacesParCycle
9: **fin tant que**
10: **fin pour**
11: groupesFacesFinale = groupeFacesParCycle

4.3.6 Deuxième étape : Répartition par cycle

La répartition par cycle va rediviser les groupes de faces issues de la répartition par composante connexe en sous-groupe. Le but est de corrigé les éléments de non-variété appartenant qu'à une seule composante connexe (Fig.4.10).



(a) (b)

Figure 4.10 – Élément condition de non-variété dans une seule composante connexe : (a) Sur une arête, (b) Sur un sommet.

Pour chaque groupe de face, on recherche les cycles autour du sommet initial en utilisant que des faces du groupe (Algo.5). Les faces d'un même cycle sont groupées ensemble dans un sous-groupe. Ces sous-groupes sont la répartition finale utilisée pour dupliquer le sommet.

La recherche d'un cycle est un parcours de faces voisines autour du sommet initial. Il commence par une face quelconque du groupe qui est ajoutée au cycle.

Cette face possède deux demi-arêtes utilisant le sommet initial. On cherche s'il existe une demi-arête jumelle à une première de ces deux demi-arêtes. Si elle existe, la face lui étant attachée est une face voisine et fait partie du cycle. Cette face voisine est ajoutée au cycle puis on réitère en cherchant s'il existe une demi-arête jumelle à l'autre demi-arête utilisant le sommet initial de la face voisine. Le parcours s'arrête lorsque la nouvelle face voisine est la face de départ du parcours. Dans ce cas le cycle est fermé (Fig.4.11,(a)).

Si lors du parcours une demi-arête n'a pas de jumelle alors on relance le parcours, mais en utilisant la deuxième demi-arête de la face de départ. Ce deuxième parcours s'arrêtera quand une nouvelle demi-arête n'aura pas de jumelle, le cycle sera alors fini. Dans ce cas le cycle peut être fermé ou ouvert (Fig.4.11,(b)(c)). Une demi-arête n'a pas de jumelle si son arête est une arête de bord ou une arête condition de non-variété dans une seul composante connexe.

Les définitions d'une face voisine et d'une demi-arête jumelle sont utilisées en ne prenant en compte que les faces du groupe et non toutes les faces du modèles. Les éléments de non-variété corrigés par la répartition par composante connexe ne sont ainsi pas pris en compte.

Quand un cycle est fini si des faces du groupe ne sont pas encore dans un sous-groupe, on cherche un nouveau cycle à partir d'une de ces faces (Fig.4.12).

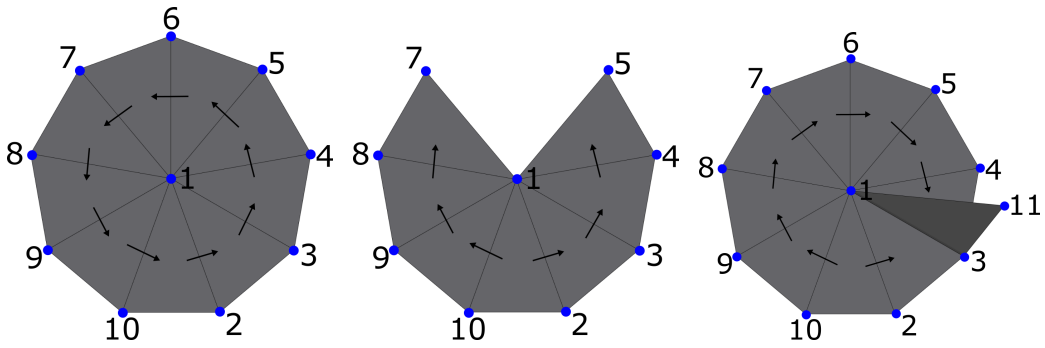
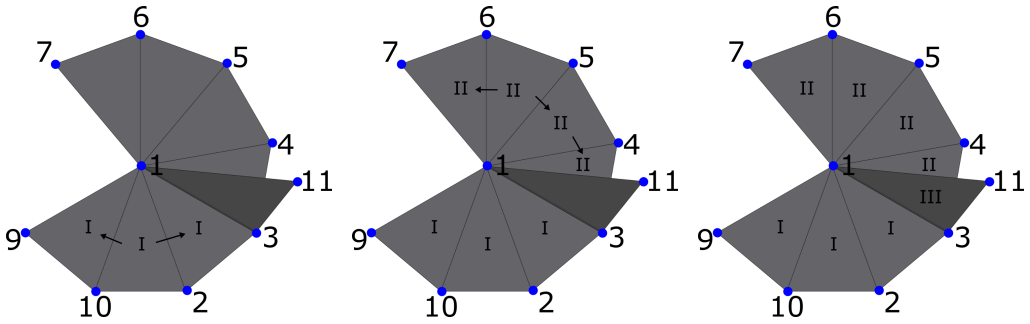


FIGURE 4.11 – Différents types de cycle : (a) Cycle complet, le parcours boucle sur lui même, (b) Cycle avec bord, le parcours change de sens après la rencontre avec le premier bord, (c) Cycle avec une arête non-variété, le parcours change de sens après la rencontre avec l'arête de non-variété.



(a) (b) (c)

FIGURE 4.12 – Étape d'une recherche de cycle : (a) Premier cycle stoppé par un bord (1,9) dans un sens et par une arête de non-variété (1,3) dans l'autre, (b) Deuxième cycle stoppé par un bord (1,7) dans un sens et par une arête de non-variété (1,3) dans l'autre, (c) Troisième cycle stoppé par un bord (1,11) dans un sens et par une arête de non-variété (1,3) dans l'autre.

Algorithme 5 Trouver un cycle, rechercheCycle(s,f,facesCC)

ENTRÉES: s, sommet ; f, face utilisant s ;
facesCC, groupe de faces utilisant s et de la même CC

SORTIES: facesCycle, groupe de faces utilisant s, de la même CC et du même cycle

1: tabDemiAreteCC = tableau trié des demi-arêtes des faces ∈ facesCC
2: Ajout de f dans facesCycle
3: a1 = demi-arête de f entrant dans s
4: a2 = demi-arête de f sortant de s
5: areteSuivante = rechercheDemiAreteJumelle(a1,tabDemiAreteCC)
6: faceSuivante = face(areteSuivante)
7: **tant que** areteSuivante != null && faceSuivante ∉ facesCycle **faire**
8: Ajout de faceSuivante dans facesCycle
9: a1 = demi-arête de faceSuivante utilisant s avec a1 != areteSuivante
10: areteSuivante = rechercheDemiAreteJumelle(a1,tabDemiAreteCC)
11: faceSuivante = face(areteSuivante)
12: **fin tant que**
13: **si** areteSuivante == null && faceSuivante ∉ facesCycle **alors**
14: areteSuivante = rechercheDemiAreteJumelle(a2,tabDemiAreteCC)
15: **tant que** areteSuivante != null **faire**
16: faceSuivante = face(areteSuivante)
17: Ajout de faceSuivante dans facesCycle
18: a2 = arête de faceSuivante utilisant s différente de areteSuivante
19: areteSuivante = rechercheAreteJumelle(a2,tabDemiAreteCC)
20: **fin tant que**
21: **fin si**

4.3.7 Remarque : Pourquoi deux étapes ?

La correction des conditions de non-variété sur les sommets et sur les arêtes se fait en deux étapes. Les faces d'un même sommet sont réparties dans une première étape sur le critère de l'appartenance à une même composante connexe, puis dans une deuxième étape sur le critère de l'appartenance à un même cycle. Mais répartir les faces seulement par cycle suffirait pour lever toutes les conditions de non-variété.

Pour lever une condition de non-variété sur une arête, celle-ci doit être remplacée par plusieurs arêtes soit de bord soit normal. Le but de la chaîne de réparation étant à terme de ne plus avoir d'arête de bord, il est donc préférable d'en créer le moins possible et de privilégier la création d'arêtes ayant deux faces voisines.

Une arête ayant deux faces voisines est créée si deux faces utilisant une même arête condition de non-variété sont répartie dans un même groupe une des deux étapes. Dans la répartition par composante connexe c'est le cas s' il existe un chemin entre ces deux faces. Mais la répartition par cycle ne travaillant que sur les faces utilisant un même sommet, peut faire un tel regroupement seulement si le chemin entre les deux faces passe uniquement par des faces du cycle (cycle fermé).

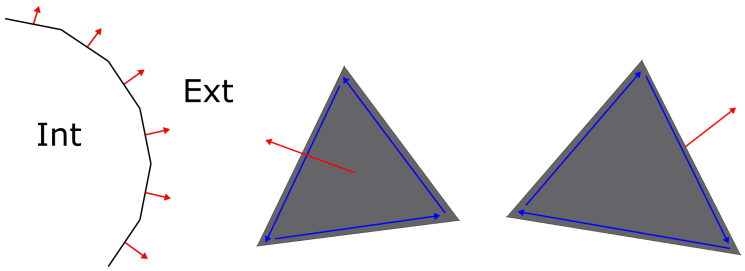
Le groupement par composante connexe en première étape permet de créer plus d'arêtes ayant deux faces voisines et moins d'arêtes de bord.

4.3.8 Qu'est-ce que l'orientation ?

Une surface représente la limite entre l'extérieur et l'intérieur d'un objet. Son orientation détermine de quel côté de cette surface se trouvent l'intérieur et l'extérieur (Fig.4.13,(a)). Il n'y a donc que deux choix d'orientation possibles. L'orientation est représentée au niveau d'une face par une normale avec une direction perpendiculaire au plan de la face et une orientation vers l'extérieur (Fig.4.13,(b)). Une surface a une orientation cohérente si chaque face de cette surface possède la même orientation que ses voisines (Fig.4.14,(a)).

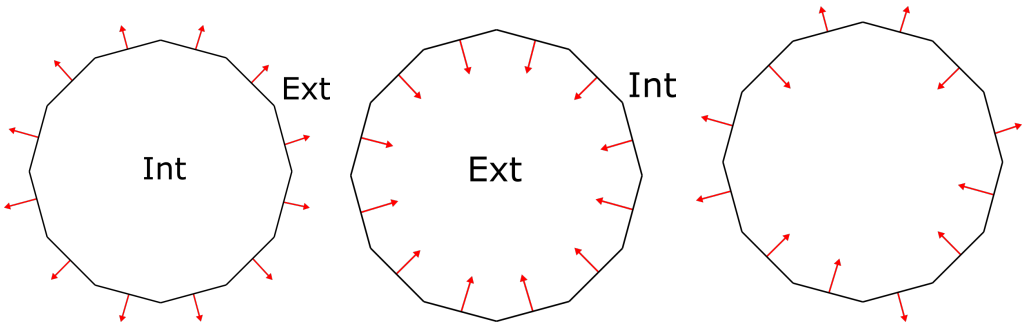
4.3.9 Pourquoi corriger l'orientation de surface

Si une surface n'a pas une orientation cohérente, l'imprimante ou le *Slicer* de l'imprimante ne pourra pas déterminer correctement où se trouve le volume à imprimer (Fig.4.14,(c)), ce qui risque de provoquer des artefacts comme de la construction interne visible, voire de l'ajout de matière. Si une surface a une orientation cohérente, mais avec une orientation vers l'intérieur, alors l'imprimante considère le volume délimité comme vide. Ce qui peut ne provoquer aucune impression si la surface est seule ou un creux si elle intersecte d'autres volumes de l'objet (Fig.4.14,(b)).



(a) (b)

FIGURE 4.13 – (a) Surface orientée : les normales des faces sont dirigées vers l'extérieur de l'objet, (b) Faces orientées : l'orientation de la normale dépend du sens de ses arêtes. La normale est dirigée vers la moitié de l'espace où les arêtes de la face tournent dans le sens trigonométrique si on les observe à partir du sommet de cette normale.



(a) (b) (c)

FIGURE 4.14 – Surface fermée avec différente orientation : (a) Surface avec une orientation cohérente, définissant un volume plein fini(Int), (b) Surface avec une orientation cohérente, définissant un vide fini(Ext), (c) Surface avec une orientation non-cohérente, impossible d'interpréter où se trouve l'intérieur ou l'extérieur.

Lors de l'étape précédente, on réalise un parcours des faces pour notamment identifier les composantes connexes (Algo.1). Lors de ce même parcours, on en profite pour regarder l'orientation des faces parcourues. Pour rappel, dans la structure temporaire les arêtes ont les deux identifiants de leurs sommets ordonnés par ordre croissant. L'information d'orientation est donnée par un booléen. Elle est fausse si l'orientation va du plus petit identifiant au plus grand et elle est vraie pour l'inverse. Ainsi deux faces voisines ont la même orientation si leurs arêtes communes ont les mêmes identifiants de sommets, mais un booléen différent (Fig.4.15,(a)). Inversement si elles ont un booléen identique, leurs orientations sont opposées (Fig.4.15,(b)).

Lors du parcours pour la construction des composantes connexes, on va attribuer à chaque face une couleur/identifiant selon son orientation. Il n'y a que deux orientations possibles, mais on distinguera deux orientations pour

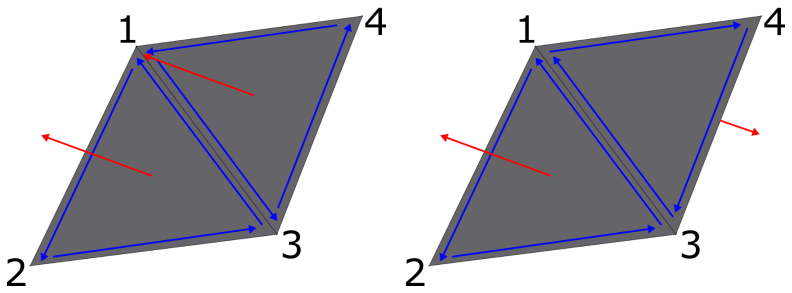


FIGURE 4.15 – Orientation des normales de deux faces voisines : (a) Les deux normales ont une orientation identique, les arêtes communes (1,3) ont un sens inversé entre les deux faces, (b) Les deux normales ont une orientation opposée, les arêtes communes (1,3) ont le même sens sur les deux faces.

chaque composante connexe. L'attribution se fait arbitrairement pour la première face de la composante connexe. Pour les suivantes elle se fait au moment de leurs rajouts dans la composante connexe (Algo.6). Si elles sont rajoutées, c'est qu'elles sont voisines à une face de la composante qui a donc déjà une orientation attribuée. On regarde alors les arêtes entre ces deux faces pour voir si la nouvelle face va prendre l'orientation dans l'ancienne ou son inverse (Fig.4.16).

Pour chaque composante connexe, on crée un compteur pour le nombre de faces qui possèdent la première orientation et un deuxième compteur pour le nombre de faces qui possèdent la deuxième orientation. Le compteur de la première orientation est incrémenté pour la première face de la composante connexe. Les compteurs sont ensuite incrémentés à chaque attribution d'une orientation à une face leur correspondant.

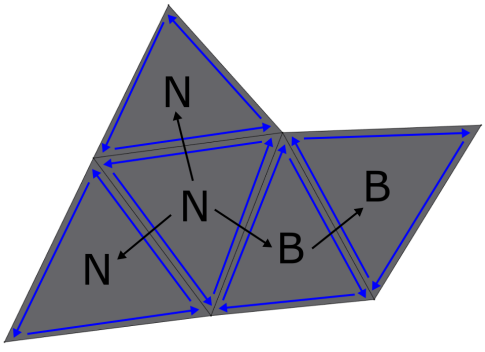


FIGURE 4.16 – Propagation des orientations des faces : Noir et Blanc désignent les deux orientations possibles, on part arbitrairement d'une face Noire puis on propage à ses voisines la même orientation ou son inverse suivant le sens des arêtes communes aux deux faces.

Algorithme 6 Définition de l'orientation de la face ajoutée à la composante connexe(definirOrientation)

Entrées: f, face avec une étiquette d'orientation ;
faceVoisine, face voisine de f sans étiquette d'orientation
Sorties: faceVoisine', face faceVoisine étiquetté (par son orientation) ;
compteursOrientation, double compteur des orientations des faces de la CC

```
1: a1 = arêteCommmune(f,faceVoisine)
2: a2 = arêteCommmune(faceVoisine,f)
3: si a1 et a2 ont une orientation différente alors
4:    Étiquetter faceVoisine avec orientation(f)
5: sinon si a1 et a2 ont la même orientation alors
6:    Étiquetter faceVoisine avec inverseOrientation(f)
7: fin si
8: compteursOrientation[orientation(faceVoisine)]++
```

4.3.10 Réparation de l'orientation des faces

La réparation se fait lors du passage de la structure temporaire à la structure Halfedge. Au moment de l'ajout d'une face dans la structure *Halfedge*, on regarde si l'orientation de la face est la même que celle choisie pour sa composante connexe. Si c'est le cas, alors on ajoute les sommets dans le même ordre que ce qui a été enregistré dans la structure temporaire, sinon on les ajoute dans l'ordre inverse. Une partie des informations d'apparence sont conservées dans les faces de la structure *Halfedge*. Les identifiants des coordonnées de textures attachés aux sommets suivent le même processus que leur sommet.

Le choix est fait de réparer "à la majorité" par composante connexe. L'orientation choisie pour une composante connexe est celle majoritaire parmi ses faces avant réparation. L'algorithme qui définit l'orientation des faces incrémente également des compteurs pour les deux orientations (Algo.6). L'orientation majoritaire est trouvée en les comparant.

Cette méthode permet de réparer les composantes connexes où l'orientation est définie, mais avec quelques erreurs. Pour rappel, l'attribution d'une face à une composante connexe ne prend pas en compte son orientation initiale.

Deux exemples où la réparation "à la majorité" n'est pas une bonne solution :

— Si toute une composante connexe est orientée initialement vers l'intérieur. Cela peut se produire lorsque le modeleur fait un effet miroir sur un élément, par exemple les jambes ou les yeux d'un personnage (Fig.4.17,(b)). Dans ce cas la composante connexe est correctement orientée, mais son orientation va poser un problème à l'impression, l'imprimante va consi-

dérer le volume défini par cette composante comme un vide.

— Si les orientations des faces sont totalement aléatoires. Dans ce cas le choix à la majorité est lui-même aléatoire (Fig.4.17,(a)). Quel que soit ce choix, l'orientation de la surface reste cohérente, mais on peut se retrouver avec le même problème que le cas précédent, c'est-à-dire une orientation vers l'intérieur représentant un vide.

(a) (b)

FIGURE 4.17 – Exemple d'orientation où la méthode de la majorité n'est pas idéale : (a) Orientation aléatoire des faces sur la sphère, l'orientation à la majorité sera lui aussi aléatoire, (b) Orientation cohérente des faces de la jambe droite, mais définissant un vide, l'orientation est cohérente la méthode de la majorité ne changera pas l'orientation.

4.3.11 Orientation par lancer de rayon

Une autre méthode possible pour choisir l'orientation d'une composante connexe est la suivante. On lance un rayon pour chaque composante connexe avec une origine extérieure à l'objet et avec une direction qui intersectera la surface de la composante connexe. On sélectionne la face intersectée par le rayon dont le point d'intersection est le plus proche de l'origine. L'orientation de cette face est définie de façon à que l'angle entre sa normale et le rayon soit supérieur à 90°. L'orientation de la composante connexe et de toutes ses faces est définie par cette première face intersectée (Fig.4.18,(a)(b)).

Cette solution requiert que la composante connexe soit une 2D-variétés sans bord. Si ce n'est pas le cas deux rayons respectant les conditions citées précédemment peuvent intersecter une même face de deux côtés différents, rendant l'orientation aléatoire (Fig.4.18,(c)). Or à cette étape de la chaîne, on ne garantit pas que la composante connexe soit sans bords. Pour appliquer cette méthode, il faudrait le faire en fin de chaîne, après les étapes de réparation de bord que ce soit par remplissage ou par offset. Et ces étapes ainsi que la structure *Halfedge* nécessite une orientation de surface cohérente. Il faut donc quand même passer par la solution de la majorité dans un premier temps. Pour

ensuite retourner entièrement la surface si besoin avec la méthode par lancer de rayon après avoir réparé les bords.

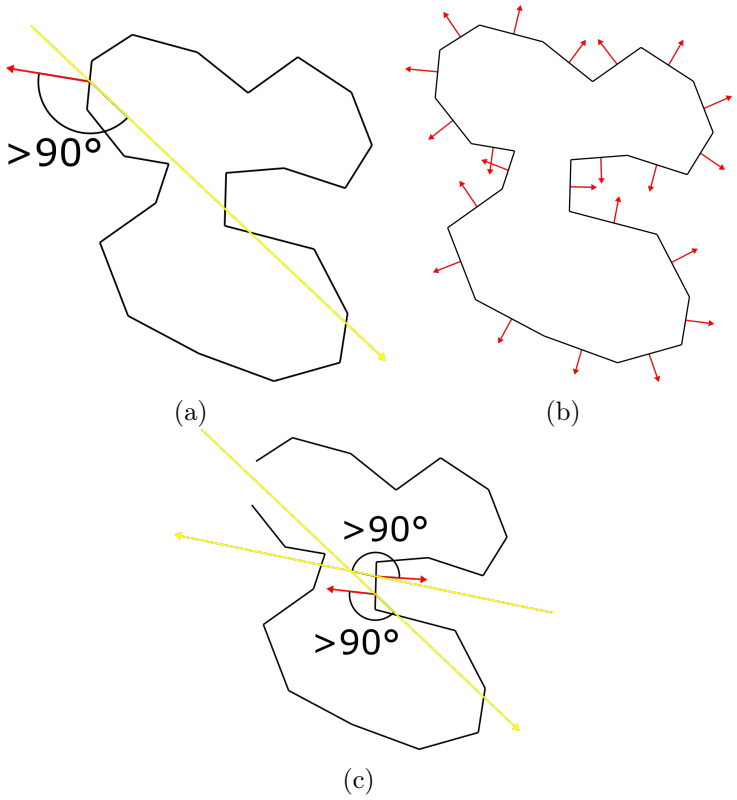


FIGURE 4.18 – Méthode d'orientation d'une surface par lancé de rayon : (a) Première étape, on lance un rayon de l'extérieur vers la surface, on oriente la face de la première intersection avec la normale qui forme un angle de plus de 90°avec le rayon, (b) Deuxième étape, on donne l'orientation de la première face à toute la surface, (c) Si la surface n'est pas entièrement fermée, la méthode ne fonctionne pas, car suivant le rayon utilisé les deux orientations peuvent être trouvées.

Avec la réparation par lancer de rayon, toutes les composantes connexes servent à délimiter un volume plein. Or si le modeleur d'un modèle 3D veut faire un vide dans le volume de son objet, cette réparation va l'empêcher. Dans le cadre de l'impression 3D, ce cas de figure est en réalité rarement utile, car les techniques d'impression ne sont pas faites pour réaliser ce genre d'objets. On peut distinguer pour ce cas deux familles de techniques d'impression (Annexe.A).

Les premières qui ont un support plein, c'est-à-dire qu'à chaque couche la matière est déposée sur l'ensemble de la surface d'impression sur laquelle on va solidifier une tranche de l'objet. Dans ce cas faire un creux ne fera qu'enfermer de la matière non solidifiée/agglomérée. Ce qui n'a peu d'intérêt, car :

— La matière non solidifiée/agglomérée ne peut pas être récupérée pour une prochaine impression.

— Cela fragilise inutilement l'objet, la matière non solidifiée/agglomérée ne participe pas à la solidité de l'objet.

— En cas de casse, la matière non solidifiée/agglomérée risque de se répandre.

Les techniques appartenant à la deuxième famille ont un support construit en fonction de la forme de l'objet pour pouvoir solidifier les tranches supérieures sans poser de la matière dans le vide. Dans ce cas, le vide serait rempli de ce support construit. Or, en général l'intérieur des objets construits par ces méthodes n'est pas plein, mais est composé par exemple d'une structure en nid d'abeille pour économiser de la matière. Enfermer un support construit à peu d'intérêt, car :

— N'utilise pas moins de matière qu'une structure interne, surtout que cela rajoute une surface externe qui utilise plus de matière que la structure interne.

— Fragilise l'objet, car un support construit est moins robuste que la structure interne, car fait pour être enlevé.

Sans compter que la majorité des matières ne sont pas transparentes et donc le vide n'est pas visible sur l'objet imprimé. On peut donc faire cette réparation de façon automatique, et dans les rares cas où un modeleur aurait besoin d'un vide il pourra retourner l'orientation de la surface manuellement après la réparation. Cette manipulation est facilitée par la décomposition en composante connexe conservée lors de l'export dans un format OBJ (Chaque composante connexe est un objet du format OBJ différent).

4.4 Conclusion

À la fin de ce chapitre, le modèle 3D d'entrée est découpé en composantes connexes, après corrections des non-variétés et réorientation des normales aux faces. Chaque composante connexe est devenue une surface variété avec bord. Les chapitres suivants consisteront à corriger les bords pour que les composantes connexes deviennent des surfaces variété sans bord, ce qui permettra leur impression.

Les informations spatiales de ces composantes connexes sont rentrées dans une structure de type *Halfedge* afin de garantir que les étapes de ce chapitre ont bien été réalisées puisque la structure ne peut définir que des surfaces varient avec ou sans bord. (Un cas particulier de non-variété est permis par le *Halfedge* lorsqu' un pincement sur un sommet est cumulé avec des arêtes de bord, par exemple deux faces d'une même composante connexe connectées par

un seul sommet. Cette non-variété est tout de même corrigée dans ce chapitre). Toutes les faces présentent dans le fichier OBJ sont présentes dans la structure *Halfedge*. L'utilisation de cette structure permettra aussi des parcours rapides sur les éléments du modèle 3D pour les étapes des chapitres suivants. Pour les chapitres suivants on appellera les composantes connexes, des surfaces.

Pour simplifier, on peut résumer de la façon suivante. On distingue trois types d'arêtes, les arêtes ayant deux faces voisines utilisées par deux faces, les arêtes condition de non-variété qui sont utilisées par plus de deux faces et les arêtes de bord utilisées par une seule face. Ce chapitre transforme toutes les arêtes condition de non-variété en arêtes ayant deux faces voisines ou en arêtes de bord. Les chapitres suivant seront consacrés à transformer les arêtes de bords en arêtes ayant deux faces voisines.

Chapitre 5

Détection et classification des contours

Après les corrections et les transformations sur les données initiales du modèle lors du chapitre précédent plusieurs caractéristiques sont validées :

— Le modèle est composé d'une ou plusieurs composantes connexes, que l'on peut aussi appeler surface.

— Chaque surface est une 2D-variété avec bord.

— Les arêtes du modèle 3D sont des éléments d'une variété et possiblement des arêtes de bords.

— Les données du modèle 3D sont enregistrées dans une structure type Halfedge.

5.1 Recherche de Contour

La première étape de cette analyse est d'identifier toutes les arêtes de bords du modèle et les regrouper en contour (Algo.7). Un contour est une suite d'arêtes de bord formant une boucle (Fig.5.1). Pour ce faire nous allons utiliser les avantages de la structure Halfedge (Fig.5.2). Dans cette structure chaque arête est représentée par deux demi-arêtes d'orientation opposée.

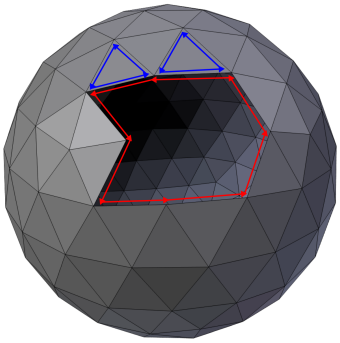
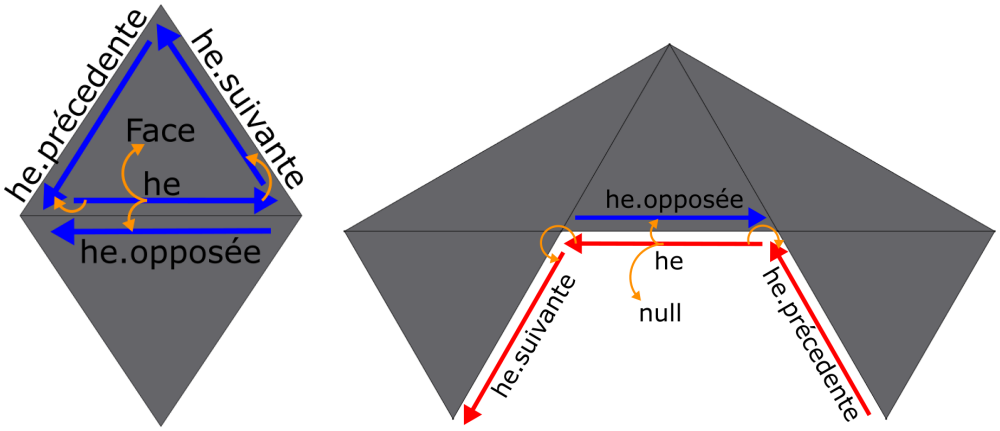


FIGURE 5.1 – Contour : Demi-arêtes d'une face (en Bleu), Demi-arêtes de bords formant un contour (en Rouge).



(a) (b)

FIGURE 5.2 – Arête dans la structure Halfedge : (a) Arête entre deux faces, (b) Arête de bord.

Premier avantage, dans cette structure chaque demi-arête a accès à l'unique face à laquelle elle est attachée. Les arêtes de bords sont celles dont une seule de leurs demi-arêtes est attachée à une face. Il suffit donc de parcourir les demi-arêtes du modèle et de vérifier la non-existence d'une face attachée pour trouver les arêtes de bords (Fig.5.3).

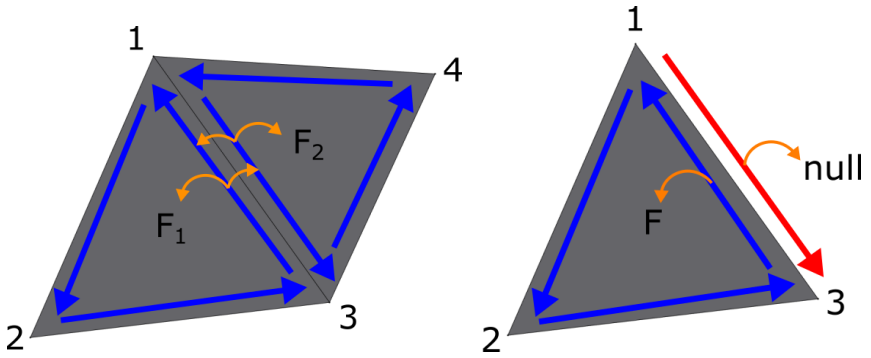


FIGURE 5.3 – Trouver une arête de bord (1,3) : (a) Les deux demi-arêtes ont une face attribuée (F1 pour (3,1) et F2 pour (1,3)). L'arête est élément d'une variété. (b) Une des demi-arêtes n'a pas de face attribué (1,3). L'arête est un bord.

Deuxième avantage : une demi-arête a accès à sa demi-arête suivante. La demi-arête suivante est une demi-arête utilisant comme sommet de départ le sommet d'arrivée de la demi-arête initiale et étant reliée à la même face. Dans le cadre d'une arête de bord, la demi-arête n'ayant pas de face attachée a pour arête suivante une autre demi-arête n'ayant pas de face attachée (Fig.5.2,(b)). Ainsi en passant d'arête suivante en arête suivante sur des arêtes de bords on trace directement un contour (Fig.5.4).

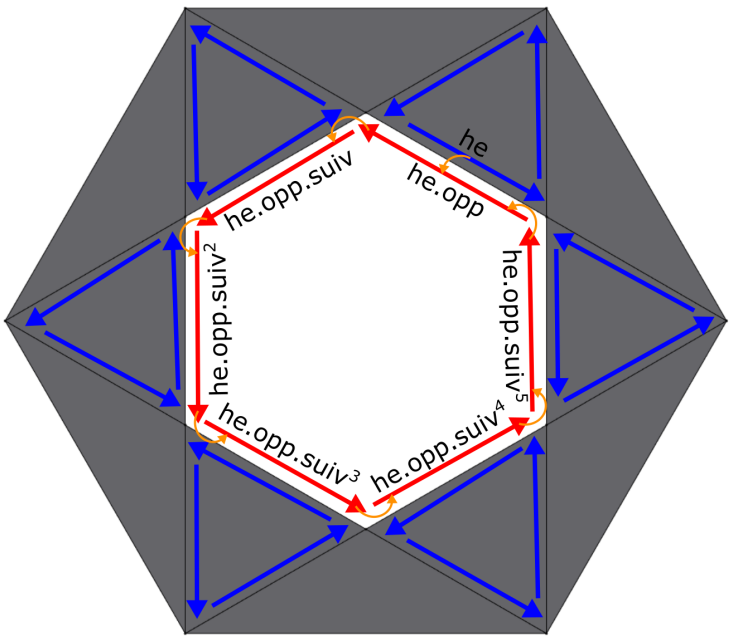


FIGURE 5.4 – Contour dans la structure Halfedge.

Algorithme 7 Recherche de contour

Entrées: HE, la liste des arêtes dans la structure Halfedge
Sorties: contourParCC, liste de contours par composante connexe

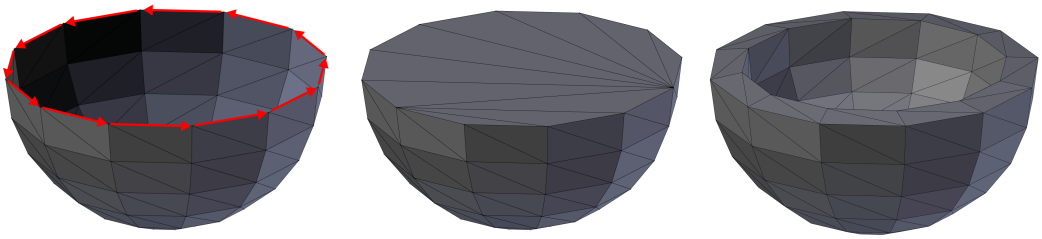
```
1: pour chaque he ∈ HE faire
2:   si he n'est pas marquée alors
3:     Marquer he
4:     si he est une arête de bord alors
5:       idCC = CC(face(areteOpposé(he)) // identifiant de la CC
6:       Ajouter he dans contour // groupe de demi-arêtes consécutives
7:       heFin=he
8:       he = areteSuivante(he)
9:       tant que he!=heFin faire
10:        Marquer he
11:        Ajouter he dans contour
12:        he = areteSuivante(he)
13:      fin tant que
14:      Ajouter contour dans contourParCC[idCC]
15:    fin si
16:  fin si
17: fin pour
```

5.2 Type de réparation

Pour qu'une surface définisse un volume dont elle est la frontière, elle doit être une 2D-variété sans bord. Pour ne plus avoir de bord sur une surface, on distingue trois types de solutions.

La première solution est de fermer le contour en créant de nouvelles faces qui vont utiliser les arêtes du contour (*Ear clipping*, *Geocenter*)(Fig.5.5,(b)). Les arêtes du contour utilisées par une deuxième face ne sont alors plus de bord. Ces solutions seront abordées dans le chapitre (Chap.6).

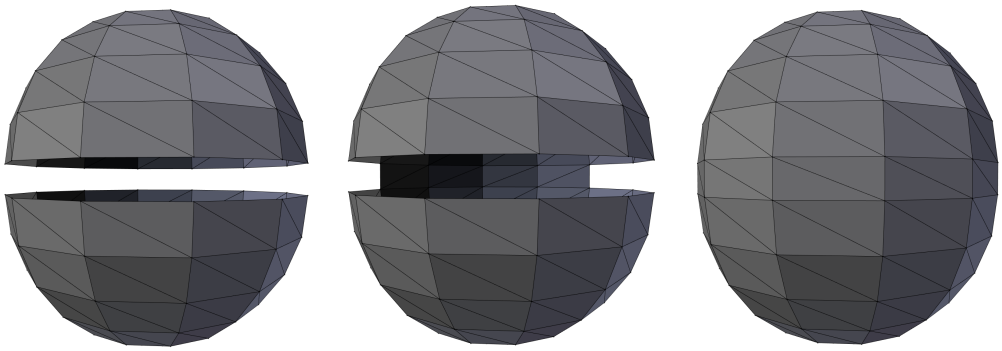
Une deuxième solution consiste à donner un volume à la surface en réalisant une extrusion de cette dernière (*Offset*)(Fig.5.5,(c)). Cette solution a la particularité de corriger tous les contours de la surface en même temps. Un choix de priorité doit donc être décidé entre ce type de réparation et un autre pour chaque contour. Ces solutions seront abordées dans le chapitre (Chap.7).



(a) (b) (c)

FIGURE 5.5 – Type de réparation d'un contour : (a) Contour, (b) Réparation par remplissage, (c) Réparation par épaississement.

La dernière solution est de relier deux contours, soit en fusionnant des éléments de ceux-ci, soit en rajoutant des faces entre les deux (Fig.5.6). Les deux contours peuvent appartenir à deux composantes connexes différentes. Ce dernier type de réparation n'est pour le moment pas utilisé, mais pourrait apporter une amélioration surtout que pour former des variétés lors de l'import on divise parfois en plusieurs composantes connexes autour d'une arête non-variété. Utiliser cette méthode permettrait d'en regrouper certaines, mais nécessiterait une interprétation pour savoir quel contour regrouper quand plusieurs sont possibles.



(a) (b) (c)

FIGURE 5.6 – Étape d'une réparation reliant deux contours proches.

5.2.1 Qualité d'une réparation

Les solutions vues précédemment peuvent être appliquées sur n'importe quel contour, mais certaines solutions conviennent mieux à certains contours qu'à d'autres. Pour juger la qualité d'une réparation, le nombre de bords restants après réparation n'est pas suffisant, car celui-ci peut être nul sur toutes les méthodes. Le point le plus important est le nombre d'auto-intersections et surtout d'auto-intersections provoquant une surface non orientable. Dans la même idée, plus l'épaisseur locale proche de la réparation est faible, moins la

réparation est qualitative. On peut considérer l'auto-intersection comme une épaisseur négative. Enfin la qualité de la réparation dépend aussi d'un critère visuel qui doit rester proche du modèle original.

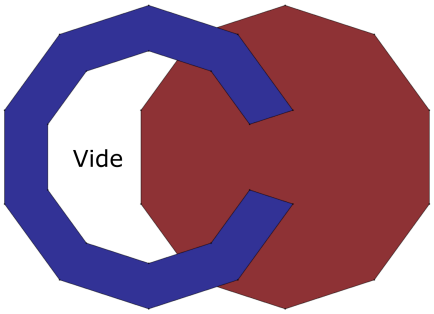


FIGURE 5.7 – Vide à l'intérieur de l'objet, ce qui peut enfermer de la matière d'impression et rendre l'objet plus fragile.

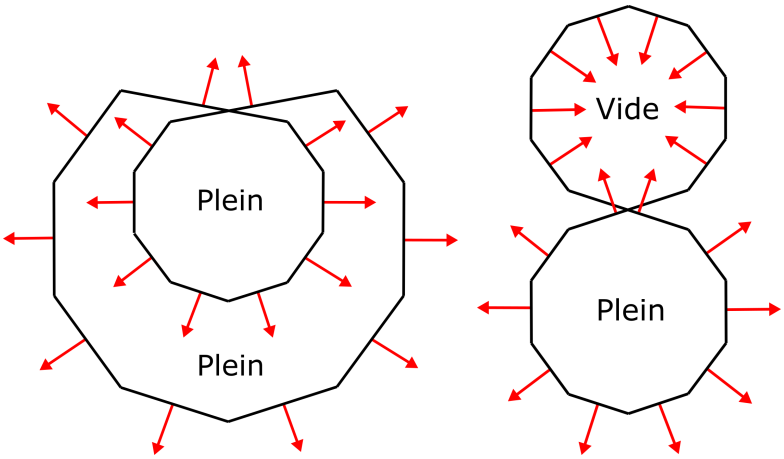


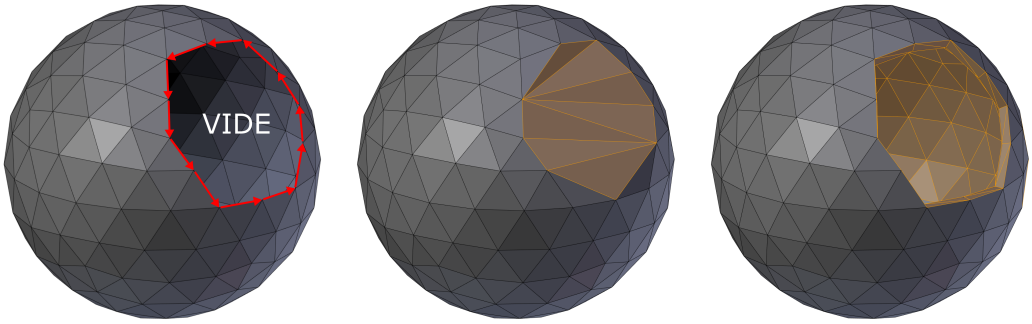
FIGURE 5.8 – Auto-intersection : (a) Auto-intersection mais surface orientable, (b) Auto-intersection mais surface non orientable.

5.3 Types de contours

L'idée est d'analyser un contour pour choisir quelle réparation lui appliquer. On va définir intuitivement deux types de contours.

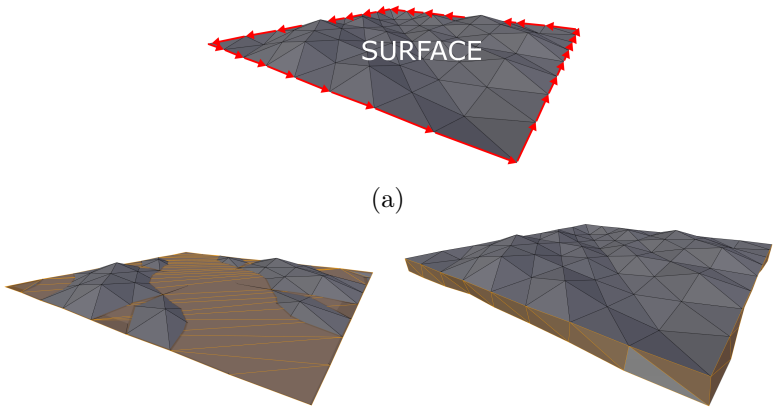
Le premier type de contour entoure une zone vide de face qu'on appelle trou dans la surface (Fig.5.9,(a)). Un trou doit être fermé avec une méthode de remplissage (Fig.5.9,(b)). Si un trou est fermé avec un épaississement, il risque d'enfermer de la matière d'impression ou de créer des vides fragilisant l'objet (Fig.5.9,(c)).

Le deuxième type de contour entoure la surface elle-même qu'on appelle surface fine (Fig.5.10,(a)). Une surface fine doit être réparée par un épaississement de la surface pour lui donner un volume (Fig.5.10,(c)). Appliquer un remplissage sur une surface fine risque de créer des auto-intersections et une surface non orientable (Fig.5.10,(b)).



(a) (b) (c)

FIGURE 5.9 – (a) Contour entourant un vide, (b) Réparation par remplissage, (c) Réparation par épaississement, Faces ajoutées par les réparations (en Jaune).



(a)

(b) (c)

FIGURE 5.10 – (a) Contour entourant une surface, (b) Réparation par remplissage, (c) Réparation par épaississement, Faces ajoutées par les réparations (en Jaune).

5.4 Algorithme de classification

Pour classer un contour, l'idée va être de regarder localement dans la zone entourée par le contour les faces proches de cette zone et leurs normales.

Dans le cas d'un contour qui entoure un trou dans la surface, les faces présentes dans cette zone sont principalement celles qui entourent le trou. D'autres faces de la surface peuvent se trouver théoriquement n'importe où dans cette zone puisqu'on n'a aucun pré requis sur les modèles 3D en entrée.

Dans le cas d'un contour qui entoure une surface, toutes les faces de la surface sont proches de cette zone.

Pour bien différencier ces deux types de contours, on va s'aider de la normale des faces. On va comparer cette normale avec une normale théorique de la zone entourée par le contour.

5.4.1 Remplissage par géocentre

Pour bien définir où se trouve la zone entourée par le contour, on va créer des faces temporaires pour la simuler. Pour cela on va appliquer sur le contour une réparation par géocentre.

L'avantage de cette méthode est qu'elle ferme toujours le contour quel que soit sa forme, après application il n'y a plus d'arête de bord. L'inconvénient est que cette une méthode de remplissage ne produit pas une réparation de qualité. Son algorithme naïf va provoquer beaucoup d'auto-intersections. Mais son utilisation dans ce contexte a pour but de simuler la zone entourée par le trou et non d'effectuer une réparation de qualité.

L'algorithme de remplissage par géocentre consiste à parcourir tous les sommets du contour pour en calculer un sommet moyen. Cette moyenne est faite en pondérant chaque sommet par la longueur de ses arêtes. On crée ce sommet moyen, puis pour chaque arête du contour on construit une face avec les deux sommets extrémités de l'arête et le sommet moyen (Fig.5.11). L'orientation des nouvelles faces dépend directement de l'arête de bord utilisée en effet cette dernière a déjà une orientation cohérente avec le reste de la surface (Fig.5.12). Les faces créées sont marquées pour les différencier des faces originales lors des calculs suivants de l'algorithme, mais aussi pour pouvoir les supprimer une fois la classification effectuée.

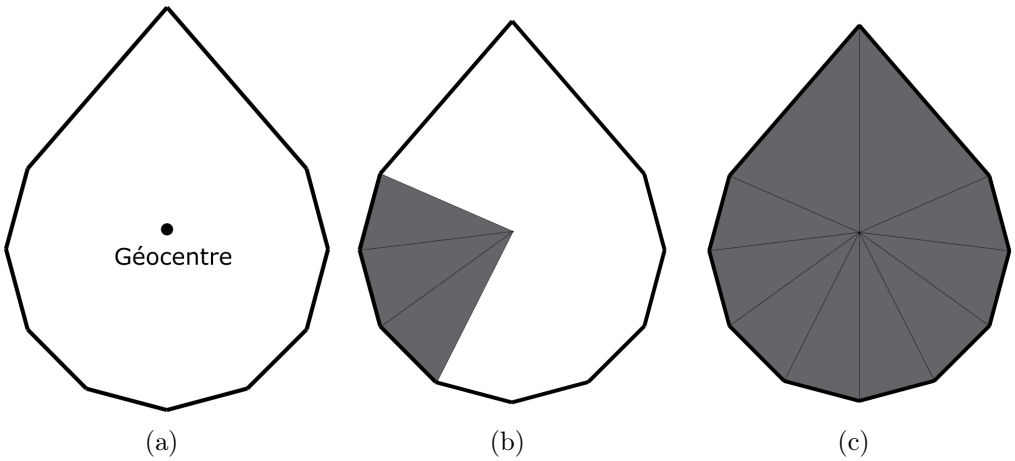


FIGURE 5.11 – Étapes de l'algorithme du géocentre.

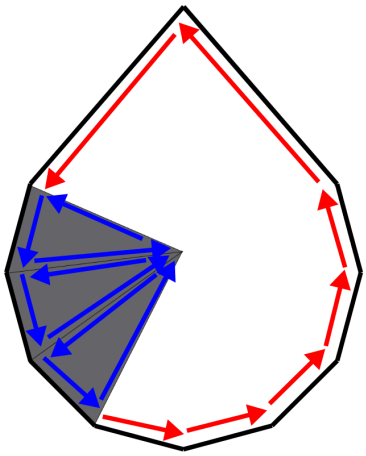


FIGURE 5.12 – Orientation des faces créées par le géocentre.

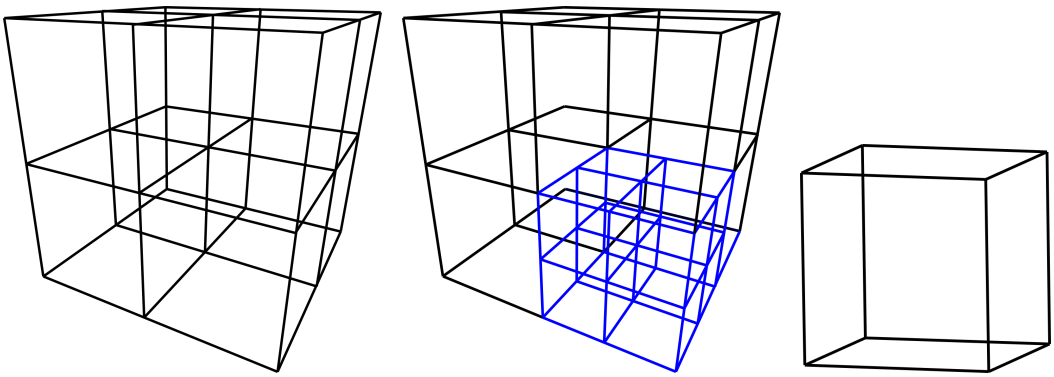
5.4.2 Création de l'octree

L'analyse de la zone entourée par le contour est précédée par une subdivision de l'espace en petit volume cubique. La subdivision va permettre d'analyser la zone de façon locale pour la simplifier. Pour cela on va utiliser une structure de type *octree*.

Pour ne pas alourdir inutilement l'*octree*, il ne comprendra que les faces de la composante connexe à laquelle appartient le contour analysé. Et par conséquent l'espace représenté par son nœud racine correspond au cube minimum englobant la composante connexe. Les arêtes du cube sont parallèles aux axes 3D.

L'*octree* est un arbre où chaque nœud définit un volume cubique. Chaque noeud possède huit enfants qui correspondent chacun à un huitième, diffé-

rent de l'espace de leurs pères. Une feuille est un nœud qui n'a pas d'enfants (Fig.5.13)(Fig.5.14). Un nœud est considéré comme une feuille selon plusieurs conditions non dépendantes.



(a) (b) (c)

FIGURE 5.13 – Octree : (a) Premier niveau de l'octree on divise l'espace en huit, (b) Deuxième niveau de l'octree sur une branche en bleu, (c)Feuille de l'octree.

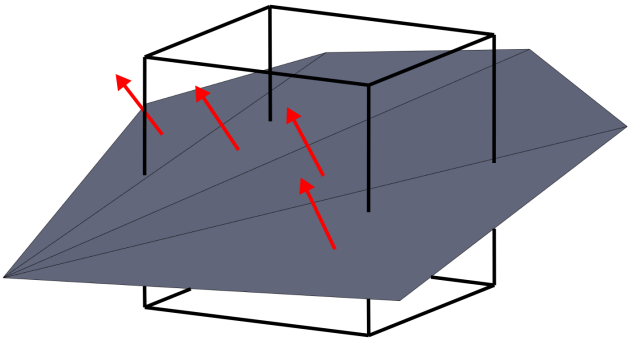


FIGURE 5.14 – Feuille avec un bout de surface, toutes les faces qui intersectent le cube appartiennent à la feuille (Une face peut appartenir à plusieurs feuilles).

Dans notre cas les deux conditions suivantes sont utilisées pour qu'un nœud soit une feuille.

Première condition, la taille de l'espace qu'il délimite correspondant à l'arête du cube. Cette taille va dépendre de l'épaisseur minimale de solidité. On cherche des faces du géocentre et des faces originales trop proches pour définir un volume imprimable, donc plus proche que l'épaisseur minimale de solidité. Pour détecter ces faces, on va par expérience utiliser une taille égale à deux fois l'épaisseur minimale de solidité, cette taille reste paramétrable.

Deuxième condition, un nombre de faces minimum égal à zéro. Si un nœud ne contient aucune face, il est inutile de le subdiviser. Notre analyse se basant

sur un pourcentage de feuille, même les nœuds ne possédant qu'une face sont subdivisés jusqu'à la taille minimale pour que toutes les feuilles aient le même poids dans l'analyse.

5.4.3 Analyse des feuilles

On va considérer trois types de feuilles :

Les feuilles qui n'ont aucune face de la réparation par géocentre dans leurs espaces (Fig.5.15). Ces feuilles délimitent une zone qui n'est pas proche de la zone entourée par le contour. Elles n'apportent donc pas d'information pour sa classification.

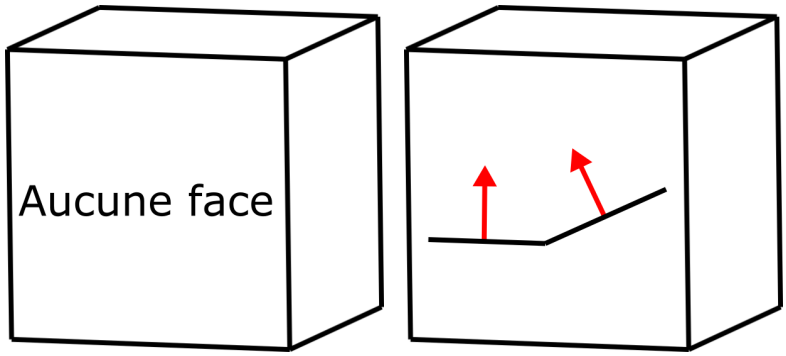
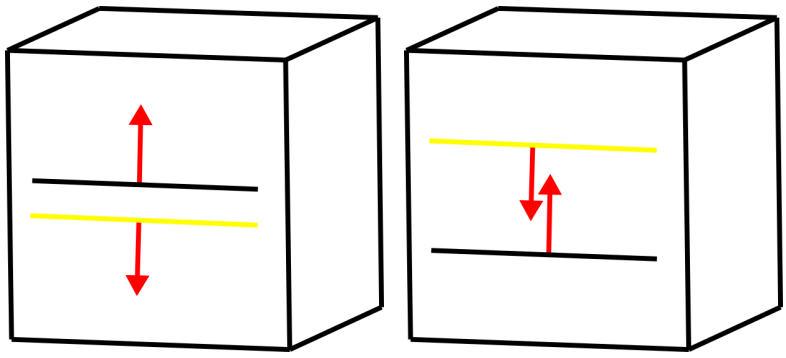


FIGURE 5.15 – Feuilles non prises en compte : (a) Feuilles sans face, (b) Feuilles avec uniquement des faces originales.

Les feuilles qui possèdent au moins une face de la réparation par géocentre (faces temporaires) vont être classées en deux catégories. Selon que leurs analyses indiquent que le contour entoure un trou ou une surface. La distinction se fait sur les critères suivant :

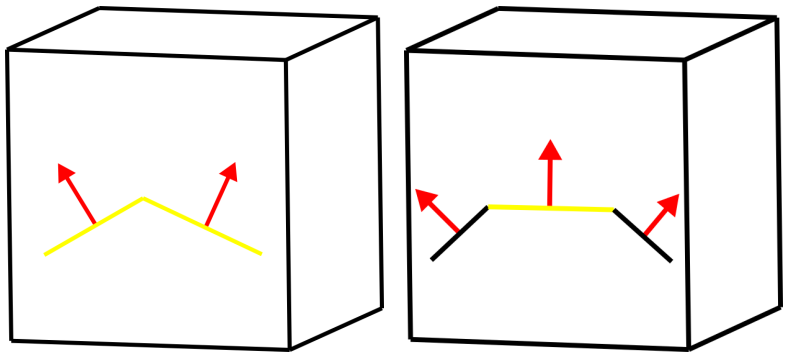
Dans le cas où le contour entoure les faces de la surface, les feuilles vont contenir des faces originales et des faces temporaires, dont les normales auront une direction proche et une orientation opposée (Fig.5.17).

Dans le cas où le contour entoure un vide dans la surface la plupart des feuilles étant au milieu du trou n'auront pas de faces originales présentes dans la feuille (Fig.5.16,(a)). Et si la feuille se trouve au bord du trou, elle contiendra des faces originales et des faces temporaires ayant des normales de direction et d'orientation proche (Fig.5.16,(b)).



(a) (b)

FIGURE 5.16 – Feuilles de type épaississement : (a)(b) Feuilles avec des faces temporaires et originales dont les normales sont opposées.



(a) (b)

FIGURE 5.17 – Feuilles de type remplissage : (a) Feuilles avec uniquement des faces temporaires, (b) Feuilles avec des faces temporaires et originales dont les normales sont proches.

L'algorithme de classification est décrit dans (Algo.8). Il calcule pour chaque feuille possédant une face temporaire, la moyenne des normales de toutes les faces temporaires présentes. Cette moyenne représente la normale de la zone entourée par le contour. Elle peut être pondérée par rapport à l'aire de chaque face présente dans le volume de la feuille.

La feuille est considérée comme de type épaississement, si pour une face originale l'angle entre sa normale et la normale moyenne est supérieur à 90 degrés.

La feuille est considérée comme de type remplissage, si pour toutes les faces originales l'angle entre leurs normales et la normale moyenne est inférieur à 90 degrés. Ou si la feuille ne contient pas de faces originales.

Le contour est classé dans une catégorie en fonction du type de feuilles qui prédomine. On calcule le pourcentage d'occurrences entre ces deux types de

feuilles. Si toutes les feuilles de l'arbre sont de type remplissage, le pourcentage sera de 0%. Si toutes les feuilles de l'arbre sont de type épaississement, le pourcentage sera de 100%.

Ce pourcentage est le résultat de l'analyse, il est attribué au contour. La façon de réparer ce contour dépendra d'un seuil paramétrable. Par défaut ce seuil est défini à 50% puisque c'est la limite entre la majorité d'un type de feuille sur l'autre.

On peut aussi utiliser un double seuil avec des pourcentages intermédiaires n'étant pas assez proches d'une catégorie pour appliquer sur leurs contours une réparation automatique.

Avant de passer aux contours suivants, on supprime les faces temporaires ainsi que le sommet créé pour le géocentre.

5.4.4 Conclusion

Chaque contour détecté est classé entre deux catégories (trou, surface fine). Ces catégories sont définies par rapport à la réparation la plus adaptée pour supprimer les arêtes de bord du contour (remplissage, épaississement). La qualité d'une réparation est jugée en fonction du fait qu'elle crée une surface non orientable, ou bien si elle fragilise le modèle (si elle enferme de la matière d'impression), ou si l'aspect visuel respecte le modèle original. Les étapes suivantes de la chaîne de réparation vont appliquer sur les contours les réparations adaptées à leurs catégories.

Pour améliorer la méthode de classification des contours, on pourrait créer de nouvelles catégories correspondant à de nouvelles réparations qui par exemple relient deux contours ensemble.

Une autre méthode envisageable pour classer les contours et d'utiliser une généralisation des *winding numbers*. La généralisation proposée par Jacobson et al. permet d'utiliser les *winding numbers* dans le cadre de maillages composés de plusieurs composantes connexes avec des auto-intersections des conditions de non-variété et des bords (Jacobson *et al.* [2013]). Les *winding numbers* ont alors une valeur continue estimant leurs degrés d'intérieur ou d'extérieur du volume (Fig.5.18). En analysant les *winding numbers* des points proches d'une composante connexe on peut la classer comme une surface fine si la majorité a une valeur inférieure à un seuil (défini entre 0 et 1). À l'inverse, si la majorité *winding numbers* des points proches d'une composante connexe a des valeurs proches de 1, alors ses contours peuvent être classés comme des trous.

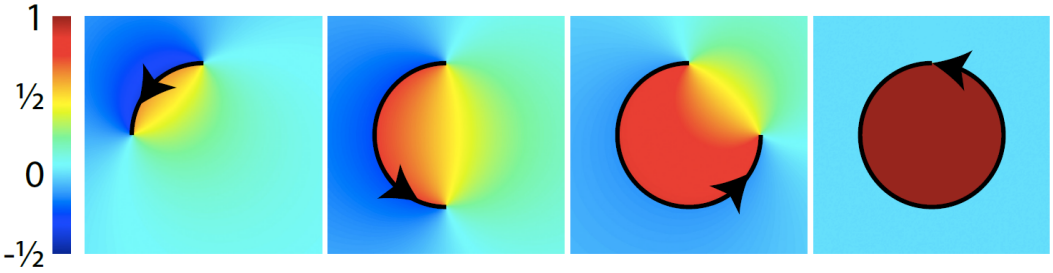


FIGURE 5.18 – *Winding numbers* autour d'arcs de cercle de plus en plus fermés (Exemples en deux dimensions). D'après Fig.6 de (Jacobson *et al.* [2013]).

Algorithme 8 Choix d'une réparation pour un contour

Entrées: contour, liste d'arêtes de bord consécutives formant un contour
Feuilles, la liste des feuilles de l'octree dans lequel on a plongé la CC du contour

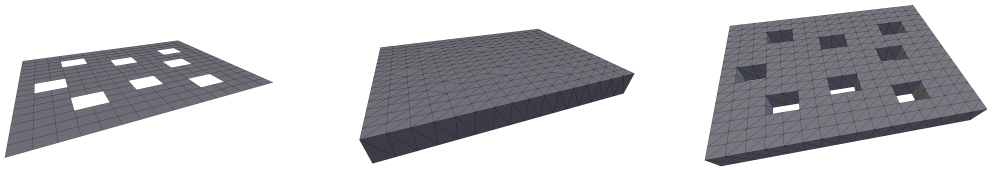
Sorties: choixRéparation, booléen indiquant la réparation à appliquer sur le contour

```
1: pour chaque feuille ∈ Feuilles faire
2:    Étiqueter feuille à neutre
3:    pour chaque face temporaire ∈ feuille faire
4:       normaleMoyenne += normale(face)
5:    fin pour
6:    pour chaque face originale ∈ feuille faire
7:       Étiqueter feuille à remplissage
8:       alpha = angle entre normale(face) et normaleMoyenne
9:       si alpha > 90° alors
10:         Étiqueter feuille à épaississement
11:      fin si
12:   fin pour
13:   si feuille est étiquetée remplissage alors
14:      cmptRemplissage++
15:   fin si
16:   si feuille est étiquetée épaississement alors
17:      cmptÉpaississement++
18:   fin si
19:   Incrémente les compteurs cmptRemplissage ou cmptÉpaississement en
      fonction de l'étiquette de feuille
20: fin pour
21: choixPourcentage = cmptÉpaississement / (cmptRemplissage + cmptÉ-
    paississement) * 100
22: si choixPourcentage < seuil alors
23:    choixRéparation = Remplissage
24: sinon
25:    choixRéparation = Épaississement
26: fin si
```

Chapitre 6

Réparation par remplissage

Une fois que tous les contours sont classés, on va appliquer la réparation correspondante sur chacun d'entre eux. La réparation par remplissage d'un contour est indépendante des réparations des autres contours appartenant à la même surface ou non. Ce n'est pas le cas de la réparation par épaississement qui implique l'ensemble des contours de sa surface. La réparation par épaississement d'une surface n'impacte pas les autres surfaces du modèle. Dans le cas où une surface posséderait des contours avec les deux classifications possibles, un choix de priorité de réparation est alors nécessaire. Il faut choisir si l'on souhaite d'abord remplir les trous de la surface puis épaissir la surface, ou si l'on applique directement l'épaississement, ce qui aura pour conséquence que les contours classés par remplissage serviront à construire les faces de côté de l'épaississement. Le premier choix sera appliqué par défaut, car il respecte mieux la classification des contours. La possibilité de rendre l'épaississement prioritaire sera optionnelle (Fig.6.1).



(a) (b) (c)

FIGURE 6.1 – Différence de priorité de réparation : (a) Surface à réparer, (b) Surface réparée par remplissage puis par épaississement, (c) Surface réparée par épaississement.

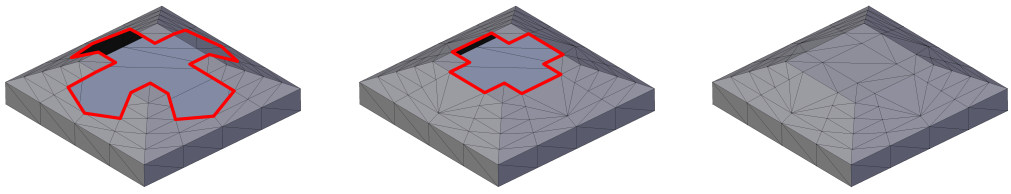
Les réparations se font par surface, car pour l'épaississement même s'il n'est appliqué qu'après les réparations de remplissage, il implique tous les contours restant de la surface. Pour chaque surface, on applique la réparation par remplissage un à un sur les contours classés comme entourant un trou.

La réparation par remplissage va être appliquée de manière incrémentale sur chaque contour. Entre chaque itération, on vérifie s'il reste des arêtes de bord du contour initial et/ou créées par la réparation. Le contour étant modifié par la réparation, on relance une recherche de contour sur ces arêtes de bord uniquement. Cette recherche peut identifier un ou plusieurs contours. L'itération suivante consiste à appliquer à nouveau la réparation par remplissage sur chacun de ces nouveaux contours et toujours de manière incrémentale (Fig.6.2).

L'algorithme a deux conditions d'arrêt après chaque itération :

— Le remplissage est complet. Toutes les arêtes de bord du contour (de cette itération) ont été réparées et aucune arête de bord n'a été créée par la réparation.

— L'itération n'a créé aucune nouvelle face. Le contour est identique à celui précédant la réparation, une nouvelle itération est donc inutile. Après cette condition d'arrêt, le contour initial n'est pas entièrement rempli et la surface a encore des arêtes de bord. Il faut prévenir l'utilisateur pour que cette composante connexe soit réparée manuellement.



(a) (b) (c)

FIGURE 6.2 – Étapes de remplissage : (a) Contour à remplir (en Rouge), (b) Contour après une première itération (en Rouge), (c) Contour rempli.

6.1 Découpe du contour d'un trou en segments planaires

Un segment : est une suite de sommets et d'arêtes consécutive d'un contour. Un segment commence et se termine par un sommet. Sa longueur est définie par le nombre de sommets qu'il contient. Sa longueur peut aller d'un sommet au nombre de sommets du contour.

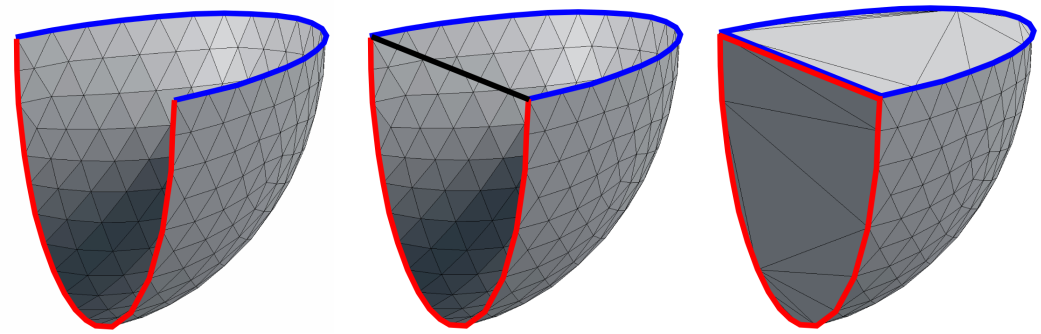
Un segment planaire : est un segment dont les éléments (sommets,arêtes) sont sur un même plan ou proche d'un même plan à un écart près. Tous les segments dont il est question dans ce chapitre sont des segments planaires.

L'idée de la réparation par remplissage est :

— De découper le contour en plusieurs segments planaires (Fig.6.3,(a)).

— De rajouter une arête entre le dernier et le premier sommet de chaque segment pour les fermer (Fig.6.3,(b)). Un segment fermé devient un contour.

— D'appliquer un algorithme de triangulation 2D le *Ear Clipping* (Eberly [2002]) sur chaque segment fermé. L'algorithme de *Ear Clipping* est modifié pour fonctionner sur un contour 3D (Fig.6.3,(c)).



(a) (b) (c)

FIGURE 6.3 – Exemple de remplissage avec découpe du contour en segments planaires : (a) Contour découpé en deux segments planaires (en Bleu et Rouge), (b) Ajout de l'arête de fermeture (en Noir), (c) Remplissage des deux sous-contours (en Bleu et Rouge).

Écart de planéarité : est l'angle maximum entre deux normales de deux plans pour que les éléments (sommets,arêtes) présents sur ces plans soient considérés comme planaire.

La planéarité des segments planaire n'est pas stricte, les éléments d'un segment planaire ne sont pas toujours sur un même plan (Fig.6.4). L'écart de planéarité permet de ne pas trop découper le contour. Pour chaque segment sur le contour, une arête de bord est rajoutée. Ces arêtes sont réparées par les itérations suivantes. En conclusion, trop découper le contour créer plus d'arêtes de bord à réparer ce qui demande plus d'itération.

Avec une planéarité stricte, les segments seraient souvent une suite de trois sommets uniquement. Une fois fermé ils n'auraient besoin que d'une face pour être remplis. Le contour initial aurait alors un remplissage global naïf et de mauvaise qualité. L'algorithme de *Ear Clipping* ne serait pas exploité.

L'adaptation du *Ear Clipping* à un contour 3D permet entre autres d'avoir un écart de planéarité dans le contour à remplir. Cette adaptation autorise l'écart de planéarité dans la découpe du contour en segments planaire, tant qu'il est inférieur ou égal à celui du *Ear Clipping* 3D.

Ces écarts de planéarité sont paramétrables, mais ont une valeur par défaut de 22,5 degrés définie par expérimentation. Ce chiffre vient en partie de

la tolérance à la 3D du *Ear Clipping* et sera expliqué dans les prochaines étapes.

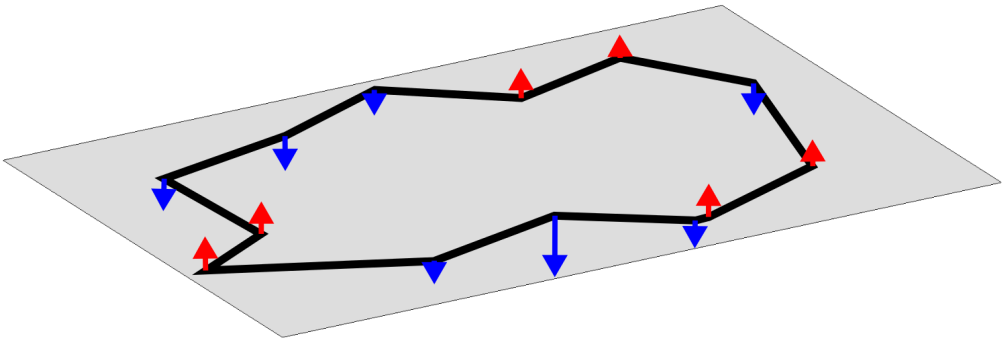


FIGURE 6.4 – Contour entièrement planaire (un seul segment planaire). Distance entre un sommet et le plan moyen, si le sommet est au-dessus du plan (en Bleu), si le sommet est en dessous du plan (en Rouge).

6.1.1 Normal locale par sommet

Dans le but de trouver des segments planaires sur le contour, on va attribuer à chaque sommet du contour une normale locale pour ensuite en faire un suivi. Pour calculer la normale locale N_{S0} d'un sommet S_0 on utilise les vecteurs a_1 et a_2 (Fig.6.5).
Le vecteur a_1 part de S_0 vers le sommet suivant sur le contour Ss.

$$a_1 = Ss - S_0 \tag{6.1}$$

Le vecteur a_2 part de S_0 vers le sommet précédent sur le contour Sp.

$$a_2 = Sp - S_0 \tag{6.2}$$

La normale locale du sommet S_0 est égale au produit vectoriel entre ces deux vecteurs.

$$N_{S0} = a_1 \times a_2 \tag{6.3}$$

La normale locale au sommet S_0 correspond à la normale de la face inexistante (S_0, Ss, Sp).

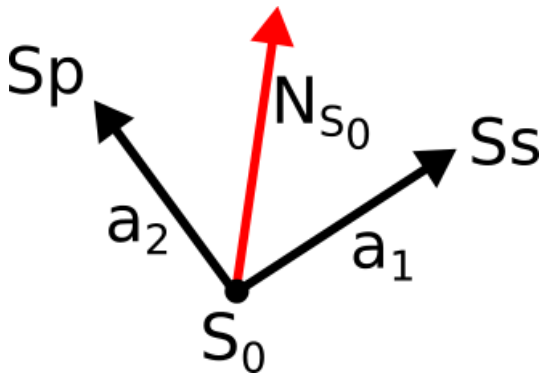


FIGURE 6.5 – Normale locale d'un sommet du contour.

Dans les prochaines étapes de l'algorithme, on va à plusieurs reprises comparer ces normales locales avec d'autres normales ou entre elles. Mais l'orientation de ces normales ne doit pas toujours être prise en compte. Par exemple pour la recherche de segments planaires seule la direction des normales compte. Un segment planaire en dent de scie aura pour chaque sommet consécutif une direction de la normale identique, mais une orientation opposée (Fig.6.6).

Les comparaisons entre normales se font en calculant l'angle qui les sépare. Pour ne pas prendre en compte l'orientation, on calcule l'angle aigu entre les directions des deux normales. En d'autres termes si l'angle entre les deux normales est obtus on utilisera l'angle entre la première normale et l'opposée de la seconde (Fig.6.7). Ce qui revient à utiliser l'algorithme suivant, avec α l'angle entre les deux normales ($Si(\alpha > 90)\{\alpha = 180 - \alpha\}$).

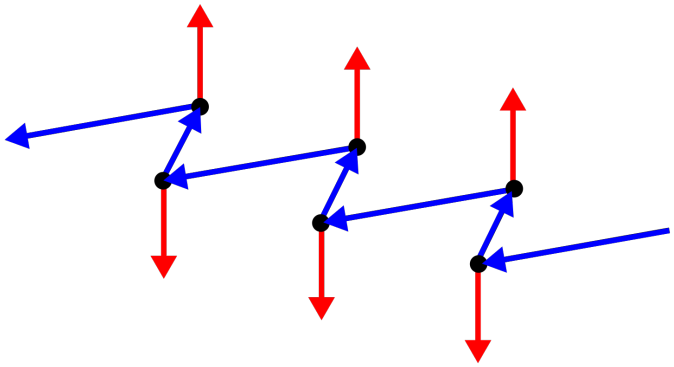
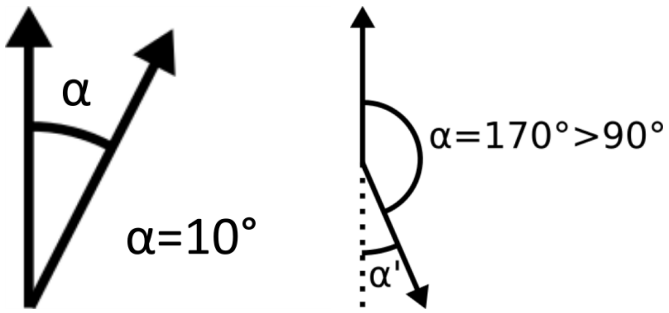


FIGURE 6.6 – Segment planaire, l'orientation des normales locales non prises en compte.



(a) (b)

FIGURE 6.7 – Comparaison de normales sans orientation : (a) Angle de 10° entre les deux normales, (b) Angle de 170°>90° entre les deux normales, on utilise alors $\alpha' = 180° - 170° = 10°$ comme angle.

Lorsque trois sommets du contour sont presque alignés, le plan qu'ils définissent est très variable. C'est-à-dire qu'un faible déplacement d'un des trois sommets peut faire grandement varier ce plan et donc la normale locale. Cette donnée est donc peu fiable pour la recherche de segment planaire, car la normale locale peut indiquer un écart important avec les normales précédentes alors que les trois sommets sont alignés et donc sur le même plan (Fig.6.8).

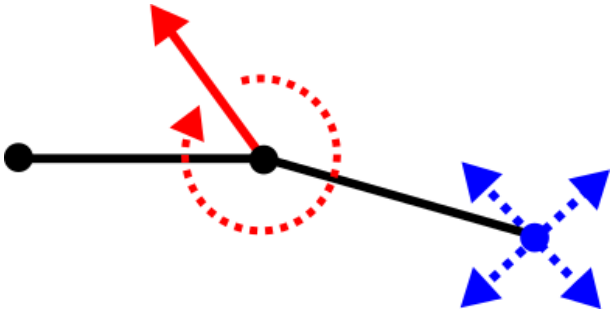


FIGURE 6.8 – Trois sommets consécutifs sur un contour proche de l'alignement. Un faible déplacement d'un sommet (en Bleu) provoque un fort changement de la normale locale (en Rouge).

Pour éviter ce problème, on va, avant le calcul de la normale locale, vérifier ces alignements. Si un sommet est proche de l'alignement avec son sommet précédent et son sommet suivant, on ne lui attribue pas de normale locale (null). Une normale locale lui sera attribuée dans l'étape suivante.

Le calcul d'alignement des trois sommets est fait en calculant le cosinus de l'angle (Sp, S_0, Ss).

$$cos(S_0) = \frac{a1 \cdot a2}{|a1| * |a2|} \tag{6.4}$$

On vérifie ensuite que le cosinus correspond à un angle (Sp, S_0, Ss) inférieur à 180 degrés moins un écart ϵ $(cos(S_0) > -1 + cos(\epsilon))$. Dans le cas contraire,

les trois sommets sont considérés comme alignés. La valeur d'ϵ est fixée par expérimentation à 10 degrés.

L'attribution d'une normale locale aux sommets trop alignés se fait en regardant les normales locales déjà attribuées. Les sommets sans normale locale sont traités par groupe consécutif sur le contour. Pour chaque groupe on prend en compte le sommet précédent le premier sommet du groupe et le sommet suivant le dernier. Ces deux sommets sont appelés respectivement "borne de départ" et "borne de fin". Les deux bornes ont une normale locale attribuée, car sinon ils auraient appartenu au groupe.

Les normales locales des deux bornes sont comparées, si elles sont proches on considère que les sommets alignés entre les deux bornes sont sur le même plan que leurs bornes. Les sommets du groupe se voient attribuer une normale locale égale à la moyenne des normales locales des deux bornes (Fig.6.9,(a)).

Deux normales locales sont proches si l'angle aigu entre leurs directions est inférieur à un écart (l'orientation n'est pas prise en compte). Cet écart est le même que celui qui définit la planéarité. Il sera aussi utilisé pour la recherche de segment planaire. Pour les groupes dont les normales locales des deux bornes sont trop éloignées, aucune normale locale ne leur est attribuée dans ce premier parcours.

L'attribution d'une normale locale aux sommets des groupes dont les bornes avaient des normales locales trop éloignées, se fait en définissant si leurs bornes sont "stables". Une borne est considérée comme stable si sont voisinage direct sur le contour à une normale locale proche de la sienne. Pour la borne de départ, on utilise le sommet précédent sur le contour et pour la borne de fin le sommet suivant sur le contour. Si ce sommet a une normale locale attribuée et que celle-ci est proche de la normale locale de la borne, alors la borne est considérée comme stable.

Pour chaque groupe, si une seule des deux bornes est considérée comme stable, tous les sommets du groupe se voient attribuer une normale locale égale à celle de la borne stable (Fig.6.9,(b)).

Si aucune des deux bornes n'est considérée comme stable, les sommets du groupe sont divisés en deux. La première moitié se voit attribuer une normale locale égale à celle de la borne de départ. La deuxième moitié se voit attribuer une normale locale égale à celle de la borne de fin (Fig.6.9,(c)).

Faire deux parcours permet d'utiliser les valeurs attribuées dans le premier pour définir la stabilité des bornes. Les bornes proviennent, elles, forcément du calcul des normales locales.

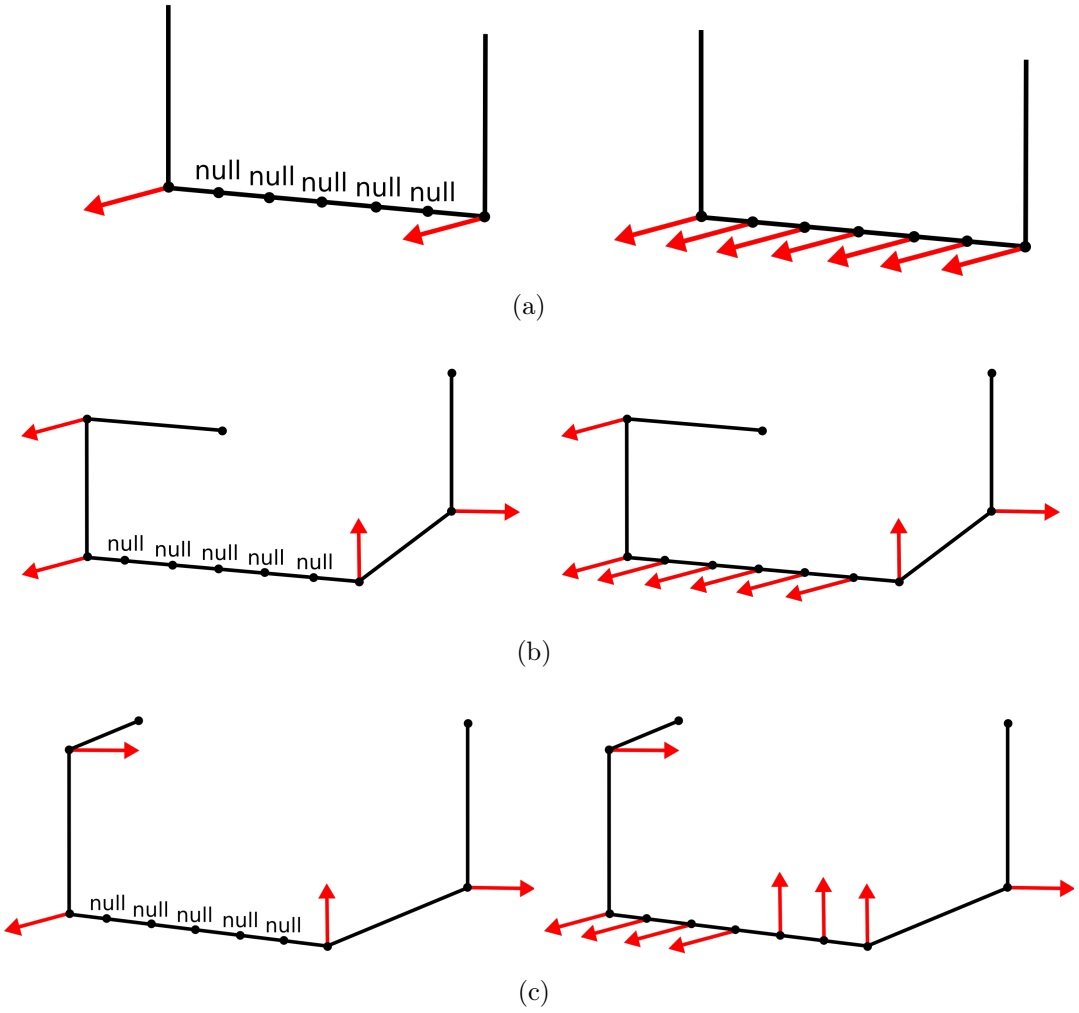


FIGURE 6.9 – Attribution d'une normale aux sommets alignés (null) : (a) Deux bornes identiques, (b) Une seule borne stable, (c) Deux bornes instables.

6.1.2 Recherche de segment planaire sur un contour de trou

Tous les sommets du contour ont une normale locale attribuée, la recherche de segments planaires utilise ces normales locales en parcourant le contour (Fig.6.10)(Algo.9). La recherche de segment planaire privilégie les segments planaire les plus longs possible. Comme vu précédemment, trop découpé, le contour revient à appliquer un remplissage naïf.

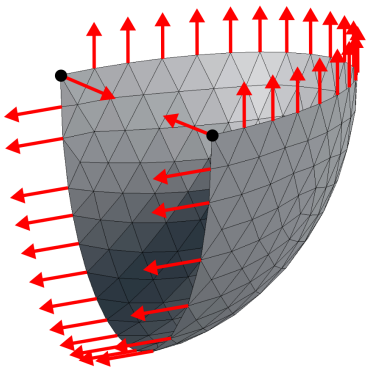


FIGURE 6.10 – Contour après calcul de toutes les normales locales.

Le parcours se fait sommet par sommet en suivant le sens du contour. Un segment planaire est construit en partant d'un sommet initial. Le sommet suivant le dernier sommet du segment en construction est ajouté au segment si l'angle entre sa normale et la normale moyenne du segment est inférieure à l'écart de planéarité. Si l'angle entre sa normale et la normale moyenne du segments est supérieur à l'écart de planéarité, alors le sommet est appelé "sommet de rupture" et le segment planaire est finis. Le sommet de rupture est le sommet initial du segment planaire suivant.

La normale moyenne ($normalMoy$) du segment est calculée en même temps que la construction du segment planaire. Elle prend au début de la construction d'un segment planaire la valeur de la normale locale de son sommet initial. Pour chaque sommet ajouté au segment planaire, la normale moyenne est mise à jour en suivant les deux instructions suivantes :

— La pondération : Pour mettre à jour la normale moyenne, on fait une moyenne pondérée entre la normale moyenne et la normale locale ($normalLocal$). Le poids des deux dépend du nombre de sommets déjà présents dans le segment (nbV). Le poids de la normale moyenne est de $\frac{nbV}{nbV+1}$ et le poids de la normale locale est de $\frac{1}{nbV+1}$.

— L'orientation des normales : Comme dit précédemment, on ne prend pas en compte l'orientation des normales (Fig.6.6). Il faut donc lors du calcul de normale moyenne orienter toutes les normales dans la même orientation pour éviter qu'elles ne s'annulent. On choisira l'orientation de la normale locale du sommet initiale. Lorsque la normale moyenne est mise à jour, on vérifie si la normale locale à la même orientation ou non, en regardant si l'angle entre les deux est aigu ([0°,90°]) ou obtus ([90°,180°]). Si l'angle est obtus on multiplie la normale locale par -1 avant de faire la moyenne pondérée.

$$normalMoy = normalMoy * \frac{nbV}{nbV+1} + (\pm normalLocal) * \frac{1}{nbV+1} \quad (6.5)$$

Pour le premier segment planaire uniquement, afin de limiter l'impacte du choix du sommet initial, le sens du parcours est alterné.

Sans utiliser deux sens de parcours, le sommet initial du premier segment planaire serais le sommet de rupture du dernier segment planaire sans avoir testé l'écart entre leurs normales. Ainsi les éléments du premier et du dernier segment planaire pourrais être sur un même plan sans être rassemblais dans un seul segment planaire.

Sans alternance du sens de parcours, les sommets du premier sens sont avantagés. L'influence de la normale locale d'un sommet ajouté sur la normale moyenne diminue au fur et à mesure que le segment grandit. La normale locale du premier sommet du second sens aura une influence moins grande lors de son ajout si le sens de parcours n'est pas alterné. Ainsi le premier sommet du second sens peut être ajouté avec l'alternance et ne pas l'être sans. Pour limiter l'impacte du choix initial on préfère qu'il le soit.

Le parcours du contour en alternant le sens ce fait sommet par sommet, en alternant entre le sommet suivant le dernier sommet du segment en construction et le sommet précédent le premier sommet du segment en construction. Le premier sommet du segment n'est plus forcément le sommet initial. Lorsqu'un sommet de rupture est trouvé dans un premier sens, le parcours se poursuit uniquement dans l'autre sens. La construction du premier segment planaire s'arrête, lorsqu'un sommet de rupture est trouvé dans les deux sens. Le sommet de rupture du sens correspondant à celui du contour sera le sommet initial du deuxième segment planaire.

La recherche de segment planaire s'arrête lorsque le parcours revient sur le premier sommet du premier segment. Le segment en construction considère ce sommet comme son sommet de rupture. Après la recherche tous les sommets du contour appartiennent à un segment et uniquement un.

Si le contour est entièrement planaire, alors aucun sommet de rupture ne va être trouvé. Le parcours s'arrête alors quand le nombre de sommets dans le segment est égal au nombre de sommets du contour (Fig.6.11).

En sortie de la recherche de segment planaire, on possède la liste des segments planaires du contour. Chaque segment est défini par son premier sommet, son dernier sommet et son plan. Le plan du segment est défini par la normale moyenne du segment (qui devient la normale au plan du segment) et un point du plan calculé comme la moyenne des sommets du segment.

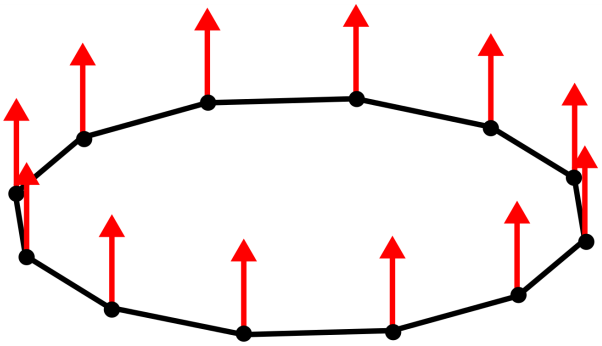


FIGURE 6.11 – Contour planaire, le segment sera identique au contour.

Algorithme 9 Recherche de segments planaires

Entrées: contour, la liste ordonnée des sommets du contour ;
normalesLocales, la liste ordonnée des normales locales des sommets du contour

Sorties: segmentsPlanaires, la liste ordonnée des segments planaires

courant=0 ;avance=0 ;recule=0
Ajout d'un nouveau segment dans segmentsPlanaires
premierSegment=VRAI
Ajout du sommet contour[courant] dans segmentsPlanaires[dernier]
normalMoy = normalesLocales[courant]
tant que $\exists$ s $\in$ contour tel que s $\notin$ segment, $\forall$ segment $\in$ segmentsPlanaires **faire**
  si premierSegment && abs(recule)<avance **alors**
    courant=(recule−−)
  sinon
    courant=(avance++)
  fin si
  normalLocal = normalesLocales[courant]
  angle = angle entre normalLocal et normaleMoy
  si angle<ecartPlanearite **alors**
    Ajout du sommet contour[courant] dans segmentsPlanaires[dernier]
    $normalMoy = normalMoy * \frac{nbV}{nbV+1} + normalLocal * \frac{1}{nbV+1}$
  sinon si 180-angle<ecartPlanearite **alors**
    Ajout du sommet contour[courant] dans segmentsPlanaires[dernier]
    $normalMoy = normalMoy * \frac{nbV}{nbV+1} + -1 * normalLocal * \frac{1}{nbV+1}$
  sinon
    Ajout d'un nouveau segment dans segmentsPlanaires
    premierSegment=FAUX
    Ajout du sommet contour[courant] dans segmentsPlanaires[dernier]
    normalMoy = normalesLocales[courant]
  fin si
fin tant que

Limite : Comme la normale moyenne est mise à jours en cours de parcours et que les comparaisons avec les normales locales des sommets aussi. Une normale locale peut avoir un angle avec la normale moyenne inférieur à l'écart de planéarité au moment de l'ajout de son sommet dans le segment. Mais après l'ajout d'autres sommets et donc après plusieurs mises à jour de la normale moyenne, cet angle peut devenir supérieur à l'écart de planéarité (Fig.6.12).

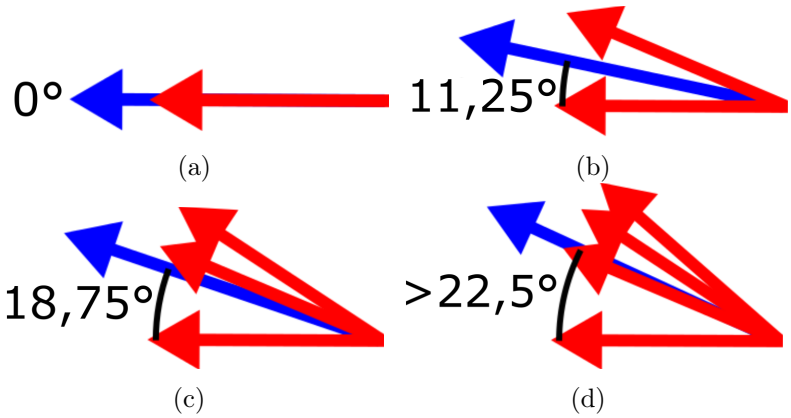


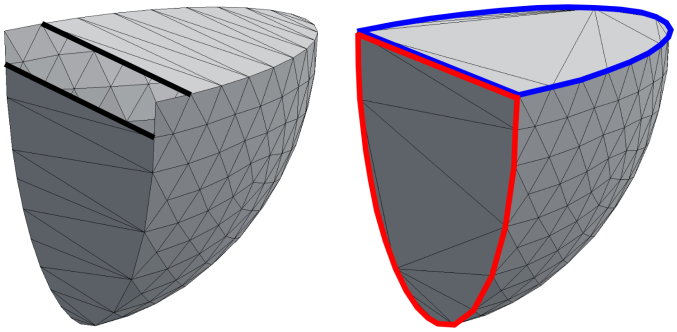
FIGURE 6.12 – Mise à jour de la normale moyenne (en Rouge). (a) La normale moyenne est identique à la première normale locale (en Bleu), (b)(c)(d) Les normales locales suivantes sont éloignées au maximum de la normale moyenne (22,5°), (d) La première normale locale ce retrouve à plus de 22,5° de la normale moyenne.

6.1.3 Fermeture des segments

Choix de l'arête de fermeture

Chaque segment planaire est fermé par la création d'une arête de fermeture. Un segment planaire fermé devient un contour planaire sur lequel l'étape de remplissage peut être appliquée. Si le contour ne contient qu'un seul segment, celui-ci est déjà fermé puisqu'il s'agit du contour original en entier : on peut lui appliquer directement l'étape de remplissage.

Si les arêtes de fermeture sont construites en utilisant le dernier et le premier sommet de chaque segment, une arête du contour reste entre chaque segment fermé. Les arêtes du contour étant des arêtes de bord, il est préférable qu'elle fasse partie d'un segment fermé à remplir. Par exemple en utilisant le dernier et le premier sommet, après la première itération (Fig.6.13,(a)) il reste six arêtes de bord. En utilisant de meilleurs sommets pour les arêtes de fermeture, après la première itération (Fig.6.13,(b)) il ne reste aucune arête de bord.



(a) (b)

FIGURE 6.13 – Exemple de choix d'arête de fermeture : (a) Arêtes de fermeture utilisant le dernier est le premier sommet des segments, (b) Arête de fermeture avec un meilleur choix de sommets.

Les normales locales de chaque sommet sont calculées à l'aide des sommets suivants et précédents de ce dernier. Donc le sommet précédent le premier sommet du segment et le sommet suivant le dernier du segment appartiennent au même plan que ce du segment. L'arête de fermeture peut donc les utiliser. Mais ces sommets appartiennent à un autre segment, on va devoir définir une priorité.

Certains segments ne contiennent qu'un ou deux sommets, on dit qu'ils sont de longueur un ou deux. Ces segments ne définissent pas un plan à eux seuls, ils ne peuvent donc pas être remplis. Par exemple un segment de longueur un peut être un sommet appartenant à la fois au plan du segment le précédent et à celui du suivant (Fig.6.10,(Point Noire)). Les sommets de ces segments vont être utilisés en priorité pour fermer leurs segments voisins.

Les segments de longueur trois ou supérieure sont fermés en priorité, puis ceux de longueur deux puis ceux de longueur un. Pour les deux derniers cas, on vérifie si certains de leurs sommets ont été utilisés pour l'arête de fermeture d'un segment traité auparavant. L'arête de fermeture est construite en règle général avec (Algo.10)(Fig.6.14) :

— Le sommet précédent le premier sommet du segment à fermer quelle que soit la longueur du segment auquel il appartient.

— Le sommet suivant le dernier sommet du segment à fermer s'il appartient à un segment de longueur un ou deux, ou le dernier sommet du segment à fermer sinon.

Pour être fermé un segment à besoin d'au minimum trois sommets et deux arêtes consécutives sur le contour. Sans utiliser de sommets de segments voisins, un segment de longueur deux est composé de deux sommets et d'une arête et un segment de longueur un d'un seul sommet.

Pour les segments de longueur deux si un seul de leurs deux sommets a était utilisé pour l'arête de fermeture d'un segment voisin, alors le sommet restant est traité comme un segment de longueur un. Si ces deux sommets ont été utilisés alors il n'est plus possible de fermer ce segment, car il ne reste que ses deux sommets et son arête d'utilisables pour construire un contour. L'arête restante appartient au contour original, elle restera une arête de bord et nécessitera une itération supplémentaire pour être traitée (Fig.6.14,(c)).

Les segments de longueur un, ne peuvent être fermé qu'en utilisant le sommet précédent et suivant son sommet. Pour cela il faut que son sommet ne soit pas utilisé par aucune des arêtes de fermeture de ses segments voisins. Le cas n'arrive que si le segment est traité avant ses voisins. Ils sont alors aussi de longueur un ou de longueur deux avec un sommet utilisé (traité comme un segment de longueur un). La fermeture se fera avec une seule face (Fig.6.14,(d)).

Les deux sommets de l'arête de fermeture d'un segment peuvent être un même sommet dans un cas particulier. Il se présente uniquement si le contour contient deux segments, un de longueur deux ou supérieur et un de longueur un. On revient dans ce cas au contour entier sur lequel on peut appliquer le remplissage (Fig.6.14,(e)).

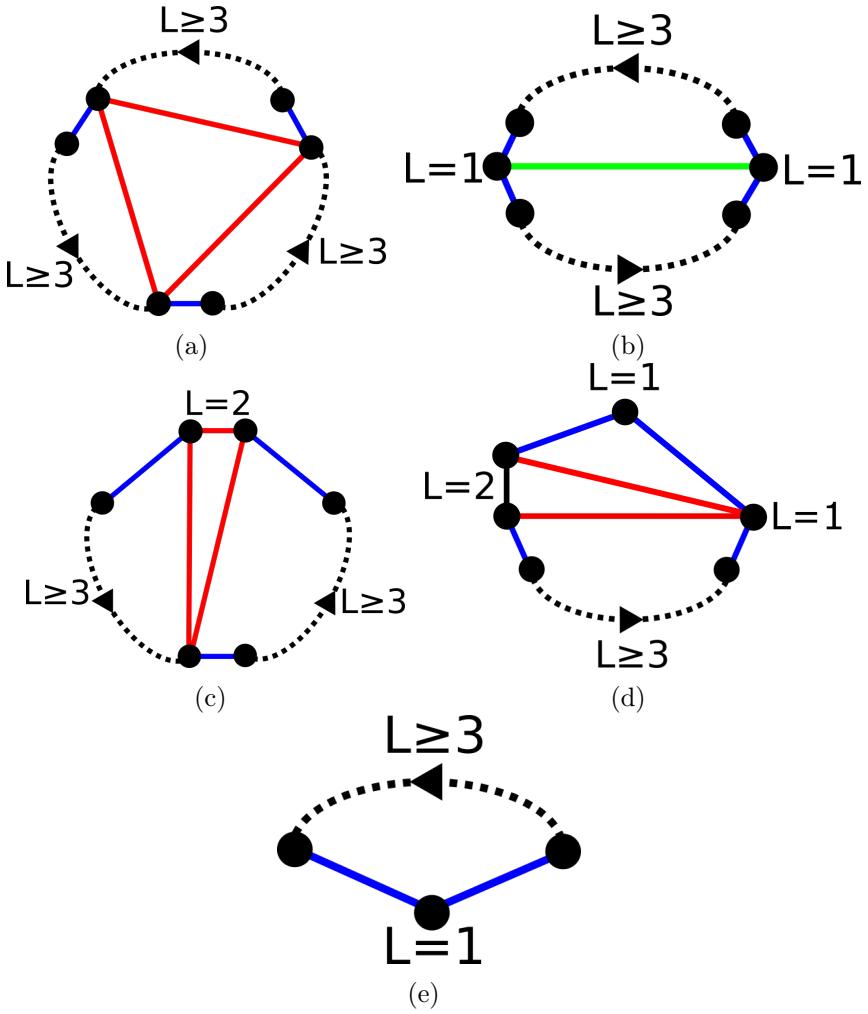


FIGURE 6.14 – Cas avec segments de différentes longueurs (L) pour le choix des sommets des arêtes de fermeture : Segment avant fermeture (en Noire), Prolongement d'un segment d'un sommet (en Bleu), Arête de fermeture de bord (en Rouge), Arête de fermeture (en Vert).

Algorithme 10 Choix des sommets des arêtes de fermeture

Entrées: contour, la liste ordonnée des sommets du contour ;
segmentsPlanaires, la liste ordonnée des segments planaires,
chaque segment contient la liste ordonnée de ses sommets
Sorties: sommetsFermeture, couples de sommets extrémités de l'arête de fermeture de chaque segment planaire

1: **pour chaque** segment segL3 de longueur ≥ 3, $\in$ segmentsPlanaires **faire**
2: sPre = sommet précédent segL3[premier] sur le contour
3: Ajout de sPre dans sommetsFermeture[segL3]
4: **si** longueur(segSuiv)<3 **alors**
5: sSuiv = sommet suivant segL3[dernier] sur le contour
6: Ajout de sSuiv dans sommetsFermeture[segL3]
7: **sinon**
8: Ajout du sommet segL3[dernier] dans sommetsFermeture[segL3]
9: **fin si**
10: **fin pour**
11: **pour chaque** segment segL2 de longueur 2, $\in$ segmentsPlanaires **faire**
12: **si** les sommets s1 et s2 $\in$ segL2, n'ont pas déjà été ajoutés **alors**
13: sPre = sommet précédent segL2[premier] sur le contour
14: sSuiv = sommet suivant segL2[dernier] sur le contour
15: Ajout de sPre dans sommetsFermeture[segL2]
16: Ajout de sSuiv dans sommetsFermeture[segL2]
17: **sinon si** un seul sommet s $\in$ segL2, n'a pas pas déjà été ajouté **alors**
18: segL2 = segment de longueur 1 avec s comme seul sommet
19: **sinon**
20: Le segment segL2 n'est pas fermé
21: **fin si**
22: **fin pour**
23: **pour chaque** segment segL1 de longueur 1, $\in$ segmentsPlanaires **faire**
24: **si** le sommet s $\in$ segL1 n'a pas pas déjà été ajouté **alors**
25: Ajout du sommet précédent s dans sommetsFermeture[segL1]
26: Ajout du sommet suivant s dans sommetsFermeture[segL1]
27: **sinon**
28: Le segment segL1 n'est pas fermé
29: **fin si**
30: **fin pour**

Note : Le passage au segment suivant, se fait qu'une fois l'étape de remplissage effectuée. Les sommets utilisés ne sont marqués comme tels qu'après la création définitive de l'arête de fermeture. Ce qui permet aux segments traités par la suite d'utiliser le premier et le dernier sommet des segments qui n'ont pu être fermés. L'utilisation d'un sommet n'est une information pertinente que pour les sommets des segments de longueur deux ou un.

Test d'intersection de l'arête de fermeture

L'arête de fermeture ne doit pas intersecter d'autres arêtes du segment fermé. Un contour planaire avec des auto-intersections ne peut être rempli qu'avec des recouvrements de faces. L'adaptation 3D du *Earclipping* ne permet pas la création de recouvrement.

Si l'arête de fermeture intersecte une arête du segment fermé, celle si n'est pas créée et le segment n'est pas remplis. L'algorithme passe alors à la fermeture du segment suivant. Une prochaine itération permettra peut-être de remplir cette partie dans une nouvelle configuration (le contour sera en partie rempli).

Le test d'intersection est fait en 2D, après la projection d'une arête du segment fermé et de l'arête de fermeture sur le plan du segment. La planéarité du segment n'étant pas strict, on souhaite détecter les intersections comme s'il l'était (Fig.6.24). La projection d'une arête se fait en projetant ses deux sommets extrémités.

Le test d'intersection est exclusif, c'est-à-dire qu'il ne compte pas comme une intersection deux arêtes s'intersectant uniquement sur un de leurs sommets extrémités. Sinon, la première et la dernière arête du segment seraient toujours intersectées avec l'arête de fermeture.

Le test est inspiré du test d'intersection entre deux triangles coplanaires de (Devillers et Guigue [2002]) et de son implémentation dans la bibliothèque CGAL [1] [2].

Le principe est de comparer le sens trigonométrique des triangles formés par les quatre sommets extrémité des deux arêtes. Soit (a,b) et (p,q) les deux arêtes à tester, on compare le sens des triangles (a,p,q) et (b,p,q) puis on compare le sens des triangles (p,a,b) et (q,a,b). Si dans les deux comparaisons les sens sont différents alors les deux arêtes s'intersectent (Fig.6.15,(a)). Si dans une des comparaisons le sens est identique, les deux arêtes ne s'intersectent pas (Fig.6.15,(b)).

1. Code CGAL : Triangle_3_Triangle_3_do_intersect.h
2. Code CGAL : Coplanar_orientation_3.h

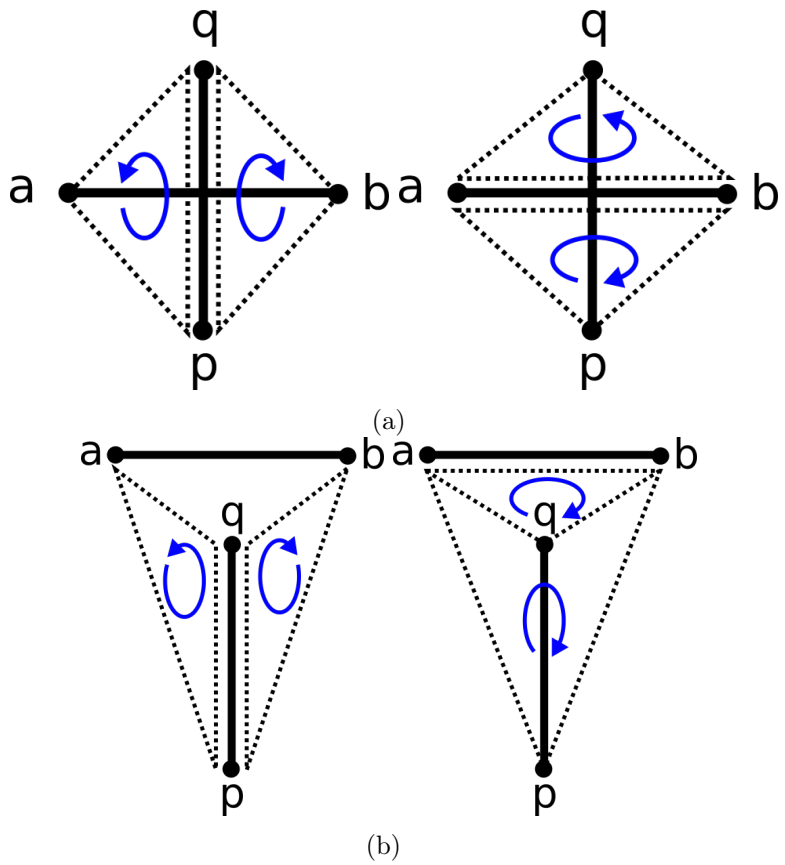


FIGURE 6.15 – Test d'intersection de deux arêtes : (a) Positif, les triangles (a,p,q) et (b,p,q) ont des sens différents et les triangles (q,a,b) et (p,a,b) aussi, (b) Négatif, les triangles (a,p,q) et (b,p,q) ont des sens différents, mais les triangles (q,a,b) et (p,a,b) ont le même sens.

Création de l'arête de fermeture

L'arête de fermeture est créée à deux conditions, qu'on n'est pas détecté d'intersection dans l'étape précédente et qu'elle n'existe pas déjà. L'arête de fermeture peut déjà exister si par exemple, un contour est coupé en deux segments planaires fermés. Les deux segments utilisent alors la même arête de fermeture, le deuxième segment traité ne doit pas créer une deuxième fois l'arête.

La structure *Halfedge* ne permet pas de créer une arête seule. On crée donc une face temporaire avec les deux sommets de l'arête de fermeture et un troisième sommet temporaire dont la position n'a aucune importance. L'ordre des sommets de la face temporaire est lui important, la demi-arête de fermeture doit être dans le sens du contour et donc sa demi-arête opposée dans la face temporaire doit être dans le sens inverse (Fig.6.16).

La face et le sommet temporaires sont supprimés dès la fin du remplissage. L'arête de fermeture sera alors rattachée à une deuxième face lors du remplissage et n'aura plus besoin de la face temporaire pour exister.

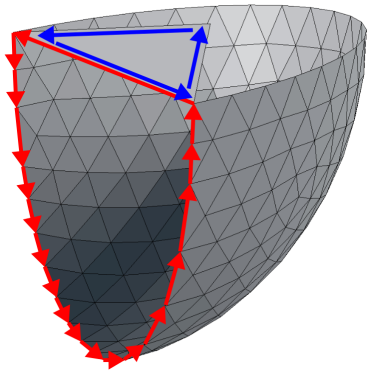


FIGURE 6.16 – Face temporaire pour créer l'arête de fermeture : Contour orienté (en Rouge), Demi-arêtes de la face temporaire orientées à l'inverse du contour (en Bleu).

Couleur et texture : Lors du remplissage, les faces nouvellement créées prendront des informations de couleur et de texture en fonction de leurs faces voisines et donc dans un premier temps des faces qui bordent le contour. La face temporaire fait partie de ces faces, mais n'a pas d'information de couleurs ou de texture.

On choisit arbitrairement de lui donner la couleur de la face qui utilise la première arête du contour. Prendre une couleur proche sur le maillage permet d'assurer une continuité de la couleur entre les faces originales et les faces créées par le remplissage.

Dans le cas où la surface est texturée, on choisit de donner une couleur unie à l'arête de fermeture. On utilise la couleur de la texture sur le sommet commun entre la face temporaire et la face dont on a pris la couleur. Comme pour la couleur on essaie d'avoir une continuité de couleur entre la texture des faces originale et la texture (unie ou étirée) des faces créées par le remplissage.

En pratique il s'agit de donner le *vertex texture* de ce sommet aux deux sommets de l'arête de fermeture sur la face temporaire (un sommet à un *vertex texture* différent par faces qui l'utilisent). Il n'est pas utile de texturer le sommet temporaire

6.1.4 Limites et seconde approche

La méthode de recherche de segment planaire présenté dans ce chapitre comporte quelques limitations :

— La méthode est basée sur les normales locales des sommets, mais celles-ci ne sont pas fiables lorsque les sommets sont trop proches d'un alignement.

— Lors de la construction d'un segment planaire sa normale moyenne est mise à jour en même temps que l'ajout de sommet dans le segment. Un segment planaire peut avoir après sa construction des sommets dont la normale locale à un angle supérieur à l'écart de planéarité, car entre l'ajout du sommet et la fin de la construction la normale moyenne a changé.

— Le résultat est impacté par le choix du sommet initial du premier sommet.

C'est limitations sont prises en compte et compensées par la méthode présentée, mais une autre approche permettrait de contourner ces limitations. Cette approche consiste à trouver un minimum de plan pour que tous les éléments du contour soient à une distance inférieure à un seuil d'au moins un des plans (Fig.6.4). Le seuil à la même fonction que l'écart de planéarité. Un autre avantage de cette approche et qu'elle permet de réunir sur un même plan des éléments non consécutifs du contour, ce qui permet une meilleure réparation sur certaines formes de contour. La méthode a aussi des inconvénients :

— La méthode travaille de manière globale sur le contour, la poursuite des arêtes saillantes du modèle original n'est pas assurée.

— Permettre d'avoir sur un même plan des éléments non consécutifs, ne garantit pas toujours l'existence d'arête de fermeture pouvant les relier en un seul contour.

Le principal problème de l'approche par plan est que même une faible distance entre un sommet et son plan suffit à faire varier dans toutes les directions la normale locale du sommet. Hors notre méthode de remplissage, le *Earclipping* adapté en 3D, utilise les normales locales des sommets du contour. Mais surtout, la qualité du remplissage dépend de la proximité entre les directions des normales locales.

6.2 Remplissage

Le *Earclipping* est un algorithme de triangulation 2D (Eberly [2002]). Chaque couple d'arêtes consécutif du contour est appelé une "oreille". Chaque oreille peut être fermée en créant une arête entre le sommet de départ de la première arête de l'oreille et le sommet d'arrivée de sa deuxième arête. L'algorithme consiste à définir des conditions pour déterminer quelle oreille doit être fermée en premier. Puis après une fermeture à mettre à jour la liste des oreilles.

En effet la fermeture d'une oreille modifie l'oreille précédente et suivante sur le contour. Et l'on recommence en cherchant la nouvelle oreille à fermer. L'algorithme s'arrête quand il ne reste plus que trois oreilles puisqu'elles sont déjà un triangle.

6.2.1 Normale au trou

L'application du *Earclipping* à un contour 3D même planaire nécessite plusieurs adaptations. Dans l'algorithme 2D, la première condition pour la fermeture d'une oreille est si son angle intérieur est saillant ([0°-180°]).

Pour distinguer l'angle intérieur de l'angle extérieur sur un contour 2D, on utilise l'orientation des arêtes de l'oreille. Dans le sens d'une arête, l'intérieur du contour est à sa gauche et l'extérieur à sa droite. L'avantage est que l'information se trouve localement au niveau des arêtes et donc des oreilles.

Sur un contour 3D distinguer l'angle intérieur de l'angle extérieur n'est pas possible localement en regardant l'orientation des arêtes de l'oreille. Une même configuration locale a un résultat différent selon le contexte global du contour (Fig.6.17). La différence vient qu'en 3D la notion de droite et gauche n'est pas définie, car il n'y a pas de plan. En 2D il y a toujours un unique plan.

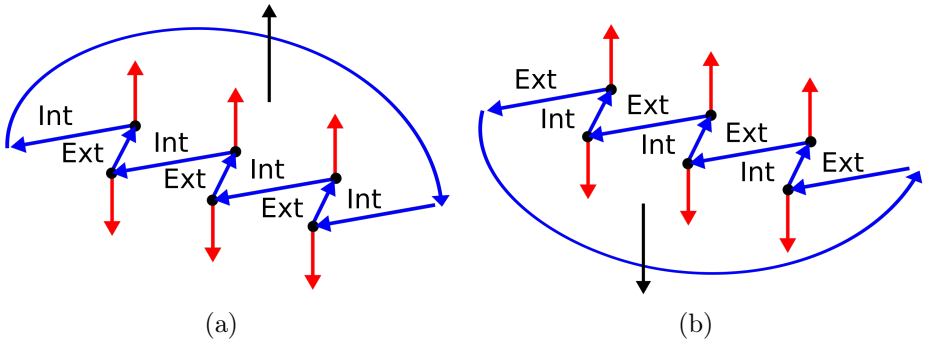
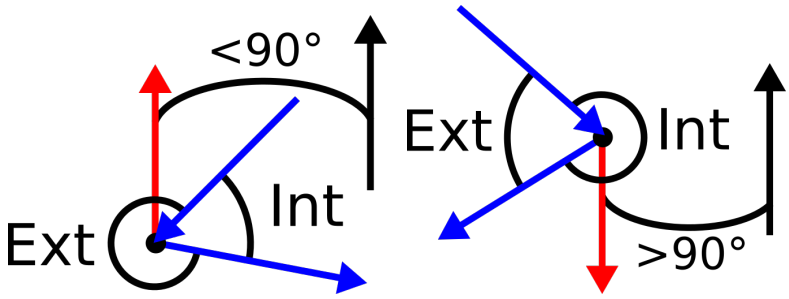


FIGURE 6.17 – Deux possibilités pour une même configuration locale de contour en fonction du contexte global du contour. La normale au trou change d'orientation en fonction du contexte (en Noir).

L'idée est d'utiliser le plan du trou entouré par le contour pour en 3D distinguer les angles intérieurs des angles extérieurs au niveau des oreilles (localement). Le plan est défini par sa normale appelée "normale au trou". Chaque oreille possède une normale locale correspondante à la normale du plan défini par les deux arêtes de l'oreille (Ce sont les mêmes normales locales que celle des étapes précédentes). La distinction des angles se fait en comparant la normale locale d'une oreille et la normale au trou.

Si les normales ont un écart inférieur à 90° alors c'est que l'angle intérieur est saillant et qu'on peut fermer l'oreille (Fig.6.18,(a)).

Si les normales on un écart supérieur à 90° on ne peut pas fermer l'oreille qui devra attendre d'être modifiée par la fermeture d'une oreille voisine (Fig.6.18,(b)).



(a) (b)

FIGURE 6.18 – Angle intérieur d'une oreille en fonction de la normale au trou (en Noir) : (a) L'angle intérieur est l'angle saillant, (b) L'angle intérieur est l'angle rentrant.

La normale au trou, est calculée comme la moyenne pondérée des normales locales des oreilles. La moyenne est pondérée en fonction de la longueur des oreilles. La longueur d'une oreille et la somme de la longueur des deux arêtes de l'oreille.

Comme pour la normale des segments planaire, on ne prend pas en compte l'orientation des normales locale dans le calcul de la moyenne. Les normales locales des oreilles sont réorientées si elles ont un écart de plus de 90° avec la normale de la première oreille traitée dans le calcul. Le contour étant planaire la direction des normales des oreilles ne varie pas énormément, prendre arbitrairement la première oreille comme référence n'a aucun impact.

La réorientation se fait par l'ajout d'un facteur -1 à la normale de l'oreille dans le calcul, l'oreille garde sa normale inchangée. La normale au trou a dans un premier temps l'orientation de la première normale locale ajoutée dans la moyenne.

La normale au trou est différente de la normale au plan du segment planaire dont provient le contour à remplir. Plusieurs sommets et arêtes ont été rajoutés lors de l'étape de fermeture du segment. La normale du segment planaire n'est pas pondérée en fonction de la longueur des arêtes. L'orientation de la normale du segment planaire n'a pas d'importance pour la recherche de segment planaires. La normale du segment planaire garde l'orientation de la normale locale du premier sommet ajouté dans le segment.

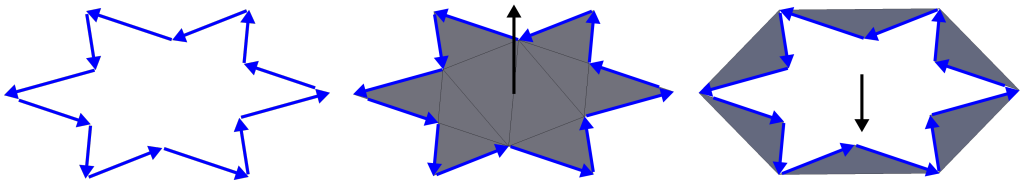
L'orientation de la normale au trou est très importante, elle détermine l'intérieur et l'extérieur des oreilles. Une mauvaise orientation provoquera la fermeture des oreilles ayant un angle saillant à l'extérieur du trou. Le remplissage va créer un contour convexe sur l'extérieur du contour original. Ce contour

convexe ne pourra pas être fermé, les oreilles auront leurs angles rentrants à l'extérieur du trou. Le remplissage se terminera avec de nombreuses arêtes de bord restantes (Fig.6.19,(c)).

L'orientation de la normale au trou est choisie comme celle des oreilles ayant la somme de leurs longueurs la plus longue. Le choix est binaire entre l'orientation des oreilles non retournées et l'orientation des oreilles retournées. Dans le premier cas l'orientation de la normale au trou ne change pas dans le deuxième cas on inverse son orientation.

Cette méthode ne permet pas toujours de trouver la bonne orientation, par exemple un contour en forme d'étoile aura exactement la même longueur pour les deux orientations (Fig.6.19). Certain contour peuvent n'avoir aucune bonne orientation, par exemple si le contour s'auto-intersecte ou s'il n'est pas assez planaire. Dans ce cas quelque soit l'orientation choisie, le remplissage finira avec des arêtes de bord restantes.

Une autre méthode plus sécuritaire est d'appliquer le remplissage avec chacune des deux orientations est de comparer le résultat. Le remplissage ayant le moins d'arêtes de bord en sortie est conservé pour la suite de la réparation.



(a) (b) (c)

FIGURE 6.19 – (a) Contour en forme d'étoile, (b) Remplissage avec le bon choix d'orientation de la normale au trou (en Noir), (c) Remplissage avec le mauvais choix d'orientation de la normale au trou.

6.2.2 Choix d'une oreille

La sélection de l'oreille à fermer, ce fait sur le critère de l'angle intérieur le plus petit. Toutes les oreilles sont triées par ordre croissant de leur angle saillant. Plusieurs conditions vont déterminer si une oreille peut être fermée, on teste dans l'ordre chaque oreille pour trouver celle qui remplit ces conditions. L'oreille sélectionnée est fermée par la création d'une face composée des trois sommets de l'oreille. Les sommets sont ordonnés dans le même sens que pour l'oreille afin que la face créée ait la même orientation que le reste de la surface. La structure *Halfedge* ne permet pas la création d'une face ne respectant pas cette orientation. La liste des oreilles doit être mise à jour après la création de la face de fermeture et avant la sélection d'une nouvelle face (Fig.6.20).

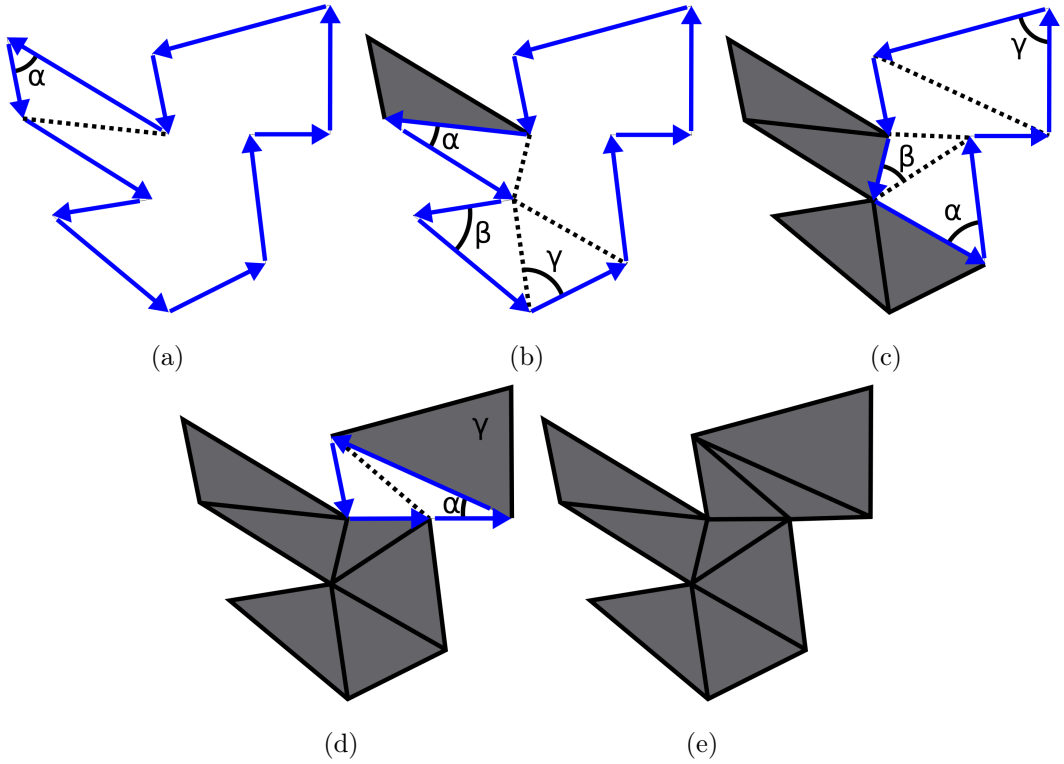


FIGURE 6.20 – Étape d'un Earclipping 3D : (α, β, γ) Angles internes les plus petits dans un ordre croissant.

Les conditions de fermeture d'une oreille sont les suivantes :

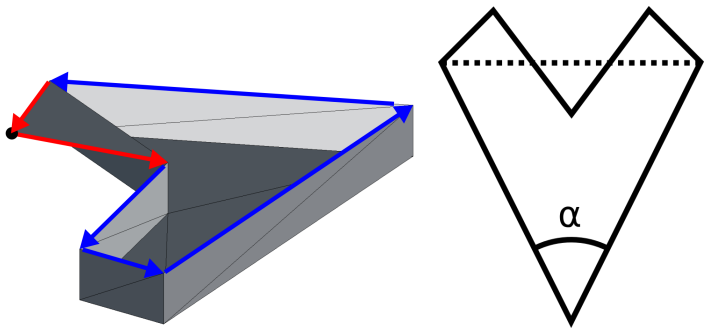
— L'angle saillant doit être à l'intérieur du contour. La normale de l'oreille doit avoir un écart avec la normale au trou inférieur à 90° (Fig.6.18,(a)). Cette condition est détaillée dans la section précédente (Section.6.2.1).

— La face constituée des trois sommets de l'oreille ne doit pas exister. Si elle existe, il n'est pas possible de fermer l'oreille puisque la fermeture consiste à créer cette face. La structure *Halfedge* ne permet pas de doubler une face, ce qui créerait un recouvrement et possiblement une zone sans volume.

  Cette face existe si les deux arêtes de l'oreille appartiennent à la même face, la face a deux arêtes de bord du contour. Elle peut être à l'extérieur du contour, dans ce cas la condition précédente interdira déjà la fermeture. Elle peut être à l'intérieur du contour (Fig.6.21,(a)), dans ce cas le remplissage peut créer un recouvrement et possiblement une zone sans volume.

— La face de fermeture n'intersecte pas le reste du contour. Dans la version 2D, on teste qu'il n'y a aucun sommet du contour à l'intérieur de l'oreille fermée. Pour la version 3D on teste que l'arête ajoutée par la face de fermeture n'intersecte pas une arête du contour (Fig.6.21,(b)).

Le test d'intersection reprend la méthode utiliser pour l'arête de fermeture des segments planaire (Section.6.1.3). L'arête de la face de fermeture et les arêtes du contour sont projetées sur le plan du trou. Le test d'intersection est toujours exclusif.



(a) (b)

FIGURE 6.21 – Conditions de non-fermeture d'oreille : (a) Une face utilise les trois sommets de l'oreille (en Rouge), (b) L'arête ajoutée par la fermeture (en Pointillé) intersecte d'autres arêtes du contour.

Couleur et texture : La couleur des faces créées par le remplissage doit être dans la continuité des couleurs des faces du modèle 3D qui bordent le trou. La face de fermeture possède deux faces voisines et ne peut prolonger la couleur que d'une seule. On choisit en priorité une face originale du modèle, car sa couleur est définie par le concepteur du modèle 3D. Si les deux ou aucune des deux n'est une face originale alors on prend arbitrairement celle qui utilise la première arête de l'oreille.

Pour des faces voisines texturées, on ne peut pas texturer la face de fermeture avec une zone de la texture. Pour garder une continuité entre les couleurs des faces texturées qui bordent le contour et les faces du remplissage, on décide d'étirer les couleurs de la texture présente sur les arêtes de bord du contour. La face de fermeture ne peut étirer les couleurs que d'une de ces deux arêtes communes à une face voisine. Le choix de l'arête et donc de la face voisine est le même que pour la couleur.

En pratique la face de fermeture attribue aux deux sommets de l'arête choisie, les mêmes *vertex texture* que ceux de sa face voisine. Le troisième sommet se voie attribuer un *vertex texture* positionner au milieu des deux précédents. C'est le fait que les trois *vertex texture* de la face de fermeture soient alignés sur la texture qui créer l'étirement.

Note : même si les faces sont texturées on propage aussi leurs informations de couleurs. Ces couleurs sont visibles à l'impression si la texture à des zones transparentes.

6.2.3 Mise à jour des oreilles

La création de la face de fermeture d'une oreille change le contour, en remplaçant les deux arêtes de bord de l'oreille par une nouvelle arête de bord créer par la face de fermeture. Certaines oreilles sont alors modifiées, il faut les mettre à jour avant de sélectionner la prochaine oreille à fermer.

Une oreille est définie par sa première demi-arête de bord (la deuxième demi-arête de bord étant la demi-arête suivante de la première dans la structure *Halfedge*), la valeur de l'angle saillant entre ses deux arêtes et sa normale locale.

Mise à jour des oreilles après fermeture d'une oreille (Fig.6.22) :

— L'oreille fermée est supprimée, ses deux demi-arêtes de bord ont été corrigées.

— L'oreille suivante est supprimée, sa première arête de bord a été corrigée.

— L'oreille précédente a sa deuxième demi-arête de bord corrigée et automatiquement remplacée par la structure *Halfedge*) à la création de la face de fermeture. L'angle saillant doit être recalculé. La normale locale doit être recalculée.

— Une oreille est créée avec pour première demi-arête de bord, la nouvelle demi-arête de bord créée par la face de fermeture. L'angle saillant doit être calculé. La normale locale doit être calculée.

La mise à jour peut faire changer l'orientation d'une normale locale pouvant par exemple rendre possible la fermeture d'une oreille qui ne l'était pas.

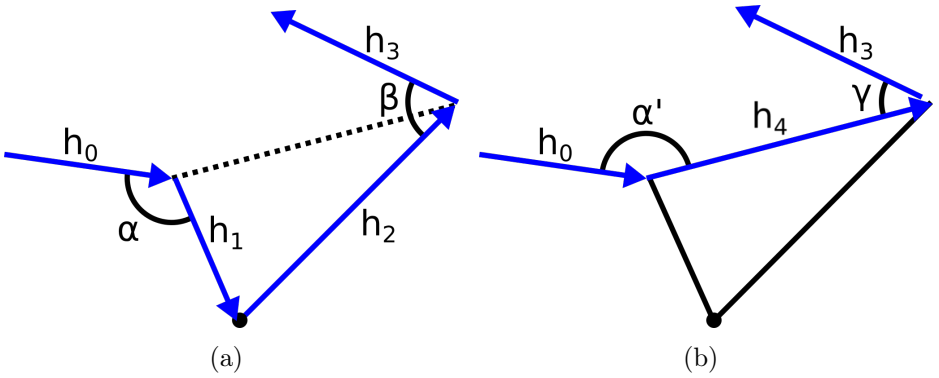


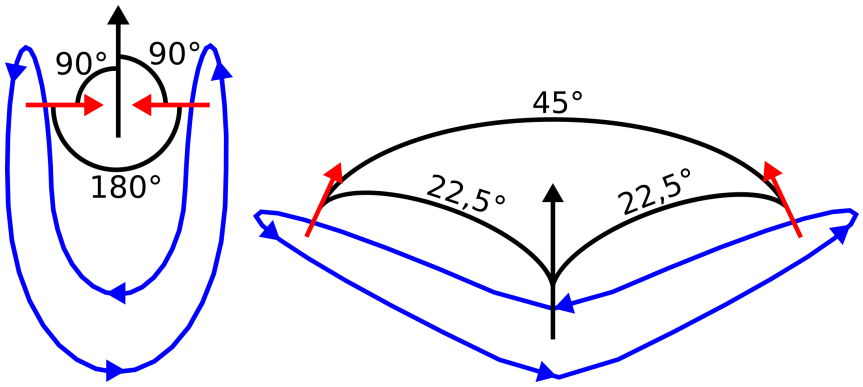
FIGURE 6.22 – Étapes de mise à jour des oreilles après fermeture de l'oreille (h1,h2) : L'oreille (h0,h1,α) devient (h0,h4,α'), L'oreille (h2,h3,β) est supprimée, L'oreille (h4,h3,γ) est ajoutée.

6.2.4 Conditions sur le *Earclipping* 3D

L'écart de planéarité que peut supporter le *Earclipping* 3D sur le contour qu'il doit remplir est théoriquement assez grand. Le fonctionnement du *Ear-*

clipping 3D est conditionné à une première chose, la distinction du côté du trou (intérieur ou extérieur) où se situe l'angle saillant de chaque oreille. Deux oreilles ayant un angle saillant à l'intérieur du trou, appelées "oreille interne", ne doivent pas avoir un angle supérieur à 90° entre leur normale locale et la normale au trou. La différence maximale entre les normales locales des deux oreilles doit être strictement inférieures à 180° (Fig.6.23,(a)). Le contour contenant ces deux oreilles aura la forme d'un demi-cercle dans l'espace.

En pratique un tel contour se heurterait au calcul de sa normale au trou qui devrait être exactement entre les normales de ces deux oreilles extrêmes. Par expérimentation on utilise un maximum de 45° entre les normales de deux oreilles internes (Fig.6.23,(b)). On retrouve notre écart de planéarité de $\frac{45°}{2} = 22.5°$ utilisé pour la recherche de segment planaire.



(a) (b)

FIGURE 6.23 – (a) Contour avec des normales d'oreille interne (en Rouge) à 180° d'écart, (b) Contour avec des normales d'oreille interne à 45° d'écart.

Le *Earclipping* 3D ne peut pas remplir un contour dont la projection sur le plan du trou provoque une auto-intersection. Ce type de contour a forcément une oreille interne avec une normale locale opposée à la normale au trou au niveau de l'auto-intersection (Fig.6.24). Une telle oreille ne peut être fermée par le *Earclipping* 3D. Ces contours sont en majorités traités par la recherche de segments planaires.

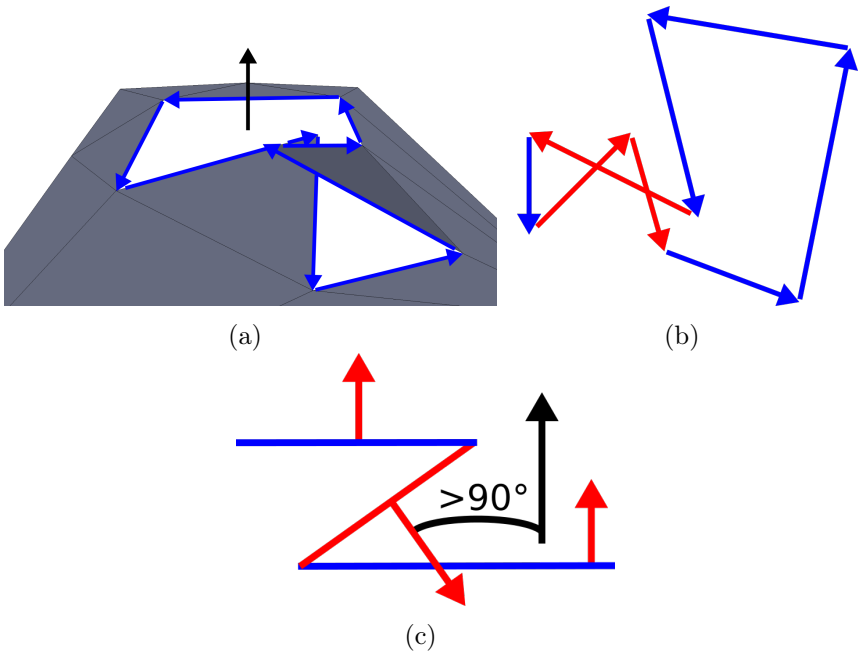


FIGURE 6.24 – (a) Contour planaire, (b) Projection du contour avec une intersection sur les arêtes (en Rouge), (c) Face originale (en Bleu), Face de fermeture d'une oreille (en Rouge) impossible.

Chapitre 7

Réparation par épaississement

Dans le (Chap.5) on a vu la classification des contours en deux types : les contours entourant un vide aussi appelés trous dans la surface, et les contours entourant la surface (Fig.5.10,(a)). Les premiers ont été traités par l'étape de remplissage du chapitre 4. La réparation par épaississement va traiter les arêtes de bords appartenant au deuxième type de contour. Les surfaces entourées par ce type de contours sont appelées des surfaces fines, car elles ont une épaisseur nulle. L'épaississement va permettre de donner une épaisseur à ces surfaces et donc un volume interne (Fig.5.10,(c)).

L'épaisseur donnée par l'épaississement doit être suffisante pour que l'objet imprimé n'ait pas de risque de cassures. Les imprimantes 3D ont selon leurs technologies et les matières utilisées une limite d'épaisseur. Cette problématique sera traitée dans le (Chap.8). Mais dans le cadre de ce chapitre, on veut que l'épaisseur donnée soit directement suffisante pour ne pas créer un problème même si une étape ultérieure de la chaîne pourrait le résoudre.

Les méthodes d'épaississement présentées dans ce chapitre sont appelées en anglais *Offset*.

7.1 Surfaces parallèles (*Offset surfaces*)

7.1.1 Définitions

Une première utilisation des surfaces parallèles est l'*offset* des surfaces paramétriques proposée par (Farouki [1985]).
Soit S une surface paramétrique, $S = r(u, v)$.
Soit n , le vecteur normal, $n = \vec{r_u} \times \vec{r_v}$.
Soit d une distance, appelée distance de l'*offset*.
La surface parallèle de S, appelée encore surface *offset* de S, est définie comme :

$$S_o = r_o(u,v) = r(u,v) + d * \frac{\vec{r_u} \times \vec{r_v}}{|\vec{r_u} \times \vec{r_v}|} \tag{7.1}$$

Une illustration de l'*offset* d'une surface paramétrique est donnée sur la (Fig.7.1).

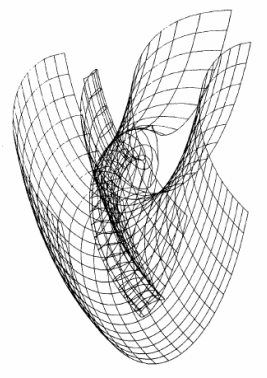
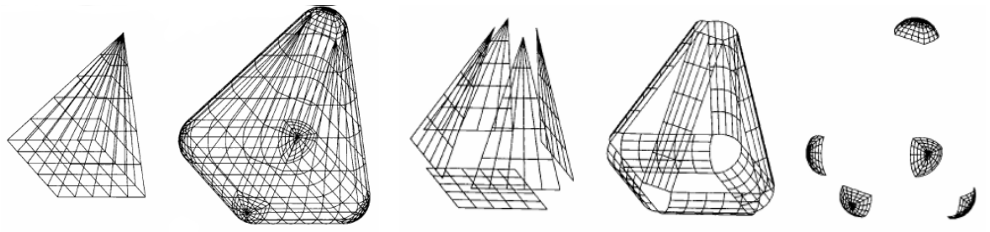


FIGURE 7.1 – *Offset* d'une surface paramétrique. D'après Fig.3 de (Farouki [1985]).

Pour construire l'*offset* d'une surface polyédrique, (Farouki [1985]) opère en deux étapes. Dans un premier temps il calcule les décalages des sommets, des arêtes et des faces de la surface polyédrique (Fig.7.2,(b)), et dans un deuxième temps, il les relie pour obtenir une seule surface polyédrique (Fig.7.2,(a)).



(a) (b)

FIGURE 7.2 – "Offset" d'une surface polyédrique. D'après Fig.6 de (Farouki [1985]).

L'utilisation des *offset surfaces* sur un modèle 3D peut introduire des défauts dans les maillages comme des trous, des intersections et des recouvrements. En vue d'une impression 3D ces défauts ne peuvent être conservés à l'exception de certaines intersections comme vu précédemment. De plus certains de ces défauts sont corrigés dans des étapes précédentes de la chaîne et on ne peut en reproduire de nouveau à cette étape.

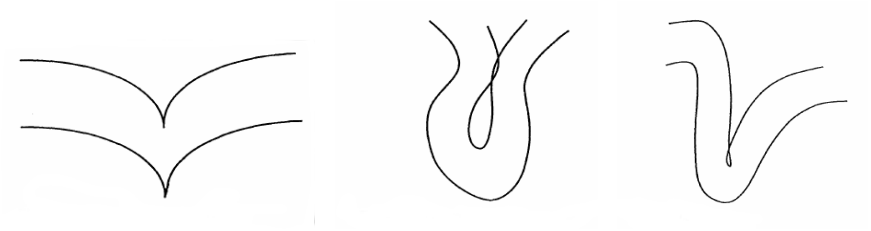
7.1.2 Cas dégénérés

L'étude des *offset surfaces* a démontré la difficulté de les produire de façon à ce qu'elles ne provoquent pas les défauts vus précédemment. Pour des exemples on peut se référer à (Farouki [1985]; Rossignac et Requicha [1986]; Pham [1992]; Maekawa [1999]; Lee [2009]). Des illustrations en $2D$ sont données sur la (Fig.7.3) et la (Fig.7.4).

Ainsi par exemple sur la (Fig.7.3,(a)) on peut voir le cas où dans un point de rebroussement, la direction de décalage suivant la normale est non-unique en ce point. Si la distance de décalage dépasse un seuil, le rayon de courbure locale, la courbe d'*offset* s'auto-intersecte, voir la (Fig.7.3,(b)(c)).

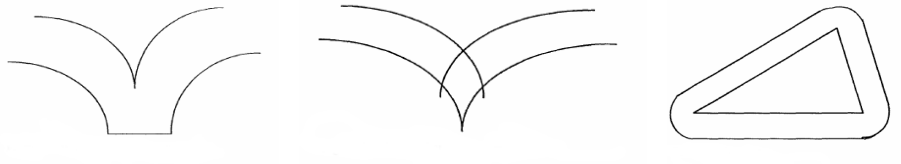
La forme de la courbe *offset* ne suit pas toujours la forme de la courbe initiale, des segments peuvent disparaître ou apparaître, voir la (Fig. 7.4).

La plupart des méthodes de construction des surfaces *offset* produisent ces problèmes.



(a) (b) (c)

FIGURE 7.3 – Cas dégénérés : (a) Orientation non-unique de la normale, (b) Auto-intersection, (c) Boucle et corne. D'après Fig.1, Fig.2 et Fig.3 de (Pham [1992]).



(a) (b) (c)

FIGURE 7.4 – Cas dégénérés : (a) Disparition d'un segment, (b) Suppression d'un segment, (c) Ajout des segments complémentaires. D'après Fig.4 et Fig.5 de (Pham [1992]).

7.1.3 Classification

Les méthodes de construction des surfaces *offset* peuvent être regroupées en trois grandes familles :

— L'approche surfacique est basée sur le calcul direct de l'*offset* des composantes (sommet et/ou arête et/ou surface) et le recollement de l'ensemble de ces *offsets* pour obtenir le résultat final.

— L'approche volumique est basée sur un calcul de volume englobant de la surface "offset" suivi par l'échantillonnage de ce volume et l'extraction d'une approximation de la surface "*offset*".

— L'*offset* peut être calculé aussi à l'aide des opérations de Minkowski, un cadre plus général pour la manipulation de modèles géométriques.

Un exemple représentatif de chaque famille sera discuté par la suite. L'objectif n'est pas d'être exhaustif, mais de donner une meilleure compréhension des choix de l'utilisation de *offset* retenus dans notre chaîne de traitements.

7.2 Approche surfacique

L'*offset* d'une surface polyédrique selon l'approche surfacique suit les étapes suivantes :

1. Construction des *offsets* (décalages) de chaque composante.
2. Fusion/subdivision des parties se recouvrant pour la suppression des chevauchements, des auto-intersections et des cas dégénérés.

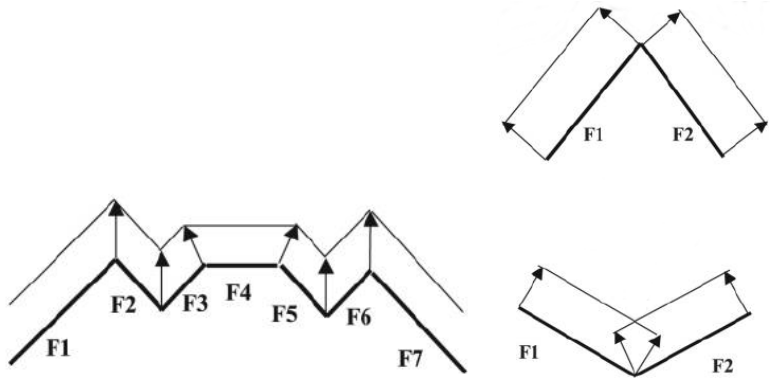
Un exemple de cette approche est la méthode de **"offset" par décalage de sommets** proposée par (Qu et Stucker [2003]).

La méthode de (Qu et Stucker [2003]) construit en premier les sommets P'_i, *offset* de l'ensemble de P_i, les sommets de la surface initiale. Soit d_{offset} la distance de l'*offset* et soit $\vec{N_{offset}}$ la normale au sommet P_i calculée comme une somme pondérée des normales des faces voisines à P_i.

$$P'_i = P_i + \vec{N_{offset}} * d_{offset} \tag{7.2}$$

Dans un deuxième temps, les sommets S'_i construits par décalage sont reliés pour former les nouvelles faces de la surface *offset*. Une illustration est donnée sur la (Fig.7.5,(a)).

L'avantage principal de cette méthode est sa simplicité et le fait qu'elle ne produit pas des cas dégénérés comme ceux illustrés sur la (Fig.7.5,(b)) et la (Fig.7.4). En effet, seuls les sommets sont décalés ce qui équivaut à changer la géométrie de l'*offset surfaces* par rapport à la surface initiale, mais à préserver sa structure topologique à savoir les voisinages sommet/arêtes/faces. Cependant, selon le choix de la direction et la distance du décalage cette méthode peut produire des intersections et des recouvrements dans le maillage final.



(a) (b)

FIGURE 7.5 – (a) "Offset" par décalage de sommets, (b)(c) Création de défauts. D'après Fig.1 et Fig.2 de (Qu et Stucker [2003]).

7.3 Approche volumique

L'*offset* d'une surface polyédrique selon approche volumique suit les étapes suivantes :

1. Construction d'une subdivision d'un volume englobant.
2. Classification des cellules de la subdivision en fonction de leur appartenance ou non à la surface en tant que vides, pleines ou partiellement occupées.
3. Construction d'un *offset* à partir des cellules partiellement occupées.

Un exemple de cette approche est l'article *Offsetting of Polygonal Model based on Layered Depth-Normal Images* de (Chen et Wang [2011]).
Soit S un solide, soit $S \uparrow^* r$ un solide après l'*offset* positif de S à une distance de r. $S \uparrow^* r$ est défini comme il suit :

$$S \uparrow^* r = \{p : d(p, S) \leq r\} \tag{7.3}$$

$$d(p, S) = \inf_{q \in S} \|p - q\| \tag{7.4}$$

Soit $\partial(S \uparrow^* r)$ la surface frontière de $S \uparrow^* r$. De part la définition (7.3) on a :

$$\partial(S \uparrow^* r) \subset \{p, d(p, S) = r\} \tag{7.5}$$

La méthode proposée par (Chen et Wang [2011]), en premier, construit un sur-ensemble de $\partial(S \uparrow^* r)$.
Soit $V(S)$, $E(S)$ et $F(S)$, les ensembles respectivement des sommets, des arêtes et des faces de S. On considère $\partial(S)$:

$$\partial(S) = V(S) \bigcup E(S) \bigcup F(S) \tag{7.6}$$

Pour chaque ensemble $V(S)$, $E(S)$ et $F(S)$ on peut construire les sur-ensembles suivants :
$V\|_r^+$, l'ensemble de boules $B(v,r)$ de rayon r et de centre un sommet v de S :

$$V\|_r^+ = \{p : p \in B(v,r), v \in V(S)\} \tag{7.7}$$

$E\|_r^+$, l'ensemble de cylindres $C(e,r)$ de rayons r et de hauteur une arête e de $E(S)$:

$$E\|_r^+ = \{p : p \in C(e,r), e \in E(S)\} \tag{7.8}$$

$F\|_r^+$, l'ensemble de faces décalées, chaque face étant décalée le long de n, sa normale, à une distance donnée r :

$$F\|_r^+ = \{p : p = q + r \cdot n, q \in f, f \in F(S)\} \tag{7.9}$$

En substituant dans (7.5) on obtient :

$$\partial(S \uparrow^* r) \subset F\|_r^+ \bigcup E\|_r^+ \bigcup V\|_r^+ \tag{7.10}$$

Ainsi, $F\|_r^+ \bigcup E\|_r^+ \bigcup V\|_r^+$ définit un sur-ensemble de $\partial(S \uparrow^* r)$.

Une fois le sur-ensemble de $\partial(S \uparrow^* r)$ construit, on procède à son échantillonnage. Soit C une grille uniforme de voxels. Chaque voxel $c \in C$, est défini par son centre (x_c, y_c, z_c) et la résolution de l'échantillonnage $size$. On échantillonne le sur-ensemble $F\|_r^+ \bigcup E\|_r^+ \bigcup V\|_r^+$ pour chaque voxel en fonction de leur point commun. On nomme un échantillon S_c.

L'étape suivante est un filtrage de cet échantillonnage pour ne retenir que les points appartenant à $\partial(S \uparrow^* r)$. Pour cela (Chen et Wang [2011]) utilise des images de profondeur ("Layered Depth-Normal Images", LDNI), basées sur de lancer de rayons le long des axes X, Y et Z. Pour chaque intersection d'un rayon avec S_c, l'orientation de la normale au point de l'intersection permet de rejeter les points appartenant à l'intérieur de $S \uparrow^* r$. Enfin, la surface frontière $\partial(S \uparrow^* r)$ est reconstruite à partir de points validés. L'algorithme est illustré sur la (Fig.7.6).

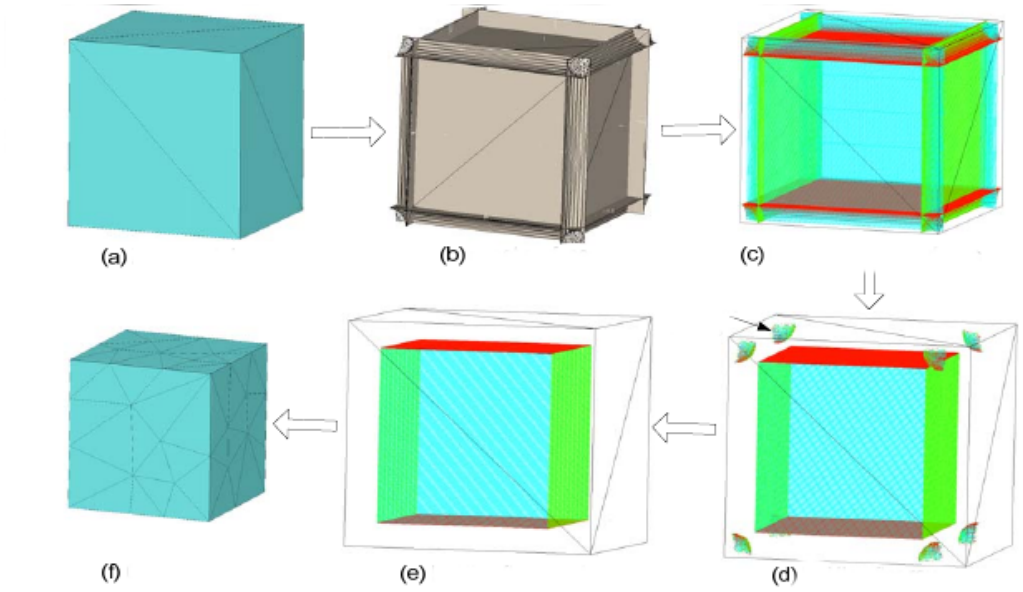


FIGURE 7.6 – (a) Modèle initial, (b) Construction du sur-ensemble, (c) Construction des LDNIs, (d) Filtre "Lancer de rayons", (e) Filtre "offset", (f) Extraction de la surface "offset". D'après Fig.14 de (Chen et Wang [2011]).

Le point fort de cette approche est la possibilité de transformer des surfaces de forme géométrique complexe. Le filtrage permet aussi de produire directement des maillages imprimables. Cependant, les modèles numériques ainsi obtenus sont des approximations des modèles originaux.

7.4 Approches hybrides

(Rossignac et Requicha [1986]) introduit la notion de *s-offsetting"*, *offset* d'un solide S. Un solide peut être étendu ou au contraire, être contracté, en fonction de la direction de décalage. Pour "étendre" S d'une distance, r il suffit d'ajouter tous les points qui se trouvent à une distance inférieure ou égale r. Ainsi on obtient :

$$S \uparrow^* r = \{p : \exists q \in S, \|p - q\| \leq r\} \tag{7.11}$$

Soit $B^*(p, r)$ une boule centrée dans p et de rayon r. On peut re-écrire la définition de $S \uparrow^* r$ comme :

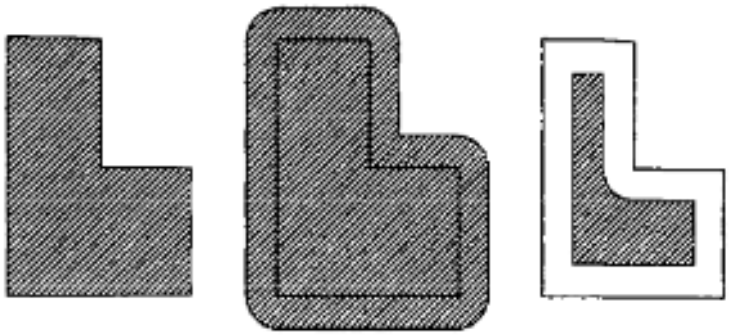
$$S \uparrow^* r = \bigcup_{p \in S} B^*(p, r) \tag{7.12}$$

Cela permet de considérer $S \uparrow^* r$ comme un volume extrudé par $B^*(p, r)$ quand p est déplacé dans S. On parle dans ce cas de solide étendu *positive*

s-offsetting (Fig.7.7,(b)).
On peut définir $S \downarrow^* r$ comme :

$$S \downarrow^* r = c^*((c^*S) \uparrow r) \tag{7.13}$$

Avec $c^*(A)$ on dénote le complément du volume A dans E^3. On parle alors de solide contracté *negative s-offsetting* (Fig.7.7,(c)).



(a) (b) (c)

FIGURE 7.7 – (a) Objet initial, (b) Objet étendu, (c) Objet contracté. D'après Fig.1 de (Rossignac et Requicha [1986]).

Les opérations de Minkowski sont utilisées pour généraliser la notion de *s-offsetting*. Soit $A \subset E^3$, $B \subset E^3$, ensembles fermés et réguliers.
On définit la **somme de Minkowski** $A \oplus B$ comme :

$$A \oplus B := \{a + b | a \in A, b \in B\} \tag{7.14}$$

Avec $a + b$ une somme vectorielle entre deux points.
On définit la **différence de Minkowski** $A \otimes B$ comme :

$$A \otimes B := \overline{\overline{A} \oplus B} \tag{7.15}$$

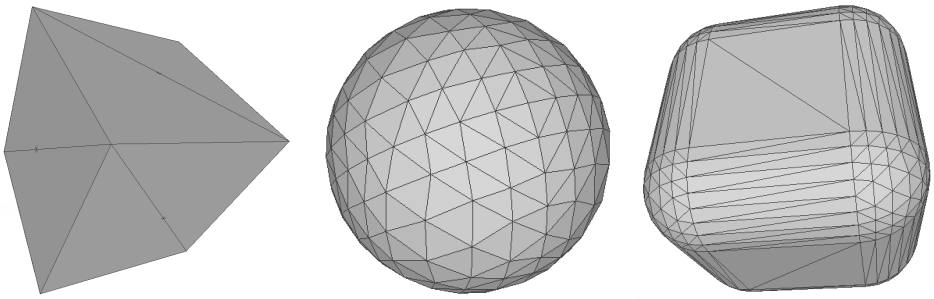
On peut alors redéfinir $S \uparrow^* r$ et $S \updownarrow^* r$ avec b_r une boule ouverte de rayon r :

$$S \uparrow^* r = S \oplus b_r \tag{7.16}$$

$$S \downarrow^* r = S \otimes b_r \tag{7.17}$$

L'algorithme proposé par (Rossignac et Requicha [1986]) pour générer $S \uparrow^* r$ ou $S \downarrow^* r$ est composé par :

1. Construction de l'ensemble de faces provisoires contenues dans $\partial(S \uparrow^* r)$ ou $\partial(S \downarrow^* r)$.
2. Construction de l'ensemble d'arêtes provisoires intersections de paires de faces provisoires.



(a) (b) (c)

FIGURE 7.8 – (a) Solide A, (b) Solide B, (c) $S = A \oplus B$, somme de Minkowski de A et B.

3. Construction de la représentation par frontière de la surface "*offset*".

On considère cette méthode comme hybride parce que d'une part à l'aide des opérations de Minkowski on construit une enveloppe volumique englobant $S \uparrow^* r$, et d'autre part, on évalue $\partial(S \uparrow^* r)$ à l'aide du calcul des intersections des faces et arêtes provisoires sous-jacentes.

Une illustration de *s-offsetting*, réalisée à l'aide des opérations de Minkowski, est donnée sur la (Fig.7.8). Les exemples sont construits avec CGAL 4.14 - 3D Minkowski Sum of Polyhedra par P.Hachenberger.

7.5 Notre approche

Pour construire un épaississement des surfaces fines, notre choix s'est porté sur la méthode de (Qu et Stucker [2003]). Ce choix est justifié par :

— Est applicable directement sur une surface avec bord ne définissant pas un solide, et permet de relier la surface initiale et son décalage pour former un volume.

— Cette méthode préserve la structure topologique de la surface initiale sur la surface décalée.

— Les dégénérescences potentiellement introduites par l'épaississement sont des auto-intersections.

Les grandes lignes de l'algorithme sont données dans (Algo.11).

Dans notre chaîne de traitements, l'épaississement des surfaces fines intervient au moment où tous les bords du maillage à réparer sont identifiés et sont classifiés selon la réparation souhaitée soit comme des trous à combler soit comme des bords de surface à épaissir. L'épaississement est effectué en

deux étapes. En premier, on construit l'*offset* de la surface à épaissir en décalant ses sommets. Puis en les reliant à l'aide d'arêtes et de faces identiques à la surface initiale. Ensuite, on assemble la surface initiale et sa surface *offset* par une suite de faces reliant les bords des deux surfaces. Au final, on obtient une surface sans bords qui englobe un volume et qui représente un maillage imprimable.

Algorithme 11 Épaississement d'une surface fine

Entrées: S surface fine (sommet,arête,face), d distance de l'*offset*
Sorties: S^* surface sans bord frontière d'un volume

Copier S dans S^*
pour chaque sommet $s_i \in S$ **faire**
  pour chaque face f_s utilisant s **faire**
    Calculer la normale au sommet $n_{si} += n_{fs}$
  fin pour
  Normaliser n_{si}
  Créer le sommet $s_i^* = s_i + d * n_{si}$
  Ajouter le sommet s_i^* dans S^*
fin pour
pour chaque face $f_i \in S$ **faire**
  pour chaque sommet $s_i \in f_i$ **faire**
    Ajouter le sommet s_i^* dans f_i^*
  fin pour
  Créer la face f_i^*
fin pour
pour chaque face de bord $fb_i \in S$ **faire**
  pour chaque arête de bord $ab_{fbi} \in fb_i$ **faire**
    Créer deux faces triangulaires reliant ab_{fbi} et son offset ab_{fb^*i}
  fin pour
fin pour

Le résultat de la méthode de (Qu et Stucker [2003]) sur la surface fine est montré sur (Fig.7.9,(a)). Avec une distance de l'*offset* inférieure au rayon de la courbure, la surface devient bien une surface sans bord, frontière d'un volume (Fig.7.9,(b)). En revanche si la distance de l'*offset* devient supérieure au rayon de la courbure alors la surface peut comporter plusieurs défauts. Principalement des auto-intersections rendant la surface non orientable, mais aussi des recouvrements (Fig.7.9,(c)).

En pratique ces cas arrivent, car l'épaisseur minimum pour l'impression est la même pour tout l'objet. Or l'objet surtout s'il est imprimé en petite taille peut comporter des surfaces représentant des détails d'une taille inférieure à celle de l'épaisseur minimum. Comme il a été vu au début de ce chapitre, la

distance de l'*offset* est basée sur l'épaisseur minimum pour l'impression. Du coup plus une surface est petite par rapport à la distance de l'*offset* moins sa courbure peut être grande sans provoquer une auto-intersection lors de l'*offset*.

Plusieurs pistes sont possibles pour pallier à ce problème :

— Adapter la distance de l'*offset* à chaque sommet en fonction du rayon de la courbure locale au sommet décalé. Le problème de cette solution est que l'on crée des fragilités dans l'objet. Mais on peut préférer une topologie correcte pour ensuite s'attaquer à la fragilité dans l'étape suivante de la chaîne. On peut aussi noter que si l'épaisseur intersecte un autre volume de l'objet alors la fragilité peut disparaître, car c'est l'épaisseur totale qui compte.

— Découper la surface non orientable le long des auto-intersections pour former plusieurs surfaces orientables. En effet une surface non orientable est frontière de plusieurs volumes positifs et négatifs. L'idée est de retrouver les surfaces frontières de ces différents volumes. L'avantage c'est que cette méthode permet de corriger les auto-intersections déjà présentes dans le modèle original. Le désavantage c'est que les volumes créés par une distance d'*offset* trop grande n'ont aucune logique dans l'esthétique du modèle original.

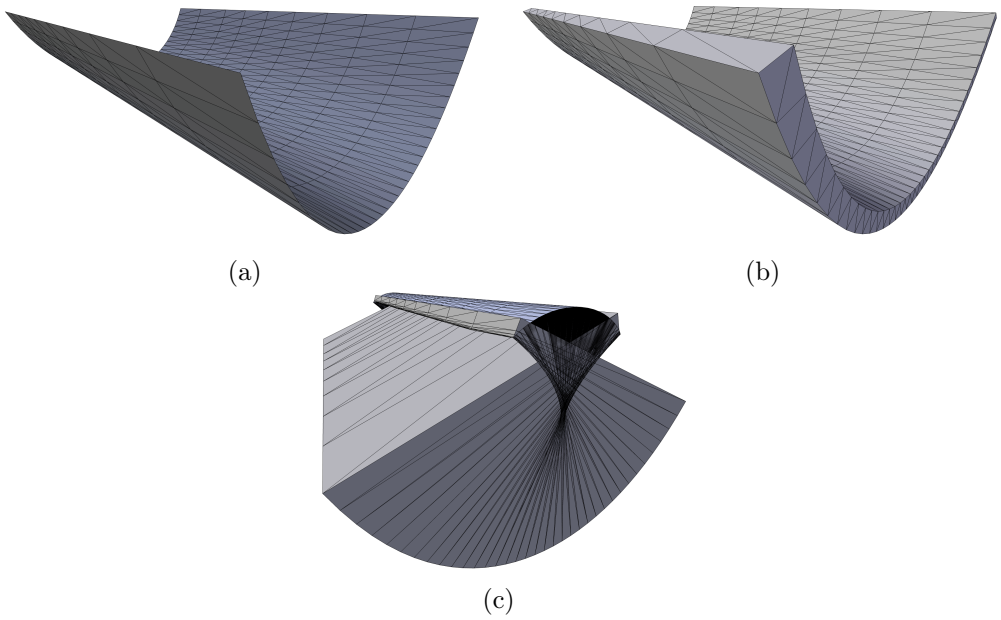


FIGURE 7.9 – (a) Surface fine à épaissir, (b) Surface sans bord après l'application de la méthode de (Qu et Stucker [2003]), (c) Surface avec une auto-intersection la rendant non orientable et avec des recouvrements.

Chapitre 8

Détection de la fragilité

8.1 Introduction

Pour qu'une représentation 3D d'un objet puisse être imprimée, il faut respecter deux types de contraintes. Les contraintes géométriques et topologiques afin de définir un volume intérieur et extérieur de l'objet. Les erreurs qui ne respectent pas ces contraintes sont identifiées et traitées dans les (Chap.4.5.6.7). Les contraintes physiques afin que l'objet imprimé ne soit pas endommagé par son propre poids ou son équilibre.

Les contraintes physiques regroupent diverses problématiques, telles que l'analyse des contraintes (Stava *et al.* [2012], Zhou *et al.* [2013], Lu *et al.* [2014]), la fabrication d'objets hétérogènes (Bickel *et al.* [2010]), l'équilibrage de modèles (Prévost *et al.* [2013]), l'impression de modèles articulés (Calì *et al.* [2012]), la partition en parties imprimables (Luo *et al.* [2012]).

Le but de ce chapitre est de présenter une méthode simple et pratique d'évaluation de l'imprimabilité 3D basée sur l'évaluation de l'épaisseur locale du maillage. Dans le but de prévenir d'une fragilité sur l'objet physique après impression. La méthode peut être appliquée sur un modèle 3D dont le volume n'est pas entièrement défini, car elle vérifie si localement le volume est défini. Dans le cas où il ne l'est pas, l'épaisseur locale n'a pas de valeur pour cet endroit. La méthode peut donc être appliquée sur des modèles non réparés, mais donne un résultat plus complet s' ils le sont. L'épaisseur critique de fragilité est définie par expérience et dépend de la matière et de la technologie d'impression 3D.

8.2 Travaux existants

Les méthodes d'estimation de l'épaisseur locale peuvent être classées en deux catégories, en surface ou en volume.

Selon l'approche basée sur la surface, l'épaisseur locale est définie pour

chaque point de la surface comme une mesure de la distance à la surface "opposée". (Jones *et al.* [2000]), par exemple, propose une mesure de l'épaisseur corticale du cerveau. L'épaisseur corticale d'un point donné du cortex est définie comme la longueur du trajet le long d'une ligne relie les deux bords opposés. L'épaisseur est définie de façon unique pour chaque point du cortex.

L'approche basée sur le volume peut être illustrée par la méthode de (Hildebrand et Rüegsegger [1997]) qui estime l'épaisseur locale en ajustant des sphères maximales à chaque point de la structure. L'épaisseur locale en un point donné est définie comme le diamètre de la sphère la plus grande qui contient le point et qui est complètement à l'intérieur de la structure. Il y a une perte de réciprocité et d'unicité de la mesure de l'épaisseur.

Les méthodes ci-dessus permettent d'estimer l'épaisseur locale indépendamment des applications en aval. Compte tenu de la technologie d'impression 3D utilisée, l'épaisseur locale peut être définie d'une manière plus dépendante de la technologie. Pour la fabrication en couches, par exemple (Alexander et Dutta [2000]), la sélection de l'orientation de construction du modèle résout certains problèmes de fragilité sans un épaississement local des parois de la surface.

8.3 Notre approche

Nous sommes intéressés par le calcul d'épaisseur locale sur un modèle d'entrée qui est composé de différentes surfaces avec ou sans bord, définissant des volumes, avec des encapsulations et des intersections possibles. On définit un point de la surface comme point frontière si en ce point la surface délimite le volume intérieur du volume extérieur. L'opposé d'un point frontière est le point frontière le plus proche, sur la demi-droite partant du premier point frontière avec une direction perpendiculaire à la surface en ce point. L'idée de base est d'utiliser un algorithme de lancer de rayon pour calculer l'épaisseur locale comme la distance entre un point frontière et son opposé. Cette distance est unique, mais pas réciproque.

Algorithme 12 Calcule de l'épaisseur locale par lancer de rayon

Entrées: F, faces du maillage

Sorties: Épaisseur$_f$, $\forall$ f, f $\in$ F, Épaisseur$_f$ mesure de l'épaisseur locale sur le géocentre de f

1: **pour chaque** f, f $\in$ F **faire**
2: Tracer un rayon r$_f$, à partir du géocentre de f avec une orientation opposée à la normale de f ;
3: **pour chaque** f', f' $\in$ F et f' $\neq$ f **faire**
4: Soit IntersecteRayon$_{f'}$(intersection$_{f'}$, distance$_{f'}$, orientation$_{f'}$) ;
5: **si** r$_f$ intersecte f' **alors**
6: intersection$_{f'}$ = vrai ;
7: distance$_{f'}$ = La distance entre le géocentre de f et l'intersection de r$_f$ et f' ;
8: **si** r$_f$ a la même orientation que la normale de f' **alors**
9: orientation$_{f'}$ = -1 ;
10: **sinon**
11: orientation$_{f'}$ = 1 ;
12: **fin si**
13: **sinon**
14: intersection$_{f'}$= faux ; distance$_{f'}$ = ∞ ; orientation$_{f'}$ = ∞ ;
15: **fin si**
16: **fin pour**
17: Épaisseur$_f$ = ÉpaisseurParenthèse(f,IntersecteRayon$_{f'}$) ;
18: **fin pour**

Soit F l'ensemble des faces du modèle d'entrée. Pour chaque face f $\in$ F, un rayon r$_f$ est tracé à partir du géocentre de f dans une direction égale à la normale de f, mais avec une orientation opposée. Les points d'intersection entre le rayon et les faces du modèle d'entrée sont calculés. À chaque point d'intersection est associée une parenthèse, ouvrante ou fermante, selon que le rayon "entre" ou "sort" de la surface, selon l'orientation de la normale de la face intersectée. Ainsi, le long du rayon, une expression de parenthèses est construite. Nous vérifions si les parenthèses sont cohérentes dans cette expression. L'épaisseur locale est définie comme la distance entre les points d'intersection correspondant à la première parenthèse ouvrante et à sa parenthèse fermante. Nous désignons l'ensemble du traitement ci-dessus comme un "Volume parenthésé". L'algorithme (Algo.12) itère sur chaque face du modèle d'entrée en deux étapes. D'abord, une phase de prétraitement pour calculer l'information nécessaire au volume parenthésé, et ensuite, grâce à l'algorithme (Algo.13), la construction et l'évaluation de l'expression du volume parenthésé.

Algorithme 13 Construction et l'évaluation du volume parenthésé (ÉpaisseurParenthèse)

Entrées: Face f, f *in* F ; IntersecteRayon$_{f'}$(intersection$_{f'}$, distance$_{f'}$, orientation$_{f'}$)
Sorties: Épaisseur, épaisseur locale sur le géocentre de f

```
1: Soit IntExt = 1 and FindExt = faux ;
2: Tri de IntersecteRayon[] par ordre croissant de distance_f' ;
3: pour chaque f' ∈ F faire
4:    si IntersecteRayon_f'.intersection_f' == vrai alors
5:       IntExt += IntersecteRayon_f'.orientation_f' ;
6:       si IntExt == 0 && FindExt == faux alors
7:          FindExt = vrai ;
8:          Épaisseur = IntersecteRayon_f'.distance_f' ;
9:       fin si
10:   fin si
11: fin pour
12: si IntExt ≠ 0 alors
13:    Épaisseur = null ;
14: fin si
```

La figure (Fig.8.1) illustre des cas en deux dimensions. Par exemple la figure (Fig.8.1,(a)) est une surface A sans bord. Le rayon r_f traverse A, en un seul point. Une parenthèse gauche est associée au géocentre de f, car r_f "entre" dans A, et une parenthèse droite est associée au point d'intersection, car r_f "sort" de A. La vérification de l'expression de parenthèses est simplement associée à un compteur IntExt, incrémenté lors du passage d'une parenthèse gauche, et décrémenté lorsqu'on passe une parenthèse droite. Au géocentre de f, IntExt est initialisé à un. L'épaisseur est mesurée lorsque IntExt est annulé pour la première fois. Le balayage des points d'intersection se poursuit sur tout le rayon r_f. Le calcul de l'épaisseur n'est valide que si, à la fin, IntExt est nulle. Une surface avec un vide est montrée à la figure (Fig.8.1,(b)).

Lorsque la face initiale appartient à une surface interne (Fig.8.1,(d)), ou si la surface est une surface avec bord (Fig.8.1,(e)), l'algorithme échoue. Les cas où les faces ne supportent pas un calcul valide, l'épaisseur locale n'a pas de valeur attribuée (null). L'échec du calcul indique la présence d'erreurs géométriques ou topologiques dans le modèle d'entrée qui doivent être corrigées avant de répéter le calcul.

8.4 Résultats

Tous les objets sont réalisés sur une imprimante "Z-Corp" avec une hauteur de 7 cm. Tout d'abord, nous illustrons un exemple d'évaluation de l'im-

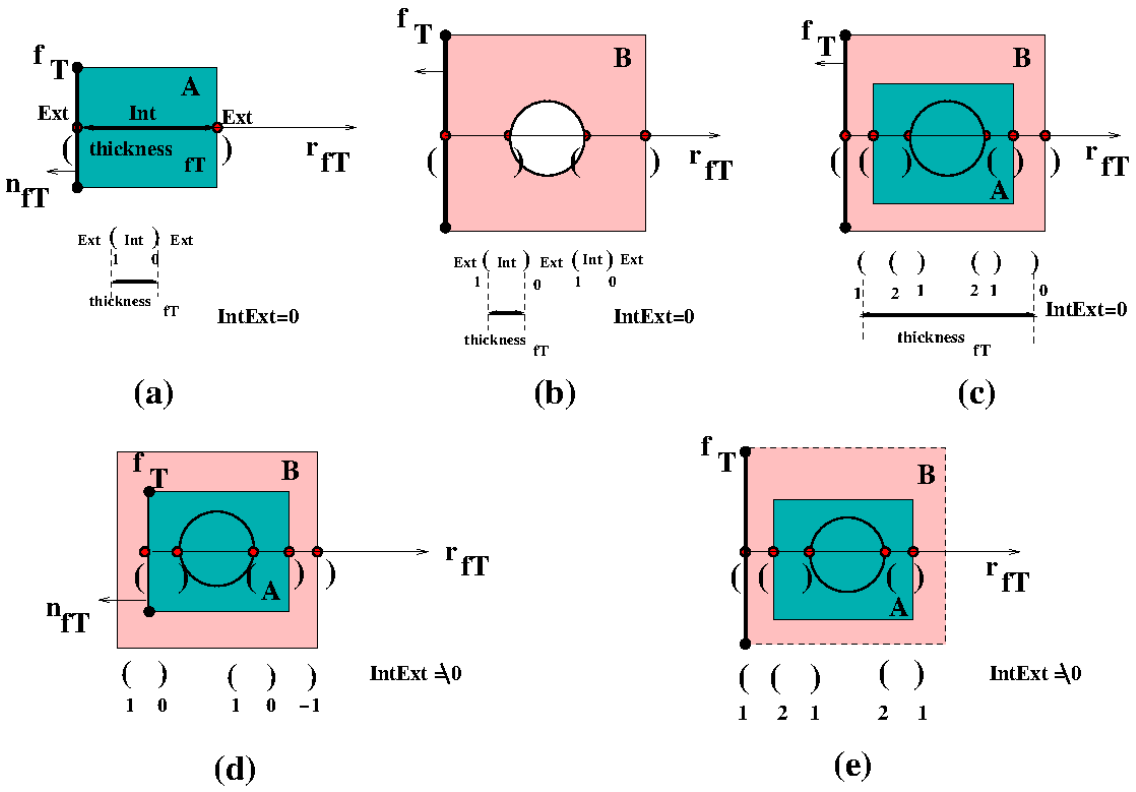


FIGURE 8.1 – (a)(b)(c) Évaluation d'épaisseur locale, (d) Évaluation à partir d'une face interne, (e) Évaluation à travers le trou d'une surface.

primabilité basé sur le modèle de "Kate". Le maillage d'entrée, illustré à la (Fig.8.2,(a)), est un modèle à plusieurs volumes avec des intersections et des surfaces planes. L'évaluation de l'épaisseur locale est donnée à la (Fig.8.2,(b)) après réparation géométrique et topologique, et l'impression 3D est présentée à la (Fig.8.2,(c)). On peut voir que les régions d'épaisseur critique, comme la main droite et la ceinture, sont détectées par l'algorithme de volume parenthésé. Dans le premier cas, le concepteur corrige l'épaisseur critique en fixant la main à la hanche, évitant ainsi la cassure de la main lors de l'impression. Dans le second cas, la zone critique reste inchangée et produit une fissure sur l'objet physique.

Dans la (Fig.8.3), nous donnons une évaluation de l'épaisseur par volume parenthésé sur des modèles de test couramment utilisés (Lu *et al.* [2014], Stava *et al.* [2012], Umetani et Schmidt [2013], Wang *et al.* [2013], Zhou *et al.* [2013]). Les résultats imprimés 3D sont présentés à la (Fig.8.4). Comme prévu, les bras et les jambes de "Banana man" se cassent à l'impression, comme l'indique leur épaisseur locale critique. Le support de "Soccer cup" est construit en six bandes. Malgré l'épaisseur critique de chaque bande, l'objet conserve son intégrité. En revanche pour "Hanging ball", la boule suspendue ne contient pas

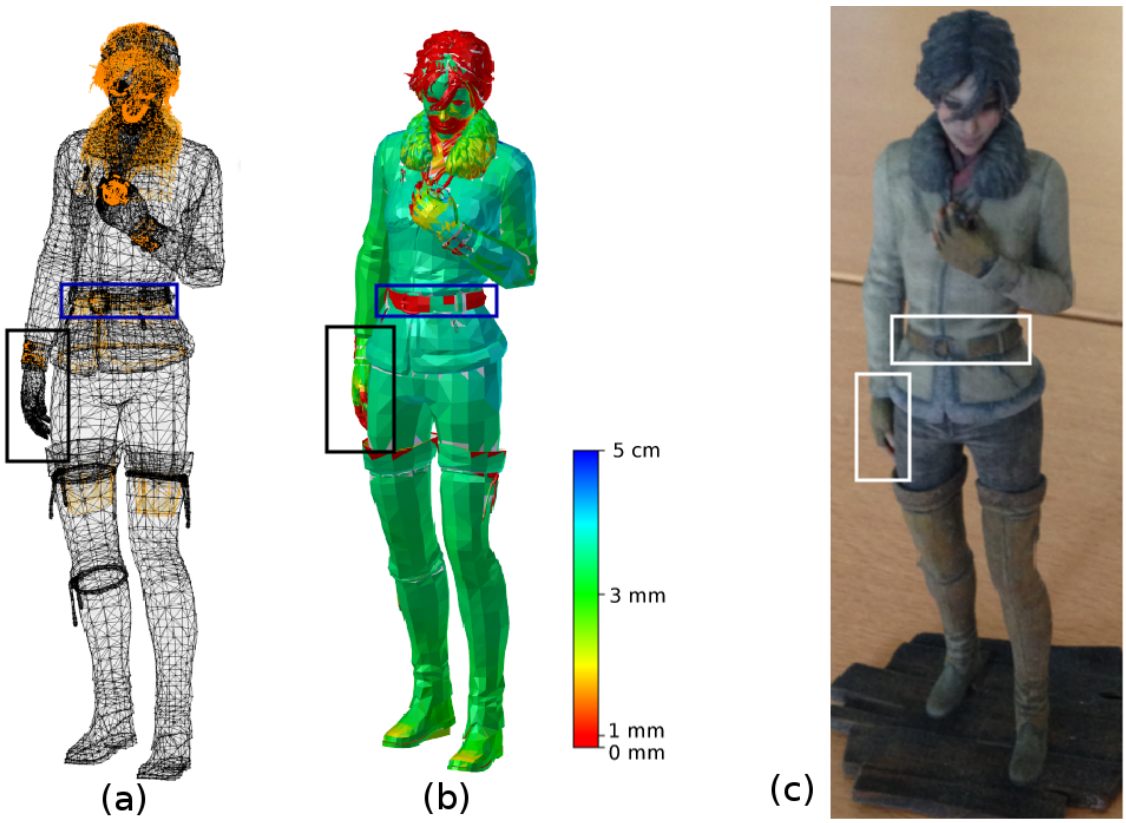


FIGURE 8.2 – (a) Modèle "Kate" (mailage), (b) Évaluation de l'épaisseur locale, (c) "Kate" imprimé.

de zones d'épaisseur critique, mais la gravité de la boule a coupé la connexion au socle. Enfin, "Shell" est d'épaisseur uniforme, tout comme l'objet imprimé qui reste en une seule pièce. Ces expériences montrent que le volume parenthésé permet d'évaluer correctement l'épaisseur locale. Notre méthode fournit une rétroaction simple et rapide sur l'imprimabilité des modèles, mais est moins précise qu'une analyse physique. Par exemple, en comparaison avec l'analyse des contraintes de Stava et al. pour le modèle "Shell" illustré à la (Fig.8.3,(e)), l'illustration tirée de (Stava *et al.* [2012]).

Même s'il y a une bonne probabilité que le prototype se brise dans les régions d'épaisseur critique, des contraintes de faiblesse physique supplémentaires sont nécessaires pour déterminer pleinement l'évaluation de l'imprimabilité.

8.5 Conclusion

L'évaluation de l'imprimabilité des modèles 3D est liée à la solidité physique et au contrôle des faiblesses. Dans ce chapitre, nous proposons une méthode

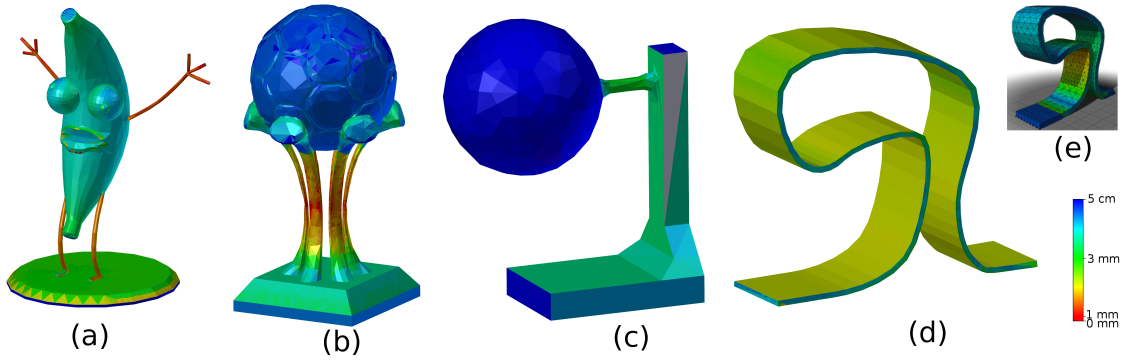


FIGURE 8.3 – Épaisseur évaluée volume parenthésé sur les modèles : (a) "Banana man", (b) "Soccer cup", (c) "Hanging ball", (d) "Shell".

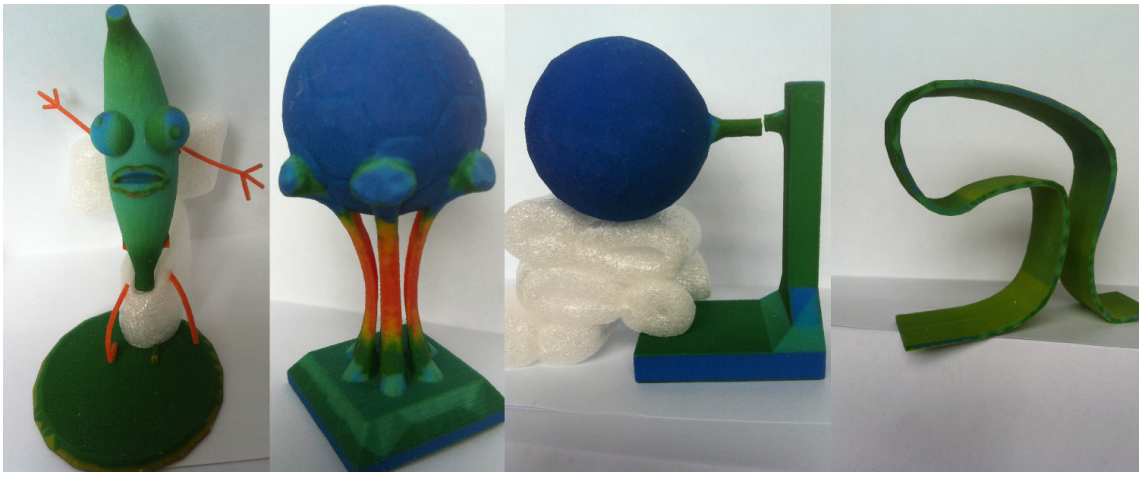


FIGURE 8.4 – Modèle de test imprimé sur une imprimante 3D "Z-Corp".

de détection des régions d'épaisseur critique basée sur le volume parenthésé. La méthode est utilisée pour le calcul de l'épaisseur locale sur une surface correctement orientée.

La méthode élaborée est mise en œuvre et expérimentée dans une chaîne industrielle d'impression 3D en tant qu'outil d'aide à la conception. Cette étape a l'avantage d'être rapide par rapport aux étapes de réparation géométrique et topologique. Ce qui lui permet d'être utilisée en amont par le client pour vérifier l'imprimabilité de l'objet qu'il souhaite imprimer.

Le retour d'expérience fourni par l'industrie implique une étude de cas plus approfondie des erreurs à réparer avant la fabrication du prototype et une meilleure compréhension des contraintes physiques. Notre objectif est de réduire autant que possible l'interaction lors de la réparation des maillages tout en conservant le concept initial du concepteur.

Troisième partie

Implémentation et résultats

Chapitre 9

Logiciel

Le logiciel permet d'effectuer les différentes étapes de la chaîne de traitement sur un modèle 3D. Il se présente sous la forme d'un exécutable en ligne de commande. Les fichiers contenant les modèles 3D d'entrées ou de sorties sont uniquement au format OBJ afin de pouvoir traiter la couleur. Le format STL pourrait être ajouté car la connectivité entre les faces non-présentes dans ce format est recalculée. Le code du logiciel peut aussi être utilisé comme une bibliothèque pour un autre logiciel, comme le plug-in de FabZat.

Sans option le logiciel a besoin de deux arguments, le chemin vers le fichier contenant le modèle 3D d'entrée et le chemin vers le fichier où écrire le modèle 3D de sortie. Les fichiers au format MTL ce trouve au même niveau que les fichier au format OBJ.

Usage : PathObjImport PathObjExport

La chaîne de réparation est appliquée sur le modèle 3D d'entrée avec les paramètres par défaut. Le modèle 3D réparé par la chaîne est écrit dans le fichier de sortie. L'étape de détection de fragilité n'est pas appliquée sans option.

Plusieurs options permettent d'adapter l'utilisation de la chaîne de réparation. Les options sont ajoutées en argument après les deux chemins. Toutes les options sont cumulables.

Usage : PathObjImport PathObjExport –option

— *–center*, cette option permet de recentrer l'objet sur l'origine (0,0,0). En effet rien n'empêche un modèle 3D d'être enregistré avec des valeurs de position de sommet très élevé positivement ou négativement. Mais ça rend l'utilisation du modèle plus complexe dans différents logiciels 3D.
— *–centerAndResizeY=*, cette option permet de paramétrer la taille d'impression qui est de 10cm par défaut. Cette taille est utilisée dans plusieurs étapes de la chaîne et peut changer la réparation. En plus de définir cette

taille, l'option recentre et redimensionne le modèle 3D en prenant comme unité le centimètre, car la taille du modèle 3D d'entrée n'est pas forcément à sa taille d'impression et de plus n'a pas d'indication d'unité.

— *–thicknessSize=*, cette option permet de paramétrer la valeur de l'épaisseur limite sous laquelle la modèle risque de casser à l'impression. La valeur est par défaut à 3mm. Cette valeur est utilisée pour plusieurs étapes de la chaîne, la classification des contours, la réparation par épaississement et la détection de faiblesse.

— *–noRepair*, cette option n'effectue que deux étapes de la chaîne de réparation l'Import et l'Export. Ce qui permet d'effectuer uniquement les repartions de non-variété et la répartition en composante connexe. La répartition en composante connexe est conservée dans le modèle 3D de sortie. Chaque composante connexe devient un "objet" du format OBJ.

— *–colorThickness*, cette option effectue la détection de fragilité en coloriant les faces du modèle en fonction de l'épaisseur à cet endroit du modèle 3D. Elle peut être effectuée après les étapes de réparation ou non avec l'option *–noRepair*. Les couleurs sont conservées dans le modèle 3D de sortie.

9.1 Diagrame de *Package*

La structure du logiciel de réparation est répartie en plusieurs *packages* (Fig.9.1) :

Application gèrent l'interface utilisateur avec la gestion des options pour la chaîne de réparations. Il implémente principalement la chaîne de réparation en utilisant les autres *packages*. Trois *packages* représentent les différentes étapes de cette chaîne de réparation.

Import qui s'occupe de toutes les étapes présentes dans le chapitre (Chap.4). Ces étapes sont l'import d'un modèle au format OBJ qui est d'abord enregistré dans une structure temporaire, la répartition en composante connexe, la correction des conditions de non-variétés et l'orientation des surfaces. La dernière étape effectuée par ce *package* est la construction du modèle dans une structure *Halfedge*.

Classification/Repair s'occupe des étapes de la chaîne qui traitent les arêtes de bord. Ces étapes sont la recherche et la classification des contours puis les étapes de réparations par remplissage et par offset. Il s'agit de l'application des chapitres Détection et classification des contours (Chap.5), Réparation par remplissage (Chap.6) et Réparation par offset (Chap.7).

Weakness s'occupe lui de l'étape de détection de la fragilité due au passage d'un modèle 3D virtuel à un objet physique. Il contient le test de fragilité décrit dans le chapitre Détection de la fragilité (Chap.8).

Surface_mesh est la bibliothèque utilisée pour la structure *Halfedge*. Une instance de cette structure est créée dans **Applications** puis transmise aux différentes étapes de la chaîne. Notamment à l'import qui construit la structure à partir des données de l'objet d'entrée et des premières réparations.

Octree, on utilise un arbre de type *octree* dans deux étapes de la chaîne pour des raisons différentes. Pour la classification des contours où l'on plonge les composantes connexes dans une grille représentée par un *octree* et dans le test de faiblesse qui utilise un *raytracing* dans un *octree* pour être plus rapide.

Tools, contient principalement des outils géométriques utilisés par tous les autres *packages*. Par exemple le produit vectoriel ou le calcul du déterminant d'un vecteur de longueur 3, mais aussi des tests d'intersection entre des éléments type face ou arête.

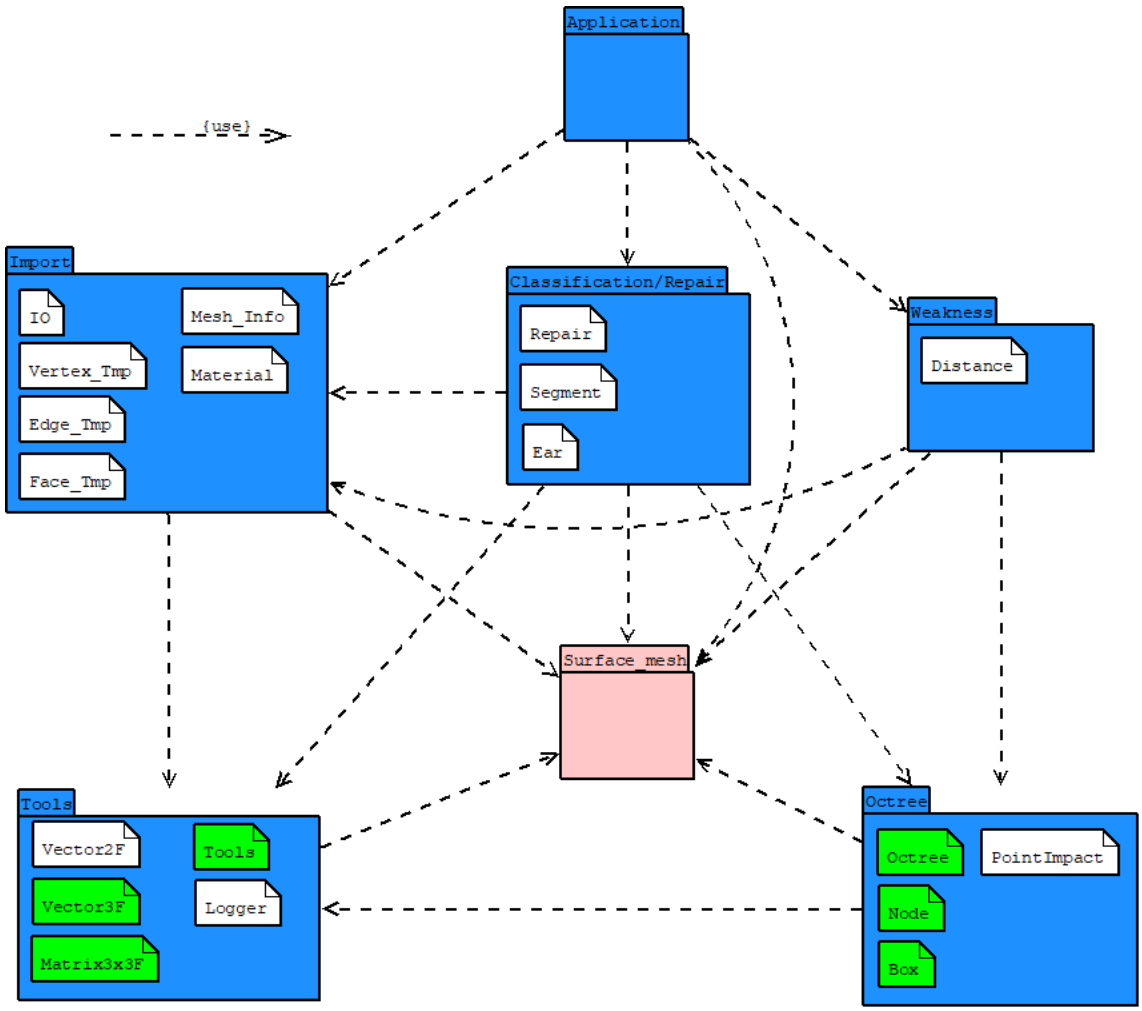


FIGURE 9.1 – Diagramme de *package* : Application (en Bleu), Code modifié (en Vert), Bibliothèque (en Rose).

9.2 Application

Pour réaliser la chaîne de réparation, le *package* **Application** crée des instances des classes des autres *packages* et utilise certaines de leurs fonctions. Trois instances sont utilisées tout au long de la chaîne. Une instance de la classe *Surface_ mesh* pour enregistrer le modèle 3D dans la structure *Halfedge*. Une instance de la classe *meshInfo* pour enregistrer des informations complémentaires du modèle 3D notamment les informations de couleur et texture. Et une instance de la classe *Logger* pour informer du déroulement de la réparation.

Les instances des classes *IO*, *Repair* et *Distance*, sont initialisées avec un pointeur sur les instances de *Surface_ mesh* et *meshInfo* afin que chaque étape ait les informations sur le modèle 3D. Chacune de ces instances est créée en fonction des options données au logiciel. On appelle ensuite des fonctions de ces instances pour réaliser la chaîne de réparation en tenant compte des options.

9.3 Import

Le *package* **Import** (Fig.9.2) contient une classe principale *IO* instanciée par le *package* **Application**. C'est en utilisant les fonctions de cette classe qu'**Application** vas exécuter certaines étapes de la chaîne :

— Les étapes du chapitre (Chap.4) qui sont, la répartition en composantes connexes, la réparation des conditions de non-variété, l'orientation des surfaces et la construction de la structure *Halfedge* représentant le modèle 3D (*read_ obj()*).

— L'étape final d'export du modèle 3D réparé dans un fichier au format OBJ (*write_ Obj()*).

La structure temporaire est utilisée pour les étapes de réparation du chapitre (Chap.4) avant la création de la structure *Halfedge*. Les éléments de cette structure sont définis par les classes *Vertex_ Tmp*, *Vertex_ Tmp* et *Face_ Tmp* pour la géométrie et la classe *Material* pour l'apparence (couleur/texture).

Une liste de chacune de ces classes est instanciée par la classe principale *IO* pour enregistrer le modèle 3D à partir des informations importées du modèle 3D d'entrée au format OBJ. Les premières réparations modifient ces listes pour enregistrer les changement effectués sur le modèle 3D.

Les instances des classes *Surface_ mesh* et *meshInfo*, créées par le *package* **Application**, contiennent les informations du modèle 3D utilisées par toutes les étapes de la chaîne sont transmises vides à la fonction (*read_ obj()*) de la classe *IO*. À partir des informations du modèle 3D dans la structure temporaire et après les premières étapes de réparation, cette fonction construit dans l'instance *Surface_ mesh* la structure *Halfedge* représentant le modèle 3D. Elle

transmet aussi les informations d'apparence à l'instance de *meshInfo* ainsi que des premières informations auxiliaires sur le modèle 3D comme le nombre de composantes connexes.

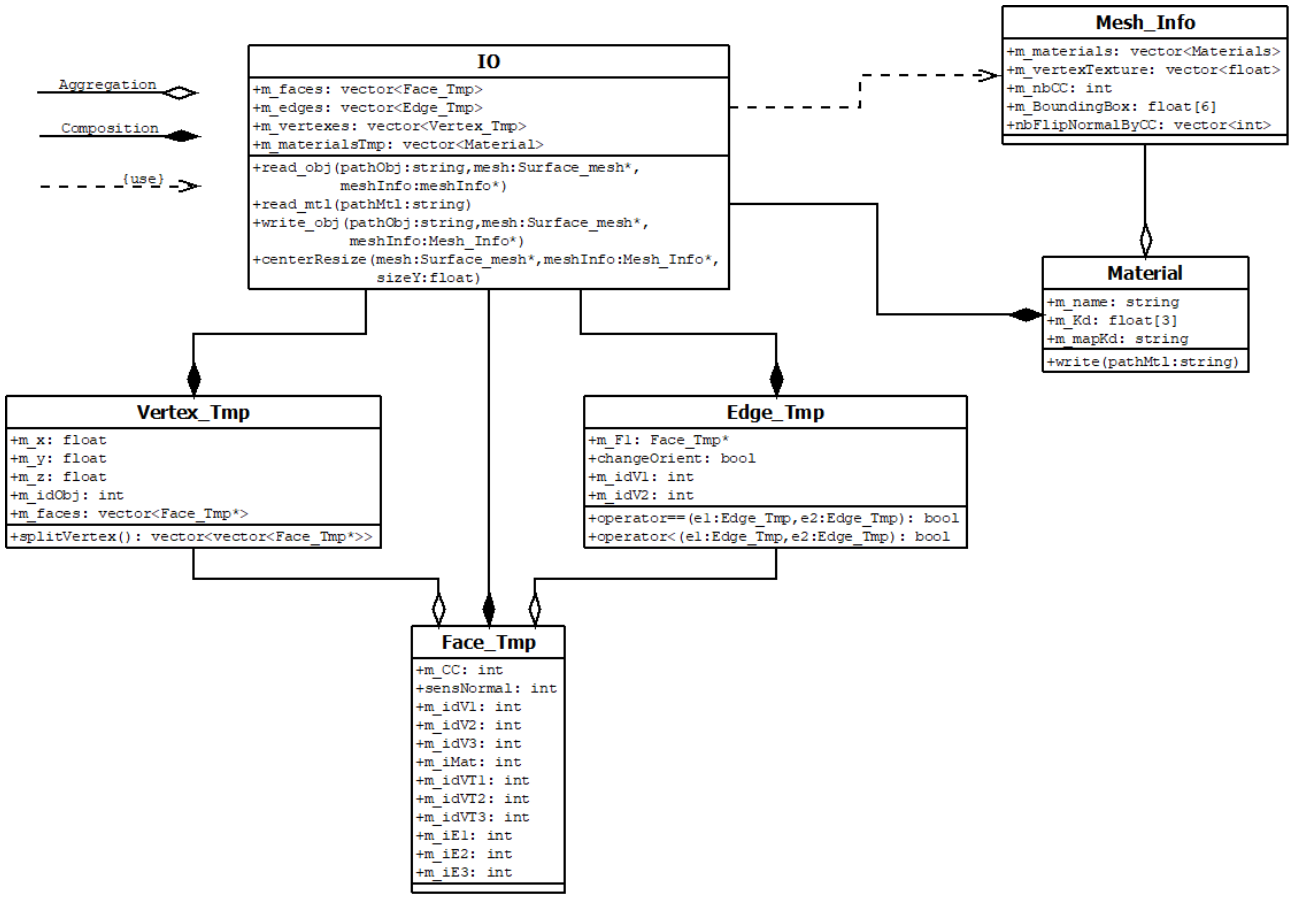


FIGURE 9.2 – Package Import.

9.4 Surface_mesh

Surface_mesh est une bibliothèque qui implémente une structure de modèle 3D de type *Halfedge*. Une instance de la structure est créée par le *package* **Application** qui s'occupera de la transmettre aux différentes étapes de la chaîne de réparation. Le *package* **Import** s'occupe lui de remplir la structure avec les informations du modèle 3D présent dans le fichier OBJ.

Certaines informations du modèle 3D, principalement sur l'apparence ne peuvent être enregistrées par la classe *meshInfo* mais doivent être enregistrées sur les faces du modèles 3D. *Surface_mesh* permet d'enregistrer des informations sur les faces avec son système de *face_Property*. On ajoute lors de

l'import les propriétés suivantes sur les faces.

— *f :vertexTexture*, pour les identifiants des textures des 3 sommets de la face.

— *f :iMat*, identifiant du matériau utilisé par la face.

— *f :idCC*, identifiant de la composante connexe de la face.

— *f :origin*, booléen indiquant si la face provient du modèle 3D original ou si c'est une face ajoutée pour la chaîne réparation.

La bibliothèque n'est pas complètement stable à la suppression d'élément, à la suite de telles opérations certaines fonctions ne retournent plus de bonnes valeurs comme le nombre de faces *n_faces()*. Ce type d'opération n'est effectué que par le *package* **Classification/Repair** qui fait attention à ne pas utiliser des fonctions instables.

9.5 Classification/Repair

Le *package* **Classification/Repair** (Fig.9.3) contient une classe principale *Repair* instanciée par le *package* **Application**. **Application** exécute certaines étapes de la chaîne en utilisant les fonctions de cette classe. **Application** transmet aux fonctions réalisant ces étapes les instances de *Surface_mesh* et *meshInfo* contenant le modèle 3D et ses informations. Les étapes réalisées par **Classification/Repair** sont :

— La recherche des contours (Algo.7)(*findOutline()*).

— La classification des contours (Algo.8)(*detectSurfacePlane()*).

— La réparation par remplissage avec l'étape de recherche de segments planaires (Algo.9)(*holeFilling()*) et l'application du *EarClipping* 3D (Section.6.2)(*repairEarClipping()*).

— La réparation par épaississement (Algo.11)(*offset()*).

La classification des contours utilise un *octree* dans son algorithme (Algo.8). Le *package* **Octree** est utilisé pour le créer et l'utiliser au cours de l'algorithme.

La réparation par remplissage construit, pour l'étape de recherche de segments planaires, une liste de segments qui sont des instances de la classe *Segment*. Cette liste est utilisée pour l' étapes de fermeture des segments planaires et pour appliquer le *EarClipping* 3D sur chacun des segments fermés.

Pour appliquer le *EarClipping* 3D, une liste des oreilles du segments fermés est construite. Les oreilles sont des instances de la classe *Ear*. Cette liste est utilisée par l'algorithme de *EarClipping* 3D notamment pour trier les oreilles par ordre croissant de leur angle saillant (*m_cosAngle*).

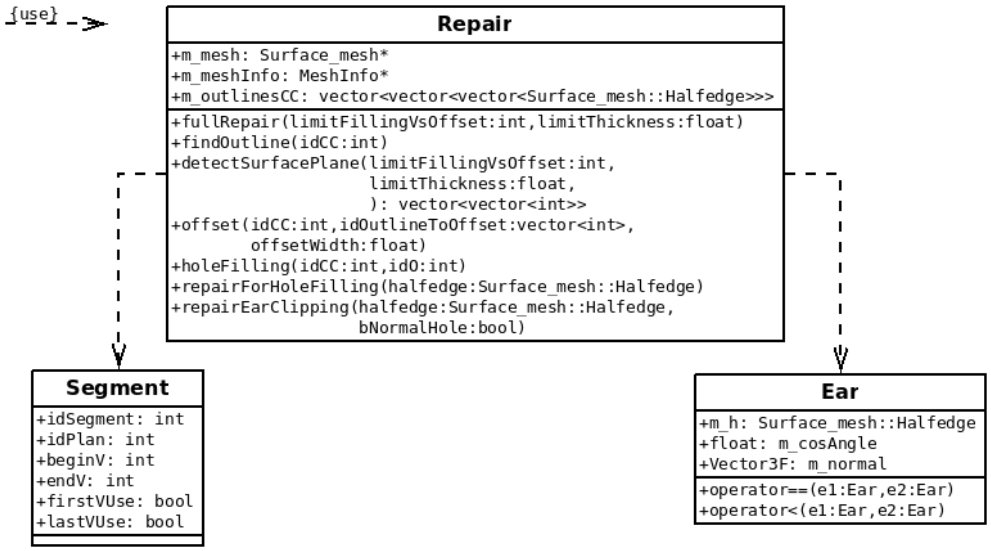


FIGURE 9.3 – Package Classification/Repair.

9.6 Weakness

Le *package* **Weakness** (Fig.9.4) contient la classe *Distance* instanciée par le *package* **Application**. La classe *Distance* contient les fonctions qui permettent de calculer les distances entre des points de la surface pour détecter et visualiser les zones de fragilité du modèle 3D.

Application exécute l'étape de détection de fragilité (Algo.12) en utilisant les fonctions de cette classe (*calculateDistanceWithOctree()*). **Application** transmet aux fonctions réalisant cette étape les instances de *Surface_mesh* et *meshInfo* contenant le modèle 3D et ses informations.

Le calcul de distance fonctionne sur le principe du lancé de rayons et utilise un *octree* pour l'optimiser. Le *package* **Octree** est utilisé pour créer l'*octree* et l'utiliser au cours du calcul.

On utilise pour la visualisation une palette de couleur type Hot Iron qui va du rouge au vert puis du vert au bleu, le rouge pour les parties fragiles qui risquent de casser à l'impression, le vert pour la distance limite de casse théorique donnée en paramètre *limitDistance* et le bleu un maximum pour les parties sans risque. Une échelle standard pour l'imprimante à poudre de FabZat est rouge de 0cm à 0.15cm, vert à 0.3cm et Bleu à 3cm. Le modèle colorié peut être visualisé après une exportation au format OBJ ou par un visualisateur interne comme dans le plug-in FabZat (Fig.10.22). Ces fonctions prennent en entrée une taille correspondant à la hauteur pour laquelle on veut tester la fragilité. Comme vu précédemment les informations spatiales du modèle 3D ne sont pas forcément à l'échelle de l'impression.

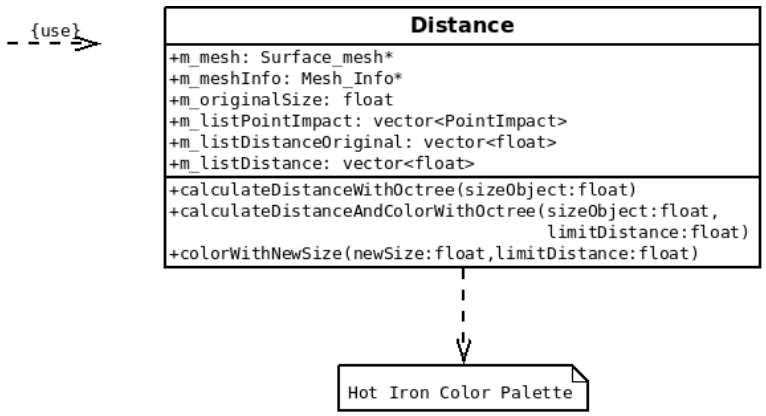


FIGURE 9.4 – Package Weakness.

9.7 Octree

Le *package* **Octree** (Fig.9.5) contient une classe principale *Octree* qui permet de plonger le modèle 3D dans une grille représentée par un *octree* afin d'exécuter certaines fonctions. La classe est instanciée à deux reprises dans la chaîne de réparation.

Le *package* **Classification/Repair** réalise l'étape de classification des contours en créant une instance de la classe **Octree** pour l'algorithme (Algo.8) (*detectSurfacePlane()*).

Le *package* **Weakness** réalise l'étape de détection de la fragilité en créant une instance de la classe **Octree** pour optimiser le lancer de rayon de l'algorithme du calcul de l'épaisseur locale (Algo.12)(*intersectOctree()*).

La classe **Octree** construit l'arbre de type *octree* avec une liste d'instance de la classe **Node**. La classe **Node** définit les nœuds de l'*octree* et la liste enregistre l'ensemble des nœuds de l'*octree*. Chaque nœud de l'*octree* définit un volume cubique de l'espace, la classe **Node** possède une instance de la classe *Box* pour définir et enregistrer ce volume. Le volume du nœud à la racine de l'*octree* est défini en fonction du modèle 3D. Le volume des autres nœuds correspond à un huitième du volume de leur nœud père.

La classe **PointImpact** sert à enregistrer un point d'intersection entre un rayon est une face. Une liste d'instance de **PointImpact** est construite à chaque lancé de rayon de l'algorithme du calcul de l'épaisseur locale (Algo.12). La liste est triée par ordre croissant de la distance entre le point d'intersection et l'origine du rayon (*m_distance*), dans l'algorithme du volume parenthésé (Algo.13).

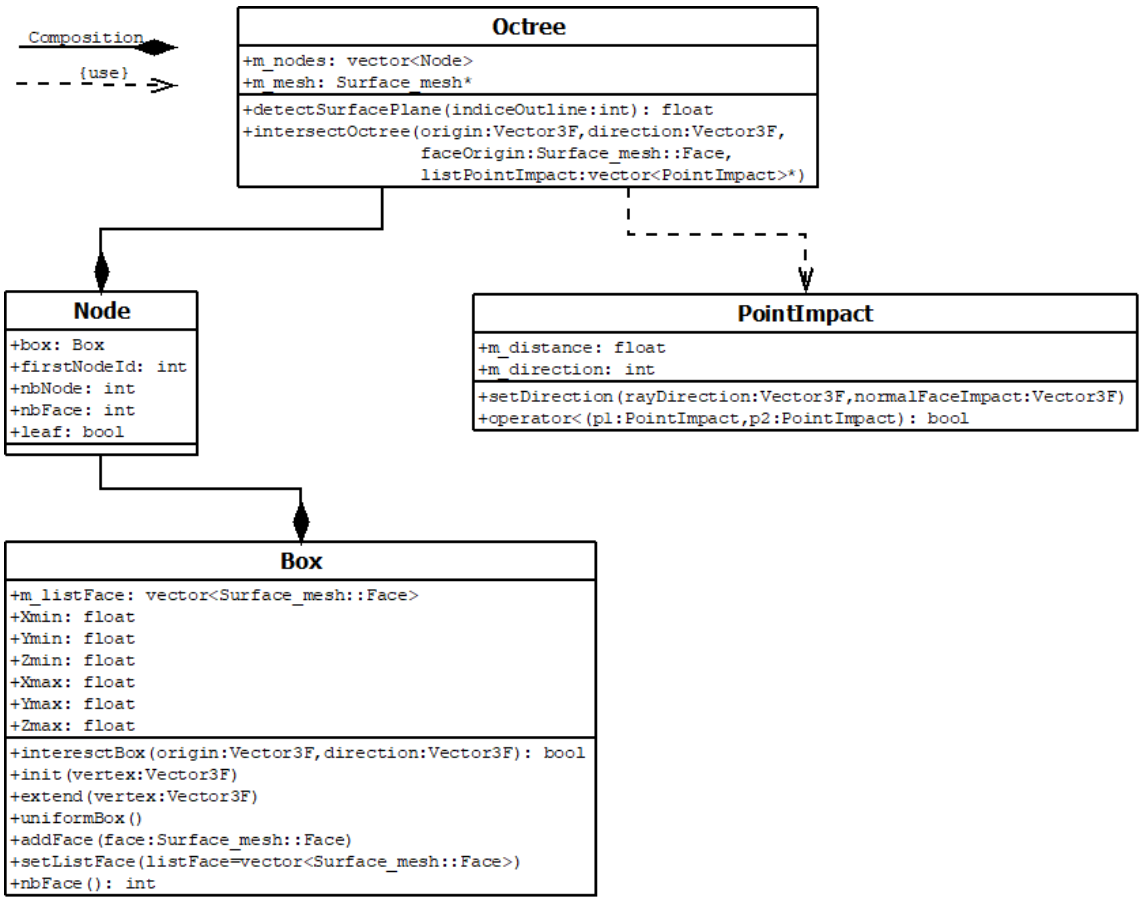


FIGURE 9.5 – Package Octree.

9.8 Tools

Le *package* **Tools** (Fig.9.6) contient différentes classes et fonctions utilisables par les autres *packages*. Les classes *Vector2F*, *Vector3F* et *Matrix3x3F* représentent des éléments mathématiques. La classe *Tools* propose des fonctions de test et de calcul sur des éléments 3D. La classe *Logger* permet l'écriture dans un fichier, des informations et des erreurs rencontrées par le logiciel en cours d'exécution.

La classe *Vector2F* représente un vecteur à deux dimensions. Il est principalement utilisé pour la sauvegarde et l'utilisation des *vertex texture* qui ont des coordonnées à deux dimensions sur la texture.

La classe *Vector3F* représente un vecteur à trois dimensions. Il est utilisé pour les sommets et comme vecteur 3D pouvant correspondre à une arête, une normale ou la direction d'un rayon. La classe comprend un ensemble de fonctions pour la manipulation de vecteur 3D.

La classe *Matrix3x3F* représente une matrice trois par trois. La classe comprend un ensemble de fonctions pour la manipulation d'une matrice trois par trois. Ces fonctions sont utilisées principalement pour le calcul d'intersection entre un rayon et une face du calcul de distance.

La classe *Tools* comprend des fonctions de test d'intersection, de test de colinéarité, de test de coplanarité sur des éléments 3D et des calculs de projections avec des éléments 3D. Les éléments 3D sont ceux de la bibliothèque *Surface_ mesh*. Le *package* **Classification/Repair** utilise ces fonctions dans l'étape de la réparation par remplissage.

La classe *Logger* est un singleton utilisé par les autres *packages* pour transmettre des messages d'erreur ou d'information vers un fichier. Les messages sont écrits en cours d'utilisation du logiciel. Le singleton est initialisé par le *package* **Application** avant le début de la chaîne de réparation.

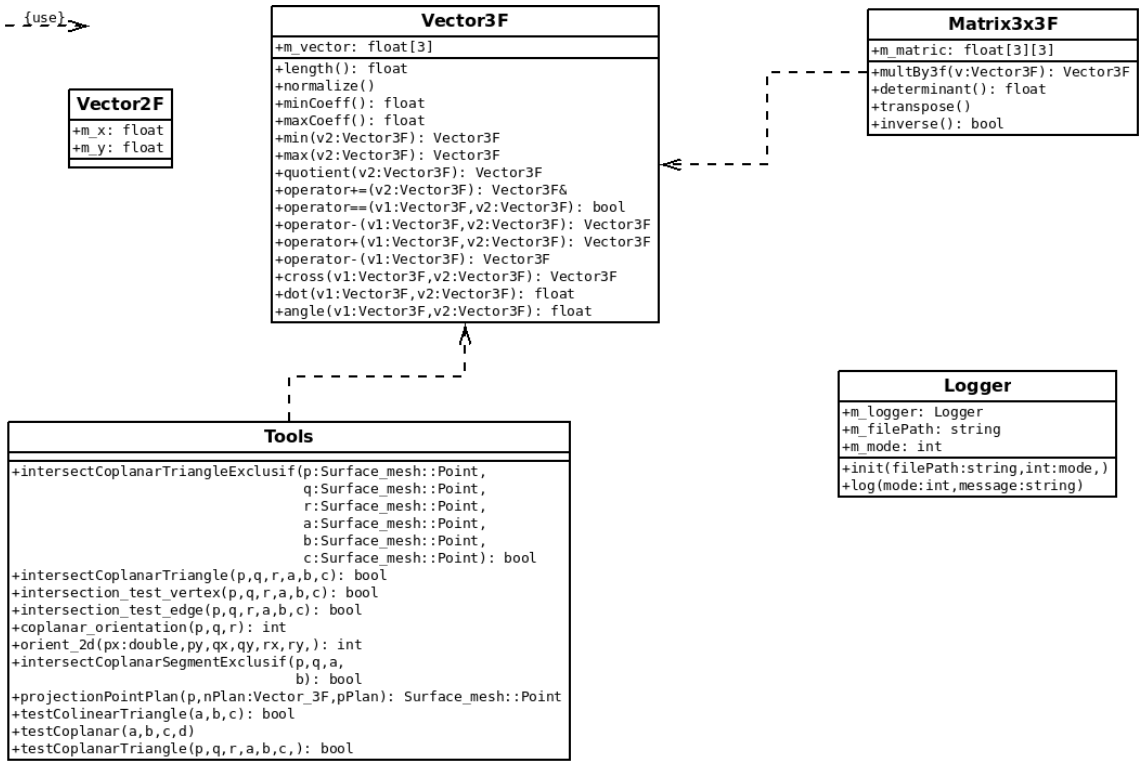


FIGURE 9.6 – Package Tools.

Chapitre 10

Résultats

10.1 Réparation

10.1.1 Réparations des conditions de non-variété

Pour les deux premiers modèles (Fig.10.1) on teste la réparation des non-variétés sur un sommet. Pour rappel cette réparation est effectué dans la partie Import. Le modèle "Vertex2CC" (Fig.10.1,(a)) est réparé par la répartition des faces en composantes connexes (Algo.1) Puis par la duplication des sommets qui appartiennent à plusieurs de ces composantes connexes. Sur l'image de droite on a légèrement déplacé les composantes connexes pour bien voir qu'après réparation il existe un sommet différent pour chacune des deux composantes. Le modèle "Vertex1CC" (Fig.10.1,(b)) est lui réparé par l'algorithme de cycle (Algo.5) car le sommet n'appartient qu'à une seule composante connexe. Sur l'image de droite le sommet a légèrement été déplacé pour bien voir qu'après réparation il existe deux sommets différents.

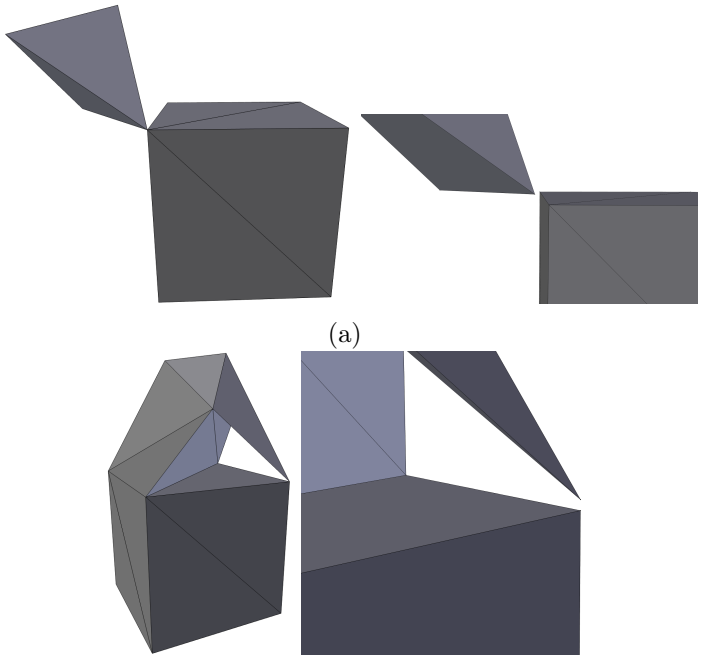
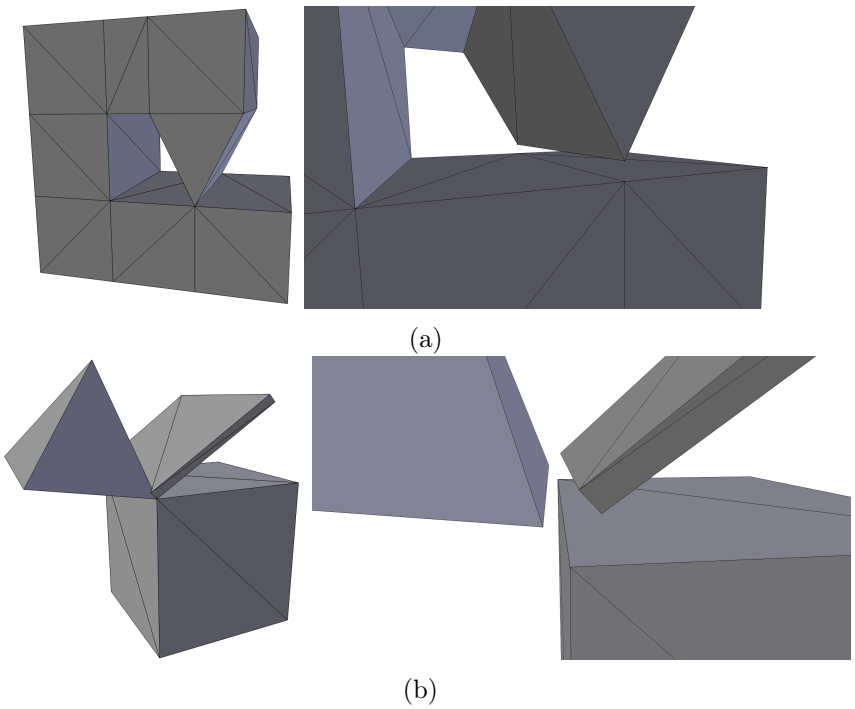


FIGURE 10.1 – Réparation d'une non variété sur un sommet : (a) Entre deux composante connexes, (b) Sur une même composante connexe.

Pour les deux modèles de la (Fig.10.2) on teste la réparation des non-variétés sur une arête. Le modèle "Edge1CC" (Fig.10.2,(a)) est réparé exactement de la même façon que "Vertex1CC". L'algorithme de cycle est simplement appliqué sur les deux sommets de l'arrêtes non-variété. La duplication de ces deux sommets et l'attribution des faces à chacun des deux va naturellement dupliquer l'arête non-variété, car les arêtes sont créées par les faces qui elles-mêmes utilisent les nouveaux sommets. Sur l'image de droite l'arête a légèrement été déplacée pour bien voir qu'après réparation il existe deux arêtes différentes. "EdgeMultiCC" (Fig.10.2,(b)) est lui réparé en deux temps, d'abord les non-variétés sont corrigées de la même façon que pour "Vertex2CC" avec la duplication des sommets appartenant à différentes composantes connexes. De la même façon que pour "Edge1CC" la duplication des deux sommets extrémités de l'arête non-variété duplique aussi l'arête. Sur cet exemple il y a trois composantes connexes, les sommets et l'arête sont donc dupliqués en trois exemplaires. La composante connexe composée de deux faces plates possède des arêtes de bords qui forment un contour de longueur quatre. La réparation automatique va donc trouver ce contour (Algo.7), puis l'analyser pour décider de la réparation à appliquer (Algo.8), et appliquer cette réparation. Dans ce cas, ça sera un épaississement puisque la composante connexe est parfaitement plane. Sur l'image de droite on a légèrement déplacé les composantes connexes pour mieux voir les réparations effectuées.



(a)

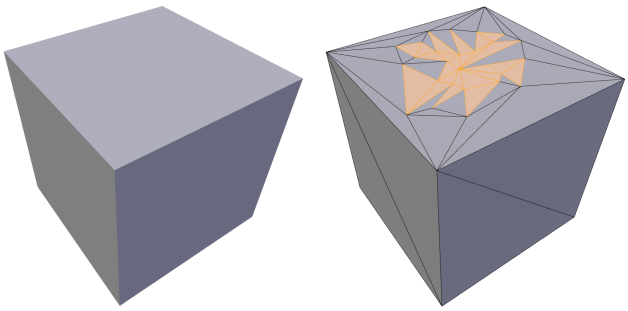
(b)

FIGURE 10.2 – Réparation d'une non-variété sur une arête : (a) Sur une même composante connexe, (b) Sur plusieurs composantes connexes plus un épaississement.

En déplaçant légèrement les sommets dédoublés, on voit quel groupe de faces leurs sont attachés après réparation. Pour les quatre modèles de test il n'y a plus de condition de non variété après réparation. Toutes les composantes connexes définissent un volume et sont imprimable.

10.1.2 Réparations de trous par remplissage

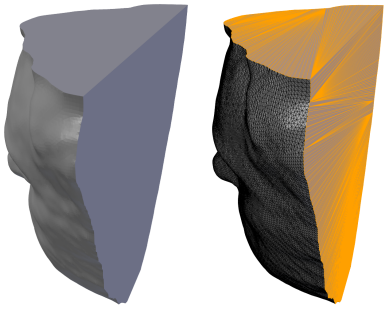
"CubeHole" (Fig.10.3) est un modèle avec comme seul problème des arêtes de bords qui forme un seul contour et toutes ces arêtes sont sur un même plan. Le but est de tester la partie remplissage par EarClipping de la réparation automatique. Mais toutes les autres étapes sont quand même effectuées. Ainsi la classification de contours fait bien le choix du remplissage et ce dernier ne détecte qu'un seul plan sur le contour. Le EarClipping fonctionne parfaitement et ne produit aucune auto-intersection ni recouvrement.



(a) (b)

FIGURE 10.3 – Réparation d'un contour sur un seul plan, (b) Faces rajoutées (en Jaune).

"FlorentHead" (Fig.10.4) est un modèle industriel issu d'une application de scanner de visage utilisé par FabZat. De la même façon que cubeHole son seul problème sont des arêtes de bords qui forment un seul contour, mais cette fois le contour se dessine sur deux plans. Le modèle sert donc à tester la partie découpe en parties planaires d'un contour de l'algorithme de remplissage . Lors de cette dernière une arête est ajoutée entre les deux sommets qui forme un coin de chaque côté de la tête puis on applique le remplissage à proprement parlé sur les deux parties du contour ainsi créé.

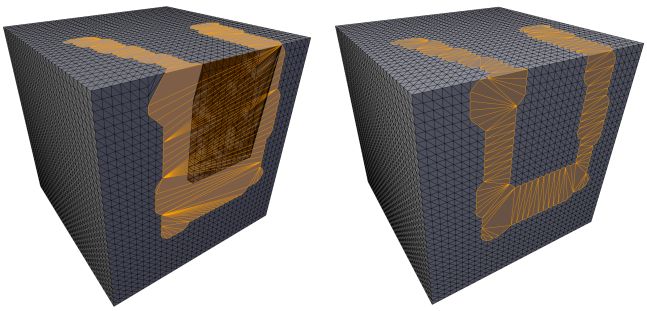


(a) (b)

FIGURE 10.4 – Réparation d'un contour sur deux plans, (b) Faces rajoutées (en Jaune).

"CubeU" (Fig.10.5) est un modèle synthétique qui teste le même problème que FlorentHead mais en plus complexe. La complexité ne vient pas du nombre de plans, il y en a toujours deux, mais lors de la découpe de contour on va trouver quatre segments planaires sauf que deux d'entre eux (le bas et le haut du U sur la face avant) ont besoin de faire partie du même contour au moment du remplissage. On peut remarquer qu'ils sont bien sur un même plan, mais ne se suivent pas directement, autrement il s'agirait d'une même partie planaire. Mais on peut remarquer que les deux contours de la face du dessus sont aussi sur un même plan et pourtant peuvent et doivent être réparés comme deux

parties de contour différentes. L'idée ici est de réparer dans un premier temps les parties qui ne posent pas de problèmes pour voir si elles modifient le contour et facilitent les parties plus compliquées. Dans cet exemple réparer en premier les deux contours du dessus permet lors d'une deuxième boucle de l'algorithme que les deux parties du U n'en soient plus qu'une.



(a) (b)

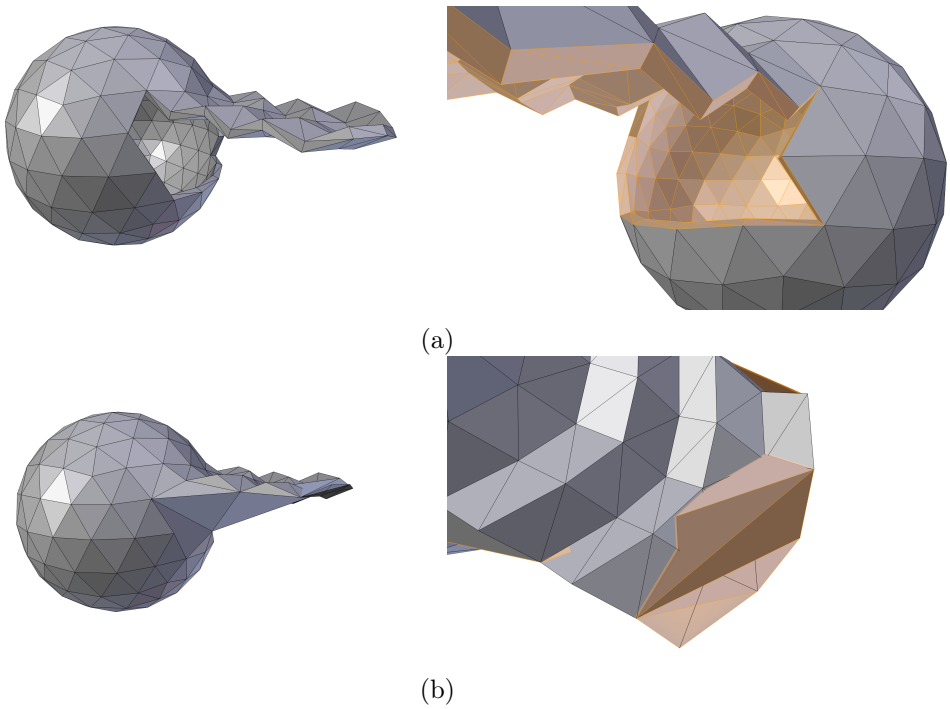
FIGURE 10.5 – Réparation d'un contour sur deux plans, cas complexe : (a) Réparation automatique, (b) Réparation semi-automatique, Faces rajoutées (en Jaune).

Pour les deux premier modèle "CubeHole" et "FlorentHead" la réparation automatique et plus particulièrement le remplissage à permis d'enlever toutes les arêtes de bords sans rajouter de nouvelle erreur. Pour "FlorentHead" le remplissage c'est fait sur deux plans séparés par une arête tranchante comme attendu. Pour le "CubeU" il a fallu ajouter manuellement une priorité dans la fermeture et recherche de contour planaire. Après ce changement la réparation enlève toutes les arêtes de bords et retrouve l'arête tranchante du cube. Les trois modèles définissent un volume et sont imprimable.

10.1.3 Classification des contours

"Casquette" (Fig.10.6) est un modèle pour tester la classification des contours (Algo.8). En effet une partie de son contour semble nécessiter un remplissage alors qu'une autre aurait besoin d'être réparée par épaississement. Le résultat de la classification est proche du seuil légèrement à l'avantage de l'épaississement. Les images (Fig.10.6,(a)) montrent le résultat de la réparation automatique avec donc le choix de l'épaississement, la réparation est parfaitement imprimable. Mais on peut facilement imaginer que la casquette fasse partie d'un modèle plus grand avec d'autres composantes connexes intersectées. Dans ce cas la cavité formée par l'épaississement sur la partie sphérique risque d'enfermer de la poudre non agglomérée. Les images (Fig.10.6,(b)) montrent le résultat de la réparation par remplissage en ayant forcé ce choix au logiciel. Sur la partie fine de la casquette, la réparation par remplissage provoque

comme prévu plusieurs problèmes, des auto-intersections qui empêchent la bonne orientation de certaines faces et des recouvrements qui provoquent des zones sans volume.



(a)

(b)

FIGURE 10.6 – Réparation d'un contour : (a) Réparation par épaississement (Choix automatique), (b) Réparation par remplissage (Choix manuel), Faces rajoutées (en Jaune).

La réparation automatique pour le modèle "Casquette" corrige les arêtes bords en effectuant un épaississement. Le résultat défini correctement un volume et peux être imprimé.

Mais le creux dans la partie sphérique peux enfermer de la poudre difficile à enlever. De plus le remplissage était proche d'être choisie et lui n'aurais pas était imprimable. Une meilleure réparation serait de découper la composante connexe en deux pour appliquer une réparation différente sur chacune des parties (Fig.10.7).

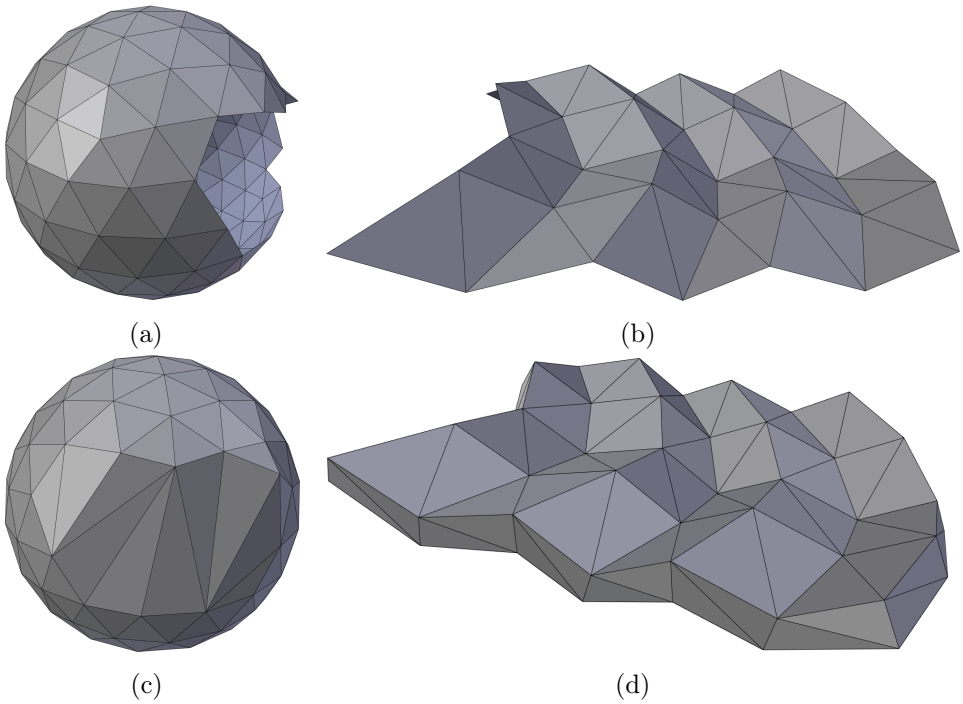


FIGURE 10.7 – Division d'un contour complexe en deux parties : (a) Partie où le contour borde un trou, (b) Partie où le contour entoure une surface plane, (c) Réparation par remplissage, (d) Réparation par épaississement.

10.1.4 Kate : Réparations de modèle industriel

Le modèle "Kate" est issu du jeu vidéo "Syberia III". C'est un modèle formé de multiple volumes, après la répartition en composantes connexes on en compte 294. Pour ce test on vas appliquer une réparation sur le modèle pour une impression de 10cm de hauteur. Le modèle comporte des surfaces non-variétés, des surfaces mal orientées, des trous dans la surface et des surfaces fines.

Les trous dans la surfaces se situent par exemple sur la surface représentant les jambes ou des plus petits trous comme sur la ceinture (Fig.10.10,(a)). Les surfaces fines se situent principalement sur les cheveux et l'écharpe (Fig.10.11,(a)) (Fig.10.13,(a)).

Le modèle de "Kate" est entièrement texturé et toutes les réparations présentées le sont également. Les images de ces réparations sont montrées avec ou sans texture uniquement en fonction d'une meilleure lisibilité de l'image.

Le résultat de la réparation automatique avec tous les paramètres par défaut est présenté dans la figure (Fig.10.8). Le résultat n'est pas bon pour deux raisons. Structurellement si il ne reste plus d'arêtes de bord, des surfaces ou des parties de surface auto-intersectées sont mal orientées et définissent des

vides. Visuellement de nouveaux volumes non présents sur le modèle original ont été ajoutés et déforment l'apparence du modèle.

FIGURE 10.8 – Réparation automatique de "Kate" : (a)(b) Réparation automatique, (c)(d) Zoom sur la tête.

La réparation a permis en revanche de répartir les faces en composantes connexes et de lever les conditions de non-variété. Les composantes connexes vont permettre d'améliorer cette réparation en modifiant pour chacune les choix de l'algorithme. Toujours dans l'objectif que si chaque composante connexe définit un volume alors le modèle entier aussi.

Les choix de l'algorithme qui vont être modifiés sont les suivants :

— L'orientation de la surface : c'est un choix binaire, si le volume définis un vide on inverse l'orientation de la surface.

— La classification des contours : c'est un choix binaire, si visuellement un contour semble avoir été mal classé ou si la réparation appliquée ne donne pas un bon résultat, on change sa classification. Attention choisir qu'un contour entoure une surface fine impacte par sa réparation par épaississement toute la composante connexe.

— Épaisseur de l'épaississement : c'est un paramètre qui définit l'épaisseur appliquée par l'algorithme d'épaississement. La valeur par défaut est l'épaisseur minimale sans risque de fragilité qui dépend de la technologie d'impression (0.3cm sur la *Zcorp 650*).

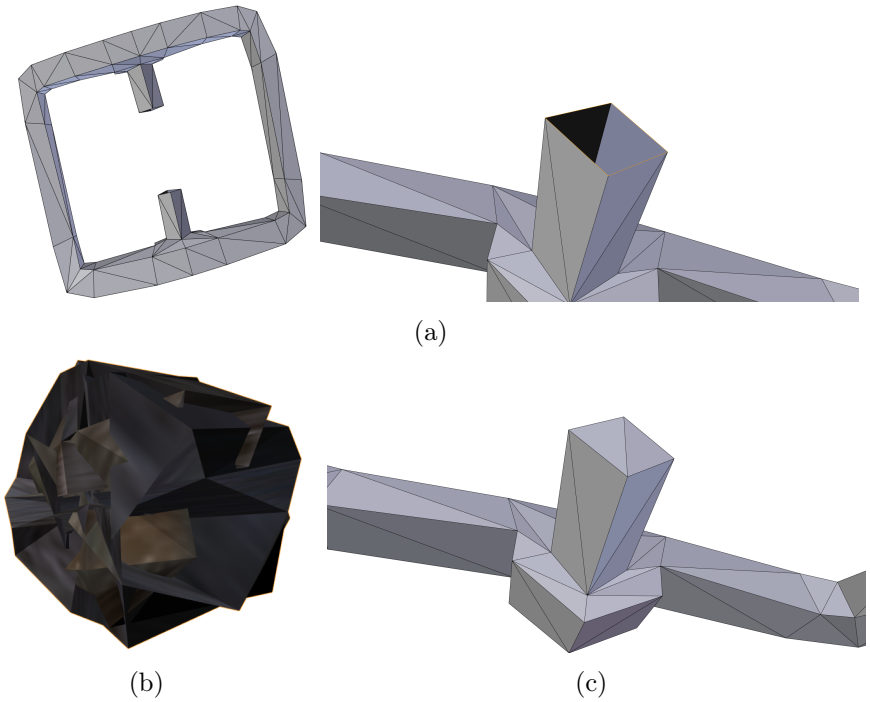
— Marge de planéarité d'un segment : c'est un paramètre qui définit l'écart autorisé entre la normale locale d'un sommet du segment et la normale moyenne du segment.

Classification de contour

La composante connexe "Boucle" possède avant réparation deux contours qui visuellement entourent un trou (Fig.10.9,(a)). La réparation automatique va pourtant les classer comme entourant une surface fine et appliquer un épaississement (Fig.10.9,(b)).

L'erreur provient de la taille de la boucle qui est un petit détail du modèle. Avec une impression du modèle en 10cm, la boucle est alors plus petite que l'épaisseur minimal de fragilité. Et pour classer un contour l'algorithme plonge la composante connexe dans un *octree* dont les feuilles ont comme taille minimum cette épaisseur. Par conséquent toute la boucle sera dans une seule feuille. La zone entourée par le contour se trouvant dans la même feuille que toute la surface, l'algorithme classera le contour comme entourant une surface fine.

La réparation semi-automatique est illustrée sur la figure (Fig.10.9,(c)).



(a)

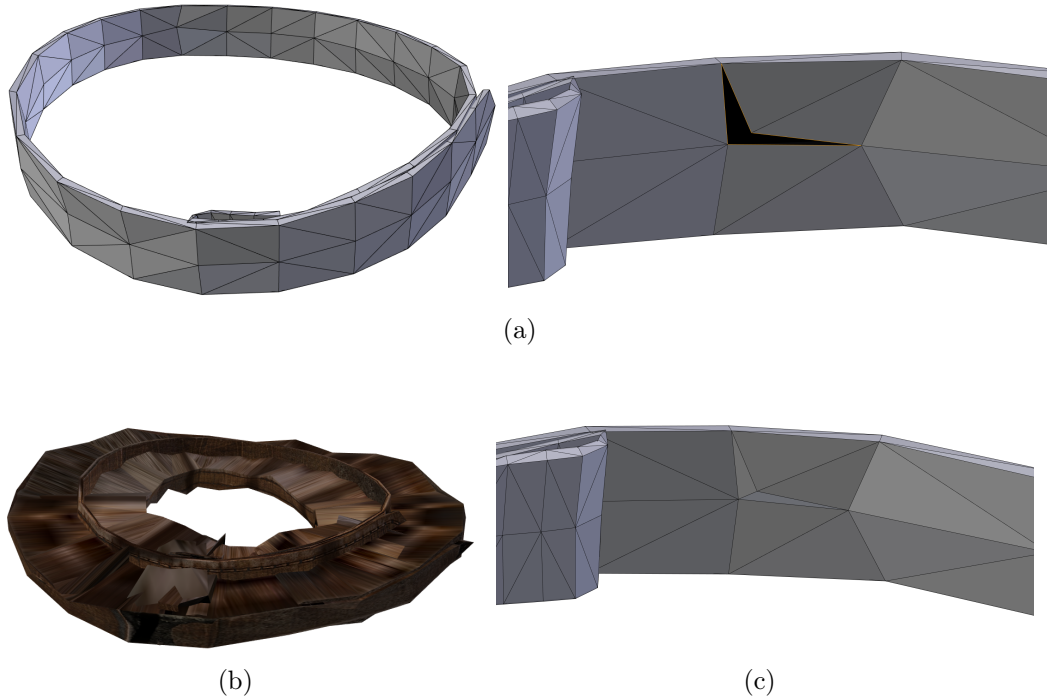
(b) (c)

FIGURE 10.9 – Réparation d'une "Boucle" : (a) Surface originale avec un zoom, (b) Réparation automatique, (c) Réparation semi-automatique.

La composante connexe "Ceinture" possède avant réparation un contour qui visuellement entoure un trou (Fig.10.10,(a)). La réparation automatique va pourtant le classer comme entourant une surface fine et appliquer un épaississement (Fig.10.10,(b)).

L'erreur provient de la forme du contour et des faces l'entourant. Les faces de la réparation par géocentre sensées définir la zone entourée par le contour se retrouvent majoritairement à l'extérieur de celui-ci. L'algorithme va donc calculer son pourcentage en prenant en compte en majorité une zone non entourée par le trou. Comme cette zone est occupée par les faces entourant le trou, le contour va être classé comme entourant une surface fine.

La réparation semi-automatique est illustrée sur la figure (Fig.10.10,(c)).



(a)

(b) (c)

FIGURE 10.10 – Réparation de la "Ceinture" : (a) Surface originale avec un zoom, (b) Réparation automatique, (c) Réparation semi-automatique.

La composante connexe "N° 16" est une surface fine avec une orientation inversée qui représente une partie des cheveux (Fig.10.11,(a)). La réparation automatique va pourtant classer le contour l'entourant comme entourant un trou dans la surface et appliquer un remplissage (Fig.10.11,(b)).

L'erreur provient à nouveau de la réparation par géocentre de l'algorithme de classification. La forme en cercle de la surface et donc du contour l'entourant va placer le géocentre au milieu de ce cercle. La zone définie par la réparation par géocentre va donc se trouver principalement dans une zone vide. L'algorithme va alors classer le contour comme entourant un trou.

La réparation semi-automatique est illustrée sur la figure (Fig.10.11,(c)). En plus du changement de classification du contour, la normale de la surface a été inversée et le paramètre d'épaisseur de l'épaississent a été diminué.

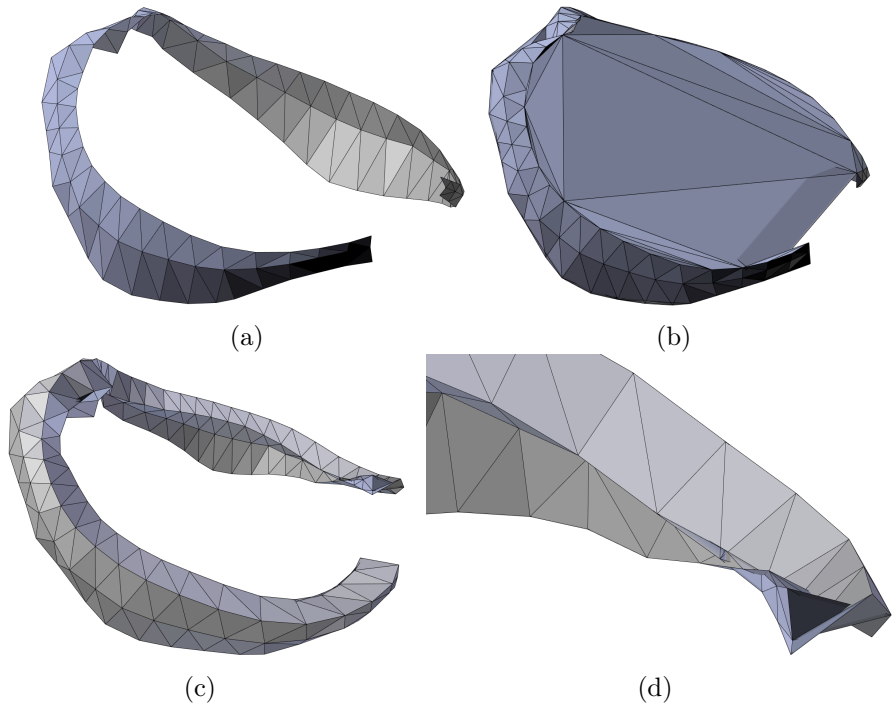


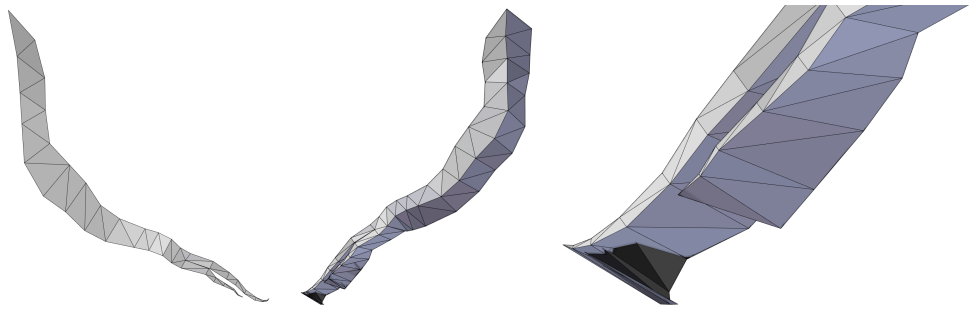
FIGURE 10.11 – Réparation d'une surface de cheveux "N° 16" : (a) Surface originale, (b) Réparation automatique, (c) Réparation semi-automatique, (d) Zoom sur une auto-intersection après réparation.

Épaississent

Le modèle de "Kate" comporte de nombreux petits détails et imprimé en 10cm ces détails deviennent très petits. Le problème c'est que l'épaisseur minimale de fragilité parait énorme comparée à ces détails. Ainsi la plus part des volumes ajoutés par la réparation proviennent de petites surfaces fines représentant les cheveux qui ont été épaissies avec une épaisseur disproportionnée (Fig.10.8,(c)(d)).

La plus part de ces surfaces fines ont leurs faces en double dans le modèle d'entrée avec une orientation inverse. Ces doublons sont supprimés dans un pré-traitement, mais l'aléatoire de la suppression d'une face ou de son inverse rend l'orientation de la surface à la majorité aléatoire également.

La composante connexe "Mèche" est une de ces surfaces fines (Fig.10.12,(a)). On diminue le paramètre d'épaisseur pour deux raisons. Visuellement se rapprocher du volume représenté par le modèle original. Structurellement pour supprimer les auto-intersections provoquées par une épaisseur trop importante. Mais réduire l'épaisseur ne permet pas toujours d'éviter les auto-intersections (Fig.10.12,(b)(c))(Fig.10.11,(c)(d)). Une solution serait d'appliquer une épaisseur variable sur la surface.

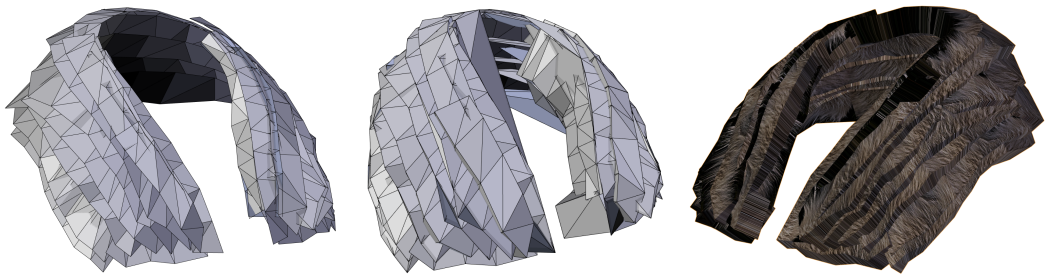


(a) (b) (c)

FIGURE 10.12 – Réparation d'une "Mèche" : (a) Surface originale, (b) Réparation semi-automatique, (c) Zoom sur une auto-intersection après réparation.

L'écharpe est constituée de nombreuses composantes connexes qui sont toutes des surfaces fines (Fig.10.13,(a)). La réparation automatique va en épaissir une partie d'entre elles avec l'épaisseur par défaut. D'autres auront le contour les entourant mal classé pour des raisons similaires à la composante connexe "N° 16" et les remplissages provoqueront des recouvrements (Fig.10.13,(b)).

La réparation semi-automatique va classer tous les contours comme entourant des surfaces fines. Chaque surface fine épaissie va intersecter celle en dessous d'elle augmentant l'épaisseur globale de l'écharpe. L'épaisseur est diminuée sans modifier la fragilité (Fig.10.13,(c)). C'est pareil pour la composante connexe "N° 16" qui elle intersecte les composantes connexes des cheveux et de la tête.



(a) (b) (c)

FIGURE 10.13 – Réparation de l'écharpe : (a) Modèle original, (b) Réparation automatique, (c) Réparation semi-automatique.

Remplissage

La composante connexe "Cheveux" possède avant réparation un contour qui visuellement entoure un trou et a une orientation de surface inversée

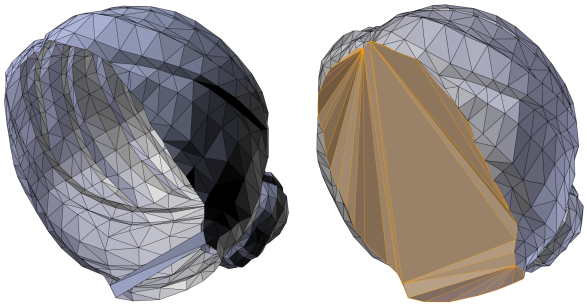
(Fig.10.14,(a)). Le remplissage avec les paramètre par défaut n'est pas idéal et provoque plusieurs recouvrements (Fig.10.14,(b)).

L'algorithme de découpe en segments planaires est assez strict sur la planéarité et a tendance à trop découper le contour. Des petits segments linéaires sont fermés et remplis le long du contour provoquant recouvrement et auto-intersection. L'intérieur du trou est fermé par les incrémentations suivantes. En modifiant le paramètre de marge de planéarité pour le rendre moins strict on peut améliorer le résultat (Fig.10.14,(c)). Un plan sous le modèle est bien construit mais il reste des petits recouvrements.

En séparant manuellement le contour en deux, puis en appliquant un remplissage sur chaque on arrive à un résultat idéal (Fig.10.15,(a)(b)).

(a) (b) (c)

FIGURE 10.14 – Réparation des "Cheveux" : (a) Modèle original, (b) Réparation automatique, (c) Réparation paramétrée.



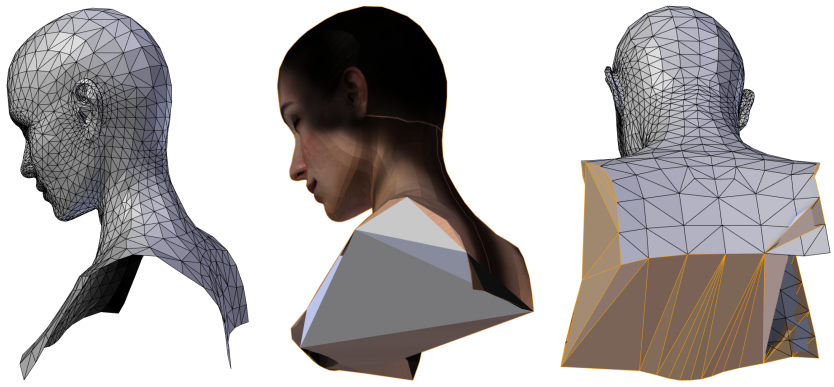
(a) (b)

FIGURE 10.15 – Réparation semi-automatique des "Cheveux" : (a) Découpe manuelle du contour, (b) Remplissage.

La composante connexe "Tête" possède avant réparation un contour difficile à classer (Fig.10.16,(a)). Mais la réparation par remplissage est par expérience la meilleure option, c'est le choix de la réparation automatique. Le contour est complexe et ressemble au modèle "CubeU", car un des plans utilise

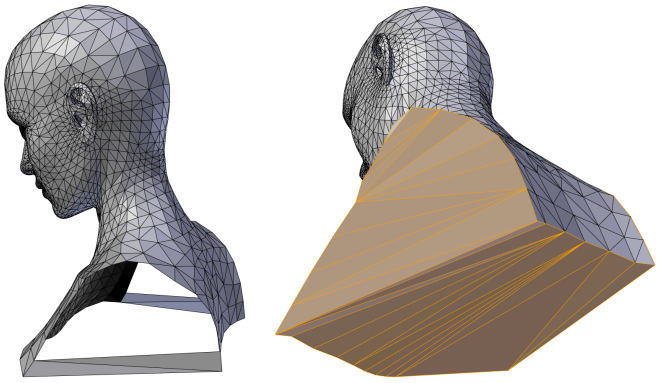
deux segments non consécutifs. Le remplissage avec les paramètre par défaut provoque des auto-intersection et des recouvrements (Fig.10.16,(b)). En modifiant le paramètre de marge de planéarité on améliore un peu le résultat mais les recouvrements sont encore importants (Fig.10.16,(c)).

En séparant manuellement le contour en trois, puis en appliquant un remplissage sur chaque on arrive à un résultat idéal (Fig.10.17,(a)(b)).



(a) (b) (c)

FIGURE 10.16 – Réparation de la "Tête" : (a) Modèle original, (b) Réparation automatique, (c) Réparation paramétrée.



(a) (b)

FIGURE 10.17 – Réparation semi-automatique de la "Tête" : (a) Découpe manuelle du contour, (b) Remplissage.

Réparation semi-automatique

Toutes les composantes connexes ayant besoin d'être corrigées ont des problèmes similaires à ceux présentés ci-dessus. En utilisant manuellement les outils du logiciel toutes les composantes connexes ont pu être corrigées. Chaque

composante connexe est, après réparation, une surface 2D-variété sans bord, orientable et orientée, plongée dans $\mathbb{R}^3$. Chaque composante connexe représente donc la surface frontière d'un volume. La version imprimable du modèle "Kate" est illustré sur la figure (Fig.10.18). Visuellement le résultat est beaucoup plus proche du modèle original que la réparation automatique.

FIGURE 10.18 – Réparation semi-automatique de "Kate" : (c)(d) Zoom sur la tête.

Un choix qui n'a pas été fait ici est de supprimer certaines composantes connexes. Soit car elles représentent un détail trop petit qui ne sera pas visible

à l'impression comme les cils. Soit car elles représentent une fragilité comme les lacets des bottes.

10.1.5 WOW : Réparations de modèle industriel

Le modèle "WOW" est un personnage issu du jeu vidéo "World of Warcraft". C'est un modèle formé de multiples volumes, après la répartition en composantes connexes on en compte 68. Pour ce test on vas appliquer une réparation sur le modèle pour une impression de 10cm de hauteur. Le modèle comporte des surfaces non-variétés, des surfaces mal orientées, des trous dans la surface et des surfaces fines.

Le résultat de la réparation automatique avec tous les paramètres par défaut est présenté dans la figure (Fig.10.19). Le résultat est de qualité identique à celui de "Kate". Les problèmes rencontrés sur les différents composantes connexes sont également similaires à ceux présentés pour "Kate".

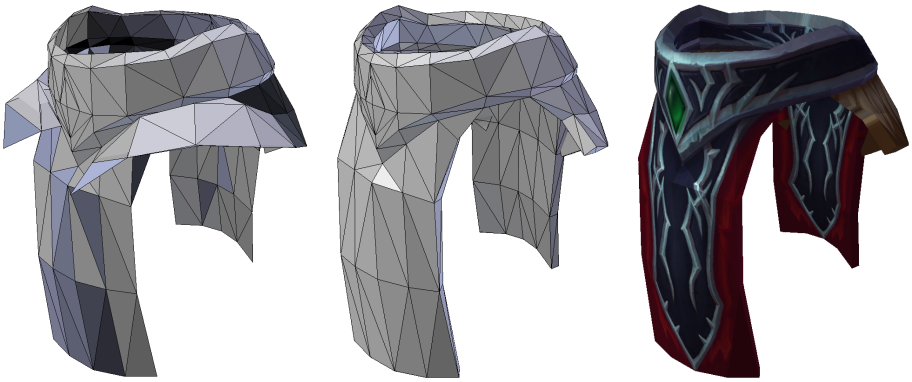
(a) (b)

FIGURE 10.19 – Réparation automatique de "WOW".

La composante connexe "Tabar" possède avant réparation de nombreux problèmes : une orientation incohérente des normales, des recouvrements de faces (faces doublées), des conditions de non-variété, un contour entourant un trou et un contour difficile à classer (Fig.10.20,(a)).

La réparation semi-automatique consiste à forcer le choix d'une réparation par épaississement sur les deux contours et à définir une épaisseur manuellement. Le résultat est une surface 2D-variété sans bord, orientable et orientée parfaitement imprimable (Fig.10.20,(b)(c)). Les faces en moins sur la répara-

tion ont été attribuées à d'autres composantes connexes.



(a) (b) (c)

FIGURE 10.20 – Réparation du "Tabar" : (a) Modèle original, (b)(c) Réparation semi-automatique.

Toutes les composantes connexes ont été corrigées de façons automatique ou semi-automatique en utilisant les outils du logiciel. Chaque composante connexe est, après réparation, une surface 2D-variété sans bord, orientable et orientée, plongée dans $\mathbb{R}^3$. Chaque composante connexe représente donc la surface frontière d'un volume. La version imprimable du modèle "WOW" est illustré sur la figure (Fig.10.21). Visuellement le résultat est beaucoup plus proche du modèle original que la réparation automatique.

Six composantes connexes, composées chacune d'une seule face du modèle original épaissie, ont été supprimées.

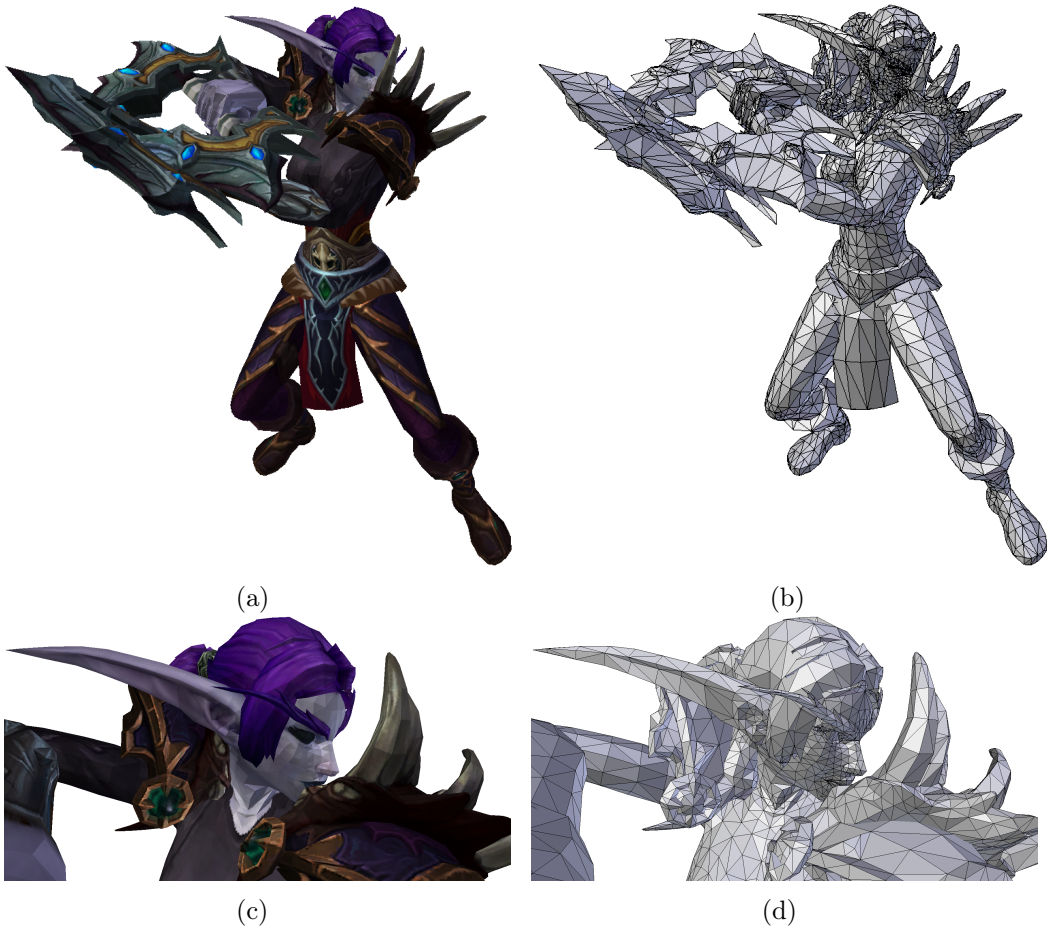
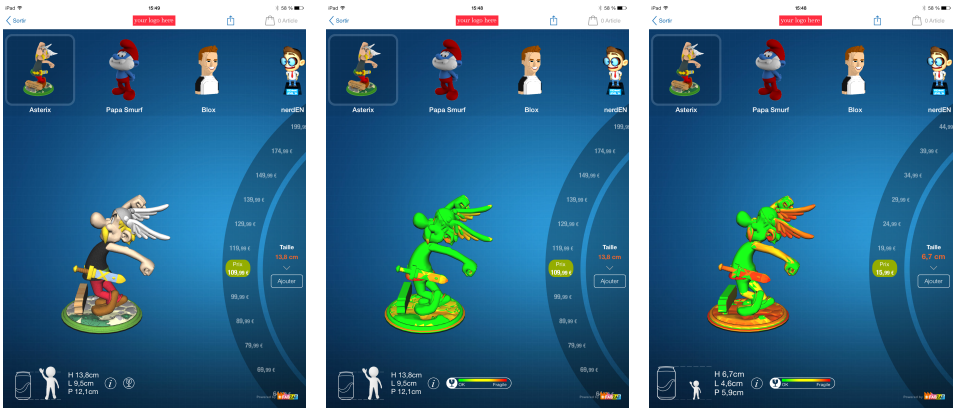


FIGURE 10.21 – Réparation semi-automatique de "WOW" : (c)(d) Zoom sur la tête.

10.2 Validation

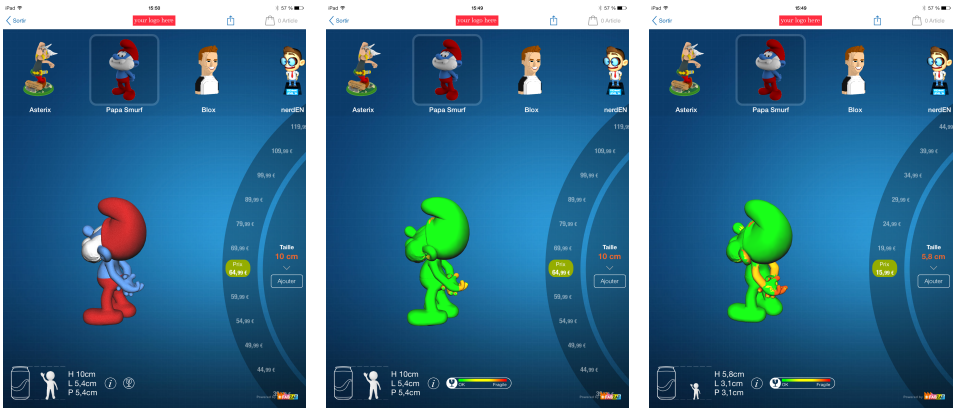
Le plug-in FabZat permet d'ajouter à une application mobile un magasin pour y vendre des impressions 3D. Les modèles vendus sont en général tirés de l'application elle-même avec des possibilités de personnalisation. Une partie du logiciel de réparation est incluse dans le plug-in. Elle permet de donner quelques informations sur la taille du modèle puisque celui-ci peut être généré par l'application. Mais c'est principalement la détection de fragilité qui est utilisée. Elle permet à l'utilisateur de visualiser les risques de casse de l'objet en fonction de la taille d'impression qu'il peut choisir. Elle permet aussi de bloquer certaine taille d'impression une trop grande partie du modèle est détectée comme fragile à cette taille. Deux exemples sont présentés dans les (Fig.10.22) et (Fig.10.23). La première image montre le modèle 3D avec les couleurs d'impression, on peut y voir aussi les dimensions de l'objet ainsi que la roue des prix en fonction de la taille d'impression. Sur les images (Fig.10.22,(b)(c)) et

(Fig.10.23,(b)(c)) on a activé l'outil de détection de fragilité. La visualisation est faite par une échelle de couleur appliquée sur le modèle 3D. Les images (Fig.10.22,(b)) et (Fig.10.23,(b)) montre une taille d'impression plus grande que sur les images (Fig.10.22,(c)) et (Fig.10.22,(c)), on peut voir qu'il y a logiquement plus de zones à risque sur cette dernière.



(a) (b) (c)

FIGURE 10.22 – Capture d'écran du *plug-in* de FabZat sur le modèle d'"Astérix" : (a) Modèle 3D à imprimer, (b)(c) Outil de détection de fragilité sur deux tailles d'impression différentes.



(a) (b) (c)

FIGURE 10.23 – Capture d'écran du *plug-in* de FabZat sur le modèle du "Schtroumpf" : (a) Modèle 3D à imprimer, (b)(c) Outil de détection de fragilité sur deux tailles d'impression différentes.

Cet outil à était utilisé dans des conditions industriels pour la partie e-commerce de FabZat.

10.3 Vulgarisation

Au cours de ma thèse, FabZat a participé à plusieurs événements grand public et d'autres, plus spécialisés. J'ai par conséquent été amené a présenter au grand public l'impression 3D et le fonctionnement de FabZat sur ces événements et salons. La réparation étant un élément important de l'impression 3D et des travaux de Fabzat, j'ai pu présenter mon travail à ces occasions. Quelques exemples de ces événements :

Un stand organisé par FabZat dans le Carrefour de Mérignac, avec une imprimante à poudre couleur sur place (Fig.10.24). Le stand avait pour but de scanner le buste des visiteurs et d'en vendre l'impression 3D. Mais aussi pour Carrefour de faire une animation avec la présentation de la technologie notamment grâce à l'imprimante sur place.

Un stand sur un salon consacré aux nouvelles technologies, avec sur place une imprimante à fil plastique et des objets imprimés à partir l'imprimante à poudre (Fig.10.25).

FIGURE 10.24 – Stand Carrefour, "La Fabrique 3D".

FIGURE 10.25 – Stand FabZat.

Conclusion

La chaîne de réparation conçue et réalisée dans le cadre de la thèse permet de réparer un modèles 3D de façon semi-automatique en vue d'une impression 3D. La réparation semi-automatique consiste à rectifier certains choix automatique. Ces choix sont principalement l'orientation de la surface, le classement des contours et l'ajustement de certains paramètres comme le seuil de l'épaississement. La répartition en composantes connexes permet de valider ou de modifier ces choix facilement au cas par cas.

La chaîne de réparation comprend des étapes qui traitent chacune d'un domaine différent de la modélisation géométrique. L'apport de la thèse n'est pas équivalent sur chacune de ces étapes. Cette différence provient de la pertinence de l'état de l'art, mais aussi des besoins du moment de l'entreprise.

Une conclusion par étape de la chaîne va aborder pour chacune d'entre elles les résultats obtenus et les perspectives d'amélioration qui sont possibles :

L'import est l'étape la plus aboutie, elle permet de passer de n'importe quel modèle 3D en entrée à des surfaces variétés avec ou sans bord. Néanmoins comme aucun choix n'est fait lors de cette étape, nous pouvons observer une sur-duplication des sommets et des arêtes ce qui était anciennement des conditions de non-variété. Le problème de ces sur-duplications sans déplacement des éléments du maillage est qu'elles créent des contours qui s'intersectent et se chevauchent, ce qui peut être source d'instabilité numérique pour les algorithmes de réparation des étapes suivantes.

Une amélioration pourrait être d'évaluer les groupes de faces qui vont utiliser la même occurrence d'un sommet et évaluer s'il est possible d'en fusionner certains. La difficulté est que souvent plusieurs choix de regroupement sont possibles. Il faut donc évaluer ces choix en prenant en compte l'ensemble du modèle 3D.

La méthode d'orientation de la surface par "lancé de rayon" est une amélioration qui peut être ajoutée dans une nouvelle étape à la suite de la réparation par épaississement.

La classification de contours fonctionne sur des exemples extrêmes, mais sur certains cas ambivalents comme le modèle "Casquette", la réparation

choisie provoque des erreurs.

Une première piste d'amélioration est de rajouter une catégorie et une étape de réparation. Cette nouvelle catégorie correspond aux réparations qui relient des éléments d'un même ou de plusieurs contours. Selon la distance des éléments à relier, il est possible de créer une face jointive ou de les fusionner s'ils sont proches ou superposés. Cette réparation fait écho à l'amélioration de l'import, puisqu'elle permettrait aussi de régler le problème de la sur-duplication.

Une deuxième piste d'amélioration est de considérer qu'un contour ne nécessite pas la même réparation sur son ensemble. Il faut alors fractionner la surface en plusieurs pour appliquer le remplissage ou l'épaississement selon la partie du contour gardée par chaque morceau.

Une troisième piste d'amélioration provient de l'analyse des résultats et des limites de la réparation temporaire par géocentre. Il serait peut-être plus efficace d'appliquer les différentes réparations possibles sur le contour puis d'évaluer la qualité de ces réparations pour en sélectionner la meilleure.

La réparation par remplissage fonctionne sur plusieurs types de trous qu'ils soient sur un ou plusieurs plans. La triangulation adaptée du *earclipping* en 3D fonctionne avec une marge de planéarité assez élevée. La principale condition pour son fonctionnement est de ne pas avoir d'auto-intersection du contour après sa projection sur son plan moyen.

Sur des contours plus complexes, la découpe en segments planaires atteint ses limites. Elle a tendance à créer trop de segments de forme allongée le long du contour. Ce qui crée beaucoup d'arêtes de fermeture et donc beaucoup d'itérations. Une amélioration serait de ne plus faire la découpe en parcourant le contour, ce qui aboutit à une vision locale de la planéarité, mais d'analyser la forme globale du contour en essayant de minimiser le nombre de plans. Le plus gros avantage serait alors la possibilité de regrouper des éléments non consécutifs du contour dans un même plan et donc dans une même triangulation.

La réparation par épaississement suit un algorithme existant connu pour la création des surfaces décalées, la méthode par décalage de sommets. Ses limites sont connues et proviennent du rapport entre le rayon de la courbure de la surface et la distance de l'épaississement. Si ce rapport n'est pas bon, l'épaississement crée alors des auto-intersections qui rendent la surface non orientable. De plus, la surface au-delà de ces auto-intersections et le vide qu'elle définit ne correspondent à aucune partie du modèle original.

Dans le contexte de l'impression 3D la difficulté est de combiner une distance d'épaississement adaptée à la taille et une distance d'épaississement suffisamment importante pour ne pas créer de fragilité.

La détection de fragilité permet d'avoir un aperçu immédiat des risques de casse lors de l'impression ou de la manipulation de l'objet imprimé. Le

calcul d'épaisseur est une solution simple pour y parvenir.

Sa limite provient de sa simplicité, car des épaisseurs plus faibles peuvent être imprimées suivant la forme de l'objet. Un futur travail pourrait consister à dresser l'analyse physique prenant en compte le poids. Certains auteurs dans les articles de l'état de l'art proposent des solutions (Stava *et al.* [2012]) mais ce sont des solutions plus complexes et coûteuses en terme de temps d'exécution.

Plusieurs étapes traitant des problèmes de faiblesse pourraient être rajoutées à la fin de la chaîne comme une étape de réparation des fragilités détectées en modifiant la géométrie de l'objet ou bien une étape pour analyser et corriger l'équilibre de l'objet en fonction d'une position donnée (Prévost *et al.* [2013]).

Un moteur physique permettrait de faire une simulation plus réaliste avant l'impression de la viabilité de l'objet et répondrait à plusieurs des problématiques de faiblesse, comme la fragilité ou l'équilibre.

Les apports essentiels de notre thèse :

1. Un algorithme original d'extraction de composantes connexes, 2D variétés avec ou sans bord et orientées, basé sur un parcours de faces voisines et de cycle de faces voisines en ignorant l'orientation de leurs normales.
2. Un algorithme innovant de détection, de suivi et de réparation des bords d'une surface, basé sur la distinction des trous dans la surface, réparés par remplissage et des bords des surfaces fines, réparées par épaississement.
3. Un nouvel algorithme pour la détection des problèmes de fragilités basé sur le "volume parenthésé".
4. Une chaîne de réparation de maillages en vue d'une impression 3D à été conçue à partir des algorithmes proposés.
5. Un logiciel industriel a été développé et mis en production pour FabZat.

Annexe A

Technologies d'impression 3D

A.1 Avant-propos

Il y a plusieurs moyens de produire un objet physique. Les moyens utilisés aujourd'hui dans les chaînes de production peuvent être regroupés en trois grandes familles. La première famille comprend la production par moules où à partir du moule de l'objet, on introduit une matière qui va en prendre la forme et ainsi constituer l'objet. L'autre famille est basée sur la production par découpe, où l'on part d'un bloc de matière dans lequel on va tailler l'objet. Par exemple, les machines de découpe laser et les tours d'usinage. Les technologies de fabrication additive sont une troisième façon de créer des objets. Elles sont basées sur le principe inverse de la découpe. Il s'agit de construire un objet en ajoutant de la matière par couches superposées. En entrée on utilise un modèle numérique 3D de l'objet à produire. Ce modèle est ensuite analysé pour calculer la forme 2D de chacune des couches à produire. Le gros avantage de la fabrication additive est d'être capable de produire n'importe quelle forme d'objets. On peut aussi enchaîner la production de formes différentes sans coût supplémentaire. Ces technologies de fabrication additive sont aujourd'hui communément appelées impression 3D et les machines des imprimantes 3D.

L'addition de matière par couches superposées pose un problème récurrent à toutes les technologies d'impression. Une couche ne peut être construite dans le vide, elle repose sur la couche précédente et/ou sur un support. Les parties qui reposent sur du support sont les parties en porte-à-faux de l'objet. L'importance du support par rapport à la couche précédente, dépend de la forme de l'objet et du sens dans lequel il est imprimé. Les supports sont construits couche par couche en même temps que l'objet. On distingue deux catégories de support :

— Support global : Ce support est construit comme un bloc plein qui entoure l'objet. Il supporte à la fois les porte-à-faux de l'objet mais il

protège aussi l'objet de toute déformation pendant son impression. Il est composé d'une matière différente de l'objet afin d'être retiré facilement.

— Support structure : Ce support est créé automatiquement en fonction de l'objet. Il vient supporter les parties en porte-à-faux de l'objet. Le support est souvent de la même matière que l'objet, il est construit afin d'avoir le moins de zones de contact possibles avec l'objet dans le but d'être retiré plus facilement. La difficulté est de trouver un juste milieu entre supporter suffisamment l'objet pour réaliser toutes les formes possibles, et ne pas avoir trop de contacts avec ce dernier pour qu'il puisse être détachable. Le support peut aussi être dans une matière différente souvent soluble pour être retiré facilement, on peut alors créer une structure plus conséquente et mieux supporter l'objet.

A.2 Evolution

Les premières technologies d'impression utilisaient des liquides photopolymères pour créer les couches (Bourell *et al.* [2009]). Ces liquides ont la propriété de se solidifier au contact de la lumière souvent ultraviolette.

Le premier système proposé est celui de O.J. Munz en 1951 (Munz [1956]), où l'on retrouve quasiment toutes les composantes des imprimantes actuelles. L'objet est construit sur un plateau qui descend dans un tube rempli de liquide photopolymère. Une lumière éclaire la surface du liquide afin de le solidifier, un cache est appliqué sur cette lumière afin de ne solidifier que la forme de la couche. Entre chaque couche on change le cache à la main et on fait descendre le plateau de la hauteur d'une couche.

En 1968 W.K. Swainson (Swainson [1977]) utilise aussi un liquide photopolymère qui se trouve dans un bac transparent. La solidification est faite à l'aide de deux lasers, le liquide se solidifie au croisement de ces deux derniers. On perd l'utilisation d'un plateau amovible, les deux lasers pouvant solidifier le liquide dans tout le bac et non plus uniquement par-dessus comme pour Munz. En plus le système est relié en entrée à des lasers qui analysent un objet physique afin de le reproduire.

En 1971 P.A. Ciraud (Ciraud [1972]) propose le premier système n'utilisant pas un liquide photopolymère mais de la poudre qui va être agglomérée pour construire l'objet. La poudre est déversée de façon continue au-dessus de la surface d'impression. Pour agglomérer la poudre on retrouve l'utilisation de deux lasers mais qui ont cette fois la vocation à chauffer la poudre pour faire fusionner les grains.

Au début des années 1980 on voit apparaître les premiers prototypes qui représentent les technologies actuelles.

En 1979 R.F. Housholder (Housholder [1981]) dépose un brevet présentant un système très proche du système SLS actuel.

H. Kodama (Kodama [1981]) propose plusieurs technologies, il reprend l'idée d'un éclairage global avec un cache de Munz. Néanmoins, il introduit l'idée de la construction par-dessous avec un plateau montant. Il propose également un système avec une tête optique qui émet de la lumière. Cette tête optique est déplacée le long de deux axes, x et y, l'axe z étant la direction du déplacement du plateau. Si on n'a pas retrouvé cette idée dans les imprimantes SLA, elle se rapproche beaucoup du fonctionnement du FDM.

A la même période A.J. Herbert (Herbert [1982]) propose un système avec un laser dirigé par un jeu de miroirs contrôlé par un ordinateur au-dessus de la surface du liquide. A.J. Herbert ne reprend pas l'idée d'un plateau mais rajoute du liquide entre chaque couche à la main.

En 1989 le brevet de la technologie Impression 3D ("3D Printing", 3DP) est déposé par le Massachusetts Institute of Technology (Sachs *et al.* [1989])(Sachs *et al.* [1990]). Les quatre inventeurs cités sur le brevet sont E.M. Sachs, J.S. Haggerty, M.J.Cima et P.A. Williams. C'est la société Z corporation qui va commercialiser la technologie en 1995. Le nom de cette technologie va être étendu à l'ensemble des technologies par fabrication additive.

En 1989 S.S. Crump (Crump [1989]) dépose le brevet de la technologie "Dépôt de matière en fusion" ("Fused Deposed Modelling", FDM). Il fonde à la même période la société Stratasys afin de commercialiser la technologie.

A.3 Utilisation de la matière sous forme de poudre

Une fabrication additive utilisant la matière sous forme de poudre a pour principe de créer l'objet en agglomérant une partie de cette poudre. Il y a plusieurs méthodes d'agglomération de poudre qui varient selon les technologies. La poudre est ajoutée par couche d'environ 0.1mm d'épaisseur (l'épaisseur d'une couche dépend de l'imprimante utilisée) dans le bac d'impression. Elle est étalée par un rouleau sur toute la surface d'impression. Le plateau qui constitue le fond de ce bac d'impression descend de l'épaisseur d'une couche en amont de l'ajout de poudre. On vient ensuite agglomérer sur cette couche de poudre la tranche d'objet lui correspondant. En fin d'impression on se retrouve donc avec un bac rempli de poudre, constituant les parties des couches n'appartenant pas à l'objet, et de poudre agglomérée, représentant l'objet produit. Toute la poudre restée volatile (non agglomérée) est récupérée pour être réutilisée pour les productions suivantes.

Avantages :

— Support global : la poudre non agglomérée sert de support sur toute la surface d'impression.

— Pas de perte de matière, la poudre non agglomérée est récupérée.

Inconvénients :

— Objets creux impossibles, il n'est pas possible de laisser du vide à l'intérieur d'un objet, pour par exemple utiliser moins de matière. Cela aura juste pour effet d'emprisonner de la poudre volatile.

A.3.1 Agglomération par fusion

Frittage sélectif par laser ("Selective Laser Sintering", SLS) :

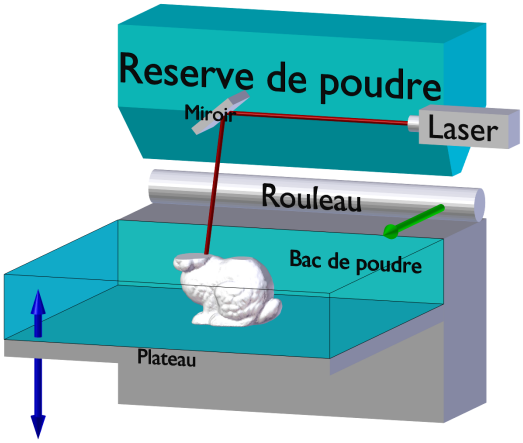


FIGURE A.1 – Schéma du frittage sélectif par laser (SLS).

L'agglomération de la poudre se fait en chauffant la poudre par un laser dirigé sur la surface d'impression à l'aide d'un jeu de miroirs (Fig.A.1). Le laser parcourt la surface en suivant la forme de la tranche de l'objet et en provoque la fusion des grains de poudre.

Avantages :

— Plusieurs matières sont utilisables, principalement des matières plastiques et des matières métalliques.

Inconvénients :

— Temps d'impression long dû aux déplacements du laser.

— Nécessité de post traitement pour finir de solidifier l'objet : L'objet est chauffé dans un four.

Technologies similaires :

— Frittage de métal par laser ("Direct Metal Laser Sintering", DMLS) : Les matières utilisées sont des poudres d'alliage métallique.

— Fusion sélective par laser ("Selective Laser Melting", SLM) : Il s'agit d'une précision sur l'agglomération de la poudre qui est faite par fusion et non par frittage. La différence est une forte réduction de la porosité

dans l'agglomération de la poudre et donc une plus grande solidité de l'objet.

— Fusion par faisceau d'électrons ("Electron Beam Melting", EBM) : On utilise un faisceau d'électrons à la place du laser pour chauffer la poudre.

Imprimante utilisant cette technologie :

— ProXTM 500 de 3DSYSTEMS

— EOS M 290 de E-Manufacturing Solutions

A.3.2 Agglomération par dépôt de liant

Impression 3D ("3D Printing", 3DP) :

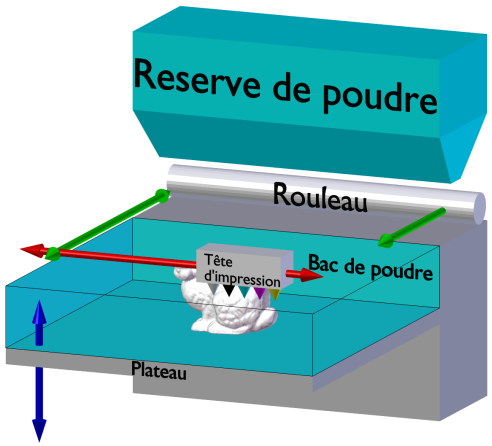


FIGURE A.2 – Schéma de l'impression 3D (3DP).

L'agglomération de la poudre se fait en utilisant un liant liquide qui va coller les grains de poudre entre eux (Fig.A.2). Ce liant est déposé à la surface d'impression par jets de façon identique à une imprimante 2D à jet d'encre. La tête d'impression se déplace avec un mouvement de balayage sur la surface de la couche de poudre. Cette technologie permet en plus d'imprimer des objets en couleurs. Pour ce faire on rajoute à côté de la tête d'impression déposant le liant, quatre autres têtes d'impression pour déposer des encres de couleurs, le cyan, le magenta, le jaune et le noir. Les couleurs sont déposées uniquement sur le contour de la tranche agglomérée de l'objet. En effet, il est inutile de colorer l'intérieur de l'objet.

Avantages :

— Impression des objets en couleurs.

— Temps d'impression rapide : Le temps d'impression d'une couche varie peu quelle que soit la taille de sa surface.

Inconvénients :

— Fragilité des objets, notamment entre les couches et dégradation au contact de l'eau.

— Nécessité de post traitement pour finir de solidifier l'objet : L'objet est trempé dans un liant.

Imprimante utilisant cette technologie :

— ProJet®660Pro de 3DSYSTEMS

A.4 Utilisation de la matière sous forme de liquide photopolymère

Ces technologies utilisent la propriété des liquides photopolymères de durcir sous l'effet de la lumière principalement UV.

On différencie deux types de méthodes d'utilisation de la matière sous forme de liquide photopolymère selon si la matière est contenue dans un bac d'impression ou envoyée par un jet sur un plateau.

A.4.1 Liquide photopolymère dans un bac d'impression.

Stéréolithographie ("StereoLithography Aparatus", SLA) :

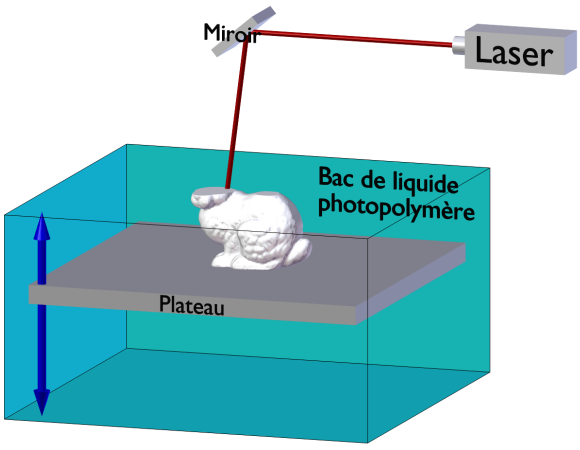


FIGURE A.3 – Schéma de la stéréolithographie (SLA).

La SLA (Fig.A.3) fonctionne avec un plateau qui peut monter et descendre comme pour les technologies à poudre. Il est plongé dans un bac rempli de liquide photopolymère. L'impression commence avec le plateau en position haute et il descend entre chaque couche de leur hauteur. Pour chaque couche,

la tranche correspondante de l'objet est solidifiée. La surface imprimée est ensuite raclée pour nettoyer les résidus. Contrairement aux technologies à poudre, il est inutile de rajouter de la matière entre les couches, le liquide étant présent dans le bac. Une variante de la SLA consiste à solidifier par le dessous du bac de liquide. Le plateau est alors montant avec l'objet accroché par dessous. Cela permet d'avoir moins de liquide dans le bac en début d'impression. Néanmoins, il faut faire attention à ce que l'objet n'accroche pas sur le fond du bac.

Deux types de méthodes de solidification sont connues : par laser ou par cache.

Solidification d'une couche par laser UV.

L'utilisation d'un laser se fait de la même façon que pour le SLS. Le laser est dirigé sur la surface d'impression à l'aide d'un jeu de miroirs. Le laser parcourt à la surface du liquide la forme 2D de la coupe en cours. Au passage du laser le liquide se solidifie pour former la couche imprimée.

Solidification d'une couche par cache.

Cette méthode consiste à utiliser une lumière UV qui éclaire l'ensemble de la surface du liquide. Un cache formant le négatif de la forme 2D de la coupe en cours est appliqué sur cette lumière. La surface du liquide se solidifie alors aux endroits où le cache laisse passer la lumière formant la couche imprimée. Ce cache est créé automatiquement entre deux couches. Les premiers systèmes utilisaient en guise de cache une plaque de verre à charge électrostatique effaçable. Aujourd'hui, on trouve principalement des écrans LCD. L'utilisation d'un cache rend l'impression d'une couche beaucoup plus rapide par rapport au laser.

Avantages :

— Post traitement rapide : Il consiste à découper manuellement le support quand celui-ci est nécessaire. L'objet est directement solide.

— Pas de perte de matière, le liquide non polymérisé est récupéré.

Inconvénients :

— Support structure : Le support est de la même matière que l'objet.

— Objets creux impossibles. Il n'est pas possible de laisser du vide à l'intérieur de l'objet. Cela aura juste pour effet d'enfermer du liquide non polymérisé à l'intérieur de l'objet.

Imprimante utilisant cette technologie :

— ProJet®7000 HD de 3DSYSTEMS

— Form 2 de FormLabs

"Solid Ground Curing" (SGC) :

Le SGC est une technologie dérivée de la SLA. Il rajoute des étapes afin de régler le problème du support des technologies de la SLA classique. Pour cela, il va après la polymérisation d'une couche, enlever le liquide polymère restant et le remplacer par de la cire liquide. En rapprochant une plaque réfrigérante on solidifie la cire. On va ensuite reprendre le processus standard en posant la nouvelle couche de polymère liquide. La couche est solidifiée par une méthode de cache avec une plaque électrostatique. On se retrouve donc en fin d'impression avec un bloc de cire qui contient notre objet imprimé. Le SGC n'est aujourd'hui plus utilisé principalement à cause du temps d'impression grandement rallongé par les étapes supplémentaires qu'il demande.

Avantages :

— Support global : La cire est utilisée comme support.

Inconvénients :

— Objets creux impossibles. Il n'est pas possible de laisser du vide à l'intérieur de l'objet. Cela aura juste pour effet d'enfermer de la cire à l'intérieur de l'objet.

— Temps d'impression élevé : Les étapes de construction du support en cire prennent du temps.

— Matière utilisée pour le support non récupérable.

A.4.2 Liquide photopolymère envoyé par jet :

Fabrication (Modelage) à jets multiples ("Multi Jet Modelling", MJM) :

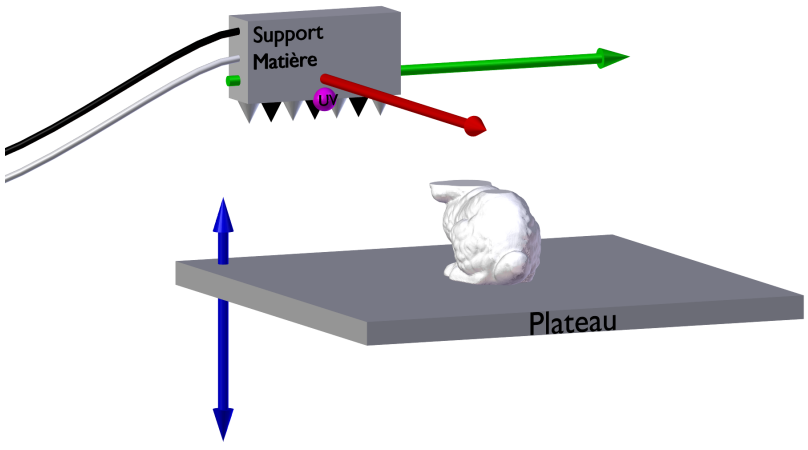


FIGURE A.4 – Schéma de la fabrication à jets multiples (MJM).

Le MJM (Fig.A.4) fonctionne avec une tête d'impression. Celle-ci comprend un grand nombre de jets. Pour chaque couche la tête va balayer la surface du plateau et va déposer le liquide photopolymère aux endroits de la forme 2D de la coupe. Le liquide est instantanément solidifié par une lampe UV qui éclaire tout le plateau. Cette lampe est généralement située sur la tête d'impression. Cette méthode fonctionne sur le principe inverse des méthodes à bac. Dans ces derniers, on a de la matière sur toute la surface et la lampe UV éclaire précisément les parties à solidifier. A l'inverse, lors du MJM, la lampe UV va éclairer toute la surface d'impression et poser la matière précisément là où l'on veut créer l'objet. La moitié des jets envoie la matière qui constitue l'objet, l'autre moitié envoie la matière du support. Le support est créé de façon à englober tout l'objet et en soutenant les porte-à-faux. Il n'y a pas de structure de support mais une enveloppe englobante qui est enlevée à la main avec l'aide d'un jet d'eau.

Avantages :

— Support global : Le support n'est pas présent sur toute la surface d'impression mais englobe l'objet.

— Temps d'impression rapide.

— Post traitement rapide, il consiste à enlever le support, l'objet lui est directement solide.

Inconvénients :

— Objets creux impossibles. Il n'est pas possible de laisser du vide à l'intérieur de l'objet. Cela aura juste pour effet d'enfermer du support à l'intérieur de l'objet.

— Matière utilisée pour le support non récupérable.

Imprimante utilisant cette technologie :

— Objet Eden260VS de STRATASYS

— Objet260 Connex 1 de STRATASYS

A.5 Autres méthodes

A.5.1 Dépôt de matière en fusion ("Fused Deposed Modelling",FDM)

Le FDM (Fig.A.5) utilise comme matière principalement du plastique PLA où ABS. On peut aussi l'utiliser avec d'autres matières comme de l'argile ou du chocolat. On va utiliser la propriété que ces matières une fois chauffées sont dans un état liquide (température de fusion : PLA 170°C et ABS 200°C) et elles peuvent alors être déposées pour former la couche à imprimer. La matière

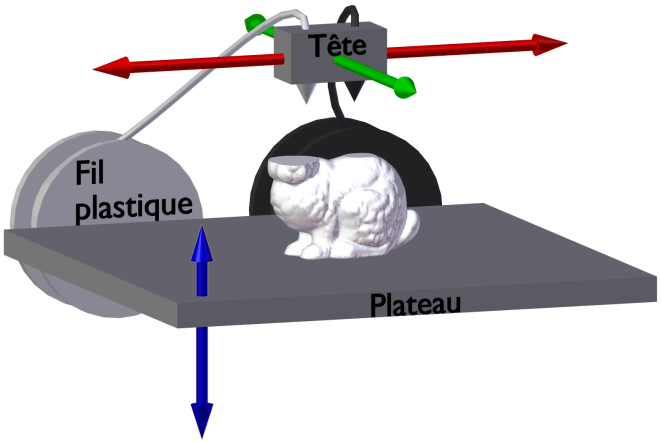


FIGURE A.5 – Schéma du dépôt de matière en fusion (FDM).

est déposée par une tête d'impression qui peut se déplacer librement sur un plan 2D (x, y) contrairement aux précédentes technologies comme le 3DP et le MJM qui ont un déplacement de leurs têtes d'impression par balayage. De manière générale le plateau descend entre chaque couche. Il existe des imprimantes qui ne bougent pas le plateau mais permettent à la tête de se déplacer en plus sur l'axe vertical (z). L'objet se construisant couche par couche sur un plateau, la matière déposée repose sur la couche précédente. On ne peut donc pas faire d'objets avec un porte-à-faux trop important. En conséquence il faut parfois rajouter un support. Ce support peut être de la même matière que l'objet mais dans ce cas il doit pouvoir être détachable de ce dernier après l'impression. D'autres imprimantes utilisent une deuxième tête avec une matière soluble dans l'eau pour créer le support. Le support peut être dans ce cas plus volumineux car plus facile à enlever.

Avantages :

— L'éventail de matières possibles est très large.

— C'est la seule technologie à pouvoir laisser une zone d'air à l'intérieur de l'objet imprimé.

— Possibilité de changer de matières entre deux impressions sur la même imprimante.

— Post traitement rapide : Il consiste à découper manuellement le support quand celui-ci est nécessaire. L'objet est directement solide.

Inconvénients :

— Support structure. Certains FDM permettent de réaliser le support dans une matière soluble ce qui nuance cet inconvénient.

— Temps d'impression très lent : Le déplacement de la tête est lente.

Technologies similaires :

— Dépôt de poudre métallique ("Direct Metal Deposition", DMD) : Il s'agit d'une adaptation du FDM à de la poudre métallique. La tête d'impression dépose la poudre tout en la chauffant à l'aide de plusieurs lasers.

Imprimante utilisant cette technologie :

— Dimension 1200es de STRATASYS
— Ultimaker 2 d'Ultimaker
— Replicator 2 de MakerBot
— Huxley de RepRap

A.5.2 Fabrication d'objets laminés ("Laminated object manufacturing", LOM)

Le LOM va utiliser le principe d'impression couche par couche et la découpe laser. La matière est sous forme d'un long rouleau adhésif. Entre chaque couche on descend le plateau d'impression et on fait tourner le rouleau pour amener une nouvelle couche de matière. Les couches se tiennent entre elles grâce à la propriété adhésive du rouleau. Le laser découpe dans un premier temps un rectangle qui correspond à la zone d'impression qui va se détacher du rouleau. On découpe ensuite la forme de la tranche de l'objet en traçant son contour avec le laser. Le laser va également découper les zones qui ne font pas partie de l'objet mais qui servent de support afin que celui-ci soit détachable de l'objet en fin d'impression. Le support est découpé soit par quadrillage soit de manière plus intelligente en fonction de la forme de l'objet.

Les matières principalement utilisées sont le plastique et le papier. On peut utiliser des feuilles adhésives de la taille de la zone d'impression à la place du rouleau ce qui évite une grande perte de matière. Il existe une version de cette technologie qui permet d'imprimer des objets en couleurs. Elle utilise comme matière en entrée des feuilles de papier mais celles-ci sont imprimées par une imprimante couleur 2D classique avant d'être déposées pour la découpe.

Avantages :

— Support global : Les feuilles sont superposées créant un support sur toute la surface.
— Impression des objets en couleurs.

Inconvénients :

— Grosse perte de matière. Que ce soit la matière qui reste sur le rouleau ou la matière qui sert de support, aucune n'est réutilisable.
— Objets creux impossibles. Il n'est pas possible de laisser du vide à l'intérieur de l'objet. Cela aura juste pour effet d'enfermer du support à l'intérieur de l'objet.

Imprimante utilisant cette technologie :

— Mcor IRIS HD de Mcor Technologies

A.6 Résumé

Pour synthétiser l'information de cette annexe la figure (Fig.A.6) présente une classification des technologies d'impression 3D en fonction du type de matière et de la technique utilisée.

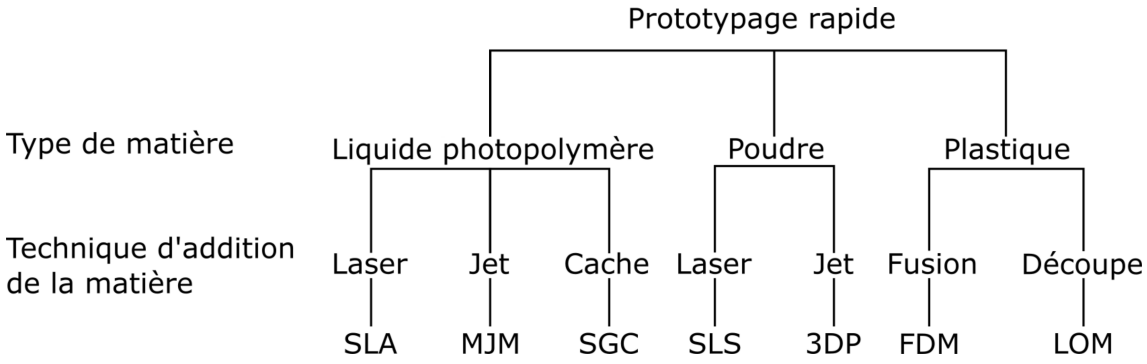


FIGURE A.6 – Classification des technologies d'impression 3D.

Annexe B

Documentation du code

B.1 Import

B.1.1 *IO*

read_ obj(),*read_ mtl()* : sont les fonctions qui permettent l'import du modèle 3D à partir d'un fichier d'entrée au format OBJ. La fonction *read_ obj()* applique aussi les premières étapes de la chaîne, répartition en composantes connexes, réparation des conditions de non-variété, réparation des normales incohérentes. Enfin la fonction construit une structure *Halfedge du modèle*.

write_ Obj() : est la fonction qui permet d'exporter le modèle 3D en fin de chaîne de réparation dans un ficher au format OBJ.

CenterResize() est une fonction qui permet de changer les valeurs de position des sommets pour recentrer le modèle sur l'origine (0,0,0) et de lui donner une taille définie sur l'axe Y correspondant par exemple à sa taille d'impression en centimètre. Le paramètre *sizeY* est par défaut à -1 si on veut uniquement recentrer le modèle sans modifier sa taille.

B.1.2 *Mesh_ Info*

La classe *Mesh_ Info* permet de stocker des informations sur le modèle 3D en dehors des éléments sommets, arêtes, faces. Une instance de cette classe est présente dans le *package* **Application**. Cette instance sera transmise aux différentes étapes de la chaîne de réparation enfin qu'elles aient accès à ces informations. Les informations de couleur et de texture y sont enregistrées. Le format OBJ enregistre ces informations de couleur et de texture en grande partie dans un second fichier au format MTL. On retrouve les informations de ce fichier dans la liste de matériaux de *Mesh_ Info* (*m_ materials*). Pour les faces texturées on a besoin de connaître leurs *vertex texture* qui sont une coordonnée 2D pour identifier une position sur la texture que possède chaque

sommet de la face. C'est la face qui possède cette information et non le sommet, ainsi un même sommet peut avoir un *vertex texture* différent par face qui l'utilise. *Mesh_Info* enregistre l'ensemble des positions des *vertex texture* (*m_vertexTexture*) pour que les faces n'aient plus qu'à conserver un identifiant de la position de leurs *vertex texture* dans ce tableau. Ce qui permet notamment de ne pas dupliquer l'information quand plusieurs faces utilisent un même *vertex texture*. L'identifiant est d'abord stocké dans *m_idVT1*, *m_idVT2* et *m_idVT3* de *Face_Tmp* puis au passage à la structure *Halfedge* dans une *face_property* de *Surface_mesh*. Toutes ces informations seront réutilisées à l'export pour redonner ces informations au format OBJ, mais aussi lors de la création de nouvelles faces pour lui attribuer une couleur ou une texture proche de ses faces voisines.

La classe *Mesh_Info* permet aussi de stocker des informations issues des différentes étapes de réparation et utiles aux réparations suivantes. Par exemple, le nombre de composantes connexes est défini lors de l'import et utilisé par toutes les étapes suivantes jusqu'à l'export qui crée un objet au sens du format OBJ différent par composante.

B.1.3 *Material*

La classe *Material* reprend les informations présentes dans un matériau du format MTL, mais en simplifié. En effet pour l'impression 3D toutes les informations sur la brillance et les reflets d'un matériau n'ont pas d'utilité. On garde donc seulement le nom du matériau (*m_name*), sa couleur diffuse, la seule couleur qui sera utilisée par l'imprimante(*m_Kd*), le chemin vers le fichier de la texture diffuse, la seule texture utilisée par l'imprimante (*m_mapKd*).

B.1.4 *Face_Tmp*

La classe *Face_Tmp* stocke les identifiants de tableau des éléments sommets et arêtes qui la composent. Elle stocke aussi l'identifiant du matériau utilisé par la face, l'identifiant des *vertex texture* du tableau de *Mesh_Info*.

Et pour finir, elle stocke deux informations définies lors de l'import, l'identifiant de la composante connexe à laquelle appartient la face qui sera transmise à la structure *Halfedge* et l'orientation de cette face par rapport au reste de la composante connexe qui servira à retourner ou non cette face en fonction de la majorité au moment de la création de la structure *Halfedge*.

B.1.5 *Vertex_ Tmp*

La classe *Vertex_ Tmp* stocke la position du sommet (*m_ x*,*m_ y*,*m_ z*). La classe enregistre également une liste de références des faces qui utilisent ce sommet (*m_ faces*). Cette liste sert à l'algorithme de répartition des faces, qui détermine quel groupe de faces doit utiliser un même sommet pour corriger les non-variétés (Algo.4). Cet algorithme est présent ici sous la fonction *splitVertex()*.

B.1.6 *Edge_ Tmp*

La classe *Edge_ Tmp* stocke les arêtes comme des éléments de faces ainsi il existe autant d'arêtes utilisant deux mêmes sommets comme extrémités que de faces utilisant ces deux mêmes sommets. Un référencement vers la face de l'arête est stocké (*m_ F1*), ainsi qu'un identifiant du tableau de sommets stocké dans *IO* pour les deux sommets extrémité (*m_ idV1*,*m_ idV2*). L'orientation de l'arête n'est pas définie par *m_ idV1* vers *m_ idV2* ou inversement, car *m_ idV1* est en fait le plus petit identifiant des deux. Pour conserver l'orientation, la classe possède un booléen qui prend la valeur "faux" si l'orientation est de *m_ idV1* vers *m_ idV2* et "vrai" si elle est de *m_ idV2* vers *m_ idV1*. Cette astuce est utile, car le tableau d'arêtes de *IO* (*m_ edges*) est trié, d'où la redéfinition des opérateurs "égal" et "inférieur". Le tri se fait d'abord selon *m_ idV1* puis *selon m_ idV2*. L'idée est que les arêtes qui utilisent les mêmes sommets se trouvent côte à côte dans le tableau pour plus facilement en compter les occurrences (Algo.3).

B.2 Classification/Repair

B.2.1 *Repair*

La classe *Repair* est la classe principale du package. Ses attributs *m_ mesh* et *m_ meshInfo* sont des pointeurs transmis par le *package* **Application** de ses instances de *Surface_ mesh* et de *Mesh_ Info*.

findOutline() est la fonction qui correspond à l'algorithme (Algo.7). Il recherche les contours présents sur chacune des composantes connexes du modèle 3D. Un paramètre est disponible pour lancer la recherche sur une composante connexe spécifique. Le résultat est enregistré dans *m_ outlinesCC* un tableau contenant pour chaque composante connexe un tableau de ses contours. Ces derniers sont sous la forme d'un tableau ordonné contenant les arêtes qui les composent (*vector<Surface_ mesh : :Halfedge*).

Si la fonction est lancée à plusieurs reprises que ce soit sur tout le modèle ou sur une composante connexe en particulier, toutes les arêtes de bord précédemment détectées et placées dans un contour ne peuvent être utilisées comme départ d'un nouveau contour. Des nouveaux contours ne peuvent donc exister que s'il y a eu de nouvelles faces créées, par exemple pendant les étapes de réparation. Par contre les arêtes de bord utilisées précédemment peuvent appartenir à un de ces nouveaux contours si elles sont à la suite d'une nouvelle arête de bord. C'est le cas si par exemple, un précédent contour n'a été réparé que partiellement.

Le résultat n'efface pas le *m_ outlinesCC* précédent, mais y rajoute les nouveaux contours. Comme on ne touche pas aux contours précédemment trouvés, si un contour a été refermé alors il reste quand même dans le tableau *m_ outlinesCC*. La raison est que les indices de ce tableau sont utilisés par d'autres étapes de la chaîne, classification, réparation. On préfère donc que l'indice corresponde toujours au même contour même si celui-ci a déjà été traité.

detectSurfacePlane() est la fonction qui réalise l'étape de classification (Algo.8). Cette étape utilise un *octree* en se servant du *package* **Octree**. Elle prend deux paramètres, *limitFillingVsOffset* et *limitThickness*.

Le premier paramètre *limitFillingVsOffset* indique le seuil en pourcentage entre le choix de classer le contour comme devant être réparé par remplissage ou par épaississement. Ce pourcentage correspond au nombre de feuilles de l'*octree* contenant au moins une face de la réparation test et une face originale avec une normale de la face de réparation à plus de 90 degrés de différence de la moyenne des normales des faces originales sur le nombre de feuilles de l'*octree* contenant au moins une face de la réparation test. Il est par défaut à cinquante pour cent.

Le second paramètre *limitThickness* correspond à la distance minimum entre deux points de la surface sous laquelle l'objet imprimé risque de se casser. Il sert à déterminer la taille des feuilles de l'*octree*. Il est par défaut à 3mm, distance pour l'imprimante Zcorp 650.

Ces deux paramètres sont transmis par le *package* **Application** par le biais de la fonction *fullRepair()*.

Le résultat est rendu sous la forme d'un double tableau d'entiers. Le premier tableau correspond aux composantes connexes et le deuxième aux contours de chacune d'entre elles. L'entier est un simple code, 1 pour une réparation par épaississement, et 2 pour une réparation par remplissage.

fullRepair() procède pour les étapes de réparation à une itération par composante connexe puis par contour afin d'appliquer la réparation par remplissage sur les contours classés comme tels. Pour cette réparation *fullRepair()* ajoute une boucle itérative avec les fonctions *holeFilling()* et *findOutline()*. La boucle

vérifie deux conditions après la réparation effectuée par *holeFilling()*.

La première, à l'aide de la fonction *findOutline()* exécutée sur la composante connexe du contour traité, on vérifie si de nouveaux contours ont été détectés. Si c'est le cas cela signifie que le trou a été partiellement réparé. On continue alors la boucle en exécutant *holeFilling()* sur tous les nouveaux contours. Sinon, c'est que le contour original ou les nouveaux contours pour les tours de boucle suivants, ont tous été réparés.

La deuxième est la différence du nombre de faces du modèle avant et après réparation. Si celui-ci n'a pas changé c'est que la réparation n'a fait aucun changement sur le modèle. Il est donc inutile de continuer la boucle et la réparation sur ce contour s'arrête sans être complètement réparée.

La réparation par remplissage est réalisée par les fonctions *holeFilling()*, *repairForHoleFilling()* et *hrepairEarClipping()*.

holeFilling() est la fonction qui s'occupe de la recherche des segments planaires sur le contour (Algo.9), puis crée une arête entre le début et la fin de chaque segment (Algo.10) en vérifiant qu'elle n'intersecte pas le segment. Et enfin, elle lance la fonction *repairForHoleFilling()* sur les segments fermés par les arêtes ajoutées.

La fonction *holeFilling()* prend deux paramètres, *idCC* l'identifiant de la composante connexe sur lequel se trouve le contour et *idO* l'identifiant du contour. Ces deux identifiants sont utilisés comme indice du tableau *m_outlinesCC* pour accéder au contour à remplir.

Pour fermer un segment il faut créer une arrête, mais la structure *Surface_mesh* ne permet pas la création d'une arête isolée. On crée alors une face temporaire en utilisant les deux sommets aux extrémités du segment et un troisième sommet temporaire créé entre les deux premiers décalés dans la direction de la normale au plan d'une unité. La face temporaire est supprimée après le remplissage du segment *repairForHoleFilling()*. L'arête est conservée, car si le remplissage s'est correctement effectué alors une face créée pour remplir le segment l'utilise aussi. Le sommet temporaire est lui supprimé en même temps que la face car aucune autre face ne l'utilise.

repairForHoleFilling() est une fonction préalable à la fonction *repairEarClipping()* pour vérifier que la bonne orientation de la normale au trou est choisie. Elle prend en paramètre *halfEdge* une arête de bord du segment qui entoure le trou à remplir, qui sera transmis à *repairEarClipping()*. Le reste du segment n'est pas transmis, il est retrouvé en suivant les arêtes de bord consécutives.

repairEarClipping() commence par calculer une normale correspondant au plan moyen du trou à remplir. L'orientation de cette normale a une grande importance, car elle va déterminer si une oreille peut être fermée (angle saillant) ou non (angle rentrant). Ce qui à pour conséquence que pour une orientation

l'algorithme va pouvoir remplir entièrement le trou alors que pour l'orientation opposée non. Pour rappel une oreille est constituée de deux arêtes consécutives du segment. Pour s'assurer de la bonne orientation de la normale au trou, on fait une sauvegarde du modèle en copiant *m_mesh* et *m_meshInfo*, avant l'appel de la fonction *repairEarClipping()*. Si le trou n'est pas entièrement rempli alors on relance *repairEarClipping()* sur la sauvegarde en passant à *true* le paramètre optionnel *bNormalHole* pour changer l'orientation. Dans le cas où aucune des deux orientations ne réussit à remplir entièrement le trou alors on choisit la réparation où il reste le moins d'arêtes de bord.

repairEarClipping() applique l'algorithme de *EarClipping* adapté en 3D. Elle prend en paramètre *halfEdge* une arête de bord du segment qui entoure le trou, transmise par *repairForHoleFilling*. Et un paramètre booléen optionnel *bNormalHole* initialisé à *false* qui quand il est passé à *true* inverse l'orientation de la normale au plan moyen du trou. La fonction retourne un entier qui vaut 0 si le trou à été entièrement rempli. Elle retourne -1 si une erreur se produit dans l'algorithme, par exemple une face n'arrive pas à être créée. Et elle retourne un nombre correspondant au nombre d'arêtes de bord restantes quand plus aucune oreille ne peut être fermée et donc que le remplissage n'est que partiel.

offset() réalise la réparation par épaississement (Algo.11). La réparation par épaississement se fait sur toute la composante connexe et répare tous les contours présents simultanément sur cette dernière. C'est pour cette raison que la fonction *fullRepair()* ne répare dans un premier temps que les contours classés dans la réparation par remplissage de la composante connexe. Une fois ces réparations effectuées si la composante connexe possède un ou plusieurs contours classés dans la réparation par épaississement alors *fullRepair()* exécute la fonction *offset()* sur la composante connexe.

Pour certains modèles l'utilisateur peut préférer que la réparation par épaississement soit prioritaire. Par exemple, si sur une composante connexe un contour est classé dans la réparation par épaississement alors tous les contours de cette composante connexe seront réparés par cette méthode. Une option pourra être ajoutée à cet effet.

La fonction *offset()* prend en paramètre l'identifiant de la composante connexe à épaissir *idCC*, les indices des contours à réparer par l'épaississement *idOutlineToOffset* sous la forme d'un tableau d'entiers. Le dernier paramètre est l'épaisseur de l'épaississement *offsetWidth*, il est donné par *fullRepair()* et correspond à la valeur de *limitThickness* qu'on a déjà utilisé pour l'étape de classification.

Les contours *idOutlineToOffset* vont servir à créer les faces de côté qui relient les faces originales de la composante connexe aux nouvelles faces créées par décalage. Les faces de côté sont créées entre les arêtes de bord de ces

contours et leurs images décalées. On inclut dans ces contours, ceux qui ont été classés dans la réparation par épaississement. On pourrait aussi inclure les contours classés dans la réparation par remplissage qui n'ont pas pu être réparés ou seulement partiellement, afin de garantir la réparation de toutes les arêtes de bords. Mais par défaut il semble préférable de les laisser pour les vérifier manuellement. Si la réparation par remplissage n'a pas fonctionné c'est que le contour est complexe, appliquer un épaississement par dessus une réparation non effectuée ou partielle risque de créer des intersections rendant la surface non orientable.

B.2.2 *Segment*

Segment est une structure dans laquelle on enregistre les informations des segments planaires détectés.

- *idSegment*, l'indice du segment actuel dans le tableau *segments* qui contient l'ensemble des segments détectés.
- *idPlan*, l'indice du plan du segment dans les deux tableaux *normalPlan* et *pointPlan* qui permet l'accès à la normale et à un point de ce plan.
- *beginV*, l'indice du premier sommet du segment dans le tableau *vertexSvg* qui contient tous les sommets du contour dans l'ordre. *vertexSvg* est organisé pour que le premier sommet du tableau soit le premier sommet du premier segment.
- *endV*, l'indice du dernier sommet du segment dans le tableau *vertexSvg*.
- *firstVUSe*, booléen indiquant si le premier sommet a été utilisé par une arête de fermeture d'un autre segment ou si le segment a lui-même été fermé. Dans le cas général, un segment utilise pour son arête de fermeture le dernier sommet du segment précédent et son dernier sommet.
- *lastVUSe*, booléen indiquant si le dernier sommet a été utilisé par une arête de fermeture d'un segment ou si le segment a lui-même été fermé. Dans certains cas le dernier sommet du segment n'est pas utilisé pour l'arête de fermeture, il est alors remplacé par le premier sommet du segment suivant.

B.2.3 *Ear*

Ear est la classe utilisée par *repairEarClipping()* pour enregistrer et manipuler les oreilles. Elles sont créées en début d'algorithme dans le même parcours des arêtes de bords qui calcule la normale au plan du trou. Elles sont ensuite enregistrées dans un tableau *earTip*.

- *m_h*, première des deux arêtes de bord qui constitue l'oreille (*Surface_mesh : :Halfedge*).

— *m_ cosAngle*, l'angle entre les deux arêtes calculé pendant le parcours . C'est toujours l'angle saillant qui est calculé.

— *m_ normal*, c'est la normale au plan de l'oreille. Elle est calculée par un produit vectoriel entre la deuxième arête de l'oreille et l'inverse de la première. C'est cette normale qui est comparée à la normale au trou pour savoir si l'oreille peut être fermée ou non. L'oreille peut être fermée si les deux normales sont espacées d'un angle de moins de 90 degrés.

— *operator==*, l'opérateur d'égalité est redéfini pour être basé sur la valeur de l'angle *m_ cosAngle*.

— *operator<*, l'opérateur d'infériorité est redéfini pour être basé sur la valeur de l'angle *m_ cosAngle*.

Le tableau *earTip* est trié à l'aide de la redéfinition des opérateurs de la classe *Ear* pour donner la priorité aux oreilles ayant l'angle le plus petit. Le tri est refait à chaque nouvelle oreille fermée.

B.3 Weakness

B.3.1 *Distance*

calculateDistanceWithOctree() est la fonction qui implémente l'algorithme (Algo.12). La fonction *calculateDistanceAndColorWithOctree()* rajoute en plus après le calcul une visualisation en changeant la couleur des faces du modèle 3D. Ces fonctions prennent en entrée une taille correspondant à la hauteur pour laquelle on veut tester la fragilité. Comme vu précédemment, les informations spatiales du modèle 3D ne sont pas forcément à l'échelle de l'impression.

colorWithNewSize() est la fonction qui permet de tester la fragilité pour une nouvelle taille d'impression sans refaire le calcul de distance. Pour cela on fait un simple produit en croix en utilisant les attributs *m_ originalSize* et *m_ listDistanceOriginal* qui sont attribués lors du calcul de distance, puis on procède au changement de couleur des faces.

B.4 Octree

B.4.1 *Octree*

detectSurfacePlane() est une fonction qui implémente une partie de l'algorithme de classification de contour. Il s'agit de la partie principale (Algo.8) qui calcule en parcourant les feuilles le pourcentage déterminant quelle réparation doit être appliquée sur le contour. Il prend en entrée l'indice du contour

à classer *indiceOutline*. Cet indice est utilisé pour reconnaître les faces de la réparation temporaire qu'on a marqué de cet indice dans une *face_property* de *Surface_mesh* appelé *f :indiceOutline*.

intersectOctree() est une fonction qui calcule les points d'intersections entre un rayon et le modèle 3D plongé dans l'*octree*. Cette fonction est appelée par le *package* **Distance** pour la détection de fragilité. Le rayon est défini par un point d'origine *origin* et une direction *direction*. Dans le cadre de la détection de fragilité le rayon est lancé depuis la surface du modèle 3D, plus exactement le centre de chaque face. On passe en paramètre la face d'origine *faceOrigin* du rayon afin que le point d'origine ne soit pas compté comme une intersection. Les points d'intersection calculés sont transmis par le pointeur sur un tableau de *PointImpact* passé en paramètre.

B.4.2 *Node*

Node est une structure pour enregistrer les informations suivantes d'un nœud de l'*octree*.

- *Box*, instance de la classe *Box*.
- *firstNodeId*, indice dans le tableau *m_nodes* du premier enfant. Tous les enfants sont consécutifs dans le tableau.
- *nbNode*, nombre de nœuds enfant entre 0 si le nœud est une feuille et 8.
- *nbFace*, nombre de faces présentes dans le nœud.
- *leaf*, booléen indiquant si le nœud est une feuille ou non.

B.4.3 *Box*

Box est une classe qui contient des informations sur son nœud et des fonctions qui permettent la construction et l'utilisation de l'*octree*. La liste des faces qui se trouvent dans la zone de l'espace que représente le nœud est enregistrée dans un tableau *m_listFace*. La zone de l'espace que représente le nœud est enregistrée sous la forme des points minimum et maximum de cet espace $Xmin, Ymin, Zmin, Xmax, Ymax$ et $Zmax$. L'espace représenté par chaque nœud est cubique.

La première *Box* est construite autour du modèle 3D. On l'initialise avec un premier sommet *init()*. Puis on agrandit la *Box* en passant un à un tous les sommets dans la fonction *extend()*. Enfin, on rend la *Box* cubique avec la fonction *uniformBox()*. La *Box* d'un nœud enfant utilisent les fonctions *init()* et *extend()* leur passant deux points de l'espace calculé par rapport à son parent correspondant directement à son points minimum et son point maximum.

Pour compléter la liste des faces *m_listFace* on utilise la fonction *addFace()* pour la première *Box*. Les faces sont ajoutées une par une dans le même temps

où l'on passe leurs sommets dans la fonction *extend()*. Pour les niveaux supérieurs la liste des faces est calculée par le constructeur de l'*octree*, on l'enregistre donc directement avec la fonction *setListFace()*.

nbFace() est la fonction qui retourne la taille du tableau *m_ listFace*. Cette information est doublée dans la structure *Node*.

intersectBox() est une fonction qui prend en entrée l'origine et la direction d'un rayon et détermine si le rayon passe dans l'espace que représente le nœud ou non. La fonction est appelée par la fonction *instersectOctree* de *Octree*.

B.4.4 *PointImpact*

PointImpact est une classe qui permet d'enregistrer les informations des points d'intersection du rayon avec des faces du modèle 3D.

m_ distance est la distance entre le point d'origine du rayon et le point d'intersection.

setDirection() est la fonction qui calcule la valeur de *m_ direction* qui correspond à la comparaison entre la direction du rayon *rayDirection* et la normale de la face intersectée *normalFaceImpact*, pour savoir de quel côté de la face le rayon arrive et part. La valeur vaut 1, si le rayon arrive par l'avant de la face, -1 s'il arrive par l'arrière et 0 si cas exceptionnel le rayon traverse la face dans sa longueur.

On redéfinit l'opérateur inférieur en se basant sur la distance *operator<()*. Un tri du tableau des points d'impact en fonction de leur distance est effectué par l'algorithme du volume parenthésé (Algo.13).

B.5 Tools

B.5.1 *Vector2F*

La classe *Vector2F* représente un vecteur à deux dimensions. Il est principalement utilisé pour la sauvegarde et l'utilisation des *vertex texture* qui ont des coordonnées à deux dimensions sur la texture.

B.5.2 *Vector3F*

La classe *Vector3F* représente un vecteur à trois dimensions. Il est utilisé pour les sommets et comme vecteur 3D pouvant correspondre à une arête, une normale ou la direction d'un rayon. La classe comprend un ensemble de fonctions pour la manipulation de vecteur 3D, avec la redéfinition de plusieurs

opérateurs et des fonctions comme le produit vectoriel *cross()*, le produit scalaire *dot()* ou le calcule de l'angle entre deux vecteurs 3D *angle()*.

B.5.3 *Matrix3x3F*

La classe *Matrix3x3F* représente une matrice trois par trois. La classe comprend un ensemble de fonctions pour la manipulation d'une matrice trois par trois : le calcul du déterminant *déterminant()*, le calcul des matrices transposées *transpose()* et inverse *inverse()* et la multiplication par un vecteur 3D *multBy3f()*. Ces fonctions sont utilisées principalement pour le calcul d'intersection entre un rayon et une face du calcul de distance.

B.5.4 *Tools*

Tools comprend une série de fonctions de test et de calcul sur des éléments 3D. Les éléments 3D sont ceux de la bibliothèque *Surface_mesh*.

intersectCoplanarTringle() est une fonction qui teste si deux triangles coplanaires s'intersectent ou non. La fonction *intersectCoplanarTringleExclusif()* rajoute à ce test la condition que les triangles ne s'intersectent pas si la zone de collision est uniquement sur une arête ou un sommet.

intersectCoplanarSegmentExclusif() est une fonction qui teste l'intersection entre deux segments en excluant les extrémités comme zone d'intersection. Ce test est utilisé pour la fermeture d'un segment planaire dans l'algorithme de remplissage afin de vérifier que la fermeture n'intersecte pas des éléments existants.

projectionPointPlan() est une fonction qui calcule la projection d'un point sur un plan. Elle est aussi utilisée dans la fermeture d'un segment planaire dans l'algorithme de remplissage, car les tests d'intersection sont faits après une projection de tous les éléments dans le plan du segment planaire.

testColinearTriangle, *testCoplanar* et *testCoplanarTriangle* sont des fonctions test, de colinéarité et de coplanarité entre différents éléments 3D.

Plusieurs fonctions sont reprises de bibliothèque "The Computational Geometry Algorithms Library" CGAL. Les fonctions *intersectCoplanarTringle()*, *intersection_test_vertex()* et *intersection_test_edge()* sont reprises du fichier "Triangle_3_Triangle_3_do_intersect.h" crédité à Olivier Devillers et Philippe Guigue. Les fonctions *coplanar_orientation* et *orient_2d* sont reprises du fichier "Coplanar_orientation_3.h" crédité à Sylvain Pion.

B.5.5 *Logger*

Logger est un singleton utilisé par les autres packages pour transmettre des messages d'erreur ou d'information vers un fichier. Les messages sont écrits en cours d'utilisation du logiciel. Le singleton est initialisé par le *package* **Application** avant le début de la chaîne de réparation.

Bibliographie

3MFConsortium, [Online][Accessed : 14-November-2019]. 3d manufacturing format.
URL https://3mf.io/3d-manufacturing-format/

Alexander, P. et Dutta, D., 2000. Layered manufacturing of surfaces with open contours using localized wall thickening. *Computer-Aided Design*, 32(3) :175–189.

Attene, M., Campen, M. et Kobbelt, L., 2013. Polygon mesh repairing : An application perspective. *ACM Comput. Surv.*, 45(2) :15.

Attene, Marco, 2014. Direct repair of self-intersecting meshes. *Graphical Models*, 76(6) :658 – 668.

Attene, Marco, 2016. As-exact-as-possible repair of unprintable stl files. *arXiv preprint arXiv :1605.07829*.

Attene, Marco, 2018. As-exact-as-possible repair of unprintable stl files. *Rapid Prototyping Journal*, 24(5) :855–864.

Autodesk, [Online][Accessed : 14-November-2019]. Adaptable file format for 3d animation software.
URL https://www.autodesk.com/products/fbx/overview

Barequet, Gill et Sharir, Micha, 1995. Filling gaps in the boundary of a polyhedron. *Computer Aided Geometric Design*, 12(2) :207–229.

Bern, M. et Eppstein, D., 1992. Mesh generation and optimal triangulation. Dans Ding-Zhu Du et Frank Kwang-Ming Hwang, rédacteurs, *Computing in Euclidean Geometry*, numéro 1 dans Lecture Notes Series on Computing, pages 23–90. World Scientific.

Bickel, B., Bächer, M., Otaduy, M.A., Lee, H.R., Pfister, H., Gross, M.H. et Matusik, W., 2010. Design and fabrication of materials with desired deformation behavior. *ACM Trans. Graph.*, 29(4) :63 :1–63 :10.

Bischoff, S. et Kobbelt, L., 2005. Structure preserving CAD model repair. *Comput. Graph. Forum*, 24(3) :527–536.

Bohn, J.H. et Wozny, M.J., 1992. Automatic cad-model repair : Shell-closure. Dans *Solid Freeform Fabrication Symposium*, pages 86–94.

Borodin, P., Novotni, M. et Klein, R., 2002. Progressive gap closing for mesh repairing. Dans *CGI Advances in Modeling, Animation and Rendering*.

Borodin, P., Zachmann, G. et Klein, R., 2004. Consistent normal orientation for polygonal meshes. Dans *2004 Computer Graphics International (CGI 2004), 16-19 June 2004, Crete, Greece*, pages 18–25.

Bourell, David L., Beaman, Joseph J., Leu, Ming C. et Rosen, David W., 2009. A brief history of additive manufacturing and the 2009 roadmap for additive manufacturing : Looking back and looking ahead. *US–Turkey Workshop on Rapid Technologies*.

Burns, M., [Online][Accessed : 14-November-2019]. Automated fabrication : The stl format.
URL http://www.fabbers.com/tech/STL_Format

Calì, J., Calian, D.A., Amati, C., Kleinberger, R., Steed, A., Kautz, J. et Weyrich, T., 2012. 3d-printing of non-assembly, articulated models. *ACM Trans. Graph.*, 31(6) :130.

Campen, M. et Kobbelt, L., 2010. Exact and robust (self-)intersections for polygonal meshes. *Comput. Graph. Forum*, 29(2) :397–406.

Carr, J. C., Beatson, R. K., Cherrie, J. B., Mitchell, T. J., Fright, W. R., McCallum, B. C. et Evans, T. R., 2001. Reconstruction and representation of 3d objects with radial basis functions. Dans *Computer Graphics (SIGGRAPH '01 Conf. Proc.), pages 67–76. ACM SIGGRAPH*, pages 67–76. Springer.

Chen, Y. et Wang, C. L., 2011. Uniform offsetting of polygonal model based on layered depth-normal images. *Computer-Aided Design*, 43(1) :31–46.

Ciraud, P.A., 1972. Process and device for the manufacture of any objects desired from any meltable material. *FRG Disclosure Publication 2263777*.

Crump, S.S., 1989. Apparatus and method for creating three-dimensinal objects. US Patent 5,121,329.

Davis, J., Marschner, S.R., Garr, M. et Levoy, M., 2002. Filling holes in complex surfaces using volumetric diffusion. Dans *1st International Symposium on 3D Data Processing Visualization and Transmission (3DPVT 2002), 19-21 June 2002, Padova, Italy*, pages 428–438.

D.Chakravorty, [Online][Accessed : 14-November-2019]. The 4 most important 3d printer file formats.
URL https://all3dp.com/3d-printing-file-formats/

Devillers, Olivier et Guigue, Philippe, 2002. Faster Triangle-Triangle Intersection Tests. Rapport technique RR-4488, INRIA.

Eberly, D., 2002. Triangulation by ear clipping.

Farouki, Rida T., 1985. Exact offset procedures for simple solids. *Computer Aided Geometric Design*, 2(4) :257–279.

FileFormatInfo, [Online][Accessed : 14-November-2019]. Wavefront obj file format summary.
URL https://www.fileformat.info/format/wavefrontobj/egff.htm

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

Guéziec, A., Taubin, G., Lazarus, F. et Horn, W., 2001. Cutting and stitching : Converting sets of polygons to manifold surfaces. *IEEE Trans. Vis. Comput. Graph.*, 7(2) :136–151.

Herbert, A.J., 1982. Solid object generation. *Journal of Applied Photographic Engineering*, 8(4) :185–188.

Hildebrand, T. et Rüegsegger, P., 1997. A new method for the model-independent assessment of thickness in three-dimensional images. *Journal of Microscopy*, 185(1) :67–75.

H.Lipson, 2014. Amf tutorial : the basics. *3D Printing and Additive Manufacturing*, 1(2).

Hoffmann, Ch.M., 1989. *Geometric and Solid Modeling : an introduction*. Morgan Kaufmann.

Hoppe, H., DeRose, T. et Duchamp, T., 1992. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2) :71–78.

Housholder, R.F., 1981. Molding process. US Patent 4,247,508.

I.Mäkelä et A.Dolenc, 1993. Some efficient procedures for correcting triangulated models. Dans *Solid Freeform Fabrication Symposium*, pages 126–134.

Jacobson, Alec, Kavan, Ladislav et Sorkine-Hornung, Olga, 2013. Robust inside-outside segmentation using generalized winding numbers. *ACM Transactions on Graphics (TOG)*, 32(4) :1–12.

Jones, S.E., Bradley, R.B. et Aharon, I., 2000. Three-dimensional mapping of cortical thickness using laplace's equation. *Human Brain Mapping*, 11(1) :12–32.

Ju, T., 2009. Fixing geometric errors on polygonal models : A survey. *J. Comput. Sci. Technol.*, 24(1) :19–29.

KhronosGroup, [Online][Accessed : 14-November-2019]. Collada – digital asset schema release 1.5.0.
URL https://www.khronos.org/files/collada_spec_1_5.pdf

Kodama, Hideo, 1981. Automatic method for fabricating a three-dimensional plastic model with photo-hardening polymer. *Review of scientific instruments*, 52(11) :1770–1773.

Lanterne, C., Gueorguieva, S. et Desbarats, P., 2016. Local thickness computation in 3d meshes and 3d printability assessment. Dans *Multi Conference on Computer Science and Information Systems (MCCSIS), Computer Graphics, Visualization, Computer Vision and Image Processing (CGrVCVIP)*, pages 287–291. Madeira, Portugal.

Lee, S.-H., 2009. Offsetting operations on non-manifold topological models. *Computer-Aided Design*, 41 :830–846.

Liepa, P., 2003. Filling holes in meshes. Dans *First Eurographics Symposium on Geometry Processing, Aachen, Germany, June 23-25, 2003*, pages 200–205.

Lu, L., Sharf, A., Zhao, H., Wei, Y., Fan, Q., Chen, X., Savoye, Y., Tu, C., Cohen-Or, D. et Chen, B., 2014. Build-to-last : strength to weight 3d printed objects. *ACM Trans. Graph.*, 33(4) :97 :1–97 :10.

Luo, L., Baran, I., Rusinkiewicz, S. et Matusik, W., 2012. Chopper : partitioning models into 3d-printable parts. *ACM Trans. Graph.*, 31(6) :129.

Maekawa, Takashi, 1999. An overview of offset curves and surfaces. *Computer-Aided Design*, 31(3) :165–173.

Marsan, Anne L., Kumar, Vinod, Dutta, Debasish, Pratt, Michael J., Daley, William M. et Kammer, Raymond G., 1998. Nistir 6216 an assessment of data requirements and data transfer formats for layered manufacturing.

Munz, O.J., 1956. Photo-glyph recording. US Patent 2,775,758.

Murali, T.M. et Funkhouser, T.A., 1997. Consistent solid boundary representation from arbitrary polygonal data. Dans *Symposium on Interactive 3D Graphics*, pages 1–9.

Nooruddin, F.S. et Turk, G., 2003. Simplification and repair of polygonal models using volumetric techniques. *IEEE Trans. Vis. Comput. Graph.*, 9(2) :191–205.

Patel, Paresh S., Marcum, David L. et Remotigue, Michael G., 2005. Stitching and filling : Creating conformal faceted geometry. Dans Byron W. Hanks, rédacteur, *Proceedings of the 14th International Meshing Roundtable*, pages 239–256. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-29090-2.

Pfeifle, R. et Seidel, H.P., 1996. Triangular b-splines for blending and filling of polygonal holes. Dans *Proceedings of the Conference on Graphics Interface '96*.

Pham, Binh, 1992. Offset curves and surfaces : a brief survey. *Computer-Aided Design*, 24(4) :223–229.

Podolak, J. et Rusinkiewicz, S., 2005. Atomic volumes for mesh completion. Dans *Third Eurographics Symposium on Geometry Processing, Vienna, Austria, July 4-6, 2005*, pages 33–41.

Porter, T. et Duff, T., 1984. Compositing digital images. Dans *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1984, Minneapolis, Minnesota, USA, July 23-27, 1984*, pages 253–259.

Prévost, R., Whiting, E., Lefebvre, S. et Sorkine-Hornung, O., 2013. Make it stand : balancing shapes for 3d fabrication. *ACM Trans. Graph.*, 32(4) :81 :1–81 :10.

Qu, X. et Stucker, B., 2003. A 3d surface offset method for stl-format models. *Rapid Prototyping Journal*, 9(3) :133–141.

Rock, S.J. et Wozny, M.J., 1992. Generating topological information from a "bucket of facets". Dans *Solid Freeform Fabrication Symposium*, pages 251–259.

Rossignac, J. et Cardoze, D. E., 1999. Matchmaker : manifold breps for non-manifold r-sets. Dans *Symposium on Solid Modeling and Applications*, pages 31–41.

Rossignac, J. et Requicha, A. A. G., 1986. Offsetting operations in solid modelling. *Computer Aided Geometric Design*, 3(2) :129–148.

Roth, G. et Wibowoo, E., 1997. An efficient volumetric method for building closed triangular meshes from 3-d image and point data. Dans *Graphics Interface*, pages 173–180.

Sachs, E., Cima, M. et Cornie, J., 1990. Three-dimensional printing : Rapid tooling and prototypes directly from a cad model. *CIRP Annals*, 39(1) :201 – 204.

Sachs, E.M., Haggerty, J.S., Cima, M.J. et Williams, P.A., 1989. Three-dimensional printing techniques. US Patent 5,204,055.

Sharf, A., Alexa, M. et Cohen-Or, D., 2004. Context-based surface completion. *ACM Trans. Graph.*, 23(3) :878–887.

Sheng, X. et Meier, I.R., 1995. Generating topological structures for surface models. *IEEE Computer Graphics and Applications*, 15(6) :35–41.

Stava, O., Vanek, J., Benes, B., Carr, N.A. et Mech, R., 2012. Stress relief : improving structural strength of 3D printable objects. *ACM Trans. Graph.*, 31(4) :48.

Swainson, W.K., 1977. Method,medium and apparatus for producing three-dimensional figure product. US Patent 4,041,476.

Thibault, W. et Naylor, B., 1987. Set operation on polyhedra using binary space partitioning trees. 21 :153–162.

Turk, G. et Levoy, M., 1994. Zippered polygon meshes from range images. Dans *Proceedings of the 21th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1994, Orlando, FL, USA, July 24-29, 1994*, pages 311–318.

Umetani, N. et Schmidt, R., 2013. Cross-sectional structural analysis for 3d printing optimization. Dans *SIGGRAPH Asia 2013, Hong Kong, China, November 19-22, 2013, Technical Briefs*, pages 5 :1–5 :4.

Varnuska, M., Parus, J. et Kolingerova, I., 2005. Simple holes triangulation in surface reconstruction. Dans *Algoritmy*, pages 280–289.

Wagner, M., Labsik, U. et Greiner, G., 2003. Repairing non-manifold triangle meshes using simulated annealing. *International Journal of Shape Modeling*, 9(2) :137–154.

Wang, C. C. L. et Manocha, D., 2013. Efficient boundary extraction of bsp solids based on clipping operations. *IEEE Transactions on Visualization and Computer Graphics*, 19(1) :16–29. doi :10.1109/TVCG.2012.104.

Wang, Ch.C.L. et Chen, Y., 2013. Thickening freeform surfaces for solid fabrication. *Rapid Prototyping Journal*, 19(6) :395–406.

Wang, W., Wang, T.Y., Yang, Z., Liu, L., Tong, X., Tong, W., Deng, J., Chen, F. et Liu, X., 2013. Cost-effective printing of 3d objects with skin-frame structures. *ACM Trans. Graph.*, 32(6) :177 :1–177 :10.

Web3D, [Online][Accessed : 14-November-2019]. Extensible 3d (x3d). URL http://www.web3d.org/documents/specifications/19775-1/V3.3/index.html

Zhou, Q., Panetta, J. et Zorin, D., 2013. Worst-case structural analysis. *ACM Trans. Graph.*, 32(4) :137 :1–137 :12.