Maillage à dominante Polycube

François Protais

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Maillage à dominante Polycube

THÈSE

présentée et soutenue publiquement le 21 octobre 2022

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

François Protais

Composition du jury

Président :	Frédéric Alauzet	– Directeur de recherche Inria
Rapporteurs :	Julie Digne	– Chargée de recherche au CNRS
	Jeroen Wackers	– Ingénieur de recherche à l'École Centrale de Nantes
Examinateur :	Dominique Bechmann	– Professeure à l'Université de Strasbourg
Invité :	Nicolas Ray	– Chargé de recherche Inria
Encadrants :	Dmitry Sokolov	– Maître de conférences à L'Université de Lorraine
	Franck Ledoux	– Directeur de recherche au CEA DAM

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Résumé

Cette thèse étudie les méthodes de polycubes pour la génération de maillages hexaédriques. Ces maillages, et plus généralement les maillages structurés par blocs, sont très recherchés pour effectuer des simulations numériques de phénomènes physiques (fission nucléaire, écoulement, aérospatiale...). Or, il n'existe pas de méthode industriellement viable pour les générer. Nous étudions une famille de méthodes très prometteuses pour combler ce vide : les méthodes de paramétrisation globales. À ce jour, il subsiste de nombreux problèmes de robustesse, que nous essayons de corriger. Pour cela, nous nous focalisons sur la sous-famille des méthodes de polycubes.

Pour obtenir un maillage hexaédrique d'un domaine avec des polycubes - amas de cubes unités -, la procédure est la suivante : le bord du domaine est colorié, puis déformé en fonction de ces couleurs pour avoir son bord aligné avec les axes, ce qui en fait un *polycuboid*. Ce *polycuboid* est ensuite intersecté avec une grille pour donner un polycube. Appliquer la déformation inverse sur ce polycube nous donne un maillage hexaédrique.

Nous apportons deux contributions majeures à cette procédure : d'abord une méthode qui permet de calculer la déformation efficacement et avec des garanties que celle-ci sera de bonne qualité. Ensuite, nous introduisons une nouvelle méthode robuste pour effectuer les étapes d'intersection avec la grille et d'inversion de la déformation. Nous présentons enfin des pistes d'études pour l'étape difficile de la coloration, dernière brique nécessaire pour avoir des méthodes robustes de génération de maillages hexaédriques à base de polycubes.

Mots-clés: Maillage, hexaèdres, déformation, Polycube.

Abstract

This thesis studies polycube methods for the generation of hexahedral meshes. These meshes, and more generally block structured meshes, are very much in demand for numerical simulations of physical phenomena (nuclear fission, flow, aerospace...). However, there is no industrially viable method to generate them. We are studying a family of methods that show great promise in filling this gap: global parametrization methods. To date, there are still many robustness problems, which we try to correct. For this purpose, we focus on the subfamily of polycube methods.

To obtain a hexahedral mesh of a domain with polycubes - clusters of unit cubes -, the procedure is the following : the boundary of the domain is colored, then deformed according to these colors to have its boundary aligned with the axes, which makes it a *polycuboid*. This *polycuboid* is then intersected with a grid to give a polycube. Applying the inverse deformation on this polycube gives us a hexahedral mesh.

We make two major contributions to this procedure : first, a method that allows to compute the deformation efficiently and with guarantees that it will be of good quality. Second, we introduce a new robust method to perform the grid intersection and deformation inversion steps. Finally, we present avenues of study for the difficult step of coloring, the last brick necessary to have robust methods for generating polycube-based hexahedral meshes.

Keywords: Meshing, hexahedra, deformation, Polycube.

Remerciements

Avant tout, je tiens à remercier Dmitry et Franck pour la direction de ma thèse et de m'avoir accordé une telle opportunité de travailler avec eux. Dmitry, de m'avoir guidé dans le labyrinthe du LORIA, avec ton approche profondément humaine, et pour toutes les découvertes et idées que nous avons pu avoir ensemble. Franck, de m'avoir offert ces moments de pause au CEA, où tu accueillais toujours patiemment mon fort enthousiasme. Tes conseils étaient toujours justes et me permettaient de prendre du recul sur mes travaux.

Nicolas, tu mériterais de te trouver avec les deux précédents. J'ai tant appris de nos échanges. Que ce soit de sciences, comment faire cette science, ou même de relations humaines. Je te remercie pour cette direction non officielle. Je regretterai nos cafés matinaux...

Je souhaite ensuite remercier les membres du Jury. Julie Digne et Jeroen Wackers pour leur rapport d'excellente qualité, ainsi que Dominique Bechmann et Frederic Alauzet de m'avoir fait l'honneur d'accepter nos invitations.

Mes remerciements vont ensuite à tous les membres de l'équipe Pixel. D'abord les permanents Étienne, Dobrina et Laurent, auxquels je pense pouvoir rattacher Bruno, Ben et Cédric. Et ensuite, partageant plus ma réalité, mes co-doctorants, Justine, Guillaume, David D, David L et Yoann. Merci pour le cadre que vous avez permis de construire, pour les moments de stress que nous avons eu ensemble, pour les moments de bonheur, et surtout le fromage ! Merci aussi à mes stagiaires qui Leila et Anna avec lesquels j'ai éprouvé beaucoup de plaisir à travailler. Il est aussi important pour moi de remercier ceux qui ont rendus cette thèse d'autant plus agréable : Céline et Emmanuelle pour leur support et leur patience pour toutes mes requêtes, ainsi que Caro et Floriane d'avoir égaillé mes journées par leur joie et leur bonne humeur.

Je remercie aussi ma seconde équipe, présente au CEA. Simon d'avoir toujours été présent pour m'accueillir dans ce monde changeant, nous voilà maintenant tous les deux libérés ! Claire d'avoir partagé ma réalité de membre intermittent, merci pour toutes les discussions intéressantes. Corentin je te remercie pour toutes les visios que nous avons pu avoir, nous avons même pu nous voir avant que je soutienne ! Et enfin merci Nicolas, Valentin, Paul, Soufiane, Christophe et Sébastien pour les échanges que nous avons pu avoir.

Un grand merci à Marco, Gianmarco, Luca, et tout le reste de l'équipe d'Unica pour leur accueil chaleureux. Merci à Maxence pour tous les échanges très intéressants que nous avons eus, et de m'avoir accueilli à Lyon. Merci à Vladimir de m'avoir fait part de ses connaissances et de son expérience. Merci à Scott et à l'équipe maillage de STAR-CCM+ pour leur accueil chaleureux.

Je tiens enfin à remercier ceux qui n'ont pas forcément participé scientifiquement, mais qui ont pour autant été si précieux pour moi lors de ces trois années. Merci à Pierre, Noé et Olivier d'avoir partagé mon quotidien. Merci à tous les GM que j'ai toujours eu le plus grand plaisir à rencontrer. Un merci particulier à Victor et Ana de m'avoir honoré de leur présence pour ma soutenance. Vient ensuite toute ma famille que je remercie pour son support. Merci tout particulièrement à Père-Grand et Mère-Grand pour leur soutien inconditionnel. Merci Laëtitia d'avoir supporté toutes ces maths pour ton grand frère ! Une pensée à Annie qui a sans aucun doute contribué à ce que j'arrive jusque-là, en nourrissant ma curiosité scientifique.

Je remercie enfin mes parents, qui m'ont infailliblement accueilli, écouté, rassuré, au cours de mes longues années d'études. Et bien sûr Mirella d'avoir fait ce chemin avec moi, et d'avoir été auprès de moi pour les hauts comme pour les bas (et pour les cookies aussi).

De tout temps
l'Homme a voulu faire des cubes...

Table des matières

Introduction

La simulation numérique est devenue un outil incontournable du développement de nouvelles technologies. Plus polyvalente, moins coûteuse, plus sûre, elle vient compléter, et parfois même remplacer, l'expérimentation pratique. Pour effectuer de telles simulations, deux composantes sont nécessaires : un, l'ordinateur, support de calcul, permettant de rapidement faire des opérations numériques, et deux, les logiciels et méthodes mathématiques, outils de prédiction des phénomènes complexes de la nature.

Dans cette thèse, nous abordons un des problèmes ouverts des méthodes de la simulation numérique : la génération de maillages hexaédriques structurés par blocs. Toute expérience physique se produit dans un espace, un domaine, et ce domaine nécessite une certaine représentation pour être stocké et manipulé par l'ordinateur et les méthodes de prédiction mathématiques. Dans certaines applications, comme l'avionique, l'aérospatiale, ou la physique nucléaire, une représentation particulièrement appréciée est le maillage hexaédrique structuré par blocs, illustré sur la Figure 1. Or, à ce jour, il n'existe pas de méthodes ou de logiciels permettant de générer de tels maillages automatiquement pour des domaines quelconques en dimension 3.

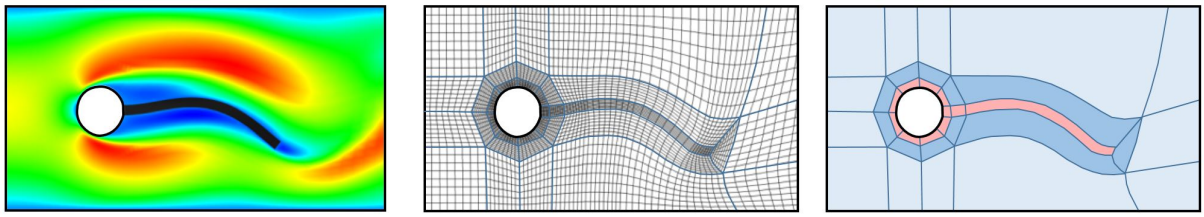


Figure 1 – Certaines simulations d'écoulement (à gauche) nécessitent une représentation particulière des domaines. Deux critères récurrents sont l'alignement particulier de la discrétisation couplé avec une certaine résolution (au milieu), proche de phénomènes physiques intéressants. Une décomposition en blocs (à droite) permet à l'utilisateur d'avoir un contrôle précis sur ces critères, pour, par exemple, éviter d'avoir une résolution élevée loin des phénomènes, qui entraînerait un surcoût calculatoire.

Cette absence de méthode automatique est un problème majeur : les entreprises doivent alors investir un temps ingénieur conséquent pour faire réaliser ces maillages manuellement. En 2008, le *Departement of Energy* américain soulignait que 73% du temps nécessaire pour simuler une centrale nucléaire était alloué à la génération du maillage [Hansen et Owen, 2008]. Cela fait en pratique bientôt 30 ans que ce problème 3D est un sujet actif de recherche. Il le fut d'abord pour la communauté du calcul scientifique, naturellement, mais aucune approche n'a permis de le résoudre. Au début des années 2000, une idée très prometteuse est apparue dans la communauté de l'infographie : utiliser des atlas de texture contraints pour extraire des maillages. L'idée, élégante, a permis d'obtenir d'excellents résultats pour la génération de

maillage quadrangulaire 2D. Il est même assez couramment admis que le problème 2D de maillage quadrangulaire structuré par blocs est, sauf certains détails, résolu. Cela est notamment dû aux travaux récents de David Bommes et Marcel Campen [Bommes *et al.*, 2013, Campen *et al.*, 2015, Lyon *et al.*, 2019]. En 3D, le problème reste ouvert.

Les approches à base de texture, aussi appelées méthodes de paramétrisation globales, ont montré de très bons résultats préliminaires, laissant espérer qu'elles permettront de résoudre le problème. Malheureusement, elles sont, pour le moment, sujettes à des problèmes de robustesse les rendant inutilisables industriellement. Cette thèse s'attaque à cet ensemble de problèmes. Nous privilégions l'étude d'un sous-ensemble de ces méthodes : les méthodes dites de Polycubes. L'objectif est de rendre viable cette approche en comblant ses lacunes. Celles-ci étant similaires à celle des paramétrisations globales, l'étape suivante sera d'étendre nos solutions dans le cadre général.

Au début de mon travail de doctorat, il existait un ensemble de verrous rendant difficile l'utilisation des polycubes pour la génération de maillages hexaédriques structurés par blocs. Nos travaux ont permis de réduire ceux-ci à un unique problème : la coloration, *i.e.* l'assignation du bord du domaine vers chacune des faces du polycube (voir FIGURE 2-a). En considérant que nous disposons d'une coloration valide, nous avons développé un ensemble d'outils permettant de produire une procédure robuste de génération de maillages à partir de cette coloration. Deux aspects prometteurs de nos travaux sont :

1. les outils développés semblent extensibles, ou sont même directement utiles, aux méthodes de paramétrisations globales ;
2. la recherche pour les méthodes de polycubes peut entièrement se focaliser sur la génération de bonnes colorations, nos travaux apportant des solutions fiables aux étapes ultérieures (voir FIGURE 2-b–c–d).

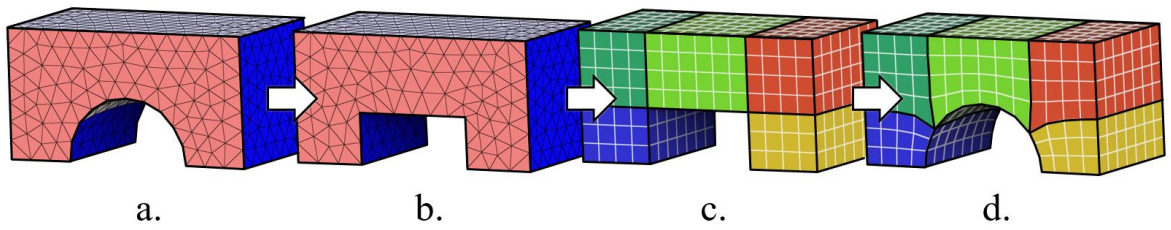


FIGURE 2 – Générer des maillages hexaédriques avec des polycubes se fait en 4 étapes : (a.) le domaine est d'abord coloré, (b.) et cette coloration permet de le déformer en polycuboid. (c.) Intersecter ce polycuboid avec une grille donne un polycube (amas de cube). (d.) Appliquer la déformation inverse sur le polycube nous donne un maillage de notre domaine d'origine, possédant une structure de bloc.

Nous structurons ce document en deux grandes parties : tout d'abord une introduction détaillée sur le maillage hexaédrique et les polycubes, puis une présentation des contributions de ce doctorat. La première partie est composée de deux chapitres : le chapitre 1 introduit les concepts de simulation et de maillages hexaédriques, en discutant des différentes variantes, avec un court aperçu des méthodes utilisées pour les générer. Le chapitre 2 se focalise sur les méthodes de paramétrisations globales et le sous-ensemble des polycubes. Nous y détaillons leur fonctionnement précis, ainsi que tous les problèmes rencontrés, qui font l'objet d'une active recherche scientifique [Pietroni *et al.*, 2022].

La partie consacrée aux contributions est scindée en trois chapitres. Nous abordons d'abord, dans le chapitre 3, une nouvelle méthode robuste [Garanzha *et al.*, 2021a] de construction de cartes et de déformations de bonne qualité. C'est un outil vital pour les méthodes de polycubes, voir FIGURE 2-b., mais aussi pour un grand nombre d'autres applications dont nous donnons des exemples. Le chapitre 4 montre ensuite une méthode robuste [Protais *et al.*, 2022] pour générer un maillage hexaédrique depuis une déformation réussie, voir FIGURE 2-c. Le chapitre 5 détaille enfin nos travaux sur la création de bonnes colorations, pour compléter le processus. Nous ne donnons pas de solution à ce problème, mais nous étudions deux méthodes, ouvrant la porte à des recherches futures : une première passant par une meilleure compréhension du bord des objets, et une seconde cherchant à relaxer les contraintes associées à cette coloration.

Le contenu de cette thèse a donné lieu à 4 publications dans des revues et conférences internationales à comité de lecture. [Garanzha *et al.*, 2021a, Garanzha *et al.*, 2021b] pour le chapitre 3, [Protais *et al.*, 2022] pour le chapitre 4 et [Desobry *et al.*, 2021] pour le chapitre 5. Les chapitres 3 et 5 sont aussi chacun sources de travaux en cours de soumission [Garanzha *et al.*, 2022, Dumery *et al.*, 2022]. Chaque chapitre possède une structure comparable aux travaux publiés : ils contiennent un état de l'art permettant de précisément situer le sujet d'étude, des comparaisons poussées par rapport aux travaux similaires, ainsi qu'une discussion sur les améliorations futures.

Première partie

Introduction au maillage Hexaédrique avec des Polycubes

Chapitre 1

Simulation et maillage Hexaédrique

Cette thèse a pour objectif de construire des méthodes de génération automatique d'hexaèdres. En particulier, nous cherchons à obtenir des maillages hexaédriques utiles pour la simulation numérique de phénomènes physiques. Commençons par éclaircir ces concepts. Dans ce chapitre, nous débutons par une clarification du besoin de la physique en maillages, pour simuler et comprendre le réel. Ensuite, nous introduisons les différentes formes de maillages, dont celle qui nous intéresse tout particulièrement : les maillages structurés, avec leur grand représentant, l'hexaèdre. Enfin, nous présentons les différentes approches développées jusqu'à ce jour, avec leurs défauts, qui nous permettent de surligner les difficultés présentes en génération de maillages hexaédriques.

1.1 Représentation du réel dans l'ordinateur

Cette section motive l'utilisation de maillages. Nous commençons par mentionner les besoins en simulation de l'industrie. Nous introduisons ensuite les EDP, outils mathématiques de référence pour réaliser ces simulations. Nous parlons enfin des similitudes avec la synthèse d'images, d'où viennent les méthodes que nous étudions durant cette thèse.

1.1.1 Outil de calcul : "Computer" = calculateur

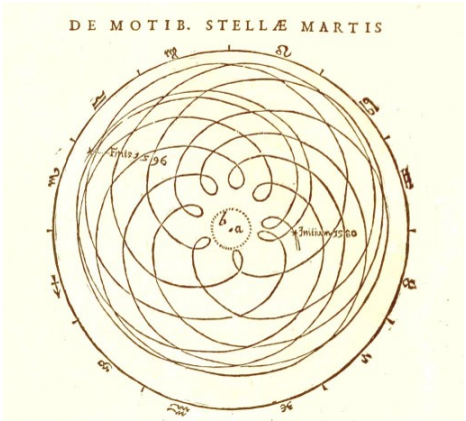


FIGURE 1.1 – Représentation des mouvements de Mars avec un point de vue géocentrique par Johannes Kepler, Astronomia nova (1609).

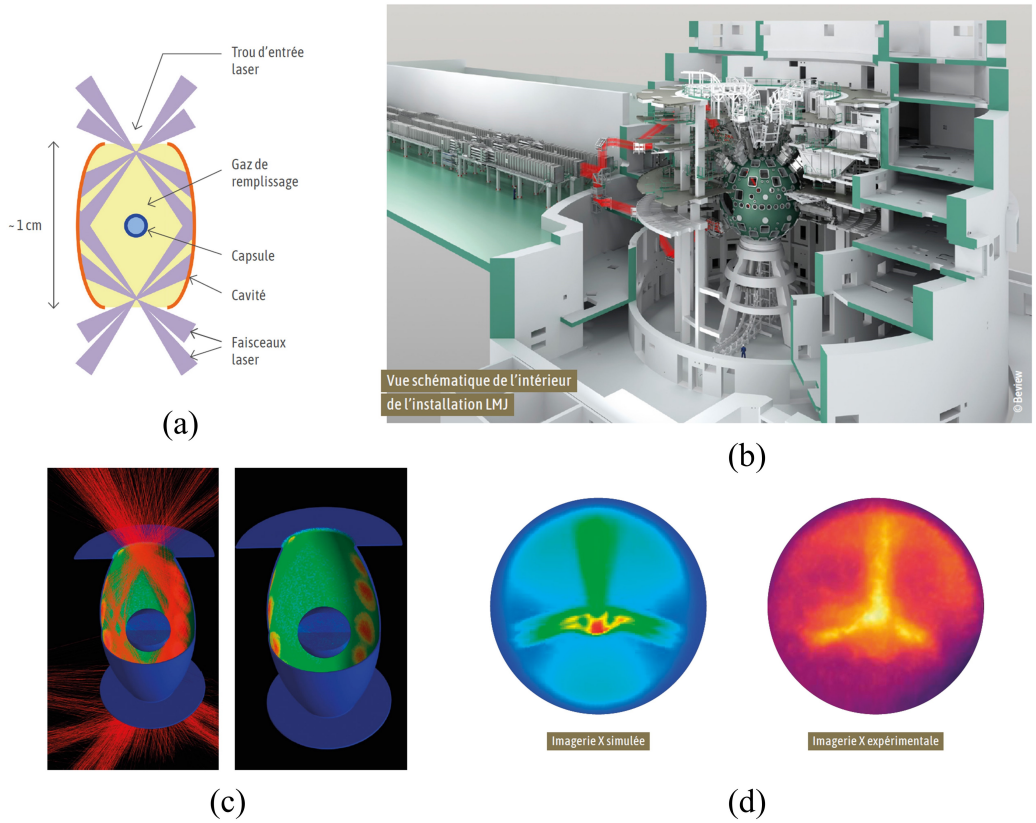


FIGURE 1.2 – Experience de fusion nucléaire réalisée au Laser MégaJoule (LMJ) au CEA Cesta (extrait de http://www-lmj.cea.fr/docs/2020/Plaquette-LMJ-fusion.pdf). (a) Schéma de l'expérience physique étudiée, (b) installation physique permettant de réaliser l'expérience, (c) simulation effectuée sur le supercalculateur Tera 1000-2 et (d) comparaison entre l'expérience physique et la simulation.

La démarche scientifique peut se résumer à « observer, comprendre, prédire ». Cette approche se retrouve dès l'antiquité chez les savants pour, par exemple, prédire les mouvements des astres. En observant les astres, ils déduisent des lois permettant d'en expliquer le mouvement (voir FIGURE 1.1), et enfin pour connaitre la position d'un astre à un instant donné, il est possible d'effectuer une application numérique de ces lois.

L'étude des astres fut un grand moteur de recherche scientifique, car les données sont facilement accessibles, et les phénomènes relativement simples (orbite en ellipse), ce qui permet de bonnes prédictions. Pour autant, lorsque l'on essaie d'appliquer les lois Newtoniennes aux astres du système solaire, pour retrouver les lois empiriques, le problème devient si complexe (Problème à N-corps), que l'on ne peut espérer trouver des solutions exactes à l'aide de calculs manuels.

La recherche en physique recèle de théories dont les modèles expliquent très bien les observations réelles, mais pour lesquelles toute prédiction est extrêmement complexe et coûteuse en calculs. L'exemple le plus en lien avec cette thèse est la simulation pour la dissuasion nucléaire effectuée au CEA, depuis l'abandon des essais nucléaires. Les situations étudiées mêlent des échelles de phénomènes très différentes (physique nucléaire et physique des écoulements), et nécessitent une précision extrême (voir FIGURE 1.2).

Cela requiert le développement de méthodes de prédictions mathématiques très développées, mais aussi des capacités de calculs considérables pour être capable d'effectuer des prédictions. On peut observer dans ce sens l'évolution de la puissance des ordinateurs utilisés par le CEA depuis l'abandon des essais nucléaires : 5 TeraFlops [1] en 2001, 60 en 2005, 1250 en 2010, et enfin 25000 en 2017 avec Tera1000-2. De la même façon, des phénomènes comme le décollage d'une fusée pour la NASA, ou l'étude de la météo pour Météo France, requièrent des capacités de calcul très importantes, comme en témoignent les entités présentes dans le classement des plus grands supercalculateurs [2].

1.1.2 Phénomène complexe et EDP.

Les phénomènes physiques sont, le plus souvent, modélisés par des équations aux dérivées partielles (EDP), qui vont régir l'évolution du système étudié. Ces équations sont des objets sources d'une quantité considérable d'études et de recherches en mathématiques. On en étudie les propriétés, l'existence de solution, et, pour ce qui nous concerne, le calcul de solution numérique. L'équation la plus étudiée, et peut-être la plus simple, est l'équation de la chaleur :

$$\frac{\partial u}{\partial t} = \Delta u$$

où u représente la chaleur à chaque emplacement de notre espace, $\frac{\partial u}{\partial t}$ sa variation en temps, et Δu sa variation en espace.

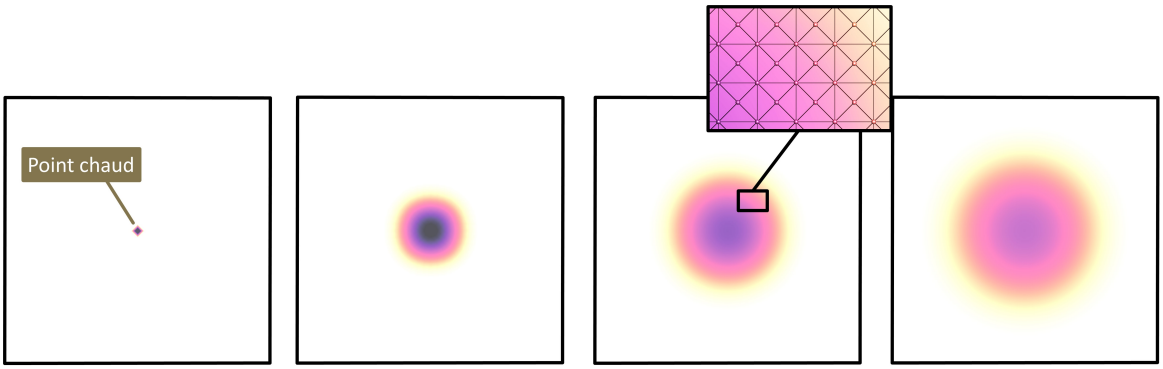


FIGURE 1.3 – Simulation d'équation de la chaleur sur un carré unité.

Trouver les solutions de cette équation est le premier exercice donné aux étudiants en simulation numérique. Sur un carré unité, représentant une plaque, simuler l'évolution de la chaleur en fonction du temps. Un exemple d'une telle simulation est illustré sur la FIGURE 1.3. L'idée est de décomposer le domaine en petits éléments, dans l'exemple de la FIGURE 1.3, ce sont des triangles mis en évidence dans l'encadré, et sur ces éléments de définir une base de fonction. Il faut ensuite résoudre un système d'équations pour trouver une fonction discrète $\tilde{u}$, somme pondérée de notre base de fonction, qui résout le mieux notre équation. Les éléments finis, méthodes usuelles pour simuler des équations de la chaleur, utilisent une base de fonctions composée de fonctions dites chapeaux en chaque sommet. Une fonction donne la valeur en son sommet, et est nulle au niveau de tous les autres. Cette méthode est donc fortement liée aux triangles représentant le domaine.

1. 10^{12} opérations par seconde
2. https://www.top500.org/

En pratique, les domaines qui intéressent les physiciens sont bien plus complexes qu'un carré, par exemple la simulation de fusion du Laser MégaJoule (voir FIGURE 1.2). De la même façon, les équations étudiées peuvent être bien plus complexes, comme l'équation de Navier-Stokes, qui régit les fluides. Celles-ci peuvent même nécessiter des représentations propres au domaine (voir section suivante). Tout cela implique un temps important alloué à la préparation des domaines. En 2014 [3], la NASA considère que la génération de bonnes représentations de ces domaines était l'un des principaux goulots d'étranglement lors de simulation de phénomènes à grandes échelles. Cette thèse s'inscrit dans l'ensemble des travaux dont l'objectif est de simplifier ce processus.

1.1.3 Des problèmes communs avec la synthèse d'images

FIGURE 1.4 – Représentation d'une carte de texture d'un objet 3D, extrait de *Least Squares Conformal Maps for Automatic Texture Atlas Generation* [Lévy *et al.*, 2002].

Les domaines représentés dans l'ordinateur ne sont pas uniquement des outils de calcul. Les objets en 3 dimensions ont très vite été utilisés pour l'animation et la création de scènes virtuelles. De nos jours, ceux-ci sont omniprésents : création d'effets spéciaux ; jeux vidéo ; réalités augmentée et virtuelle. Ils permettent de simplifier la conception d'objets (assistée par ordinateur), ou encore le travail des architectes.

Les méthodes et les types de représentation de ces objets sont très similaires à celles employées pour générer des domaines de calculs. De façon intéressante, le domaine de l'infographie, source des moteurs de représentation des objets 3D, et le calcul scientifique ont évolué de façon majoritairement indépendante. Ce n'est que récemment que des ponts ont été construits entre les différentes techniques des deux domaines. Tout particulièrement, l'utilisation de méthodes de paramétrisations globales (outil de l'infographie) pour générer des maillages hexaédriques (besoin du calcul scientifique) est due à ce rapprochement. Nous détaillons ces méthodes en détail dans le chapitre 2.

L'histoire de l'équipe Pixel, dans laquelle s'est déroulé mon doctorat, est étroitement liée à ce rapprochement. L'équipe Pixel a récemment pris la suite de l'équipe Alice. Cette équipe Alice, sous la direction de Bruno Lévy, fut un chef de file introduisant de nombreuses méthodes qui ont permis le développement des méthodes de paramétrisations globales : LCSM [Lévy *et al.*, 2002], illustré FIGURE 1.4, PGP [Ray *et al.*, 2006], Geometry aware direction fields [Ray *et al.*,

3. https://ntrs.nasa.gov/citations/20140003093

2009], etc. Ces méthodes ont d'abord eu impact dans la communauté de l'infographie, et ensuite, de nos jours, dans la communauté du calcul scientifique. Signe du rapprochement récent entre les communautés, l'équipe Alice avait été créée avec l'infographie comme principal objectif, son héritière l'équipe Pixel souligne la simulation numérique comme un de ses axes majeurs de développement. Localement, cela constitue un cycle, Bruno Lévy ayant fait sa thèse à l'École nationale supérieure de géologie de Nancy, sur des outils pour la simulation géologique, avant de s'orienter vers l'infographie.

1.2 Maillages et représentation structurée

1.2.1 Construction du domaine

Les représentations nécessaires pour les diverses simulations physiques sont très variées. En général, les domaines représentent des objets 3D, parfois composés de plusieurs parties, dans leur environnement. On peut distinguer deux principales méthodes d'obtention de ces domaines.

1. *Acquisition du réel*
   Travailler sur un objet déjà existant implique de "numériser" cet objet. Il existe de nombreuses approches de numérisation du réel, par exemple les caméras LIDAR [Collis, 1970] ou la photogrammétrie [Linder, 2009]. Deux exemples de l'utilisation de ces méthodes sont l'étude topographique, et la vérification de conformité de pièces mécaniques produites. Ces acquisitions sont le plus souvent représentées sous forme de nuages de points.
2. *Modélisation assistée par ordinateur (CAO)*
   Pour avoir plus de liberté sur la forme des objets, les physiciens peuvent confectionner les pièces à l'aide de logiciel de CAO comme Solidworks [Dassault Systèmes, 2021], Maya [Autodesk, INC., 2019] ou même Blender [Blender Foundation, 2022] en infographie. Ces modèles sont donc représentés à l'aide de courbes et de surfaces paramétriques qui permettent de créer les formes désirées.

Dans cette thèse, nous utiliserons principalement des domaines issus de la seconde catégorie, souvent de meilleure qualité (sans bruit). Ils représentent fidèlement la volonté de la personne réalisant la simulation. Avec cette fidélité, le physicien voudra probablement représenter son objet d'une manière particulière pour procéder à sa simulation.

1.2.2 Petite brique pour représenter le volume

Dans le cadre de la simulation numérique, l'objectif d'un modèle de CAO est de représenter un domaine d'étude de type pièce mécanique. Or, les méthodes de représentation pratiques pour créer des objets ne sont pas celles privilégiées pour la simulation. Les approches de simulation les plus courantes, méthode des éléments finis et méthode des volumes finis, reposent sur une décomposition des domaines en petits éléments.

La forme la plus simple, et la plus courante, est le simplexe : triangle en 2D et tétraèdre en 3D. C'est un élément convexe, et dont les propriétés ont été énormément étudiées pour leurs caractères d'éléments "minimaux". De nos jours, leurs propriétés sont bien comprises, et il existe des algorithmes robustes et efficaces pour générer des maillages simpliciaux. Nous pouvons mentionner les logiciels open-sources GMSH [Geuzaine et Remacle, 2009] et Tetgen [Si, 2015], et le logiciel propriétaire 3D Precise Mesh [Dassault Systèmes, 2022] (successeur de MeshGems). Ces méthodes sont d'une telle qualité, que nous travaillerons directement avec des maillages simpliciaux pour représenter les modèles de CAO.

1.2.3 Structure dans un maillage.

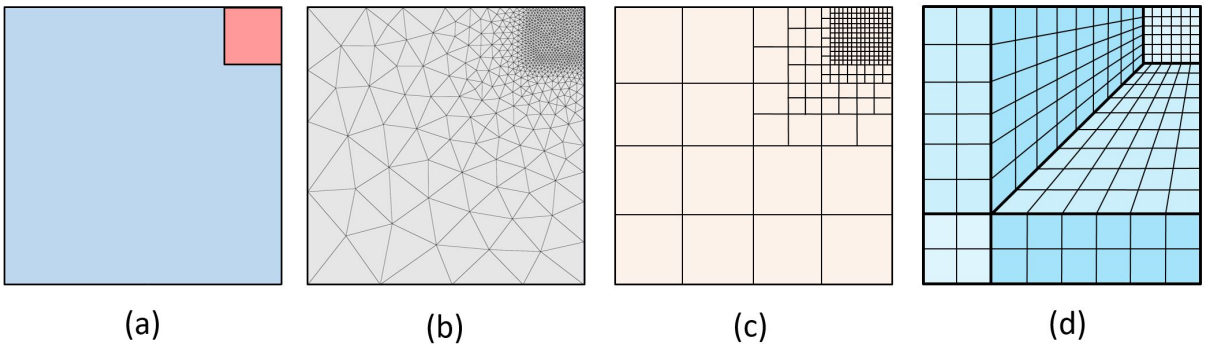
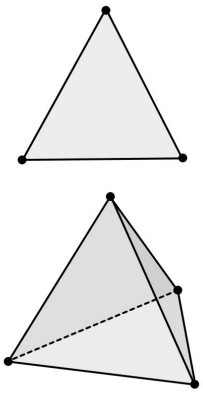


FIGURE 1.5 – (a) Domaine de simulation avec en rouge une partie nécessitant une plus forte résolution, (b) maillage triangulaire anisotropique, (c) Maillage *AMR*, (d) maillage quadrangulaire structuré.



Les propriétés recherchées dans un maillage pour une application spécifique peuvent varier considérablement. La FIGURE 1.5-(a) montre un exemple où l'on cherche un maillage plus fin dans une partie particulière du domaine. Cela arrive lorsqu'une partie du domaine nécessite une plus forte précision, sans que cet ajout de précision pénalise trop fortement le temps de calcul. Pour satisfaire cette contrainte, différentes solutions de maillage peuvent être envisagées, comme illustrées sur la FIGURE 1.5 avec un maillage triangulaire (a), un maillage *AMR* (Raffinement Adaptatif de Maillage) (b), et enfin un maillage quadrangulaire structuré (c).

Pour certaines applications, une régularité dans le maillage est fondamentale pour bien capturer les phénomènes. Un exemple omniprésent est l'étude de l'aérodynamisme des avions, où le phénomène clé se situe très proche de la paroi de l'avion. Il nécessite donc un maillage très régulier et très fin le long de la coque. Nous pouvons aussi mentionner l'expérience du laser MégaJoule (voir FIGURE 1.2), qui nécessite une certaine structure (voir FIGURE 1.6), pour pouvoir capturer l'évolution extrême des matériaux.

En général, les simplexes ne permettent pas une telle régularité. Les physiciens se tournent alors vers des maillages hexaédriques structurés par blocs, car les blocs leur permettent d'obtenir la structure nécessaire, et les éléments à l'intérieur de ces blocs conservent la régularité du bloc original : l'intérieur d'un bloc est tout simplement une grille. Il n'en demeure pas moins que la gestion de ces blocs de façon automatique est un sujet très complexe. Nous détaillons cela dans la section suivante.

1.3 Premiers pas pour la génération automatique de maillages hexaédriques et problématiques

Il est important de remarquer que, bien que ce qui nous intéresse soit les maillages hexaédriques structurés, il est possible d'en générer des non structurés. Nous commençons dans une première section par présenter les méthodes dites “non structurées", puis dans une seconde section nous introduisons les approches "structurées". Nous finissons par un tableau récapitulatif

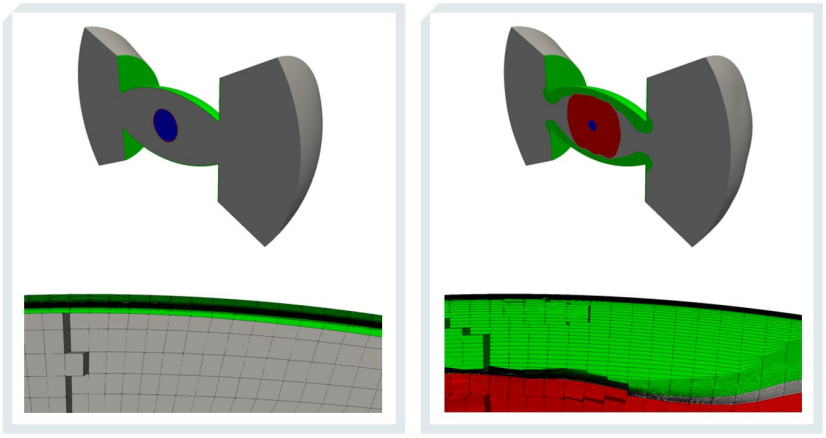
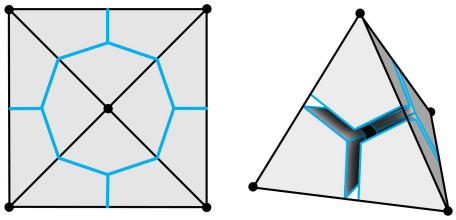


FIGURE 1.6 – Maillages hexaédriques structurés à deux instants de la simulation du Laser MégaJoule (voir FIGURE 1.2)

offrant une vue d'ensemble. Nous faisons uniquement une introduction des différents concepts importants pour la suite de cette thèse. Pour une vue exhaustive, nous renvoyons au très récent et complet état de l'art de Pietroni *et al.* [Pietroni *et al.*, 2022].

1.3.1 Maillages hexaédriques non structurés

Commençons par remarquer que, travaillant avec un maillage simplicial, une simple subdivision permet d'obtenir un maillage entièrement hexaédrique. Bien sûr, comme illustré sur les figures à droite, ce maillage est de très mauvaise qualité (les valences différentes de 4 sont très nombreuses). Cette méthode est donc à proscrire seule, mais elle peut trouver une utilité pour une partie du domaine. Les outils de simulation requièrent parfois un maillage entièrement hexaédrique, mais de bonne qualité uniquement en certains lieux. L'ingénieur peut donc se concentrer sur le maillage proche de ces lieux, et utiliser un maillage non structuré ailleurs.



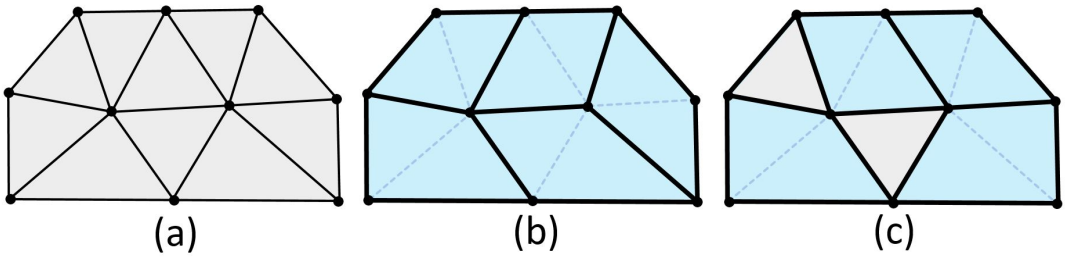


FIGURE 1.7 – Pour obtenir un maillage quadrangulaire du maillage triangulaire (a), une possibilité est de procéder à une fusion de triangles (b). Selon l'ordre des fusions, il peut être impossible de générer un maillage fait entièrement de quadrilatères (c).

Plus intéressant que la division, la fusion de simplexes est une approche similaire. Comme l'illustre la FIGURE 1.7, l'idée est de fusionner des simplexes voisins qui pourraient former un

quadrilatère. Bien sûr, le faire naïvement n'assure pas un résultat optimal (voir FIGURE 1.7-(c)), mais il existe des algorithmes efficaces, avec de bonnes propriétés, comme celui proposé dans GMSH [Remacle *et al.*, 2012].

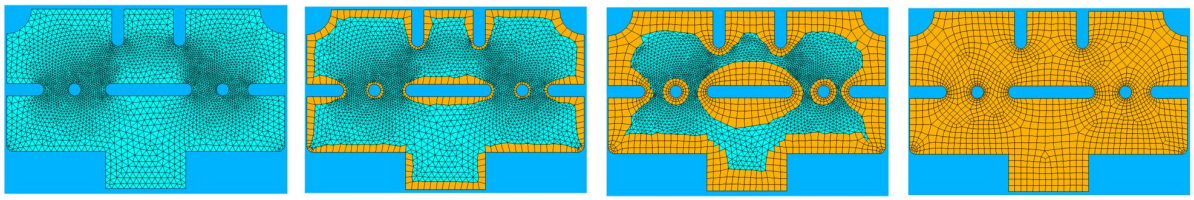
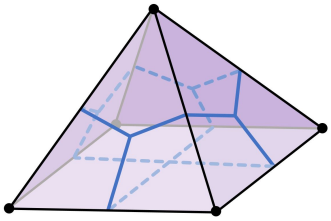


FIGURE 1.8 – Différentes étapes de l'algorithme *Qmorph* [Owen *et al.*, 1999] pour générer un maillage quadrangulaire depuis un maillage triangulaire. Images issues de [Owen, 2016].

Couplée avec une méthode d'avancée de front, où un maillage du bord est déjà choisi, la fusion de simplexes donne de très bons résultats en 2D. *Qmorph* [Owen *et al.*, 1999] donne un algorithme pour effectuer une telle avancée de front, comme illustrée sur la FIGURE 1.8. De nos jours, c'est l'approche privilégiée par les mailleurs commerciaux comme Star-CCM+ [Siemens Digital Industries Software, 2021], car elle assure une excellente robustesse et, appliquée sur un maillage triangulaire astucieusement généré, donne de bons maillages quasi-structurés.



Les résultats sont bien moins satisfaisants en 3D. Il y a deux raisons à cela : premièrement, deux tétraèdres ne forment pas un hexaèdre. Plus concrètement, les combinaisons à étudier sont bien plus complexes, ce qui implique qu'une approche telle que *Hmorph* [Owen et Saigal, 2000] produise des maillages mixtes, contenant des pyramides et des tétraèdres. Cette problématique se retrouve dans des méthodes plus récentes cherchant la robustesse [Sokolov *et al.*, 2016, Ray *et al.*, 2018, Gao *et al.*, 2017a, Bukenberger *et al.*, 2022]. Une étude récente [Pellerin *et al.*, 2018] montre qu'il existe 174 combinaisons de tétraèdres formant un hexaèdre. En plus de cela, il n'existe aucune garantie qu'il soit possible d'extraire un maillage purement hexaédrique d'un maillage tétraédrique avec uniquement des fusions.

Le second frein aux méthodes d'avancées de front est le manque de correspondance théorique et pratique entre les maillages du bord (surfacique) et de l'intérieur (volumique). Cela est très bien illustré par la fameuse pyramide de Schneiders [Schneiders, 1996], pyramide à base carrée dont le maillage surfacique est une simple subdivision de ses bords. Faire correspondre à ce bord un maillage volumique est très difficile. Les premiers résultats [Yamakawa et Shimada, 2010] donnent un maillage à 88(!) hexaèdres. Des études plus récentes ont réussi à descendre ce chiffre à 44 [Verhetsel *et al.*, 2019a] puis 36 [Verhetsel *et al.*, 2019b] hexaèdres, solutions restant très complexes. Cette limitation rend difficile les méthodes extrapolant un maillage hexaédrique depuis le bord, car les solutions se réduisent régulièrement à ces cas absurdement complexes. La méthode la plus aboutie dans ce domaine est le *Whisker weaving* [Folwell et Mitchell, 1999, Kawamura *et al.*, 2008], qui extrait des couches d'hexaèdres en fonction des boucles de quadrilatères sur le bord.

La méthode phare de l'industrie pour la génération de maillage hexaédrique est donc une approche qui ne s'appuie pas sur un maillage surfacique, mais sur une décomposition *octree* du domaine. Maréchal [Maréchal, 2009] a construit une approche permettant d'extraire un maillage hexaédrique depuis une grille d'*octree* (voir la FIGURE 1.9). Ce maillage est ensuite amélioré

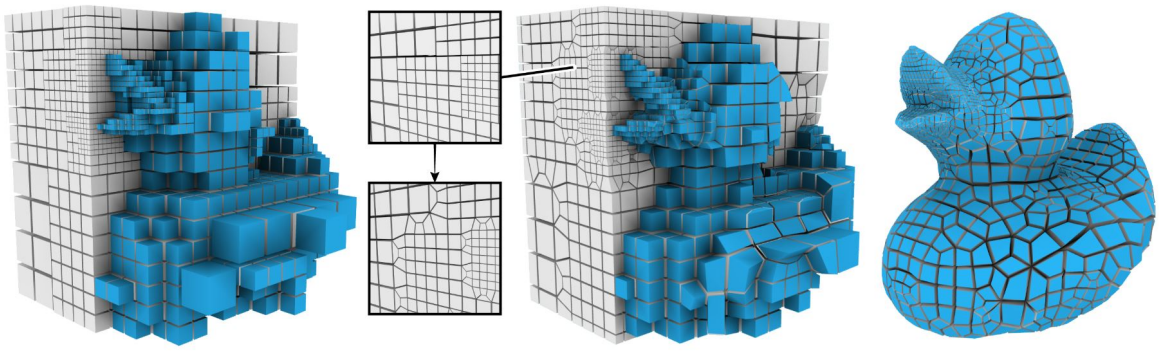


FIGURE 1.9 – Maillage hexaédrique (droite) obtenu à partir d'une grille d'*octree* (gauche). Image issue de [Livesu *et al.*, 2022].

par des étapes astucieuses de reprojection et d'insertion de couches d'hexaèdres supplémentaires [Maréchal, 2016]. Cette méthode a un défaut majeur : une qualité aléatoire proche des bords. Elle est en revanche très efficace, robuste et permet de créer des maillages adaptatifs facilement, très demandés dans l'industrie. Le mailleur (commercial) *Hexotic* [Dassault Systèmes, 2022] développé par L. Maréchal a un statut d'état de l'art pour le maillage hexaédrique automatique. Il crée bien sûr des maillages principalement non structurés.

1.3.2 Structure, déformation de grilles et blocs

La méthode classique pour décomposer un domaine en blocs est le travail manuel d'un ingénieur armé d'un logiciel adapté. Hansen et Owen [Hansen et Owen, 2008] soulignent que, en 2008, cette tâche représentait 31% du temps total lors des simulations de réacteurs nucléaires au *Departement of energy*. Une fois cette décomposition obtenue, une grille régulière est placée dans chacun des blocs, avec des critères de résolution de la grille sur chacune des interfaces entre blocs, souvent renseignés par l'ingénieur. C'est pour automatiser ce processus que les méthodes de paramétrisations globales sont étudiées. Avant de nous attarder sur celles-ci, mentionnons deux approches possibles pour des applications particulières :

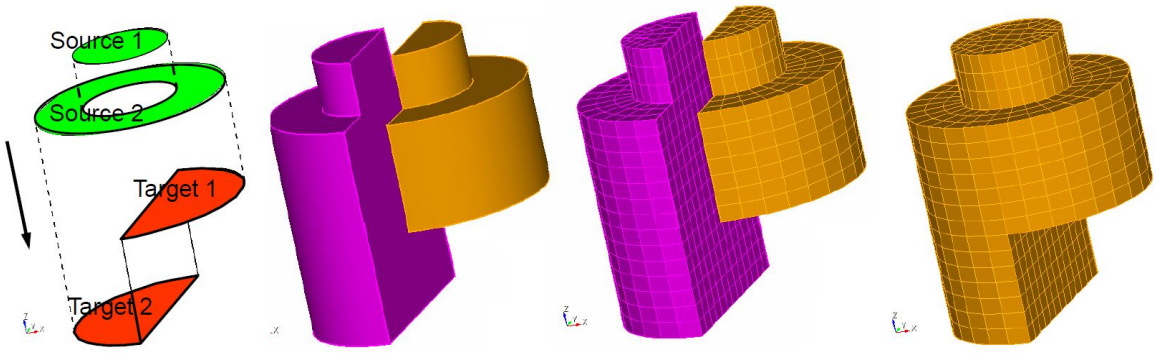


FIGURE 1.10 – Illustration d'un algorithme de *sweeping*. Image issue de [Weill et Ledoux, 2016].

Les méthodes de *sweeping* [Scott *et al.*, 2006], illustrées sur la FIGURE 1.10, sont utilisées sur les objets dont un maillage peut être extrudé depuis le bord. Dans certaines situations, différents

sources et objectifs peuvent être compatibles, comme sur la gauche de la FIGURE 1.10, ce qui permet d'appliquer ces méthodes dans une variété de contextes. Cette famille de méthode reste largement privilégiée dans les solutions par blocs comme Cubit [CUBIT, 2022]. Lorsque certains fronts se rencontrent, il est possible de trouver un maillage pouvant concilier les deux, mais cela reste un problème difficile [Staten *et al.*, 2010].

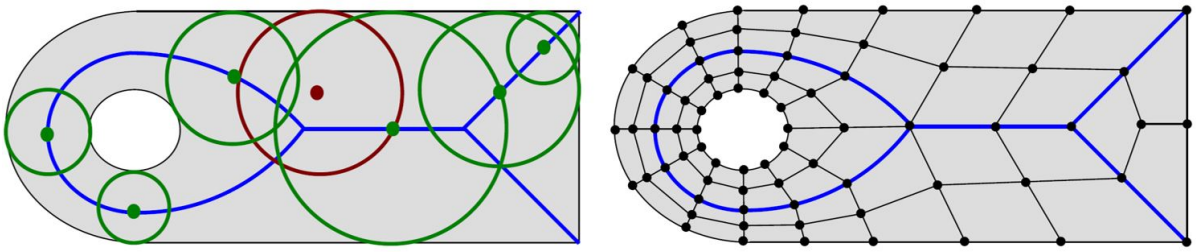


FIGURE 1.11 – À gauche, extraction du squelette d'un domaine à partir d'axes médians : les courbes bleues sont obtenues en extrayant tous les centres de cercles touchants deux points du bord, en vert dans l'image. Cela décompose la surface en blocs. À droite, maillage de la structure de blocs associé. Images issues de [Weill et Ledoux, 2016].

Certaines applications étudient des domaines particuliers pour lesquels une approche à base d'axe médian permet une bonne décomposition en blocs. L'idée est d'étudier la structure du "squelette" du domaine pour en déduire une décomposition, comme illustrée sur la FIGURE 1.11. Il est important de noter que cette approche fonctionne mal sur des domaines génériques. L'exemple d'un cube en donne une bonne intuition : celui-ci est découpé en 6 pyramides, qui, comme mentionné plus tôt, sont des blocs difficilement "maillables" .

Toutes les approches présentées jusqu'ici ont principalement été étudiées par la communauté du calcul numérique. Une approche récente très prometteuse, introduite par Ray et al [Ray *et al.*, 2006], trouve son origine dans la communauté de l'infographie : les paramétrisations globales. L'idée vient des travaux sur les applications de textures sur des objets 3D. S'il est possible d'appliquer une texture pour avoir une image sur un objet, alors appliquer une image de grille donne un maillage quadrangulaire de l'objet (voir la FIGURE 1.12). Bien sûr, de nombreuses contraintes sont nécessaires sur la carte de texture pour obtenir un maillage valide, nous développons cela dans le chapitre suivant.

En dehors de la notion de texture, cela revient à transporter chaque bloc du domaine vers un sous-ensemble de la grille. L'introduction de singularité (valence différente de 4 dans le maillage) nécessite d'étudier des déformations particulières, contenant des découpes. Dans ce sens, on distingue les méthodes dites de Polycubes, où la déformation est continue, sans découpe.

Après un travail de recherche considérable [Kaelberer *et al.*, 2007, Bommes *et al.*, 2009, Bommes *et al.*, 2013, Myles *et al.*, 2014, Campen *et al.*, 2015, Lyon *et al.*, 2021], les méthodes 2D approchent de nos jours une maturité industrielle. C'est loin d'être le cas en 3D. L'introduction des méthodes de Polycubes [Gregson *et al.*, 2011] a permis de simplifier l'étude de certaines étapes. L'objectif de cette thèse est de produire des algorithmes permettant de rendre les méthodes de polycubes robustes. Celles-ci donnent des résultats imparfaits car sans singularités, nécessaires à la génération de "beaux" blocs. Des travaux futurs auront pour objectifs d'étendre les résultats obtenus avec les polycubes vers les méthodes générales de paramétrisations globales.

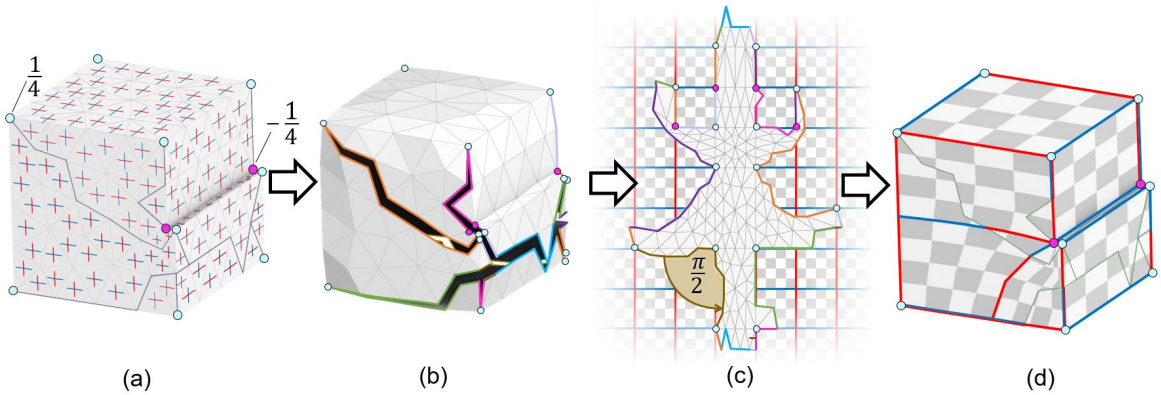


FIGURE 1.12 – Génération de maillage quadrangulaire avec une paramétrisation globale. (a) Un champ de croix est calculé avec des singularités, (b) ce qui permet de découper le maillage triangulaire pour (c) le déformer vers un plan. Cela permet d'y appliquer une texture de quadrillage, (d) qui crée un maillage quadrangulaire de notre domaine.

1.3.3 Tableau récapitulatif des méthodes de maillages hexaédriques

Méthode	*Tout type de domaines*	*Respect d'un maillage du bord*	*Alignement des éléments au bord*	*Non-sensible à l'orientation*	*Maillage structuré*	*Maillage adaptatif*	*Maturité industrielle*	*Domaine de recherche prometteur*
Division de simplexes	🟩	🟥	🟥	🟩	🟥	🟥	🟥	🟥
Fusion de simplexes	🟩	🟨	🟩	🟩	🟥	🟥	🟨	🟥
Avancée de Fronts	🟩	🟩	🟩	🟩	🟥	🟥	🟨	🟥
Whisker Weaving	🟨	🟩	🟩	🟩	🟥	🟥	🟨	🟥
Méthode à base d'*octree*	🟩	🟥	🟥	🟥	🟨	🟩	🟩	🟨
Sweeping	🟥	🟩	🟩	🟩	🟨	🟨	🟨	🟥
Axe médian	🟨	🟥	🟨	🟩	🟨	🟨	🟨	🟨
Polycube	🟩	🟨	🟨	🟥	🟩	🟨	🟨	🟩
Paramétrisation globale	🟩	🟨	🟩	🟩	🟩	🟨	🟥	🟩

🟩 Oui 🟥 Non 🟨 ça dépend

Dans le chapitre suivant, nous discutons plus en détail des méthodes de paramétrisations globales et des Polycubes, axes de recherches les plus prometteurs pour la génération automatique de maillages hexaédriques structurés.

Chapitre 2

Calcul de cartes pour la génération de maillages

Les méthodes de Polycubes, sujet d'étude de cette thèse, font partie des méthodes de paramétrisations globales. Ces dernières ont été introduites en infographie pour calculer des maillages quadrangulaires sur des surfaces et reposent sur le calcul d'une carte assez particulière. Dans la première section 2.1 de ce chapitre, nous détaillons la construction de ces cartes en 2D. Nous discutons ensuite, dans une seconde section 2.2, des nombreuses difficultés pour, à partir d'algorithmes fonctionnant bien en 2D, obtenir de bons algorithmes en 3D. De nombreux travaux ont cherché des solutions à ces problèmes, et notamment la famille des méthodes de Polycubes. Ces approches analysent un sous-ensemble des cartes possibles, ce qui en simplifie la compréhension. Nous décrivons dans une troisième section 2.3 l'état actuel des méthodes utilisant des Polycubes avec tous les problèmes qui restaient ouverts au début de ce doctorat.

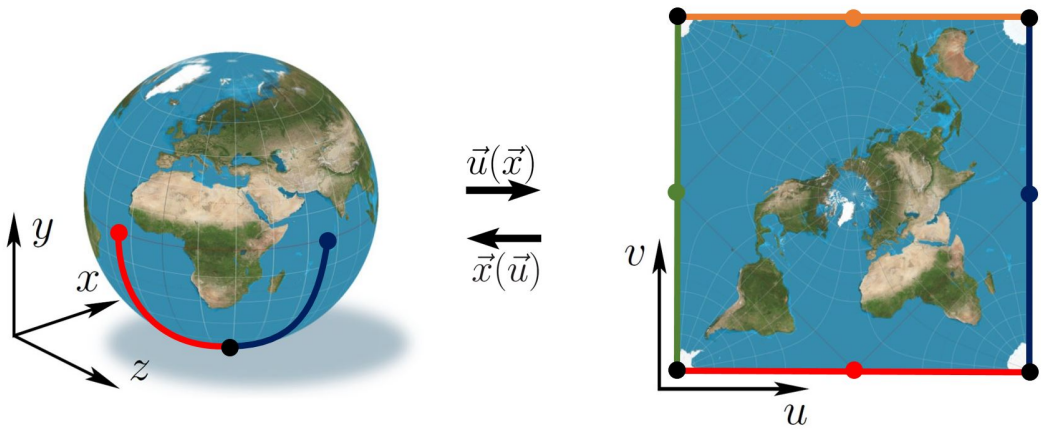


FIGURE 2.1 – Carte de la surface de la Terre, variété 2D d'un espace 3D, vers le plan.

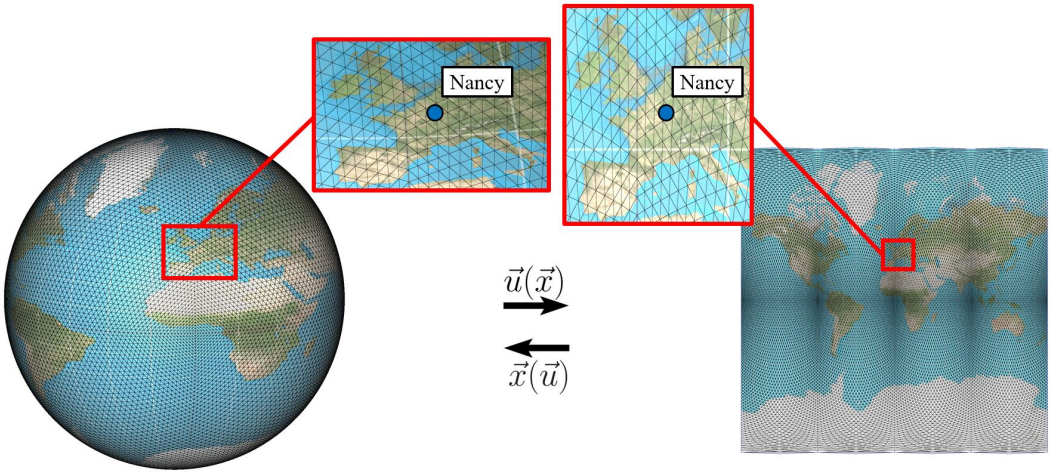


FIGURE 2.2 – Version discrétisée de planisphère. Chaque triangle sur la sphère trouve son équivalent dans l'espace cartographique.

2.1 Triangulations, cartes et génération de maillages quadrangulaires

Cette section commence par une description des cartes sur un maillage et de leurs propriétés. Nous introduisons ensuite les propriétés particulières qui permettent, depuis une carte, d'extraire un maillage. Nous mentionnons enfin les champs de croix (en anglais, *frame field*), clés permettant de calculer de telles cartes, et sujets de conséquentes recherches.

2.1.1 Cartographie de domaine

Élément de la vie courante, une carte est représentée en mathématiques par une fonction de l'objet cartographié, Ω, vers le plan, $\mathbb{R}^2$. L'exemple le plus commun est le planisphère, carte de la surface de la Terre, illustrée à la FIGURE 2.1. Par convention, cette fonction, dite aussi application, est notée $\vec{u}(\vec{x})$, où $\vec{\cdot}$ souligne le caractère vectoriel des coordonnées. Chaque point $\vec{x} \in \Omega$ est donc cartographié vers un point $\vec{u} \in \mathbb{R}^2$ du plan.

Pour représenter cette carte dans un ordinateur, nous travaillons avec une discrétisation de celle-ci. Nous avons une version discrète de notre domaine, une triangulation, sur laquelle nous définissons la carte, comme illustrée FIGURE 2.2. Dans un premier temps nous décrivons cette triangulation, puis nous introduisons la représentation des cartes discrètes.

Notre domaine Ω sera en général une variété 2D de $\mathbb{R}^3$, par exemple une sphère. Pour travailler avec un objet sous cette forme, nous travaillerons avec des aplatissements locaux de la surface. Pour cela nous profitons de la structure discrète [4], introduite dans la suite de la section, avec laquelle nous représentons la carte. Chaque triangle de notre objet peut préalablement être aplati sur le plan, indépendamment des autres, ce qui nous donne une référence dans $\mathbb{R}^2$ de celui-ci, et nous permet ainsi de nous rapporter à des définitions 2D. Dans la suite, nous considérons donc que Ω est un sous-ensemble de $\mathbb{R}^2$.

4. Cela est possible aussi dans le cas de cartes en général, c'est le sujet de la géométrie différentielle. Nous ne détaillons pas ces concepts dans cette thèse, car nous travaillerons uniquement avec des cartes discrètes.

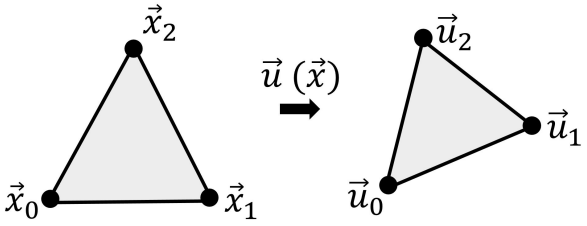


FIGURE 2.3 – Une carte discrète applique une déformation à chaque triangle du domaine.

Triangulation

Une triangulation est formée de deux composantes : $X = \{\vec{x}_i\}_{i=1}^{N_X}$, un ensemble de N_X points de Ω, et T un ensemble de N_T triangles dont les sommets sont dans X. Pour ne pas alourdir les notations, toutes les définitions suivantes seront à l'échelle d'un triangle dont nous noterons les trois points $\{\vec{x}_0, \vec{x}_1, \vec{x}_2\}$.

Chaque triangle i de T possède une base locale, que nous représentons à l'aide d'une matrice 2×2, B_i^X, de la façon suivante :

$$B_i^X = (\vec{x}_1 - \vec{x}_0, \vec{x}_2 - \vec{x}_0). \tag{2.1}$$

Cette base locale permet, entre autres, de calculer l'aire du triangle, avec la formule suivante :

$$\text{Aire}(T_i) = \frac{1}{2}\det(B_i^X). \tag{2.2}$$

Cette formule donne une aire signée du triangle. En effet, l'ordre de numérotation des points influe sur le signe du déterminant de B_i^X. Nous considérons dans la suite que la triangulation $\{X, T\}$ ne contient que des triangles à aires strictement positives. À noter que cela implique que B_i^X est inversible, puisque son déterminant est non nul.

Carte discrète

Les cartes discrètes de $\mathbb{R}^2$ sont des objets incontournables de l'infographie. Pour un aperçu de leurs utilisations, nous recommandons le cours SIGGRAPH de K. Hormann, B. Lévy et A. Sheffer [Hormann *et al.*, 2007]. Nous les représentons par une structure similaire à la triangulation : chaque triangle de T sera porté vers un nouveau triangle de $\bar{T}$, nouvel ensemble de N_T triangles avec des sommets dans $U = \{\vec{u}_i\}_{i=1}^{N_U}$, avec N_U le nombre de points de la carte et $\vec{u}_i \in \mathbb{R}^2$. Un exemple d'un couple triangulation-carte est donné sur la FIGURE 2.2.

Nous pouvons considérer le cas particulier des déformations où seule la géométrie des points est affectée, et donc $N_X = N_U$ et $T = \bar{T}$. Dans cette situation, chaque point de $\vec{x}_i$ de X est bien porté vers une nouvelle position $\vec{u}_i$ de U. Les polycubes sont des déformations, mais les méthodes de paramétrisations globales nécessitent la définition plus large que nous avons introduite. En effet, ces méthodes travaillent sur des déformations locales des triangles avec des contraintes particulières aux arêtes, que nous détaillons dans la suite.

Chaque triangle i de T composé des points $\{\vec{x}_0, \vec{x}_1, \vec{x}_2\}$ est porté vers le triangle i de $\bar{T}$, composé des trois nouveaux points $\{\vec{u}_0, \vec{u}_1, \vec{u}_2\}$, comme illustré à la FIGURE 2.3. Il est important de voir que la transformation de $\{X, T\} \to \{U, \bar{T}\}$ est linéaire sur chacun des triangles. Avec la base locale du nouveau triangle : $B_i^U = (\vec{u}_1 - \vec{u}_0, \vec{u}_2 - \vec{u}_0)$, nous pouvons exprimer, sur chaque

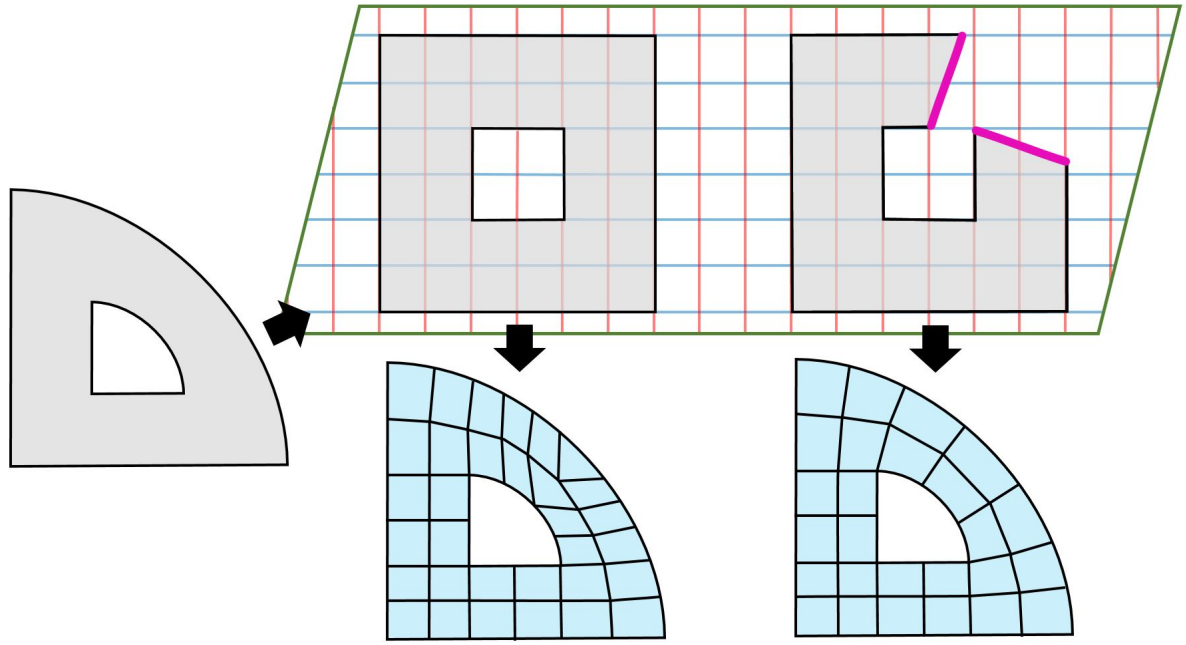


FIGURE 2.4 – Une carte depuis un objet vers un espace muni d'une grille permet, sous certaines contraintes, d'obtenir des maillages quadrangulaires de l'objet. L'introduction de découpes dans la carte (en violet) permet d'élargir les maillages possibles par cette méthode.

triangle i, la matrice jacobienne $J_i^{X\to U}$ de cette transformation linéaire avec la formule suivante :

$$B_i^X J_i^{X\to U} = B_i^U \tag{2.3}$$

Ce qui nous donne, en notant $J_i = J_i^{X\to U}$,

$$J_i = \left(B_i^X\right)^{-1} B_i^U \tag{2.4}$$

Les études d'une carte donnée se feront principalement à l'aide des matrices jacobiennes associées aux triangles. Une première propriété intéressante de J_i est sa linéarité par rapport aux coordonnées U. Cela permet de l'inclure aisément dans des systèmes d'optimisation sur les coordonnées de la carte. Une autre chose intéressante est qu'un triangle d'aire négative dans la carte implique $\det(J_i) < 0$. Ne pas avoir de triangle inversé, et donc $\forall i \det(J_i) > 0$, est une propriété remarquable, la carte est alors dite *injective*, ou *localement-bijective*.

Les *valeurs singulières* de J_i constituent la base des principales mesures de qualités utilisées pour les cartes. Celles-ci représentent les racines des valeurs propres de la matrice $J_i^T J_i$, et sont notées σ_1, pour la plus grande, et σ_2 pour la plus petite. La matrice $J_i^T J_i$ représente la *métrique* de la transformation, et ses valeurs propres donnent l'étirement que produit la transformation, dans ses principales directions.

2.1.2 Cartes pour la génération de maillages quadrangulaires

L'idée d'utiliser une carte pour générer un maillage quadrangulaire [Surazhsky *et al.*, 2003] est assez intuitive : utiliser une texture de grille sur un objet produit un résultat très similaire à

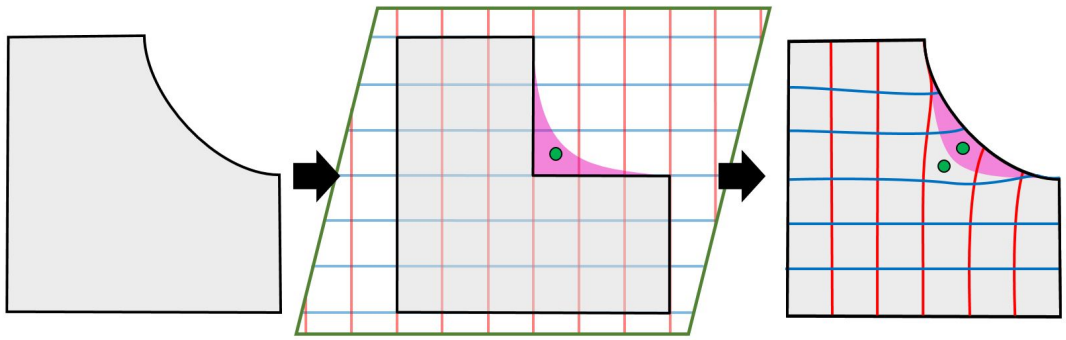


FIGURE 2.5 – Une carte contenant des parties inversées, représentées en violet, va générer des quadrilatères dégénérés, même si ses bords se situent sur les lignes de la grille. Le point vert se dédouble par la carte, alors même qu'il devrait se situer en dehors du domaine.

un maillage. Pour s'assurer que cette texture représente un maillage, il faut tout de même que celle-ci respecte un ensemble de contraintes particulières [5].

Plus précisément, la carte doit permettre que la grille conserve ses "propriétés" sur l'objet vers lequel elle est projetée. Nous disons dans ce cas que la carte est *grid-preserving*. Il est important de noter que nous définissons cette notion de conservation localement sur chacun des carrés de la grille. Cela permet d'introduire des singularités, sommets de valence différente de 4, permettant d'obtenir tous les maillages quadrangulaires possibles, et cela depuis une grille régulière.

Pour permettre cette liberté, nous considérons des cartes plus complexes que de simples déformations. En effet, celles-ci peuvent contenir des découpes, comme illustrées à la FIGURE 2.4. Sur notre triangulation, ces découpes interviendront à l'interface entre les triangles. Nous discutons dans la sous-section suivante de la méthode pour générer de telles découpes, et d'éventuelles singularités.

Pour qu'une telle carte soit *grid-preserving*, elle doit respecter trois types de contraintes :

1. **Injectivité :** Propriété fondamentale pour une carte de textures. Le carré de la grille doit être porté sans "défauts" sur l'objet (voir FIGURE 2.5). Un triangle retourné par la carte peut localement créer des formes non quadrangulaires. La propriété mathématique équivalente, et donc recherchée, est la suivante :

$$\det J_i > 0. \tag{2.5}$$

2. **Alignement avec le bord :** le bord de l'objet doit être aligné avec une ligne de la grille, pour que les quadrilatères épousent bien le bord de l'objet. Cela se traduit par le fait que :

   2.a deux points d'une arête du bord de la triangulation doivent avoir la même valeur dans l'une de leurs coordonnées, voir FIGURE 2.6-milieu,

   2.b cette valeur doit être entière, voir FIGURE 2.6-droite.

3. **Coupure *seamless* :** les quadrilatères doivent être compatibles des deux côtés d'une découpe, comme illustré dans la FIGURE 2.7. Cela se traduit par deux contraintes : les arêtes de chaque côté d'une découpe doivent être égales :

   3.a par rotation de $k\frac{pi}{2}$, voir FIGURE 2.7-milieu,

5. La définition imposée à la carte ne préserve pas la linéarité des arêtes des quadrilatères. Cela peut engendrer d'autres difficultés, que nous ne détaillons pas ici, car en pratique, l'impact est minime.

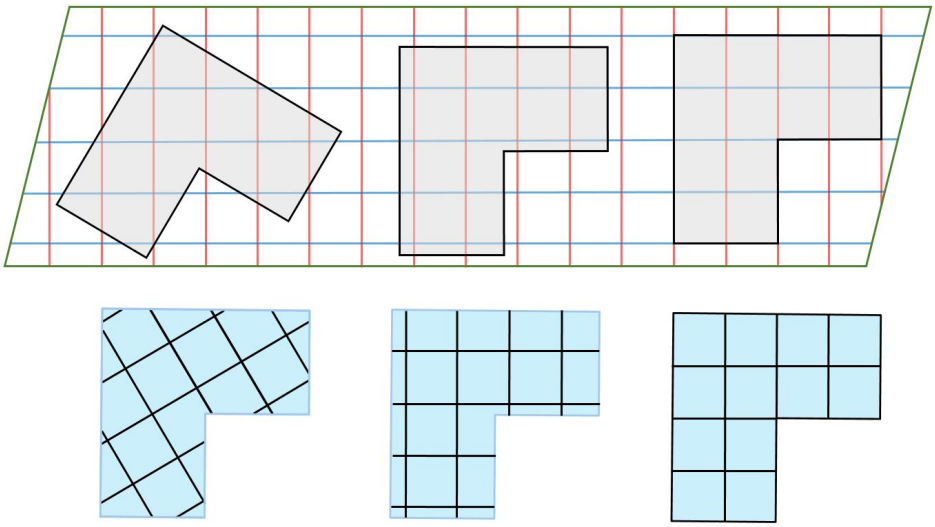


FIGURE 2.6 – Pour qu'une carte représente un bon maillage quadrangulaire, il faut que ces bords se trouvent sur des lignes de la grille.

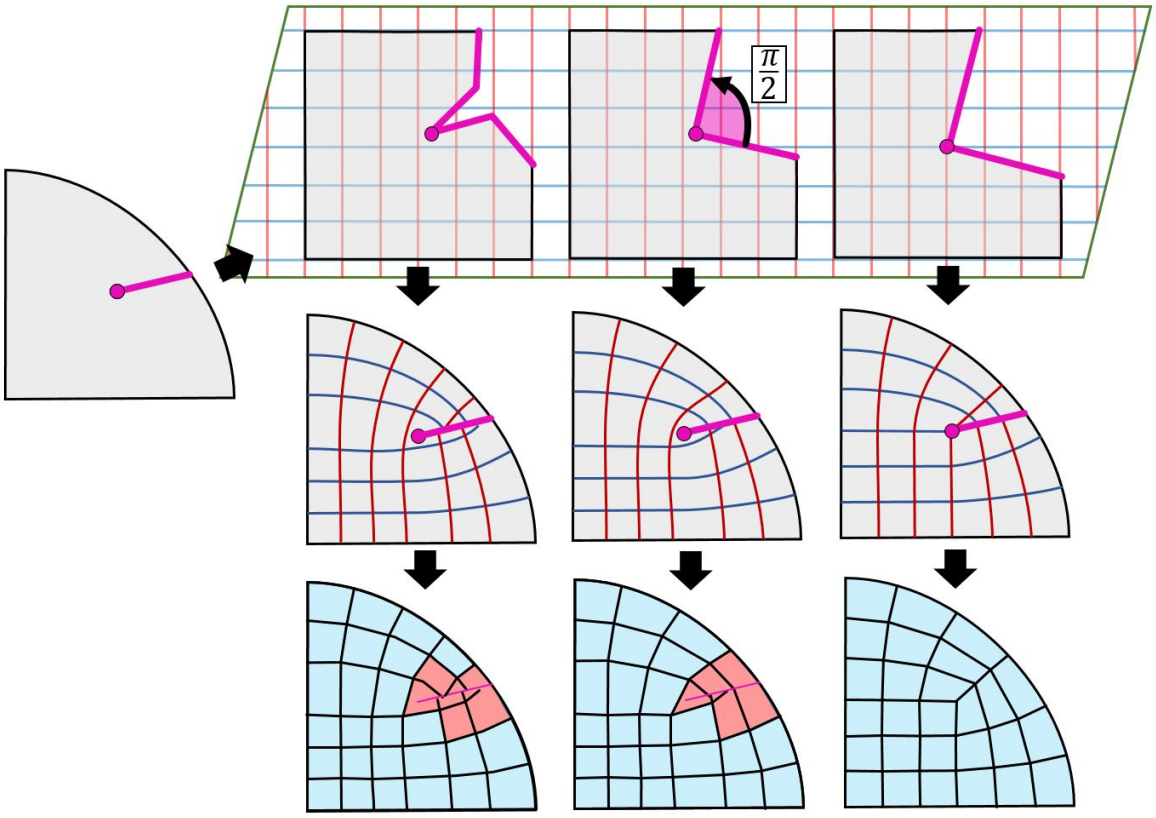


FIGURE 2.7 – L'ajout de découpes et de singularités dans la carte nécessite des contraintes supplémentaires pour obtenir un maillage quadrangulaire valide. Les arêtes de part et d'autre de la découpe doivent être égales par rotation de $k\frac{pi}{2}$, pour que les quadrilatères soient aussi compatibles (au milieu), et les singularités doivent être sur des valeurs entières, pour se trouver à l'intersection de lignes de la grille (à gauche).

3.b par une translation d'une valeur entière dans chaque dimension.

La translation permet de centrer les arêtes pour faire la rotation. Si une singularité est placée sur la découpe, cela contraint celle-ci à avoir des coordonnées entières, intuitivement, pour que les lignes de la grille se rencontrent bien au point singulier (voir FIGURE 2.7-droite).

En plus de ces trois contraintes, la carte recherchée doit distordre le moins possible la grille originale pour obtenir les quadrilatères de la meilleure qualité possible. Respecter ces trois contraintes tout en cherchant une carte de bonne qualité est un problème très difficile.

L'approche usuelle est de travailler en 2 temps : d'abord calculer un ensemble de singularités et de découpes à l'aide d'un champ de croix, ou *frame field* (nous présentons cela dans la sous-section suivante), et ensuite calculer une carte respectant ces contraintes. Kälberer *et al.* [Kaelberer *et al.*, 2007], avec leur approche *quadcover*, proposent de résoudre un premier système avec uniquement les contraintes **2.a** et **3.a**. De cette solution, un ensemble de valeurs entières est choisi permettant de respecter **2.b** et **3.b**, et cela permet de résoudre un second système pour trouver une carte en fonction des valeurs choisies.

Le travail de Kälberer *et al.* est particulièrement intéressant, car il donne une méthode permettant de travailler avec les contraintes **2.a** et **3.a** sur un très grand nombre de variables, et cela efficacement. En revanche, leur solution n'apporte aucune garantie de respecter la contrainte **1**, d'autant moins après le choix des valeurs entières pour **2.b** et **3.b** (celles-ci peuvent même être incompatibles avec une carte respectant **1**). Trouver ces valeurs entières, et produire la carte associée, est le sujet d'importantes recherches [Myles *et al.*, 2014, Campen *et al.*, 2015, Lyon *et al.*, 2019, Lyon *et al.*, 2021], nous en discutons plus en détail dans le chapitre 4, qui étudie ce problème pour le cas des polycubes. Au sujet de ces polycubes, et leurs équivalents 2D, les polycarrés, leur caractère de déformation permet de ne pas avoir de contrainte **3.** à respecter.

Il est intéressant de remarquer que le premier article proposant d'utiliser des cartes pour générer des maillages quadrangulaires, Ray *et al.* [Ray *et al.*, 2006], suggère de calculer une carte ne respectant pas exactement ces contraintes, mais une carte dite périodique. Cela permet une plus grande robustesse de la méthode, mais les cartes produites ne sont pas partout *grid-preserving*. La robustesse de cette approche est tout de même intéressante, et a donné lieu à de nombreuses suites [Knöppel *et al.*, 2015, Fang *et al.*, 2018], entre autres pour la génération de maillages quad-dominants [Jakob *et al.*, 2015] et hex-dominant [Sokolov *et al.*, 2016, Gao *et al.*, 2017a, Ray *et al.*, 2018], où les parties non *grid-preserving* de la carte sont remplies par d'autres types d'éléments (triangles, tétraèdres, pyramides...).

2.1.3 Champs de croix : guides pour le maillage

Les champs de croix sont la clé qui permet de faire fonctionner les méthodes utilisant des cartes pour le maillage quadrangulaire : les méthodes de paramétrisations globales. Il serait en pratique trop complexe d'optimiser [6] à la fois la qualité de la carte (distorsion) et sa combinatoire (singularités et découpes). Les champs de croix permettent de découper ce processus en 2 étapes.

L'intuition vient de la remarque que les champs de croix lisses ressemblent sensiblement aux gradients d'une carte donnant un bon maillage quadrangulaire. En effet, ces champs, introduits dans [Hertzmann et Zorin, 2000], possèdent les singularités désirées, et donnent l'orientation qu'auraient des quadrilatères de bonne qualité.

6. Ils existent tout de même des méthodes [Coiffier et Corman, 2022] essayant cette approche, mais elles sont très sensibles à des problèmes d'optimisations à cause des minima locaux.

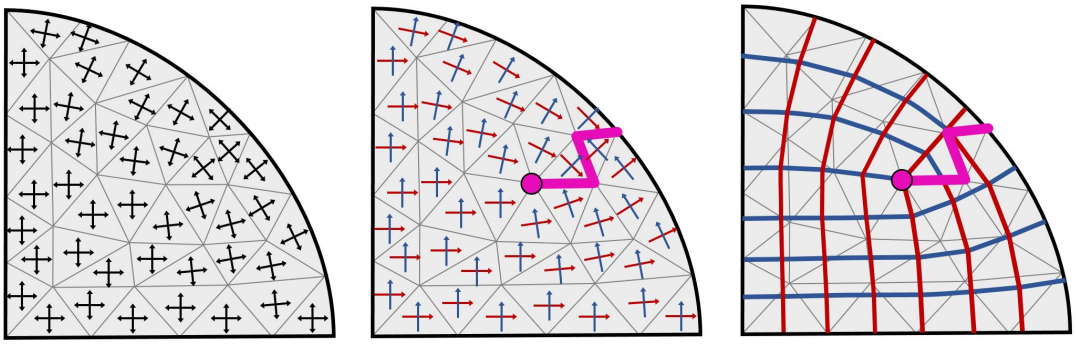


FIGURE 2.8 – Un champ de croix lisse et orthogonal au bord (à gauche) permet d'obtenir un champ de gradient contenant des singularités et des sauts induisants des découpes (au milieu). Un algorithme de type *quadcover* permet enfin d'en extraire un maillage quadrangulaire (à droite).

La procédure pour générer un maillage quadrangulaire est donc celle présentée à la FIGURE 2.8. Partant d'une triangulation, un champ de croix lisse et orthogonal au bord est calculé (voir la FIGURE 2.8-gauche). Un parcours permet de fixer l'orientation des gradients (chaque croix devient un repère orienté) et de déduire les découpes et les singularités (FIGURE 2.8-milieu). Le maillage final (FIGURE 2.8-droite) est obtenu en utilisant ces découpes dans un algorithme de type *quadcover*.

L'étude de ces champs de croix a donné lieu à une importante littérature scientifique, bien détaillée dans l'état de l'art de Vaxman *et al.* [Vaxman *et al.*, 2017]. Ils ont la propriété intéressante d'avoir une combinatoire théoriquement très proche d'un gradient de carte *grid-preserving* [Ray *et al.*, 2008]. Pour autant, ils ne sont pas exactement équivalents : les croix calculées sont généralement normées, pour en simplifier la manipulation, ce qui n'est pas le cas du gradient. Cela est la source de petites incompatibilités avec la méthode *quadcover*, largement discuté dans la littérature [Diamanti *et al.*, 2015, Coiffier et Corman, 2022].

Nous décrivons plus en détail la représentation et le calcul des champs de croix, et des champs de repères, dans le chapitre 5, où nous présentons une nouvelle méthode permettant de construire des champs de bonne qualité pour des modèles de CAO (voir FIGURE 2.9-à droite). À cette occasion, nous illustrons des situations où des champs lisses peuvent être incompatibles avec l'extraction d'un maillage quadrangulaire.

2.2 Problèmes de la méthode en dimension 3

Dans cette section, nous commençons par une description de la méthode de paramétrisation globale en dimension 3, en soulignant les équivalences avec la dimension 2. Nous énumérons ensuite tous les problèmes qu'implique le changement de dimension. Nous discutons enfin des différentes approches introduites dans la littérature pour contourner ces problèmes, dont font partie les méthodes de Polycubes.

2.2.1 Extension en dimension 3

Nous détaillons d'abord la transition d'une triangulation vers une tétraédrisation, puis nous expliquons les nuances introduites par le changement de dimension sur la méthode de paramétrisations globales.

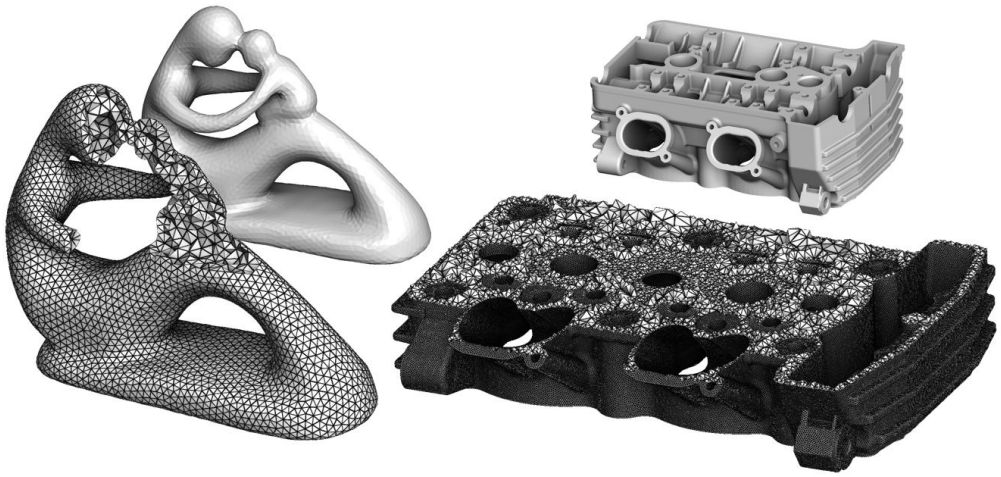


FIGURE 2.9 – Exemple de maillages tétraédriques pour (à gauche) un modèle classique de l'infographie, *Fertility*, dont le bord est lisse et (à droite) le modèle d'un moteur, exemple typique des modèles de CAO comprenant de nombreux angles saillants.

Tétraédrisation

Nous travaillons maintenant avec $\Omega \in \mathbb{R}^3$, et de la même façon, les cartes étudiées sont elles aussi dans $\mathbb{R}^3$. Contrairement à la dimension 2, dans laquelle nous pouvions considérer les surfaces comme des objets 2D plongés en dimension 3, il n'y a pas d'exemples pour lesquels il serait intéressant d'étudier une variété 3D d'un espace de dimension supérieure. Tous les objets que nous étudions vivent bien dans un espace 3D.

Leur représentation discrète est très semblable à la dimension 2, les deux étant des complexes simpliciaux. La triangulation devient une tétraédrisation, des exemples en sont donnés en FIGURE 2.9. Nous avons donc notre ensemble de points $X = \{\vec{x}_i\}_{i=1}^{N_X}$, où la seule différence est la dimension des points : $\vec{x}_i \in \mathbb{R}^3$. Il en est de même pour les points de la carte, $U = \{\vec{u}_i\}_{i=1}^{N_U}$, $\vec{u}_i \in \mathbb{R}^3$. Les tétraèdres sont composés de 4 points, T est donc un tableau de N_T quadruplets de points de X, et son équivalent $\bar{T}$ vers les points de U.

Sur un tétraèdre i donné, composé des points $\{\vec{x}_0, \vec{x}_1, \vec{x}_2, \vec{x}_3\}$, et $\{\vec{u}_0, \vec{u}_1, \vec{u}_2, \vec{u}_3\}$ sur la carte, nous utilisons des constructions analogues à la dimension 2 :

$$B_i^X = (\vec{x}_1 - \vec{x}_0, \vec{x}_2 - \vec{x}_0, \vec{x}_3 - \vec{x}_0) \tag{2.6}$$

est la matrice 3x3 représentant la base locale du tétraèdre i. Nous considérons que la tétraédrisation ne contient que des tétraèdres tels que $\det B_i^X > 0$, impliquant que les points sont numérotés tels que $\{\vec{x}_1 - \vec{x}_0, \vec{x}_2 - \vec{x}_0, \vec{x}_3 - \vec{x}_0\}$ est une base directe. Le volume d'un élément i se calcule avec la formule suivante :

$$\mathrm{Vol}(T_i) = \frac{1}{6} \det B_i^X, \tag{2.7}$$

et la matrice jacobienne de la carte peut se calculer avec :

$$J_i = \left(B_i^X\right)^{-1} B_i^U. \tag{2.8}$$

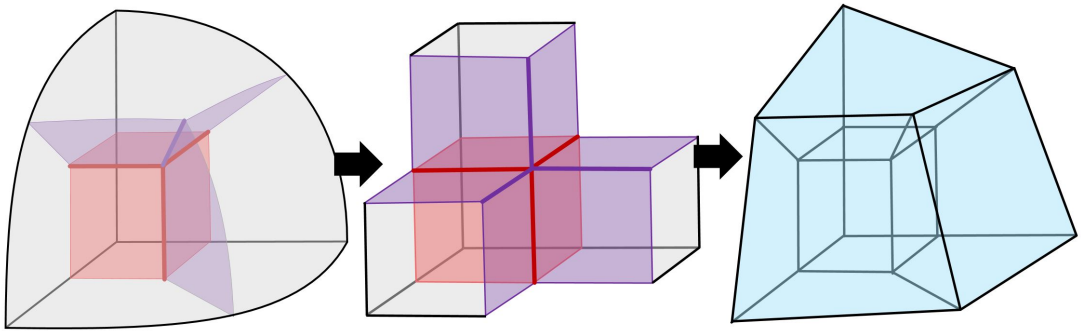


FIGURE 2.10 – Exemple de paramétrisation globale en 3D. L'objet à gauche est découpé pour être déformé vers une grille, et cette déformation est inversée pour extraire le maillage à droite.

Paramétrisations globales

La FIGURE 2.10 montre un exemple de paramétrisation globale en dimension 3. Une différence clé est la "dimension" des contraintes. Les singularités deviennent des lignes qui peuvent se rejoindre en certains points. De ces singularités, les découpes deviennent des nappes se prolongeant dans le domaine. Elles sont représentées par des sauts à l'interface entre les tétraèdres, cette interface étant un triangle.

Les contraintes pour qu'une carte 3D soit *grid-preserving* sont les mêmes que celles représentées dans la section précédente en dimension 2, au détail près des supports de ces contraintes : pour la contrainte **2**, il faut maintenant que les trois points d'un triangle sur le bord du domaine aient la même valeur dans une coordonnée, et bien sûr que cette valeur soit entière. Pour la contraintes **3.a**, ce sont des triangles qui doivent être égaux par rotation, mais ces rotations sont un peu plus complexes, nous y revenons juste après. La contrainte **3.b** diffère légèrement, en effet, les points d'une arête singulière doivent avoir deux de leurs coordonnées entières.

La méthode de *cubecover* introduite par Nieser et al. [Nieser *et al.*, 2011] est l'évolution de la méthode *quadcover*. La méthode repose donc sur une optimisation préalable d'un champ de croix 3D pour trouver les découpes et les singularités. Nuance importante, là où en dimension 2, un angle est nécessaire pour mesurer la différence entre 2 croix, il en faut 3 en dimension 3 (angles d'Euler). Il y a 4 symétries en dimension 2, c'est-à-dire 4 façons de représenter une même croix : une croix étant égale à elle même par une rotation de $\frac{\pi}{2}$, π et $\frac{3\pi}{2}$. En dimension 3, il y a 24 symétries, la contrainte **3**. est donc légèrement plus complexe. Ces symétries, et la méthode standard de représentation des croix 3D, les harmoniques sphériques d'ordre 4, sont expliquées en détail dans les travaux de Ray et Sokolov : *On smooth Frame Field Design* [Ray et Sokolov, 2015] et *Pratical Frame Field generation* [Ray *et al.*, 2016].

Il reste beaucoup de problèmes ouverts empêchant l'utilisation des paramétrisations globales comme méthode robuste de maillage hexaédrique. C'est le sujet de la section suivante.

2.2.2 Problèmes de la méthode

Les méthodes de paramétrisations globales comportent de nombreux challenges pratiques et théoriques encore non résolus. Certains d'entre eux empêchent la création de méthodes robustes. Dans cette section, nous présentons 8 de ces problèmes :

1. *Un maillage quadrangulaire du bord d'un objet peut ne pas être équivalent à un maillage simple de l'intérieur.*

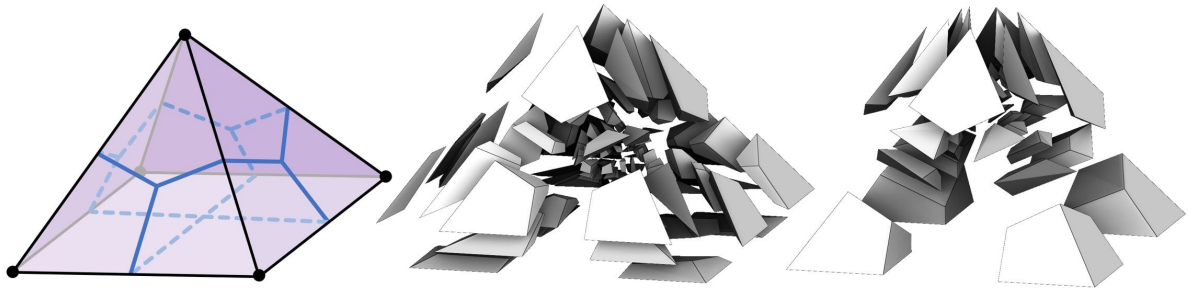


FIGURE 2.11 – A gauche, le bord de la pyramide introduite par Schneiders [Schneiders, 1996]. Au milieu, une solution à 88 hexaèdres introduite par Yamakawa et Shimada [Yamakawa et Shimada, 2010]. A droite le résultat le plus simple pour lequel il existe une géométrie valide, avec 44 hexaèdres, résultat obtenu par Verhetsel *et al.* [Verhetsel *et al.*, 2019a].

Même pour des objets simples, dont un maillage du bord est trivial, comme la pyramide dite de Schneiders, illustrée FIGURE 2.11, mailler avec des hexaèdres en conservant le maillage externe est très difficile. Cela est le sujet d'un travail récent réalisé par Verhetsel *et al.* [Verhetsel *et al.*, 2019b]. Pour une surface donnée, avec un maillage quadrangulaire imposé, ils créent l'ensemble des maillages hexaédriques topologiquement valides associés. Pour cela, ils développent un algorithme complexe, sans même garantir l'existence d'une solution géométriquement valide.

Les approches industrielles reposent malheureusement sur des méthodes dites *bottom-up*, maillant d'abord le bord, puis l'intérieur, avec le bord comme contrainte. Les résultats de Verhetsel *et al.* montrent que cette approche est vouée à l'échec pour les méthodes de paramétrisations globales.

2. *Les contraintes de bords pour les champs de croix 3D sur les modèles de CAO ne sont pas triviales.*
   En dimension 2, les contraintes données à l'algorithme *quadcover* vont associer un nombre de quadrilatères voulus en chaque point du bord : 1 quadrilatère pour un angle aigu, 2 quadrilatères si le bord est plat, 3 si l'angle est supérieur à 180°. Ces contraintes sont aussi présentes en dimension 3, mais il y a une nuance importante : elles vont se propager sur des lignes le long du bord, et ces lignes doivent respecter un ensemble de contraintes de cohérence globale. Ce problème semble accessible, contrairement à d'autres problèmes, mais il n'existe pas encore de solutions disponibles dans la littérature. Il existe des approches pour ajouter des hexaèdres sur un maillage déjà valide [Kowalski *et al.*, 2012], ce qui est un problème traitant des contraintes similaires.

3. *Les opérations de rotation 3D ne sont pas commutatives.*
   Les méthodes de génération de champs de croix utilisent les rotations entre les croix voisines. Pour obtenir le champ le plus lisse possible, elles minimisent ces rotations, ce qui nécessite d'utiliser les opérations définies sur l'espace des rotations. En dimension 2, celles-ci sont équivalentes à l'opération de multiplication sur des complexes, qui est commutative ($a \times b = b \times a$) et associative ($a \times [b \times c] = [a \times b] \times c$). En dimension 3, elles perdent la propriété de commutativité, comme les quaternions. Cette perte rend significativement plus complexe la création de bons schémas numériques pour la minimisation.

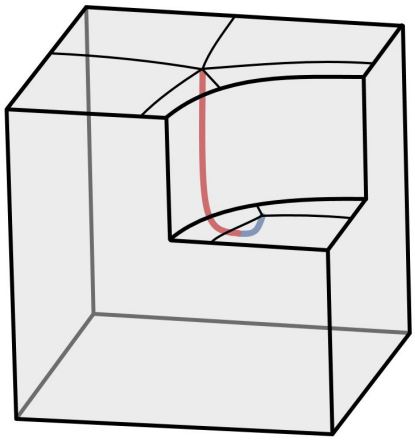


FIGURE 2.12 – La ligne rouge représente une singularité de valence 5 dans un champ de croix. Celle en bleu est une singularité de valence 3. Sur ce type de géométries, les méthodes de génération de champs de croix produisent des configurations où celles-ci se rejoignent, ce qui est incompatible avec la génération d'un maillage hexaédrique.

4. *Un champ de croix lisse n'est pas nécessairement maillable à l'aide d'un algorithme de type cubecover.*
   Des champs de croix peuvent contenir des réseaux de singularités incompatibles avec un maillage hexaédrique. Un exemple est donné à la FIGURE 2.12, où une singularité de valence 3 se transforme en une singularité de valence 5. Il est aisé de détecter *a posteriori* ces défauts, mais il n'existe pas à ce jour de méthode pour en éviter l'apparition lors de la minimisation. Malheureusement, les configurations où apparaissent ces défauts sont courantes sur les modèles de CAO.

5. *La discrétisation des singularités est de mauvaise qualité.*
   L'extraction des singularités depuis le champ de croix se fait généralement le long des arêtes de la tétraédrisation. La discrétisation peut donc rendre très zigzagant le tracé de celle-ci. Cela va induire des contraintes très complexes lors de la résolution de la méthode *cubecover*, à cause de la contrainte **3.**. La carte sera donc de très mauvaise qualité autour des singularités. HexEx [Lyon *et al.*, 2016] propose une approche pour extraire des cartes dans ce type de situation, mais requiert tout de même en entrée une carte respectant **2.a** et **2.b**. Or, les méthodes pour trouver les valeurs entières nécessitent des cartes de bonne qualité, et donc ne sont pas utilisables dans ces situations. On peut ajouter à cela qu'augmenter la résolution de la discrétisation n'est pas une solution : l'énergie habituellement utilisée pour la génération du champ de croix diverge autour des singularités, et se comporte mal lors d'une augmentation de la résolution.

6. *Les calculs de cartes 3D sont significativement plus difficiles qu'en dimension 2.*
   Il y a deux aspects à cela : une partie théorique, et une pratique :
   Théorie : l'outil par excellence pour faire des cartes est le *Tutte' embedding* [Tutte, 1963]. Celui-ci est garanti, en 2D, de construire des cartes bijectives (vers un convexe). Cette propriété ne s'étend pas en 3D, dans les versions discrètes et continues du problème. Il n'existait même aucune méthode fournissant des garanties de converger vers des bonnes cartes. Dans cette thèse, nous proposons une approche qui, pour la première fois, donne

de telles garanties. Nous détaillons cela dans le chapitre 3.
Pratique : Les tétraédrisations ont tendance à être largement plus grandes que les triangulations pour des modèles industriels (100 000 éléments *vs* 10 000 000 éléments). Il faut enfin ajouter à cela que les matrices de connectivité des points sont largement plus denses : un sommet a en moyenne 6 voisins dans une triangulation, contre 14 dans une tétraédrisation [Garimella, 2002]. Cela implique des matrices plus denses, et plus difficiles à inverser, et donc des méthodes plus lentes et numériquement plus difficiles. Un algorithme de type *quadcover* prend plusieurs minutes sur des problèmes difficiles, une méthode de *cubecover* prendra facilement plusieurs heures.

7. *Toute décomposition en blocs nécessite de manipuler des surfaces.*
   Les méthodes pour trouver les valeurs entières nécessaires au calcul de la carte s'appuient sur des décompositions en blocs des objets. En dimension 2, celles-ci peuvent être obtenues en découpant l'objet à l'aide de *motorcycles*, lignes tracées dans la paramétrisation non entière ou dans le champ de croix. Cette découpe est déjà complexe, et a donné lieu à de nombreux travaux [Myles et Zorin, 2013, Ray et Sokolov, 2014]. Cela devient encore plus difficile en dimension 3. Le tracé est simplement mal-défini dans un champ de croix, comme l'illustre la méthode proposée par Kowalsky *et al.* [Kowalski *et al.*, 2014]. Dans une paramétrisation, celui-ci devient possible [Brückler *et al.*, 2021], mais reste très fastidieux. De plus cela requiert une carte parfaitement injective.

8. *La méthode de midpoint subdivision ne donne pas un maillage hexaédrique sur des polyèdres quelconques*
   Si jamais une carte 2D n'est pas parfaitement *grid-preserving*, et qu'il reste des polygones non quadrangulaires, une procédure de subdivision supplémentaire (un point est placé au milieu de chaque élément et relié au milieu de chaque arête) permet de générer un maillage entièrement quadrilatère. Cette méthode s'appelle *midpoint subdivision*. Un triangle introduit un point de valence 3, un pentagone de valence 5, etc. En dimension 3, elle ne s'applique qu'aux polyèdres dont chacun des sommets est de valence 3, ce qui, en pratique, est rarement le cas. Cela empêche de simplement reboucher des cavités où la carte est de mauvaise qualité. Cela rend considérablement plus difficile la création de méthode *fullhex*. Dans cette thèse, nous utilisons tout de même cette approche en décomposant des objets en polyèdres astucieusement choisis pour de la génération robuste de maillage. C'est le sujet du chapitre 5, section 2.

2.2.3 Alternatives

Bien que d'importantes contributions aient été faites récemment, comme le travail de Buckler *et al.* [Brückler *et al.*, 2021], il semble trop complexe de chercher à directement résoudre le problème de paramétrisation globale pour le maillage hexaédrique. Dans ce sens, un ensemble de travaux étudie des versions alternatives du problème permettant de contourner certaines limitations.

Maillage hex-dominant

Une simplification classique du problème est de relaxer la propriété grid-preserving de la carte. Pour cela, l'approche est de calculer une carte dont le champ de croix n'est plus une contrainte, mais seulement un guide. La méthode privilégiée est d'utiliser des paramétrisations périodiques [Sokolov *et al.*, 2016, Gao *et al.*, 2017a]. Celles-ci peuvent ajouter des découpes et singularités

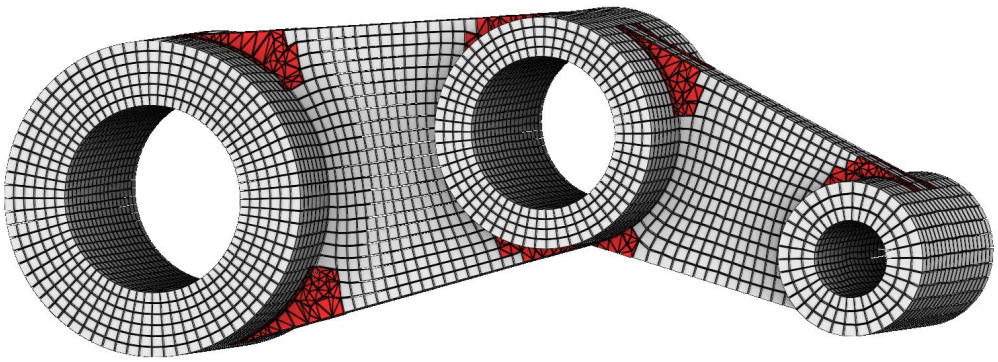


FIGURE 2.13 – Maillage hex-dominant d'un modèle de CAO. Les parties où le calcul de carte a échoué sont remplies à l'aide de tétraèdres et de pyramides, en rouge.

supplémentaires, non présentes dans le champ de croix, qui leur permettent de produire un résultat majoritairement grid-preserving, même si le champ de croix est invalide.

L'approche pour obtenir le maillage est d'ensuite extraire tous les sommets de la grille, et de les connecter à l'aide de tétraèdres en essayant de préserver les arêtes et facettes de la grille. La dernière étape est de les combiner pour obtenir un maximum d'hexaèdres. Cela a donné lieu à des études exhaustives de combinaisons de tétraèdres [Pellerin *et al.*, 2018]. Ray et al. [Ray *et al.*, 2018] ont proposé une approche approfondissant chaque étape du processus, pour atteindre une robustesse considérable. En effet, leur approche est capable de travailler efficacement même aux endroits où la carte calculée n'est plus injective. Un exemple de maillage hex-dominant généré avec cette approche est donné FIGURE 2.13.

Les maillages hex-dominant constituent une alternative intéressante pour l'industrie. Leur principal défaut est le manque de contrôle sur les zones où apparaîtront les éléments non hexaédriques. Certaines applications nécessitent des hexaèdres de bonne qualité à des endroits précis (par exemple proche du bord), mais peuvent travailler avec d'autres éléments loin de ces zones. Rien ne garantit en pratique que les méthodes seront capables de générer une bonne carte à ces endroits. La plupart des problèmes surviennent même proches du bord, rendant inutilisables ces maillages dans ces cas de figure. À noter qu'il existe des applications nécessitant des maillages entièrement hexaédriques, avec seulement la possibilité d'avoir des éléments de moins bonne qualité loin des phénomènes étudiés.

Dans cette thèse, nous étudierons uniquement des méthodes *fullhex*. Cela n'écarte pas l'utilisation de méthode hex-dominant dans certaines situations, par exemple lorsque nous voudrons récupérer un maillage sur carte partiellement de mauvaise qualité.

Maillage via Polycube

Par définition, un polycube est un amas de cubes[7]. Une déformation (carte sans découpe) respectant la contrainte **2.** va porter le domaine vers une forme qui est un polycube. Lorsqu'une déformation respecte la contrainte **2.a**, mais pas la contrainte **2.b**, nous l'appelons alors *polycuboid*, littéralement plusieurs pavés droits, car ses bords ne sont pas sur des valeurs entières.

Le fait que la carte ne contienne pas de découpe en simplifie considérablement l'étude. En effet, comme illustré à la FIGURE 2.14, les déformations peuvent ainsi être appliquées progressivement.

7. Nous considérons que ces cubes sont unitaires.

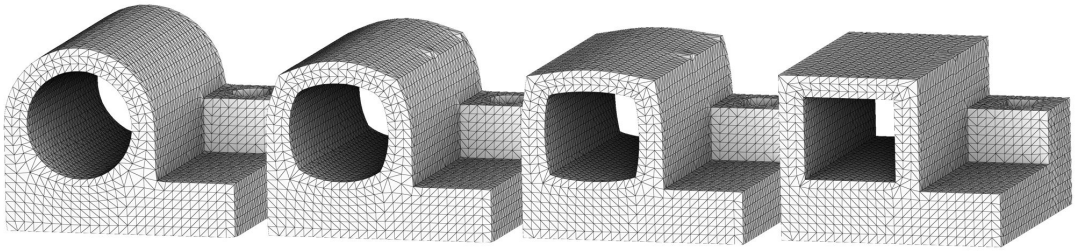


FIGURE 2.14 – De gauche à droite : déformation progressive d'un objet vers un Polycube.

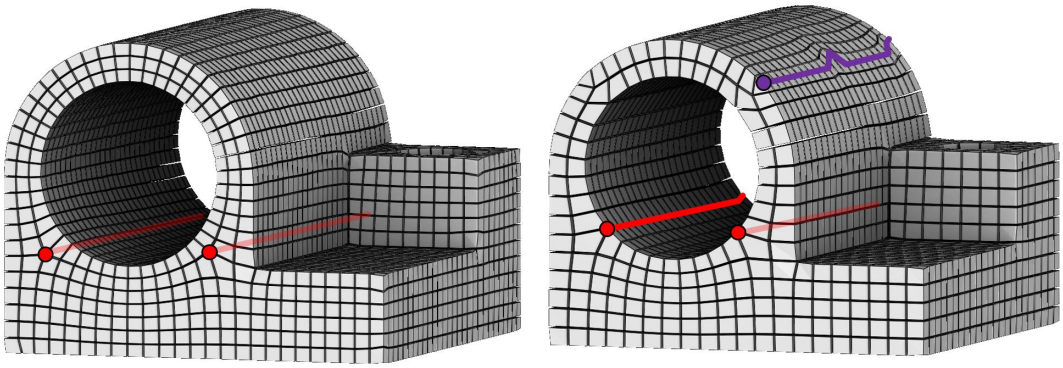


FIGURE 2.15 – À gauche, maillage obtenu à l'aide d'une paramétrisation globale [Corman et Crane, 2019], à droite modèle obtenu à l'aide d'une déformation en polycube. Le maillage de droite n'a pas de ligne de singularités internes, celles-ci se trouvent sur le bord de l'objet. Les lignes de singularités en rouge sont donc repoussées sur le bord de l'objet. D'autres lignes de singularités, comme celle en violet, apparaissent sur les parties courbes du domaine. Cela implique des éléments de moins bonne qualité le long du bord.

Cela permet d'ajouter graduellement les contraintes, et de pouvoir adapter ses choix au cours de la déformation, pour assurer de conserver la validité de la carte à toutes les étapes.

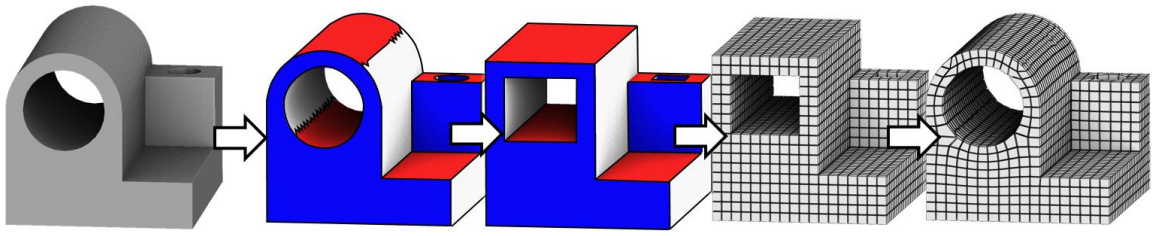
Les maillages produits possèdent, en revanche, des éléments de moins bonne qualité. Cela arrive particulièrement sur le bord, à la jonction entre les différentes faces des cubes, comme illustré à la FIGURE 2.15 . Pour corriger ces défauts, certains suggèrent d'ajouter de nouvelles découpes *a posteriori* permettant de corriger localement ces défauts [Fang *et al.*, 2016, Guo *et al.*, 2020]. D'autres proposent des insertions de couches d'hexaèdres pour également d'améliorer les résultats [Cherchi *et al.*, 2019].

Dans la section suivante, nous précisons la procédure standard pour obtenir des maillages hexaédriques avec des polycubes.

2.3 Méthode des Polycubes

Le sujet central de cette thèse est la méthode des polycubes. Cette section détaille les étapes clés de la méthode. Nous avons développé de nouvelles approches pour chacune de ces étapes. Dans cette section, nous détaillons leur thématique, les problèmes présents au début du doctorat, et enfin les différentes contributions que nous apportons dans cette thèse.

Nous pouvons distinguer quatre étapes dans la méthode des polycubes, illustrées à la FIGURE 2.16 . Le modèle est d'abord coloré, de cette coloration, une carte *polycuboid* est construite



1. **2.** **3.** **4.**

FIGURE 2.16 – Différentes étapes de la méthode des polycubes pour le maillage hexaédrique. Le modèle de gauche est **1.** coloré, puis **2.** déformé en fonction des couleurs. **3.** une quantification permet d'extraire une grille, qui **4.** par inversion donne un maillage de l'objet original.

par déformation (éventuellement itérativement en alternant des phases de colorations et de déformations). Celle-ci permet de construire un polycube par quantification, duquel nous obtenons enfin un maillage hexaédrique. Nous ajoutons à cela une étape de *post-processing* permettant d'améliorer le maillage final.

2.3.1 Coloration

Le principe de la coloration est de déterminer sur quelle face du polycube finira chacun des triangles du bord de notre objet. Nous distinguons 6 couleurs, $\{+X, -X, +Y, -Y, +Z, -Z\}$, qui correspondent aux normales sortantes d'un cube unité. Une coloration naïve est de simplement colorier chaque triangle avec la normale la plus proche de leur normale sortante. Pour des modèles très simples, cela peut donner de bons résultats, mais bien sûr sans aucune garantie.

Avoir des garanties pour la coloration est un problème très complexe, sujet souligné par Sokolov et Ray [Sokolov et Ray, 2015] dans un travail où ils essaient de corriger de mauvaises colorations, mais restent incapables de précisément définir les bonnes... En fait, ce problème est très similaire au problème de construction d'un bon graphe de singularité, mais ce graphe se trouve maintenant sur le bord, avec des contraintes de cohérences globales et locales, en plus de critères de qualité. Certaines méthodes [Gregson *et al.*, 2011] calculent un champ de croix sans singularité lié à la coloration.

Nous abordons les travaux de cette thèse concernant la coloration dans le chapitre 5. Nous y développerons aussi les cas typiques posant problème lors du calcul d'une coloration. Bien que ce soit surement le problème le plus fondamental de la génération de polycube, c'est celui sur lequel nos résultats sont "les moins bons". Nous avons tout de même essayé deux approches : une première cherchant à mieux comprendre les bords des modèles de CAO, et une seconde qui permet de relaxer les contraintes sur le bord, et rend donc moins problématiques des colorations invalides.

2.3.2 Déformation

La déformation donne une composante volumique à la coloration. L'objectif est dans un premier temps d'obtenir une carte respectant la contrainte **2.a** d'alignement du bord sur un axe. La contrainte d'entièreté **2.b** sera gérée par la quantification. Nous cherchons donc une carte injective où chaque triangle sur le bord a ses 3 points sur la même valeur dans la dimension de la coloration. Plus que cela, tout point d'une même face du polycube aura la même valeur.

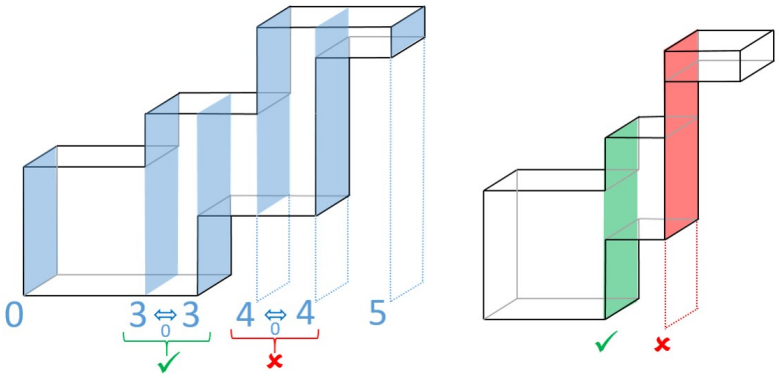


FIGURE 2.17 – Le processus de quantification choisit des valeurs entières pour chacun des bords du modèle. Certains choix peuvent conduire à des cartes dégénérées.

Résoudre un système en utilisant ce changement de variables permet d'automatiquement obtenir un *polycuboid*.

Le challenge réside dans la nécessité d'obtenir des cartes injectives, qui respectent donc la contrainte **1.** : aucun tétraèdre ne doit avoir un volume négatif dans la carte. Cela est bien sûr conditionné par la qualité de la coloration, mais reste un challenge pour de bonnes colorations sur des modèles complexes. Avant le début de la thèse, il n'existait pas d'approches robustes pour le calcul de carte bijective en 3D possédant des garanties d'injectivité. Nous présentons dans le chapitre 3 une nouvelle approche permettant le calcul de cartes de bonne qualité, même pour des contraintes complexes, et cela avec des garanties de converger vers une bonne solution.

2.3.3 Quantification

L'étape de quantification transforme un polycuboid en polycube, ce qui revient à trouver une carte respectant la contrainte **2.b**. Si cette carte est en plus injective, elle devient *grid-preserving*, et il devient possible d'en extraire des hexaèdres. L'approche naïve est de prendre les valeurs du bord dans le *polycuboid*, de les arrondir, puis de recalculer une nouvelle carte. Malheureusement, comme illustré sur la FIGURE 2.17, ces valeurs peuvent être incompatibles avec une carte injective, et dégénérer certaines parties de la carte.

Dans le chapitre 4, nous présentons une nouvelle approche permettant de garantir une quantification conservant l'injectivité de la carte. Plus que ça, si le *polycuboid* n'est pas parfaitement injectif, nous utilisons un processus de décomposition en blocs robuste aux éléments volumiques inversés, ce qui nous permet d'extraire de bons maillages dans ces situations.

2.3.4 Inversion ou extraction

Pour obtenir le maillage final, il y a deux possibilités : appliquer l'inverse de la déformation à la grille, ou directement extraire localement le maillage. Les méthodes récentes privilégient la seconde approche en utilisant l'utilitaire HexEx [Lyon *et al.*, 2016]. Celui-ci permet d'extraire des hexaèdres de bonne qualité dans des cartes comportant des défauts. Dans le chapitre 4, nous proposons notre propre approche d'inversion de la grille, qui permet d'éviter d'avoir à explicitement construire la carte polycube sur la tétraédrisation, ce processus pouvant s'avérer fastidieux. Dans le chapitre 5, nous développons également une méthode pour extraire des maillages *fullhex*

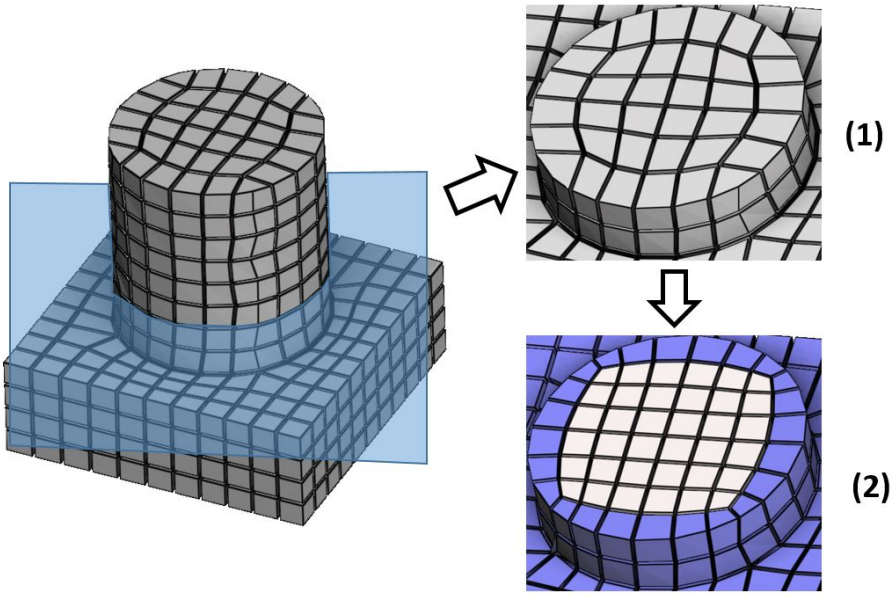


FIGURE 2.18 – (1) Un maillage obtenu avec des polycubes (à gauche) peut contenir des éléments de mauvaise qualité sur le bord. (2) Une procédure d'insertion de couches le long du bord, dite *padding*, permet réduire ces problèmes, et d'améliorer la qualité des maillages obtenus.

depuis des déformations lorsque celles-ci ne respectent pas parfaitement la contrainte **2**.

2.3.5 Amélioration a posteriori

Le défaut apparent des polycubes est la présence du graphe de singularité sur le bord de l'objet, qui induit de mauvais éléments. Une procédure dite de *padding* permet de repousser ce graphe vers l'intérieur de l'objet en insérant des couches d'hexaèdres, comme illustré à la FIGURE 2.18. En plus de cela, un lissage final des hexaèdres permet de corriger des petits défauts dans la carte obtenue. Nous présentons une méthode de lissage d'hexaèdres dans le chapitre 3.

les polycubes dans cette thèse

Les différentes étapes présentées dans cette section vont structurer le reste de cette thèse. Nous commençons par le chapitre 3 proposant une méthode pour calculer de bonnes déformations. Le chapitre 4 présente une nouvelle approche pour effectuer la quantification suivie de l'inversion. Le chapitre 5 discute enfin de nos travaux sur la coloration. À la fin de ce doctorat, l'ensemble de ces travaux ont été compilés dans un code : https://github.com/fprotais/robustPolycube, permettant d'extraire des hexaèdres depuis une coloration. Si cette coloration est valide, ce code est très robuste. Les recherches futures peuvent alors se focaliser sur la génération de ces colorations valides.

Deuxième partie

Contributions

Chapitre 3

Calcul de cartes de bonne qualité

Dans le chapitre précédent, nous avons distingué 4 étapes clés dans la méthode des polycubes. Ce chapitre aborde la seconde étape : la déformation. Le travail de Gregson *et al.* en 2011, introduisant les polycube pour la génération de maillages hexaédriques, repose sur une différence clé par rapport au travail originel de Tarini *et al.* en 2004 : une déformation volumique. Cette déformation permet à la fois de déterminer la géométrie interne du maillage hexaédrique, mais aussi de s'assurer que celui-ci ne soit pas dégénéré par endroit, par exemple à cause d'une mauvaise coloration du bord. Dès les premiers travaux sur le sujet, on cherche donc à calculer des cartes bijectives (éventuellement uniquement localement), et si possible de bonne qualité. En effet, la bijectivité donne l'intuition que l'on pourra obtenir des maillages hexaédriques non dégénérés (nous détaillons dans le chapitre 4). La qualité, de son côté, permet d'obtenir une meilleure géométrie, *i.e.* avec des éléments de meilleure qualité. Elle n'est pas primordiale, mais appréciable, puisque les processus permettant d'améliorer *a posteriori* la qualité peuvent s'avérer fastidieux...

Nombre d'articles [Gregson *et al.*, 2011, Livesu *et al.*, 2013, Huang *et al.*, 2014, Fu *et al.*, 2016, Guo *et al.*, 2020] traitent de ces notions, parfois en associant le calcul de la coloration à des déformations successives. Le constat est cependant toujours le même : il n'existe pas de méthode robuste et efficace pour calculer des déformations bijectives. Pour pallier à ce problème, nous présentons une nouvelle approche qui permet de systématiquement calculer des déformations bijectives. Nous développons cette approche en 4 étapes. Nous commençons par discuter, dans la section 3.1, des thématiques du calcul de déformations, ainsi que de cartes plus généralement. Nous présentons ensuite, dans la section 3.2, notre méthode pour récupérer la bijectivité d'une carte via une approche de pénalisation. Dans la section 3.3, nous étendons l'approche de pénalisation pour borner la qualité des cartes, ce qui permet de maximiser leur pire qualité. Nous détaillons enfin, dans la section 3.4, deux extensions pratiques de ces méthodes : la première (sous-section 3.4.1) étudie le cas particulier des déformations à bord libre, et la seconde (sous-section 3.4.2) montre l'utilisation de carte, et donc de nos approches, pour améliorer la qualité (lisser) des maillages hexaédriques.

Il est important de noter que ces contributions sortent largement du cadre des polycubes, et sont même des avancées importantes dans le domaine de la modélisation géométrique, où le calcul de carte est un outil incontournable. Le travail présenté a donné lieu aux publications suivantes :

- Sur la section 3.2 :
  Garanzha, V., Kaporin, I., Kudryavtseva, L., Protais, F., Ray, N. et Sokolov, D. (2021a). Foldover-free maps in 50 lines of code. *ACM Transactions on Graphics*, 40(4)
- Sur la sous-section 3.4.1 :

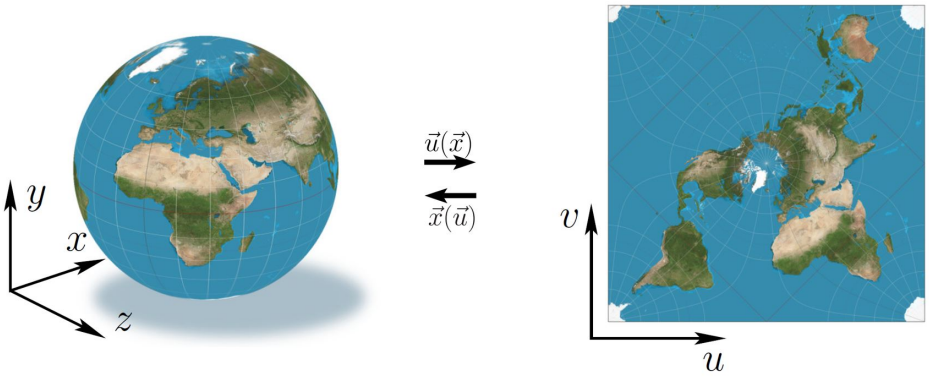


FIGURE 3.1 – L'objet le plus cartographié : la terre. La terre vit un espace 3D dont les coordonnées sont communément représentées par $\vec{x} = (x, y, z)$, et sa carte est un espace 2D, avec pour coordonnées $\vec{u} = (u, v)$.

Garanzha, V., Kaporin, I., Kudryavtseva, L., Protais, F., Ray, N. et Sokolov, D. (2021b). On Local Invertibility and Quality of Free-boundary Deformations. *In IMR 2021 - 29th International Meshing Roundtable*, Virtual, United States

Et a un article en cours de soumission :

- Sur la section 3.3 :
  Garanzha, V., Kaporin, I. E., Kudryavtseva, L. N., Protais, F., Desobry, D. et Sokolov, D. (2022). Practical lowest distortion mapping. *CoRR*, abs/2201.12112

3.1 Cartes et déformations, un sujet important de recherche

Notre intérêt principal est l'utilisation des cartes pour des déformations en polycubes. Mais leur étude est bien plus générale. Nous proposons dans un premier temps un aperçu des méthodes de construction de cartes, avec les thématiques associées, et ensuite nous présentons le problème variationnel sur lequel reposent nos contributions.

La plupart des objets géométriques sont représentés par une surface triangulée ou un maillage tétraédrique. La construction de carte (ou *cartographie*), illustrée FIGURE 3.1, est un problème qui consiste à générer une carte 2D ou 3D de ces objets. Il s'agit d'un problème fondamental de l'infographie, car il est beaucoup plus facile pour de nombreuses applications de travailler dans cet espace cartographique que de manipuler directement l'objet lui-même. En plus du remaillage hexaédrique avec des Polycubes [Gregson *et al.*, 2011] et des méthodes de *paramétrisations globales* telles que [Bommes *et al.*, 2013] en 2D et [Nieser *et al.*, 2011] en 3D, il existe de nombreuses utilisations des cartes, abordées dans le cours SIGGRAPH [Hormann *et al.*, 2007] de K. Hormann, B. Lévy et A. Sheffer. Quelques exemples : le *texture mapping* [Lévy *et al.*, 2002] stocke les couleurs d'une surface sous forme d'images dans l'espace cartographique, [Aigerman et Lipman, 2016] utilise des cartes de différents objets vers un même espace, pour en déduire des correspondances. Les algorithmes de cartographie sont aussi utilisés pour déformer des volumes [Li *et al.*, 2020], ou bien générer des coquilles à partir de surfaces [Jiang *et al.*, 2020].

3.1.1 Qu'est-ce qu'une bonne carte ?

Le plus souvent, les cartes sont représentées par la position des sommets dans l'espace cartographique, et interpolées linéairement sur chaque élément (triangle ou tétraèdre). Dans un

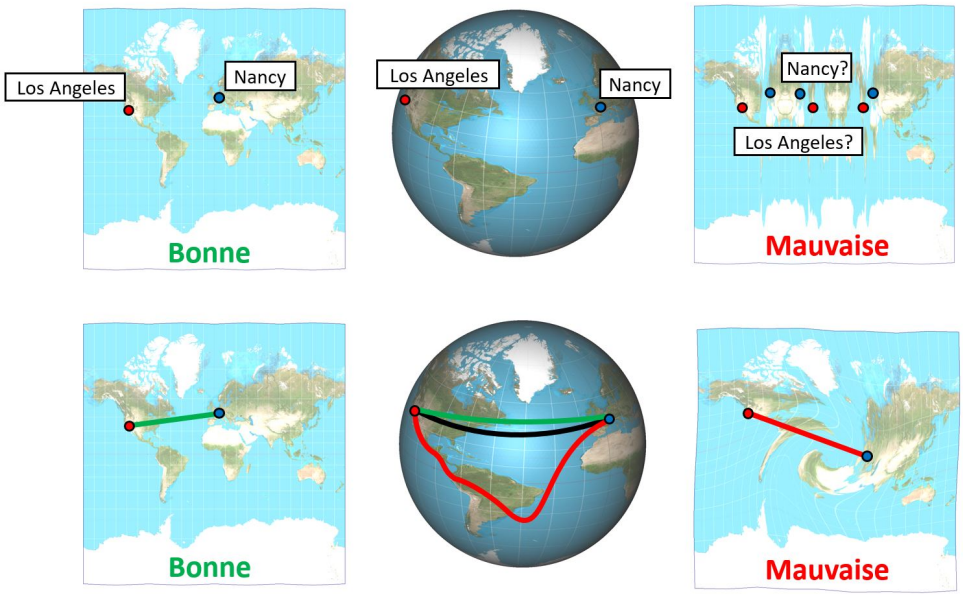


FIGURE 3.2 – Les deux éléments permettant de juger de la qualité d'un carte sont : sa bijectivité (ligne du haut), et sa distorsion (ligne du bas).

monde parfait, l'espace cartographique conserverait les distances géodésiques de l'objet, comme illustré FIGURE 3.2. Malheureusement, cela est généralement impossible en raison de la courbure gaussienne et des contraintes spécifiques à l'application, telles que des contraintes de position des sommets ou des chevauchements dans l'espace cartographique. Par conséquent, l'objectif des algorithmes de construction de carte est de minimiser la distorsion entre l'espace géométrique et l'espace cartographique, ce qui nécessite des méthodes d'optimisation numérique comme détaillé dans le cours [Hormann *et al.*, 2008].

Qu'en est-il de l'inversibilité ? Malheureusement, lorsqu'une forte distorsion est nécessaire pour satisfaire les contraintes, ces algorithmes peuvent perdre la propriété fondamentale d'une carte : la bijectivité [8]. Une solution pour la préserver [Floater, 1997] repose sur le théorème de Tutte [Tutte, 1963] (voir FIGURE 3.3). Cependant la frontière de la surface doit être contrainte sur un polygone convexe. Malgré cette forte limitation, il reste l'algorithme de référence pour générer des cartes injectives[8]. Une distorsion plus faible peut être obtenue en changeant les poids des coordonnées barycentriques [Eck *et al.*, 1995] (tant qu'ils ne sont pas négatifs), et des solutions alternatives [Campen *et al.*, 2016, Shen *et al.*, 2019] ont été explorées pour améliorer la robustesse aux imprécisions numériques en modifiant la connectivité du maillage. Une fois la carte inversible obtenue, sa distorsion peut être minimisée, mais toujours en portant une attention particulière aux chevauchements globaux [Su *et al.*, 2020, Ye *et al.*, 2020].

Invertibilité locale. Dans de nombreuses applications, les cartes sont utilisées pour accéder au voisinage d'un point dans un système de coordonnées local cohérent. À cette fin, l'injectivité

8. Nous utilisons une définition discrète de la bijectivité et de l'injectivité. Une carte f est bijective s'il existe une correspondance unique entre un point du domaine Ω et l'espace $f(\Omega)$. Concrètement, cela implique pour une carte discrète que deux simplexes ne peuvent pas se chevaucher dans l'espace cartographique. L'injectivité assure cette propriété mais uniquement localement (dans un voisinage local), nous parlons parfois de bijectivité locale. Pour qu'une carte discrète soit injective, il suffit qu'elle ne retourne aucun simplexe.

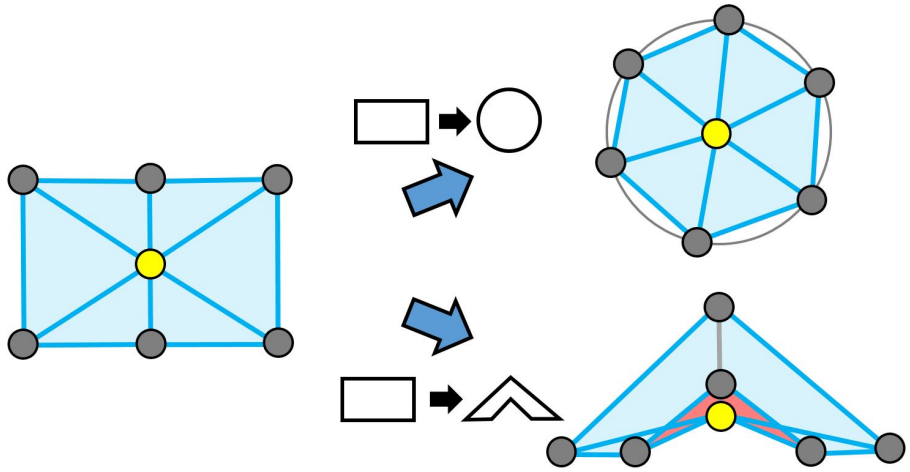


FIGURE 3.3 – Illustration du *Tutte's embedding*. On déforme le rectangle à gauche en bougeant les points en gris. On place ensuite le point jaune au milieu de ses voisins. Dans le cas de la déformation vers le cercle, en haut à droite, qui est convexe, [Tutte, 1963] a montré que la carte obtenue sera bijective. Ce n'est plus le cas lors du déformation vers une forme non-convexe, comme en bas à droite, dont la carte contient des retournements en rouge.

globale n'est pas requise, et nous recherchons plutôt l'injectivité locale [Schüller *et al.*, 2013, Smith et Schaefer, 2015]. Dans ces travaux, l'approche choisie part d'une carte injective [Floater, 1997], et maintient l'injectivité locale lors de la minimisation de la distorsion et de l'application des contraintes. Cela permet d'optimiser à la fois la paramétrisation et l'assemblage de la texture [Jiang *et al.*, 2017] avec une possibilité d'extension à des maillages plus grandes [Rabinovich *et al.*, 2017].

3.1.2 Récupérer l'injectivité locale

L'injectivité locale peut également être récupérée pour une carte avec peu de triangles inversés. Par exemple, en 2D [Lipman, 2012] et en 3D [Aigerman et Lipman, 2013], la carte est projetée sur une classe de cartes à distorsion bornée. L'alternance entre les étapes de projection et d'optimisation [Kovalsky *et al.*, 2015, Fu et Liu, 2016, Su *et al.*, 2019, Naitsat *et al.*, 2020] permet souvent de produire des cartes sans repliement et avec une faible distorsion. Les méthodes numériques ont cependant peu de chances de réussir pour les problèmes difficiles. La méthode que nous développons permet de récupérer des cartes injectives dans les cas les plus difficiles, et cela aussi bien en 2D qu'en 3D.

La récupération de l'injectivité locale est également connue sous le nom de démêlage de maillages (*mesh untangling*). Initialement lié à l'approche *Arbitrary Lagrangian-Eulerian moving mesh*[9], le problème du démêlage considère un complexe simplicial dont les éléments sont mal orientés et tente de les retourner en optimisant la position des sommets. Il existe une littérature abondante sur le démêlage [Du *et al.*, 2020, Knupp, 2001, Freitag et Plassmann, 2000, Escobar *et al.*, 2003, Toulorge *et al.*, 2013]. Cependant, l'opinion commune est que le démêlage est un problème très difficile et que les algorithmes ne sont pas assez robustes. Une approche tout à

9. Méthode de simulation utilisée en mécanique des fluides où la solution et le support (le maillage) sont liés. De fortes contraintes peuvent retourner le maillage, ce qui rend une étape démêlage nécessaire.

l'opposé du démêlage, [Danczyk et Suresh, 2013] étudie une méthode d'éléments finis fonctionnant directement sur des maillages contenant des triangles inversés.

Déformations élastiques. Pour récupérer l'injectivité locale, nous proposons une méthode issue du calcul numérique. Il est très important de noter qu'il existe une riche littérature sur la déformation des maillages dans la communauté travaillant sur la génération de grilles pour le calcul scientifique. Dans les années 60, Winslow et Crowley, indépendamment l'un de l'autre, ont introduit des méthodes de génération de maillage basées sur des cartes harmoniques inverses [Crowley, 1962, Winslow, 1966].

Depuis lors, beaucoup d'efforts ont été consacrés à la génération de maillage fondée sur les déformations élastiques [Jacquotte, 1988], mais principalement pour des grilles régulières. En 1988, à l'époque de la domination des méthodes de génération de grilles cartographiées par différences finies, S. Ivanenko a introduit le concept de méthodes de génération de grilles variationnelles à barrière garantissant la construction de grilles non dégénérées [Ivanenko, 1988, Charakhch'yan et Ivanenko, 1997]. Pour générer des déformations avec une distorsion globale bornée (constante de quasi-isométrie bornée), Garanzha a proposé de minimiser une énergie élastique pour un matériau hyperélastique avec raidissement supprimant les déformations singulières [Garanzha, 2000]. Le théorème d'inversibilité pour la déformation de ce matériau a été établi dans le cas 3D également [Garanzha *et al.*, 2014].

3.1.3 Formulation variationnelle

La base de notre contribution est l'utilisation d'une énergie bien posée et bien étudiée, reposant sur une formulation variationnelle du problème. Désignons par $\vec{u}(\vec{x})$, $\vec{u} : \Omega \subset \mathbb{R}^d \to \mathbb{R}^d$, une carte vers un domaine paramétrique, d dénote la dimension, les équations étant valables en 2D et 3D. Dans nos notations, nous utilisons des flèches pour tous les vecteurs de dimension d. Nous avons donc pour le cas plat en 2D, $\vec{u}(x, y) = (u(x, y), v(x, y))$, et pour une carte en 3D $\vec{u}(x, y, z) = (u(x, y, z), v(x, y, z), w(x, y, z))$.

Considérons le problème variationnel suivant :

$$\arg\min_{\vec{u}} \int_{\Omega} \left(f(J) + \lambda g(J)\right)\, dx, \tag{3.1}$$

où J est la matrice jacobienne de la cartographie $\vec{u}(\vec{x})$, et

$$f(J) = \begin{cases} \frac{\operatorname{tr} J^\top J}{(\det J)^{\frac{2}{d}}}, & \det J > 0 \\ +\infty, & \det J \leq 0 \end{cases}$$

$$g(J) = \begin{cases} \det J + \frac{1}{\det J}, & \det J > 0 \\ +\infty, & \det J \leq 0 \end{cases}$$

Le problème (3.1) peut être soumis à certaines contraintes que l'on n'écrit pas explicitement. Pour donner un exemple, on peut épingler certains points dans la carte. Dans cette formulation, les fonctions $f(J)$ et $g(J)$ ont des buts concurrents, l'une préserve les angles et l'autre préserve l'aire, et donc λ sert de paramètre de compromis.

Définition 1 (Carte admissible)**.** *On dit qu'une carte u est **admissible** si :*

$$\min_{\Omega} \det J > 0.$$

En d'autres termes, si elle ne contient aucun élément retourné.

Remarque 1. *Une carte u admissible est* localement injective.

Notez que si une estimation initiale n'est pas admissible alors la fonctionnelle (3.1) n'est pas définie. Notre première contribution, détaillée dans la seconde section, intervient à ce niveau : proposer une régularisation qui permet d'avoir une énergie définie, avec un système d'optimisation qui permet de retrouver une carte admissible.

Remarque 2. *avec $\lambda = 0$ et $d = 2$, le problème* (3.1) *présente une formulation variationnelle d'un problème de carte harmonique inverse. En d'autres termes, si nous écrivons les équations d'Euler-Lagrange pour le problème* (3.1) *et échangeons les variables dépendantes et indépendantes*[10]*, nous obtenons l'équation de Laplace $\Delta\vec{x}(\vec{u}) = \vec{0}$ (à ne pas confondre avec l'omniprésent $\Delta\vec{u}(\vec{x}) = \vec{0}$!). Dans ce cas, le problème* (3.1) *est souvent appelé la fonctionnelle de Winslow, cependant Alan Winslow lui-même n'a jamais formulé le problème variationnel, travaillant avec des équations de Laplace inverses. À notre connaissance, la première publication du problème variationnel est faite par [Brackbill et Saltzman, 1982].*

Il existe une riche histoire derrière les deux termes f et g, et celle-ci fut plutôt négligée en infographie. La fonction f a été introduite dans la théorie des cartes quasi-conformes [Reshetnyak, 1966] de dimension d dans les années 60. [Ivanenko, 1988] a été le premier à utiliser le terme f tel quel pour le maillage 2D. Il a été introduit dans la communauté graphique par [Hormann et Greiner, 2000]. La première implémentation 3D de f pour la génération de maillage peut être attribuée à [Knupp, 2000a], cependant les premières implémentations 3D d'énergies similaires pour les déformations de solides remontent à 1988 [De Borst *et al.*, 1988]. Il semble que [Garanzha, 2000] ait été le premier à utiliser g pour le maillage.

Polyconvexité Notez que la fonction $F(J) = f(J) + \lambda g(J)$ n'est pas convexe, mais polyconvexe. La notion de polyconvexité est une généralisation de la notion de convexité pour les fonctions définies sur des espaces de matrices. Une fonction $\phi(J) : \mathbb{R}^{d\times d} \to \mathbb{R} \cup +\infty$ est dite polyconvexe [Ball, 1976] s'il existe une fonction convexe $\Phi(\#J)$, telle que $\phi(J) = \Phi(\#J)$, où $\#J$ désigne l'ensemble de tous les mineurs de J. Pour nous, il suffit de considérer uniquement de telles fonctions polyconvexes où la fonction étendue Φ est la fonction convexe de d^2+1 arguments et $\phi(J) = \Phi(J, \det J)$.

Toute fonction polyconvexe ϕ est convexe de rang un [Ball, 1976], c'est-à-dire que

$$\phi((1-\theta)J + \theta(J+\delta J)) \leq (1-\theta)\phi(J) + \theta\phi(J+\delta J),$$

où la variation δJ est définie telle que $\operatorname{rank}\delta J = 1$, et satisfait donc les conditions de Hadamard-Legendre (conditions d'ellipticité pour l'équation d'Euler-Lagrange de problèmes variationnels)

$$\sum_{i,k,j,m=1}^{d} \frac{\partial^2 \phi}{\partial(J)_{ij}\,\partial(J)_{km}} p_i p_k q_j q_m \geq 0$$

pour des vecteurs arbitraires $\vec{p}, \vec{q} \in \mathbb{R}^d$. Une preuve est donnée par le lemme 1 (voir remarque 5 dans l'analyse mathématique, sous-section 3.2.5).

Puisque F est une fonction polyconvexe satisfaisant les conditions d'ellipticité, elle est donc très bien adaptée à une optimisation numérique à condition que nous ayons une estimation initiale dans le domaine admissible.

10. Attention, cette étape suppose que la solution du problème (3.1) soit un difféomorphisme.

En élasticité finie, F qui est une somme de termes de distorsion de forme et de distorsion de volume, est appelée "division isochorique-volumétrique". L'idée d'une telle division remonte aux années 70 [Flory, 1961], [Penn, 1970], [De Borst *et al.*, 1988] et reste un sujet étudié en analyse de la convexité [Voss *et al.*, 2021].

3.1.4 Polyconvexité et ensemble des cartes admissibles

La fonction $F(X)$ possède une barrière infinie impénétrable sur la frontière de l'ensemble des maillages uniquement des éléments à volume positif, que l'on peut décrire de la façon suivante :

$$\frac{\text{vol}(P_k)}{\text{vol}(T_k)} > 0, k = 1, \ldots, \#T \tag{3.2}$$

avec T_k le simplexe k, et P_k son image par la carte X. Celui-ci est une approximation en dimension finie de l'ensemble

$$\det J > 0. \tag{3.3}$$

Cet ensemble a une structure assez compliquée. Pour le k-ième simplexe, $vol(P_k)$ est une fonction polylinéaire des coordonnées de ses sommets, donc chaque terme dans (3.2) définit un ensemble non convexe. On peut difficilement s'attendre à ce que l'ensemble (3.2) donne lieu à un domaine convexe. De plus, Ciarlet [Ciarlet et Geymonat, 1982] a prouvé que la propriété de barrière et la convexité de la densité d'énergie de déformation sont incompatibles. De sa déclaration, il résulte essentiellement que lorsque l'approximation par éléments finis $S(X)$ de l'énergie de déformation est bornée dans $\mathcal{A}$ et que

$$S(X) \to +\infty \text{ lorsque } X \in \mathcal{A}, X \to \partial\,\mathcal{A},$$

alors elle ne peut pas être une fonction convexe de X. Heureusement, les mesures de distorsion de barrière peuvent être polyconvexes, comme l'a montré J. Ball [Ball, 1976]. Par exemple, cela signifie que le domaine $\mathcal{A}$ est constitué de composantes connectées où tout couple de points de $\mathcal{A}$ peuvent être connectés. Considérons le vecteur $X + sY \in \mathbb{R}^{\#Vd}$, $0 \leq s \leq 1$, $X \in \mathcal{A}, X + Y \in \mathcal{A}$, et supposons que pour la $k^{\text{ième}}$ cellule de la maille, la matrice jacobienne de la maille définie par $X + sY$ s'écrit comme suit

$$J_k + sB_k, N - rankB_k = 1. \tag{3.4}$$

Puisque $-\det J$ est une fonction convexe de rang un de J, nous obtenons

$$-\det(J_k + sB_k) \leq -(1-s)\det J_k - s\det(J_k + B_k),$$

ce qui signifie qu'une telle déformation reste dans l'ensemble admissible quelque soit s. Nous considérons ci-dessous les sous-ensembles de $\mathcal{A}$ où toute paire de points est reliée par une séquence de segments de rang un.

L'ensemble admissible $\mathcal{A}$ peut avoir une structure assez compliquée. Même dans le cas de bords fixes, il peut contenir des sous-ensembles disjoints. Un exemple d'ensembles disjoints est montré sur la FIGURE 3.4. Pour un carré avec un trou carré, nous montrons 3 maillages admissibles avec les mêmes conditions sur le bord et la même connectivité, mais des rotations différentes du bord interne par rapport au bord externe. Il n'est pas possible de déformer un des maillages vers un autre sans inverser d'éléments, en conservant les sommets des bords fixes.

La FIGURE 3.4(d) montre trois composantes de la fonction barrière et une fonction de pénalité globale. Les sous-ensembles secondaires créent une sorte de "trous de lapin", minima locaux, avec de petits domaines d'attraction, ce qui signifie que si nous n'avons pas besoin d'un "sous-ensemble

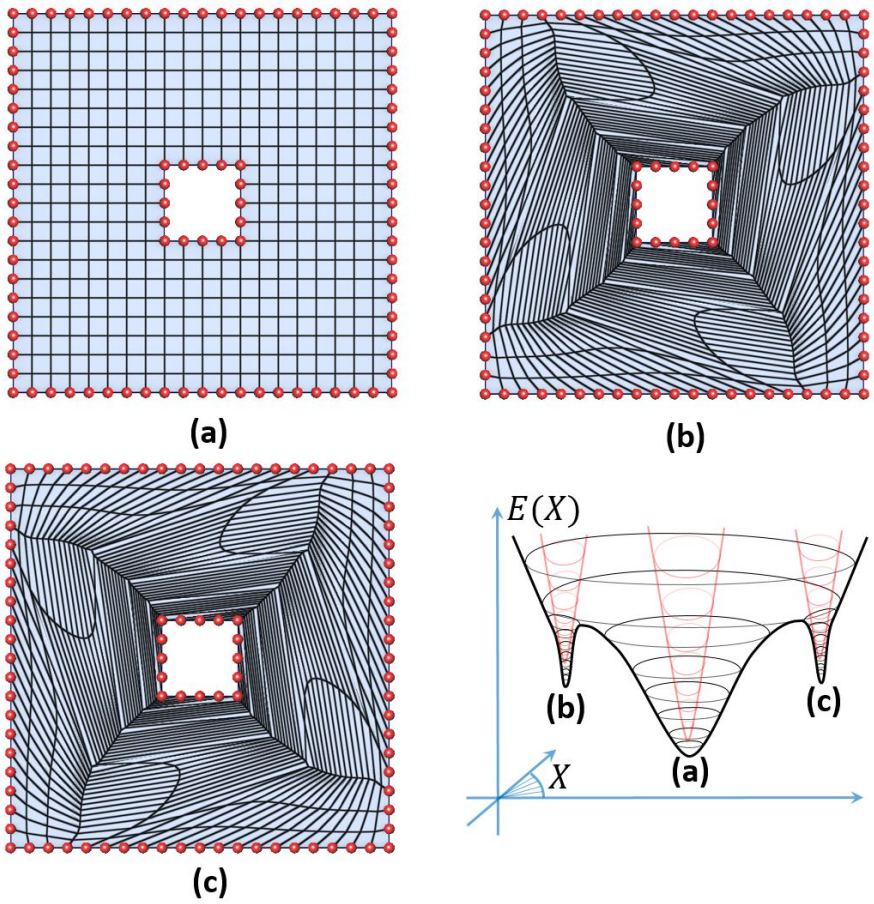


FIGURE 3.4 – Ensemble admissible avec composants disjoints : (a) maillage avec distorsion minimale, (b), (c) maillages admissibles avec les mêmes conditions aux bords, mais avec des d'enroulement opposés, (d) illustration de la fonction barrière $F(X, 0)$ (graphique rouge) avec sous-ensembles disjoints et "trous de lapin" correspondant à ces sous-ensembles sur le graphique noir de la fonction pénalisé $F(X, \varepsilon)$ (section suivante).

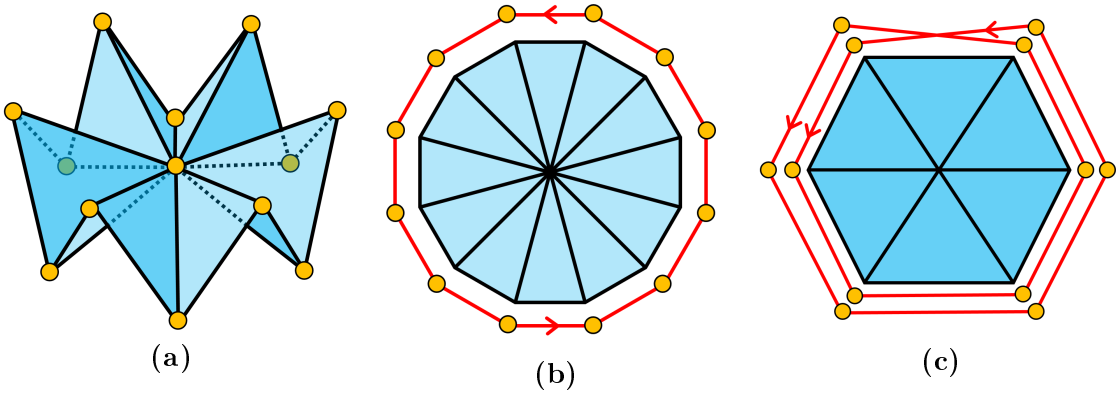


FIGURE 3.5 – Aplatissement à bord libre de surface. **(a) :** la surface à aplatir est constituée de 12 triangles équilatéraux. **(b) :** un aplatissement inversible correspondant à un minimum local de l'énergie élastique. **(c) :** minimum global de l'énergie élastique. Cet aplatissement est sans éléments inversés, cependant il n'est pas inversible au voisinage du sommet central.

principal" avec un grand domaine d'attraction, il faudra utiliser des algorithmes d'optimisation globale afin d'atteindre les "trous de lapin".

Pour le cas des déformations de mailles avec des frontières libres, la structure de $\mathcal{A}$ devient encore plus complexe. Le nombre d'ensembles disjoints peut fortement augmenter. De plus, des ensembles disjoints parasites de faible énergie peuvent apparaître, ce qui constitue de véritables pièges pour les algorithmes de démêlage. Considérons l'aplatissement du voisinage d'un point constitué de 12 triangles réguliers autour d'un point, voir FIGURE 3.5(a). Alors que l'aplatissement standard serait un dodécagone régulier, illustré par la FIGURE 3.5(b), l'énergie de déformation minimale est fournie par le "2-covering", illustré par la FIGURE 3.5(c).

L'homéomorphisme entre ces deux solutions n'existe pas, donc elles appartiennent définitivement à des sous-ensembles disjoints de $\mathcal{A}$. La deuxième solution définit un aplatissement qui n'est pas localement inversible puisqu'on ne peut pas trouver de petit voisinage ouvert du sommet central qui puisse être mis en correspondance unique avec le voisinage ouvert de la projection plane. Nous parlons plus en détail de ce problème et proposons une solution pratique pour l'éviter dans la sous-section 3.4.1.

3.2 Récupérer la bijectivité d'une carte via pénalisation [Garanzha *et al.*, 2021a]

Que ce soit pour les polycubes, ou pour des cartes en général, il est récurrent que la construction d'une carte admissible ne soit pas triviale. Cela rend l'utilisation de l'énergie (3.1) compliquée. Notre première contribution, présentée dans [Garanzha *et al.*, 2021a], est une méthode numérique simple qui permet de calculer une carte, et cela même quand l'initialisation de la carte n'est pas admissible, et même arbitrairement mauvaise (voir sous-section 3.2.4). Notre méthode surpasse l'état de l'art récent sur la construction de cartes localement injectives [Du *et al.*, 2020] en termes de robustesse, de qualité et de caractéristiques supportées.

L'idée principale de la méthode est de partir d'un maillage arbitraire et d'arriver directement au maillage optimal dans l'ensemble admissible, et donc "du bon côté de la barrière". Nous introduisons une régularisation du problème (3.1) qui permet de travailler lorsque la carte n'est pas admissible, tout en pénalisant de façon progressive les éléments retournés. Si les concepts généraux utilisés dans notre algorithme (déformation d'un matériau hyperélastique et pénalisation des éléments inversés) sont connus depuis les années 80, le passage à un algorithme fonctionnel n'est pas trivial. Nous sommes les premiers à apporter réellement des garanties pour le problème du démêlage de maillages.

En résumé, notre principale contribution est une recette dont les ingrédients sont soigneusement choisis pour obtenir des sous-problèmes bien posés : nous présentons la toute première stratégie de pénalisation avec des garanties théoriques de possibilité de démêlage en un nombre fini d'étapes. Nous prouvons que notre choix de la séquence des paramètres de régularisation garantit qu'un algorithme de minimisation [11] peut trouver un maillage exempt d'éléments inversés en un nombre fini d'étapes. En plus de cela, l'énergie proposée est "sympathique" à optimiser, plus précisément, nous majorons l'énergie de déformation et nous montrons que la partie définie positive de la matrice hessienne est spectralement équivalente à la matrice de rigidité des éléments finis pour l'opérateur de Laplace. Cela garantit l'absence de problèmes extrêmement difficiles lors du franchissement de la barrière.

11. L'algorithme de minimisation est soumis aux conditions du théorème (1), sous-section 3.2.5 ; d'après nos expériences numériques nous observons que notre algorithme de minimisation respecte presque toujours les conditions.

Enfin le dernier point, mais non le moindre, est le "en 50 lignes de code" du titre l'article [Garanzha *et al.*, 2021a]. La solution que nous proposons est simple et ne nécessite pratiquement aucun réglage de la part de l'utilisateur, ce qui facilite l'adoption de notre approche cartographique par un large éventail d'applications potentielles. Pour faciliter la reproductibilité, nous proposons ce code en 50 lignes de python (voir Listing 3.1) qui présente une utilisation de base. Avec ça, nous avons publié sur github un version libre du code C++ pour les applications plus avancées.

Le reste de la section est organisée comme suit : nous commençons par présenter la méthode de pénalité (sous-section 3.2.1), ensuite nous détaillons des calculs utiles pour l'implementation (sous-section 3.2.2), puis nous évaluons ses performances avec tout un ensemble de comparaisons (§ 3.2.4) et enfin nous discutons des propriétés théoriques de l'approche (sous-section 3.2.5) avant de finir sur ses limitations (sous-section 3.2.6).

3.2.1 Méthode de pénalisation

Le problème (3.1) est connu depuis des décennies, il fournit un outil simple et efficace pour optimiser la qualité d'une carte. Comme mentionné précédemment, c'est une fonction polyconvexe, ce qui signifie que les équations d'Euler-Lagrange pour la déformation optimale satisfont les conditions d'ellipticité. Elle est donc très bien adaptée à une optimisation numérique, à condition que nous ayons une initialisation dans le domaine admissible $\min_{\Omega} J(\vec{u}) > 0$.

Cependant, bien que théoriquement valable, cet énoncé du problème n'offre aucun moyen pratique de se débarrasser des éléments inversés dans une carte. En effet, pour une carte hors de l'ensemble admissible, l'énergie est infinie et ne fournit aucune indication sur la façon d'améliorer la situation. Nous proposons donc de modifier légèrement l'énoncé du problème.

L'optimisation numérique est difficile en raison des dénominateurs nuls dans les termes f et g. Par exemple, pour les cartes quasi-conformes en 2D, le coefficient de dilatation permet [Weber *et al.*, 2012] de les éviter. Ici, nous avons besoin d'une stratégie plus générale de pénalisation des éléments inversés. L'idée remonte à [Ivanenko, 1988]. Nous améliorons (sous-section 3.2.5) la technique de pénalisation heuristique proposée dans [Garanzha et Kaporin, 1999]. A savoir, nous utilisons une fonction de régularisation χ pour une valeur positive de ε (FIGURE 3.6) :

$$\chi(D,\varepsilon) := \frac{D+\sqrt{\varepsilon^2+D^2}}{2}, \tag{3.5}$$

Nous définissons ensuite des versions régularisées $f_\varepsilon, g_\varepsilon$ des fonctions f et g :

$$f_\varepsilon(J) := \frac{\operatorname{tr} J^\top J}{(\chi(\det J,\varepsilon))^{\frac{2}{d}}}, \qquad g_\varepsilon(J) := \frac{\det^2 J+1}{\chi(\det J,\varepsilon)}, \tag{3.6}$$

Et le problème (3.1) prend la forme suivante :

$$\lim_{\varepsilon\to 0^+} \arg\min_{\vec{u}} \int_\Omega \left(f_\varepsilon(J)+\lambda g_\varepsilon(J)\right) dx. \tag{3.7}$$

Sous certaines hypothèses,[12] les solutions du problème (3.7) sont des solutions du problème (3.1). Mais, en plus de cela, le problème (3.7) offre un moyen de se débarrasser des inversions si l'on ne possède pas d'initialisation admissible.

12. le fait que la solution du problème (3.7) soit un difféomorphisme est suffisant (mais pas nécessaire) pour l'équivalence.

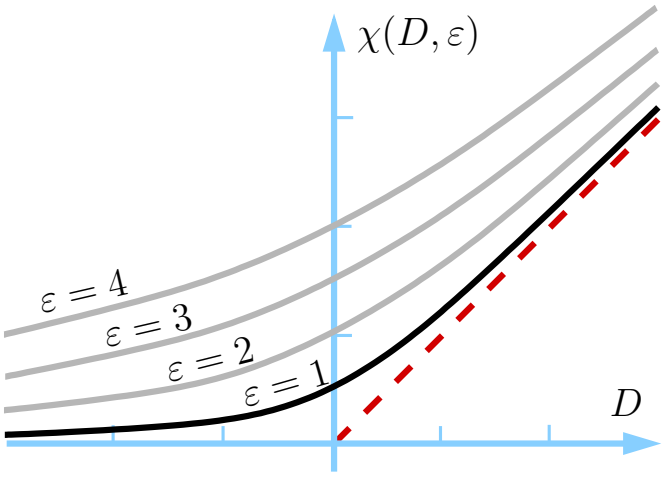


FIGURE 3.6 – Fonction de régularisation pour le dénominateur dans l'équation (3.6). Lorsque ε tend vers zéro, $\chi(\varepsilon, D)$ tend vers D pour les valeurs positives de D, et vers 0^+ pour les valeurs négatives de D.

En pratique, la carte $\vec{u}$ est affine par morceaux, la matrice jacobienne J est donc constante par morceaux, et peut être représentée par les coordonnées des sommets du domaine paramétrique $\{\vec{u}_i\}_{i=1}^{\#V}$. Désignons le vecteur de toutes les variables par $U := \left(\vec{u}_1^\top \dots \vec{u}_{\#V}^\top\right)^\top$, alors notre problème d'optimisation peut être discrétisé comme suit :

$$\lim_{\varepsilon \to 0^+} \arg\min_U F(U, \varepsilon), \tag{3.8}$$
$$\text{où} \quad F(U,\varepsilon) := \sum_{t=1}^{\#T} \left(f_\varepsilon(J_t) + \lambda g_\varepsilon(J_t)\right) \mathrm{vol}(T_t),$$

$\#V$ est le nombre de sommets, $\#T$ est le nombre de simplexes, J_t est la matrice jacobienne pour le simplexe t et $vol(T_t)$ est le volume du simplexe T_t dans le domaine original.

3.2.2 Schéma de résolution

Pour résoudre le problème (3.8), nous utilisons une méthode de descente itérative. Nous partons d'une estimation initiale U^0, et nous construisons une séquence d'approximations $U^{k+1} := U^k + \Delta U^k$. Pour chaque itération, nous devons choisir soigneusement le paramètre de régularisation ε^k. En partant de $\varepsilon^0 := \sqrt{10^{-12} + 4 \cdot 10^{-2} \cdot \left[\min(0, D_-^0)\right]^2}$ [13], nous définissons la séquence suivante :

$$\varepsilon^{k+1} := \begin{cases} 2\sqrt{\mu^k(\mu^k - D_-^{k+1})} & \text{if } D_-^{k+1} < \mu^k \\ 0 & \text{if } D_-^{k+1} \geq \mu^k, \end{cases} \tag{3.9}$$

où $D_-^{k+1} := \min_{t \in 1 \dots \#T} \det J_t^{k+1}$ est la valeur minimale du déterminant jacobien sur toutes les cellules du maillage à l'itération $k+1$, $\sigma^k := \max\left(\frac{1}{10}, 1 - \frac{F(U^{k+1},\varepsilon^k)}{F(U^k,\varepsilon^k)}\right)$ est le coefficient de descente et $\mu^k := (1-\sigma^k)\chi(D_-^{k+1}, \varepsilon^k)$. Cette formule est justifiée par le théorème 1 (§ 3.2.5) qui montre

13. valeur trouvée expérimentalement, proposée dans [Garanzha et Kaporin, 1999]

qu'avec cette suite, et une optimisation idéalisée, on obtiendra une carte admissible en un nombre fini d'itérations (s'il en existe une, bien sûr).

Algorithme 1 : Calcul d'une carte admissible

Entrées : U^0 ; // *initialisation (vecteur de taille $\#V \times d$)*
Entrées : *utiliserQuasiNewton* ; // *booléen pour choisir le schéma d'optimisation*
Output : U ; // *carte finale admissible (vecteur de taille $\#V \times d$)*
$k \leftarrow 0$; **répéter**
  calculer ε^k ; // *paramètre de régularisation, équation* (3.9)
  si *utiliserQuasiNewton* **alors**
    $U^{k+1} \leftarrow$ L-BFGS (U^k, ε^k) ; // *boucle L-BFGS interne*
  sinon
    calcul d'une matrice hessienne modifiée $H^+(U^k, \varepsilon^k)$;
    $\Delta U^k \leftarrow (H^+)^{-1} \nabla F(U^k, \varepsilon^k)$;
    $U^{k+1} \leftarrow \arg\min_{\tau} F(U^k + \tau\, \Delta U^k, \varepsilon^k)$; // *recherche linéaire (line search)*
  $k \leftarrow k + 1$;
jusqu'à $\min_{t \in 1 \dots \#T} \det J_t^k > 0$ ***et*** $F(U^k, \varepsilon^k) > (1 - 10^{-3})\, F(U^{k-1}, \varepsilon^{k-1})$;
$U \leftarrow U^k$;

Une description détaillée du schéma de résolution est donnée à l'algorithme 1. La manière la plus simple de trouver ΔU^k est d'appeler un solveur quasi-newtonien tel que L-BFGS [Zhu *et al.*, 1997]. La seule chose que nous devons implémenter est le calcul de la fonction $F(U^k, \varepsilon^k)$ et de son gradient $\nabla F(U^k, \varepsilon^k)$, détaillé dans la section suivante. Pour illustrer cette simplicité, le listing 3.1 et la FIGURE 3.7 montrent un exemple complet fonctionnel d'une implémentation en langage Python, et cela en moins de 50 lignes de codes, avec les entrées/sorties correspondantes générées par le code.

Remarque 3. *Notre méthode n'est pas limitée aux maillages simpliciaux : sur le listing 3.1, nous évaluons la matrice jacobienne pour chaque triangle formant des coins du quadrilatère, ce qui correspond à la règle de quadrature trapézoïdale. Notre énergie est définie continûment, et peut être utilisée avec des discrétisations arbitraires. Nous montrons une application sur des hexaèdres dans la sous-section 3.4.2.*

Il devrait également être possible d'adapter des solveurs spécialisés dans les problèmes de traitement de la géométrie [Zhu *et al.*, 2018, Shtengel *et al.*, 2017, Smith *et al.*, 2019]. Une autre option consiste à calculer analytiquement la matrice hessienne au lieu de l'estimer et ensuite d'utiliser une méthode de Newton pour obtenir ΔU^k. Le problème, cependant, est que la matrice hessienne $\frac{\partial^2 F}{\partial U\, \partial U^\top}$ n'est pas définie positive, ce qui rend l'optimisation incertaine, voire impossible. Nous proposons une approximation de cette matrice qui elle sera garantie comme étant définie positive.

Remarque 4. *Le fait que la matrice hessienne $\frac{\partial^2 F}{\partial U\, \partial U^\top}$ ne soit pas définie positive implique que l'énergie* (3.7) *n'est pas polyconvexe lorsque $\varepsilon \neq 0$. Pour autant, nous proposons une approximation H^+ définie positive et donc nous montrons que celle-ci possède une qualité spectrale très proche d'une matrice éléments finis. Donc bien que nous n'ayons pas de garanties théoriques aussi fortes que pour* (3.1), (3.7) *sera en pratique facile à optimiser.*

Listing 3.1 – Exemple complet d'une implémentation de démêlage utilisant L-BFGS sur un maillage quadrangulaire, une exemple de résultat est montré à la FIGURE 3.7–droite.

```
from mesh import Mesh
import numpy as np
from scipy.optimize import fmin_l_bfgs_b

mesh, n = Mesh(), Mesh().nverts() # generate a test quad mesh; this untangling routine is not limited to regular grids
eps = 1 # the regularization parameter ε^0
Q = [ np.matrix('-1,-1;1,0;0,0;0,1'), np.matrix('-1,0;1,-1;0,1;0,0'),  # quadratures for every quad corner (trapezoidal rule):
      np.matrix('0,0;0,-1;1,1;-1,0'), np.matrix('0,-1;0,0;1,0;-1,1') ] # evaluate the Jacobian matrix J(u) four times per quad

def jacobian(U, qc, quad): # evaluate J(u) at the given quadrature point
    return np.matrix([[U[quad[0]  ], U[quad[1]  ], U[quad[2]  ], U[quad[3]  ]],            # u-coordinates
                      [U[quad[0]+n], U[quad[1]+n], U[quad[2]+n], U[quad[3]+n]]]) * Q[qc]   # v-coordinates
def energy(U): # compute the energy F and its gradient ∇F for the map u, Eq. (5)
    F, G = 0, np.zeros(2*n) # zero out the energy and the gradient
    for quad in mesh.quads: # sum over all quads
        for qc in range(4): # for every quad corner
            J = jacobian(U, qc, quad) # evaluate the Jacobian matrix J(u)
            det = np.linalg.det(J) # det J
            chi  = det/2 + np.sqrt(eps**2 + det**2)/2    # the penalty function χ(ε^k, det J)
            chip = .5 + det/(2*np.sqrt(eps**2 + det**2)) # its derivative χ'(ε^k, det J)
            f = np.trace(np.transpose(J)*J)/chi # quad corner shape quality
            F += f # add the term to the energy
            dfdj = (2*J - np.matrix([[J[1,1],-J[1,0]],[-J[0,1],J[0,0]]])*f*chip)/chi # ∂f_ε/∂a : derivative w.r.t the Jacobian
            dfdu = Q[qc] * np.transpose(dfdj) # chain rule for the real variables (Appendix A)
            for i,v in enumerate(quad): # add the contribution to ∇F from current simplex
                if (mesh.boundary[v]): continue # the boundary verts are locked
                G[v  ] += dfdu[i,0]
                G[v+n] += dfdu[i,1]
    return F, G

while True: # outer L-BFGS loop, Alg. 1, line 2
    [Fprev, _] = energy(mesh.x) # compute F(U^k, ε^k)
    [mesh.x, F, _] = fmin_l_bfgs_b(energy, mesh.x, factr=1e12) # inner L-BFGS loop, compute F(U^{k+1}, ε^k)
    mindet = min([ np.linalg.det( jacobian(mesh.x, qc, quad) ) for quad in mesh.quads for qc in range(4) ]) # min det J^k
    mu = min(F/Fprev, 9e-1)*(mindet + np.sqrt(eps**2 + mindet**2))/2 # μ^k, Eq. (6)
    eps = 2*np.sqrt(mu*(mu-mindet)) if mindet<mu else 0 # ε^{k+1}, Eq. (6)
    if (mindet>0 and F>999e-3*Fprev): break # stopping criterion, Alg. 1, line 11
print(mesh) # print wavefront .obj file
```

Listing 3.2 – Classe basique de maillage quadrangulaire pour le Listing 3.1

```
import numpy as np
class Mesh():
    def __init__(self,m=8): # generate the test problem: a regular 2d grid with upper half shifted (Fig. 3)
        self.x = [   i/m + int(j>=m//2)*3/5 for j in range(m) for i in range(m) ] + \
                 [ 2*j/m - int(j>=m//2)*3/5 for j in range(m) for i in range(m) ]       # 2D geometry
        self.quads = [ [i+j*m, i+1+j*m, i+1+(j+1)*m, i+(j+1)*m] for j in range(m-1) for i in range(m-1) ] # connectivity
        self.boundary = [ i==0 or i==m-1 or j==0 or j==m-1  for j in range(m) for i in range(m) ] # vertex boundary flags
    def nverts(self):
        return len(self.x)//2
    def __str__(self): # wavefront .obj output
        return ''.join("v %f %f 0\n" % (self.x[v], self.x[v+self.nverts()]) for  v in range(self.nverts())) + \
               ''.join("f %d %d %d %d\n" % (f[0]+1, f[1]+1, f[2]+1, f[3]+1) for f in self.quads)
```

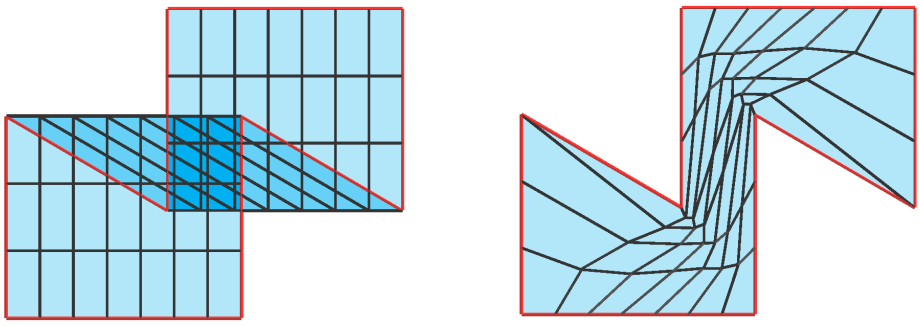


FIGURE 3.7 – Le maillage d'entrée avec des éléments inversés et le démêlage produit par les listings 3.1 et 3.2. **Gauche :** un maillage 2D quadrangulaire à démêler. La frontière (en rouge) est verrouillée, et le maillage noir est libre de se déplacer. **Droite :** résultat sans éléments inversés.

Newton et matrice hessienne

La matrice hessienne modifiée $H^+(U^k, \varepsilon^k)$ de la fonction F par rapport à U au point U^k est construite à partir de $d \times d$ blocs (d étant la dimension du problème)

$$H_{ij}^+ \approx \frac{\partial^2 F}{\partial \vec{u}_i \, \partial \vec{u}_j^\top}(U^k, \varepsilon^k).$$

Ici, la matrice H_{ij}^+ est placée à l'intersection de la ligne du bloc i et de la colonne du bloc j ; le symbole $\approx$ signifie que nous supprimons tous les termes dépendants de la dérivée seconde de χ et des dérivées secondes de $\det J$ pour garder H^+ définie positivement.

Rappelons que dans notre schéma de résolution, nous utilisons la matrice hessienne modifiée $(d\#V) \times (d\#V)$ $H^+(U, \varepsilon)$ de la fonction $F(U, \varepsilon)$ construite à partir de $d \times d$ blocs H_{ij}^+, placés à l'intersection de la ligne du $i^{\text{ème}}$ bloc et de la colonne du $j^{\text{ème}}$ bloc. Il est courant d'ajouter des termes de régularisation à la matrice hessienne pour la rendre définie positive, mais nous proposons de modifier la procédure d'assemblage des matrices d'éléments finis (EF) en éliminant certains termes pouvant conduire à une matrice EF indéfinie.

À cette fin, nous limitons notre attention à un seul simplexe et nous étudions une fonction $\phi(J)$ de la matrice jacobienne définie comme suit :

$$\phi(J) := f_\varepsilon(J) + \lambda g_\varepsilon(J) = \frac{\operatorname{tr} J^\top J}{(\chi(\det J, \varepsilon))^{\frac{2}{d}}} + \lambda \frac{\det^2 J + 1}{\chi(\det J, \varepsilon)}. \tag{3.10}$$

Désignons par $a \in \mathbb{R}^{d^2}$ l'aplatissement (en colonnes) de la matrice jacobienne J, c'est-à-dire le vecteur composé des éléments de J. Nous décomposons la matrice hessienne $d^2 \times d^2$ de ϕ par rapport aux entrées de la matrice jacobienne en deux parties : $\frac{\partial^2 \phi}{\partial a \, \partial a^\top} = M^+ + M^\pm$, où M^+ est une matrice définie positive, et $M^\pm$ peut être une matrice indéfinie que nous négligeons. La matrice $M^\pm$ contient tous les termes dépendant de χ'' et les dérivées secondes de $\det J$ par rapport aux éléments de la matrice jacobienne J. Notre carte est affine sur le simplexe d'intérêt, donc sa matrice jacobienne J est une fonction linéaire des sommets du simplexe. L'idée est de calculer une matrice définie positive $M^+(J)$, et d'utiliser la règle de dérivation d'une fonction composée (*chain rule*) pour obtenir la matrice hessienne par rapport à nos variables U et assembler la matrice H^+.

Ainsi, nous choisissons un point arbitraire J_0 et nous voulons montrer comment décomposer $\frac{\partial^2 \phi}{\partial a \, \partial a^\top}(J_0)$ en une somme de $M^+(J_0)$ et $M^\pm(J_0)$ avec $M^+(J_0) > 0$. Pour ce faire, nous commençons par écrire le développement de Taylor du premier ordre $q(D)$ de la fonction $\chi(D, \varepsilon)$ autour d'un point $D_0 = \det J_0$:

$$q(D) := \chi(D_0, \varepsilon) + \frac{\partial \chi}{\partial D}(D_0, \varepsilon)(D - D_0).$$

Ensuite, nous définissons une fonction $\Phi(a, D)$ comme suit :

$$\Phi(a, D) := \frac{|a|^2}{(q(D))^{\frac{2}{d}}} + \lambda \frac{D^2 + 1}{q(D)}.$$

Notez que Φ diffère un peu de ϕ : il a un argument de plus et χ est remplacé par sa linéarisation au dénominateur. Si cette manœuvre peut sembler obscure, la lumière va être faite très prochainement. La fonction Φ a la vertu majeure d'être convexe :

Lemme 1. $\nabla\nabla^\top\Phi > 0$

Démonstration. Il est facile de voir que la matrice hessienne $(d^2+1)\times(d^2+1)$ de Φ peut être écrite dans la représentation par blocs de 2×2 :

$$\nabla\nabla^\top\Phi = P + \lambda Q,$$

où

$$P := \begin{pmatrix} \frac{2}{q^{\frac{2}{d}}} I & -\frac{4}{d}\frac{q' a}{q^{1+\frac{2}{d}}} \\ -\frac{4}{d}\frac{q' a^\top}{q^{1+\frac{2}{d}}} & \frac{2}{d}(1+\frac{2}{d})\frac{|a|^2 q'^2}{q^{2+\frac{2}{d}}} \end{pmatrix} \quad \text{et}$$

$$Q := \begin{pmatrix} 0 & 0 \\ 0 & \frac{2}{q} - 4D\frac{q'}{q^2} + 2(1+D^2)\frac{q'^2}{q^3} \end{pmatrix}.$$

Il est trivial de vérifier que $Q \geq 0$, puisque Q_{22} est une fonction quadratique strictement positive d'argument D. Puisque les blocs de tête de la matrice P sont définis positifs et que le complément de Schur

$$P_{22} - P_{21}P_{11}^{-1}P_{12} = \frac{|a|^2}{q^{2+\frac{2}{d}}}\frac{2}{d}\left(1-\frac{2}{d}\right) \geq 0$$

est définie positif, la convexité globale de Φ est établie. ■

Remarque 5. *Notons que nous venons de prouver la convexité de la fonction* Φ. *Comme conséquence immédiate, nous obtenons la polyconvexité de la fonctionnelle* (3.1), *car elle est un cas particulier de notre fonctionnelle* (3.7) *avec* $\chi = q$.

Après avoir construit une fonction convexe Φ, on utilise la décomposition suivante :

$$\frac{\partial^2\phi}{\partial a\,\partial a^\top}(J_0) = M^+(J_0) + M^\pm(J_0), \tag{3.11}$$

où

$$M^+ := \begin{pmatrix} I & \frac{\partial D}{\partial a} \end{pmatrix} \begin{pmatrix} \frac{\partial^2\Phi}{\partial a\partial a^\top} & \frac{\partial^2\Phi}{\partial a\partial D} \\ \frac{\partial^2\Phi}{\partial D\,\partial a^\top} & \frac{\partial^2\Phi}{\partial D^2} \end{pmatrix} \begin{pmatrix} I & \frac{\partial D}{\partial a} \end{pmatrix}^\top, \text{ et}$$

$$M^\pm := \frac{\partial\Phi}{\partial D}\frac{\partial^2 D}{\partial a\,\partial a^\top} - \frac{\chi''}{\chi}\left(\frac{2}{d}f_\varepsilon + \lambda g_\varepsilon\right)\frac{\partial D}{\partial a}\frac{\partial D}{\partial a^\top}.$$

La façon la plus simple de vérifier que l'égalité (3.11) est vraie est de noter qu'au point J_0 on a $q = \chi$, $q' = \chi'$, et donc nous avons

$$\frac{\partial\phi(a)}{\partial a} = \frac{\partial\Phi(a, D(a))}{\partial a} + \frac{\partial\Phi(a, D(a))}{\partial D}\frac{\partial D(a)}{\partial a}.$$

Pour calculer le hessien $\frac{\partial^2\phi}{\partial a\,\partial a^\top}(J_0)$, il suffit de différencier cette expression une fois de plus et d'ajouter les termes en χ'' qui ont été mis à zéro par la linéarisation.

En résumé, dans nos calculs, pour chaque simplexe nous approximons la matrice hessienne $\frac{\partial^2\phi}{\partial a\,\partial a^\top}$ par la matrice $d^2\times d^2$ M^+ et nous négligeons le terme $M^\pm$. Grâce à la convexité de Φ, il est trivial de vérifier que pour tout choix de J_0 la matrice M^+ est définie positive. Ensuite, nous utilisons *chain rule* sur M^+ pour obtenir la matrice hessienne par rapport à nos variables U, et nous assemblons une approximation $(d\#V)\times(d\#V)$ H^+ de la matrice hessienne pour

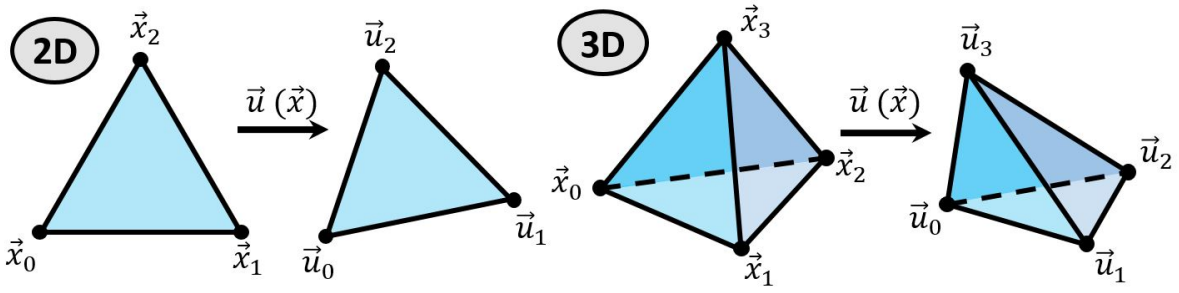


FIGURE 3.8 – Sur chaque simplexe, la carte $\vec{u}(\vec{x})$ est affine et est entièrement définie par la position des sommets du simplexe du domaine $\{\vec{x}_i\}$ et son image $\{\vec{u}_i\}$.

la fonction d'énergie $F(U, \varepsilon)$. La matrice H^+ est définie positive à condition qu'au moins d sommets de maille soient fixés. Si moins de d points sont fixés, les transformations de corps rigides sont autorisées. L'énergie est invariante par rapport aux transformations du corps rigide, donc lorsque les contraintes permettent de telles transformations, la matrice H^+ devient semi-définie positive. Notez que les blocs de tête H^+_{ii} sont toujours définis positifs. Nous détaillons dans la sous-section 3.2.3 la procédure d'assemblage par éléments finis. De plus, nous donnons dans la sous-section 3.2.5 une preuve que la partie définie positive de la matrice hessienne est spectralement équivalente à la matrice de rigidité des éléments finis pour l'opérateur de Laplace.

3.2.3 Formules et calculs

L'énergie a jusque-là été exprimée par rapport à la jacobienne J de la carte, or pour l'optimisation, nous cherchons un gradient et une hessienne par rapport aux positions des points du maillage dans la carte. Pour cela nous introduisons un changement de variable qui permet, avec des compositions des dérivés, d'obtenir les formules nécessaires pour l'optimisation.

Étant donné une carte $\vec{u}$, désignons par $\vec{a}_i, \quad i = 1, 2\ (, 3)$ la base tangente, c'est-à-dire les vecteurs formant les colonnes de la matrice jacobienne J. Par exemple, en 2D, nous avons $\vec{a}_1 := \left(\frac{\partial u}{\partial x} \quad \frac{\partial v}{\partial x}\right)^\top$ et $\vec{a}_2 := \left(\frac{\partial u}{\partial y} \quad \frac{\partial v}{\partial y}\right)^\top$. Désignons par $\vec{b}_i$ la base duale, c'est-à-dire les vecteurs choisis de telle sorte que $\vec{a}_i^\top \vec{b}_j = \delta_{ij} \det J$ pour tous les indices i, j. En particulier, pour les paramètres 2D, la base duale peut être écrite comme suit : $\vec{b}_1 := \left(\frac{\partial v}{\partial y} \quad -\frac{\partial u}{\partial y}\right)^\top$. et $\vec{b}_2 := \left(-\frac{\partial v}{\partial x} \quad \frac{\partial u}{\partial x}\right)^\top$. Dans le cas 3D, $\vec{b}_k = \vec{a}_i \times \vec{a}_j$, où i, j, k est une permutation cyclique de $1, 2, 3$. Le choix des variables est pratique, en particulier, $\operatorname{tr} J^\top J = \sum_i |\vec{a}_i|^2$ et $\frac{\partial \det J}{\partial \vec{a}_i} = \vec{b}_i$. Pour simplifier encore les notations, nous utiliserons χ pour $\chi(D, \varepsilon)$, χ' pour $\frac{\partial \chi(D,\varepsilon)}{\partial D}$ et $a^\top = (\vec{a}_1^\top \dots \vec{a}_d^\top), b^\top = (\vec{b}_1^\top \dots \vec{b}_d^\top)$.

Gradient

Afin de dériver des expressions pour le gradient et la matrice hessienne de F, on écrit explicitement la matrice jacobienne J pour la carte affine d'un simplexe T avec des sommets $\vec{u}_0, \vec{u}_1, \dots, \vec{u}_d$:

$$\begin{aligned} J = (\vec{a}_1 \dots \vec{a}_d) &= (\vec{u}_1 - \vec{u}_0; \vec{u}_2 - \vec{u}_0; \dots; \vec{u}_d - \vec{u}_0)\, S^{-1} \\ &= (\vec{u}_0 \dots \vec{u}_d) Z, \end{aligned}$$

où

$$S := (\vec{x}_1 - \vec{x}_0; \vec{x}_2 - \vec{x}_0; \dots; \vec{x}_d - \vec{x}_0),\ \det S > 0$$

est la matrice de forme, $\vec{x}_i$ sont les sommets de forme "idéale" ou "cible" [14] pour l'image du simplexe T, et Z est une matrice $(d+1) \times d$ définie comme suit

$$Z := \{z_{ij}\} := \begin{pmatrix} -1 & \dots & -1 \\ I & & \end{pmatrix} S^{-1}$$

Comme la matrice jacobienne est une fonction linéaire de $\vec{u}_i$, on a

$$\frac{\partial \vec{a}_i}{\partial \vec{u}_j^\top} = z_{ji} I, \quad i = 1, \dots, d,\ j = 0, \dots, d.$$

En notant $g_0 \cdots g_d$ les indices globaux des $d+1$ sommets du simplexe T , on peut écrire la contribution (additive) du gradient de F sur le simplexe de la façon suivante :

$$\begin{aligned}(\nabla F)_{g_j} \mathrel{+}= & \frac{\det S}{d!} \sum_{i=1}^{d} \frac{\partial \vec{a}_i^\top}{\partial \vec{u}_j} \frac{\partial \phi}{\partial \vec{a}_i} \\ = & \frac{\det S}{d!} \sum_{i=1}^{d} z_{ji} \frac{\partial \phi}{\partial \vec{a}_i}, \qquad j = 0, \dots, d,\end{aligned}$$

où la fonction $\phi(J)$ est définie par l'équation (3.10) et son gradient $\frac{\partial \phi}{\partial \vec{a}_i}$ explicite est :

$$\frac{\partial \phi}{\partial \vec{a}_i} = \frac{2}{\chi^{\frac{2}{d}}} \vec{a}_i - \frac{1}{\chi} \left(\frac{2}{d} f_\varepsilon \chi' - 2\lambda \det J + \lambda g_\varepsilon \chi' \right) \vec{b}_i.$$

Matrice hessienne

Les blocs de la partie définie positive de la matrice hessienne de F peuvent être mis à jour à l'aide de la formule générale suivante

$$H^+_{g_j g_i} \mathrel{+}= \frac{\det S}{d!} \sum_{m,l} \frac{\partial \vec{a}_m^\top}{\partial \vec{u}_j} M^+_{ml} \frac{\partial \vec{a}_l}{\partial \vec{u}_i^\top},$$

où M^+_{ml} désigne un bloc de $d \times d$ de la matrice $d^2 \times d^2$ définie positive M^+ définie dans l'équation (3.11). L'expression explicite de la matrice est la suivante :

$$M^+ = \begin{pmatrix} I & b \end{pmatrix} \begin{pmatrix} \frac{\partial^2 \Phi}{\partial a \partial a^\top} & \frac{\partial^2 \Phi}{\partial a \partial D} \\ \frac{\partial^2 \Phi}{\partial D \partial a^\top} & \frac{\partial^2 \Phi}{\partial D^2} \end{pmatrix} \begin{pmatrix} I & b \end{pmatrix}^\top, \text{ où}$$

$$\begin{aligned}\frac{\partial^2 \Phi}{\partial a \partial a^\top} &= \frac{2}{\chi^{\frac{2}{d}}} I, \\ \frac{\partial^2 \Phi}{\partial D^2} &= \frac{2}{d} \left(1 + \frac{2}{d}\right) |a|^2 \frac{\chi'^2}{\chi^{2+\frac{2}{d}}} + \lambda \left(\frac{2}{\chi} - 4D \frac{\chi'}{\chi^2} + 2(1+D^2) \frac{\chi'^2}{\chi^3} \right), \\ \frac{\partial^2 \Phi}{\partial a \partial D} &= -\frac{4}{d} \frac{\chi'}{\chi^{1+\frac{2}{d}}} a\,.\end{aligned}$$

14. S est équivalente à la matrice B_i^X avec les notations du chapitre 2. De la même façon : $(\vec{u}_1 - \vec{u}_0; \vec{u}_2 - \vec{u}_0; \dots; \vec{u}_d - \vec{u}_0) = B_i^U$

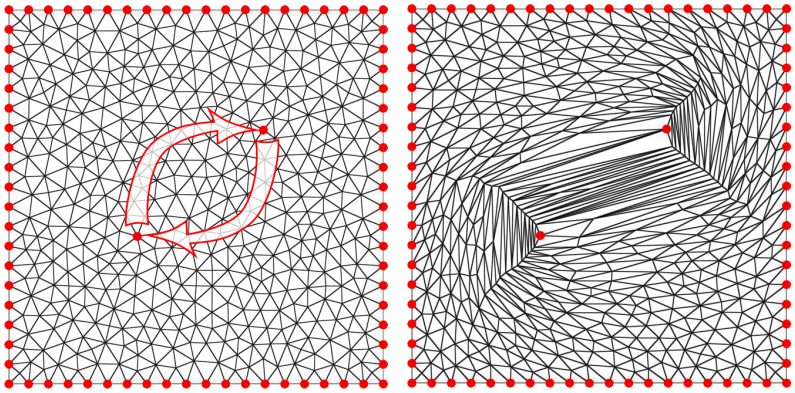


FIGURE 3.9 – Vérification de la validité de la cartographie injective : échange de deux points quelconques à l'intérieur d'un carré. À gauche : le problème d'entrée, tous les points verrouillés sont indiqués en rouge. À droite : résultat sans repliement obtenu avec notre méthode.

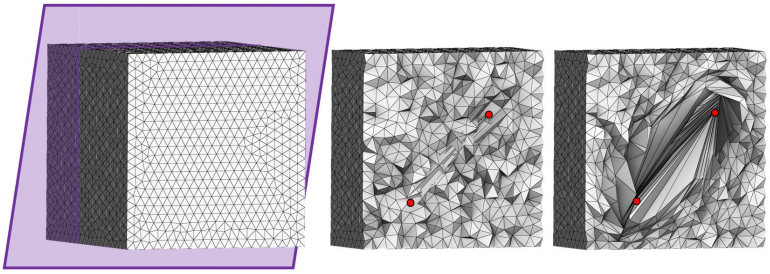


FIGURE 3.10 – Inversion de deux points dans un objet 3D. Deux points sont choisis dans une coupe d'un cube (à gauche), puis mis à la place l'un de l'autre (au milieu en rouge), une déformation sans éléments inversés (à droite) est ensuite calculée à l'aide de l'algorithme 1.

3.2.4 Résultats et discussion

Dans cette sous-section, nous évaluation expérimentalement la méthode. Nous commençons par un simple test, puis nous testons notre code sur une importante base de données de l'état de l'art. Enfin, nous faisons un ensemble de comparaisons avec l'état de l'art, et montrons l'influence du paramètre λ de l'énergie.

Test éliminatoire

Dans le domaine de l'infographie, toute affirmation concernant l'injectivité d'une carte est toujours confrontée à un simple contrôle (voir la FIGURE 3.9) : Prenez un carré et échangez deux points intérieurs quelconques. Notre méthode réussit ce test aisément, en 2D et en 3D, illustré FIGURE 3.10. Il faut noter que ce test, bien que facile à mettre en œuvre, reste un vrai challenge pour la plupart des méthodes de l'état de l'art.

Base de données de référence

Du *et al.* ont récemment publié une méthode permettant de construire des cartes localement injectives pour les problèmes à bords contraints [Du *et al.*, 2020]. Il y a une idée élégante derrière leur *Total Lifted Content* (TLC) : les auteurs proposent de minimiser la surface totale non signée

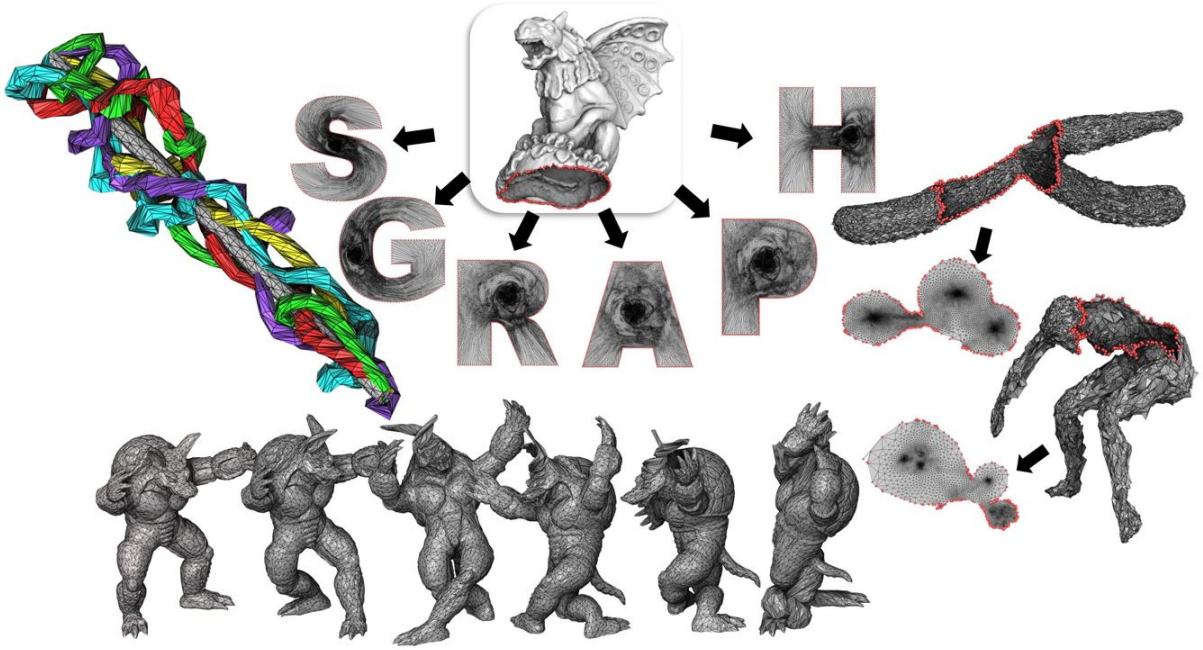


FIGURE 3.11 – Exemples de déformations et cartes inclus dans la base de données de [Du *et al.*, 2020].

d'un maillage à démêler. Cette énergie est régularisée en élevant les simplexes vers un espace de plus haute dimension. Après la régularisation, l'énergie devient lisse, et chaque minimum global de l'énergie est atteint par un plongement injectif.

Parallèlement à leur article, Du et al. ont publié une précieuse base de données de référence. Elle contient un grand nombre de défis de cartographie injective de frontières sous contraintes en 2D et 3D (voir la FIGURE 3.11). À notre connaissance, la méthode TLC et la notre sont les seules à passer le benchmark sans aucun échec. Pour les défis 2D, le benchmark contient des surfaces triangulées 3D à aplanir, et non des maillages 2D plats comme nous l'avons décrit dans à la sous-section 3.1.3. Néanmoins, notre méthode peut les traiter directement, car la carte est toujours définie de $\mathbb{R}^2 \to \mathbb{R}^2$ sur chaque triangle.

Un exemple représentatif de la base de données est donné dans la ligne supérieure de la FIGURE 3.12 Le défi consiste à faire correspondre le maillage de statuette "Lucy" du laboratoire d'infographie de Stanford à un domaine en forme de P. Ce maillage a la topologie d'un disque, et les sommets de son bord sont uniformément espacés sur le bord du domaine en forme de "P". En guise d'initialisation du problème, Du et al. calculent un *Tutte embedding*, résolvant $\Delta\vec{u} = 0$. Comme la forme P n'est pas convexe, celui-ci contient manifestement des inversions (voir la FIGURE 3.12- en haut à gauche). Le problème se résume alors à un démêlage de mailles avec une frontière verrouillée.

Mesure de la qualité d'une carte Comment mesurer la qualité d'une carte ? Cela dépend de l'objectif. L'identité est un idéal inatteignable ; les objectifs traditionnels, concurrents, sont des cartes préservant (autant que possible) l'angle (conforme) et l'aire (authalique ou équivalente). Nos cartes étant affines par morceaux, la matrice jacobienne J est constante par élément. Définissons sa plus grande valeur singulière comme $\sigma_1(J)$, et sa plus petite valeur singulière

comme $\sigma_d(J)$. Nous utilisons deux métriques pour calculer la qualité : la distorsion maximale $\max \frac{\sigma_1(J)}{\sigma_d(J)}$, et le facteur d'échelle minimal $\min \det J$ (le déterminent mesurant la variation de l'aire par rapport à la forme originale).

Pour l'exemple "Lucy-to-P" (FIGURE 3.12–Ligne du haut), notre carte diffère du résultat de la TLC avec 12 ordres de grandeur en termes de facteur d'échelle minimal, et deux ordres de grandeur en terme de distorsion maximale. Pour visualiser cette différence d'échelle, nous avons fourni des gros plans : La FIGURE 3.12– en haut au milieu montre une carte de la torche de Lucy, alors que le même niveau de zoom sur le résultat de Du et al. (FIGURE 3.12–en haut à droite) contient non seulement la torche, mais aussi les deux ailes, la tête et le bras droit !

Notez également que le maillage d'entrée de "Lucy" est légèrement anisotrope ; notre méthode nous permet de prescrire la forme cible des éléments, de sorte que les plis de la robe sont clairement visibles dans notre carte.

Pourquoi une telle différence ? Lorsque le maillage est sans inversion, l'énergie de TLC devient dégénérée (tous les maillages ayant la même aire). Cela implique qu'elle va trouver une solution le plus proche possible de l'initialisation, et ne pourra pas améliorer plus. L'idée avancée par les auteurs est que l'on pourra ensuite utiliser une énergie du type (3.1) puisque le maillage est maintenant admissible. En pratique, les éléments peuvent être de tellement mauvaise qualité, qu'il semble numériquement insensé d'essayer d'optimiser avec (3.1). C'est l'un des grands avantages de notre méthode, nous optimisons directement avec la bonne énergie, donc nous essayons directement d'avoir un bon maillage, en plus de le démêler.

Benchmarking La base de donnée fournie par [Du *et al.*, 2020] consiste en 10743 maillages en 2D et 904 maillages en 3D avec des contraintes de bords données. Nous arrivons à obtenir une carte admissible sur l'ensemble de ces maillages. Plus que cela, nous arrivons à obtenir des résultats pour différentes initialisations : tous les points placés en 0, ou alors placés aléatoirement, et cela sur tous les maillages. [Du *et al.*, 2020] y arrive uniquement avec une initialisation par un *Tutte's embedding*.

Dans la FIGURE 3.13, nous fournissons des graphiques de qualité des cartes admissibles résultantes. Il s'agit de diagrammes de dispersion $\log - \log$: chaque point correspond à une qualité de la carte correspondante réduite à deux nombres : la distorsion maximale ($\max \frac{\sigma_1(J)}{\sigma_d(J)}$) et le facteur d'échelle minimal ($\min \det J$). La colonne de gauche de la figure montre les pires mesures de qualité pour chaque problème 2D (en haut) ainsi que pour chaque défi 3D (en bas) de l'ensemble de données. Nos résultats sont indiqués en bleu, tandis que les résultats de TLC sont indiqués en rouge. Pour illustrer la distribution de la qualité des éléments, pour chaque carte injective, nous avons supprimé 5% des pires mesures. La colonne de droite de la FIGURE 3.13 montre la distorsion maximale et le facteur d'échelle minimal pour les 95% de mesures les meilleures.

Notez les suites de points formant des lignes dans le graphe : ces points correspondent aux quelques suites de déformation progressives d'un même objet présentes dans la base de données. Des exemples sont visibles dans la FIGURE 3.11, où l'on voit plusieurs déformations du maillage *Armadillo* et d'une barre.

Timings La FIGURE 3.14 fournit un diagramme de dispersion $\log - \log$ de notre temps d'exécution en fonction de la taille des maillages pour tous les exemples de la base de données [Du *et al.*, 2020] : pour chaque exécution, le temps varie d'une fraction de seconde à plusieurs minutes pour les maillages les plus grands. Ces temps ont été obtenus avec un CPU 12 cœurs i7-6800K @ 3.40 GHz. Comme dans la FIGURE 3.13, les lignes verticales dans les données 3D correspondent aux suites de déformation d'un même objet dans la base de données.

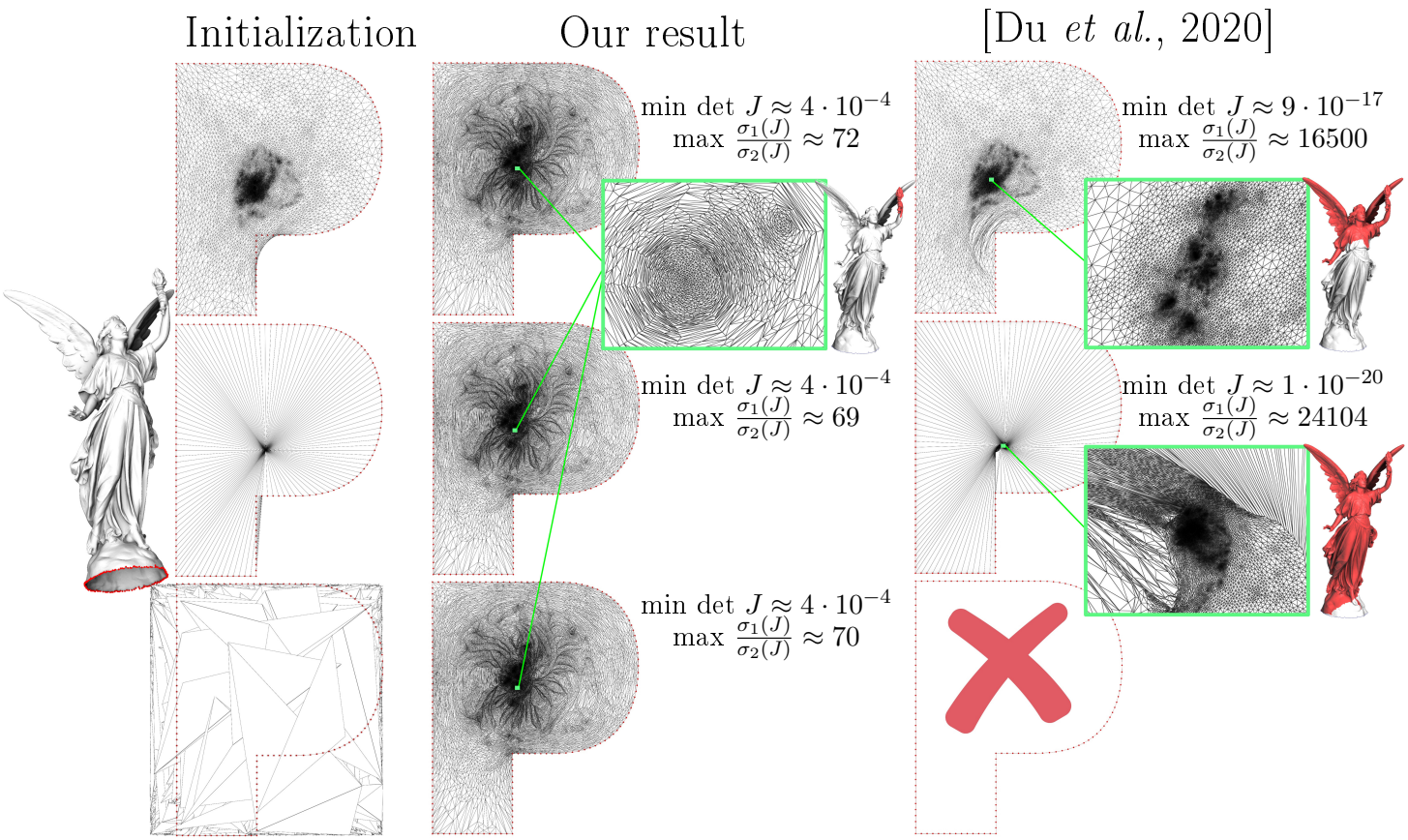


FIGURE 3.12 – Exemple de carte admissible à bord contraint proposé par [Du *et al.*, 2020] : le maillage "Lucy" est cartographié sur un domaine en forme de P en contraignant les sommets indiqués en rouge. Colonne de gauche : trois initialisations différentes pour le même problème. Colonne du milieu : notre méthode produit le même résultat (à une précision numérique près) sur les trois initialisations, nous avons choisi $\lambda = 0.01$. Colonne de droite : la méthode du TLC de [Du *et al.*, 2020] échoue dans sa résolution pour des sommets intérieurs initialisés aléatoirement, et produit des résultats très différents sur les deux autres initialisations. Trois images miniatures de "Lucy" montrent en rouge la portion de la surface visible dans les gros plans correspondants.

Deux diagrammes de dispersion sont superposés, tous deux représentent le même schéma de résolution avec une exception correspondant à la façon dont nous calculons ε^k (Algorithme 1–ligne 3). Le nuage de points verts correspond à une règle de mise à jour conservatrice (équation (3.9)) offrant des garanties sur le démêlage en un nombre fini d'étapes (voir théorème 1), alors que le nuage de points bleus est obtenu en utilisant la règle de mise à jour heuristique [Garanzha et Kaporin, 1999, Eq. 6.3]. Cette formule a été choisie de manière empirique, mais elle donne de bons résultats dans la grande majorité des situations. Par exemple, elle permet à toutes les bases de données [Du *et al.*, 2020] de passer le test d'injectivité.

Plus de tests

Sensibilité à l'initialisation Notre prochain test concerne la sensibilité à l'initialisation. Nous avons généré deux autres initialisations pour le défi "Lucy-to-P" : celle où tous les sommets

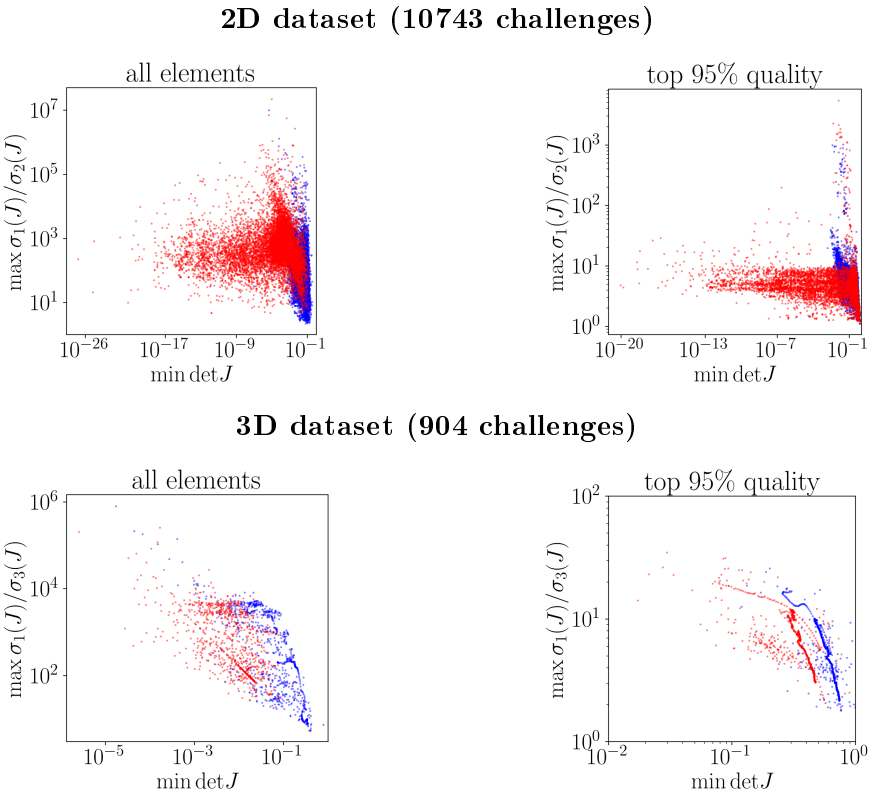


FIGURE 3.13 – Plot de qualité des cartes localement injectives résultantes pour chaque défi de la base de données fournie par [Du *et al.*, 2020]. Nos résultats sont indiqués en bleu, tandis que les résultats de Du et al. sont indiqués en rouge. Chaque point correspond à une qualité de la carte correspondante réduite à deux nombres : la distorsion maximale et le facteur d'échelle minimale. **Ligne du haut :** qualité de la cartographie sur l'ensemble de données 2D. **Ligne du bas :** qualité de la cartographie sur l'ensemble de données 3D. La colonne de gauche indique la distorsion maximale absolue et le facteur d'échelle minimale absolue, tandis que la colonne de droite indique la distorsion maximale et le facteur d'échelle minimale pour les 95% supérieurs des mesures.

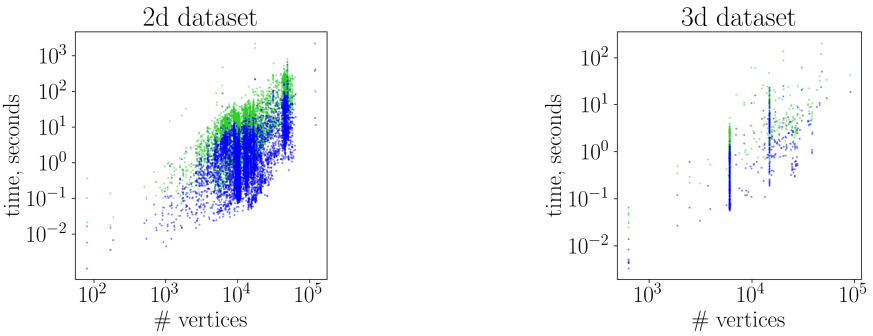
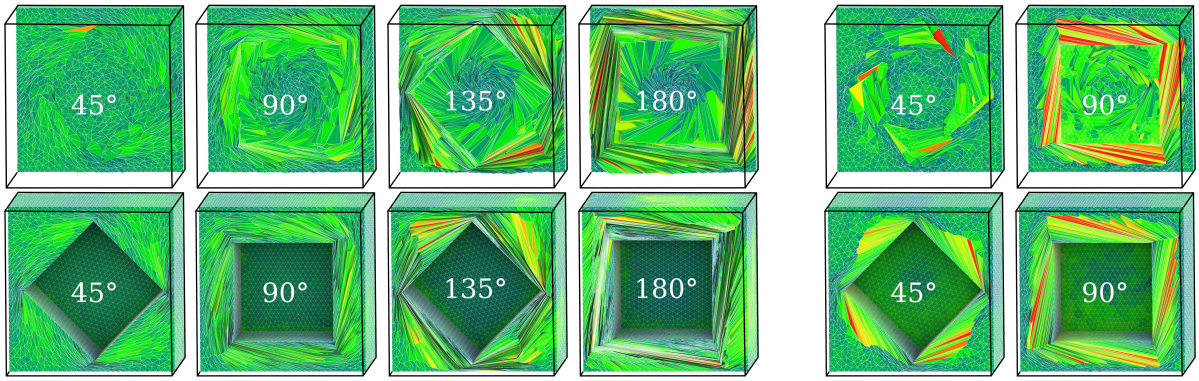


FIGURE 3.14 – Performance de notre méthode testée sur le benchmark [Du *et al.*, 2020]. Chaque point correspond à un défi de la base de données (10743 en 2D et 904 en 3D). Les points bleus montrent les temps d'exécution obtenus en utilisant une régularisation heuristique [Garanzha et Kaporin, 1999, Eq. 6.3], les points verts correspondent à l'équation (3.9).



(a)	(b)	(c)	(d)	(e)	(f)
$\max \frac{\sigma_1(J)}{\sigma_3(J)} \approx 37$	$\max \frac{\sigma_1(J)}{\sigma_3(J)} \approx 58$	$\max \frac{\sigma_1(J)}{\sigma_3(J)} \approx 759$	$\max \frac{\sigma_1(J)}{\sigma_3(J)} \approx 659$	$\max \frac{\sigma_1(J)}{\sigma_3(J)} \approx 524$	$\max \frac{\sigma_1(J)}{\sigma_3(J)} \approx 3391$
$\min \det J \approx 0.03$	$\min \det J \approx 0.02$	$\min \det J \approx 0.003$	$\min \det J \approx 0.002$	$\min \det J \approx 0.002$	$\min \det J \approx 0.0004$

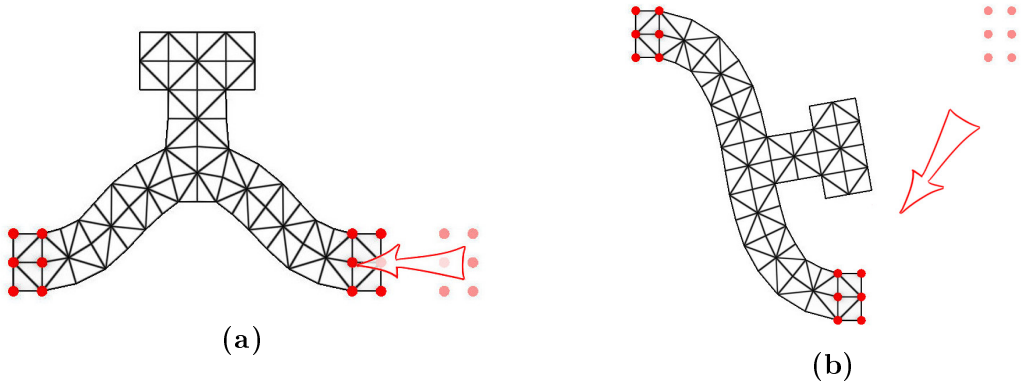
FIGURE 3.15 – Calcul de carte à bord contraint : test de résistance de la méthode. Nous avons généré un maillage tétraédrique isotrope d'un cube soustrait d'un cube plus petit, et nous faisons tourner le cube intérieur pour tester la robustesse. La rangée supérieure et la rangée inférieure correspondent à deux tranches différentes du même maillage. Colonnes **(a)**–**(d)** : cartes admissibles produites par notre méthode, colonnes **(e)** et **(f)** : cartes admissibles produites par [Du *et al.*, 2020]. La méthode de [Du *et al.*, 2020] ne parvient pas à générer des cartes injectives pour les rotations de 135° et 180° du cube interne. La carte des couleurs illustre le facteur d'échelle du volume relatif : vert pour $\det J \approx 1$, rouge pour l'inflation, bleu pour la compression.

intérieurs sont regroupés en un seul point (FIGURE 3.12–ligne du milieu), et avec les sommets intérieurs placés aléatoirement à l'intérieur d'un carré délimité (FIGURE 3.12–ligne du bas).

Notre méthode produit pratiquement le même résultat pour les trois initialisations, alors que la méthode TLC génère des résultats très différents pour les deux premières, et échoue pour la troisième. Il est intéressant de noter que la méthode TLC est fortement dépendante de l'initialisation : elle modifie très peu la géométrie d'entrée. Nos expériences avec le code source [Du *et al.*, 2020] montrent que la plupart des défis du benchmark échouent sous une initialisation aléatoire. De plus, notre "test éliminatoire" (FIGURE 3.9) échoue également.

Test de contrainte de grande déformation Pour notre prochain test, nous avons généré un maillage tétraédrique isotrope d'un cube avec une cavité, et nous avons fait tourner le bord interne pour tester la robustesse de notre méthode aux grandes déformations. La FIGURE 3.15 montre les résultats. Notre schéma d'optimisation basé sur L-BFGS réussit jusqu'à la rotation de 135°, et nous avons dû passer à une méthode de Newton pour atteindre la rotation de 180°. La méthode TLC a réussi sur 45° et 90°, et a échoué pour les 135° et 180°. Notez que comme dans le test précédent, même lorsque le démêlage est réussi, la méthode TLC modifie très peu la carte d'entrée, produisant ainsi des tétraèdres fortement étirés, alors que notre méthode diffuse uniformément la contrainte sur tout le domaine.

Calcul de carte à bord libre À notre connaissance, notre méthode est la seule à passer l'ensemble de la base de données tout en étant capable de travailler avec des cartes à bords libres. Puisque la méthode TLC tente de minimiser le volume global, le relâchement des contraintes de



(a) **(b)**

FIGURE 3.16 – Carte admissible à bord libre. Les sommets indiqués en rouge sont contraints, tous les autres sommets sont libres de se déplacer. **(a) :** un essai de compression, **(b) :** un essai de flexion. Se référer à la FIGURE 3.17–a pour l'objet initial.

bord donne lieu à des cartes dégénérées.

La FIGURE 3.16 montre deux cartes obtenues avec notre méthode : une forme 2D étant compressée et la même forme étant pliée. Le bord est libre de se déplacer, nous verrouillons les sommets indiqués en rouge. Référez-vous à la FIGURE 3.17-a pour l'objet initial. La forme se comporte exactement comme un humain l'attendrait : lors de la compression, la forme choisit l'un des deux résultats possibles (FIGURE 3.16–a), et passe avec succès le test de flexion (FIGURE 3.16–b), notez les détails géométriques qui sont naturellement tournés.

Compromis forme-aire Nos derniers tests illustrent l'influence du paramètre λ dans le problème (3.8) sur la carte résultante. Nous avons calculé trois cartes à bord libre de l'objet initial (voir FIGURE 3.17–a) sous un étirement. Nous avons d'abord choisi $\lambda = 0$, c'est-à-dire que seul le terme sur la forme des triangle, *i.e.* les angles, est pris en compte dans le problème (3.8). Lorsque nous optimisons pour les angles, la surface des triangles est forcée de changer (se référer à la FIGURE 3.17–b pour la carte résultante). Naturellement, une carte préservant l'aire ($\lambda = 10^4$) doit déformer les éléments pour satisfaire la contrainte d'aire (FIGURE 3.17–c). Enfin, sur la FIGURE 3.17–d, nous montrons un exemple avec un compromis entre la préservation de l'aire et des angles.

Qualité de la carte : comparaison avec LSCM Pour évaluer la qualité de nos cartes, nous avons calculé une carte conforme [15] discrète du modèle "Lucy" en fixant $\lambda = 0$; nous comparons le résultat à *Least Square Conformal mapping* (LSCM) [Lévy *et al.*, 2002]. LSCM est une méthode très répandue qui nécessite de résoudre un système linéaire avec une matrice symétrique définie positive. L'idée est de calculer une approximation par éléments finis P^1 des conditions de Cauchy-Riemann sur tous les triangles du maillage. Les résultats numériques sont présentés dans la FIGURE 3.18. Comme précédemment, pour comparer la qualité des cartes, nous utilisons le conditionnement de la matrice jacobienne $\frac{\sigma_1(J)}{\sigma_2(J)}$, où σ_1 et σ_2 représentent les valeurs singulières de J. Dans notre carte, la valeur maximale est égale à $5,1$. La grande majorité des éléments dans la carte obtenue par LSCM a une distorsion faible, *i.e.* est très faiblement déformée, cependant le nombre d'éléments mal déformés (la distorsion dépasse 1000) est considérable et même certains éléments inversés sont présents.

15. Une carte conforme est une carte préservant les angles.

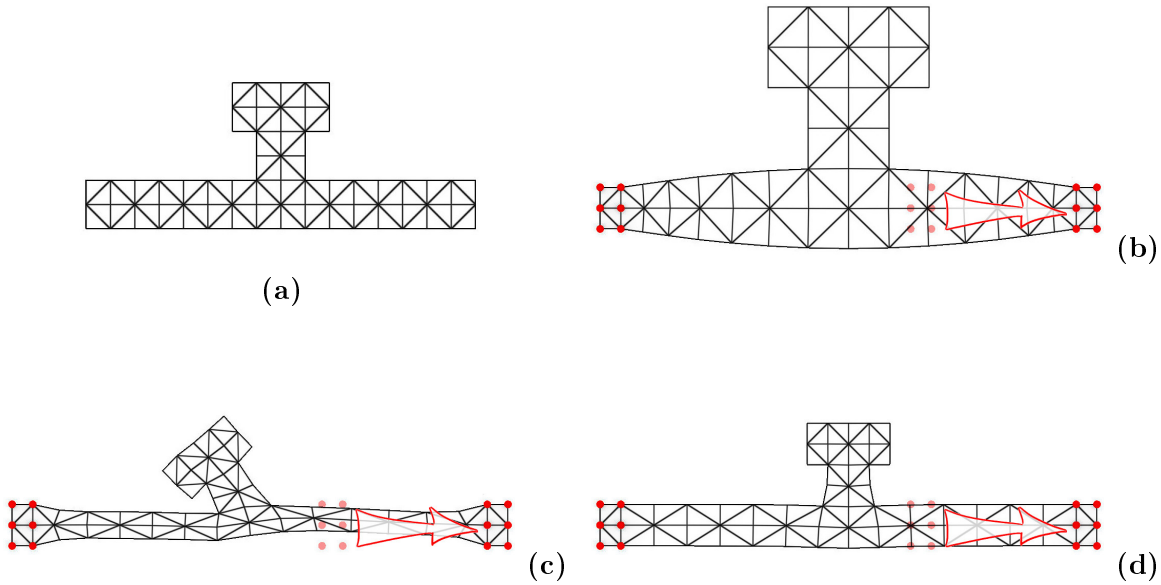
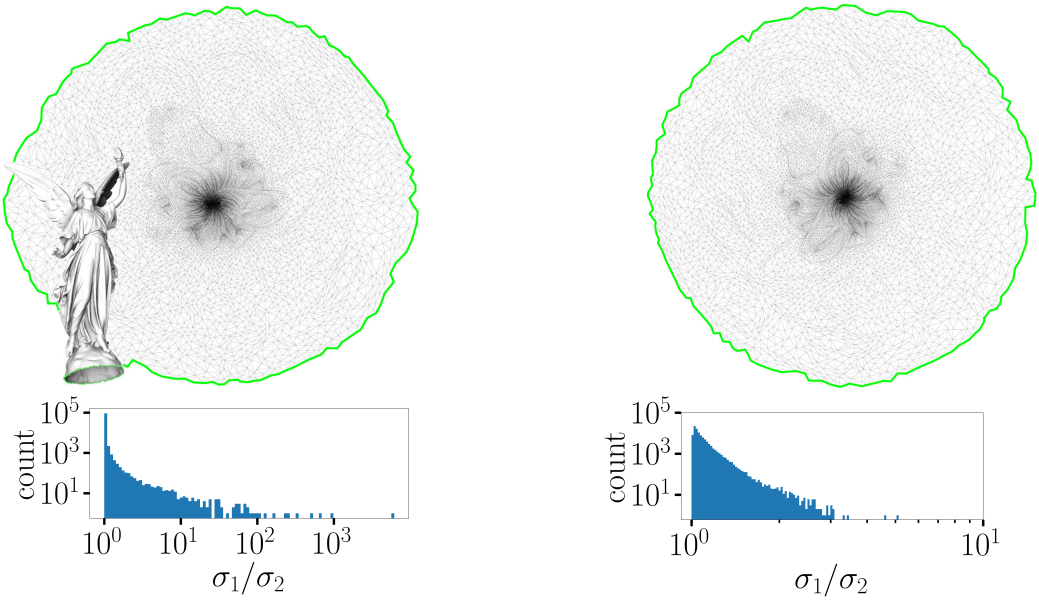


FIGURE 3.17 – Carte admissible à bord libre : influence du paramètre λ dans le problème (3.8). Les sommets indiqués en rouge sont contraints, tous les autres sommets sont libres de se déplacer. **(a) :** objet initial ; **(b) :** un étirement préservant la forme des éléments ($\lambda = 0$) ; **(c) :** un étirement préservant la surface ($\lambda = 10^4$) ; **(c) :** un compromis entre la forme et la préservation de la surface ($\lambda = 1$).

Qualité des cartes : comparaison avec SA Simplex Assembly (SA) [Fu et Liu, 2016] est une méthode permettant de calculer des cartes sans inversion avec une distorsion limitée sur des maillages simpliciaux. L'idée est de projeter chaque simplexe dans l'espace sans inversion et à distorsion bornée. Après avoir désassemblé le maillage, les simplexes sont ensuite assemblés en minimisant la distorsion de la carte, tout en gardant la carte admissible.

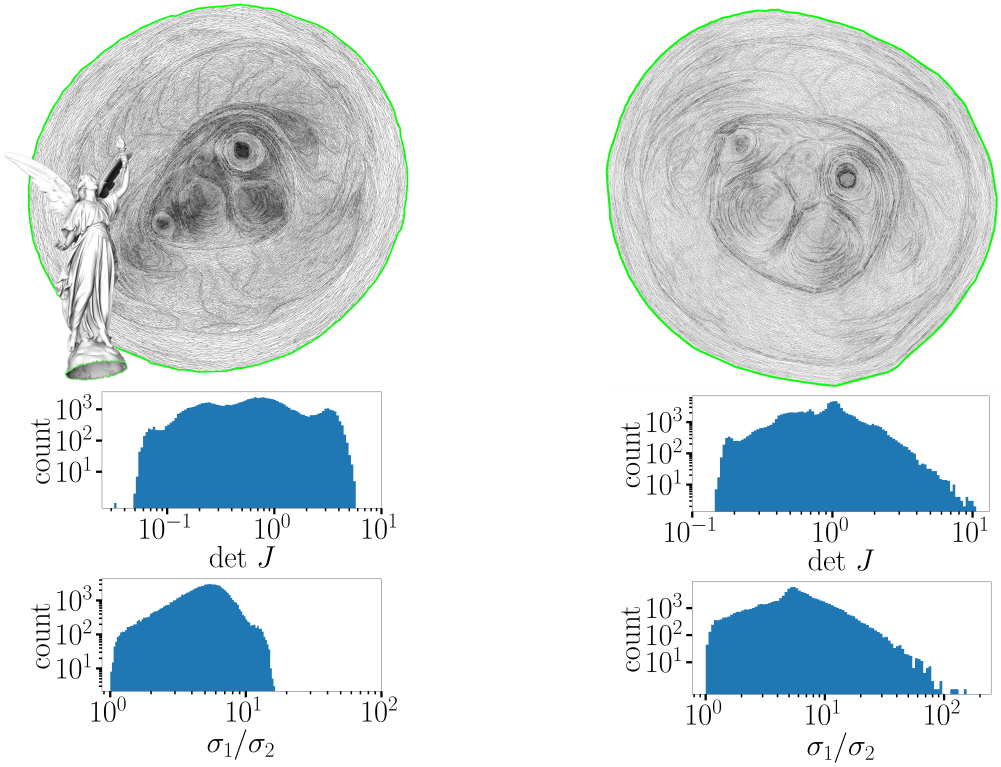
SA est une méthode relativement robuste, qui présente néanmoins quelques cas d'échec sur la base de données de référence. La FIGURE 3.19 fournit une comparaison de la qualité de SA avec notre carte quasi-isométrique ($\lambda = 1$) pour une carte à bord libre du maillage "Lucy". Notez que pour le cas 2D l'énergie de SA est exactement la même que celle que nous utilisons (pour le cas 3D l'énergie provient de [Knupp, 2000b]), cependant SA inclut une optimisation pour les bornes de la distorsion, atteignant ainsi une meilleure qualité de carte.

Qualité de la carte : comparaison avec LBD Large-scale Bounded Distortion Mappings (LBD) [Kovalsky *et al.*, 2015] est une autre méthode de calcul de cartes à bord libre. Étant donnée une carte d'entrée (potentiellement avec des éléments inversés), LBD recherche une carte admissible aussi proche que possible de la carte d'entrée, mais satisfaisant certaines contraintes telles que les limites d'orientation et de distorsion. La FIGURE 3.20 fournit une comparaison de LBD avec notre méthode. La surface 3D à aplanir est un maillage régulier simplicial d'un patch rectangulaire qui a été soulevé et bruité. LBD a une optimisation des bornes de la distorsion, ainsi la qualité du pire élément de la carte par LBD est meilleure que dans notre carte. Notez cependant que LBD a beaucoup d'éléments proches de la pire limite, alors que notre méthode se base sur la théorie de l'élasticité, et fournit une meilleure distribution de la qualité globale.



(a) **(b)**

FIGURE 3.18 – Comparaison de deux cartes conformes discrètes pour le maillage "Lucy". **(a) :** Carte conforme via [Lévy *et al.*, 2002], **(b) :** notre carte obtenue avec $\lambda = 0$. **Ligne du haut :** aplatissements, **Ligne du bas :** histogrammes log-log de la qualité des éléments des cartes conformes.



(a) **(b)**

FIGURE 3.19 – Deux cartes quasi-isométriques pour le maillage "Lucy". **(a) :** Simplex Assembly, **(b) :** notre carte obtenue avec $\lambda = 1$. **Ligne du haut :** aplatissements, **Lignes du milieu et du bas :** histogrammes log-log de la qualité des éléments.

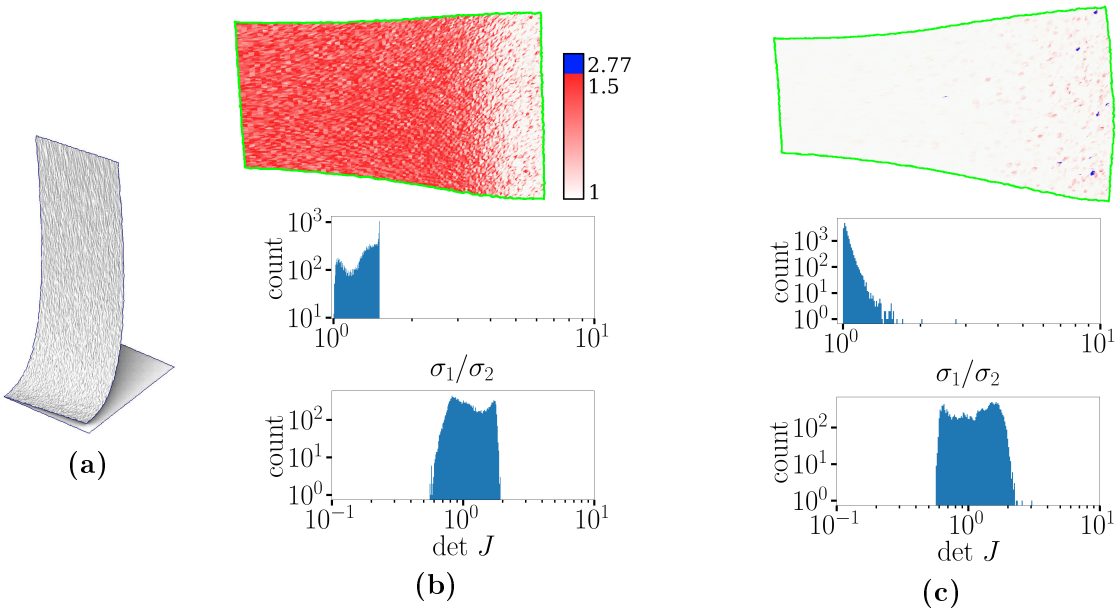


FIGURE 3.20 – Comparaison de la méthode LBD et de la nôtre **(a)** La surface 3D à aplanir est un maillage triangulaire régulier d'un patch carré qui a été soulevée et bruitée. **(b)** La carte obtenue par LBD. **(c) :** La carte calculée par notre méthode en fixant $\lambda = 0$. **Ligne du haut :** aplatissements de **(a)**, les couleurs correspondent à la qualité des éléments. **Lignes du milieu et du bas :** histogrammes log-log de la qualité des éléments.

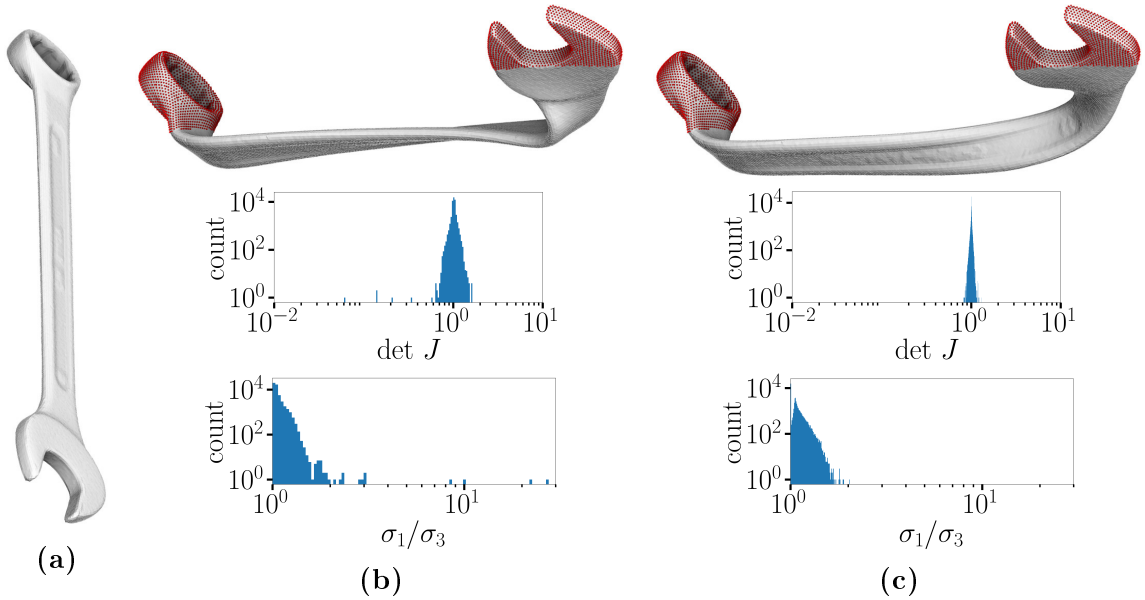


FIGURE 3.21 – Déformation du maillage tétraédrique, les sommets verrouillés sont indiqués en rouge. **(a) :** Objet initiale, **(b) :** ABCD, **(c) :** notre résultat avec $\lambda = 1$. **Ligne du haut :** aplatissements, **Lignes du milieu et du bas :** histogrammes log-log de la qualité des éléments.

Qualité de la carte : comparaison avec ABCD Nous concluons la comparaison de la qualité par un exemple de déformation de maillage 3D. Adaptive Block Coordinate Descent for Distortion Optimization (ABCD) [Naitsat *et al.*, 2020]. La FIGURE 3.21 présente les graphiques de qualité. Nous avons pris un maillage tétraédrique d'une clé, et nous avons imposé des contraintes de position sur les sommets situés aux deux extrémités de la clé. La déformation par ABCD est calculée en utilisant l'énergie ARAP (*as-rigid-as-possible* [Sorkine et Alexa, 2007]), alors que notre déformation est une quasi-isométrie avec $\lambda = 1$. Dans cette expérience, notre déformation présente une qualité légèrement supérieure, mais globalement comparable.

3.2.5 Analyse mathématique

Cette section présente différentes analyses de la méthode de pénalité. Nous commençons par une discussion sur les conditions d'inversibilité dans le cas discret et continu. Ensuite, nous montrons les origines de l'équation (3.9) pour la suite de paramètres de régularisation $\{\varepsilon^k\}_{k=0}^{\dots}$ qui permet d'obtenir des garanties d'obtention d'une carte admissible. Enfin, nous montrons que la hessienne utilisée a un comportement similaire aux matrices éléments finis, une propriété appréciée par les méthodes d'inversion de matrices.

Inversibilité du minimiseur

Une question légitime se pose : le minimiseur de notre problème nous donne-t-il vraiment une cartographie inversible ? Cette question n'est pas facile. Au niveau discret, la fonctionnelle utilisée pour le problème (3.8) a une barrière infinie à la limite de l'ensemble des cartes admissibles. Cela signifie que tout minimiseur d'énergie finie est exempt d'éléments inversés. En augmentant cette propriété de barrière avec des conditions de bord appropriées, par exemple avec un homéomorphisme de bords prescrit, on peut prouver que la déformation de maillage en général est un homéomorphisme [Prokhorova, 2008, Aigerman et Lipman, 2013].

En revanche, pour les paramètres continus, la situation est beaucoup plus subtile. Le problème variationnel (3.1), bien que provenant de l'élasticité, viole les conditions formulées par Ball pour ses théorèmes d'existence [Ball, 1976] et d'inversibilité [Ball, 1981]. Plus précisément, la fonctionnelle ne respecte pas les conditions d'accroissement des théorèmes. Cela ne signifie pas que le problème (3.1) est mal posé, mais nous ne pouvons pas en dire plus.

Pourquoi cela fonctionne-t-il si bien dans un cadre discret ? En fait, pour toute déformation du maillage avec une énergie finie, la mesure de la distorsion pour chaque cellule est bornée par le haut. Par conséquent, notre algorithme numérique agit sur l'espace des homéomorphismes quasi-isométriques.

Considérons le problème (continu) suivant lié à la construction de déformations avec une qualité prescrite [Garanzha, 2000] :

$$\arg\min_{\vec{u}(\vec{x})} \int_{\Omega} \frac{\beta(J)}{1 - t\beta(J)}\, dx, \quad \beta(J) = \frac{f(J) + \lambda g(J)}{d + 2\lambda}. \tag{3.12}$$

Cette intégrale ne peut être finie que dans le cas $\beta(J) < \frac{1}{t}$. Notons que le problème (3.1) est un cas particulier du problème (3.12) pour $t = 0$, où le paramètre t est la limite inférieure de qualité de la déformation. Il est important de noter que la densité de l'énergie de déformation est polyconvexe et donc le problème variationnel (3.12) est bien posé. De plus, le théorème d'existence peut être prouvé et sous des conditions aux limites appropriées, les déformations admissibles pour le problème (3.12) sont des homémorphismes quasi-isométriques [Garanzha *et al.*, 2014] à la fois en 2D et en 3D, c'est-à-dire que le théorème d'inversibilité [Ball, 1981] peut être appliqué.

On peut considérer le problème (3.8) comme une minimisation de la fonctionnelle (3.12) avec une constante arbitrairement petite $t > 0$. À son tour, le problème (3.12) avec une valeur proche de zéro de t peut être vue comme une solution régularisée de (3.1).

À titre d'information, une transformation similaire à (3.12) peut être appliquée à une très large classe d'énergies polyconvexes isochoriques-volumétriques basées sur la séparation qui violent à l'origine les conditions du théorème d'inversibilité de Ball. Cette transformation supprime les singularités des déformations, ce qui est tout à fait naturel puisque les déformations singulières sont hors de portée des hypothèses de la théorie élastique et doivent s'appuyer sur d'autres modèles physiques.

Choix de ε^k

Dans cette section, nous fournissons une stratégie pour le choix du paramètre de régularisation ε^k à chaque itération. En particulier, nous prouvons qu'un algorithme de minimisation *idéalisée* (définition suivante) atteint l'ensemble admissible $\min \det J > 0$ en un nombre fini d'itérations. Plus précisément, nous prouvons que si le problème a une solution, alors une méthode de minimisation *idéalisée* peut atteindre l'ensemble admissible $\min \det J > 0$ en un nombre fini d'étapes. Une conséquence immédiate de ce théorème est que si le problème a une solution, alors pour un certain $K < \infty$ la solution $\arg\min_U F(U, \varepsilon^K)$ appartient à l'ensemble admissible.

Définition 2 (méthode de minimisation idéalisée pour σ)**.** *On dit qu'une méthode de minimisation est idéalisée pour σ, avec $0 < \sigma < 1$, si, étant donnée une condition initiale U^k pour une fonctionnelle Φ, le résultat de la minimisation U^{k+1} respecte l'une des conditions suivantes :*

— ***soit** la condition de descente essentielle est respectée*

$$\Phi(U^{k+1}) \leq (1 - \sigma)\Phi(U^k), \tag{3.13}$$

— ***soit** le vecteur U^k satisfait la condition de quasi-minimalité :*

$$\min_U \Phi(U) > (1 - \sigma)\Phi(U^k). \tag{3.14}$$

De façon moins rigoureuse mathématiquement, une méthode de minimisation idéalisée peut être vu comme une méthode qui, pour une fonctionnelle donnée, soit fait un effort considérable de minimisation (3.13), ou bien renvoie une solution très proche du minimum global de la fonctionnelle (3.14).

Théorème 1. *Supposons que l'ensemble admissible n'est pas vide, à savoir qu'il existe un maillage U^* satisfaisant $F(U^*, 0) < +\infty$. Nous supposons également que nous disposons d'un algorithme de minimisation idéalisé pour un certain $0 < \sigma < 1$.*

Alors l'ensemble admissible est atteignable en résolvant un nombre fini de problèmes de minimisation dans U avec ε^k fixé pour chaque problème. En d'autres termes, sous un choix approprié de la séquence de paramètres de régularisation $\varepsilon^k, k = 0 \dots K$, on obtient $F(U^K, 0) < +\infty$.

Démonstration. L'idée principale est d'exposer une manière explicite de construire une séquence décroissante $\{\varepsilon^k\}_{k=0}^{\infty}$ telle que la séquence $\{F(U^k, \varepsilon^k)\}_{k=0}^{\infty}$ est bornée par le haut. On peut alors prouver par contradiction que l'ensemble admissible est atteignable en un nombre fini d'étapes, car s'il ne l'est pas, $F(U^k, \varepsilon^k)$ doit croître sans borne.

Tout d'abord, la fonction $F(U, \varepsilon)$ peut être réécrite de la façon suivante

$$F(U, \varepsilon) = \sum_i \alpha_i \frac{\psi_i(U)}{\chi(D_i, \varepsilon)}, \tag{3.15}$$

où $D_i = D_i(U)$ désigne le déterminant jacobien pour le i-ième simplexe du maillage ($D_i = \det J_i$), et $\alpha_i > 0$ sont des poids positifs, non nuls, attribués à chaque simplexe. Les fonctions

$$\psi_i(U, \varepsilon) := \chi(D_i, \varepsilon)^{1-\frac{2}{d}} \operatorname{tr} J_i^\top J_i + \lambda(D_i^2 + 1)$$

définis par l'équation (3.6) sont positifs et bornés par le bas comme suit

$$\alpha_i \psi_i(U, \varepsilon) \geq \lambda \min_i \alpha_i.$$

Notons également que les $\psi_i(U, \varepsilon)$ sont des fonctions croissantes de ε.

Notre but est de construire une suite décroissante $\{\varepsilon^k\}_{k=0}^\infty$ telle que la suite $\{F(U^k, \varepsilon^k)\}_{k=0}^\infty$ soit bornée par le haut. Nous divisons la construction en deux parties : d'abord nous supposons qu'à une certaine itération k la condition essentielle (3.13) est satisfaite, et ensuite nous explorons le cas (3.14).

Supposons que la condition (3.13), $F(U^{k+1}, \varepsilon^k) \leq (1-\sigma)F(U^k, \varepsilon^k)$, *est satisfaite à l'itération* k.

Afin de garantir que la fonction n'est pas croissante, il suffit d'établir l'inégalité suivante :

$$(1-\sigma)F(U^{k+1}, \varepsilon^{k+1}) \leq F(U^{k+1}, \varepsilon^k). \tag{3.16}$$

En notant que $\psi_i(U^{k+1}, \varepsilon^{k+1}) \leq \psi_i(U^{k+1}, \varepsilon^k)$, inégalité (3.16) est impliquée si la condition suivante est vérifiée :

$$\forall i: \qquad (1-\sigma)\chi(D_i^{k+1}, \varepsilon^k) \leq \chi(D_i^{k+1}, \varepsilon^{k+1}) \tag{3.17}$$

où $D_i^{k+1} := D_i(U^{k+1})$ désigne le déterminant jacobien du simplexe i à l'itération $k+1$.

Montrons une manière de construire ε^{k+1} de sorte que l'inégalité (3.17) soit satisfaite. Supposons que ε^{k+1} soit trouvé comme solution de l'équation

$$\chi(D_-^{k+1}, \varepsilon^{k+1}) = (1-\sigma)\chi(D_-^{k+1}, \varepsilon^k), \tag{3.18}$$

où $D_-^{k+1} := \min_i D_i^{k+1}$. De toute évidence, si $D_-^{k+1} < (1-\sigma)\chi(D_-^{k+1}, \varepsilon^k)$, il existe une solution positive unique à cette équation. Maintenant, évaluons la dérivée de la fonction $s(D) := \frac{\chi(D,t_1)}{\chi(D,t_2)}$, $t_1 < t_2$. Il est simple de vérifier que la dérivée $s'(D) > 0$:

$$s'(D) = \left(\frac{\chi'(D,t_1)}{\chi(D,t_1)} - \frac{\chi'(D,t_2)}{\chi(D,t_2)}\right) s = \left(\frac{1}{\sqrt{D^2+t_1^2}} - \frac{1}{\sqrt{D^2+t_2^2}}\right) s > 0.$$

Cette inégalité signifie que équation (3.18) induit une inégalité pour tout $D_i^{k+1} \geq D_-^{k+1}$

$$\frac{\chi(D_i^{k+1}, \varepsilon^{k+1})}{\chi(D_i^{k+1}, \varepsilon^k)} \geq \frac{\chi(D_-^{k+1}, \varepsilon^{k+1})}{\chi(D_-^{k+1}, \varepsilon^k)} = 1 - \sigma,$$

ce qui est précisément l'inégalité (3.17). Par conséquent, si U^{k+1} est une solution approximative du problème de minimisation $\arg\min_U F(U, \varepsilon^k)$ avec un paramètre fixe ε^k, nous pouvons utiliser la règle de mise à jour suivante pour ε^{k+1} qui est la solution explicite de l'équation (3.18) :

$$\varepsilon^{k+1} = 2\sqrt{\mu^k(\mu^k - D_-^{k+1})}, \tag{3.19}$$

où

$$\mu^k := (1-\sigma)\chi(D_-^{k+1}, \varepsilon^k). \tag{3.20}$$

Cette règle de mise à jour garantit que l'inégalité (3.16) est satisfaite ; couplée avec l'hypothèse (3.13) du théorème, cela implique la propriété de non-croissance requise pour la séquence de valeurs de la fonction :

$$F(U^{k+1}, \varepsilon^{k+1}) \leq F(U^k, \varepsilon^k). \tag{3.21}$$

Considérons maintenant le cas où la condition (3.14), $\min_U F(U, \varepsilon^k) > (1-\sigma)F(U^k, \varepsilon^k)$, *se vérifie à l'itération* k.

Notez que la condition (3.14) signifie essentiellement que notre solution actuelle U^k est très proche du minimum global de $F(U, \varepsilon^k)$, et donc l'inégalité (3.13) ne peut pas être satisfaite. Néanmoins, nous pouvons utiliser la même règle de mise à jour (3.19) pour le calcul de ε^{k+1}. En effet, avec ce choix, nous avons

$$\begin{aligned} F(U^{k+1}, \varepsilon^{k+1}) &\leq \frac{1}{(1-\sigma)} F(U^{k+1}, \varepsilon^k) \leq \frac{1}{(1-\sigma)} F(U^k, \varepsilon^k) \\ &< \frac{1}{(1-\sigma)^2} \min_U F(U, \varepsilon^k) < \frac{1}{(1-\sigma)^2} \min_U F(U, 0). \end{aligned}$$

Ici, la dernière inégalité fournit une limite globale sur la séquence de valeurs de la fonction, et elle est basée sur l'observation $\frac{\partial}{\partial \varepsilon}\chi(D, \varepsilon) > 0$.

En résumé, nous avons montré un moyen de construire une séquence $\{\varepsilon^k\}_{k=0}^{\infty}$ telle que la séquence $\{F(U^k, \varepsilon^k)\}_{k=0}^{\infty}$ est bornée par le haut. Prouvons maintenant que la règle de mise à jour (3.19) permet d'atteindre l'ensemble admissible en un nombre fini d'étapes. Pour ce faire, nous procéderons avec un raisonnement par l'absurde.

Supposons que l'ensemble admissible ne soit jamais atteint pour une séquence infiniment décroissante $\{\varepsilon^k\}_{k=0}^{\infty}$ construite en utilisant la règle de mise à jour (3.19), (3.20) c'est à dire $D_-^{k+1} < 0 \ \forall k \geq 0$.

On peut facilement voir que l'identité suivante peut être déduite de (3.19), (3.20), et (3.5) :

$$(\varepsilon^{k+1})^2 = (1-\sigma)\left((\varepsilon^k)^2 - 4\sigma(\chi(D_-^{k+1}, \varepsilon^k))^2\right). \tag{3.22}$$

En particulier, (3.22) montre la décroissance stricte de la séquence ε^k. De plus, à partir de (3.22) on a évidemment

$$(\varepsilon^k)^2 \geq 4\sigma(\chi(D_-^{k+1}, \varepsilon^k))^2,$$

et l'utilisation de cette dernière inégalité avec (3.22) donne l'inégalité suivante :

$$\begin{aligned} (\varepsilon^k)^2 - (\varepsilon^{k+1})^2 &= \sigma(\varepsilon^k)^2 + 4\sigma(1-\sigma)(\chi(D_-^{k+1}, \varepsilon^k))^2 \\ &= 4\sigma^2(\chi(D_-^{k+1}, \varepsilon^k))^2 + 4\sigma(1-\sigma)(\chi(D_-^{k+1}, \varepsilon^k))^2 \\ &= 4\sigma(\chi(D_-^{k+1}, \varepsilon^k))^2. \end{aligned}$$

Donc pour un certain $K > 0$:

$$(\varepsilon^0)^2 - (\varepsilon^K)^2 \geq 4\sigma \sum_{k=0}^{K-1} (\chi(D_-^{k+1}, \varepsilon^k))^2 \geq 4\sigma K \min_{0 \leq k < K} (\chi(D_-^{k+1}, \varepsilon^k))^2,$$

avec une conséquence immédiate que pour un K arbitrairement grand nous avons l'inégalité suivante :

$$\max_{0 \leq k < K} \frac{1}{\chi(D_-^{k+1}, \varepsilon^k)} \geq \frac{\sqrt{4\sigma K}}{\varepsilon^0}.$$

Puisque tous les termes $\alpha_i\psi_i(U)$ dans (3.15) sont bornés par le dessous, l'estimation résultante contredit le caractère borné de $F(U^k, \varepsilon^k)$, concluant ainsi notre preuve. ■

Remarque 6. *Un corollaire important du théorème 1 est que, à condition que l'ensemble admissible ne soit pas vide, il existe une itération $K < \infty$ telle que le minimum global de la fonction $F(U, \varepsilon^K)$ appartient à l'ensemble admissible. La preuve est assez évidente : supposons que nous ayons un minimiseur idéalisé tel que $U^{k+1} = \arg\min_U F(U, \varepsilon^k)$. Ce minimiseur satisfait toujours les conditions du théorème 1, il peut donc démêler le maillage en un nombre fini d'étapes.*

Remarque 7. *En pratique, l'estimation globale σ n'est pas connue à l'avance, et la routine d'optimisation peut être loin de l'idéal. Pour chaque étape de minimisation, nous calculons le coefficient de descente locale σ^k :*

$$\sigma^k := 1 - \frac{F(U^{k+1}, \varepsilon^k)}{F(U^k, \varepsilon^k)}.$$

Lorsque $\sigma^k \geq \sigma$, nous pouvons utiliser la règle de mise à jour (3.19) *en utilisant la valeur locale σ^k garantissant que l'inégalité* (3.21) *tient. Dans le cas où $\sigma^k < \sigma$ on doit vérifier que la condition* (3.14) *tient pour σ prescrit. Si elle est positive, on peut attribuer $\sigma^k = \sigma$ et utiliser la règle de mise à jour* (3.19)*. Si on ne peut pas assurer* (3.14)*, cela signifie que la procédure de minimisation pour ε^k a échoué et que le théorème ne peut pas être appliqué (cela ne signifie pas que Algorithme 1 n'atteindra pas l'ensemble admissible !) Dans les expériences numériques, nous utilisons la valeur $\sigma = \frac{1}{10}$.*

Bornes spectrales pour le hessien

En partant de l'hypothèse que la fonction $F(U, \varepsilon)$ est bornée, dérivons des limites spectrales (non strictes) pour la partie définie positive de la matrice hessienne au point $J = J_0, D = D_0, q = \chi(J_0, \varepsilon)$. Évidemment, la contribution de chaque simplexe est bornée $\phi(J_0) < K$, ce qui signifie que

$$1 + D^2 < \frac{K}{\lambda}\chi(D, \varepsilon), \quad |a|^2 < K(\chi(D, \varepsilon))^{2/d}$$

De ces inégalités et du fait que $\frac{\lambda}{K} < q < \sqrt{D^2 + \varepsilon^2}$, on peut déduire

$$D^2 + \varepsilon^2 > \frac{\lambda^2}{K^2}, \quad D^2 < \frac{K^2}{\lambda^2}, \quad q < \sqrt{\frac{K^2}{\lambda^2} + \varepsilon^2}.$$

Nous obtenons donc immédiatement

$$|a|^2 < K\left(\frac{K^2}{\lambda^2} + \varepsilon^2\right)^{1/d}, \ |b| < d|a|^d < dK^d\left(\frac{K^2}{\lambda^2} + \varepsilon^2\right).$$

La dernière inégalité découle du fait que b est constitué de colonnes de la comatrice de J.
Nous pouvons maintenant estimer les limites spectrales de la matrice $P + \lambda Q$. Nous avons

$$\lambda_{\max}(P + \lambda Q) \leq \operatorname{tr}(P + \lambda Q)$$
$$= \frac{2d^2}{q^{\frac{2}{d}}} + \frac{2}{d}(1 + \frac{2}{d})\frac{|a|^2 q'^2}{q^{2+\frac{2}{d}}} + \lambda\left(\frac{2}{q} - 4D\frac{q'}{q^2} + 2(1 + D^2)\frac{q'^2}{q^3}\right)$$

$$= \frac{2d^2}{q^{\frac{2}{d}}} + \frac{2}{d}(1+\frac{2}{d})\frac{|a|^2 q'^2}{q^{2+\frac{2}{d}}} + \frac{2\lambda}{q}\frac{1}{D^2+\varepsilon^2}\left(1+\frac{\varepsilon^4}{4q^2}\right),$$

où les relations

$$\frac{q'}{q} = \frac{1}{\sqrt{D^2+\varepsilon^2}}, \qquad q = \frac{1}{2}(D+\sqrt{D^2+\varepsilon^2})$$

ont été utilisés pour obtenir la dernière égalité.

Une borne inférieure pour la valeur propre minimale λ_1 découle de la simple estimation obtenue à partir de l'inégalité de la moyenne arithmétique-géométrique écrite pour les valeurs propres $\lambda_2 \leq \ldots \leq \lambda_{d^2+1}$ of $P+\lambda Q$:

$$0 < \lambda_{\min}(P+\lambda Q) = \lambda_1 = \frac{\det(P+\lambda Q)}{\prod_{k=2}^{d^2+1}\lambda_k}$$

$$\geq \frac{\det(P+\lambda Q)}{\left(\frac{1}{d^2}\sum_{k=2}^{d^2+1}\lambda_k\right)^{d^2}} > \left(\frac{d^2}{\operatorname{tr}(P+\lambda Q)}\right)^{d^2}\det(P+\lambda Q).$$

Le déterminant peut être borné par

$$\det(P+\lambda Q) = \frac{2^{d^2}}{q^{2d}}\lambda\left(\frac{2}{q} - 4D\frac{q'}{q^2} + 2(1+D^2)\frac{q'^2}{q^3}\right)$$
$$\geq \frac{2^{d^2+1}}{q^{2d+1}}\frac{\lambda}{1+D^2},$$

où la dernière inégalité est obtenue en prenant le minimum sur $(q'/q) > 0$ considéré comme variable indépendante.

Les estimations spectrales de la matrice $P+\lambda Q$ peuvent être exprimées par K, λ, ε et sont garanties uniformément bornées par le bas et par le haut, à condition que $\lambda > 0$ et que ε soit borné par le haut. Puisque le vecteur b est uniformément borné par le haut, on obtient immédiatement des bornes uniformes

$$k_1 I < M^+ < k_2 I, \; k_1 < k_2$$

où les paramètres $k_i = k_i(K,\lambda,\varepsilon) > 0$ sont uniformément bornés par le haut et par le bas. Cela signifie que

$$k_1 \mathcal{D}_h(U) < \frac{1}{2}U^\top H^+ U < k_2 \mathcal{D}_h(U),$$

où $\mathcal{D}_h(U)$ est la fonction de Dirichlet discrète pour les éléments finis linéaires linéaires standard qui approche la fonctionnelle de Dirichlet

$$\mathcal{D}(u(x)) = \frac{1}{2}\int_\Omega \sum_i |\nabla u_i|^2 \, dx.$$

Ainsi, nous avons démontré la stabilité de la partie définie positive de la matrice hessienne près de la barrière.

3.2.6 Limitations

Bien que globalement très performante en pratique, notre méthode présente encore quelques limitations. Nous avons deux sources principales de limitations : des limitations théoriques ainsi que des limitations très pratiques liées à la stabilité numérique de notre schéma de résolution.

Superpositions Une carte admissible n'implique pas une bijectivité globale, où chaque point dans l'espace paramétrique aurait une image unique sur l'objet original. Cela s'illustre dans deux situations : la première dans le cas de superpositions globales, où deux composantes éloignées l'une de l'autre de l'objet se trouveraient cartographiées au même endroit sur la carte. C'est un problème classique de la construction de carte. Il existe un ensemble de méthodes comme l'utilisation d'un maillage englobant lui aussi cartographié ou bien l'insertion de termes quadratiques permettant de repousser les parties se superposant. La deuxième, illustrée FIGURE 3.5, est plus sournoise. Dans notre méthode, nous estimons le jacobien de la carte uniquement au niveau des triangles, celui-ci peut donc avoir des valeurs arbitraires au niveau des points. Ainsi la carte peut "s'enrouler" autour de certains sommets, c'est-à-dire que l'angle total des triangles incidents à un sommet peut être supérieur à 2π, comme mentionné dans [Weber et Zorin, 2014]. Dans les deux cas, le bord de la carte s'auto-intersecte, comme démontré dans [Aigerman et Lipman, 2013] pour le cas 3D, impliquant qu'un maillage englobant l'objet peut éviter tous ses problèmes. Cependant, ce type méthode n'est pas possible dans certaines applications, comme l'aplatissement de surface. Typiquement, les superpositions se produisent à proximité de fortes contraintes provoquant une compression locale de l'objet. Nous avons proposé une méthode d'ajout de triangle qui permet d'éviter ces superpositions [Garanzha *et al.*, 2021b], nous la présentons dans la sous-section 3.4.1.

Défis numériques Même lorsque le problème est bien posé, une implémentation robuste peut présenter d'importantes difficultés. Comme nous l'avons dit plus haut, le schéma d'optimisation quasi-newtonien fonctionne bien pour les problèmes "simples" (il passe entièrement la base de données de référence !), mais peut échouer pour les grandes déformations, où des itérations de Newton sont nécessaires. Bien que notre matrice hessienne modifiée soit symétrique définie positive , il faut aussi considérer que pour des problèmes très difficiles, les gradients conjugués peuvent échouer et que l'on peut avoir besoin de méthode de Choleski, voir au-delà, pour inverser la matrice. Dans la pratique, nous avons constaté que la méthode est très robuste dans des contextes 2D : nous n'avons pas rencontré de cas pratique que nous n'ayons pu traiter avec notre méthode. En 3D, cependant, elle peut échouer en raison des difficultés numériques que posent des maillages très anisotropes et très tordus.

Pire qualité et qualité moyenne Bien que notre méthode trouve systématiquement une carte admissible, et que l'énergie minimisée converge vers des cartes de bonne qualité, la pire qualité de ces cartes peut encore laisser à désirer. En effet, l'énergie, pour améliorer la qualité globale, peut décider de sacrifier certains éléments. Cela nous permet d'arriver à la section suivante, où nous montrons qu'il est possible d'utiliser un schéma similaire à celui développé pour obtenir des cartes admissibles, mais cette fois-ci pour borner (par le dessous) la qualité de la carte, et maximiser celle-ci.

3.3 Étendre la pénalisation pour garantir la qualité

Dans cette section, l'objectif est d'obtenir des cartes maximisant la pire qualité. Plus précisément, chaque élément de notre discrétisation aura une certaine valeur d'énergie et la méthode d'optimisation présentée dans la section précédente peut sacrifier la qualité de certains éléments pour avoir une meilleure qualité moyenne. Or, dans certaines applications, la pire qualité peut justement être l'objectif que l'on cherche à maximiser. Nous proposons une variation de l'énergie précédente qui permet de pénaliser les pires qualités.

Nous reformulons le concept de l'ensemble des maillages admissibles comme un ensemble où toutes les cellules ont une qualité de déformation supérieure à un seuil donné. Pour atteindre le seuil de qualité spécifié par l'utilisateur, nous introduisons une nouvelle fonction $W(U,t)$, comme illustrée dans la FIGURE 3.22. Nous prouvons que lorsque l'ensemble admissible n'est pas vide, nous pouvons construire une séquence de seuils de qualité de telle sorte que l'ensemble admissible soit atteint après avoir résolu partiellement un nombre fini de problèmes d'optimisation. Cette technique nous permet de créer des déformations de maillage avec les plus petites estimations de distorsion connues (constantes de quasi-isométrie) et de construire des paramétrisations presque conformes pour des problèmes très rigides.

Avec cela, nous pouvons construire une chaîne de problèmes variationnels complète pour la construction de cartes et déformations optimales : Nous optimisons d'abord la fonctionnelle régularisée du problème (3.7) par rapport au paramètre ε pour la construction d'une première carte admissible, ensuite nous minimisons le problème (3.1) pour obtenir une déformation qui est optimale *en moyenne*. Enfin, nous minimisons le nouveau problème (3.26) que nous introduisons dans cette section, qui permet la maximisation de la pire mesure de qualité.

La caractéristique commune à ces trois problèmes variationnels est que nous définissons un ensemble des déformations admissibles et (a) essayons de forcer l'estimation initiale dans cet ensemble, (b) optimisons à l'intérieur de l'ensemble admissible et (c) essayons de comprimer l'ensemble admissible pour améliorer la pire qualité atteinte. De même, nous utiliserons des simplexes pour discrétiser et optimiser nos énergies, mais celle-ci étant définie continument, nos résultats peuvent être obtenus sur des éléments plus complexes (quadrilatères, hexaèdres, éléments B-spline).

Dans la suite de la section, nous définissons la nouvelle énergie et ses propriétés (sous-section 3.3.1), ensuite nous présentons un théorème qui donne une suite de problème à résoudre pour obtenir la solution désirée (sous-section 3.3.2). Nous illustrons enfin nos résultats dans un ensemble de situations (sous-section 3.3.3).

3.3.1 Amélioration de la pire qualité

Dans la section précédente, la mesure de la qualité se fait en un point par $w(J) = f(J) + \lambda g(J)$, l'intérieur de l'intégrale (3.1). Pour un maillage donné, prenons comme objectif de qualité minimum une qualité w^*. Ce w^* peut être donné par un utilisateur, ou bien obtenu en cherchant de façon dichotomique sa valeur. Nous voulons donc chercher une carte respectant cette qualité :

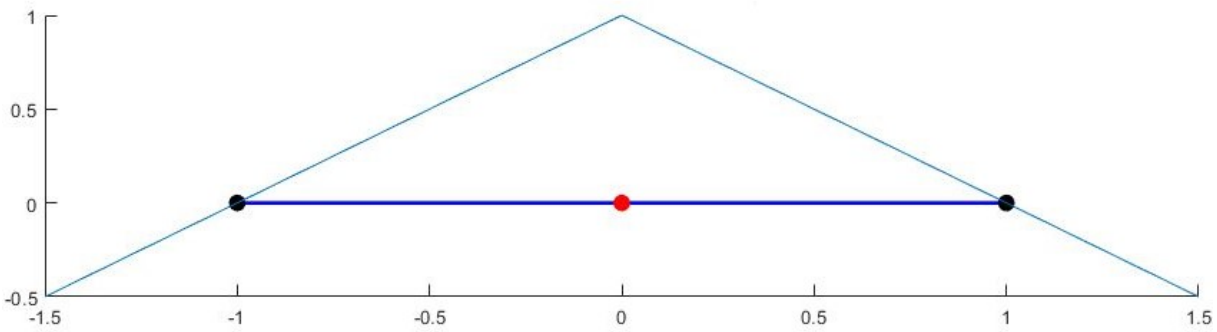
Définition 3 (carte à qualité admissible pour w^*)**.** *On dit qu'une carte a une qualité admissible pour w^* lorsque $w(J) < w^*$ partout.*

Dans la suite du texte, on parlera de carte à *qualité admissible*, implicitement pour un w^* choisi. Pour optimiser une énergie dans ce sens, il nous faut une fonctionnelle permettant de caractériser cette information. Pour cela nous utilisons la densité introduite dans [Garanzha, 2000] :

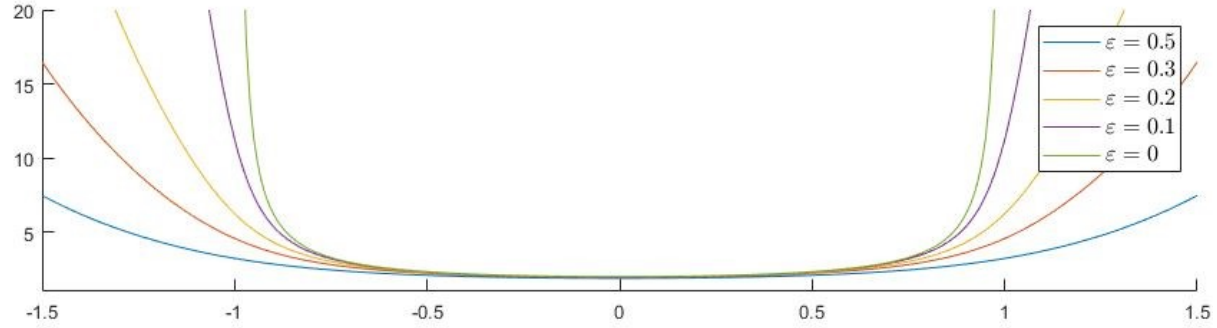
$$d_t(x) := \begin{cases} \frac{x}{1-tx} & \text{si } x < \frac{1}{t} \\ +\infty & \text{sinon} \end{cases} \tag{3.23}$$

Sur cette densité, on remarque que pour $t^* = \frac{1}{w^*}$, une carte à qualité admissible est équivalente à $d_{t^*} < \infty$. Plus précisément, d_t introduit une barrière en $\frac{1}{t}$, comme illustré dans la FIGURE 3.23. Étudions maintenant le problème variationnel suivant :

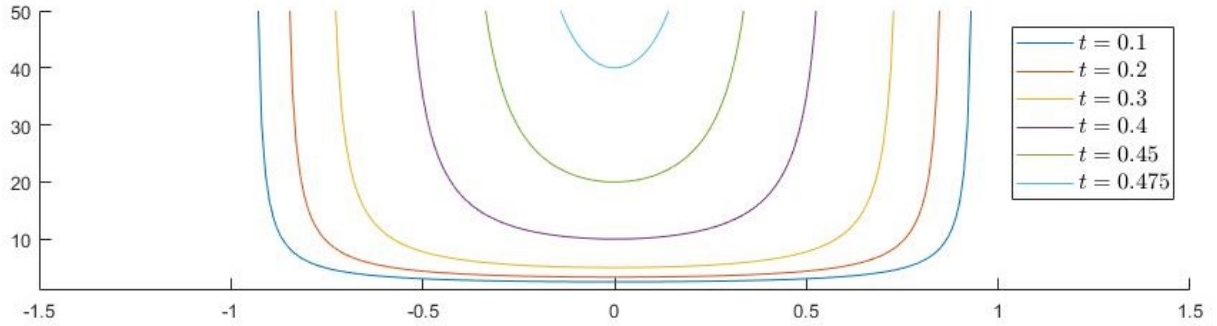
$$\arg\min_{\vec{u}} \int_{\Omega} d_t(w(J))\, dx, \tag{3.24}$$



Valeurs de $\min \det J$ en fonction de U la position du point rouge

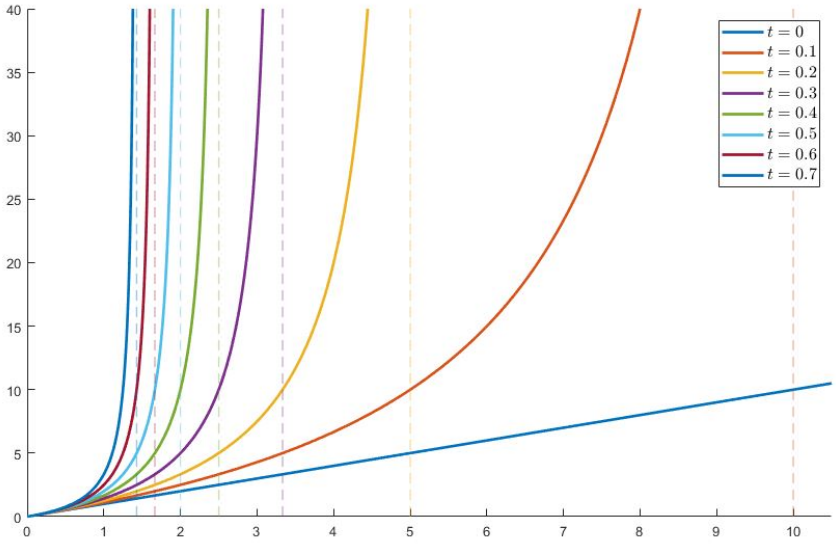


Valeurs de $F(U, \varepsilon)$ en fonction de U la position du point.



Valeurs de $W(U, t) = \frac{F(U,0)}{1-tF(U,0)}$ en fonction de U la position du point.

FIGURE 3.22 – Étude 1D des fonctions introduites. Soit trois points $(-1, 0, 1)$ reliés par deux segments, on calcule une carte en bougeant le point rouge du milieu. Le premier graphique montre les segments de références avec l'évolution du $\min \det J$ (à noté que, en 1D, $\det J = J$). Le graphique du milieu montre l'effet de ε sur $F(U, \varepsilon)$: la fonction forme progressivement une barrière empêchant les valeurs négatives de J. Le graphique du bas décrit l'effet de la fonction $W(U, t)$ introduit dans cette section, où t est l'inverse de la pire qualité. La barrière se positionne maintenant à $\frac{1}{t}$, empêchant des éléments de qualité inférieure.



Valeurs de la densité $d_t(x)$ en fonction de x.

FIGURE 3.23 – Graphique de la densité en fonction de t et x. On peut voir que d_t introduit une barrière en $\frac{1}{t}$. Faire bouger cette barrière va permettre de progressivement pénaliser les mauvaises qualités.

Bien sûr, cette intégrale est finie uniquement si

$$w(J) < \frac{1}{t} \tag{3.25}$$

Dans quel cas on peut l'écrire sous la forme :

$$\arg\min_{\vec{u}} \int_{\Omega} \frac{w(t)}{1 - tw(t)}\, dx, \tag{3.26}$$

L'idée de notre méthode est donc d'optimiser une suite de problèmes (3.26), avec un t qui tend progressivement vers t^*. A chaque itération, la barrière progressive va réduire la taille de l'ensemble de cartes "admissibles", et pénaliser fortement les pires éléments, et nous permettre d'obtenir notre carte à qualité admissible (et cela en un nombre fini de problèmes (3.26), voir dans la sous-section 3.3.2).

Remarque 8. *La densité modifiée de l'énergie de déformation est polyconvexe et ce problème variationnel est bien posé (voir sous-section 3.2.5).*

Remarque 9. *En pratique, nous utiliserons $w(J)$ sous la forme $w = (1-\theta)f + \theta g$ avec $\theta \in [0, 1]$. En effet, en remarquant que $f \geq 1$ et $g \geq 1$, cela permet d'avoir 1 comme minimum de w quelque soit la valeur de θ, et donc d'homogénéiser les résultats, tout en conservant tous les résultats précédents.*

Discrétisation

Nous pouvons discrétiser le problème variationnel (3.26) de la façon suivante :

$$\lim_{t\to t^*} \arg\min_U W(U,t), \tag{3.27}$$

$$\text{où} \quad W(U,t) := \sum_{k=1}^{\#T} \frac{w(J_k)}{1-tw(J_k)} \operatorname{vol}(T_k),$$

Nous considérerons donc qu'une carte U est à qualité admissible lorsque $W(U,t^*) < \infty$. Soit $\alpha_i > 0$ des poids strictement positifs, attribués à chaque simplexe et les fonctions

$$w_i(U) := w(J_i) \geq 1 \tag{3.28}$$

des fonctions de distorsion positives. Nous réécrivons donc $W(U,t)$:

$$W(U,t) = \sum_i \alpha_i \frac{w_i(U)}{1-tw_i(U)} = \sum_i \alpha_i d_t(w_i(U)), \tag{3.29}$$

Notez que l'expression $d_t(w(U))$ est une fonction strictement croissante de t.

Propriétés

Clarifions la signification géométrique de l'inégalité (3.28) en suivant les idées de [Garanzha, 2000]. En utilisant la fonction w sous la forme, $w = (1-\theta)f + \theta g$, on peut déduire que

$$\frac{(\frac{1}{d}\operatorname{tr} J_i^\top J_i)^{d/2}}{\det J_i} < c_1,\ c_1 = \left(\frac{1-t\theta}{t(1-\theta)}\right)^{d/2}$$

et

$$\frac{1}{2}\left(\det J_i + \frac{1}{\det J_i}\right) < c_2,\ c_2 = 1 + \frac{1-t}{t\theta}.$$

D'après l'inégalité de Reshetnyak [Reshetnyak, 1966], nous avons

$$\kappa(J_i) < (c_1 + \sqrt{c_1^2 - 1})$$

où

$$\kappa(A) = \frac{\max_j \sigma_j(A)}{\min_j \sigma_j(A)}$$

est le conditionnement singulier de la matrice A et $\sigma_j(A)$ désigne les valeurs propres de A. À partir des estimations ci-dessus, nous obtenons les limites suivantes pour $\sigma_j(J_i)$ [Garanzha, 2000] :

$$\frac{1}{\Gamma} < \sigma_j(J_k) < \Gamma, \tag{3.30}$$

où

$$\Gamma = (c_1 + \sqrt{c_1^2-1})^{(d-1)/d}(c_2 + \sqrt{c_2^2-1})^{1/d}$$

ce qui signifie que la distorsion en longueur de la carte est bornée et que dans le cas $t = 1$ la distorsion n'est pas autorisée.

Remarque 10. *Notez que dans le cas 2d et avec $\theta = 1/2$ la formule pour Γ prend la forme la plus simple*

$$\Gamma = \frac{(1+\sqrt{1-t})^2}{t}.$$

Remarque 11. *Avec des conditions aux limites appropriées,*

$$\max_t \arg\min_U W(U,t)$$

donne une carte quasi-isométrique [Garanzha et al.*, 2014].*

Remarque 12. *Dans la sous-section 3.2.5, nous avons vu que la solution du problème bien posé* (3.26) *avec une valeur arbitraire petite, mais constante t peut être considérée comme une solution régularisée de* (3.1).

3.3.2 Carte à qualité admissible en un nombre fini d'étapes d'optimisation

Dans cette sous-section, nous formulons un théorème dont le principe est très proche du théorème 1. Cette fois-ci, nous démontrons qu'une carte à qualité admissible pour $w^* = \frac{1}{t^*}$ peut être atteinte en un nombre fini d'étapes d'optimisation idéalisées.

Théorème 2. *Supposons que l'ensemble admissible des cartes à qualité admissible pour $w^* = \frac{1}{t^*}$ ne soit pas vide, à savoir qu'il existe une constante $0 < t^* < 1$ et une carte U^* satisfaisant $W(U^*, t^*) < +\infty$. Nous supposons également que nous disposons d'un algorithme de minimisation idéalisé (voir définition 2) pour un certain $0 < \sigma < 1$.*

Alors l'ensemble admissible est atteignable en résolvant un nombre fini de problèmes de minimisation dans U avec t^k fixé pour chaque problème. En d'autres termes, sous un choix approprié de la séquence de paramètres de continuation $t^k, k = 0 \dots K$, on obtient $W(U^K, t^*) < +\infty$.

Démonstration. L'idée principale est d'exposer un moyen explicite de construire une séquence croissante $\{t^k\}_{k=0}^{\infty}$ telle que la suite $\{W(U^k, t^k)\}_{k=0}^{\infty}$ est bornée par le haut. Nous pouvons alors prouver que l'ensemble admissible est atteignable en un nombre fini d'étapes par un simple raisonnement par l'absurde.

Supposons que la condition (3.13)*, $W(U^{k+1}, t^k) \leq (1-\sigma)W(U^k, t^k)$, est vérifiée à l'itération k.*

Afin de garantir que la fonction W n'est pas croissante, il suffit d'établir l'inégalité suivante :

$$(1-\sigma)W(U^{k+1}, t^{k+1}) \leq W(U^{k+1}, t^k). \tag{3.31}$$

Dénotons par w_+^{k+1} la valeur de distorsion maximale pour le maillage U^{k+1}.

$$w_+^{k+1} = \max_i w_i(U^{k+1}).$$

Nous suggérons d'utiliser la règle de mise à jour suivante pour t^{k+1} :

$$1 - t^{k+1} w_+^{k+1} = (1-\sigma)(1 - t^k w_+^{k+1}),$$

où

$$t^{k+1} = t^k + \sigma \frac{1 - t^k w_+^{k+1}}{w_+^{k+1}}. \tag{3.32}$$

D'après (3.32) et le fait que le rapport

$$\frac{1-t_1\psi}{1-t_2\psi},\ t_2 > t_1$$

est une fonction croissante de l'argument ψ il s'ensuit (3.31).

Supposons que la condition (3.14), $\min\limits_U W(U,t^k) > (1-\sigma)W(U^k,t^k)$, *est satisfaite à l'itération* k. Nous utilisons la même règle de mise à jour (3.32) et sous l'hypothèse que $t^{k+1} < t^*$ nous obtenons

$$W(U^{k+1},t^{k+1}) \le \frac{1}{1-\sigma}W(U^{k+1},t^k) \le$$

$$\frac{1}{(1-\sigma)^2}\min_X W(U,t^k) \le \frac{1}{(1-\sigma)^2}W(U^*,t^k) \le$$

$$\le \frac{1}{(1-\sigma)^2}W(U^*,t^*).$$

Supposons maintenant que pour une suite infinie de U^k, t^k nous obtenions $t^k < t^*$. L'inégalité suivante est donc vraie :

$$t^k - t^0 = \sigma\sum_j(\frac{1}{w_+^{j+1}} - t^j) \le 1$$

Dans la somme infinie, chaque terme est strictement positif et nous pouvons donc extraire une sous-suite

$$(\frac{1}{w_+^{j_s+1}} - t^{j_s}) \to +0,$$

et comme $w_+^{j_s+1}$ est strictement positif, cela donne

$$1 - t^{j_s}w_+^{j_s} \to +0,$$

qui se trouve être le dénominateur dans les termes de l'équation (3.29). Le fait que ce dénominateur tende vers 0 pour de grandes valeurs de k contredit évidemment le fait que la fonctionnelle soit bornée.

■

Remarque 13. *On peut utiliser une autre règle de mise à jour pour* t^{k+1} *en supposant que*

$$W(U^{k+1},t^{k+1}) = \frac{1}{1-\sigma}W(U^{k+1},t^k). \tag{3.33}$$

Cette équation non linéaire peut être résolue par la méthode de Newton. Puisque

$$\frac{\partial W}{\partial t} > 0, \frac{\partial^2 W}{\partial t^2} < 0,$$

chaque estimation de la méthode de Newton est plus petite que la solution exacte de (3.33), *à condition que l'estimation initiale soit définie par* (3.32).

Remarque 14. *En pratique, comme pour [Garanzha* et al.*, 2021a], nous attribuons*

$$\sigma = \max(1 - \frac{W(U^{k+1},t^k)}{W(U^k,t^k)}, \sigma_0),$$

où $\sigma_0 > 0$ *est une constante.*

Notez que le théorème du "nombre fini d'étapes" ne peut plus être appliqué lorsque la taille du domaine admissible converge vers zéro, ce qui peut arriver lorsque le paramètre t converge vers la valeur maximale atteignable. Pour traiter les cas marginaux, nous attribuons au paramètre σ_0 une valeur assez faible (10^{-3}). Pour le moment, il n'est pas clair que nous pouvons atteindre numériquement la déformation avec la meilleure distorsion possible.

3.3.3 Résultats et discussion

Pour illustrer l'efficacité de notre méthode, nous nous comparons à la méthode de la section précédente, puis à des méthodes similaires de la littérature, d'abord sur un maillage classique de la littérature, "Lucy", et ensuite nous discutons les propriétés des cartes obtenues. Nous complétons les comparaisons effectuées dans la section précédente (FIGURE 3.18, FIGURE 3.19 et FIGURE 3.27) avec notre nouvelle méthode.

Avant cela, l'implémentation de l'énergie ne nécessite qu'une simple composition de dérivés. Il faut tout de même faire à attention à un point : la stabilité numérique des algorithmes. En effet, les termes de barrières sont beaucoup plus abrupts que dans la section précédente, car ici l'énergie d'un élément peut rapidement passer à l'infinie. Une attention particulière est donc nécessaire dans le choix, ou bien l'implémentation, de l'algorithme de *linesearch* [Moré et Thuente, 1994] (que ce soit pour une méthode de Newton, ou L-BFGS).

Carte de Lucy

Afin de vérifier les performances de notre approche variationnelle (3.27) à améliorer les éléments de plus mauvaise qualité dans des cartes, nous répétons l'expérience d'aplatissement optimal (conformes et quasi-isométrique [16]) du modèle "Lucy" (FIGURE 3.18 et FIGURE 3.19), cette fois avec notre nouvelle méthode.

Carte conforme : comparaison avec *Least Squares Conformal Maps* Nous avons d'abord calculé trois cartes conformes discrètes (voir FIGURE 3.24- ligne supérieure) :

1. La plus simple à calculer est *Least Squares Conformal Maps* (LSCM) [Lévy *et al.*, 2002], cette méthode très répandue nécessite la résolution d'un système linéaire avec une matrice symétrique définie positive. L'idée derrière LSCM est de calculer une approximation par éléments finis P^1 des conditions de Cauchy-Riemann sur tous les triangles du maillage.
2. Ensuite, nous avons résolu l'équation (3.26) avec $\theta = 0$, $t = 0$ (carte conforme discrète équilibrée) ;
3. Enfin, nous avons résolu l'équation (3.26) avec $\theta = 0$ tout en maximisant t (carte conforme discrète optimale).

Pour comparer la qualité des cartes, nous utilisons le conditionnement de la matrice jacobienne $\frac{\sigma_1(J)}{\sigma_2(J)}$, où σ_1 et σ_2 représentent les valeurs singulières de J. La grande majorité des éléments du maillage dans la méthode LSCM est cartographiée avec une très petite erreur de déformation, cependant le nombre d'éléments mal déformés est considérable et certains éléments sont même inversés. Pour une carte conforme discrète équilibrée, la mesure de déformation de forme maximale est égale à $5,13$, alors que notre optimisation permet de réduire la pire qualité à $2,38$. Notons que la maximisation de t dans (3.27) tend à répartir la distorsion sur le maillage au lieu de l'accumuler sur des éléments isolés.

16. Pour rappel, une carte conforme préserve les angles. Une carte quasi-isométrique cherche à conserver les distances.

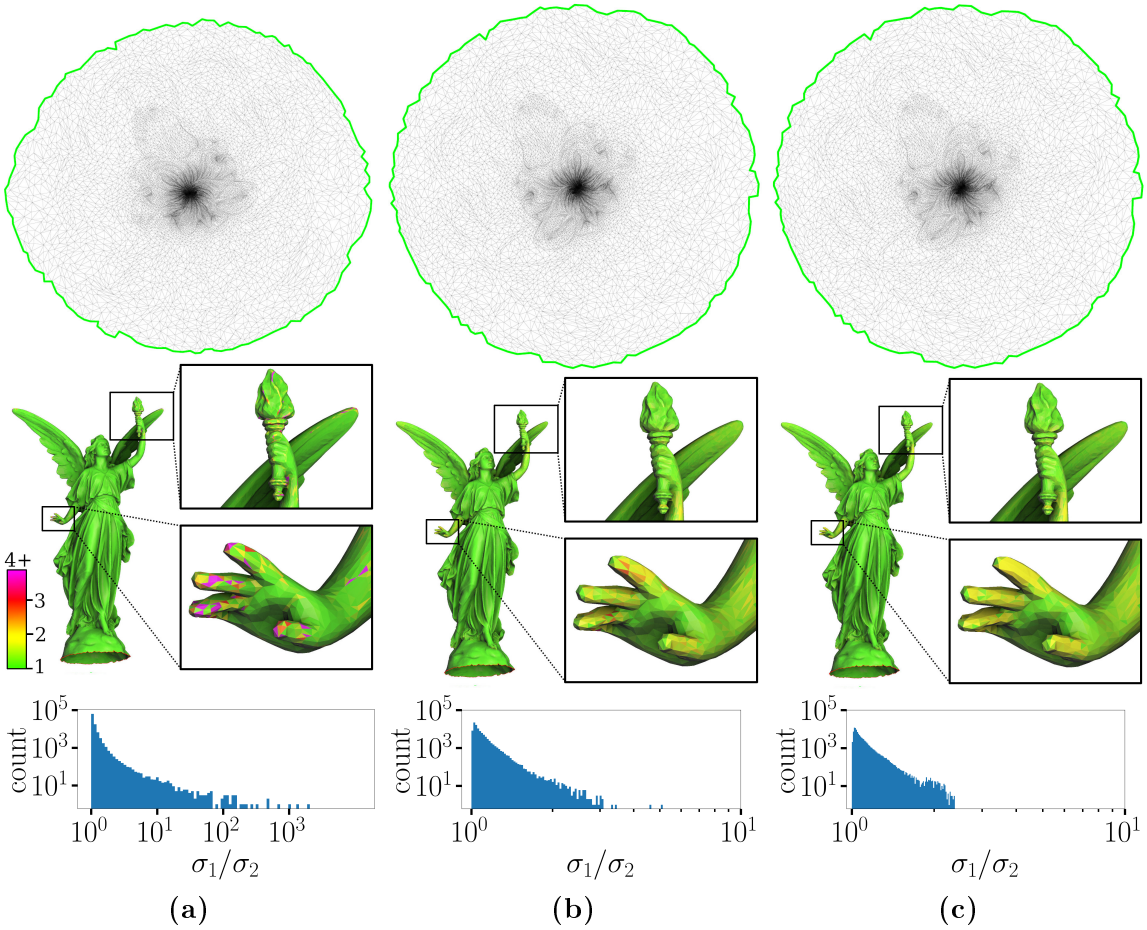


FIGURE 3.24 – Comparaison de trois cartes conformes discrètes. **(a) :** *Least Squares Conformal Maps*, **(b) :** équilibrée ($t = 0$), **(c)** optimale. **Ligne du haut :** Cartes, **Ligne du milieu :** distribution de la qualité des éléments sur la surface, **Ligne du bas :** histogrammes log-log de la qualité des éléments.

Cartographie quasi-isométrique : comparaison avec SA Nous comparons ensuite notre méthode à *Simplex Assembly* (SA) [Fu et Liu, 2016]. SA est une méthode pour calculer des cartes sans inversion avec une distorsion bornée sur des maillages simpliciaux. L'idée est de projeter chaque simplexe dans l'espace sans inversion et à distorsion bornée. Après avoir désassemblé le maillage, les simplexes sont ensuite ré-assemblés en minimisant la distorsion de la carte, tout en gardant la carte (à qualité) admissible.

La FIGURE 3.25 fournit une comparaison de qualité de SA avec notre carte quasi-isométrique ($\theta = \frac{4}{5}$) pour un calcul de carte à bord libre de "Lucy". Tout d'abord, on peut observer une réduction considérable de la pire distorsion de la forme des cellules ainsi qu'une réduction considérable de la distorsion de la surface après la carte optimale par rapport à la carte équilibrée. Dans la solution équilibrée, nous observons des zones minces localisées avec la plus grande distorsion, tandis que la maximisation de t étale ces zones, en répartissant la distorsion uniformément sur la surface.

Ensuite, notre méthode donne des cartes systématiquement meilleures par rapport à la méthode *Simplex Assembly*. En particulier, le pire conditionnement $\frac{\sigma_1(J)}{\sigma_2(J)}$ est de 16,5 pour SA et de 14,9 pour notre méthode. Le maximum du déterminant jacobien est égal à 5,76 pour SA et 4,63

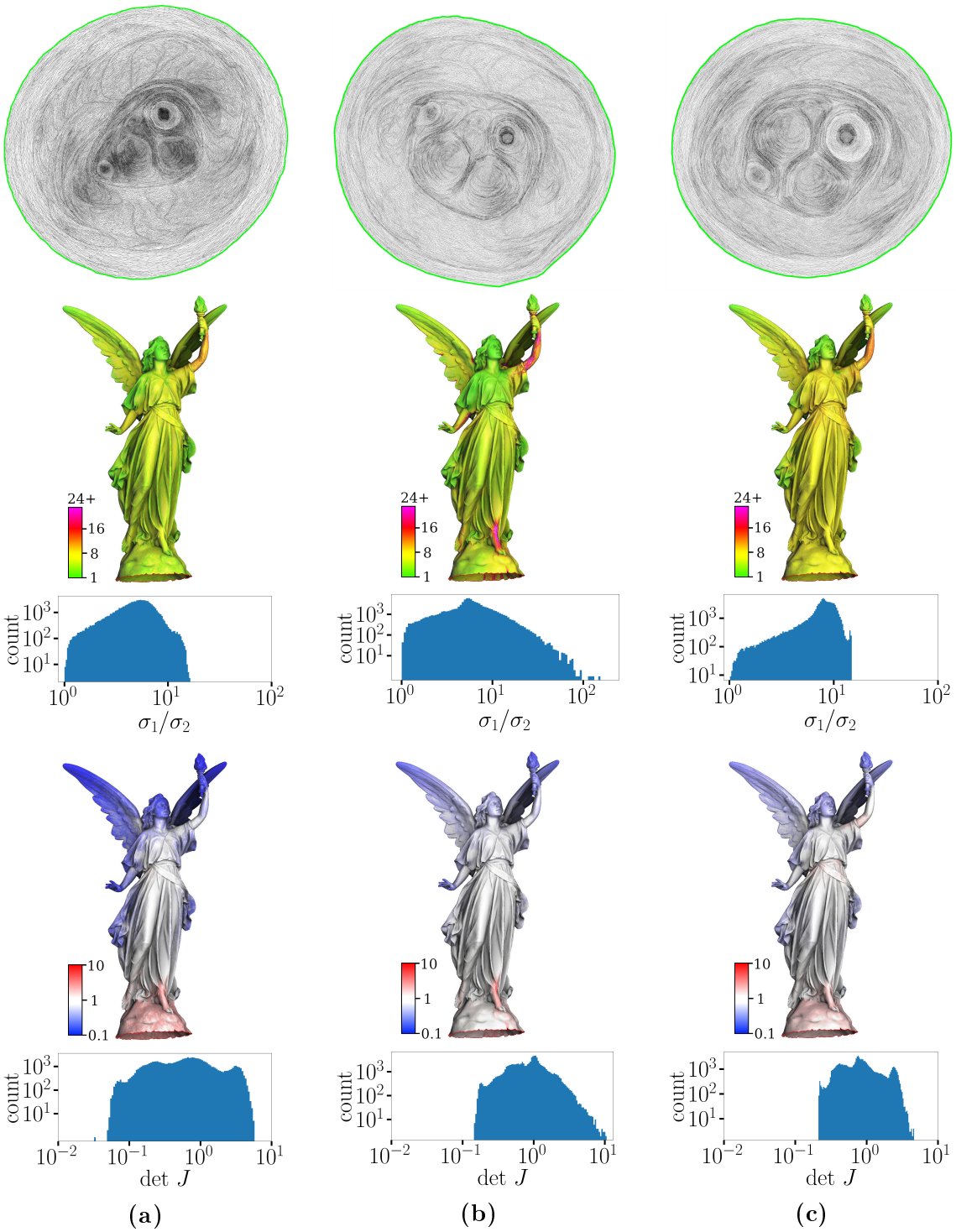


FIGURE 3.25 – Trois cartes quasi-isométriques pour le maillage "Lucy". **(a) :** *Simplex Assembly*, **(b) :** équilibrée $t = 0, \theta = \frac{4}{5}$, **(c) :** optimale, $\theta = \frac{4}{5}$. La qualité des éléments et leur répartition sur la surface sont illustrées par des histogrammes log-log et les couleurs correspondantes sur le maillage.

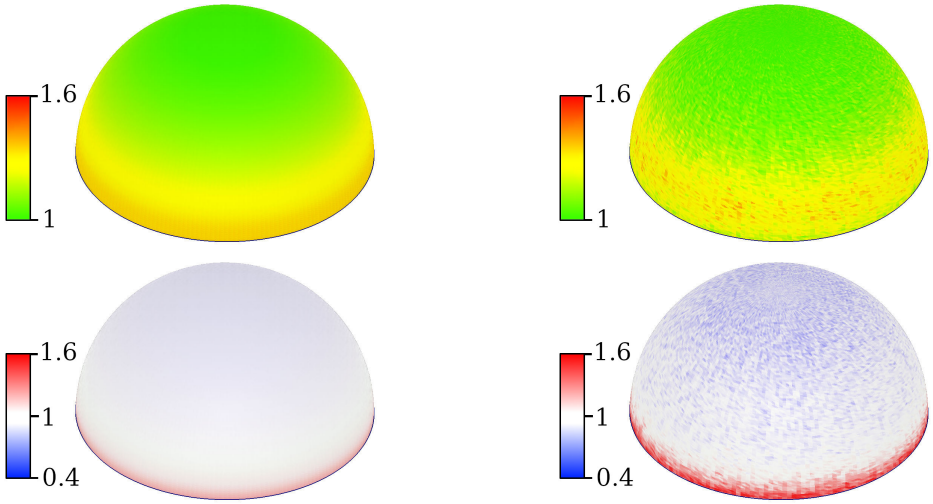


FIGURE 3.26 – Carte d'une demi-sphère sur un disque : notre solution avec $\theta = \frac{4}{5}$ **(a)** vs Simplex Assembly **(b)**. **Ligne du haut :** conditionnement de la matrice jacobienne, **Ligne du bas :** déterminant de la matrice jacobienne.

pour notre méthode. Le minimum du déterminant jacobien est égal à 0,03 et 0,21, respectivement.

Comportement des méthodes

Simplex Assembly Notez que si, en 2D, SA a la même fonction objectif que notre méthode, la façon dont SA pose le problème (minimisation de la distorsion maximale) conduit à des solutions non lisses. La FIGURE 3.26 montre un exemple : si nous transposons une demi-sphère sur un disque, notre méthode fournit une solution lisse, alors que le résultat de SA est bruité et perd la symétrie angulaire. Ce résultat est confirmé par les indicateurs numériques : le maximum du déterminant jacobien dans notre cas est 1,58, alors que SA donne 1,73. Le conditionnement varie entre 0,91 et 1,27 dans notre cas, et entre 0,73 et 1,68 pour SA. Ce cas test est intéressant, car il fournit un aplatissement analytique qui possède la plus petite constante de quasi-isométrie connue Γ, égale à $\sqrt{\pi/2}$. Cette mise en correspondance peut être obtenue par projection isométrique des méridiens sur des segments de droite sur le plan en partant du pôle Nord tout en gardant la projection angulaire uniforme. De toute évidence, les valeurs singulières de cette carte sont comprises entre 1 et $\pi/2$, et en utilisant la meilleure mise à l'échelle, on obtient $\Gamma = \sqrt{\pi/2} \approx 1,253$. Notez que dans le cas où $\theta = 1/2$, nous obtenons le seuil de qualité $t = 0,9683$, donc en utilisant la formule (3.30) nous obtenons l'estimation $\Gamma = 1,43$, tandis que la valeur calculée pour la projection numérique est égale à $\sqrt{\sigma_{\max}/\sigma_{\min}} = 1,278$, ce qui est à moins de 2% de la meilleure valeur connue.

Large-scale Bounded Distortion Mappings *Large-scale Bounded Distortion Mappings* (LBD) [Kovalsky *et al.*, 2015] est une autre méthode de calcul de cartes à bord libre avec laquelle nous nous sommes comparés dans la section précédente, pour notre méthode de pénalisation, voir FIGURE 3.27. Étant donnée une carte d'entrée (potentiellement avec des éléments inversés), LBD recherche une carte admissible aussi proche que possible de la carte d'entrée, mais satisfaisant certaines contraintes telles que des bornes de rotation et de distorsion. La FIGURE 3.27 fournit

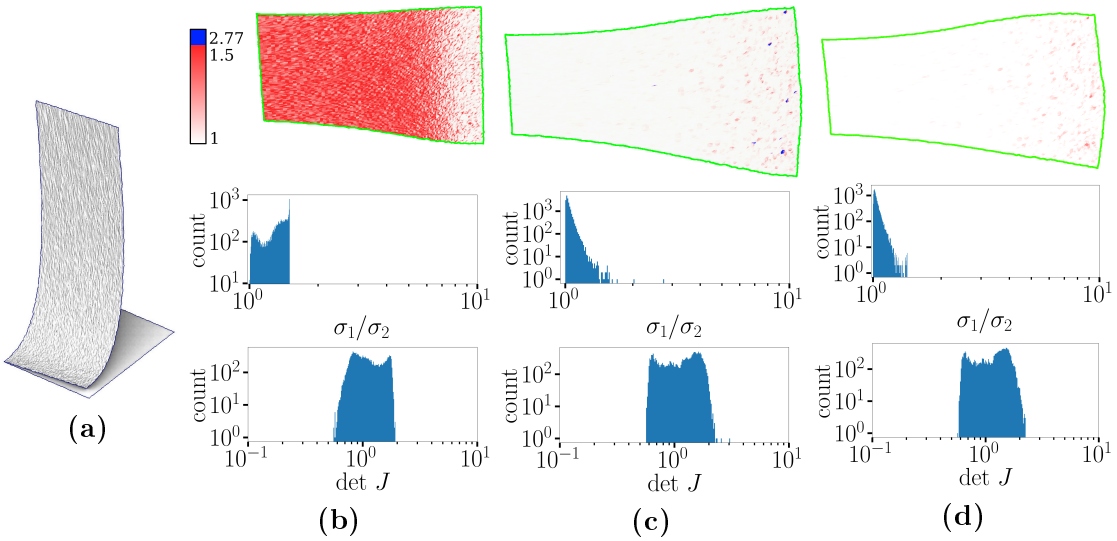


FIGURE 3.27 – Comparaison de la méthode LBD et de la nôtre **(a) :** La surface 3D à aplanir est un maillage triangulaire régulier d'un patch carré qui a été soulevé et bruité. **(b) :** La carte obtenue par LBD. **(c) :** carte équilibrée ($t = 0$, $\theta = \frac{4}{5}$). **(d) :** carte optimale ($\theta = \frac{4}{5}$). **Ligne du haut :** carte de **(a)**, les couleurs correspondent à la qualité des éléments (conditionnement du jacobien). **Rangées du milieu et du bas :** histogrammes log-log de la qualité des éléments.

une comparaison de LBD avec notre méthode. La surface 3D à aplanir est un maillage régulier simplicial d'un patch rectangulaire qui a été soulevé et bruité. LBD a une optimisation des bornes de la distorsion, ainsi la qualité du pire élément de la carte par LBD est meilleure que dans notre carte. Notons cependant que LBD a beaucoup d'éléments proches de la pire qualité, alors que notre méthode est basée sur la théorie de l'élasticité, et fournit une meilleure distribution de la qualité globale.

Conclusion

Nous pensons que cette approche est très prometteuse, parce qu'elle permet à la fois de coupler les qualités de l'énergie de la section précédente (polyconvexité, adaptabilité), et d'obtenir des cartes à qualité uniforme, *i.e.* sans sacrifier la qualité de certains éléments. Ces résultats sont par ailleurs compilés dans un article en cours de soumission à TOG [Garanzha *et al.*, 2022].

3.4 Extensions pratiques

Dans cette section nous discutons de deux sujets qui touchent aux utilisations pratiques des méthodes de calcul de cartes précédentes. Dans un première temps nous analysons les problèmes de superpositions qui apparaissent lorsque l'on travaille avec des cartes à bords libres (sous-section 3.4.1). Ensuite nous présentons une application très utile pour le maillage, et pas forcément triviale, le lissage de maillages hexaédriques (sous-section 3.4.2).

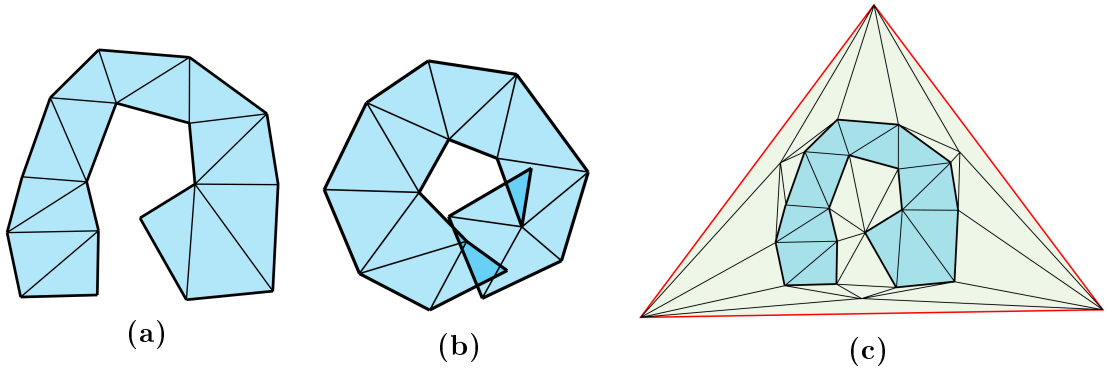


FIGURE 3.28 – Les limites du calcul de carte à bord libre. **(a)** Maillage original. **(b) :** Une carte présentant deux types de problèmes, à savoir un chevauchement global et enroulement autour d'un sommet du bord. **(c) :** Ces deux problèmes peuvent être évités en intégrant le maillage dans une triangulation externe.

3.4.1 Bijectivité dans les cartes à bords libres [Garanzha *et al.*, 2021b]

Calculer une carte à bord non contraint est un problème récurrent, que ce soit pour calculer des cartes conformes, ou plus intéressant pour nous, des polycubes[17]. Or, comme le montre la FIGURE 3.28-**(b)**, cette application révèle une nouvelle difficulté : les superpositions dans la carte. Ces superpositions peuvent être réparties en deux catégories :

— les superpositions globales, deux parties éloignées sur l'objet se retrouvent au même endroit dans la carte ;
— la carte s'enroule autour d'un point.

Dans ces deux situations, aucun élément de la carte n'est retourné. Pourtant celles-ci posent manifestement des problèmes pour les méthodes utilisant la carte. En pratique, il convient tout de même de distinguer les deux cas : en effet, dans la première situation, la carte est tout de même localement injective partout, alors que dans la deuxième, ce n'est pas le cas au point où la carte s'enroule. Plus que cela, les méthodes d'extractions d'hexaèdres à partir de carte, telle que HexEx [Lyon *et al.*, 2016], sont faites pour fonctionner avec des cartes seulement localement injectives. Cela nous invite donc à nous concentrer sur le deuxième point, pour lequel nous apportons une solution pratique dans cette sous-section.

Avant de présenter notre approche, il convient de noter : [Aigerman et Lipman, 2013] a montré que ces situations arrivent uniquement lorsque le bord de l'objet s'auto-intersecte dans la carte. Plonger l'objet dans un sur-maillage pour construire la carte, comme illustré sur la FIGURE 3.28-**(c)** permet donc d'éviter ces problèmes. Cette stratégie présente deux défauts : la qualité du surmaillage contraint fortement les déformations possibles du bord de l'objet ; et il n'est pas toujours trivial de pouvoir construire une triangulation externe (ou alors celle-ci peut devenir très contraignante !), certaines applications comme les paramétrisations globales nécessitant même des superpositions globales...

Nous proposons donc une méthode d'ajout de simplexes qui permet d'éviter que la carte s'enroule autour d'un point. Nous présentons d'abord notre algorithme d'ajout, en 2D puis en 3D, discutons des contraintes que cela rajoute sur les cartes construites, puis nous montrons quelques résultats. Enfin, nous discutons d'une petite particularité des paramétrisations globales.

17. Par exemple en déformant progressivement l'objet, et donc en utilisant de contraintes "faibles".

Algorithme d'ajout 2D

Afin d'expliquer l'idée de l'algorithme d'ajout, nous allons travailler sur les voisinages de chaque point dans le maillage, avec traitement dépendant des valences de ceux-ci. Nous traitons d'abord les superpositions à l'intérieur du maillage, puis celles sur le bord. À l'intérieur, la somme des angles autour d'un sommet doit être égale $k2\pi$, avec $k > 0$. Lorsque $k > 1$, nous disons qu'il y a une k-superposition. En effet, dans ce cas la, la triangulation "tourne" k fois autour du point (voir FIGURE 3.29 et FIGURE 3.5).

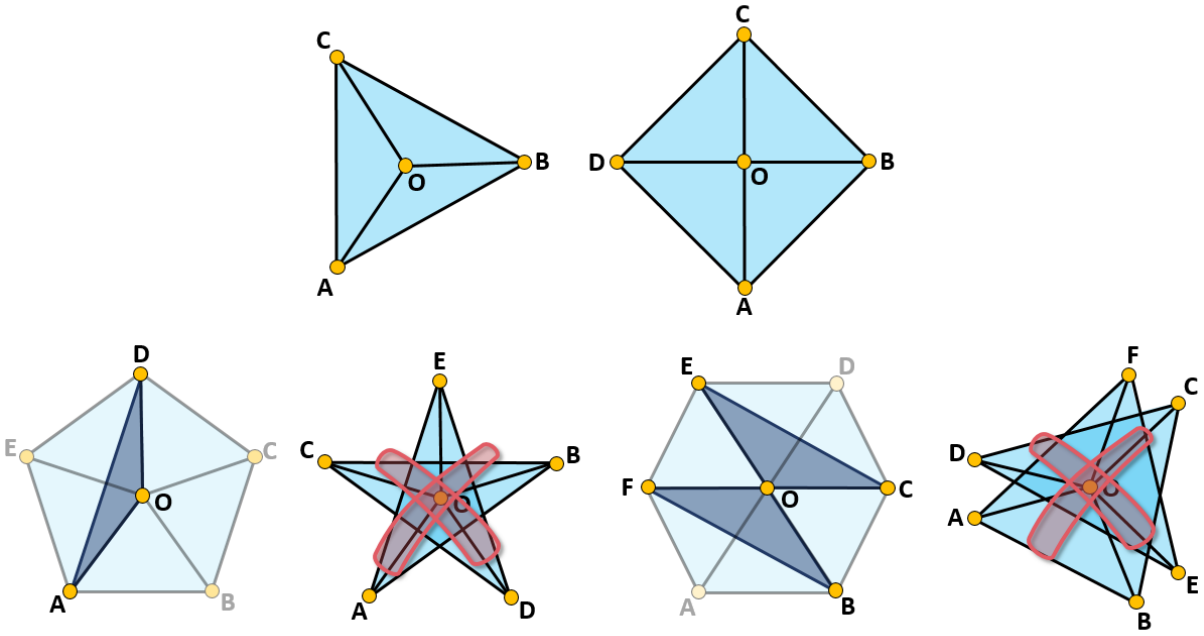


FIGURE 3.29 – Le concept de protection des voisinages de sommet 2D par des triangles fantômes supplémentaires : les sommets de valence 3 et 4 ne permettent pas de superposition. Pour les valences 5, ajouter un triangle supplémentaire (se chevauchant) empêche une superposition, puisqu'il réduit essentiellement la configuration à une valence 4. Pour la valence 6, 2 triangles suffisent à empêcher toute superposition.

Les sommets de valence 3 et 4 ne permettent pas de k-superposition puisqu'un total autour du sommet supérieur ou égal à 4π nécessiterait des angles de triangle supérieurs à π, rendant le maillage inadmissible. L'idée de notre algorithme est donc d'ajouter aux voisinages des points de valences élevées des triangles fantômes entre le sommet et ses voisins, pour qu'ils possèdent une valence "fantôme" de 3 ou 4, voir FIGURE 3.29. Pour un sommet de valence 5, nous pouvons ajouter un seul triangle, tandis que pour une valence 6, deux triangles supplémentaires sont suffisants. Par conséquent, le nombre d'inconnues du problème variationnel est fixé, tandis que la définition (3.2) de l'ensemble admissible $\mathcal{A}$ est augmentée par des inégalités supplémentaires qui coupent des sous-ensembles disjoints liés à ce sommet de maillage, reflétant ces triangles supplémentaires.

Lorsque le maillage est surfacique, nous étalons les voisinages de chacun des points pour obtenir un voisinage de référence sur lequel travailler. Sur chacun des voisinages, l'algorithme est le suivant.

L'idée est de réduire la valence en ajoutant des triangles fantômes jusqu'à ce qu'un simple

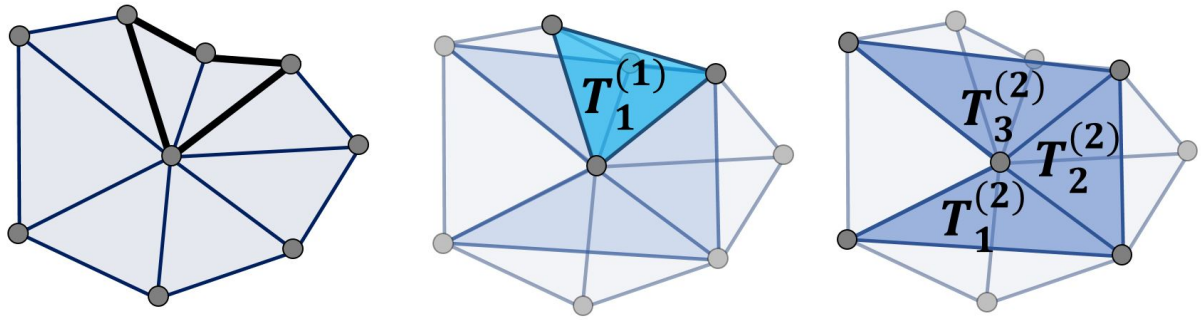


FIGURE 3.30 – Agrégation récursive de triangles adjacents pour créer des triangles de protection fantômes. (a) Voisinage initial de 8 sommets, (b) Ajout d'un triangle fantôme bleu de niveau 1, (c) Ajout de trois triangles fantômes de niveaux 2 pour obtenir une valence 4.

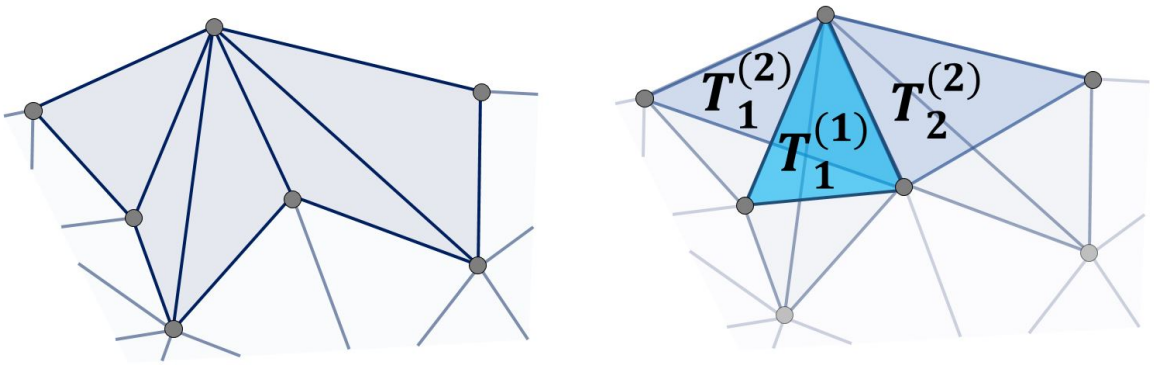


FIGURE 3.31 – Algorithme de bord standard : l'ajout récursif de triangles fantômes pour obtenir une valence 3 (donnant uniquement 2 triangles sur le bord)

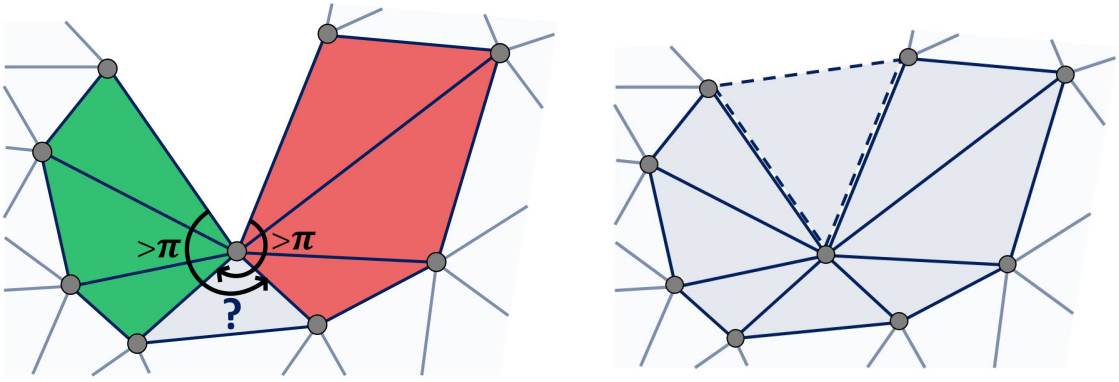


FIGURE 3.32 – Cas particulier sur le bord : impossibilité d'obtenir une valence 3. Ajouter un triangle fantôme extérieur transforme le sommet externe en un sommet interne.

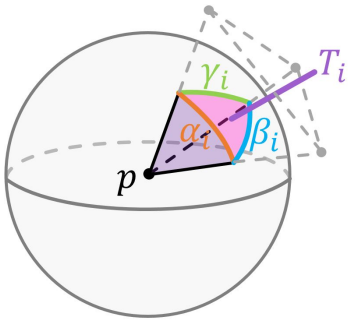


FIGURE 3.33 – L'intersection entre le $i^{\text{ème}}$ tétraèdre incident à p avec une boule de centre p produit un cône polyédrique à trois faces. Au niveau de la sphère unité, ce cône a la forme d'un triangle sphérique, que nous notons T_i, dont les arêtes représentent les angles sur les facettes incidentes à p : α_i, β_i et γ_i.

ajout suffise pour obtenir une valence 3 ou 4. Nous commençons donc par trouver la paire de triangles qui forme le quadrilatère (en général non convexe) qui donnerait le triangle fantôme $T_1^{(1)}$ de meilleure qualité, comme le montre la FIGURE 3.30(a)–(b), et nous considérons que le point "recouvert" ne fait plus partie du voisinage. Nous répétons cette opération jusqu'à ce qu'il existe un ajout de triangle permettant d'obtenir une valence 3 ou 4, avec un triangle ajouté couvrant seulement un unique point, que nous appliquons, comme dans la FIGURE 3.30(c) avec $T_i^{(2)}, i = 1, 3$.

Traitement du bord Pour le bord nous distinguons deux cas : dans le premier nous pouvons appliquer un algorithme similaire au précédent, illustré sur la FIGURE 3.31, le seul élément différent est la dernière étape où nous nous arrêtons lorsque qu'on peut couvrir le reste des sommets avec uniquement 2 triangles fantômes, qui vont assurer que la somme des angles ne soit pas supérieure à 2π (et même strictement inférieure, *i.e.* , le bord ne s'auto-intersecte pas autour du point). Le deuxième cas intervient lorsqu'il n'existe pas d'ajout de sommet permettant de créer deux triangles couvrant le voisinage, comme illustré sur la FIGURE 3.32 à gauche. Dans cette situation, nous ajoutons un triangle à l'extérieur pour fermer le voisinage, et nous appliquons ensuite l'algorithme utilisé pour les sommets intérieurs.

Extension vers la 3D

Dans le cas de la 3D, il n'est pas nécessaire de faire une analyse fastidieuse des valences des sommets. Considérons l'ensemble des cônes polyédriques à trois côtés provenant du sommet intérieur p. Les faces du cône correspondent aux trois faces du tétraèdre incident à p. L'intersection du $i^{\text{ème}}$ cône avec la sphère unité définit le triangle sphérique T_i avec les angles $\alpha_i, \beta_i, \gamma_i$ (voir FIGURE 3.33). Ces angles coïncident avec trois angles dièdres du cône/tet. Le nombre de triangles sphériques n_t est égal au nombre de tétraèdres incidents à p, tandis que le nombre n_v de sommets de la triangulation sphérique est égal au nombre d'arêtes du maillage provenant de p. La relation évidente $n_t = 2n_v - 4$ s'applique. Puisque tous les tets ont un volume algébrique positif, l'aire de chaque triangle sphérique est positive.

$$\textbf{Aire}(T_i) = \alpha_i + \beta_i + \gamma_i - \pi > 0.$$

Alors

$$\sum_{i=1}^{n_t} \mathbf{Aire}(T_i) = \sum_{k=1}^{n_v} \Delta\alpha_k - n_t\pi$$

où $\Delta\alpha_k$ est la somme des angles autour du $k^{\text{ième}}$ sommet. Si nous imposons la condition

$$\Delta\alpha_k = 2\pi,$$

ce qui signifie qu'aucune 2-superposition n'est autorisée pour les arêtes du maillage tétraédrique, nous obtenons

$$\sum_{i=1}^{n_t} \mathbf{Aire}(T_i) = 2\pi n_v - n_t\pi = 4\pi,$$

ce qui signifie qu'une 2-superposition sur le sommet est interdite.

Par conséquent, nous considérons un algorithme quasi-2D qui interdit les superpositions pour les éventails tétraédriques autour de chaque arête du maillage tetraèdrique. La FIGURE 3.34 montre un exemple de 2-covering pour des tétraèdres autour d'une arête.

La FIGURE 3.35 illustre l'algorithme en 3D. L'ensemble des tétraèdres autour d'un bord définit la disposition des triangles en 2D autour du sommet dans le plan orthogonal à l'arête. L'idée de l'algorithme 2D peut être utilisée pour créer des tétraèdres fantômes autour de l'arête en conservant un éventail d'arêtes géométriquement et topologiquement correct. La seule différence est qu'il faut utiliser les critères de qualité 3D pour créer les meilleurs éléments à chaque étape.

Traitement du bord Pour les sommets du bord, nous pouvons rencontrer des situations régulières et irrégulières. Pour une situation régulière (voir FIGURE 3.36–à gauche), il existe une arête interne e_{int} qui n'est pas loin d'être orthogonale au bord. On peut alors appliquer l'algorithme quasi-2D autour de toutes les arêtes du bord et ainsi que e_{int}, en créant des tétraèdres (e_{int}, T_i) nécessaires. Cela permet de garantir que l'angle sphérique total autour du sommet est inférieur à 4π. Pour le cas irrégulier montré sur la FIGURE 3.36–à droite, il n'existe pas de bonne arête interne pour le sommet p, on doit donc ajouter un nouveau sommet fantôme p^* à l'extérieur, créant ainsi une nouvelle arête et fermant le bord en ajoutant tous les tétraèdres avec les sommets p, p^*, p_i, p_{i+1} pour tous les sommets du bord p_i de voisins à p. Avec cela, on peut traiter p comme un sommet interne via l'arête pp^*.

Dans le cas "irrégulier", le sommet p devient le sommet interne et l'analyse de l'angle sphérique total peut être appliquée directement, garantissant l'absence de superposition. Dans le cas régulier, sur la partie du bord adjacent à p, chaque face satisfait à la condition de visibilité depuis n'importe quel point situé sur l'arête interne e_{int}, et l'angle dièdre total autour de cette arête est égal à 2π. Par conséquent, l'arête e_{int} admet une prolongation droite en dehors du domaine de calcul de telle sorte que l'on peut compléter le demi-éventail tétraédrique par le demi-éventail complémentaire extérieur, créant ainsi un éventail tétraédrique complet satisfaisant la contrainte de l'angle dièdre pour toutes les arêtes internes.

En principe, nous pourrions ajouter une couche prismatique de cellules fantômes autour de toutes les faces libre du bord. Cela garantirait l'absence de superposition, mais le pré-traitement s'avère coûteux et augmente considérablement le nombre de tétraèdres fantômes. À noter cette différence avec l'algorithme 2D : en 3D la taille du problème variationnel augmente.

Effet sur l'ensemble admissible

En général, notre algorithme permet de réduire la taille de l'ensemble admissible $\mathcal{A}$ en éliminant ses composantes parasites. Cependant, elle peut potentiellement non seulement rendre le

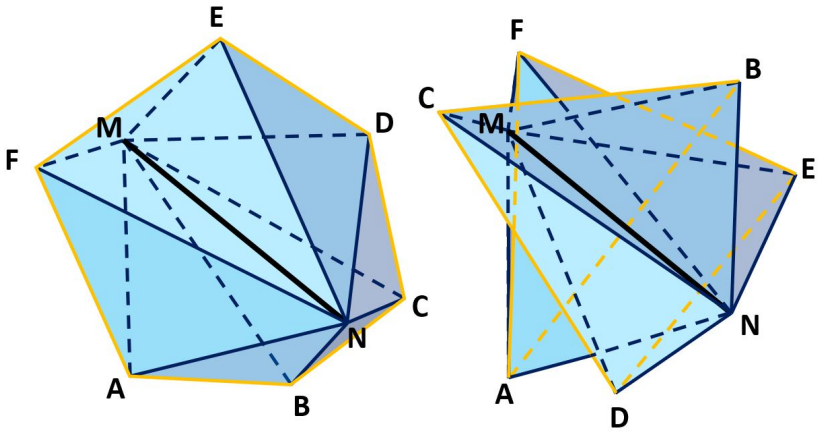


FIGURE 3.34 – (À gauche) un éventail régulier de 6 tétraèdres partageant l'arête commune MN. (À droite) une 2-superposition autour de l'arête MN.

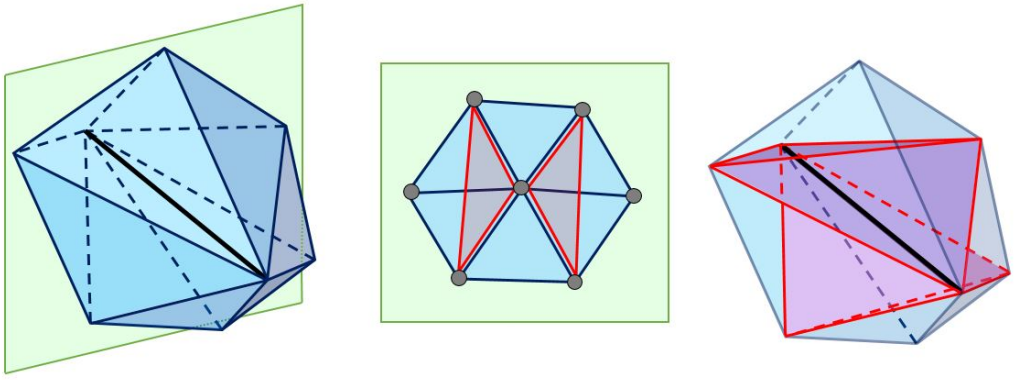


FIGURE 3.35 – L'ajout de tétraèdres fantômes autour d'une arête est équivalent à un problème 2D.

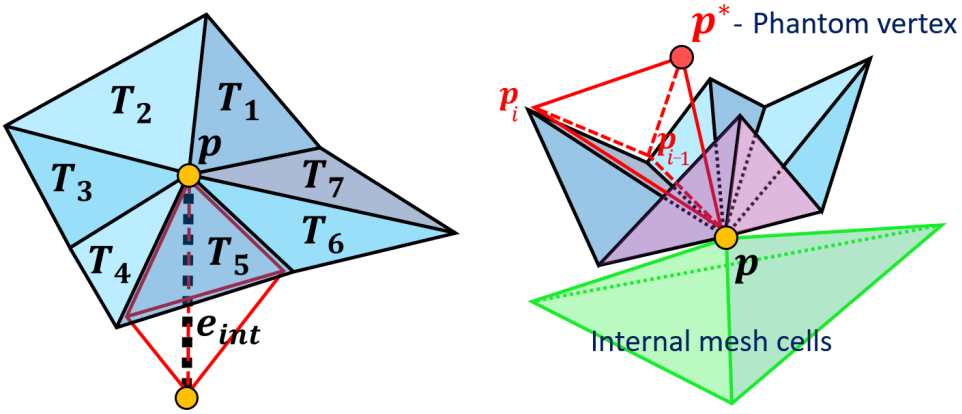


FIGURE 3.36 – (À gauche) Situation régulière : L'algorithme basé sur la 2D est appliqué sur les arêtes du bord, ainsi que sur e_{int}. (À droite) Situation irrégulière : un sommet extérieur p^* est ajouté, créant une configuration interne.

problème d'optimisation plus rigide, mais aussi surréduire l'ensemble admissible, rendant le problème insoluble. Bien qu'il soit très difficile de faire des déclarations constructives sur la structure de l'ensemble admissible, il semblerait, d'après nos tests numériques, que pour les problèmes à bord libres, notre algorithme ne crée pas de problème de démêlage insoluble. De plus, puisque tous les éléments fantômes peuvent être éliminés après la procédure de démêlage, la qualité du maillage résultant n'est pas affectée. Par ailleurs, l'utilisation des éléments fantômes simpliciaux est très naturelle pour l'optimisation de maillage en général puisque les éléments non simpliciaux (les quadrilatères étant l'exemple le plus simple) sont modélisés de cette manière.

Cependant, lorsque l'on considère un problème contraint qui combine des bords libres avec des sommets fixes, nos algorithmes peuvent potentiellement aboutir à un ensemble admissible vide. Considérons le voisinage FIGURE 3.37–à gauche dont les sommets jaunes sont libres. Nous imposons une déformation des points bleus vers la position FIGURE 3.37–à droite. Le triangle fantôme ajouté à gauche est bien valide. Il n'existe cependant pas de configuration géométrique valide pour n'obtenir que des triangles non inversés (à droite sur la même figure). L'ensemble admissible $\mathcal{A}$ devient donc vide lorsqu'on ajoute le triangle fantôme.

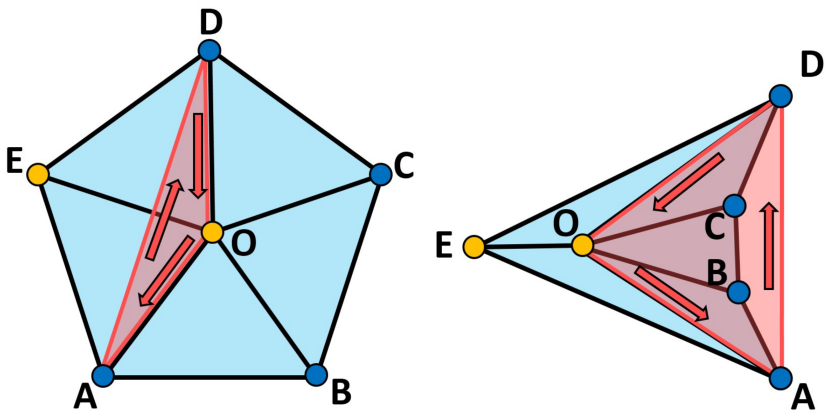


FIGURE 3.37 – (À gauche) Triangle fantôme correct pour le sommet O, (à droite) l'aire du même triangle fantôme est forcée d'être négative en raison de la présence de sommets contraints (bleu foncé).

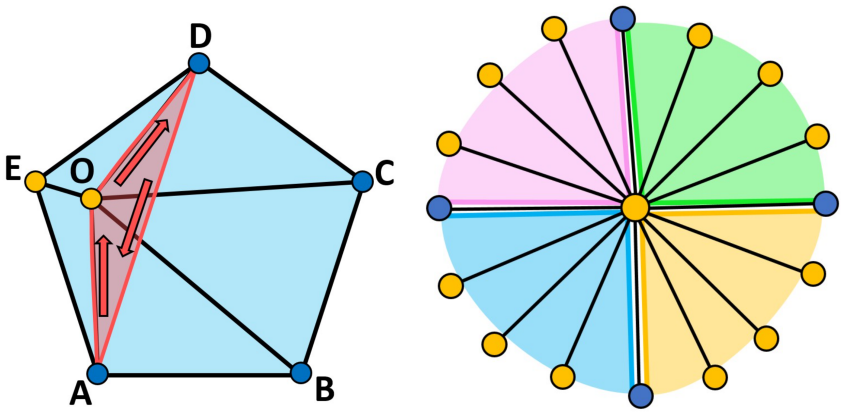
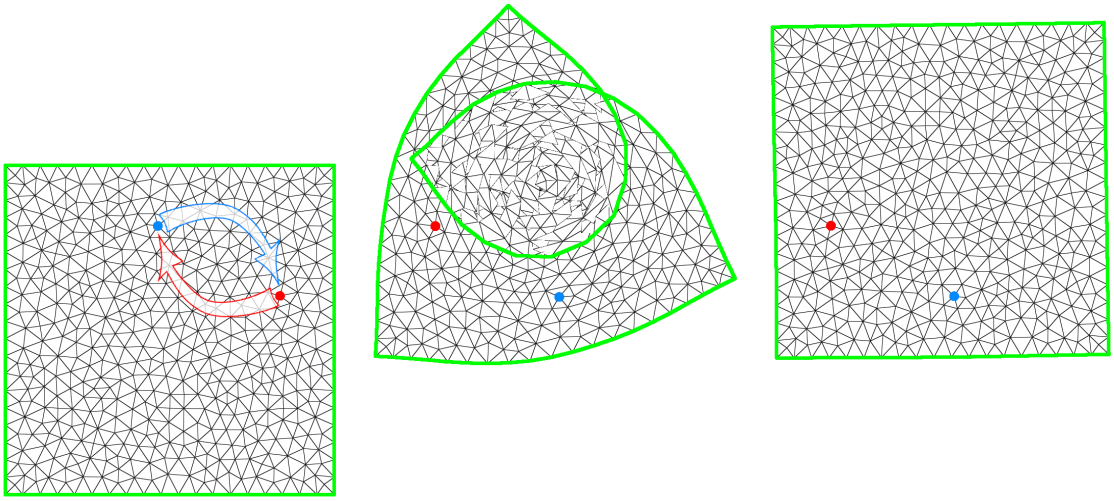


FIGURE 3.38 – (À gauche) Le triangle fantôme admissible dans le domaine de calcul peut forcer la création de triangles de mauvaise qualité, (à droite) solution pratique *i.e.* diviser le voisinage sous contrainte en secteurs et créer des triangles fantômes indépendamment à l'intérieur de chaque secteur en interdisant la création d'arêtes de séparation.



(a) **(b)** **(c)**

FIGURE 3.39 – Exemple standard des déformations à bord libre : échange de deux sommets (représentés en rouge et bleu) dans le maillage, puis trouver la déformation. Seuls deux sommets sont verrouillés, le reste du maillage est libre de bouger. **(a) :** maillage d'entrée, **(b) :** déformation admissible, mais non inversible, **(c) :** déformation inversible.

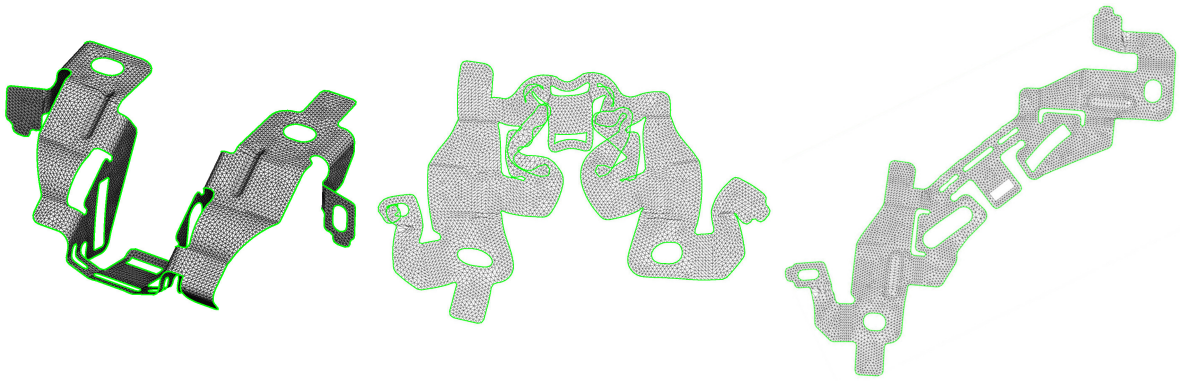
Notez que la principale source de problèmes est la création d'arêtes de séparation, c'est-à-dire les arêtes avec deux sommets fixes, qui divisent en fait notre domaine de calcul en créant des coupes avec des conditions de bords prescrites. La FIGURE 3.38–à gauche montre que même dans le cas où le triangle fantôme n'est pas forcé d'être mal orienté, son arête de séparation peut couper une partie du domaine de calcul et rendre le problème de démêlage trop rigide.

Nous suggérons l'ajout d'une règle simple dans l'algorithme, en présence de sommets contraints. Les sommets contraints indiqués en bleu foncé divisent essentiellement le voisinage 2d des sommets en secteurs comme le montre la FIGURE 3.38–à droite. Nous traitons donc chaque secteur indépendamment, en appliquant l'algorithme jusqu'à ce que deux triangles couvrent le secteur complet. Cet algorithme ne crée pas d'arêtes de séparation et évite le phénomène de verrouillage montré ci-dessus. Nous ne pouvons pas garantir que nos algorithmes créent toujours un bon ensemble admissible, mais on peut facilement construire un ensemble complet de triangles fantômes pour toute maille admissible existante.

Résultats

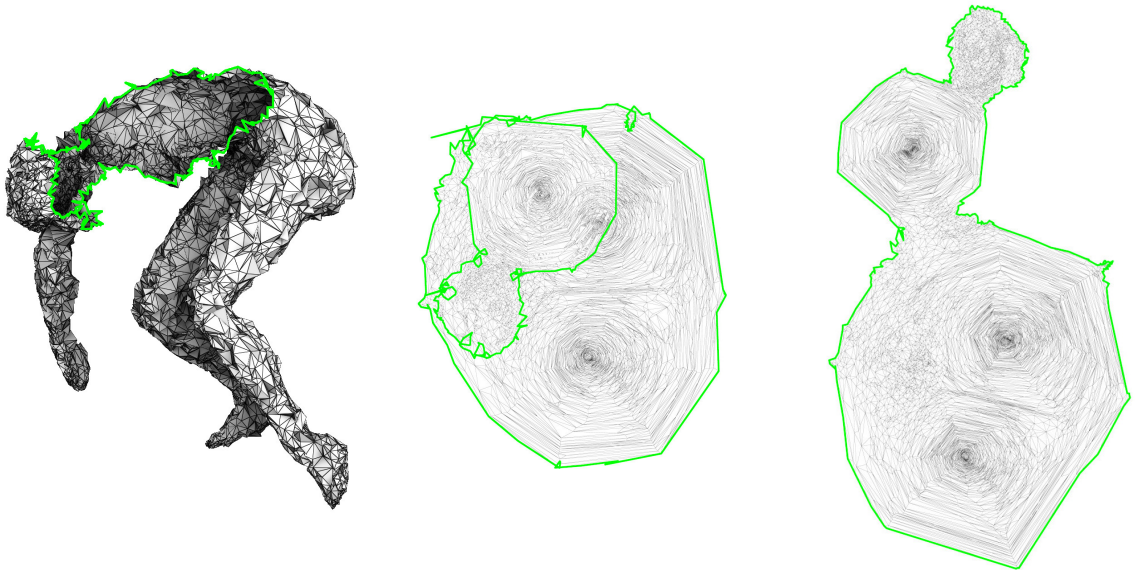
Nous avons testé nos algorithmes dans différentes situations, en lien avec les résultats des sections précédentes :

Test basique deux sommets arbitraires d'un maillage sont échangés, et l'on essaie d'obtenir une carte le plus proche du maillage original. Mais contrairement au test effectué FIGURE 3.9, seuls deux sommets sont verrouillés, le reste du maillage est libre de se déplacer. La FIGURE 3.39–(a) fournit la forme de repos, les sommets à échanger sont surlignés en rouge et bleu, respectivement. La FIGURE 3.39–(b) montre que le démêlage peut aboutir à une carte avec des superpositions. Utiliser nos algorithmes d'ajout de simplexes permet de produire le résultat correct, à savoir une transformation rigide du maillage d'entrée (voir FIGURE 3.39–(c)).



(a) **(b)** **(c)**

FIGURE 3.40 – Formage inverse de la tôle. **(a)** modèle d'entrée, **(b)** le maillage non protégé aplati contient des 2-superpositions, **(c)** le maillage protégé définit un aplatissement localement inversible.



(a) **(b)** **(c)**

FIGURE 3.41 – Aplanissement d'une surface très rugueuse. **(a)** Surface d'entrée à aplanir, **(b)** le maillage non protégé aplani contient des superpositions, **(c)** le maillage protégé définit un aplanissement localement inversible.

Aplatissement de surfaces développables L'idée dans la FIGURE 3.40 est de défaire une pièce mécanique venant d'une feuille de métal, c'est-à-dire d'aplatir un modèle de pièce mécanique avec une courbure gaussienne presque nulle. Dans ce test, tous les sommets sont libres de se déplacer, nous effectuons une projection orthogonale du modèle sur le plan Oxy, puis nous démêlons le maillage, en cherchant une carte conforme à l'aide des algorithmes précédemment développés. Si le démêlage est exécuté sans ajouter de triangles fantômes, le maillage se retrouve pris dans des pièges de superpositions (voir FIGURE 3.40–(b)). Enfin, la FIGURE 3.40–(c) montre le résultat du démêlage après ajout de simplexes. Il est facile de voir que déformation inverse de la tôle a réussi.

Aplatissement de surfaces complexes Nous avons ensuite effectué le même test, mais sur un modèle beaucoup plus difficile. La FIGURE 3.41(a) montre le maillage d'entrée à aplanir, le maillage est fortement courbé et présente des éléments de très mauvaise qualité. Les résultats du calcul de carte avec et sans ajouts d'éléments sont donnés FIGURE 3.41(a–b).

Déformations 3D Enfin, il est important de voir que ce problème arrive aussi en 3D. La FIGURE 3.42 est un test difficile de déformation à bord libre, similaire au test du cube tourné à l'intérieur du cube considéré dans la section 3.2. Prenons un maillage tétraédrique d'un cuboïde avec deux cavités cubiques à l'intérieur. Nous avons ensuite fait pivoter le bord des cavités de 135 degrés autour de l'axe vertical, produisant ainsi des tétraèdres inversés. Nous contraignons le bord des cavités et laissons le reste des sommets libre de bouger. Comme le montre la FIGURE 3.42, sans ajout d'éléments, on peut tomber dans des superpositions en 3D également, alors qu'un ajout de simplexes permet d'obtenir des déformations localement inversibles.

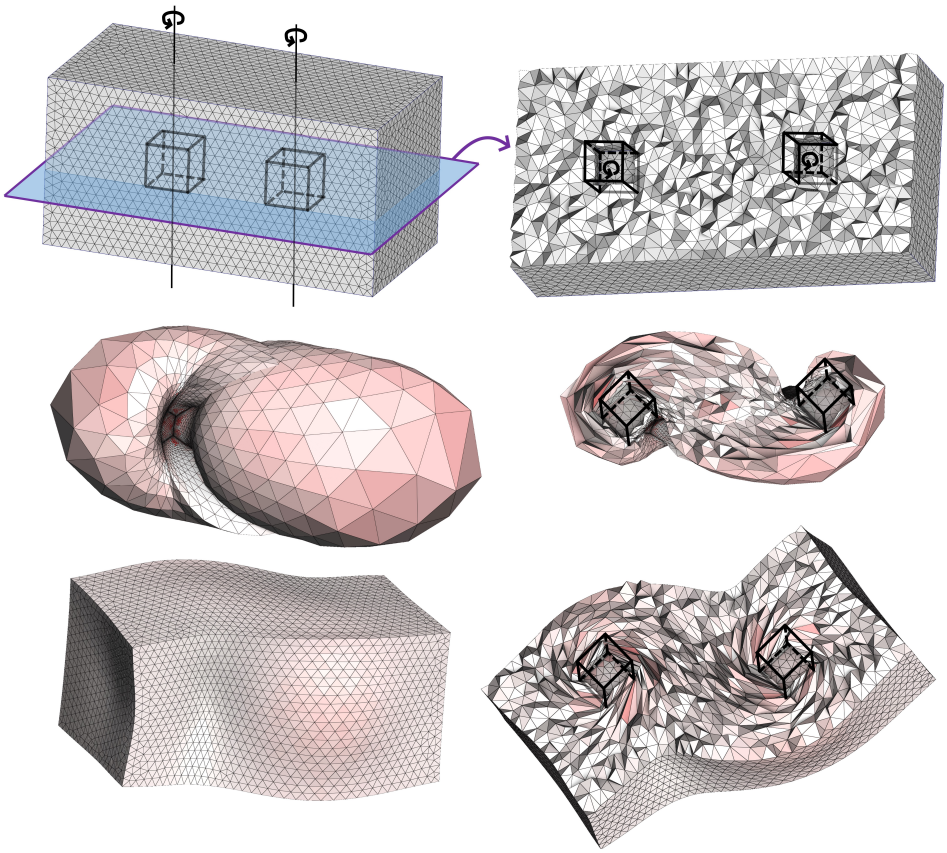


FIGURE 3.42 – Ligne du haut : deux cavités cubiques à l'intérieur d'un cuboïde sont tournées de 135 degrés. Le bord extérieur est libre. Ligne du milieu : maillage original déformé avec superpositions. Ligne du bas : maille avec ajout de simplexes déformé, globalement inversible.

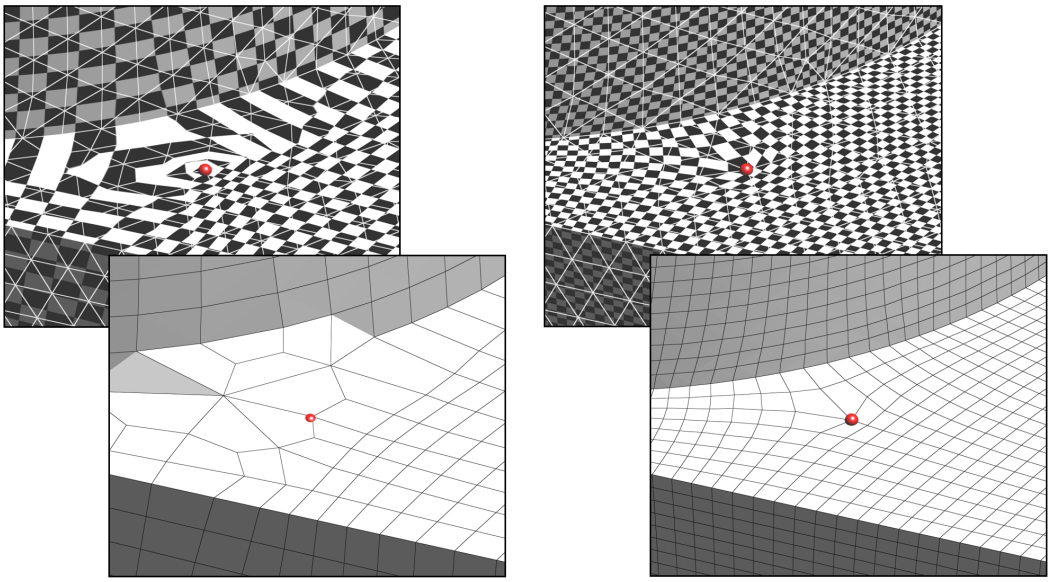


FIGURE 3.43 – La paramétrisation globale classique (QuadCover) intègre un champ de croix (en haut à gauche) : le maillage quadrangulaire (en bas à gauche) a besoin d'algorithme de récupération de maillages en cas de perte d'injectivité locale. Le lisseur variationnel (à droite) n'a pas ce problème.

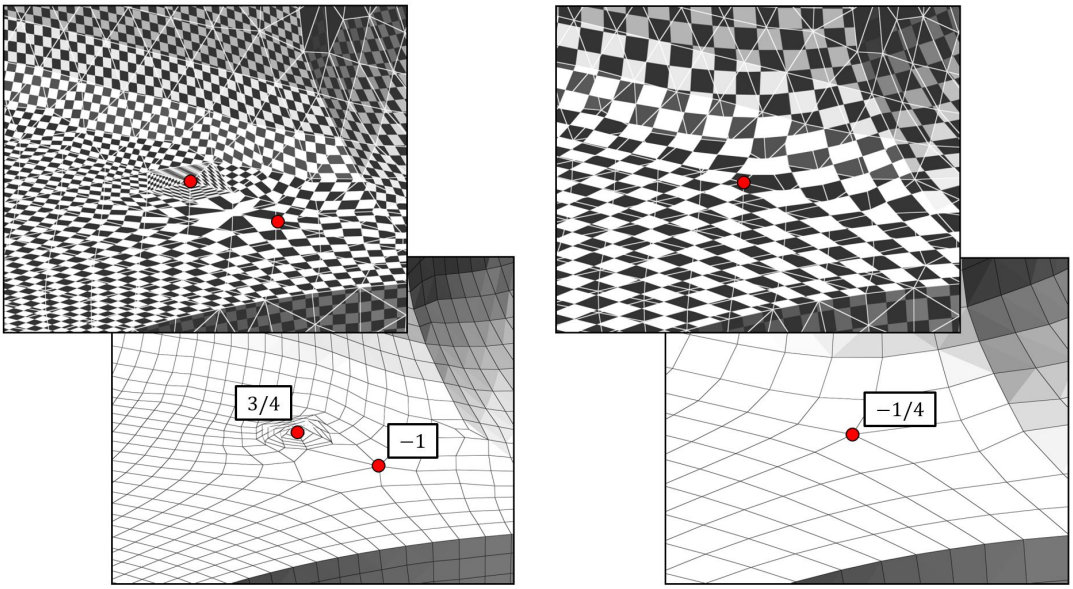


FIGURE 3.44 – Le lisseur variationnel peut générer des superpositions (en haut à gauche). Il produit un sommet singulier supplémentaire dans le maillage quadrangulaire (en bas à gauche). Nous pouvons résoudre ce problème en utilisant notre ajout de triangles fantômes.

Paramétrisations globales

Enfin, comme illustré dans la FIGURE 3.43 et la FIGURE 3.44, notre algorithme d'ajout s'applique pour une application particulière : la génération de maillages quadrangulaire. L'idée est de définir une déformation de la surface d'entrée telle que si le maillage quadrangulaire final (le résultat) subit cette déformation, il correspond à une grille unitaire, alignée sur les axes. L'application directe de cette idée est de calculer cette déformation, appliquer son inverse à la grille unitaire, et obtient un maillage quadrangulaire. En pratique, il est préférable d'introduire plus de degrés de liberté en considérant des paramétrisations globales au lieu d'une déformation, nous avons détaillé cela dans le chapitre 2. Dans ce cas, les paramétrisations présentent des discontinuités qui permettent de représenter une famille beaucoup plus grande de maillages quadrangulaires : la grille déformée peut être coupée et collée à elle-même d'une manière non triviale.

L'exécution de QuadCover [Kälberer *et al.*, 2007] pour ce faire entraîne souvent une perte locale d'injectivité et, comme l'illustre la FIGURE 3.43–gauche, on peut avoir besoin d'appeler une solution de secours telle que QEx [Ebke *et al.*, 2013] pour extraire un maillage malgré les défauts de la paramétrisation. L'introduction d'un lissage variationnel permet d'éviter ce problème, en produisant des paramétrisations valides (voir FIGURE 3.43–droite). Nous devons cependant "protéger" le maillage pour éviter les k-superpositions, comme l'illustre la FIGURE 3.44, car profitant des propriétés du QuadCover, des singularités $-1/4$ peuvent se scinder en un couple de singularités -1 et $3/4$. Nous utilisons notre algorithme d'ajout de triangle pour empêcher l'apparition de telles configurations.

3.4.2 Lissage d'hexaèdre

Le lissage d'hexaèdre est un sujet complexe qui à donné lieu à une importante littérature [Knupp, 2001, Knupp, 2003, Shepherd *et al.*, 2006, Wilson *et al.*, 2012, Ruiz-Gironés *et al.*, 2014, Livesu *et al.*, 2015, Wang *et al.*, 2018, Wang *et al.*, 2021, Huang *et al.*, 2022]. Cela est dû à son importance : il est rare qu'une méthode donne immédiatement un maillage géométriquement bon, et il est donc classique d'améliorer le maillage *a posteriori*. Cela est fait à l'aide d'une procédure de lissage du maillage, qui se trouve être très complexe, pour deux raisons :

— les hexaèdres (tri-linéaires) sont des éléments bien plus complexes que les tétraèdres (linéaires) ;
— les géométries auxquelles doivent correspondre les maillages peuvent être très compliqués.

Dans cette sous-section, nous proposons une approche de lissage de maillage reposant sur le calcul de carte avec les énergies définies dans les sections précédentes.

Nous commençons par rappeler les propriétés des hexaèdres, ainsi que les qualités recherchées. Nous montrons ensuite comment il est possible d'optimiser un maillage tout en respectant des contraintes de bord complexes sur des modèles de CAO.

Hexaèdres : forme tri-linéaire

Mathématiquement, un hexaèdre est communément représenté comme une déformation tri-linéaire d'un cube unité, comme illustré sur la FIGURE 3.45. L'intérieur d'un élément est représenté comme une combinaison linéaire de ses sommets. Avec cette définition, on peut procéder à un raisonnement similaire à celui utilisé avec les tétraèdres : un élément est dit valide lorsque son déterminant est positif, et donc un hexaèdre sera valide si le déterminant de la forme tri-linéaire est positif partout. C'est ici que le problème devient difficile, comment savoir si le déterminant est positif ? Pour un tétraèdre, cela est immédiat, puisqu'il suffit de calculer le déterminant d'une

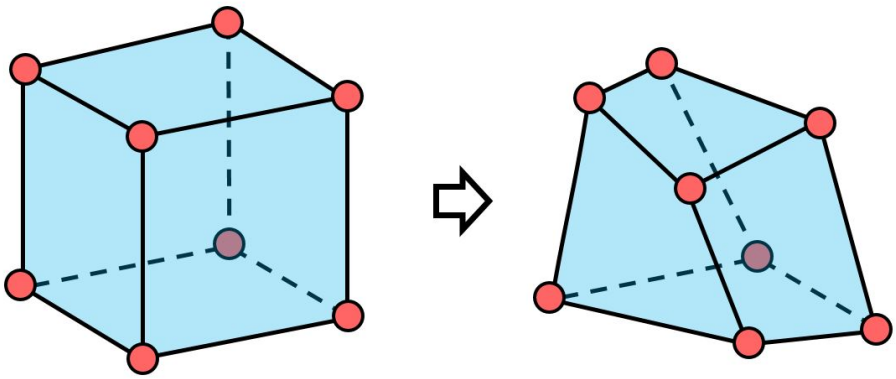


FIGURE 3.45 – Un hexaèdre est communément représenté comme la déformation tri-linéaire d'un cube unité.

matrice, la matrice gradient de la forme linéaire définissant ce tétraèdre. Mais pour une forme tri-linéaire cela est nettement plus difficile.

La métrique la plus utilisée pour parler de la qualité des éléments est le *scaled jacobian* (SJ) qui étudie les déterminants des tétraèdres formant les coins de l'hexaèdre. [Marschner *et al.*, 2020] illustre savamment les limites de cette métrique : tourner le quadrilatère supérieur d'un cube parallèlement à son quadrilatère inférieur ne produit jamais un SJ négatif, alors que l'élément devient vraisemblablement de mauvaise qualité. [Johnen *et al.*, 2017] montre une procédure complexe utilisant un algorithme de *branch and bound* pour calculer la qualité minimum dans un hexaèdre.

Lissage

Étudier la qualité des hexaèdres n'est donc pas chose aisée. Cela rend difficile la construction d'un algorithme d'amélioration d'hexaèdres ayant des garanties concernant cette qualité. Bienheureusement, [Branets et Garanzha, 2002] ont apporté une solution élégante à ce problème. Les auteurs montrent que, en utilisant l'énergie elliptique détaillée dans ce chapitre, le déterminant de la carte tri-linéaire peut être minoré en utilisant 64 points de quadrature, détaillés sur la FIGURE 3.47, qui peuvent être immédiatement utilisés pour produire une méthode d'optimisation.

D'une façon similaire à ce qui est proposé par [Branets et Garanzha, 2002], nous utilisons l'énergie (3.8) pour obtenir un maillage valide, nous la réécrivons de la façon suivante :

$$E_\lambda(J, \varepsilon) = f_\varepsilon(J) + \lambda g_\varepsilon(J)$$

et nous avons comme énergie de qualité pour nos éléments

$$F_\lambda(U, \varepsilon) = \sum_{h=1}^{\#H} \sum_{q=1}^{64} w_q E_\lambda(J_q, \varepsilon)$$

où H représente les hexaèdres du maillage, et J_q la q-ème quadrature définie sur la FIGURE 3.47 avec w_q le poids associé, avec $\sum w_q = 1$. [Branets et Garanzha, 2002] propose des poids différents pour chaque "catégorie" de quadrature (numéro de la FIGURE 3.47), $w_I = \frac{1}{27}, w_{II} = \frac{1}{2\times27}, w_{III} = \frac{1}{4\times27}$ et $w_{IV} = \frac{1}{8\times27}$.

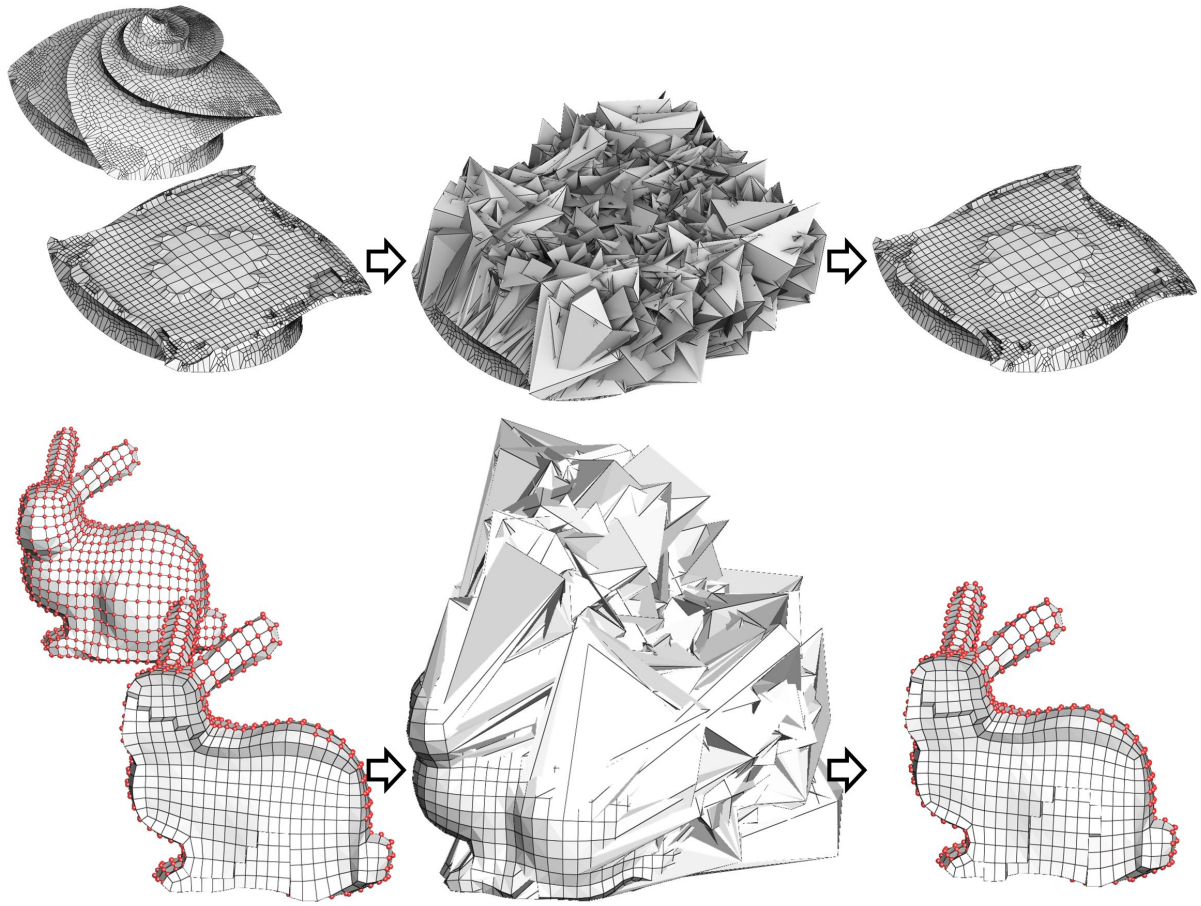


FIGURE 3.46 – Lissage de maillages hexaédriques à bords contraints (et valides). On prend un maillage hexaédrique auquel on verrouille le bord, puis on place tous les points intérieurs à un emplacement aléatoire. Utilisant l'énergie (3.34), on récupère un maillage de bonne qualité, sans connaissance de l'original.

Nous associons à cela un terme d'attache au bord $D(U)$ qui est une distance au carré aux *features* correspondantes pour chaque point du bord. Notre problème devient donc :

$$\arg\min_{U} F_\lambda(U,\varepsilon) + \mu D(U) \tag{3.34}$$

Que nous résolvons avec l'algorithme 1 précédemment proposé en introduisant la distance $D(U)$ dans le gradient L-BFGS. En pratique nous utilisons $\mu = 10^5$ et un $\lambda = 10^{-5}$ très faible.

L'utilisation de 64 quadratures permet de garantir la qualité des éléments finaux, mais donne une implémentation particulièrement lente. Il est en revanche possible d'étudier moins de points de quadrature, par exemple 32, 8 ou même 4 (minimum couvrant les 12 arêtes) pour améliorer les performances, au prix de moins de garanties. La FIGURE 3.46 donne des exemples d'optimisation de maillages dans des situations particulièrement complexes.

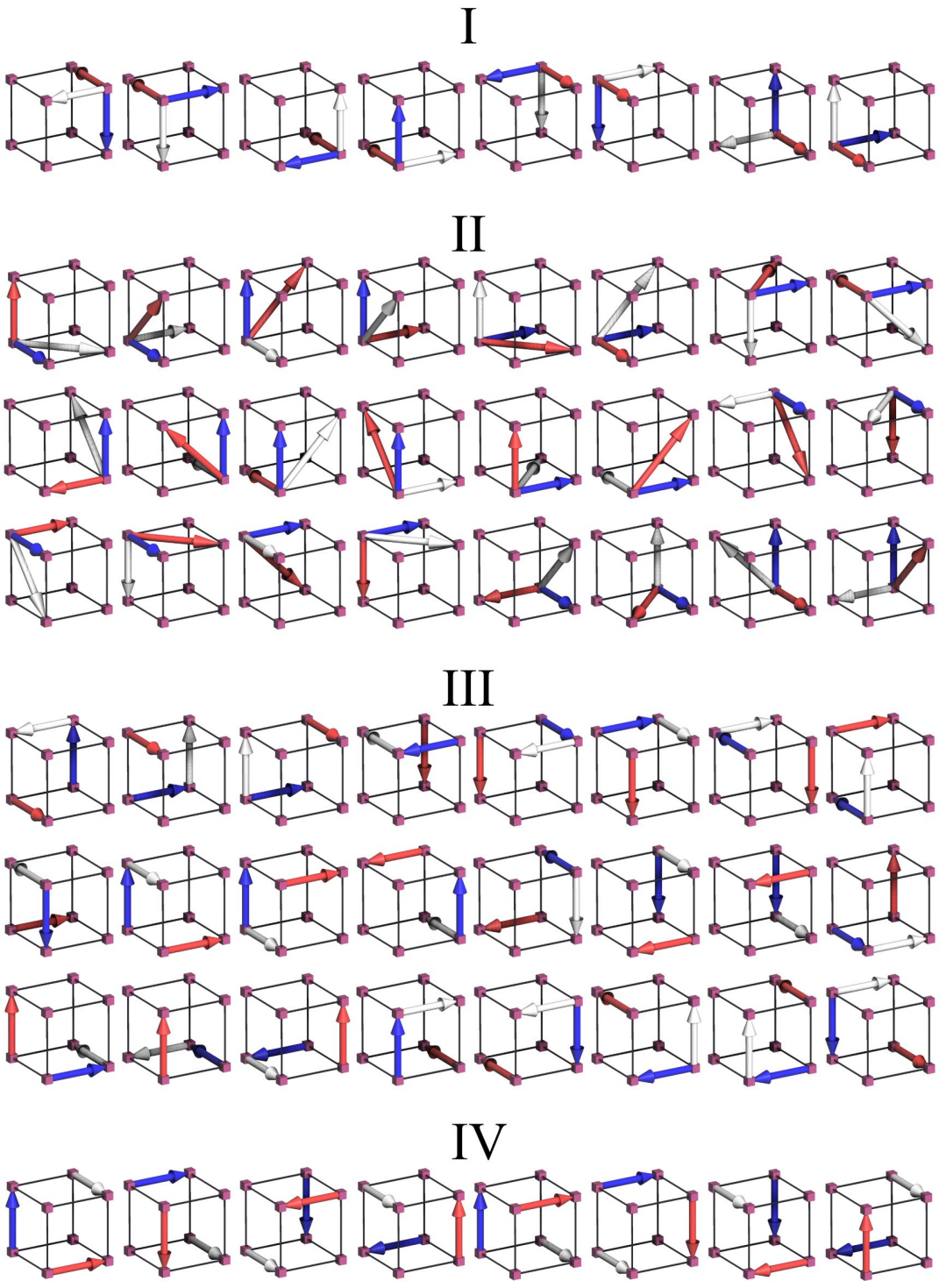


FIGURE 3.47 – Schéma de quadratures d'un hexaèdre. Les flèches blanche, rouge et bleu (dans l'ordre) forment une base qui permet de calculer une matrice Jacobienne. Ces schémas sont obtenus en étudiant toutes les permutations de triplés d'arêtes contenant une arête de chaque dimension, voir [Branets et Garanzha, 2002].

Chapitre 4

Extraction robuste d'hexaèdres depuis un polycuboid [Protais *et al.*, 2022]

Dans ce chapitre, nous étudions l'étape d'extraction d'hexaèdres à partir d'un *polycuboid*, avec pour objectif d'obtenir des maillages hexaédriques aussi grossiers que possible, et cela de façon robuste. Le contenu de ce chapitre a donné lieu à une présentation dans la conférence internationale SPM avec une publication des actes dans le journal CAD :
Protais, F., Reberol, M., Ray, N., Corman, E., Ledoux, F. et Sokolov, D. (2022). Robust quantization for polycube maps. *Computer-Aided Design (SPM)*, page 103321

Ce chapitre est construit de la façon suivante : nous commençons par un rappel sur les étapes des méthodes de polycubes introduisant le problème de quantification. Nous complétons ce rappel par un état de l'art (section 4.1), nous développons ensuite un formalisme (section 4.2) qui nous permet d'introduire notre nouvelle approche de quantification (section 4.3). Nous détaillons enfin l'implémentation de notre méthode (section 4.4) et nous présentons un ensemble de résultats et de comparaisons (section 4.5).

Procédons tout d'abord à un rappel des étapes des méthodes de polycubes : à partir d'un volume Ω, les méthodes polycubes comportent généralement trois parties (voir FIGURE 4.1) :

1. **Déformation colorée** — Le bord $\partial\Omega$ du volume Ω est segmenté en *charts* auxquels sont attribués l'une des six couleurs possibles $\{\vec{u}, -\vec{u}, \vec{v}, -\vec{v}, \vec{w}, -\vec{w}\}$. Le volume est ensuite déformé de manière continue afin que chaque *chart* devienne plan et orthogonal à la direction canonique désignée par la coloration. Le résultat de cette première étape sera appelé un *polycuboid*. Les définitions formelles de la coloration et de la carte (vers un *polycuboid*) sont données dans §4.2.1.
2. **Quantification** — Cette deuxième étape permet de s'assurer que les faces du polycuboid soient alignées avec la grille entière pour obtenir un *Polycube*. Dès lors, le volume peut être rempli par une grille unitaire.
3. **Inversion** — La déformation du polycube est inversée pour extraire le maillage hexaédrique final.

Ces trois étapes présentent des problèmes de robustesse. Dans ce chapitre, nous abordons les problèmes de la quantification et de l'inversion. La première étape (calcul d'une coloration menant à un polycuboid valide) est très difficile [Sokolov et Ray, 2015,Zhao *et al.*, 2019], et aucun algorithme existant n'offre la garantie de produire une carte à jacobien positif. La résolution de ce problème est un sujet de recherche actif, qui est orthogonal aux questions de robustesse traitées dans ce chapitre. Nous supposerons donc ici avoir en entrée un polycuboid valide ; nous précisons

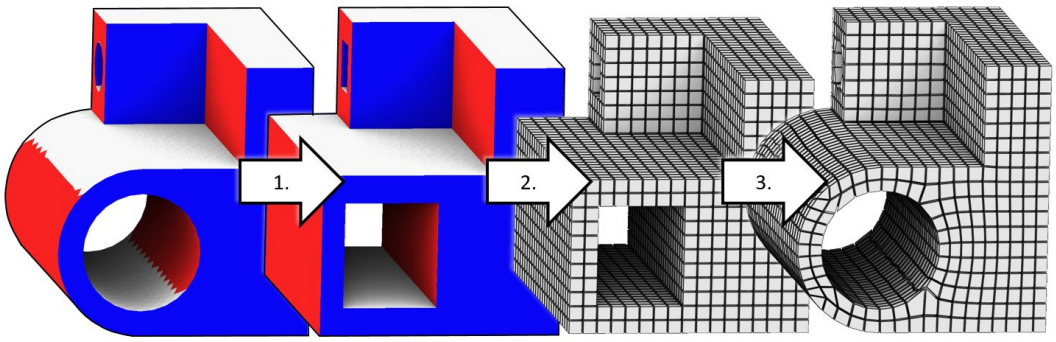


FIGURE 4.1 – Pipeline de génération de maillage hexaédrique avec la méthode des polycubes.

cette définition dans la section 4.2 et poursuivons la discussion autour de ce sujet de recherche actif dans le prochain chapitre.

Ainsi, étant donné une carte continue $\mathfrak{g}$ de l'objet Ω vers l'espace paramétrique avec un jacobien positif :

— Nous quantifions la carte, c'est-à-dire que nous trouvons un ensemble valide de contraintes de bord entières.
— Nous calculons la structure combinatoire du maillage hexaédrique induite par ces contraintes de bord.
— Nous récupérons la géométrie du maillage hexaédrique.

L'approche habituelle pour quantifier la carte (étape 2.) consiste à projeter les faces du polycuboid sur le plan de coordonnées entières le plus proche. Cette méthode est risquée, car plusieurs faces peuvent être projetées au même endroit, ce qui entraîne des modifications topologiques du domaine. Cela rend alors impossible l'inversion, la déformation et l'extraction d'un maillage hexaédrique valide. Cette technique gomme généralement les petits détails géométriques du modèle, mais lorsqu'on recherche des polycubes grossiers, toute la géométrie peut être considérée comme un détail, ce qui peut avoir l'effet dramatique de générer des polycubes dégénérés en un point. *Nous proposons une méthode fournissant des garanties théoriques et pratiques empêchant la perte de détail (voir* FIGURE *4.2).* Étant donné un polycuboid valide (étape (1)), nous produisons un maillage hexaédrique valide, quelle que soit la résolution souhaitée.

Pour inverser la carte polycube (étape (3)), l'approche habituelle [Gregson *et al.*, 2011, Livesu *et al.*, 2013] consiste à calculer la carte comme une carte linéaire par morceaux sur un maillage tétraédrique de Ω. Cela est suffisant pour produire les résultats des travaux précédents, mais nous sommes confrontés à une situation plus difficile : les maillages extrêmement grossiers impliquent une distorsion beaucoup plus élevée (voir FIGURE 4.3). Pour gérer de manière robuste cette situation extrême, nous extrayons la structure combinatoire du maillage hexaédrique résultant de la carte quantifiée, et nous optimisons sa géométrie. Cela supprime l'étape de calcul d'une paramétrisation globale susceptible d'avoir des dégénérescences/éléments inversés que même l'algorithme robuste HexEx [Lyon *et al.*, 2016] ne serait pas en mesure de récupérer.

Contributions

En résumé, nous proposons une approche qui, compte tenu d'une déformation colorée $\mathfrak{g}$ à jacobien positif (étape (1)), produit un maillage hexaédrique valide, même pour les **maillages très grossiers**. Nous contribuons à trois aspects du maillage hexaédrique basé sur les polycubes :

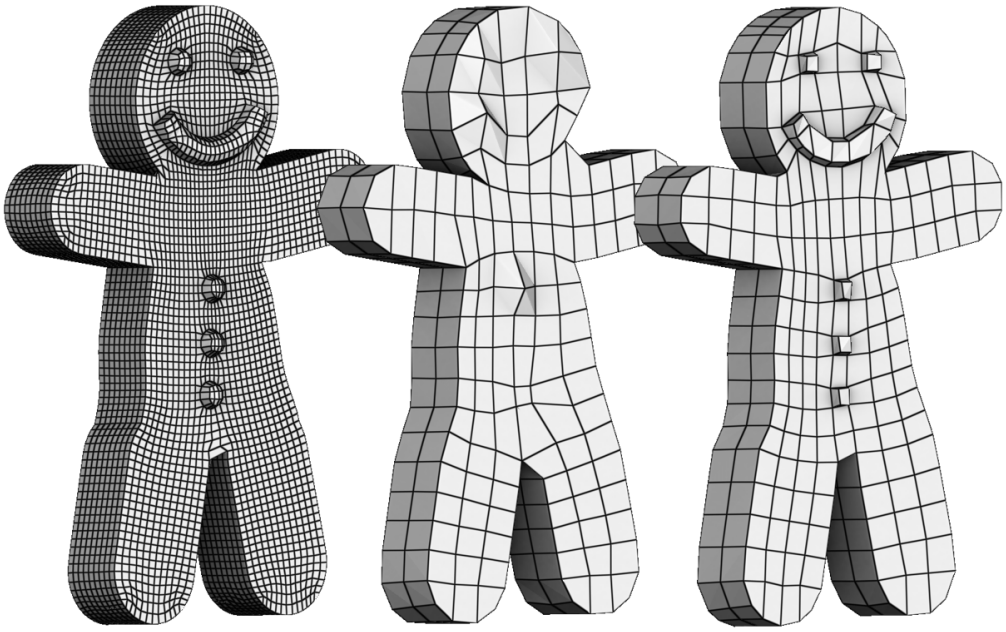


FIGURE 4.2 – Le simple fait d'arrondir les variables à l'entier le plus proche ne pose pas de problème pour produire un maillage haute résolution (à gauche), mais devient critique pour un maillage plus grossier (au milieu). Nos contraintes empêchent de telles pertes de détail (à droite).

— *Nous garantissons de bien quantifier la déformation colorée :* en d'autres termes, nous construisons une paramétrisation à jacobien positif avec des bords entiers.
— *Nous générons des maillages hexaédriques de manière robuste :* la positivité du jacobien pour la déformation colorée en entrée $\mathfrak{g}$ n'est pas nécessaire.
— *Nous générons des maillages hexaédriques de manière robuste :* notre génération de maillage ne repose pas sur le calcul de carte globale à jacobien positif pour l'étape d'inversion.

4.1 Travaux antérieurs

Les polycubes ont été introduits en infographie pour le *seamless texturing* de surfaces triangulées [Tarini *et al.*, 2004]. Le premier algorithme automatique de génération de polycubes [He *et al.*, 2009] calcule un champ scalaire sur l'objet qui contraint la coordonnée z de l'objet déformé. Les autres dimensions sont résolues par une rastérisation 2D. Les méthodes récentes [Gregson *et al.*, 2011, Huang *et al.*, 2014, Fu *et al.*, 2016, Zhao *et al.*, 2018] préfèrent optimiser toutes les coordonnées de la déformation simultanément. L'idée est de déformer progressivement le modèle de manière à ce que chaque triangle du bord de l'objet déformé ait une de ses coordonnées constante et entière. À chaque itération, la normale du triangle devient de plus en plus alignée avec l'un des axes du système de coordonnées. En principe, chaque triangle doit être aligné avec l'axe le plus proche de sa normale, mais des stratégies plus élaborées [Livesu *et al.*, 2013] peuvent améliorer les résultats. Le maillage hexaédrique final est généralement amélioré par une étape dite de *pillowing* [Gregson *et al.*, 2011], éventuellement avec une optimisation globale [Cherchi *et al.*, 2019].

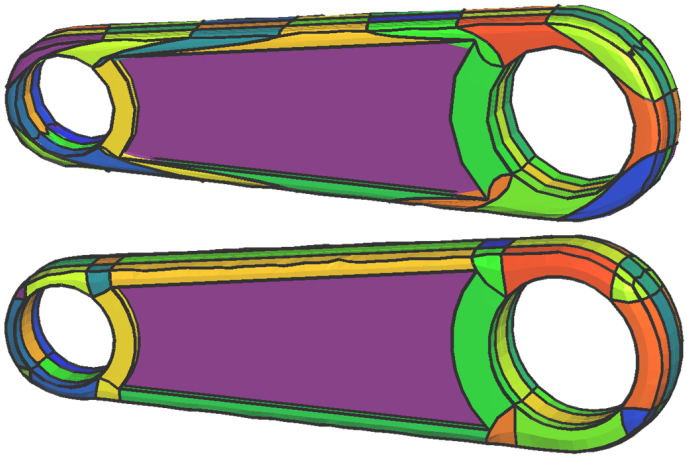


FIGURE 4.3 – Une approche directe pour calculer la carte de l'intérieur consiste à fixer d'abord le bord (initialisation dans [Livesu *et al.*, 2013]). Elle produit des hexaèdres fortement distordus lors de la génération de maillages très grossiers (image supérieure). De plus, l'extraction du maillage hexaédrique peut échouer en présence d'éléments inversés. Nos cartes ont une bien meilleure géométrie (image du bas), et nous proposons une extraction robuste, car elle ne repose pas sur le calcul d'une paramétrisation globale valide.

Ces algorithmes présentent deux sources d'échecs majeures. Tout d'abord, il peut ne pas exister de déformation valide alignant tous les triangles par rapport à une coloration donnée. De nombreux cas d'échecs peuvent être corrigés par des heuristiques [Gregson *et al.*, 2011, Huang *et al.*, 2014, Fu *et al.*, 2016], mais leurs résolutions dans le cas général est très difficile [Sokolov et Ray, 2015]. Deuxièmement, le respect de la coloration n'est que la moitié de l'histoire : les coordonnées entières du bord doivent être choisies de manière à nous permettre de produire un maillage hexaédrique valide. Dans ce chapitre, nous nous concentrons sur ce second point.

Plus un maillage hexaédrique est fin, plus il est facile à produire. une façon naturelle d'aborder le problème de la décomposition des blocs est donc de générer un maillage hexaédrique fin, et de le décimer avec un algorithme général de simplification de maillage hexaédrique [Gao *et al.*, 2015, Gao *et al.*, 2017b]. A chaque étape, l'algorithme applique une opération de simplification (suppression de feuillets ou de cordes) au maillage hexaédrique. La géométrie du nouveau maillage est obtenue par une reparamétrisation. Les résultats sont similaires aux nôtres, mais il faut commencer par un maillage hexaédrique fin, et l'optimisation est gloutonne, pouvant ne pas conduire à la meilleure simplification, en plus de nécessiter des étapes de reparamétrisation très coûteuses.

Il est également possible d'obtenir une structure grossière avant la phase de génération du maillage, en travaillant directement sur les coordonnées des cartes représentant le polycube. Cherchi *et al.* [Cherchi *et al.*, 2016] alignent localement une paire de coins de polycube selon une coordonnée. Zhao *et al.* [Zhao *et al.*, 2019] optimisent les longueurs des arêtes du polycube et Chen *et al.* [Chen *et al.*, 2019] optimisent la position des *charts*, mais ces deux travaux ne considèrent que le bord du polycube, ce qui est insuffisant pour éviter la perte de bijectivité. Guo *et al.* [Guo *et al.*, 2020] étendent l'approche de [Chen *et al.*, 2019], en considérant des *charts* opposés les uns aux autres, mais leurs contraintes ne sont toujours pas exhaustives. Notre algorithme a des attentes plus faibles sur l'entrée, fournit une solution optimale par rapport à nos contraintes (au lieu d'une stratégie gloutonne), et est garanti de fournir une sortie valide.

L'approche des polycubes peut être étendue en introduisant des coupures dans la déformation pour supprimer certaines contraintes inutiles, comme le proposent Fang *et al.* [Fang *et al.*, 2016] et Guo *et al.* [Guo *et al.*, 2020]. Une idée similaire est développée par Li *et al.* [Li *et al.*, 2013] où une définition généralisée des Polycubes est employée. En ce qui nous concerne, nous nous limitons à une définition standard des Polycubes.

Il est intéressant de noter que si nous considérons toutes les coupes possibles, nous retombons dans la famille des algorithmes de remaillage hexaédrique basés sur des paramétrisations globales [Nieser *et al.*, 2011]. Les paramétrisations globales présentent des problèmes de robustesse similaires à ceux du remaillage polycube : les problèmes de champs de *frames* remplacent les problèmes de coloration, et l'exigence de la bijectivité de la déformation est remplacée par la positivité du jacobien de la paramétrisation globale. La conception des champs de *frames* est actuellement le goulot d'étranglement du processus et constitue un domaine de recherche actif [Li *et al.*, 2012, Liu *et al.*, 2018] et comme dans le cas du polycube, la bijectivité devient critique lorsqu'un maillage grossier est souhaité.

Il faut noter que pour le cas 2D (i.e. la génération de maillages quadrangulaires) [Ray *et al.*, 2006, Kaelberer *et al.*, 2007], il existe des algorithmes robustes basés sur des paramétrisations globales [Myles *et al.*, 2014, Campen *et al.*, 2015]. L'ingrédient principal de ces algorithmes est la décomposition du domaine en patchs en forme de quadrilatères en coupant le domaine le long d'un graphe de *motorcycles* (par exemple, en traçant les *streamlines* du champ de *frames*). Grâce à cette décomposition, on peut générer un maillage quadrangulaire respectant la structure de ces patchs. Un travail très récent de Lyon *et al.* [Lyon *et al.*, 2021] propose d'étudier un graphe de *motorcycles* où les *streamlines* ne s'arrêtent pas à leur première rencontre et continuent un peu. Cela génère une décomposition en quadrilatères plus complète qui permet de mieux contrôler la distorsion introduite par la quantification.

Notre travail peut être compris comme une extension de ces approches à la 3D. Kowalki *et al.* [Kowalski *et al.*, 2014, Kowalski *et al.*, 2016] ont observé que le traçage 3D dans un espace paramétrique, qui implique des iso-surfaces, est un problème mal posé. Dans cet article, nous limitons notre attention aux polycubes. Dans ce cadre, nous observons que le graphe de *motorcycle*s 3D devient un simple ensemble de plans alignés sur un axe dans le domaine paramétrique. De plus, nous pouvons tracer les iso-surfaces jusqu'à la fin, c'est à dire jusqu'à rencontrer un bord, ce qui n'est pas possible dans une paramétrisation globale. La décomposition en blocs a la structure combinatoire d'un polycube (pas de jonctions en T) et les longueurs des bords sont manipulées par les coordonnées de leurs extrémités, qui sont directement les variables du problème de paramétrisation. En échange d'un ensemble restreint de maillages possibles, nous avons un contrôle total sur la distorsion induite par la quantification. Un compromis entre les deux est laissé pour un travail futur.

4.2 Formalisme adapté aux polycubes

N.B. Nous présenterons d'abord le problème en 2D, puis nous détaillerons notre implémentation et nos résultats en 3D. Notons toutefois que cela ne signifie pas que le problème de quantification soit le même en 2D et en 3D. Nous sommes en mesure de le faire parce que nous avons développé une nouvelle approche pour manipuler nos polycubes qui passe immédiatement de la 2D à la 3D.

Comme mentionné précédemment, les méthodes de paramétrisation donnent d'excellents résultats pour la génération de maillages quadrangulaires. L'idée est de calculer une paramétrisation f, ou, en d'autres termes, deux champs scalaires f_1 et f_2 sur le domaine Ω (voir la

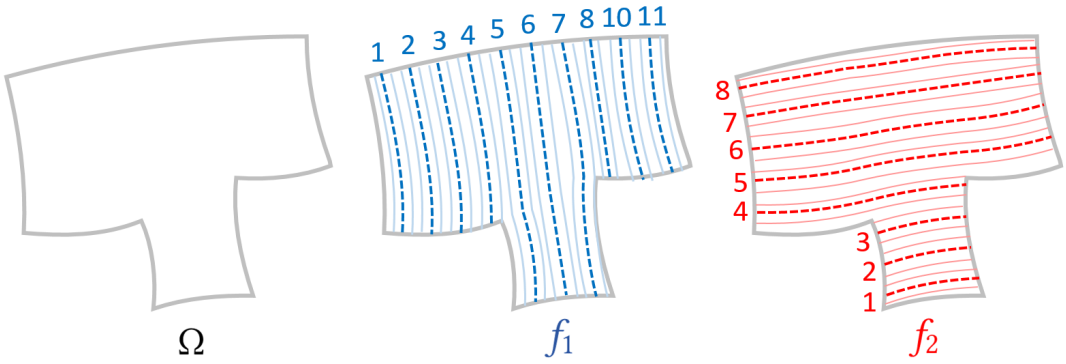


FIGURE 4.4 – Exemple d'une paramétrisation $f = (f_1, f_2)$ d'un domaine Ω, représentée par ses courbes de niveau.

FIGURE 4.4). Les iso-lignes entières de ces champs scalaires forment un maillage quadrangulaire structuré.

Pour qu'un maillage quadrangulaire soit bien défini, ces champs scalaires doivent avoir les propriétés suivantes (voir la FIGURE 4.5) :

1. le Jacobien de f doit être positif.
2. Ils doivent être alignés avec le bord $\partial\Omega$, c'est-à-dire que le bord doit être sur une iso-line de l'un des champs scalaires.
3. De plus, en chaque point du bord $\partial\Omega$, au moins un des champs scalaires doit être *entier*.

Dans ce chapitre, nous proposons de partir d'une paramétrisation qui respecte les propriétés (1) et (2), mais viole la propriété (3) et d'en extraire une nouvelle qui vérifie les trois propriétés. Dans la sous-section 4.2.1, nous introduisons quelques notations nécessaires à la formalisation du problème (sous-section 4.2.2).

4.2.1 Notations

Nous désignons le domaine géométrique à discrétiser par $\Omega \subset \mathbb{R}^2$, et son bord par $\partial\Omega$. Soit $X = (x, y)$ un point de $\mathbb{R}^2$. La matrice jacobienne d'une paramétrisation $f : (x, y) \to (f_1(x, y), f_2(x, y)) \in \mathbb{R}^2$ est désignée par Df, et son déterminant par $\det(Df)$.

Définition 4 (Cartes polycuboid $\mathcal{P}_\mathbb{R}$). *Une carte $f : \Omega \to \mathbb{R}^2$ est dite polycuboid si 1. elle est continue ; 2. elle a un jacobien positif ($\det(Df) > 0$) ; et 3. son bord se trouve sur un ensemble fini d'iso-lignes. L'ensemble des cartes polycuboid est noté $\mathcal{P}_\mathbb{R}$.*

Nous désignons alors $L^r = \{L_1^r, L_2^r\}$ l'ensemble des valeurs réelles des iso-lignes décrivant le bord de $f(\Omega)$ (L_1^r lorsque le bord se trouve sur une iso-ligne dans la première dimension, et L_2^r pour la deuxième dimension).

Définition 5 (Cartes polycube $\mathcal{P}_\mathbb{Z}$). *Une carte $f \in \mathcal{P}_\mathbb{R}$ est dite polycube si $L^r \subset \mathbb{Z}$, i.e. les iso-lignes sont sur des valeurs entières, et L^r s'écrit alors $= L$. L'ensemble des cartes polycube est noté $\mathcal{P}_\mathbb{Z}$.*

Remarque 15. *Dans le reste du chapitre, nous dirons que l'image de Ω par une carte polycuboid est un* polycuboid*, et devient un* polycube *si c'est une carte polycube.*

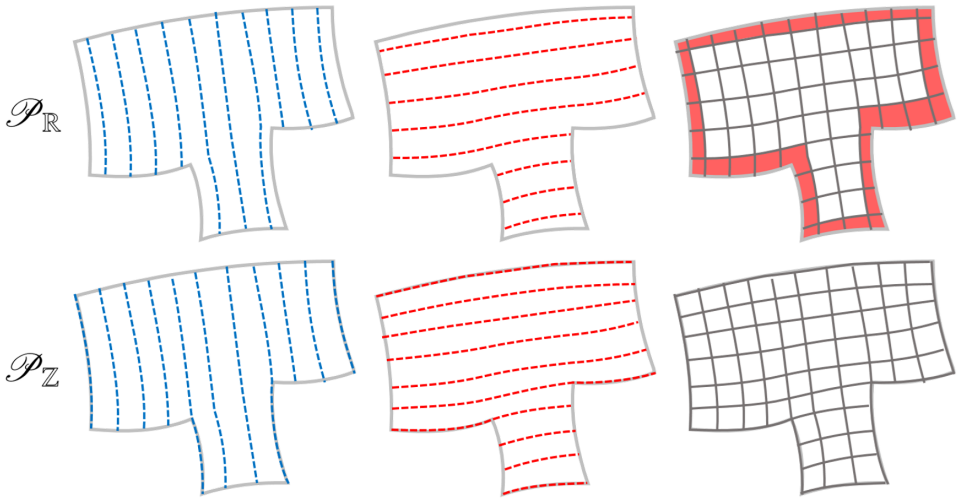


FIGURE 4.5 – **Rangée du haut :** un maillage extrait d'une carte dans $\mathcal{P}_{\mathbb{R}}$ (se référer à §4.2.1) possède des éléments non alignés sur les bords (en rouge). **Rangée du bas :** Pour obtenir des maillages valides, il faut considérer la carte dans $\mathcal{P}_{\mathbb{Z}}$.

Prenons $f \in \mathcal{P}_{\mathbb{R}}$. Le bord $\partial\Omega$ peut être décomposé en un ensemble de *charts* U_i pour la première dimension et V_j pour la seconde dimension. Les cartes sont des iso-lignes de f telles que $f_{1|_{U_i}} \in L_1^r$ et $f_{2|_{V_j}} \in L_2^r$.

Pour simplifier les notations, nous désignons ces valeurs constantes par $f_{1|_{U_i}}$ et $f_{2|_{V_j}}$ (voir FIGURE 4.6). Chaque fois que cela est possible, nous mettrons les expressions pour la 1ère dimension seulement, et nous mentionnons "resp. V_j" pour signifier qu'il est trivial de retrouver la 2ème expression.

4.2.2 Problématique

Étant donnée une carte polycuboid d'entrée $\mathfrak{g} \in \mathcal{P}_{\mathbb{R}}$, le problème de quantification consiste à trouver une carte polycube similaire $\bar{f} \in \mathcal{P}_{\mathbb{Z}}$, solution de :

$$\bar{f} = \underset{f \in \mathcal{P}_{\mathbb{Z}}}{\arg\min} \left\{ E(f) := \int_{\Omega} \|\nabla f_1 - \nabla \mathfrak{g}_1\| + \|\nabla f_2 - \nabla \mathfrak{g}_2\| dX \right\}. \tag{4.1}$$

avec $\|\cdot\|$ la norme euclidienne, et $E(f)$ est une énergie mesurant la distance entre f et $\mathfrak{g}$.

Une approche courante pour résoudre ce problème consiste à trouver de façon gloutonne un ensemble L de nouvelles valeurs pour le bord, de sorte que $l_{U_i} \approx \mathfrak{g}_{1|_{U_i}}$ (resp. V_j). Ensuite, une nouvelle carte contrainte par ces valeurs limites et minimisant l'équation (4.1) est calculée. Le problème, cependant, est que pour certains ensembles $L = \left\{ \{l_{U_i}\}_{i=1}^{|L_1^r|}, \{l_{V_j}\}_{j=1}^{|L_2^r|} \right\}$, il est impossible de définir une carte polycube respectant les trois propriétés clés, c'est-à-dire que l'ensemble $\mathcal{P}_{\mathbb{Z}}$ est vide. La quantification échoue alors. L'exemple trivial est de mettre toutes les contraintes à zéro ($l_{U_i} = 0, \forall i$) ; un cas d'échec non trivial assez commun est montré sur la FIGURE 4.7.

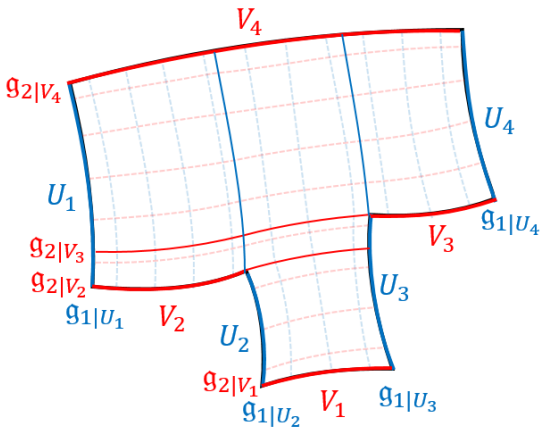


FIGURE 4.6 – Sous réserve d'une carte polycuboid $\mathfrak{g}$ sur le domaine Ω, le bord $\partial\Omega$ est divisé en un ensemble de *charts* U_i (resp. V_j) ; les iso-valeurs correspondantes de $\mathfrak{g}_1$ et $\mathfrak{g}_2$ sont désignées par $\mathfrak{g}_{1_{|U_i}}$ et $\mathfrak{g}_{2_{|V_j}}$.

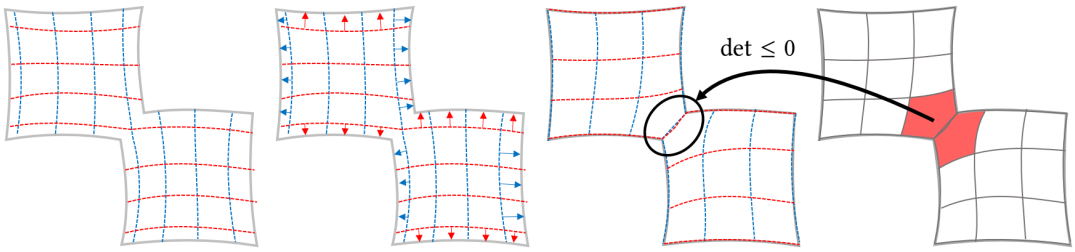


FIGURE 4.7 – Si l'on considère une carte polycuboid (un élément de $\mathcal{P}_{\mathbb{R}}$), une approche naïve pour calculer une carte polycube (un élément de $\mathcal{P}_{\mathbb{Z}}$) consiste à arrondir les valeurs du bord à l'entier le plus proche, puis à recalculer une carte afin de répartir la distorsion résultante. Dans certains cas, il est impossible de générer une carte polycube à jacobien positive, et donc l'extraction du maillage quadrangulaire échoue.

4.3 Notre approche de quantification

Dans cette section, nous introduisons deux contraintes [C1] et [C2] sur L qui intuitivement nous permettront d'obtenir une carte polycube valide. Ensuite nous construisons une carte F_L : $\mathfrak{g}(\Omega) \to \mathbb{R}^2$ pour un ensemble donné de contraintes de bord entières L (sous-section 4.3.2). Puis nous montrons que "L respecte [C1] et [C2]" si et seulement si $F_L \in \mathcal{P}_{\mathbb{Z}}$ (sous-section 4.3.3). Enfin, nous approximons le problème (4.1) comme suit :

$$\overline{f} = \underset{F_L\text{: } L \text{ respecte [C1] et [C2]}}{\arg\min} E(F_L). \tag{4.2}$$

Nous montrons que ce problème peut être résolu comme un problème d'optimisation linéaire en nombre entier (sous-section 4.3.4).

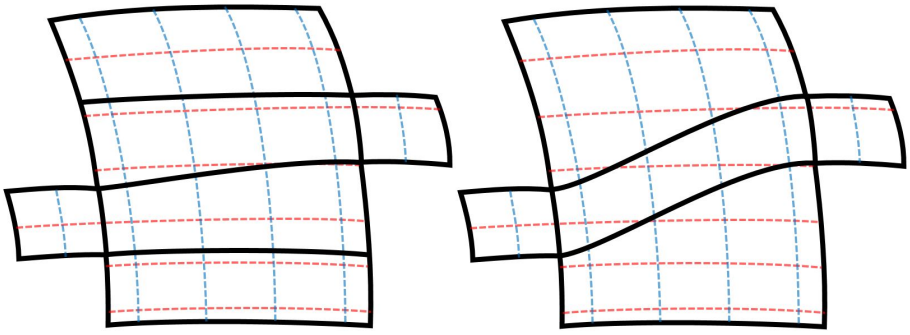


FIGURE 4.8 – Nous optimisons sur un sous-ensemble de toutes les décompositions de blocs possibles. (À gauche) la décomposition en blocs la plus grossière utilisant F_L ; (à droite) la décomposition en blocs la plus grossière sur le domaine.

4.3.1 Contraintes sur L

Comme le montre la FIGURE 4.6, les valeurs des *charts* nous permettent de décomposer Ω en blocs. La contrainte [C1] va nous assurer de garder un jacobien positif dans chaque bloc :

Définition 6. *on dit que L respecte [C1] si, étant donné un bloc limité par U_i et $U_{i'}$ (resp. V_j),*

$$i < i' \implies l_{U_i} \leq l_{U_{i'}} \qquad ([\texttt{C1}])$$

La condition [C2] va interdire l'effondrement entre deux parties du bord :

Définition 7. *On dit que L respecte [C2] si : pour tout couple de sommets du bord situés sur des* charts *différents, mais de la même dimension, et dont il existe un chemin monotone entre eux, on a*

$$signe(i' - i)(l_{U_{i'}} - l_{U_i}) + signe(j' - j)(l_{V_{j'}} - l_{V_j}) > 0, \qquad ([\texttt{C2}])$$

en considérant que les points sont issus respectivement des charts *U_i, V_j et $U_{i'}, V_{j'}$.*

La fonction *signe* renvoie 1 pour une valeur positive, et -1 autrement. Nous définissons également un chemin monotone de la façon suivante :

Définition 8. *Nous disons qu'un chemin P sur les arêtes de la décomposition en blocs est monotone si ses coordonnées $h_1(P)$ et $h_2(P)$ sont (non strictement) monotones.*

Remarque 16. *La condition [C1] restreint l'ensemble des décompositions en blocs étudiées (voir FIGURE 4.8). En pratique, cette approximation n'est pas un problème, car nous restons cohérents avec l'estimation initiale $\mathfrak{g}$. Cette limitation pourrait être évitée en utilisant des jonctions en T, mais nous laissons cela pour des travaux futurs.*

4.3.2 Construction de F_L

Rappelons qu'un domaine triangulé Ω et une carte **polycuboid** à jacobien positif $\mathfrak{g} \in \mathcal{P}_{\mathbb{R}}$ sont donnés en entrée. Nous montrons d'abord comment construire une fonction F_L pour un ensemble

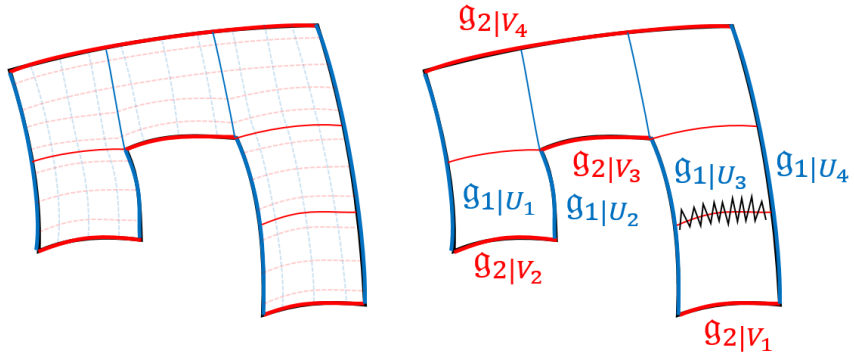


FIGURE 4.9 – La carte polycuboid d'entrée $\mathfrak{g}$ permet de décomposer le domaine Ω en blocs quadrangulaires. Nous n'extrayons pas les composantes connectées des isolignes qui ne touchent pas les "coins" du domaine Ω.

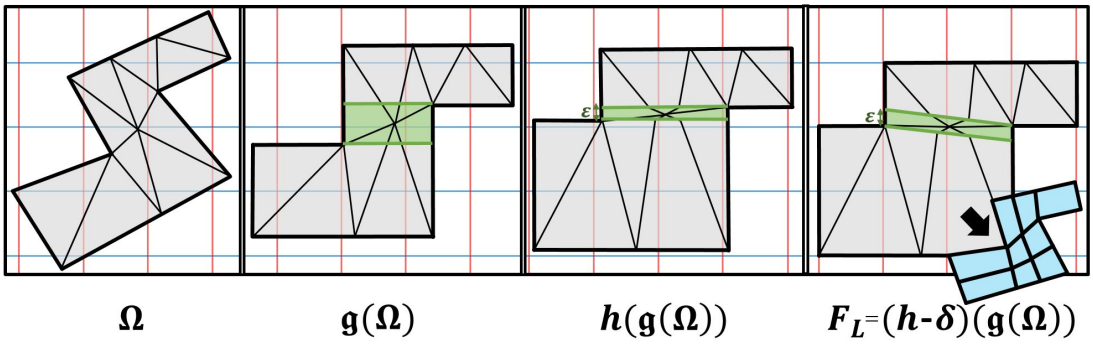


FIGURE 4.10 – La fonction h introduit une petite déformation linéaire qui déplace les bords vers une valeur presque entière. La fonction δ introduira la correction finale tout en s'assurant que la fonction $F_L = h - \delta$ est toujours injective. Notons que la fonction δ est bi-linéaire sur la décomposition en blocs (vivant sur la triangulation).

donné $L \subset \mathbb{Z}$, puis nous prouvons dans la section suivante que F_L est une carte **polycube** valide ($F_L \in \mathcal{P}_{\mathbb{Z}}$) si et seulement si L respecte les conditions [C1] et [C2].

Pour chaque *charts* U_i (resp. V_j), on peut extraire les iso-lignes de $\mathfrak{g}_1$ telles que $\mathfrak{g}_1(X) = \mathfrak{g}_{1|_{U_i}}$. Puisque $\det(D\mathfrak{g}) > 0$, ces iso-lignes décomposent le domaine Ω en blocs quadrangulaires (voir FIGURE 4.9). Pour simplifier un peu la décomposition des blocs d'entrée, nous ignorons les composantes connectées des iso-lignes qui ne touchent pas le *chart* correspondant (à droite sur la FIGURE 4.9).

Chaque bloc devient un rectangle par $\mathfrak{g}$, aligné sur les axes, limité par les valeurs des quatre *charts* $U_i, U_{i'}, V_j, V_{j'}$ qui le délimitent. Sans perte de généralité, on considère que $\mathfrak{g}_{1|U_i} < \mathfrak{g}_{1|U_{i'}} \iff i < i'$ (resp. V_j).

En s'appuyant sur cette décomposition en blocs, nous définissons F_L comme une différence de deux fonctions :

$$F_L := h - \delta. \tag{4.3}$$

L'idée sous-jacente est de définir la fonction h qui est **presque** une carte polycube (voir FIGURE 4.10). Si L satisfait la contrainte [C1], la fonction h est toujours une carte **polycuboid** (valide) ($h \in \mathcal{P}_{\mathbb{R}}$), mais le bord $\partial\Omega$ a des valeurs très proches des valeurs entières données par

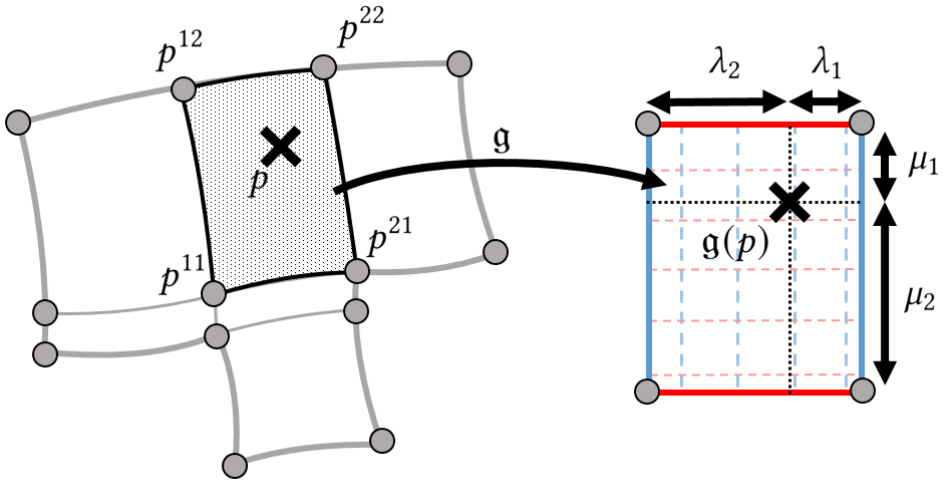


FIGURE 4.11 – Chaque bloc quadrangulaire est mis en correspondance avec un rectangle aligné sur les axes par $\mathfrak{g}$. Pour chaque point p d'un bloc, nous pouvons trouver les coordonnées *pseudo*-barycentriques de son image $\mathfrak{g}(p)$ par rapport au rectangle aligné sur les axes ; cela nous permet de définir des fonctions bi-linéaires sur le bloc original en prescrivant 4 valeurs aux sommets du bloc.

L. La fonction h a un jacobien positif ($\det(Dh) > 0$), et une petite correction δ est nécessaire pour faire de $h - \delta$ une carte **polycube** valide ($h - \delta \in \mathcal{P}_{\mathbb{Z}}$). Cette correction donne ce résultat uniquement si L satisfait également à la contrainte [C2].

La construction de h et de δ repose sur des ε arbitrairement petits, ce qui nous permet de construire des preuves qui garantissent la validité de la carte dans la section suivante. En pratique, nous travaillons directement sur la combinatoire du maillage hexaédrique final (voir sec. 4.4.3), et n'avons pas besoin d'une expression explicite de F_L partout. Cela signifie que nous pouvons utiliser une valeur symbolique pour ε.

Nous représentons h et δ en prescrivant leurs valeurs aux sommets de la décomposition du bloc. Référons-nous à la FIGURE 4.11 : étant donné un bloc avec quatre sommets p^{11}, p^{12}, p^{21} et p^{22}, il est mis en correspondance avec un rectangle aligné sur l'axe par $\mathfrak{g}$. Si l'on veut évaluer une fonction f représentée par quatre valeurs $f(p^{11}), f(p^{12}), f(p^{21})$ et $f(p^{22})$, alors pour tout point p du bloc on peut trouver ses coordonnées *pseudo*-barycentriques $\lambda_1, \lambda_2, \mu_1, \mu_2$ par rapport au rectangle aligné sur les axes, et $f(p)$ peut être évalué comme suit :

$$f(p) = \sum_{i,j \in \{1,2\}} \frac{\lambda_i \mu_j}{(\lambda_1 + \lambda_2)(\mu_1 + \mu_2)} f(p^{ij}).$$

Ainsi, la décomposition en blocs nous permet de représenter des fonctions bi-linéaires continues par blocs ; h et δ sont définis de cette façon.

Construction de la fonction h

Comme mentionné précédemment, nous représentons la fonction h en spécifiant ses valeurs aux sommets de la décomposition en blocs. Ainsi, pour chaque sommet généré par deux cartes U_i et V_j, nous définissons h comme étant égale à $(l_{U_i} + i\varepsilon,\ l_{V_j} + j\varepsilon)$ avec $\varepsilon > 0$ et l'interpolation bi-linéaire permet d'interpoler ces valeurs dans l'interieur des blocs.

Algorithme 2 : Calcul de δ

Résultat : Valeur de $\delta(s)$ at tout les points s de la décomposition en bloc B

1 **pour** $s \in B$ **faire**
2 | $\delta_1(s) = 0$; `visited`$[s]$ = `false`;
3 **pour** $s \in U_i \cap \partial\Omega$ **faire**
4 | $\delta_1(s) = i\varepsilon$; `visited`$[s]$ = `true`;
5 **tant que** *il y a une arête* (s,t) *telle que* $\|h(s) - h(t)\| < |L|\varepsilon$ *et* `visited`$[s]$ *et* !`visited`$[t]$ **faire**
6 | $\delta_1(t) = \delta_1(s)$;
7 | `visited`$[t]$ = `true`;
8 δ_2 est calculé de la même façon.;

Nous montrons dans la section suivante que $\det(Dh) > 0$ si L satisfait la contrainte [C1]. Ainsi, la carte h appartient à $\mathcal{P}_{\mathbb{R}}$; autrement dit, c'est une carte polycuboid (à jacobien positive). De plus, elle ne diffère que de $O(\varepsilon)$ de la cible, et seule une petite correction δ est nécessaire pour que la fonction $F_L = h - \delta$ soit une carte **polycube**.

Construction de la fonction δ

Nous devons calculer une fonction de correction δ telle que :

1. $h - \delta$ a les valeurs entières désirées L sur le bord $\partial\Omega$;
2. le Jacobien $\det(DF_L)$ est positif.

Comme h, la fonction δ est représentée par ses valeurs aux sommets de la décomposition en blocs, et est interpolée ailleurs. L'algorithme 2 montre comment calculer les valeurs en question en itérant sur les bords de la décomposition en blocs.

La construction est simple : d'abord, nous garantissons que $h - \delta$ a les valeurs désirées L sur le bord (lignes 3–4 de Algorithme 2). Ensuite, pour assurer la positivité du jacobien $\det(D(h - \delta)) > 0$, nous effectuons une propagation de front (lignes 5–7) sur les arêtes de la décomposition en blocs en partant de chaque carte. L'idée est d'avoir toutes les variations de δ situées où la variation de h est suffisante pour garder la positivité du jacobien.

Plus précisément, notre algorithme garantit que si une arête (s,t) de la décomposition en blocs est "courte" sous l'action de h (i.e. $\|h(s) - h(t)\| < |L|\varepsilon$), alors la correction δ garde sa longueur inchangée, i.e. $\delta(s) = \delta(t)$. Dans la section suivante, nous démontrons que cette condition implique la positivité du jacobien $\det(D(h - \delta)) > 0$.

Il y a cependant une mise en garde. L'algorithme propage les fronts à partir des *charts*, la propagation se fait le long des arêtes "courtes" de la décomposition en blocs. Que se passe-t-il si deux fronts différents, provenant par exemple des *charts* U_i et U_j, se rencontrent sur un bord ? Par construction, cela implique qu'il existe un chemin entièrement constitué d'arêtes "courtes" qui relie un sommet de U_i à un sommet de U_j. Cela implique l'égalité des contraintes de limites entières $l_{U_i} = l_{U_j}$ et $l_{V_{i'}} = l_{V_{j'}}$. À son tour, cela implique qu'il est impossible de construire une carte polycube valide représentée comme une interpolation bi-linéaire sur la décomposition en blocs.

La Figure 4.7 fournit un exemple typique de violation de la contrainte [C2] : deux points différents de Ω sont placés au même endroit dans l'espace paramétrique, ce qui conduit à un Jacobien dégénéré. Pour éviter cela, nous posons des contraintes supplémentaires sur l'ensemble L : nous obligeons certains chemins sur les arêtes de la décomposition en blocs à contenir des arêtes "longues". Il suffit de ne considérer que des chemins monotones, c'est-à-dire des chemins qui ne font pas d'aller-retour dans chacune des dimensions. Ainsi, [C2] contraint chaque chemin

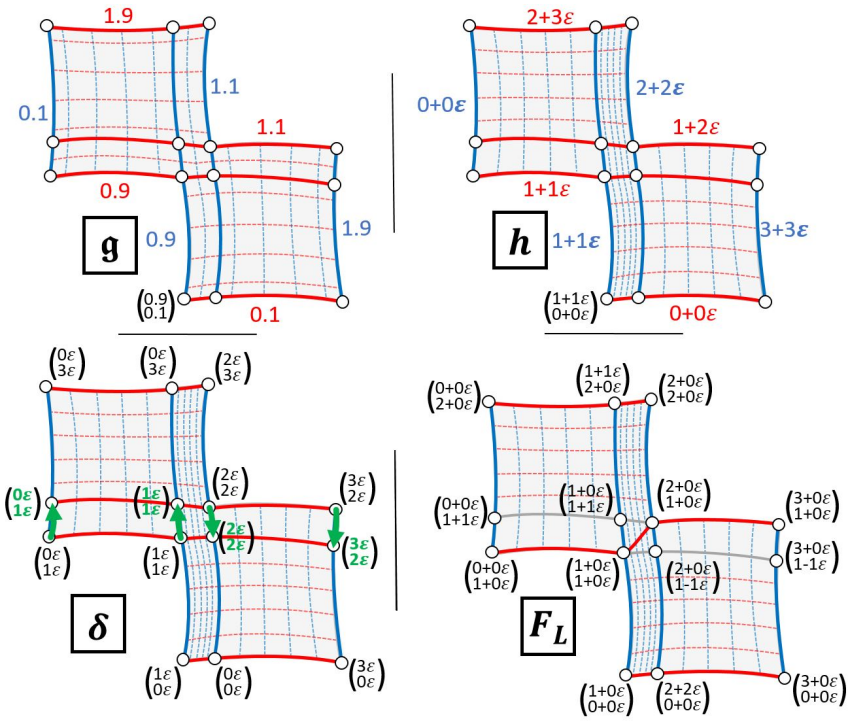


FIGURE 4.12 – Valeurs des fonctions définies $\mathfrak{g}, h, \delta$ et F_L sur un exemple pratique. Pour les fonctions δ et F_L, les valeurs sont données aux points, et peuvent ensuite être interpolées dans les blocs. La différence entre deux lignes pointillées de même couleur représente une variation de $0,2$ dans la fonction correspondante. Les flèches vertes (en bas à gauche) représentent la procédure de propagation introduite par l'algorithme 2, notez que les lignes pointillées sont ici issues de h, pour mettre en évidence les blocs "courts" tels que définis dans la sous-section 4.3.2.

monotone entre deux points quelconques de cartes différentes dans la même dimension, doit contenir une arête "longue". La FIGURE 4.12 montre les valeurs obtenues pour δ et F_L par la contrainte [C2] sur le même domaine.

4.3.3 Preuve de positivité de $\det(DF_L)$ sous [C1] et [C2]

Dans cette section, nous donnons une preuve de la positivité de $\det(DF_L)$ sous [C1] et [C2]. Pour cela, nous commençons par montrer que $\det(Dh) > 0$, et puis que $det(DF_L) > 0$ en 2D et en 3D.

Positivité de $\det(Dh)$

Nous procéderons en deux étapes, d'abord nous donnerons une définition mathématique de h puis, à partir de celle-ci, nous déduirons la nécessité de la contrainte [C1].

Définition de h Dans cette section, nous donnons une définition de h qui est équivalente à celle donnée dans la Section 4.3.2, mais apporte une autre intuition. Nous savons que chaque bloc créé par quatre cartes $U_i, U_{i'}, V_j, V_{j'}$ devient, par $\mathfrak{g}$, un rectangle aligné sur l'axe (FIGURE 4.13) délimité par $\mathfrak{g}_{1_{|U_i}}, \mathfrak{g}_{1_{|U_{i'}}}, \mathfrak{g}_{2_{|V_j}}$ et $\mathfrak{g}_{2_{|V_{j'}}}$. Il est facile d'étirer ce rectangle vers tout autre rectangle aligné sur un axe en modifiant chaque axe indépendamment. Nous pouvons donc définir h comme

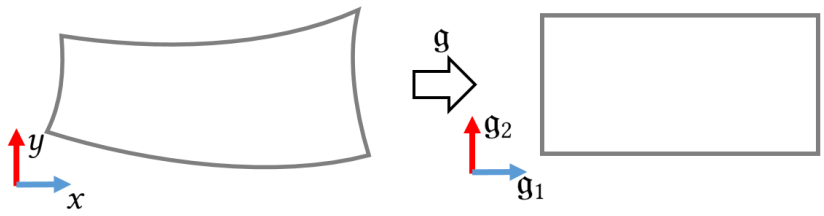


FIGURE 4.13 – Chaque bloc de Ω est mis en correspondance avec un rectangle par $\mathfrak{g}$, ce qui donne un changement de système de coordonnées.

$h(X) := (A_1(\mathfrak{g}_1(X)),\ A_2(\mathfrak{g}_2(X)))$ avec $A_1, A_2 : \mathbb{R} \to \mathbb{R}$ étant deux fonctions affines continues par blocs.

Par définition, la fonction h fait correspondre chaque bloc défini par quatre *charts* $U_i, U_{i'}, V_j, V_{j'}$ au rectangle aligné sur l'axe délimité par $l_{U_i} + i\varepsilon,\ l_{U_{i'}} + i'\varepsilon,\ l_{V_j} + j\varepsilon,\ l_{V_{j'}} + j'\varepsilon$ pour un petit $\varepsilon > 0$.

Sur un bloc défini par $U_i, U_{i'}, V_j, V_{j'}$, la fonction $\mathfrak{g}_1$ est bornée par $\mathfrak{g}_{1|U_i}$ et $\mathfrak{g}_{1|U_{i'}}$ en première dimension. La transformation affine A_1 est définie de la façon suivante :

$$\begin{aligned} A_1(\boldsymbol{x}) :&= \frac{l_{U_{i'}} + i'\varepsilon - (l_{U_i} + i\varepsilon)}{\mathfrak{g}_{1|U_{i'}} - \mathfrak{g}_{1|U_i}}\boldsymbol{x} + b_1 \\ &= a_1\mathfrak{g}_1(x) + b_1 \end{aligned}$$

avec b_1 constante choisie de telle sorte que A_1 transforme $\left[\mathfrak{g}_{1|U_i}, \mathfrak{g}_{1|U_{i'}}\right]$ en $\left[l_{U_i} + i\varepsilon, l_{U_{i'}} + i'\varepsilon\right]$. Les valeurs de $\mathfrak{g}_{1|U_i}$ et l_{U_i} étant définies le long des iso-lignes, la fonction h est continue à l'interface entre blocs (mais non différentiable).

La transformation affine dans la deuxième dimension, A_2, est donnée par la même construction.

Calcul de $\det(Dh)$ Les variations de h sont strictement liées à A_1, A_2 et $\mathfrak{g}$, *i.e.* donc Dh a la forme suivante :

$$Dh = D\begin{pmatrix} A_1(\mathfrak{g}_1(X)) \\ A_2(\mathfrak{g}_2(X)) \end{pmatrix} = \begin{pmatrix} \frac{\partial h_1}{\partial \mathfrak{g}_1} & \frac{\partial h_1}{\partial \mathfrak{g}_2} \\ \frac{\partial h_2}{\partial \mathfrak{g}_1} & \frac{\partial h_2}{\partial \mathfrak{g}_2} \end{pmatrix} D\mathfrak{g} = \begin{pmatrix} a_1 & 0 \\ 0 & a_2 \end{pmatrix} D\mathfrak{g}$$

ce qui signifie que

$$\det(Dh) = a_1 a_2 \det(D\mathfrak{g}).$$

En se rappelant que $a_1 = \frac{l_{U_{i'}} + i'\varepsilon - (l_{U_i} + i\varepsilon)}{\mathfrak{g}_{1|U_{i'}} - \mathfrak{g}_{1|U_i}}$, et que $\mathfrak{g}_{1|U_{i'}} - \mathfrak{g}_{1|U_i} > 0$ et $\mathfrak{g}_{2|V_{j'}} - \mathfrak{g}_{2|V_j} > 0$, pour que $\det(Dh)$ soit positif, $l_{U_{i'}} + i'\varepsilon - (l_{U_i} + i\varepsilon)$ et $l_{V_{j'}} + j'\varepsilon - (l_{V_j} + j\varepsilon)$ doivent avoir le même signe. Ceci étant nécessaire pour tous les blocs par continuité, cela signifie que les signes doivent être les mêmes sur tous les blocs. Pouvant considérer que $\widetilde{l_{U_i}} = -l_{U_i}$, et $\widetilde{l_{V_j}} = -l_{V_j}$, pour tous les *charts*, on en déduit la contrainte [C1], *i.e.* $l_{U_i} \leq l_{U_{i'}}$ et $l_{V_j} \leq l_{V_{j'}}$ sur chaque bloc.

Positivité de $\det(DF_L)$ **sous** [C1] **et** [C2]

Étudions maintenant DF_L, d'abord en dimension 2, puis en dimension 3, avec enfin une étude du paramètre ε.

Dimension 2 : Par hypothèse, [C1] et [C2] sont satisfaits par L. Formellement, cela signifie que :

1. les coefficients a_1 et a_2 sont positifs (sous-section précédente),
2. pour tout bord donné de la structure de bloc (s,t), $\|h(s)-h(t)\| < |L|\varepsilon \implies \delta(s) = \delta(t)$.

Notre but est de montrer que, sous ces hypothèses, $\det(DF_L) > 0$. Étudions le déterminant du jacobien de F_L, en utilisant les résultats développés précédemment sur $\det(Dh)$:

$$\begin{aligned}
\det(DF_L) &= \det(Dh - D\delta) \\
&= \det\left(\left[\begin{pmatrix} a_1 & 0 \\ 0 & a_2 \end{pmatrix} - \begin{pmatrix} \frac{\partial\delta_1}{\partial\mathfrak{g}_1} & \frac{\partial\delta_1}{\partial\mathfrak{g}_2} \\ \frac{\partial\delta_2}{\partial\mathfrak{g}_1} & \frac{\partial\delta_2}{\partial\mathfrak{g}_2} \end{pmatrix}\right] D\mathfrak{g}\right) \\
&= \det(D\mathfrak{g})\left[\left(a_1 - \frac{\partial\delta_1}{\partial\mathfrak{g}_1}\right)\left(a_2 - \frac{\partial\delta_2}{\partial\mathfrak{g}_2}\right) - \frac{\partial\delta_1}{\partial\mathfrak{g}_2}\frac{\partial\delta_2}{\partial\mathfrak{g}_1}\right]
\end{aligned}$$

En rappelant que, par construction, $|\frac{\partial\delta_i}{\partial\mathfrak{g}_j}| < |L|\varepsilon$ (voir Section 4.3.2), on peut distinguer deux cas :

1. $a_1 \geq 1$ **et** $a_2 \geq 1$ **:**

Dans ce cas, ε peut être choisi suffisamment petit pour que $\left(a_1 - \frac{\partial\delta_1}{\partial\mathfrak{g}_1}\right)$ et $\left(a_2 - \frac{\partial\delta_2}{\partial\mathfrak{g}_2}\right)$ soient positifs, et que $\frac{\partial\delta_1}{\partial\mathfrak{g}_2}\frac{\partial\delta_2}{\partial\mathfrak{g}_1}$ soit négligeable par rapport à a_1a_2.

2. $1 > a_1 > 0$ **ou (inclusif)** $1 > a_2 > 0$ **:**

Ce cas est le plus difficile. Par construction de h, avoir $1 > a_i > 0$ implique que a_i est d'ordre ε. Dans la suite, en utilisant l'hypothèse (2), nous allons montrer que si a_i est d'ordre ε, alors $\frac{\partial\delta_1}{\partial\mathfrak{g}_i} = \frac{\partial\delta_2}{\partial\mathfrak{g}_i} = 0$, ce qui nous permettra de conclure que $\det(F_L) > 0$. Tout d'abord, nous devons rappeler que chaque bloc de Ω est porté sur un rectangle, avec $(\mathfrak{g}_1, \mathfrak{g}_2)$ le nouveau système de coordonnées (se référer à la FIGURE 4.13). A partir de là, on remarque que le long d'une arête (s,t) de direction $\mathfrak{g}_1$ on a $\|h(s) - h(t)\| = a_1(\mathfrak{g}_1(s) - \mathfrak{g}_1(t))$, ce qui signifie que $\|h(s)-h(t)\| < |L|\varepsilon$. Cela implique que a_1 est d'ordre ε, et réciproquement, car $\mathfrak{g}$ est ici une constante.

Ensuite, en utilisant l'hypothèse (2), le long d'une arête (s,t) de direction $\mathfrak{g}_1$, $\|h(s) - h(t)\| < |L|\varepsilon \implies \delta(s) = \delta(t)$. Remarquons que $\delta(s) = \delta(t)$ est équivalent à $\frac{\partial\delta_1}{\partial\mathfrak{g}_1} = \frac{\partial\delta_2}{\partial\mathfrak{g}_1} = 0$ le long du bord en raison de la linéarité de δ le long de (s,t). Cela signifie que a_1 d'ordre ε implique $\frac{\partial\delta_1}{\partial\mathfrak{g}_1} = \frac{\partial\delta_2}{\partial\mathfrak{g}_1} = 0$. Le même raisonnement vaut pour a_2 et $\mathfrak{g}_2$.

Le dernier problème est de généraliser cette propriété au reste du bloc (pas seulement le long des bords). Notre construction de δ est bilinéaire, utilisant les coordonnées de chaque sommet du bloc. Comme le bloc est un rectangle dans h, les deux bords opposés se comportent de la même manière, et le comportement de ces bords sera interpolé linéairement dans le bloc. Si $\frac{\partial\delta_i}{\partial\mathfrak{g}_j} = 0$ sur les deux bords opposés, alors $\frac{\partial\delta_i}{\partial\mathfrak{g}_j} = 0$ dans tout le bloc. Ce qui signifie que notre résultat sur le bord est également vrai dans le bloc.

■

Dimension 3 : Nous proposons une étude plus rapide sur le cas 3D. Commençons par effectuer une décomposition du déterminant comme pour le cas 2D :

$$\begin{aligned}\det\left(DF_L\right) &= \det\left(\left[\begin{pmatrix} a_1 & 0 & 0 \\ 0 & a_2 & 0 \\ 0 & 0 & a_3 \end{pmatrix} - \begin{pmatrix} \frac{\partial\delta_1}{\partial\mathfrak{g}_1} & \frac{\partial\delta_1}{\partial\mathfrak{g}_2} & \frac{\partial\delta_1}{\partial\mathfrak{g}_3} \\ \frac{\partial\delta_2}{\partial\mathfrak{g}_1} & \frac{\partial\delta_2}{\partial\mathfrak{g}_2} & \frac{\partial\delta_2}{\partial\mathfrak{g}_3} \\ \frac{\partial\delta_3}{\partial\mathfrak{g}_1} & \frac{\partial\delta_3}{\partial\mathfrak{g}_2} & \frac{\partial\delta_3}{\partial\mathfrak{g}_3} \end{pmatrix}\right] D\mathfrak{g}\right) \\ &= \det\left(\begin{bmatrix} a_1 & 0 & 0 \\ 0 & a_2 & 0 \\ 0 & 0 & a_3 \end{bmatrix} - \begin{bmatrix} \frac{\partial\delta_1}{\partial\mathfrak{g}_1} & \frac{\partial\delta_1}{\partial\mathfrak{g}_2} & \frac{\partial\delta_1}{\partial\mathfrak{g}_3} \\ \frac{\partial\delta_2}{\partial\mathfrak{g}_1} & \frac{\partial\delta_2}{\partial\mathfrak{g}_2} & \frac{\partial\delta_2}{\partial\mathfrak{g}_3} \\ \frac{\partial\delta_3}{\partial\mathfrak{g}_1} & \frac{\partial\delta_3}{\partial\mathfrak{g}_2} & \frac{\partial\delta_3}{\partial\mathfrak{g}_3} \end{bmatrix}\right) \det(D\mathfrak{g}) \\ &= \det(M)\det(D\mathfrak{g})\end{aligned}$$

Comme $\det(D\mathfrak{g}) > 0$, on se retrouve à étudier $\det(M)$, qui a la forme :

$$\begin{aligned}\det(M) = &\left(a_1 - \frac{\partial\delta_1}{\partial\mathfrak{g}_1}\right)\left[\left(a_2 - \frac{\partial\delta_2}{\partial\mathfrak{g}_2}\right)\left(a_3 - \frac{\partial\delta_3}{\partial\mathfrak{g}_3}\right) - \frac{\partial\delta_3}{\partial\mathfrak{g}_2}\frac{\partial\delta_2}{\partial\mathfrak{g}_3}\right] \\ &+ \frac{\partial\delta_1}{\partial\mathfrak{g}_2}\left[\frac{\partial\delta_2}{\partial\mathfrak{g}_3}\frac{\partial\delta_3}{\partial\mathfrak{g}_1} - \left(a_3 - \frac{\partial\delta_3}{\partial\mathfrak{g}_3}\right)\frac{\partial\delta_2}{\partial\mathfrak{g}_1}\right] + \frac{\partial\delta_1}{\partial\mathfrak{g}_3}\left[\frac{\partial\delta_2}{\partial\mathfrak{g}_1}\frac{\partial\delta_3}{\partial\mathfrak{g}_2} - \left(a_2 - \frac{\partial\delta_2}{\partial\mathfrak{g}_2}\right)\frac{\partial\delta_3}{\partial\mathfrak{g}_1}\right]\end{aligned}$$

Que l'on peut développer de la façon suivante :

$$\det(M) = \left(a_1 - \frac{\partial\delta_1}{\partial\mathfrak{g}_1}\right)\left(a_2 - \frac{\partial\delta_2}{\partial\mathfrak{g}_2}\right)\left(a_3 - \frac{\partial\delta_3}{\partial\mathfrak{g}_3}\right) \tag{4.4}$$

$$- \left(a_1 - \frac{\partial\delta_1}{\partial\mathfrak{g}_1}\right)\frac{\partial\delta_3}{\partial\mathfrak{g}_2}\frac{\partial\delta_2}{\partial\mathfrak{g}_3} \tag{4.5}$$

$$- \left(a_2 - \frac{\partial\delta_2}{\partial\mathfrak{g}_2}\right)\frac{\partial\delta_1}{\partial\mathfrak{g}_3}\frac{\partial\delta_3}{\partial\mathfrak{g}_1} \tag{4.6}$$

$$- \left(a_3 - \frac{\partial\delta_3}{\partial\mathfrak{g}_3}\right)\frac{\partial\delta_2}{\partial\mathfrak{g}_1}\frac{\partial\delta_1}{\partial\mathfrak{g}_2} \tag{4.7}$$

$$+ \left(\frac{\partial\delta_1}{\partial\mathfrak{g}_2}\frac{\partial\delta_2}{\partial\mathfrak{g}_3}\frac{\partial\delta_3}{\partial\mathfrak{g}_1} + \frac{\partial\delta_1}{\partial\mathfrak{g}_3}\frac{\partial\delta_2}{\partial\mathfrak{g}_1}\frac{\partial\delta_3}{\partial\mathfrak{g}_2}\right) \tag{4.8}$$

Maintenant il faut remarquer que $a_i = O(\varepsilon) \implies \frac{\partial\delta_1}{\partial\mathfrak{g}_i} = \frac{\partial\delta_2}{\partial\mathfrak{g}_i} = \frac{\partial\delta_3}{\partial\mathfrak{g}_i} = 0$, comme détaillé pour la dimension 2. De même, rappelons que $\frac{\partial\delta_j}{\partial\mathfrak{g}_i} = O(\varepsilon)$. Nous avons que, si $\forall i, a_i = O(1)$, alors le terme (4.4) l'emporte sur tous les autres, et $\det(M) > 0$.

Imaginons maintenant que $a_1 = O(\varepsilon)$, nous avons donc $\frac{\partial\delta_1}{\partial\mathfrak{g}_1} = \frac{\partial\delta_2}{\partial\mathfrak{g}_1} = \frac{\partial\delta_3}{\partial\mathfrak{g}_1} = 0$, ce qui implique que les termes (4.6), (4.7) et (4.8) deviennent nuls. $\det(M)$ se récrit donc de la façon suivante :

$$\det(M) = a_1\left[\left(a_2 - \frac{\partial\delta_2}{\partial\mathfrak{g}_2}\right)\left(a_3 - \frac{\partial\delta_3}{\partial\mathfrak{g}_3}\right) - \frac{\partial\delta_3}{\partial\mathfrak{g}_2}\frac{\partial\delta_2}{\partial\mathfrak{g}_3}\right] \tag{4.9}$$

Et comme $\left(a_2 - \frac{\partial\delta_2}{\partial\mathfrak{g}_2}\right) = O(1)$, $\left(a_3 - \frac{\partial\delta_3}{\partial\mathfrak{g}_3}\right) = O(1)$ et $\frac{\partial\delta_3}{\partial\mathfrak{g}_2}\frac{\partial\delta_2}{\partial\mathfrak{g}_3} = O(\varepsilon^2)$, $\det(M)$ reste bien positif. On peut effectuer un calcul similaire lorsque a_2 et a_3 deviennent d'ordre ε, et l'on obtient bien que $\det(DF_L)$ reste strictement positif. En effet, $\det(M)$ prendra les formes suivantes :

$$a_1 = O(\varepsilon), a_2 = O(\varepsilon), a_3 = O(1) \implies \det(M) = a_1 a_2\left(a_3 - \frac{\partial\delta_3}{\partial\mathfrak{g}_3}\right)$$

$$a_1 = O(\varepsilon), a_2 = O(\varepsilon), a_3 = O(\varepsilon) \implies \det(M) = a_1 a_2 a_3$$

■

Étude de ε

Nous pouvons remarquer que s'il est nécessaire de construire explicitement F_L, alors une valeur est nécessaire pour ε, car avec un ε arbitrairement petit, le déterminant de F_L peut lui aussi devenir arbitrairement petit ($O(\varepsilon^3)$).

Dans la suite, nous utiliserons une approche d'epsilon symbolique, évitant la construction explicite de F_L, qu'il est possible d'utiliser, car nous travaillons directement sur la combinatoire du maillage final. Plus que cela, il est nécessaire de l'utiliser, car nous travaillerons sur des cartes $\mathfrak{g}$ non parfaitement bijectives, ce qui pose un problème sur les constructions précédemment posées. Nous détaillons tout cela dans la section 4.4.

De futures avancés permettant des constructions de cartes $\mathfrak{g}$ bijectives pourraient rendre pertinent la construction explicite de la carte F_L, pour éviter les constructions complexes de notre implémentation actuelle, et d'obtenir une simple modification de la carte $\mathfrak{g}$ qui pourrait être utilisée pour directement extraire les hexaèdres. Pour cela, nous pouvons trouver une valeur de ε qui nous permettrait de calculer explicitement F_L.

Plus précisément, nous avons le théorème suivant :

Théorème 3. *Sous les contraintes* [C1]*et* [C2]*,* $\varepsilon = \frac{1}{4|L|} \implies F_L \in \mathcal{P}_{\mathbb{Z}}$.

Ce résultat nous permet d'obtenir une construction explicite pour F_L. En se rappelant, de la preuve précédente, que $\det(DF_L) = \det(M)\det(D\mathfrak{g})$, nous pouvons déduire la proposition 3 du lemme suivant :

Lemme 2. *Soit* $m > 3$ *et* $\varepsilon = \frac{1}{m|L|}$*, alors* $\det(M) > 0$.

Démonstration. Commençons par réécrire $\det(M)$ de la façon suivante :

$$\det(M) = \frac{1}{4}\left(a_1 - \frac{\partial\delta_1}{\partial\mathfrak{g}_1}\right)\left[\left(a_2 - \frac{\partial\delta_2}{\partial\mathfrak{g}_2}\right)\left(a_3 - \frac{\partial\delta_3}{\partial\mathfrak{g}_3}\right) - 4\frac{\partial\delta_3}{\partial\mathfrak{g}_2}\frac{\partial\delta_2}{\partial\mathfrak{g}_3}\right] \tag{4.10}$$

$$+ \frac{1}{4}\left(a_2 - \frac{\partial\delta_2}{\partial\mathfrak{g}_2}\right)\left[\left(a_1 - \frac{\partial\delta_1}{\partial\mathfrak{g}_1}\right)\left(a_3 - \frac{\partial\delta_3}{\partial\mathfrak{g}_3}\right) - 4\frac{\partial\delta_1}{\partial\mathfrak{g}_3}\frac{\partial\delta_3}{\partial\mathfrak{g}_1}\right] \tag{4.11}$$

$$+ \frac{1}{4}\left(a_3 - \frac{\partial\delta_3}{\partial\mathfrak{g}_3}\right)\left[\left(a_1 - \frac{\partial\delta_1}{\partial\mathfrak{g}_1}\right)\left(a_2 - \frac{\partial\delta_2}{\partial\mathfrak{g}_2}\right) - 4\frac{\partial\delta_2}{\partial\mathfrak{g}_1}\frac{\partial\delta_1}{\partial\mathfrak{g}_2}\right] \tag{4.12}$$

$$+ \frac{1}{4}\left(a_1 - \frac{\partial\delta_1}{\partial\mathfrak{g}_1}\right)\left(a_2 - \frac{\partial\delta_2}{\partial\mathfrak{g}_2}\right)\left(a_3 - \frac{\partial\delta_3}{\partial\mathfrak{g}_3}\right) + \left(\frac{\partial\delta_1}{\partial\mathfrak{g}_2}\frac{\partial\delta_2}{\partial\mathfrak{g}_3}\frac{\partial\delta_3}{\partial\mathfrak{g}_1} + \frac{\partial\delta_1}{\partial\mathfrak{g}_3}\frac{\partial\delta_2}{\partial\mathfrak{g}_1}\frac{\partial\delta_3}{\partial\mathfrak{g}_2}\right) \tag{4.13}$$

Pour prouver notre lemme, il suffit de trouver une valeur de ε pour laquelle toutes les lignes (4.10) à (4.13) sont positives. Pour cela, revenons aux définitions de a_i et $\frac{\partial\delta_i}{\partial\mathfrak{g}_j}$. Premièrement, nous avons

$$a_1 = \frac{l_{U_{i'}} + i'\varepsilon - (l_{U_i} + i\varepsilon)}{\mathfrak{g}_{1|U_{i'}} - \mathfrak{g}_{1|U_i}} = \frac{l_{U_{i'}} + i'\varepsilon - (l_{U_i} + i\varepsilon)}{k_1}$$

avec $k_1 = \mathfrak{g}_{1|U_{i'}} - \mathfrak{g}_{1|U_i}$. Remarquons que si $a_1 \neq O(\varepsilon)$, alors $l_{U_{i'}} > l_{U_i}$ (la relation d'ordre est due à [C1]) ce qui implique :

$$a_1 > \frac{1}{k_1}. \tag{4.14}$$

Et nous avons le même résultat pour a_2 et a_3 avec k_2 et k_3. Ensuite, il convient de remarquer, en utilisant l'algorithme 2, et la tri-linéarité de δ, que :

$$\left|\frac{\partial\delta_j}{\partial\mathfrak{g}_i}\right| < \frac{|L|\varepsilon}{k_i} = c_i^{\varepsilon} \tag{4.15}$$

En effet, la variation le long d'une arête de la décomposition en blocs est bornée en ε. Nous pouvons maintenant étudier l'effet de ε. Nous proposons de poser :

$$\varepsilon = \frac{1}{m|L|}$$

avec $m \in \mathbb{R}$, ce qui permet d'obtenir c_i^ε sous la forme

$$c_i^\varepsilon = \frac{1}{mk_i}. \tag{4.16}$$

Revenons maintenant à $\det(M)$ et étudions plus particulièrement les lignes (4.10) et (4.13), les lignes (4.11) et (4.12) étant équivalentes à (4.10). Commençons par remarquer que, grâce à [C2], si $a_1 = O(\varepsilon)$, alors tous les termes pouvant être négatifs s'annulent, de la même façon que dans la preuve précédente, permettant d'avoir $\det(M) > 0$. Il reste donc le cas où $a_1 \neq O(\varepsilon)$, pour lequel l'inéquation (4.14) est vraie. Commençons par (4.10) :

Positivité de (4.10) **:** nous avons un produit de 2 termes, trouvons une valeur de m pour que chaque terme soit positif. Le premier terme est plutôt trivial, nous avons par (4.14), (4.15) et (4.16), que

$$\begin{aligned}
\left(a_1 - \frac{\partial \delta_1}{\partial \mathfrak{g}_1}\right) > \left(\frac{1}{k_1} - \frac{1}{mk_1}\right) > 0 \\
\iff \frac{m-1}{mk_1} > 0 \\
\iff m > 1
\end{aligned} \tag{4.17}$$

Nous procédons de la même façon pour le second terme :

$$\begin{aligned}
\left(a_2 - \frac{\partial \delta_2}{\partial \mathfrak{g}_2}\right)\left(a_3 - \frac{\partial \delta_3}{\partial \mathfrak{g}_3}\right) - 4\frac{\partial \delta_3}{\partial \mathfrak{g}_2}\frac{\partial \delta_2}{\partial \mathfrak{g}_3} > \left(\frac{1}{k_2} - \frac{1}{mk_2}\right)\left(\frac{1}{k_3} - \frac{1}{mk_3}\right) - 4\frac{1}{mk_2}\frac{1}{mk_3} > 0 \\
\iff \left(\frac{1}{k_2} - \frac{1}{mk_2}\right)\left(\frac{1}{k_3} - \frac{1}{mk_3}\right) > 4\frac{1}{mk_2}\frac{1}{mk_3} \\
\iff \left(\frac{m-1}{m}\right)^2 \frac{1}{k_2 k_3} > \left(\frac{2}{m}\right)^2 \frac{1}{k_2 k_3} \\
\iff \frac{m-1}{m} > \frac{2}{m} \\
\iff m > 3
\end{aligned} \tag{4.18}$$

Positivité de (4.13) **:** Ce dernier terme est le plus verbeux à étudier, mais l'idée reste la même :

$$\begin{aligned}
\frac{1}{4}\left(a_1 - \frac{\partial \delta_1}{\partial \mathfrak{g}_1}\right)\left(a_2 - \frac{\partial \delta_2}{\partial \mathfrak{g}_2}\right)\left(a_3 - \frac{\partial \delta_3}{\partial \mathfrak{g}_3}\right) + \left(\frac{\partial \delta_1}{\partial \mathfrak{g}_2}\frac{\partial \delta_2}{\partial \mathfrak{g}_3}\frac{\partial \delta_3}{\partial \mathfrak{g}_1} + \frac{\partial \delta_1}{\partial \mathfrak{g}_3}\frac{\partial \delta_2}{\partial \mathfrak{g}_1}\frac{\partial \delta_3}{\partial \mathfrak{g}_2}\right) > \\
> \frac{1}{4}\left(\frac{1}{k_1} - \frac{1}{mk_1}\right)\left(\frac{1}{k_2} - \frac{1}{mk_2}\right)\left(\frac{1}{k_3} - \frac{1}{mk_3}\right) - \left(\frac{1}{mk_2}\frac{1}{mk_3}\frac{1}{mk_1} + \frac{1}{mk_3}\frac{1}{mk_1}\frac{1}{mk_2}\right) > 0
\end{aligned}$$

$$\iff \frac{1}{4}\left(\frac{m-1}{m}\right)^3 \frac{1}{k_1k_2k_3} > 2\left(\frac{1}{m}\right)^3 \frac{1}{k_1k_2k_3}$$

$$\iff \frac{m-1}{m} > \frac{\sqrt[3]{8}}{m}$$

$$m > 3 \tag{4.19}$$

Avec $m > 3$, $\det(M)$ est donc une somme de termes positifs. ■

Avec les constructions introduites dans la preuve du lemme précédent, nous pouvons obtenir la borne suivante sur le $\det(F_L)$ sur un bloc b :

Corollaire 1. $\varepsilon = \frac{1}{4|L|} \implies \det(F_L) \geq \frac{1}{4^3|L|^3k_1k_2k_3}\det(\mathfrak{g})$

Démonstration. Commençons par remarquer que lorsque $a_1 = O(\varepsilon)$, on a :

$$a_1 = \frac{(i-i')\varepsilon}{k_1} \geq \frac{\varepsilon}{k_i} = \frac{1}{4|L|k_1}$$

Il en est de même pour a_2 et a_3. Étudions maintenant tous les cas possibles pour le calcul de $\det(M)$:

— $a_1 = O(\varepsilon), a_2 = O(\varepsilon), a_3 = O(\varepsilon)$
En utilisant les formes de $\det(M)$ calculées précédemment :

$$\det(M) = a_1a_2a_3 \geq \frac{1}{4^3|L|^3k_1k_2k_3} \tag{4.20}$$

— $a_1 = O(\varepsilon), a_2 = O(\varepsilon), a_3 = O(1)$

$$\det(M) = a_1a_2(a_3 - \frac{\partial\delta_3}{\partial\mathfrak{g}_3}) \geq \frac{3}{4^3|L|^2k_1k_2k_3} \tag{4.21}$$

— $a_1 = O(\varepsilon), a_2 = O(1), a_3 = O(1)$

$$\det(M) = a_1\left[\left(a_2 - \frac{\partial\delta_2}{\partial\mathfrak{g}_2}\right)\left(a_3 - \frac{\partial\delta_3}{\partial\mathfrak{g}_3}\right) - \frac{\partial\delta_3}{\partial\mathfrak{g}_2}\frac{\partial\delta_2}{\partial\mathfrak{g}_3}\right] \geq \frac{1}{4|L|k_1}\left(\frac{1}{k_2}\frac{1}{k_3} - \frac{1}{4k_2}\frac{1}{4k_3}\right)$$

$$\iff \det(M) \geq \frac{15}{4^3|L|k_1k_2k_3} \tag{4.22}$$

— $a_1 = O(1), a_2 = O(1), a_3 = O(1)$

$$\det(M) \geq \frac{1}{k_1k_2k_3}\left[\left(\frac{3}{4}\right)^3 + \frac{3}{4^3} + \frac{3}{4^3} + \frac{3}{4^3} + \frac{1}{4^3} + \frac{1}{4^3}\right]$$

$$\iff \det(M) \geq \frac{1}{k_1k_2k_3}\frac{27+9+2}{4^3} = \frac{38}{4^3}\frac{1}{k_1k_2k_3} \tag{4.23}$$

Et on peut remarquer que la pire situation arrive dans le premier cas, qui nous donne notre borne pour $\det(F_L)$. ■

4.3.4 Estimation de $E(F_L)$ et optimisation

Après avoir défini une construction d'une fonction F_L pour un ensemble donné de contraintes entières L, nous devons évaluer l'énergie $E(F_L)$ pour pouvoir résoudre notre problème d'optimisation (4.2).

Pour une valeur donnée de ε, l'évaluation de $E(F_L)$ est assez lourde. Heureusement, nous pouvons remarquer le fait suivant :

$$\lim_{\varepsilon\to 0} E(F_L) = \lim_{\varepsilon\to 0} E(h),$$

Tout simplement car, pour $\varepsilon = 0$, $F_L = h$. Étudions donc $E(h)$: la fonction h est définie indépendamment sur chacun des blocs, $E(h)$ peut donc être exprimée de la manière suivante :

$$E(h) = \sum_{b \text{ bloc de } \Omega} \int_b \|\nabla h_1(X) - \nabla \mathfrak{g}_1(X)\| + \|\nabla h_2(X) - \nabla \mathfrak{g}_2(X)\| dX$$

Avec b un bloc de Ω limité par les iso-lignes des *charts* $U_i, U_{i'}, V_j$ et $V_{j'}$ (voir la FIGURE 4.9), h_1 est défini tel que $h_1(X) = \frac{l_{U_{i'}} - l_{U_i} + (i'-i)\varepsilon}{\mathfrak{g}_{1|U_{i'}} - \mathfrak{g}_{1|U_i}} \mathfrak{g}_1(X) + C$ (avec $C \in \mathbb{R}$ une constante. voir la définition de h donnée au début de la sous-section 4.3.3). Cela nous donne le calcul suivant :

$$\begin{aligned}
\int_b \|\nabla h_1(X) - \nabla \mathfrak{g}_1(X)\| dX &= \int_b \left\| \left(\frac{l_{U_{i'}} - l_{U_i} + (i'-i)\varepsilon}{\mathfrak{g}_{1|U_{i'}} - \mathfrak{g}_{1|U_i}} - 1 \right) \nabla \mathfrak{g}_1(X) \right\| dX \\
&= \left| \frac{l_{U_{i'}} - l_{U_i} + (i'-i)\varepsilon}{\mathfrak{g}_{1|U_{i'}} - \mathfrak{g}_{1|U_i}} - 1 \right| \int_b \|\nabla \mathfrak{g}_1(X)\| \, dX \\
&= \left| \frac{l_{U_{i'}} - l_{U_i} + (i'-i)\varepsilon - (\mathfrak{g}_{1|U_{i'}} - \mathfrak{g}_{1|U_i})}{\mathfrak{g}_{1|U_{i'}} - \mathfrak{g}_{1|U_i}} \right| \int_b \|\nabla \mathfrak{g}_1(X)\| \, dX \\
&= \left| l_{U_{i'}} - l_{U_i} + (i'-i)\varepsilon - (\mathfrak{g}_{1|U_{i'}} - \mathfrak{g}_{1|U_i}) \right| \frac{\int_b \|\nabla \mathfrak{g}_1(X)\| \, dX}{|\mathfrak{g}_{1|U_{i'}} - \mathfrak{g}_{1|U_i}|}.
\end{aligned}$$

On note $\alpha_1 = \frac{\int_b \|\nabla \mathfrak{g}_1(X)\| dX}{|\mathfrak{g}_{1|U_{i'}} - \mathfrak{g}_{1|U_i}|}$ et en passant aux limites, on obtient :

$$\lim_{\varepsilon\to 0} \int_b \|\nabla h_1(X) - \nabla \mathfrak{g}_1(X)\| dX = \alpha_1 \left| l_{U_{i'}} - l_{U_i} - (\mathfrak{g}_{1|U_{i'}} - \mathfrak{g}_{1|U_i}) \right|$$

On fait le même calcul pour h_2 avec α_2 et on obtient l'équation :

$$\lim_{\varepsilon\to 0} E(F_L) = \sum_{b \text{ block of } \Omega} \begin{bmatrix} \alpha_1 \left| l_{U_{i'}} - l_{U_i} - (\mathfrak{g}_{1|U_{i'}} - \mathfrak{g}_{1|U_i}) \right| \\ +\alpha_2 \left| l_{V_{j'}} - l_{V_j} - (\mathfrak{g}_{2|V_{j'}} - \mathfrak{g}_{2|V_j}) \right| \end{bmatrix}, \tag{4.24}$$

où α_1 et α_2 sont constants par bloc :

$$\alpha_1 = \frac{\int_b \|\nabla \mathfrak{g}_1(X)\| \, dX}{|\mathfrak{g}_{1|U_{i'}} - \mathfrak{g}_{1|U_i}|} \qquad \alpha_2 = \frac{\int_b \|\nabla \mathfrak{g}_2(X)\| \, dX}{|\mathfrak{g}_{2|V_{j'}} - \mathfrak{g}_{2|V_j}|}.$$

Cela nous permet de reformuler le problème (4.2) comme suit :

$$\overline{f} = \underset{F_L:\ L \text{ respects } [\texttt{C1}] \text{ et } [\texttt{C2}]}{\arg\min} \lim_{\varepsilon\to 0} E(F_L). \tag{4.25}$$

En utilisant le changement de variable suivant, classique en optimisation linéaire :

$$\begin{cases} \min & |x_1 - x_2| \\ s.t. & x_1, x_2 \in \mathbb{R} \end{cases} \iff \begin{cases} \min & a \\ s.t. & a \geq x_1 - x_2, \\ & a \geq x_2 - x_1, \\ & x_1, x_2, a \in \mathbb{R}, \end{cases}$$

le problème d'optimisation (4.25) se résume à un programme linéaire en nombres entiers mixtes, c'est-à-dire une minimisation d'une énergie linéaire sous des contraintes linéaires et entières, un problème bien posé pour lequel il existe des logiciels de minimisation très performants, tels que CPLEX [CPLEX, 2019] ou Gurobi [Gurobi Optimization, LLC, 2022]. Pour obtenir notre carte polycube, nous minimisons l'énérgie 4.25, ce qui nous donne un jeu de valeurs entières L telles que la fonction F_L est une carte polycube, qui de plus se trouve être une très bonne approximation de $\bar{f}$, solution du problème général de quantification.

Remarque 17. *Ce problème a toujours une solution ; en effet, le choix $l_{U_i} = i$ (resp. $l_{V_j} = j$) est un ensemble de contraintes réalisables.*

4.4 Implémentation

Par souci de clarté, notre méthode a été présentée jusqu'à présent, outres les preuves, en 2 dimensions. Les propriétés des polycubes utilisés jusqu'à présent s'étendent aisément en 3D. Nous discutons de notre mise en œuvre et de nos résultats en 3 dimensions.

Il est important de noter que la construction que nous avons proposée dans la section 4.3 repose sur une bijectivité (locale) de $\mathfrak{g}$ partout dans le domaine. En pratique, c'est *beaucoup* demander : généralement, les cartes polycubes réelles ont de petits défauts qui rendent caduc notre construction, en particulier si nous voulions effectivement découper le maillage d'entrée le long des iso-surfaces générées par les cartes afin d'extraire la décomposition en blocs.

Pour contourner ce problème, nous évitons de calculer les iso-surfaces en procédant en plusieurs étapes, que nous détaillons par la suite :

- à la sous-section 4.4.1, nous extrayons la **structure combinatoire de la décomposition en blocs** en rastérisant l'image de Ω par $\mathfrak{g}$;
- à la sous-section 4.4.2, cette information suffit pour déduire la **structure combinatoire du maillage hexaédrique résultant** en résolvant le programme linéaire en nombres entiers donné par l'équation (4.25) ;
- à la sous-section 4.4.3, nous utilisons enfin F_L pour récupérer la **géométrie du maillage hexaédrique**. Notez qu'avec les informations que nous avons en main, F_L^{-1} n'est pas définie partout dans le domaine, cependant il est possible de l'évaluer aux sommets du maillage hexaédrique.

4.4.1 Extraction de la décomposition en blocs

Comme indiqué précédemment, $\mathfrak{g}$ présente souvent de petits défauts comme des tétraèdres inversés ou dégénérés. Ce n'est pas un problème tant que les *charts* restent bien définies (leur coloration est cohérente, et leur coin se trouvent à des coordonnées "valides", *i.e.* non dégénérées) et que l'image $\mathfrak{g}\left(\{U_i\} \bigcup \{V_j\} \bigcup \{W_k\}\right)$ forme le bord valide d'un volume. Dans ce cas, nous pouvons découper $\mathbb{R}^3$ avec des plans alignés sur l'axe placés aux valeurs $\mathfrak{g}_{1|U_i})$ (resp. V_j, W_k). Ceci

définit une grille alignée sur l'axe dans l'espace paramétrique, et par une simple rastérisation [18], nous pouvons trouver si une cellule courante de la grille est dans l'image de Ω par $\mathfrak{g}$, ou en dehors. Après avoir extrait tous les blocs internes, nous obtenons la structure combinatoire de la décomposition en blocs.

4.4.2 Calcul de la combinatoire des hexaèdres

La structure combinatoire du maillage hexaédrique résultant est complètement définie par la quantification L, et on peut la trouver en résolvant l'équation (4.25). En pratique, nous résolvons le programme linéaire en nombres entiers en utilisant le logiciel CPLEX [CPLEX, 2019]. Nous devons tout de même prêter attention à deux détails :

— L'énergie proposée (Eq. (4.25)) repose sur le calcul des poids α_i, calcul qui nécessite d'intégrer $\nabla\mathfrak{g}_i$ sur chaque bloc de Ω. Notez que pour des raisons de robustesse, nous avons choisi de ne pas calculer explicitement la décomposition en blocs de Ω, nous approximons donc α_i dans chaque bloc $b \in \Omega$ par le volume du bloc sous l'action de $\mathfrak{g}$, c'est-à-dire le volume du cuboïde.

— La contrainte [C2] comporte beaucoup d'inégalités ($O(N^2)$, avec N le nombre de sommets de la décomposition en blocs). Dans notre implémentation, nous les traitons comme des contraintes de façon "paresseuse". Plus précisément, nous résolvons le problème sans appliquer entièrement [C2], *i.e.* nous commençons par ne contraindre que les *charts* qui sont opposés les uns aux autres, de manière similaire à [Guo *et al.*, 2020]. Ensuite, si la solution viole [C2], nous ajoutons l'inégalité "la moins respectée" au problème (celle concernant les points les plus éloignés se retrouvant à la même position), et recommençons jusqu'à ce que nous trouvions une solution qui respecte [C2]. En pratique, nous n'avons jamais rencontré un cas où plus de quelques itérations étaient nécessaires (se référer à la colonne de droite du Tableau 4.1).

Pour calculer la structure combinatoire finale du maillage hexaédrique, nous créons d'abord le maillage hexaédrique (toujours non doté de géométrie) hérité de la décomposition en blocs. Ensuite, nous supprimons ou divisons les hexaèdres en fonction de la quantification L.

Jusqu'à présent, nous avons calculé la connectivité du maillage hexaédrique de sortie, mais il nous manque encore les positions des sommets. Dans la sous-section 4.4.3, nous allons appliquer F_L^{-1} et $\mathfrak{g}^{-1}$ pour déterminer le maillage final.

4.4.3 Calcul de la géométrie des hexaèdres

La dernière étape du processus consiste à calculer la géométrie du maillage hexaédrique final. La géométrie actuelle de notre maillage se trouve actuellement dans l'image F_L, et nous devons donc lui appliquer F_L^{-1}.

Le problème, cependant, est que nous avons choisi de calculer uniquement la structure combinatoire de la décomposition en blocs et non sa géométrie dans Ω ; par conséquent, F_L n'est pas définie partout dans le domaine avec les informations que nous avons entre les mains. Heureusement, nous pouvons étudier F_L avec un ε symbolique (variation de i), en déduire où se situent les valeurs entières dans la décomposition en blocs, et donc trouver la position de chaque sommet du maillage hexaédrique. Avant d'appliquer $\mathfrak{g}^{-1}$, nous verrouillons les sommets des bords

18. La rastérisation est un processus d'affichage d'objets vectoriels avec des pixels. Pour cela, on détermine si les objets à afficher sont, ou non, sur chacun des pixels étudiés. Nous utilisons ce processus en 3D, avec des voxels, pour déterminer l'intérieur de la décomposition en blocs, en établissant si chacun des voxels est à l'interieur, ou non, du polycuboid.

sur leurs *charts*, c'est-à-dire que nous verrouillons une dimension, et nous lissons la distorsion introduite par F_L en utilisant l'énérgie de lissage d'hexaèdres introduite dans le chapitre 3, en sous-section 3.4.2. Celle-ci permet d'améliorer la forme des éléments tout en garantissant de ne pas les inverser. Enfin, nous appliquons $\mathfrak{g}^{-1}$ aux sommets qui nous donnent la géométrie finale.

Étant donné qu'en pratique, $\mathfrak{g}$ peut présenter des imperfections locales (par exemple, des plis), le maillage résultant peut ne pas être parfaitement conforme au bord, ou posséder des éléments de mauvaise qualité. Les modèles montrés dans les figures de ce chapitre sont le résultat du processus, sans post-traitement. Notez qu'une carte parfaitement bijective n'implique pas de bons éléments, elle nécessiterait un raffinement, car nous appliquons la carte uniquement sur les sommets des hexaèdres. Pour une utilisation pratique des maillages, il serait préférable d'appliquer un post-traitement, tel que la méthode présentée par [Livesu *et al.*, 2015], pour améliorer la conformité des bords et se débarrasser des éléments de mauvaise qualité.

4.5 Résultats

Nous avons testé notre algorithme sur un processeur i7 4.3GHz avec 64GB de RAM sur une implémentation mono-thread. Nous calculons la carte polycuboid initiale $\mathfrak{g}$ avec un algorithme comparable à [Gregson *et al.*, 2011]. Nous résolvons le problème d'optimisation de la Section 4.3.4 en utilisant CPLEX Optimizer [CPLEX, 2019].

4.5.1 Maillage hexaédrique

Nous utilisons notre algorithme pour générer des maillages hexaédriques minimisant la distorsion par rapport à la carte initiale des polycubes réels $\mathfrak{g}$. La résolution des hexaèdres est déterminée par l'échelle de la carte de polycubes réels d'entrée $\mathfrak{g}$. La FIGURE 4.14 montre des résultats pour deux résolutions différentes. Contrairement à la technique naïve d'arrondi parfois utilisée pour la quantification (FIGURE 4.2), notre quantification produit des maillages hexaédriques valides (aucun hexaèdre manquant) sans perte de caractéristiques géométriques. Notre extraction de la décomposition en blocs repose toujours sur une coloration non dégénérée. Nous avons exécuté notre code sur un sous-ensemble du jeu de données [Hu *et al.*, 2018], composé de plus de 1000 modèles où une coloration de qualité suffisante a pu être extraite. En ce qui concerne nos affirmations sur la robustesse à la non-bijectivité locale, il faut noter que **des 1391 modèles testés, 803 (58%) avaient au moins un tétraèdre retourné** par $\mathfrak{g}$. Plus généralement, les maillages ont pris en moyenne 2,5 sec. à traiter, et on nécessitait respectivement 1,25 et 1,04 appels de CPLEX en moyenne pour des maillages grossiers et fins. Dans certains cas, une partie de la géométrie peut être perdue, mais cela est dû à la mauvaise qualité de la carte initiale $\mathfrak{g}$.

4.5.2 Décomposition en blocs aussi grossière que possible

Pour tester la robustesse de notre algorithme, nous générons des maillages hexaédriques aussi grossiers que possible en utilisant un $\mathfrak{g}$ à échelle réduite, c'est-à-dire en remplaçant le $\mathfrak{g}$ d'entrée par $\eta\mathfrak{g}$ avec $\eta \approx 0$. Le résultat peut être interprété comme une décomposition en blocs qui minimise la distance entre les paires de *charts*. Le tableau 4.1 fournit des statistiques sur 7 modèles, les décompositions sont présentées dans la FIGURE 4.15. Notez que $\mathfrak{g}$ a été calculé avec [Gregson *et al.*, 2011] pour tous les modèles sauf les deux premiers modèles du tableau 4.1 qui proviennent du matériel supplémentaire de [Livesu *et al.*, 2013].

Nous avons reporté le nombre de blocs dans le polycuboid d'entrée, le nombre de blocs dans la sortie générée ainsi que les timings et le nombre de fois où nous avons dû appeler CPLEX.

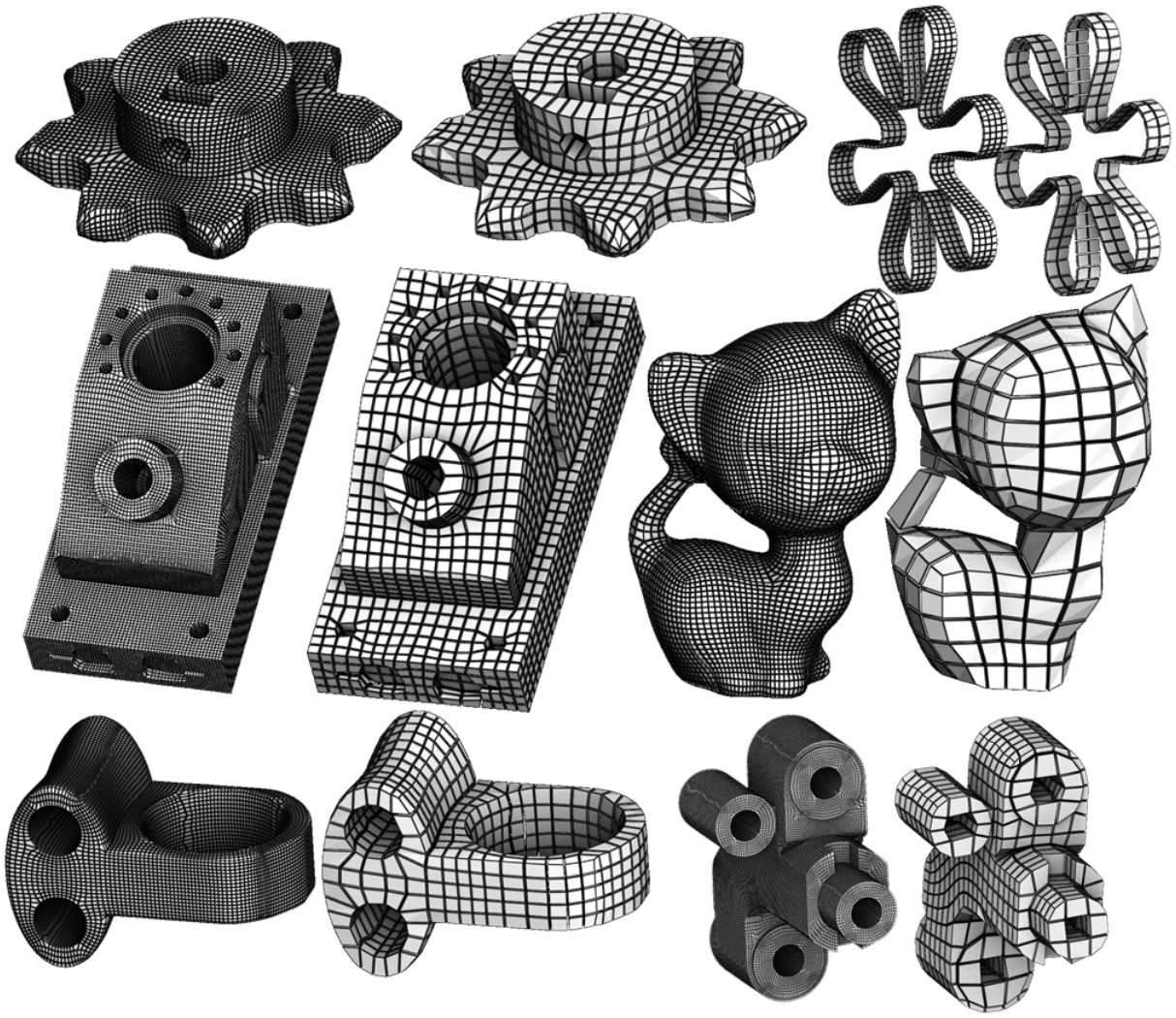


FIGURE 4.14 – Notre processus de quantification robuste permet d'obtenir des maillages à plusieurs résolutions, sans risque de perte de détails topologique.

Id	Nom	taille (tets)	# blocs	# hexaèdres	temps total (sec)	appels CPLEX
1	kiss	302292	2558	410	150	2
2	Bunny	731085	426	58	130	2
3	grid	6718	285	56	2.1	1
4	street	2327	483	55	3.0	3
5	connecting rod	14380	203	54	2.6	1
6	engine1	18864	6151	219	4.6	1
7	engine2	56789	9415	404	9.2	1

TABLE 4.1 – Statistiques pour les maillages montrés sur la FIGURE 4.15.

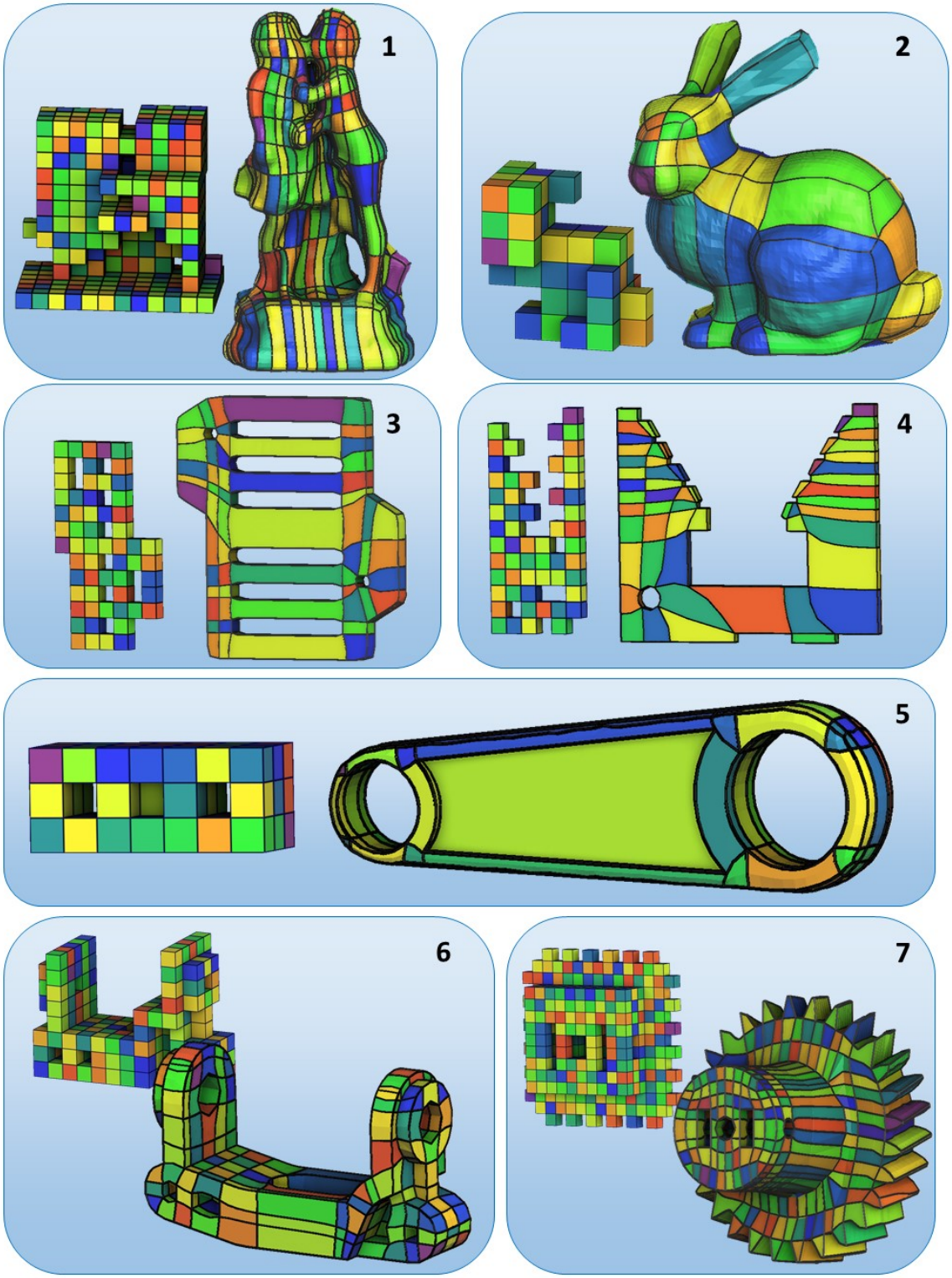


FIGURE 4.15 – Pour chaque modèle, nous optimisons la décomposition des blocs pour obtenir un polycube aussi grossier que possible. Le maillage associé donnera une décomposition très grossière du domaine original Ω.

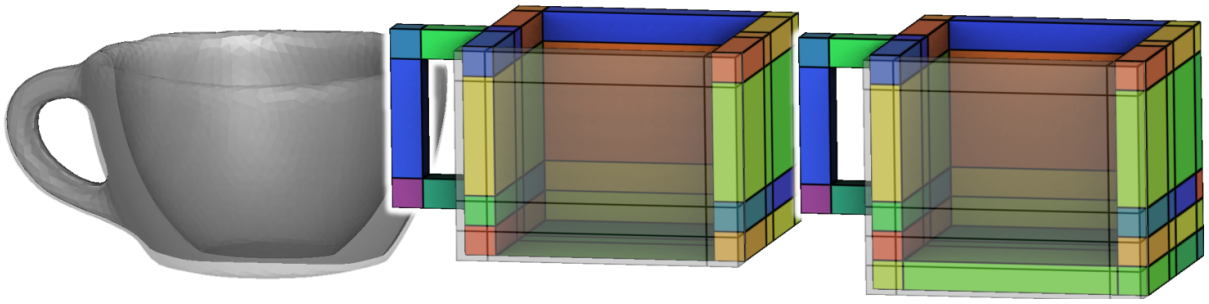


FIGURE 4.16 – La quantification sans contraintes volumiques [Zhao *et al.*, 2019, Chen *et al.*, 2019] ne peut pas détecter l'effondrement du fond de la tasse (au milieu). Nous les prenons en compte (à droite). [Guo *et al.*, 2020] réussirait sur ce modèle, mais échouerait sur des exemples plus délicats (comme ceux de la FIGURE 4.7).

On peut voir que nous obtenons une simplification significative de la structure du maillage, et que notre algorithme est robuste aux simplifications extrêmes, étant capable de récupérer des maillages valides même pour des cartes polycubes fortement déformées.

4.5.3 Comparaisons

Comparaison avec les méthodes de simplification des polycubes

[Zhao *et al.*, 2019], [Chen *et al.*, 2019] et [Guo *et al.*, 2020] proposent un ensemble de contraintes pour garantir la validité du polycube. Le problème, cependant, est que leurs contraintes ne sont pas suffisantes et peuvent potentiellement conduire à des polycubes dégénérés lors de la simplification. Elles ne garantissent que l'intégrité du bord du polycube, sans tenir compte de la déformation à l'intérieur du domaine. La FIGURE 4.16 montre un exemple d'échec : le fond de la tasse s'effondre dans un plan sans violer leurs contraintes, alors que notre ensemble de contraintes garantit la validité de la carte polycube. Cherchi *et al.* [Cherchi *et al.*, 2016] tentent d'aligner localement les coins du polycube. En conséquence, ils ne parviennent pas à aligner des sommets très éloignés les uns des autres. Notre méthode, au contraire, considère tous les alignements de cartes possibles. Par exemple, pour le modèle 4 de la FIGURE 4.15, nous sommes en mesure d'aligner le trou avec l'une des dents du haut, ce qui ne peut pas être fait par [Cherchi *et al.*, 2016].

Comparaison avec [Gao *et al.*, 2017b].

Une autre façon de calculer une décomposition en blocs minimale consiste à calculer un maillage hexaédrique fin et à le décimer à l'aide d'un algorithme générique de simplification de maillages. La méthode de pointe [Gao *et al.*, 2017b] réduit récursivement les feuillets et les cordes du maillage jusqu'à ce qu'aucune autre simplification ne soit possible. Dans leur algorithme, chaque opération se voit attribuer un score et les opérations sont effectuées de manière gloutonne. Cette approche est strictement locale et cette séquence de simplifications ne permet pas toujours d'obtenir le maillage le plus grossier, comme l'illustre la FIGURE 4.17. En fait, notre problème d'optimisation *global* conduit systématiquement à une décomposition en blocs plus grossière ou équivalente, Il est à noter que dans [Gao *et al.*, 2017b] la géométrie du maillage est recalculée à chaque étape pour prendre en compte l'opération de simplification. Dans notre méthode, les contraintes assurent la validité de la carte polycube à tout moment, de sorte que

	# de blocs			Temps (min)	
	Original	GAO et al. 2017	Us	GAO et al. 2017	Us
haut	33	23	28	27	<1
Milieu	347	80	80	>200	<1
Bas	70	24	16	24	<1

TABLE 4.2 – Comparaison avec la méthode de simplification de maillage [Gao *et al.*, 2017b] : nombre de blocs avant et après simplification et timing. Les modèles sont montrés sur la FIGURE 4.17. Notre méthode conduit à un nombre similaire de blocs avec une accélération considérable ($\sim 20\times$). Pour le modèle "haut", Gao *et al.* a moins de blocs uniquement parce qu'il permet une déformation non polycube (gros plan).

la géométrie du maillage peut être calculée une fois à la toute fin du processus. Ceci conduit à l'accélération considérable rapportée dans le Tableau 4.2. Enfin, la simplification des cordes introduit de nouvelles singularités qui ne sont pas toujours souhaitables. Par exemple, dans la FIGURE 4.17 du milieu, Gao *et al.* crée sur le bord du maillage des sommets de valence 4 près de sommets de valence 2, ce qui conduit à générer des cellules fortement déformées.

Conclusion

Ce travail fournit une importante garantie de robustesse, mais il suppose qu'une déformation $\mathfrak{g}$ bijectif est donné en entrée. Pour obtenir une méthode de remaillage hexaédrique entièrement robuste basée sur les polycubes, la principale difficulté restante est de résoudre le problème de coloration. De plus, l'ensemble des maillages pouvant être obtenu est encore restreint avec notre approximation. L'approche utilisant le T-mesh pourrait repousser de cette limitation.

Elle ouvre également des opportunités de recherche intéressantes pour le cas général du remaillage hexaédrique basé sur des paramétrisations globales. Fondamentalement, nous devons introduire de nouvelles variables entières correspondant aux discontinuités (préservant la grille) dans la carte. Le principal problème est que la décomposition en blocs est moins simple à obtenir : le domaine paramétrique devient une couverture universelle où l'intersection d'un plan avec le modèle peut être infinie, et le tracer dans le domaine géométrique est un problème mal posé [Kowalski *et al.*, 2014, Kowalski *et al.*, 2016].

Nous avons également vu que la structure combinatoire du maillage hexaédrique peut être dérivée de nos variables entières. Des travaux futures pourront essayer d'étendre cela au cas plus général des paramétrisations globales.

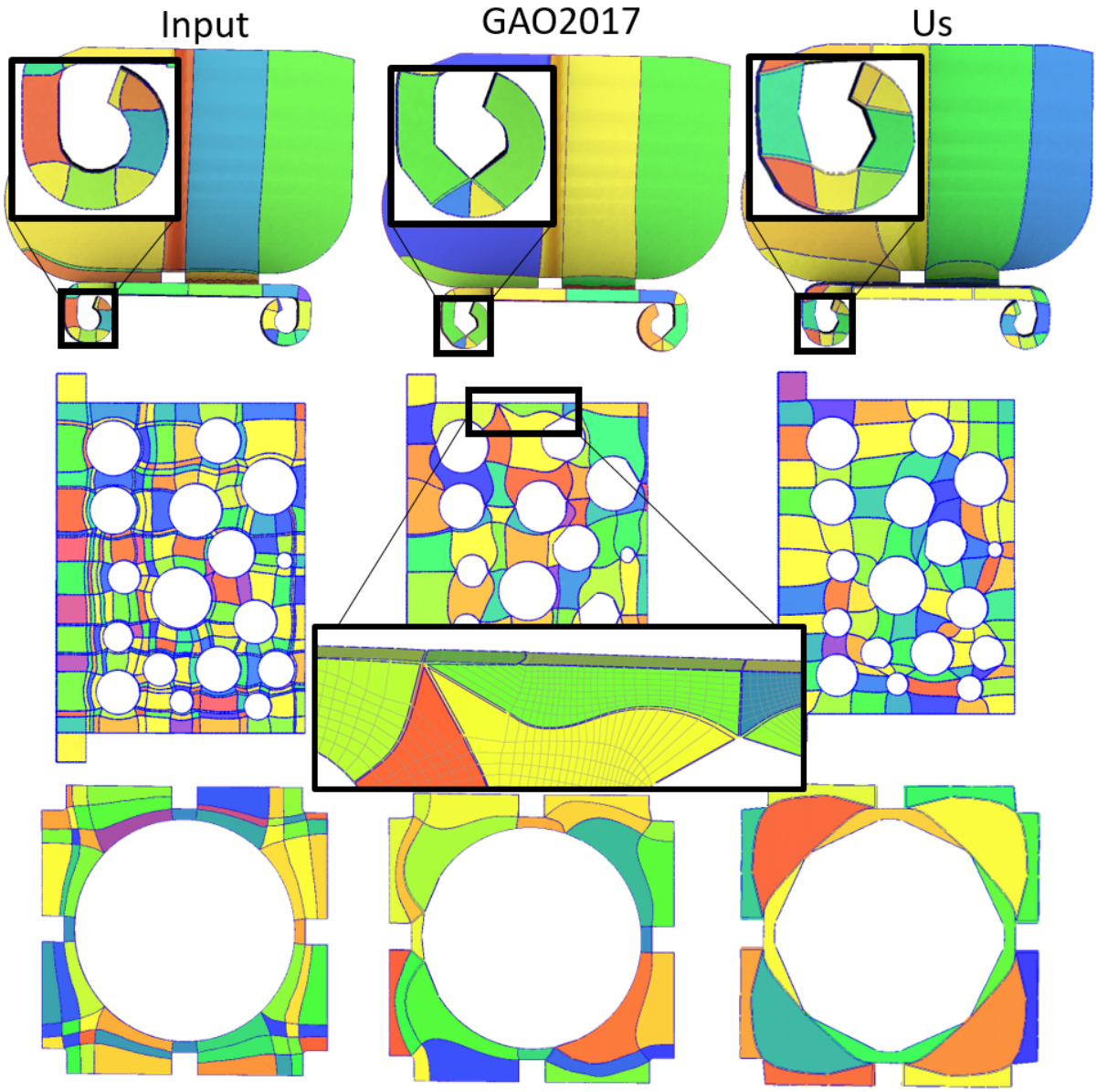


FIGURE 4.17 – Comparaison visuelle entre notre méthode et la méthode de simplification du maillage [Gao *et al.*, 2017b]. Les données quantitatives sont reportées dans le tableau 4.2. Notre décomposition en blocs crée des cellules de meilleure qualité lorsque Gao *et al.* crée des sommets de valence 4 sur le bord du maillage (gros plan au milieu).

Chapitre 5

Gestion des contraintes sur les bords

La méthode de génération de maillages hexaédriques à l'aide de Polycubes présentée jusque-là peut être considérée comme une combinaison de trois taches : colorer, déformer et extraire. Si nous avons obtenu une bonne déformation, le chapitre précédent nous donne une procédure robuste pour en extraire des hexaèdres. De la même façon, la boîte à outil développée dans le chapitre 3 permet d'efficacement calculer de bonnes déformations si, bien sûr, nous avons une coloration qui y est adaptée.

Trouver une telle coloration, « adaptée » à une déformation en polycube, est un problème difficile, comme détaillé par [Sokolov et Ray, 2015]. Une approche pour générer une coloration « valide », qui permet d'obtenir de bonnes déformations sur tout type de modèle est un sujet central des premiers travaux [Gregson *et al.*, 2011, Livesu *et al.*, 2013] utilisant des polycubes pour le maillage hexaédrique, et qui est toujours présent dans les derniers articles parus [Yang *et al.*, 2019, Guo *et al.*, 2020]. Dans ce chapitre, nous présentons nos tentatives pour obtenir une méthode, focalisée sur la coloration, nous permettant de rendre la procédure de génération de maillage via des polycubes robuste de bout en bout. Contrairement aux chapitres précédents, nous proposons pas ici de méthode permettant de résoudre le problème. Nous avons tout de même étudié principalement 2 approches, qui ont permis de soulever des questions intéressantes, qui ont donné lieu à une publication :
Desobry, D., Protais, F., Ray, N., Corman, E. et Sokolov, D. (2021). Frame fields for CAD models. *In International Symposium on Visual Computing*, pages 421–434. Springer
et un code publié open-source :
https://github.com/fprotais/marchinghex
Et avec cela, j'ai collaboré sur un travail soumis à *Computer Graphics Forum* :
Dumery, C., Protais, F., Mestrallet, S., Bourcier, C. et Ledoux, F. (2022). Evocube: a genetic labeling framework for polycube-maps. *arXiv preprint arXiv:2205.00738*

Le chapitre est structuré de la façon suivante : nous discutons d'abord de la problématique de la coloration et des essais effectués. Nous présentons ensuite les travaux qui en ont émergé : dans un premier temps la génération de champs de croix surfaciques pour les modèles de CAO et enfin la méthode *marchinghex* pour l'extraction de maillages.

5.1 Problématique

Cette section présente les axes étudiés pour débloquer le « verrou technologique » qu'est actuellement la coloration pour les polycubes. Malheureusement, nos travaux n'ont pas permis de dégager une approche claire qui permet d'arriver à nos fins. Ils ont tout de même permis de

développer des outils ayant des applications orthogonales, que nous présentons dans les sections suivantes. Nous avons commencé par essayer d'améliorer la coloration en étudiant plus précisément le bord des objets, nous détaillons cela dans la sous-section 5.1.1, et cela a donné lieu au travail présenté dans la section 5.2. Nous avons donc ensuite essayé de relaxer les contraintes de bord, comme présenté dans la sous-section 5.1.2, qui a nécessité le développement de la méthode que nous présentons dans la section 5.3. Nos études n'ayant pas abouti en des méthodes fiables, nous présenterons particulièrement les problématiques et les difficultés rencontrées, qui nous ont menés aux approches développées dans les sections suivantes.

Toujours sur le sujet de la coloration, j'ai collaboré à l'utilisation d'algorithmes génétiques pour la coloration qui a donné lieu à la soumission de l'article [Dumery *et al.*, 2022]. Mon apport sur ce projet était principalement l'utilisation des approches que j'ai développées dans les chapitres précédents, une fois la coloration obtenue. Le contenu du papier n'est donc pas présenté dans cette thèse. Cela permet en revanche de confirmer le constat : l'unique point bloquant pour une génération robuste de maillages hexaédriques avec des polycubes est bien la coloration. En effet, dans les nombreux exemples où nous avons testé l'approche, une coloration valide se transforme systématiquement en un maillage de bonne qualité.

5.1.1 Améliorer la coloration

Effectuer une coloration « naïve » sur une surface triangulée, c.-à-d. colorer chaque triangle selon l'axe le plus proche de sa normale extérieure, produit 3 types de problèmes : le bruit, les incohérences locales, et les incohérences globales. Le bruit peut être assez facilement filtré. Pour cela [Livesu *et al.*, 2013] propose d'utiliser une approche de graph-cut où la couleur d'un triangle est aussi conditionnée à celle de ses voisins. Cette approche permet de systématiquement obtenir des colorations sans bruit, et est, *de facto*, le point de départ pour une coloration dans la littérature. Il reste donc à étudier les deux problèmes d'incohérence.

Les problèmes d'incohérence locale se caractérisent par certaines arêtes entre zones de couleurs différentes qui ne trouveraient pas leur place sur un polycube. Certains de ces cas sont illustrés à la Figure 5.1. Les problèmes d'incohérence globale sont moins intuitifs, ils ont été soulignés par [Sokolov et Ray, 2015]. Ils interviennent lorsque la coloration ne peut pas représenter la surface d'un polycube, à cause d'un agencement global particulier des couleurs. La Figure 5.2 illustre ce problème. Dans ce cas, le lacet vert va être contraint sur une seule valeur dans une direction, ce qui fait que sa composante sur la zone rouge se trouvera écrasée, ce qui empêche la construction d'une déformation en polycube bijective.

Sokolov et Ray dressent un constat : si la coloration initiale semble suffisamment bonne, il est possible de détecter les incohérences, globales et locales, et en plus de cela, l'insertion de « marches », illustrées en Figure 5.3, permet de les supprimer. En revanche, ils restent incapables de donner des contraintes nécessaires réalistes sur la coloration initiale. Le manque de spécification sur une coloration initiale est actuellement ce qui bloque la création de méthodes de coloration robustes. Nous pouvons tout de même nous pencher sur la question de la création de marche : Sokolov et Ray proposent une approche reposant sur le tracé de lignes, pour lequel un champ de croix est nécessaire. Pour que cette approche fonctionne efficacement, il faut un champ pertinent, tout particulièrement pour les modèles CAO. A noter également, ce tracé de lignes est aussi grandement simplifié si l'on travaille sur un maillage quadrangulaire, qui va directement « résumer » le champ de croix. Toutes ces remarques nous ont poussés à chercher des champs de croix de qualité pour des modèles de CAO, ce que nous détaillons à la section 5.2.

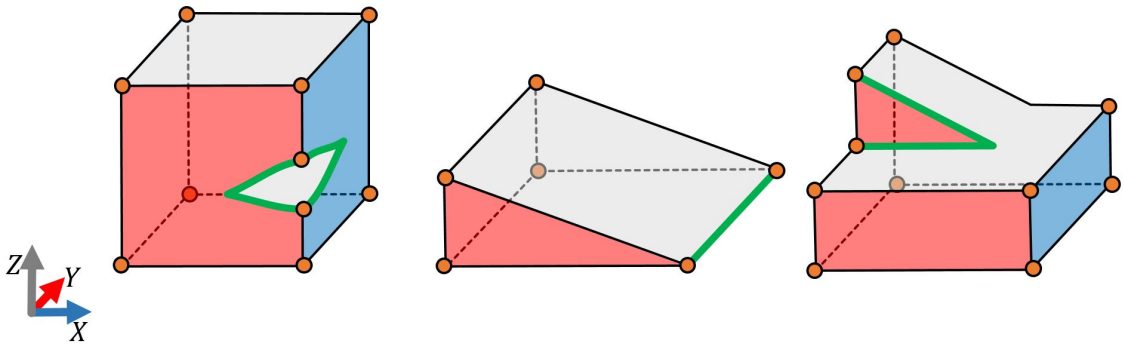


FIGURE 5.1 – Problèmes d'incohérence locale dans les colorations. Les arêtes vertes sont incompatibles avec des déformations bijectives en Polycube.

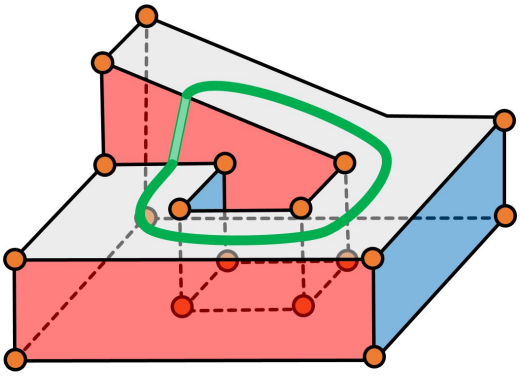


FIGURE 5.2 – Problème d'incohérence globale dans une coloration. La coloration contraint l'ensemble du lacet vert à avoir la même hauteur, ce qui aura pour effet d'écraser sa composante sur la zone de couleur rouge.

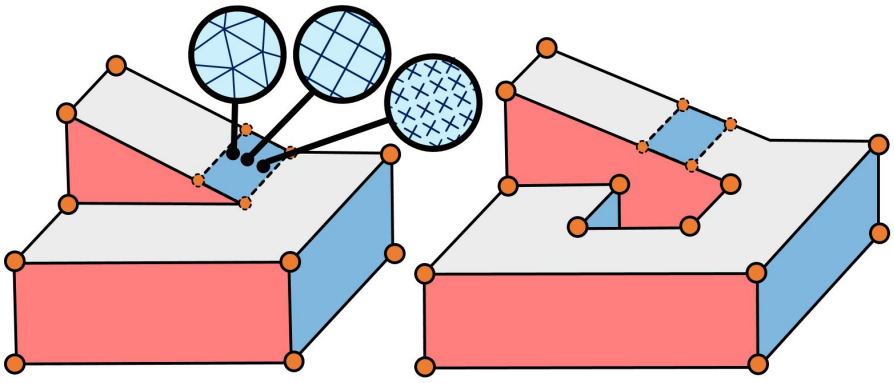


FIGURE 5.3 – L'insertion de marches peut éviter les problèmes d'incohérences. Il faut en revanche être capable de tracer de telles marches, ce qui demande une bonne compréhension des domaines étudiés. Tout particulièrement pour obtenir un maillage final de bonne qualité.

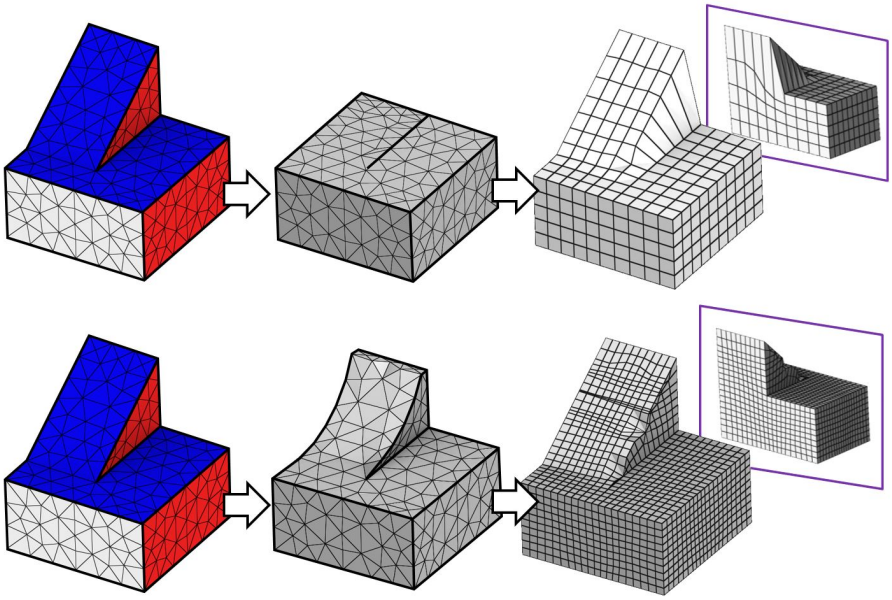


FIGURE 5.4 – Pour éviter les problèmes issus des mauvaises colorations (Ligne du haut), une possibilité est de ne pas effectuer la déformation complètement, et d'ensuite utiliser une méthode d'extraction adaptée aux déformations partielles (Ligne du bas).

5.1.2 Relaxer la coloration

Une approche commune [Gregson *et al.*, 2011, Guo *et al.*, 2020] pour la génération de coloration est de progressivement déformer le domaine étudié vers un polycube. Cela permet de surligner les plans, et d'isoler les lieux problématiques, où la déformation ne converge pas. Dans cette situation, l'approche habituelle est d'essayer de résoudre ces lieux problématiques pour obtenir le polycube recherché. Nous pouvons procéder différemment. Si la déformation initiale est bien avancée, elle contient presque toutes les composantes du polycube, et avec cela un ensemble de lieux problématiques. Nous pouvons directement essayer d'extraire un maillage de cet état. Les endroits où la déformation s'est bien passée vont donner un maillage de qualité, et il restera les endroits problématiques, pour lesquels nous devons construire une méthode robuste d'extraction de maillages. Nous illustrons cette approche dans la FIGURE 5.4. Dans ce but, nous avons développé le logiciel *marchinghex*, que nous présentons à la section 5.3.

Une des grandes qualités de cette approche est qu'il est immédiat de l'étendre aux paramétrisations globales. Elle nous semble donc constituer une bonne candidate pour les recherches futures. Son défaut principal est que ces déformations « incomplètes » restent très techniques à obtenir, surtout avec des déformations de bonne qualité partout. En effet, les heuristiques actuelles ont tendance à écraser les lieux problématiques (quasiment), laissant très peu de liberté à *marchinghex* pour en extraire un maillage de bonne qualité, comme on peut voir à la base de la pente dans la FIGURE 5.4.

5.2 Champs de croix pour les modèles CAO [Desobry *et al.*, 2021]

Dans cette section, nous détaillons le travail présenté à ISVC : *Frame fields for CAD models* [Desobry *et al.*, 2021]. L'origine de ce projet est la volonté de mieux pouvoir étudier le bord

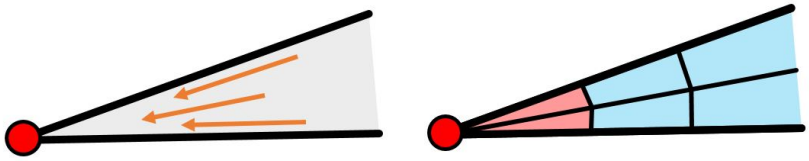


FIGURE 5.5 – Si deux arêtes caractéristiques produisent une d'angle faible, un champ de croix classique s'alignera sur les deux arêtes comme s'ils étaient parallèles. Cela produit une singularité 1/2 qui ne peut pas être maillée avec des quadrilatères valides.

des domaines 3d pour en déduire des contraintes pour les méthodes de Polycube, et plus généralement les paramétrisations globales. Pour cela, nous étudions la génération de champs de croix surfaciques spécialement pour les modèles de CAO, et par extension les maillages quadrangulaires que l'on peut en extraire.

5.2.1 Introduction

La génération de maillage quadrangulaire basée sur l'utilisation de champs de croix se déroule généralement en deux étapes : d'abord, un champ de croix est calculé, puis une paramétrisation représentant les quadrilatères est extraite. Le champ de croix définit l'orientation des quadrilatères en chaque point du domaine, tandis que l'étape de paramétrisation détermine les positions des sommets des quadrilatères. Les champs de croix sont donc un résultat intermédiaire dans le pipeline de génération de maillages quadrangulaires.

Un maillage quadrangulaire de haute qualité présente généralement une faible distorsion (éléments proches d'un carré) et peu de sommets singuliers (valence $\neq 4$). Les champs de croix sont produits par une optimisation numérique qui maximise la régularité du champ, empêchant naturellement la création de singularités : à proximité d'un point singulier, un champ présente une forte courbure qui est pénalisée par l'optimisation.

Les modèles de CAO sont des surfaces particulières, car elles possèdent un réseau d'arêtes caractéristiques qui doivent être préservées pendant le processus de maillage. Ainsi, une courbe caractéristique du maillage triangulaire d'entrée doit se retrouver dans le maillage quadrangulaire de sortie. Malheureusement, ces courbes peuvent se rencontrer avec un angle aigu, ce qui rend impossible l'insertion d'un quadrilatère parfait dans un tel angle, comme l'illustre la FIGURE 5.5. En conséquence, le champ de croix de guidage calculé comme étape intermédiaire ne peut pas être transformé en un maillage quadrangulaire et la méthode échoue. Les cas d'échec typiques sont illustrés sur la FIGURE 5.12.

Ces angles aigus problématiques sont omniprésents dans les modèles de CAO, car ils apparaissent naturellement à partir de filets et de chanfreins – des conceptions courantes pour les pièces mécaniques. Dans l'ensemble de données ABC [Koch *et al.*, 2019] (une collection d'un million de modèles de conception assistée par ordinateur (CAO)), nous avons constaté que plus de 65% des modèles présentent des angles vifs potentiellement problématiques. Dans Thingi10k [Zhou et Jacobson, 2016], qui n'est pas spécialisé dans les modèles de CAO, environ 35% de l'ensemble de données présentent au moins un angle vif. C'est pourquoi la non-orthogonalité et la déformation métrique ont récemment fait l'objet d'études [Panozzo *et al.*, 2014, Jiang *et al.*, 2015]. Dans cette section, nous proposons une méthode plus simple et beaucoup plus rapide que les travaux précédents pour traiter ces cas problématiques.

Contribution

Notre solution produit automatiquement un champ de croix orthogonales de haute qualité aligné sur les courbes caractéristiques dont la topologie est compatible avec des maillages quadrangulaires sur les modèles de CAO. Nous relaxons ensuite l'orthogonalité des croix pour nous adapter encore mieux aux configurations qui le nécessitent.

Ces deux étapes sont réalisées en résolvant deux systèmes linéaires, de sorte que l'impact sur les performances par rapport à la génération standard de champs de croix orthogonales est très faible. Des travaux antérieurs [Jiang *et al.*, 2015, Fang *et al.*, 2018, Iarussi *et al.*, 2015] fournissent quelques solutions pour résoudre ces problèmes, mais elles sont beaucoup plus difficiles à mettre en œuvre, nécessitent des optimisations non linéaires qui ont un impact important sur les performances, et un réglage fin des paramètres pour obtenir des résultats proches des nôtres.

Le reste de la section est organisé comme suit : après une revue des travaux similaires, nous formalisons le problème à la sous-section 5.2.3, rappelons le contexte mathématique de la génération de champs de croix orthogonales § 5.2.4, et présentons notre première contribution pour éviter les champs dégénérés à la sous-section 5.2.4. Dans la sous-section 5.2.5, nous optimisons davantage la courbure du champ en relaxant son orthogonalité. Nous évaluons la qualité de nos champs dans la sous-section 5.2.6 pour produire des maillages quadrangulaires.

5.2.2 Travaux similaires

La génération de champs de croix a été un domaine de recherche actif au cours de la dernière décennie [Vaxman *et al.*, 2017]. Ils ont été introduits en infographie pour placer des hachures dans des rendus non photoréalistes [Hertzmann et Zorin, 2000]. Cependant, l'application principale est le maillage quadrangulaire par paramétrisation globale qui a été découverte quelques années plus tard [Ray *et al.*, 2006], et améliorée dans [Bommes *et al.*, 2009, Kaelberer *et al.*, 2007].

Les champs de croix sont généralement produits par optimisation numérique. L'objectif est de maximiser la régularité du champ, avec quelques modifications mineures dépendant de l'application, comme suivre la courbure de la surface [Ray *et al.*, 2006], ou mieux représenter un champ 3D projeté en 2D [Iarussi *et al.*, 2015]. La plupart des travaux contribuent à mieux optimiser le lissage ou à ajouter des contraintes géométriques et topologiques. Le principal défi de la génération de champs de courbure vient de la topologie du champ (principalement la position et le type de singularités du champ), qui nécessite l'ajout de contraintes entières dans le problème d'optimisation.

La première solution consistait à utiliser des fonctions périodiques où la topologie du champ était "encodée" dans le module de la fonction. L'idée vient de l'observation que si les $k^{\text{ième}}$ directions d'une croix font un angle de $\alpha + k\pi/2$ avec un vecteur fixe, alors $\cos(4(\alpha + k\pi/2))$ est le même pour toutes les directions. Par conséquent, le codage de Hertzmann *et al.* [Hertzmann et Zorin, 2000] représente une croix par un cosinus et minimise la norme L^2 entre les échantillons adjacents. Il est cependant plus efficace d'optimiser le champ avec le sinus et le cosinus [Ray *et al.*, 2006], que l'on appelle généralement vecteur de représentation ou représentation complexe. Le problème lorsque l'on travaille avec ces variables est de préserver la contrainte selon laquelle $\cos^2(4\alpha) + \sin^2(4\alpha) = 1$, ce qui rend le problème hautement non linéaire. Forcer la norme du champ de croix [Knöppel *et al.*, 2013, Beaufort *et al.*, 2017] améliore les résultats, mais impacte sérieusement les performances.

Dans certains cas, il est souhaitable de contrôler la topologie du champ. Par exemple, pour réduire le nombre de singularités, changer leurs types, ou imposer le comportement du champ le long de boucles non contractiles. Le contrôle complet de la topologie du champ peut être obtenu

directement [Ray *et al.*, 2008], ou par modification d'un champ existant [Palacios et Zhang, 2007]. Le problème est que la topologie est définie par un grand ensemble d'entiers (un peu plus que l'indice de chaque point de la surface qui peut être une singularité), qui sont très difficiles à fixer. Dans [Bommes *et al.*, 2009, Iarussi *et al.*, 2015], ces variables entières sont automatiquement optimisées, mais pourraient probablement être contraintes pour forcer certains des degrés de liberté topologiques.

En pratique, la topologie d'un champ généré automatiquement n'est pas toujours la meilleure, mais il est difficile de choisir une topologie avant de générer le champ de croix. On peut fixer la topologie d'un champ de croix existant [Palacios et Zhang, 2007] en post-traitement. Une autre possibilité consiste à modifier le problème d'optimisation de manière à favoriser une courbure localement élevée du champ et à éviter de produire des singularités sur les détails géométriques. Notre méthode s'inspire de ce travail pour contrôler l'impact des angles aigus.

Les champs de croix non orthogonales ont été introduits dans [Liu *et al.*, 2011] pour générer des maillages quadrangulaires planaires, où les directions doivent être conjuguées au lieu d'être orthogonales. Le changement de la métrique de la surface [Panozzo *et al.*, 2014, Jiang *et al.*, 2015] étire naturellement les croix, qui perdent leur orthogonalité. Cependant, en définissant la métrique avant les contraintes des croix, il est difficile de soutenir l'alignement avec les arêtes caractéristiques, et l'optimisation devient plus difficile elle aussi. Représenter le champ comme racine d'un polynôme complexe [Diamanti *et al.*, 2014, Diamanti *et al.*, 2015] prend également en charge les croix non orthogonales, mais introduit des problèmes d'optimisation non linéaires et nécessite un réglage fin des paramètres. Dans [Iarussi *et al.*, 2015], le champ de croix 2D est la projection d'un champ de croix défini sur une surface, conduisant à des champs de croix non orthogonales, obtenus en minimisant une fonction objectif différente. Il devrait être possible de l'adapter à notre cas, mais cela fait dépendre la solution d'une implémentation non triviale de leur solveur en nombres entiers. Nous résolvons plutôt la topologie avec des croix orthogonales, puis nous optimisons sa géométrie en relaxant l'orthogonalité. Il devrait être possible de produire des champs de croix similaires aux nôtres en utilisant ces méthodes, mais cela nécessite d'adapter un cadre d'optimisation complexe, alors que notre méthode ne nécessite que la résolution de quelques systèmes linéaires. De plus, nous supportons des distorsions extrêmes sur des angles aigus qui mettraient les solveurs en difficulté.

5.2.3 Formalisation et aperçu de notre approche

Notre pipeline prend en entrée un maillage triangulaire avec des arêtes caractéristiques identifiées, et calcule un champ de croix constant non-orthogonal par triangle aligné avec les arêtes caractéristiques. L'objectif est de calculer un champ de croix avec la rotation totale la plus faible ; le champ ne doit pas avoir de singularité d'indice 1/2 aux angles aigus. En pratique, nous relâchons l'orthogonalité du champ à la toute dernière étape, c'est pourquoi nous donnons dans cette section des définitions formelles de la rotation totale et des indices de singularité pour les champs orthogonaux et non orthogonaux.

N.B. Notre champ de croix sera représenté sur les faces d'une surface triangulée. Cela rend les croix indépendantes sur les arêtes caractéristiques, nous pouvons donc couper le maillage le long des arêtes caractéristiques sans perte de généralité. À partir de maintenant, **les arêtes caractéristiques seront considérées comme des bords** ; cela simplifie considérablement les notations.

Champs de croix orthogonales : Une croix sur un triangle est constituée d'un ensemble de quatre vecteurs unitaires tangents $\{v, v^{\perp}, -v, -v^{\perp}\}$. Il est courant de doter chaque triangle i

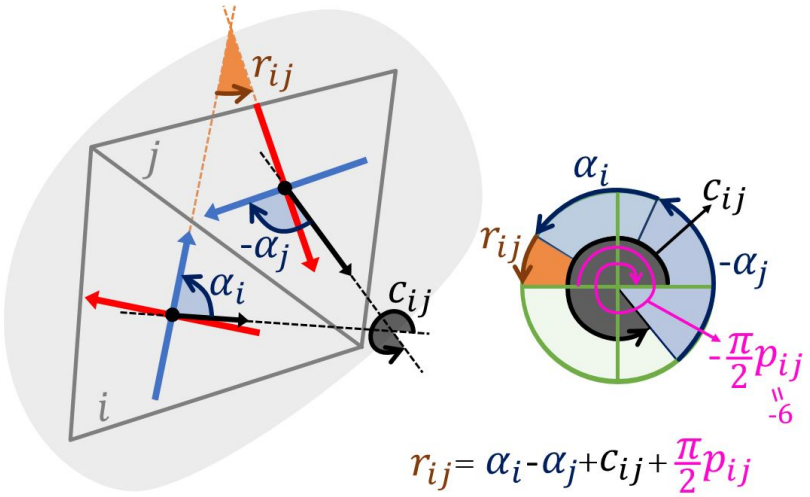


FIGURE 5.6 – Les valeurs (α_i) sont les angles de la croix, c_{ij} est la rotation entre les vecteurs de référence des triangles i et j. L'entier p_{ij} désambiguïse le modulo $\pi/2$.

d'un vecteur de référence b_i et de décrire la croix comme un angle de rotation α_i du vecteur de référence b_i autour de la normale. Ainsi, un champ de croix peut être représenté par l'ensemble des angles $\{\alpha_i, \alpha_i + \pi/2, \alpha_i + 2\pi/2, \alpha_i + 3\pi/2\}$.

La rotation d'un champ de croix orthogonales est généralement mesurée par la rotation r_{ij} entre les croix de deux triangles adjacents i, j :

$$r_{ij} = -\alpha_i + \alpha_j + c_{ij} + p_{ij}\pi/2, \tag{5.1}$$

où c_{ij} représente le changement (une fois les deux triangles pivotés dans un plan commun) entre les vecteurs de référence b_i et b_j, donc $-\alpha_i + \alpha_j + c_{ij}$ est la rotation entre deux croix adjacentes. Puisque la croix est invariante sous une rotation de $\pi/2$, l'entier p_{ij} reflète la correspondance entre les branches des croix (voir FIGURE 5.6). Dans la plupart des cas, p_{ij} est choisi pour minimiser la rotation.

Le problème de la génération du champ de croix orthogonales le plus lisse peut être formulé comme la minimisation de la rotation totale du champ :

$$\arg\min \sum_{ij} r_{ij}^2 \quad \text{sujet à des contraintes de bords.} \tag{5.2}$$

La dernière chose que nous devons définir pour les champs de croix orthogonales est l'indice du champ de croix. Pour un sommet v, nous définissons l'indice comme suit :

$$\mathcal{I}(v) := \frac{1}{2\pi}\left(\sum_{ij \in E(v)} r_{ij} - \theta_v\right) + 1 - \frac{\mathbb{1}_{\partial\Omega}(v)}{2}, \tag{5.3}$$

où $\mathbb{1}_{\partial\Omega}(v) \in \{0, 1\}$ est la variable indicatrice nous indiquant s'il s'agit d'un sommet de frontière ou non, $E(v)$ désigne l'ensemble des arêtes duales sur l'anneau 1 du sommet v, et θ_v est la somme de tous les angles autour du sommet (tous les angles de triangle incidents au sommet). Les indices les plus courants sont illustrés sur la FIGURE 5.7, ils suivent la convention utilisée dans [Liu *et al.*, 2018].

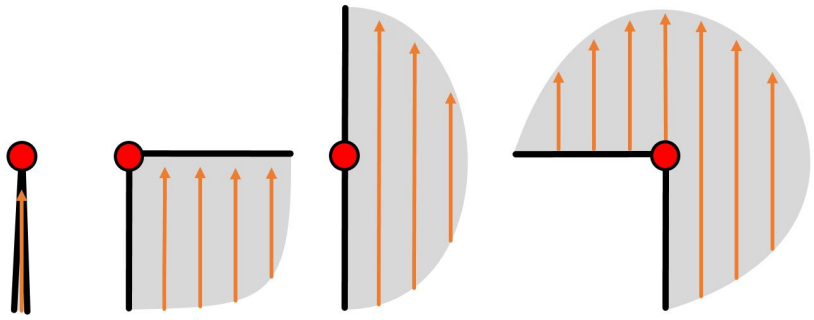


FIGURE 5.7 – Indices pour les sommets du bord avec un champ constant. De gauche à droite : indices $1/2, 1/4, 0, -1/4$.

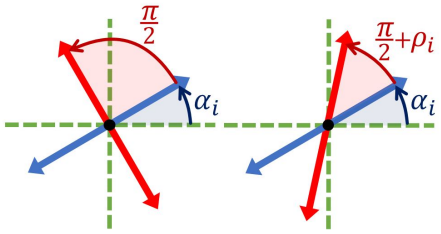


FIGURE 5.8 – Une croix non orthogonale (à droite) possède un degré de liberté supplémentaire ρ_i qui éloigne la deuxième branche de l'orthogonalité.

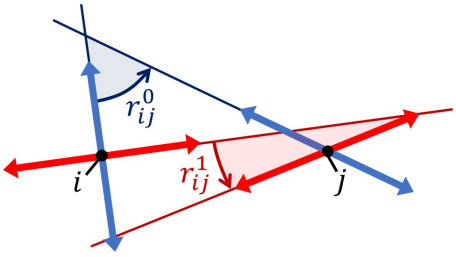


FIGURE 5.9 – Les champs de croix non orthogonales ont deux différences angulaires r^0_{ij} et r^1_{ij}, une pour chaque paire de vecteurs.

Champs de croix non orthogonaux : Nous voulons assouplir la contrainte d'orthogonalité pour que les croix non orthogonales s'adaptent aux angles nets des frontières. Ainsi, nous devons ajouter un paramètre supplémentaire à la représentation ci-dessus pour prendre en compte l'asymétrie de la croix. Nous introduisons l'angle ρ_i mesurant la déviation de l'orthogonalité ; les croix non orthogonales sont représentées par un ensemble de quatre vecteurs unitaires obtenus en faisant tourner b_i d'un angle $(\alpha_i, \alpha_i + \pi/2 + \rho_i, \alpha_i + 2\pi/2, \alpha_i + 3\pi/2 + \rho_i)$ (voir la FIGURE 5.8 pour une illustration).

Les croix non orthogonales perdent l'invariance sous rotation de $\pi/2$. Ainsi, il est important de noter que le paramètre d'asymétrie ρ_i est, par convention, toujours appliqué aux deuxième et quatrième branches. Afin de mesurer la rotation d'une croix sur le bord partagé par deux triangles adjacents i et j, nous devons définir deux angles r^0_{ij} et r^1_{ij}. Ces angles rendent compte de la rotation de la première et de la troisième branche et de la partie oblique de la deuxième et de la quatrième branche respectivement, comme l'illustre la FIGURE 5.9.

Imaginons que la première branche de la croix sur le triangle i soit appariée avec la première (ou troisième) branche de la croix sur le triangle j (c'est-à-dire que $p_{ij} = 0 \bmod 2$) alors r^0_{ij} est donné par l'équation (5.1). Dans ce cas, la deuxième branche correspond à i, la deuxième (ou quatrième) branche à j et r^1_{ij} est calculé en réappliquant l'équation (5.1) avec les angles $\alpha_i + \rho_i$ et $\alpha_j + \rho_j$.

Lorsque $p_{ij} \bmod 2 = 1$, la première branche de la croix sur le triangle i est appariée avec la

deuxième (ou quatrième) branche du cadre sur le triangle j alors r^0_{ij} est obtenu en remplaçant α_j par $\alpha_j + \rho_j$ dans l'équation (5.1).

La formule globale peut être résumée par :

$$\begin{cases} r^0_{ij} = -\alpha_i + \alpha_j + c_{ij} + p_{ij}\frac{\pi}{2} + (p_{ij} \bmod 2)\rho_j \\ r^1_{ij} = -\alpha_i - \rho_i + \alpha_j + c_{ij} + p_{ij}\frac{\pi}{2} + (1 - (p_{ij} \bmod 2))\rho_j \end{cases} \tag{5.4}$$

Ensuite, le champ de croix non-orthogonales le plus lisse est celui qui minimise la rotation totale :

$$\arg\min \sum_{ij} \left(r^0_{ij}\right)^2 + \left(r^1_{ij}\right)^2 \quad \text{sujet à des contraintes de bord.} \tag{5.5}$$

Aperçu de notre approche : Comme expliqué à la sous-section 5.2.4, l'omniprésence des angles aigus constitue un défi de taille pour le maillage quadrangulaire d'un modèle de CAO. Cela force les singularités des bords d'indice 1/2 à apparaître dans les champs de croix les plus lisses, rendant impossible l'extraction d'un maillage quadrangulaire valide. Pour éviter cette situation, l'idée principale est de "sacrifier" une partie de la propriété de "champ le plus lisse" pour obtenir des singularités acceptables sur les coins aigus. Plus précisément, nous modifions la notion de lissage pour interdire les configurations non maillables.

Ainsi, étant donné un maillage triangulaire avec un bord (rappelons que toutes les arêtes caractéristiques sont considérées comme des bords), notre algorithme calcule deux directions tangentes par triangle en suivant les étapes décrites dans les sections suivantes :

— § 5.2.4 : calculer un champ de croix **orthogonales** sans singularités d'index 1/2 aux angles aigus. L'idée est de modifier la rotation cible ω (§ 5.2.4) puis de calculer le champ de croix le plus lisse par rapport à ω. (§ 5.2.4)

— § 5.2.5 : calcule un champ **non-orthogonal** qui réduit la rotation du champ par rapport au champ orthogonal. En d'autres termes, nous relâchons la contrainte d'orthogonalité.

5.2.4 Calcul d'un champ de croix orthogonales corrigé

Une approche standard pour calculer le champ de croix le plus lisse (orthogonal) consiste à minimiser la rotation totale du champ $\sum_{ij} r^2_{ij}$ sous des contraintes de bord. Le problème, cependant, est que cette approche conduit souvent à des champs de croix non maillables. Pour éviter ce piège, nous ne générons pas de champs de croix dont la rotation est la plus faible possible, mais plutôt la plus proche d'une rotation prescrite ω qui permet de préserver les angles aigus. Nous représentons ω par un angle de rotation ω_{ij} entre chaque paire de triangles adjacents i et j. Une fois cette rotation cible obtenue, le champ de croix est simplement calculé en minimisant la distance à la cible $\sum_{ij}(r_{ij} - \omega_{ij})^2$.

Caractérisation du problème

Un champ de croix ne permet d'obtenir un maillage quadrangulaire chaque fois qu'il existe un sommet du bord avec un indice 1/2 (voir FIGURE 5.5 et FIGURE 5.11 à gauche). (se référer à l'équation (5.3) pour la définition).

Soit v un sommet du bord avec un angle cumulé θ_v proche de 0. Le champ de croix minimisant la rotation proche du sommet aura une forme proche de $\sum_{ij \in E(v)} r_{ij} = \theta_v$, ce qui implique un indice de 1/2 sur le sommet v : le champ va converger vers le point v. Notez que l'équation (5.3) fournit un moyen de détecter les configurations potentiellement problématiques. En effet, si l'angle total à un sommet de la frontière est $\theta_v < \pi/4$, minimiser les rotations des croix autour

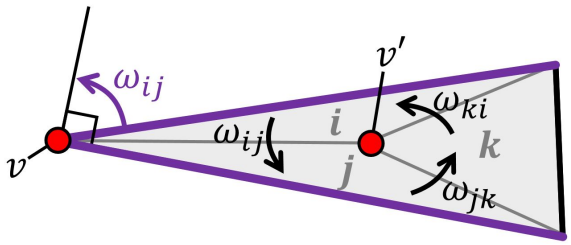


FIGURE 5.10 – Correction locale d'un angle vif : On fixe $\omega_{ij} \leftarrow \theta_v - \pi/2$ pour forcer $I(v) = 1/4$. Si ω_{jk} et ω_{ki} sont nuls, alors $\mathrm{d}\omega(v') = \theta_v - \pi/2$. Par conséquent, le champ de croix final aura $r_{ij} = \omega_{ij}$, $r_{jk} = r_{ki} = \theta_v/2$, conduisant à un indice $1/4$ à v'.

de ce sommet conduit à la singularité problématique d'indice $1/2$, avec une rotation totale du champ de θ_v. Ce cas peut être évité en forçant la rotation de la croix à être aussi proche que possible d'un champ de rotation cible ω [19] dans cette situation.

Prescrire la rotation cible ω.

Dans cette section, nous commençons par examiner une correction naïve : une prescription de rotation de la cible très locale, puis une prescription globale qui donne des résultats de grande qualité.

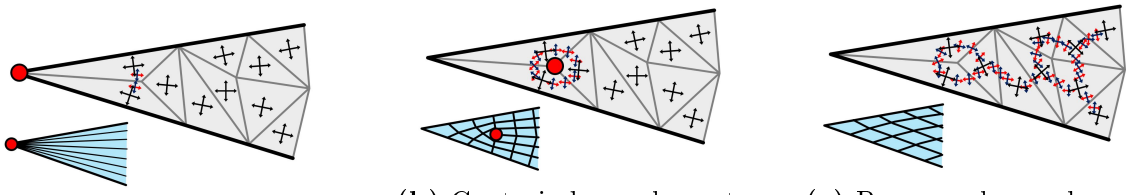
Un correctif local naïf. Le champ de rotation cible ω devrait promouvoir une singularité d'indice $1/4$ là où le champ de croix le plus lisse (équation (5.2)) est susceptible de placer une singularité d'indice $1/2$. Plus précisément, pour chaque sommet de frontière v avec un angle total $\theta_v < \pi/2$, le fait de fixer son indice à $1/4$ contraint la somme de la rotation du champ dans le 1-voisinage réécrivant l'equation (5.3) : $\sum_{ij \in E(v)} r_{ij} = \theta_v - \pi/2$. Nous ajoutons donc cette contrainte à la rotation cible, que nous écrivons comme $\mathrm{d}\omega(v) = \theta_v - \pi/2$, où $\mathrm{d}\omega(v) = \sum_{ij \in E(v)} \omega_{ij}$ désigne la dérivée extérieure de ω, étendue au bord de la surface.

Pourquoi est-ce si local ? Si on se contente de répartir uniformément $\theta_v - \pi/2$ sur le ω_{ij} de chaque arête duale $ij \in E(v)$, cela éloigne effectivement la singularité du bord, mais la positionne dans le voisinage immédiat de celui-ci. Le cas courant de deux triangles du bord adjacents i et j (FIGURE 5.11 milieu) est très représentatif de ce comportement. Comme détaillé sur la FIGURE 5.10, imposer $\mathrm{d}\omega(v) = \omega_{ji} = \theta_v - \pi/2$ conduira à une rotation correspondante de $r_{ji} = \theta_v - \pi/2$. Comme nous recherchons le champ le plus lisse, la rotation sur les arêtes duales voisines jk et ki sera $r_{jk} = r_{ki} = \theta_v/2$. Par l'équation (5.3), l'indice du sommet central est alors $-1/4$: la singularité s'est à peine éloignée de la frontière.

Ce qui s'est passé ici, c'est que $\mathrm{d}\omega(v)$ perturbe l'énergie de l'équation (5.8) nécessaire pour choisir l'indice de v : comme observé dans [Ray *et al.*, 2008], elle agit comme le défaut d'angle. De plus, si les triangles i et j partagent l'arête v, v', augmenter ω_{ij} augmente mécaniquement $\mathrm{d}\omega(v)$ et diminue $\mathrm{d}\omega(v')$ de la même quantité. En conséquence, la somme reste nulle, c'est-à-dire que $\sum_v \mathrm{d}\omega(v) = 0$. En modifiant ω uniquement dans le 1-voisinage de v, on déplace simplement $\mathrm{d}\omega(v)$ vers son sommet voisin... qui devient naturellement le nouveau sommet singulier.

Une solution globale. Au lieu de forcer le sommet le plus proche à devenir une singularité, il est souvent préférable de le placer un peu plus loin (FIGURE 5.11 droite). Pour ce faire, la modification de $\mathrm{d}\omega(v)$ qui empêche l'indice $1/2$ sur les angles aigus ne doit pas seulement être

19. Du point de vue de la géométrie différentielle, ω est une $1-$forme que nous utilisons pour redéfinir le transport parallèle. Contrairement à la redéfinition de la métrique [Panozzo *et al.*, 2014, Jiang *et al.*, 2015], elle ne modifie pas l'orthogonalité des croix.



(a) Champ le plus lisse

(b) Contraindre seulement le 1-voisinage du coin aigu.

(c) Propager la courbure du champ

FIGURE 5.11 – Sur un coin pointu, le champ le plus lisse produit des lignes de quadrilatères qui s'écrasent dans le sommet du coin **(a)** . Une correction locale résout ce problème **(b)** , mais la propagation de la courbure du champ est ce qui produit les lignes de quadrilatères les plus exploitables **(c)** .

transférée à son voisin le plus proche : il serait préférable d'avoir $\mathrm{d}\omega(v) = 0$ pour tous les autres sommets. Malheureusement, la contrainte $\sum_{ij} \mathrm{d}\omega(v) = 0$ rend la chose impossible, nous distribuons donc à la place les $\mathrm{d}\omega(v)$ contraints des sommets de coin pointus sur tous les autres sommets.

Une méthode d'optimisation en deux étapes peut être utilisée pour déterminer ω :

— on calcule un scalaire $K(v)$ par sommet v tel que $\sum_v K(v)^2$ est minimal, $K(v) = \pi/2 - \theta_v$ sur les coins aigus v, et $\sum_v K(v) = 0$. La solution distribue simplement la somme des $K(v)$ des coins pointus sur le reste des sommets du maillage. Cette optimisation est effectuée indépendamment pour chaque composante connexe de la surface (qui sont susceptibles d'être générées lors de la découpe de la surface le long des bords de la caractéristique).

— nous minimisons ensuite $\arg\min \sum_{ij} \|\omega_{ij}\|^2$ sous la contrainte $\mathrm{d}\omega(v) = K(v)$. La contrainte $\sum_v K(v) = 0$ garantit l'existence d'une solution.

Cette solution globale n'est pas toujours optimale, nous ne voulons pas que la courbure du champ s'étende trop loin de l'angle vif et produise des distorsions globales. En pratique, nous initialisons $K(v) \leftarrow \pi/2 - \theta_v$ pour les coins pointus et $K(v) \leftarrow 0$ pour les autres sommets du maillage. Ensuite, pour chaque coin pointu v_s, nous calculons l'ensemble des sommets qui peuvent être atteints en suivant moins de 5 arêtes. Ce compromis permet d'éviter une forte distorsion locale ou globale. L'idée est de se dire que les arêtes des maillages triangulaire et quadrangulaire auront à peu près la même taille, et que la propagation déplacera l'eventuelle singularité à au moins à 5 quadrilatères du bord.

Soit n la taille de cet ensemble de sommets, nous mettons à jour K pour tous les sommets de l'ensemble : $K(v) \leftarrow K(v) - K(v_s)/n$. Nous optimisons ensuite les valeurs d'oméga $\arg\min \sum_{ij} \|\omega_{ij}\|^2$ sous la contrainte $\mathrm{d}\omega(v) = K(v)$.

Calcul d'un champ de croix orthogonales

Commençons par une brève introduction à la génération de champs de croix orthogonales les plus lisses (équation (5.2)). L'état de l'art consiste à verrouiller les croix du bord pour avoir une direction tangente à la frontière, et de minimiser la rotation totale du champ. Notez que si un triangle a deux bords de frontière, on peut diviser le triangle en trois en ajoutant un sommet au centre pour éviter de sur-contraindre le problème.

Le champ le plus lisse est celui dont la rotation r_{ij} est minimale. Pour éviter d'avoir à manipuler la variable entière p_{ij}, nous utilisons la modélisation bien connue consistant à utiliser un "vecteur de représentation" [Ray *et al.*, 2016]. Le champ de croix orthogonales le plus lisse est celui qui a une rotation nulle sur chaque bord double (c'est-à-dire $r_{ij} = 0$). Ainsi, dans l'équation

(5.1), on peut multiplier par 4i et prendre l'exponentielle complexe, ce qui donne :

$$e^{4\mathrm{i}\alpha_i} = e^{4\mathrm{i}\alpha_j} e^{4\mathrm{i}c_{ij}} e^{2\mathrm{i}p_{ij}\pi}. \tag{5.6}$$

Comme p_{ij} est entier, le dernier terme est égal à un. Nous pouvons effectuer un changement de variable $z_i = e^{4\mathrm{i}\alpha_i}$, en représentant une croix par un nombre complexe unitaire par triangle. Ensuite, le champ de croix le plus lisse est obtenu en résolvant le problème d'optimisation suivant :

$$\underset{\|z\|=1}{\arg\min} \sum_{ij} \|z_i - z_j e^{4\mathrm{i}c_{ij}}\|^2 \qquad \text{sujet à des contraintes de bord} \tag{5.7}$$

Comme les modèles de CAO comportent de nombreuses arêtes caractéristiques qui forcent $\|z_i\| = 1$ sur les triangles voisins, une pratique courante pour résoudre ce problème consiste à utiliser une méthode ordinaire des moindres carrés sans contrainte de normes unitaires, puis de reprojeter les croix vers une solution unitaire avec une renormalisation des valeurs complexes. À la fin du processus, nous retrouvons les p_{ij} en choisissant les valeurs qui minimisent les r_{ij} dans l'équation (5.1).

Notre méthode nécessite de calculer un champ de croix dont la variation est aussi proche que possible de la rotation prescrite ω. Il s'avère que ce problème peut être résolu exactement comme le champ de croix le plus lisse par une simple mise à jour du problème d'optimisation (5.7) :

$$\underset{\|z\|=1}{\arg\min} \sum_{ij} \|z_i - z_j e^{4\mathrm{i}(c_{ij}-\omega_{ij})}\|^2 \qquad \text{sujet à des contraintes de bord} \tag{5.8}$$

Quant aux variables p_{ij}, elles sont maintenant définies pour minimiser $r_{ij} - \omega_{ij}$.

5.2.5 Relâcher l'orthogonalité

À ce stade, nous générons un champ de croix orthogonales en minimisant $\sum_{ij}(r_{ij} - \omega_{ij})^2$. Sa topologie est suffisamment bonne pour une application de maillage quadrangulaire, sauf dans certaines configurations délicates, où [Myles *et al.*, 2014] insère un dipôle $1/2, -1/2$. La géométrie peut être encore optimisée en relâchant la contrainte d'orthogonalité. L'objectif ici est de minimiser $\sum_{ij} \left(r^0_{ij}\right)^2 + \left(r^1_{ij}\right)^2$ comme défini dans l'équation (5.4), mais avec cette fois-ci les p_{ij} fixés.

Au lieu de travailler avec les variables ρ_i, α_i, nous allons optimiser les angles de rotation à appliquer à chaque vecteur du champ de croix : γ^0_i pour les vecteurs pairs et γ^1_i pour les vecteurs impairs. Le champ d'images final sera obtenu en actualisant $\alpha_i \leftarrow \alpha_i + \gamma^0_i$, et en fixant $\rho_i \leftarrow \gamma^1_i - \gamma^0_i$.

Nous pouvons réécrire l'équation (5.4) pour les nouvelles variables γ^0_i et γ^1_i :

$$\begin{cases} r^0_{ij} &= -(\alpha_i + \gamma^0_i) + \alpha_j + \gamma_j^{p_{ij} \bmod 2} + c_{ij} + p_{ij}\frac{\pi}{2} \\ r^1_{ij} &= -(\alpha_i + \gamma^1_i) + \alpha_j + \gamma_j^{(p_{ij}+1) \bmod 2} + c_{ij} + p_{ij}\frac{\pi}{2} \end{cases}$$

Le nouveau champ est obtenu en minimisant :

$$\arg\min \left\{ \sum_{ij} \left[\left(r^0_{ij}\right)^2 + \left(r^1_{ij}\right)^2 \right] + \varepsilon \sum_i \left(\gamma^0_i - \gamma^1_i\right)^2 \right\},$$

où $\varepsilon = 10^{-2}$. Notez que le terme $\sum_i \left(\gamma^0_i - \gamma^1_i\right)^2$ favorise le fait que le champ soit orthogonal, ce qui est nécessaire pour les configurations sous contraintes telles que l'anneau ou le tore, où une des dimensions des croix peut ne pas avoir de contraintes.

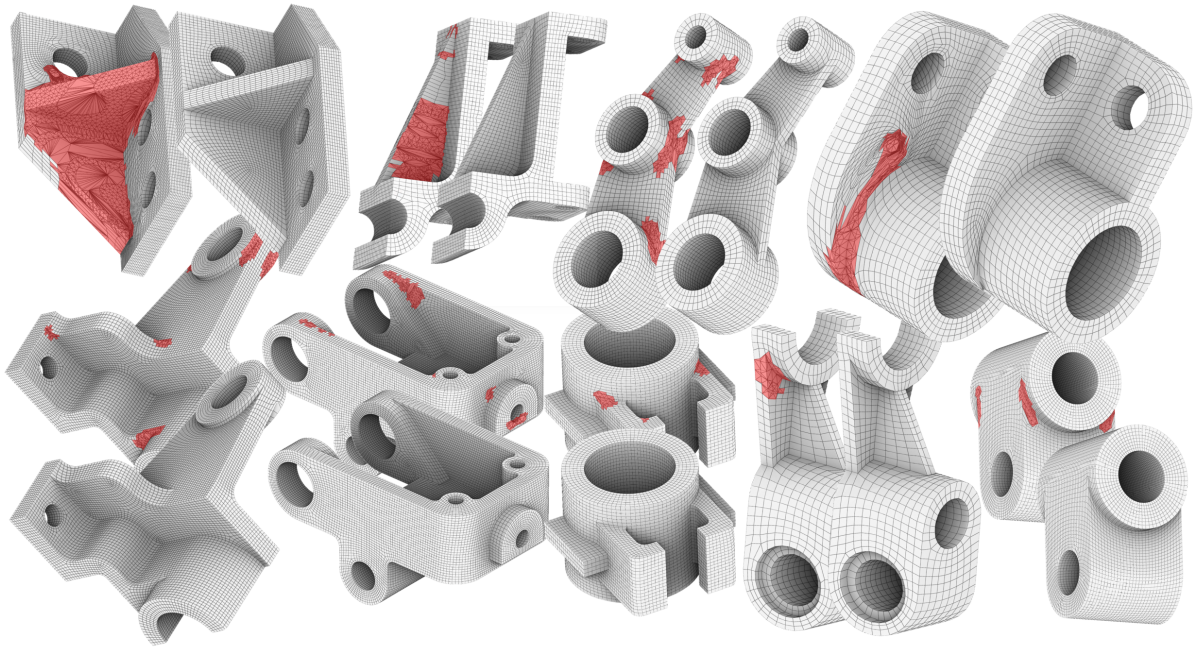


FIGURE 5.12 – Chaque modèle est remaillé avec (premier plan) et sans (arrière-plan) correction des angles vifs par notre méthode. En rouge les éléments invalides.

Pour éviter les croix dégénérées ($\rho_i \approx \pi/2$, les deux branches sont quasi colinéaires), une fois l'énergie minimisée, nous vérifions si $|\gamma_i^0 - \gamma_i^1| > \alpha$, où $0 < \alpha < \pi/2$ est la déviation maximale de l'orthogonalité. Nous utilisons $\alpha = 0.45\pi$ dans nos expériences. Si la condition n'est pas satisfaite, nous fixons la croix localement en appliquant la plus petite modification possible à γ_i^0 et γ_i^1 :

— si $\gamma_i^0 > \gamma_i^1$ alors $\gamma_i^0 \leftarrow (\gamma_i^0 + \gamma_i^1 + \alpha)/2$ et $\gamma_i^1 \leftarrow (\gamma_i^0 + \gamma_i^1 - \alpha)/2$,
— sinon $\gamma_i^0 \leftarrow (\gamma_i^0 + \gamma_i^1 - \alpha)/2$ et $\gamma_i^1 \leftarrow (\gamma_i^0 + \gamma_i^1 + \alpha)/2$.

5.2.6 Résultats et applications

Notre méthode possède un temps de calcul un peu supérieur à celles de génération d'un champ de croix en utilisant des méthodes moindres carrés. En effet, elle nécessite la résolution de 3 problèmes moindres carrés successifs sur le maillage : un premier pour trouver ω (le temps de calcul de $K(v)$, et donc la quantité de coins problématiques, est négligeable), un second pour le champs orthogonal, et enfin un troisième pour relaxer l'orthogonalité.

Dans le cas de la génération de maillages quadrangulaires, ces temps sont négligeable à ceux des méthodes de type *quadcover* [Kaelberer *et al.*, 2007], utilisées pour transformer le champ de croix en maillage quadrangulaire. À titre d'exemple, un modèle de 35K triangles nécessite 3 secondes de calcul avec notre méthode : $0,4$ secondes pour calculer ω, $0,5$ secondes pour optimiser le champ orthogonal, et $1,3$ pour relaxer l'orthogonalité ; contre 155 secondes pour le calcul d'une paramétrisation globale.

Maillage quadrangulaire

Toute une famille d'algorithmes de maillage quadrangulaire prend un champ de croix en entrée, calcule une paramétrisation globale et en extrait des quadrilatères. Pour obtenir les résultats présentés, nous utilisons le quadcover [Kaelberer *et al.*, 2007]. À partir d'une paramétrisation de

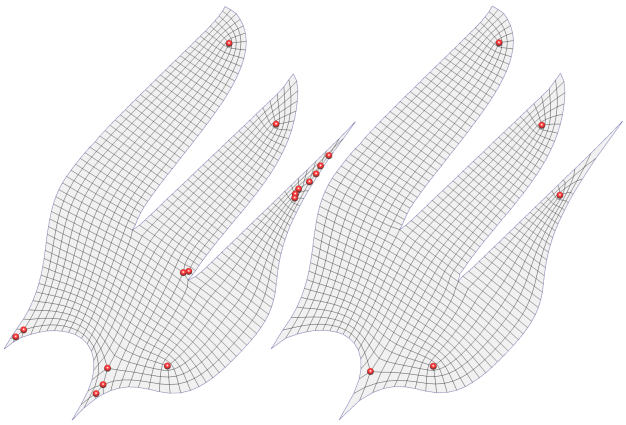


FIGURE 5.13 – Le maillage quadrangulaire généré par [Fang *et al.*, 2018] (à gauche) nécessite beaucoup plus de singularités pour traiter les bords des caractéristiques que le nôtre (à droite).

quadcover, les maillages extraits ont des sommets extraordinaires (valence $\neq 4$) uniquement sur les singularités du champ de croix d'entrée. Des travaux antérieurs comme [Ray *et al.*, 2006, Jakob *et al.*, 2015, Fang *et al.*, 2018] ont traité ces cas de modèles de CAO à partir de champs de croix orthogonales traditionnels, mais au prix de la création de sommets plus extraordinaires dans le maillage quadrangulaire de sortie (voir FIGURE 5.13). Au lieu de cela, nous traitons les modèles de CAO avec des angles aigus, en travaillant sur l'intégrabilité du champ de croix et en utilisant des techniques d'intégration simples comme le quadcover [Kaelberer *et al.*, 2007].

Conclusion

Les précédentes méthodes de génération de maillages quadrangulaires utilisant des champs de repères étaient peu adaptées aux modèles de CAO. Cela est principalement dû aux configurations de coins aigus omniprésentes sur ces modèles. Notre méthode permet de légèrement modifier la génération du champ pour grandement améliorer la robustesse des étapes suivantes, particulièrement du *quadcover*. Nos champs permettent d'obtenir d'excellents maillages sur tout un ensemble de géométries qui étaient problématiques. En plus de la contribution pour la génération de maillages quadrangulaires, ces champs de qualité, et les maillages associés, nous permettent de construire des supports pertinent pour l'étude du bord des objets, étape inévitable pour la création de méthode robuste de génération de maillages volumiques (hexaédriques).

5.3 *marchinghex*, extracteur robuste

Dans cette section nous présentons la méthode *marchinghex* pour extraire un maillage d'un domaine depuis une grille, et cela de façon robuste. En liaison avec les polycubes, l'objectif est d'obtenir un maillage avec des polycubes correspondants globalement au domaine étudié, et dans les lieux où la différence est trop forte, appliquer l'algorithme de *marchinghex* pour avoir un maillage parfaitement fidèle au bord.

Le maillage hexaédrique automatique a été un sujet de recherche actif au cours des deux dernières décennies. À ce jour, nous pouvons classer les algorithmes de maillage entièrement hexaédrique en 2 catégories :

1. Non automatique ;

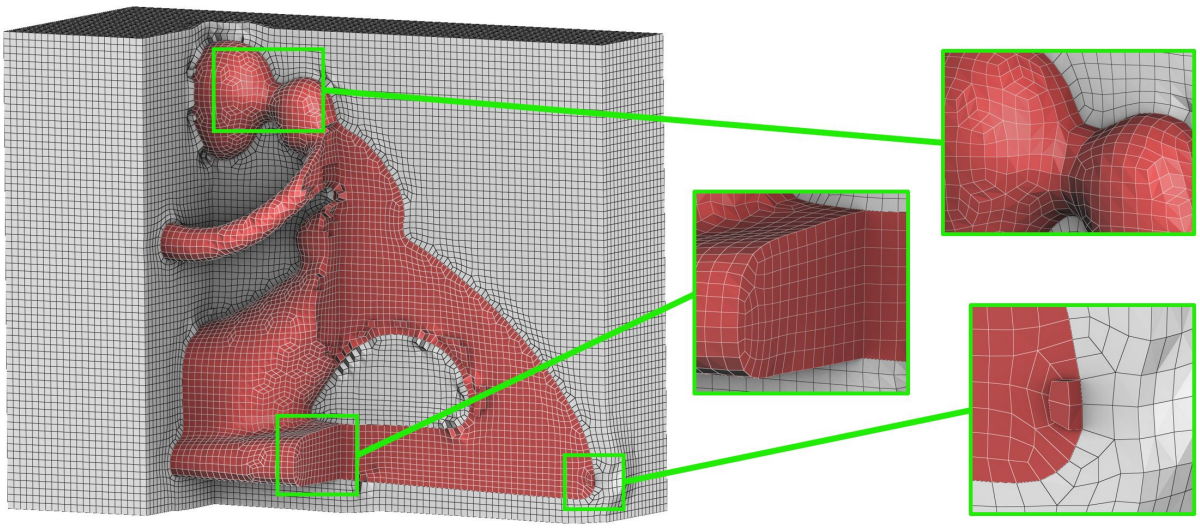


FIGURE 5.14 – Notre méthode permet de construire des maillages bimatériaux qui sont (1) entièrement hexaédriques, (2) assortis à l'intérieur et à l'extérieur et (3) garantis sans éléments inversés (*scaled jacobian* > 0).

2. Méthode basée sur une intersection de grilles avec des raffinement locaux.

La deuxième catégorie a un défaut majeur, c'est que le maillage est mal structuré. Mais pour contrebalancer cela, elle a plusieurs grands avantages : elle part d'un maillage valide et ne peut que s'améliorer, a une taille d'éléments contrôlable et est très efficace. Introduit en 1984 par Yerry et Shephard [Yerry et Shephard, 1984], elle extrait un maillage hexaédrique à partir du dual modifié d'une grille Octree. Ce maillage doit ensuite subir des modifications telles que du *padding*[20] sur le bord et une projection pour suivre au mieux le bord du domaine maillé et ses principales caractéristiques. Industriellement, l'état de l'art est le logiciel *Hexotic* de Loïc Maréchal décrit dans [Maréchal, 2009, Maréchal, 2016].

Les méthodes d'intersection de grilles font encore l'objet de recherches récentes : [Gao *et al.*, 2019, Livesu *et al.*, 2022, Pitzalis *et al.*, 2021]. Bien que sur le papier, elles semblent avoir atteint leur plein potentiel, on peut tout de même espérer obtenir de meilleurs résultats en les combinant avec d'autres méthodes, comme la paramétrisation ou la déformation préalable. Malheureusement, l'un des problèmes est la reproductibilité et la souplesse d'utilisation, qui réside dans la complexité de l'algorithme et de la méthode utilisée, ainsi que dans la quantité d'ingénierie nécessaire. [Livesu *et al.*, 2022] a fait un excellent travail en spécifiant la partie octree de la méthode, mais les parties de projection et lissage restent difficilement reproductibles. Dans cette section, nous présentons un cadre simple pour traiter la frontière et la projection, en utilisant une approche basée sur les coupes introduite par Dhondt [Dhondt, 2001]. Notre travail vient compléter les travaux de Livesu *et al.* pour proposer des méthodes abordables d'interesction de grilles, permettant de simplifier leur étude en conjonction avec d'autres approches, comme les déformations.

L'entrée de notre algorithme est une grille recouvrant le domaine Ω, que nous prenons régulière dans nos exemples, mais qui pourra en pratique être un maillage hexaédrique quelconque.

20. Insertion d'une couche d'hexaèdres entourant le maillage hexaédrique du domaine. Sur la FIGURE 5.15–au milieu, la couche d'éléments en rouge est ajoutée aux éléments originaux en bleu, pour améliorer la qualité du maillage le long du bord.

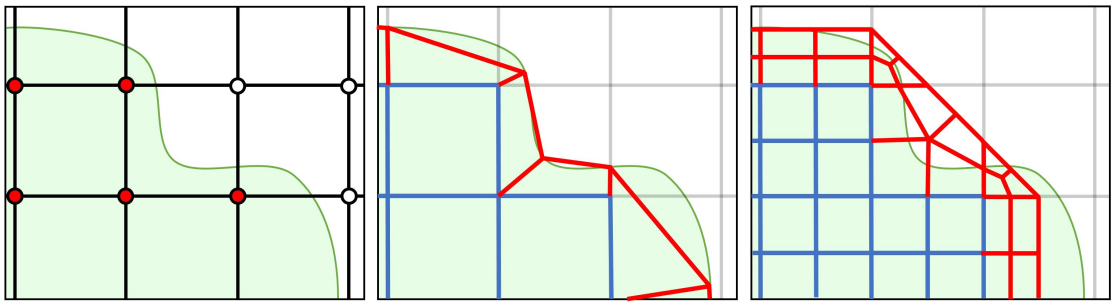


FIGURE 5.15 – À partir d'une grille donnée **(a)**, [Maréchal, 2009] projette les sommets sur le bord du domaine pour obtenir une couche de *padding* correspondant à la frontière **(b)**. tandis qu'avec l'approche de Dhondt, nous extrayons les éléments nécessairement valides par une subdivision supplémentaire **(c)**. Dans les deux approches, une meilleure qualité et une meilleure correspondance au bord sont obtenues avec le lissage.

Sur chaque voxel, nous extrayons un motif hexaédrique avec des éléments valides (voir FIGURE 5.15–droite), avec un intérieur et un extérieur. Sur le motif, certains sommets (en vert (voir FIGURE 5.21) sont clairement candidats à la projection sur des éléments clés. Pour s'adapter au bord et à ces caractéristiques clés, nous projetons et améliorons localement le maillage en utilisant une énergie elliptique. Cela garantit la validité du maillage à toutes les étapes du processus.

En améliorant le travail de Dhondt, nous proposons un algorithme simple pour obtenir chaque motif automatiquement, sans dépendre du traitement exhaustif des cas. De plus, grâce à la correspondance automatique des caractéristiques par rapport au modèle, nous obtenons une procédure de lissage simple et robuste. Enfin, pour faciliter la reproductibilité et les utilisations futures, le code est disponible sur github : *https://github.com/fprotais/marchinghex*.

Discussion sur les travaux connexes

La méthode de génération d'hexaèdre à partir d'un octree peut être divisée en 3 étapes :

Génération de la grille Octree : Nous ne détaillons pas ce sujet, car cela a déjà été bien fait dans le récent [Livesu *et al.*, 2022]. Nous nous baserons ici sur une grille régulière, et laisserons l'étude de l'effet de notre approche sur des grilles octree générales à un travail ultérieur.

Extraction hexaédrique : Comme le montre la FIGURE 5.15, [Maréchal, 2009, Maréchal, 2016] travaille directement sur la sortie de la génération de la grille basée sur l'octree, sur laquelle ils doivent ajouter une couche d'hexaèdres pour s'assurer que le maillage correspondra au bord du domaine. Cette étape est assez délicate, car elle peut introduire des éléments mal formés. Nous préférons utiliser l'approche de Dhondt [Dhondt, 2001], qui garantit l'introduction de bons éléments sur une grille régulière. L'inconvénient est qu'elle nécessite une subdivision supplémentaire qui multiplie le nombre d'éléments par un facteur 8 environ. Cela nécessite donc des réglages différents sur la génération de la grille pour obtenir des résultats similaires.

Lissage final du maillage : [Maréchal, 2009] repose sur une procédure simple pour déplacer localement un sommet. Bien qu'en pratique tous leurs éléments soient positifs, il n'y a pas de

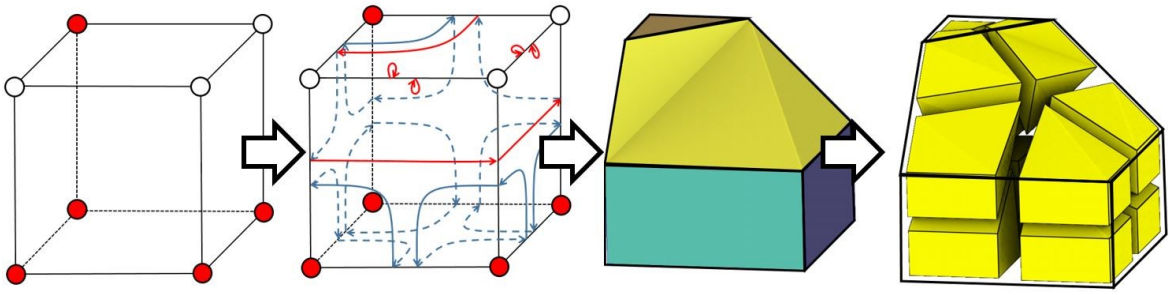


FIGURE 5.16 – À partir d'une configuration de voxel **(a)**, le dual du voxel est simplifié **(b)**, à partir duquel un polyèdre peut être extrait **(c)**. Enfin, le motif hexaédrique est obtenu par *mid-point subdivision* **(d)**.

garanties publiées. Nous avons proposé une procédure de lissage qui s'aligne lentement sur les limites et les éléments clés. Elle s'appuie sur une énergie de lissage elliptique que nous avons proposé dans [Garanzha *et al.*, 2021a] (chapitre 3), et ne retourne pas les éléments. Nous échantillonnons cette énergie sur un ensemble de tétraèdres, de manière similaire à *Mesquite* [Knupp, 2012]. L'ensemble choisi est suffisant pour que le *scaled jacobian* soit garanti positif.

Notre idée principale est d'extraire un maillage hexaédrique valide qui ne correspond pas au bord, puis d'améliorer lentement la fidélité au bord tout en conservant la validité. Nous présentons une extraction de motifs locaux dans la sous-section 5.3.1, qui est ensuite améliorée en utilisant une énergie elliptique pour mieux correspondre à la frontière dans la sous-section 5.3.2. Enfin, nous présentons un ensemble de résultats pour différentes applications dans la sous-section 5.3.3.

5.3.1 Extraction de motifs

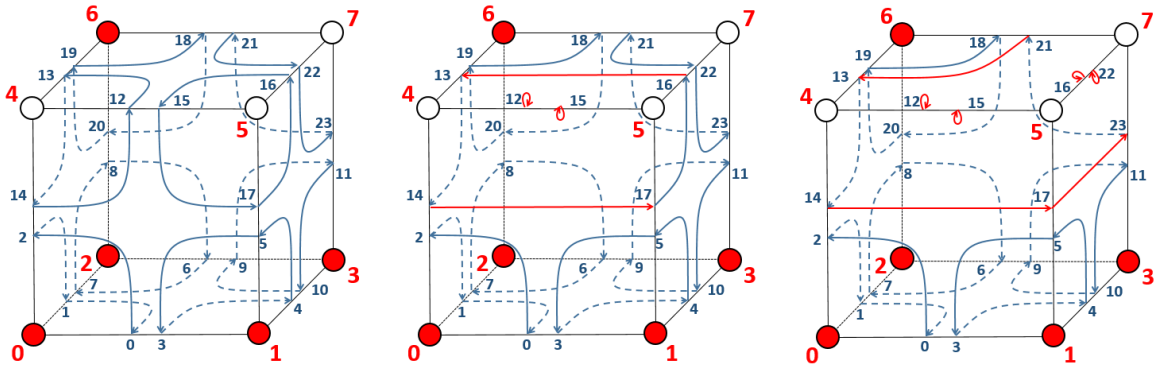
Soit Ω le domaine à mailler. Nous disposons d'une grille couvrant ce domaine, et, à partir de cette grille, nous extrayons un maillage hexaédrique. Pour éviter toute confusion, la grille pouvant elle-même être considérée comme un maillage hexaédrique, nous utilisons les noms suivants :

— Un élément de la grille couvrant le domaine, appelé par la suite *grille*, sera appelé *voxel* et ses sommets *noeuds*,
— Un élément du maillage hexaédrique final, appelé par la suite *maillage hexaédrique*, sera appelé un *hexaèdre* et ses sommets *sommets*,
— Un *point* fera référence à un point du domaine Ω.

La seule information nécessaire pour commencer le processus d'extraction de motifs est de savoir si chaque nœud de la grille est à l'intérieur ou à l'extérieur de Ω. Nous commençons par extraire un motif sur chaque voxel en garantissant la correspondance entre les voxels voisins. Nous présentons ensuite un pré-traitement de la grille assurant la validité de tous les motifs. Enfin, nous proposons une analyse de la qualité des hexaèdres générés sur une grille régulière.

Génération automatique de motifs

Le nombre de motifs nécessaires pour couvrir toutes les configurations est simplement de $2^8 = 256$ (comme pour Marching Cubes [Lorensen et Cline, 1987]). Cela signifie que la création manuelle de toutes les configurations est une possibilité, mais elle est très fastidieuse. Au lieu de cela, nous proposons une alternative qui repose sur le calcul de polyèdres correspondants au domaine. Comme le montre la FIGURE 5.16, nous commençons par simplifier le dual du voxel.



(a) **(b)** **(c)**

ID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
(a) HE =	2	0	1	4	5	3	7	8	6	11	9	10	13	14	12	17	15	16	20	18	19	22	23	21
(b) HE =	2	0	1	4	5	3	7	8	6	11	9	10	-1	14	17	-1	13	16	20	18	19	22	23	21
(c) HE =	2	0	1	4	5	3	7	8	6	11	9	10	-1	14	17	-1	-1	23	20	18	19	13	-1	21

FIGURE 5.17 – Exemple d'itération de l'algorithme 3. **(a)** est l'initialisation du tableau dual HE avec tous les nœuds du voxel à l'intérieur. **(b)** et **(c)** montrent les itérations successives à mesure que les arêtes $\{4,5\}$ et $\{5,7\}$ sont supprimées.

Le primal du dual nouvellement obtenu nous donne le polyèdre, et le modèle hexaédrique est obtenu par *mid-point subdivision* [Li *et al.*, 1995].

Simplification du dual La clé de la simplification consiste à utiliser un simple tableau dual HE de 24 entiers codant la connectivité, comme le montre la FIGURE 5.17-**(a)**. À l'aide de l'algorithme 3, HE est modifié de manière itérative pour correspondre à la propriété du voxel (voir FIGURE 5.17-**(b)**-**(c)**).

Algorithme 3 : Extraction duale d'un voxel V

input : Tableau dual HE de V avec tous ses noeuds *in*
Résultat : Tableau dual HE de V avec les propriétés voulues
pour $\{v_1, v_2\}$ *arête de* V **faire**
  si $isOut(v_1)$ ***et*** $isOut(v_2)$ **alors**
    $HE[\text{prec}(v_1)] = HE[v_1]$;
    $HE[\text{prec}(v_2)] = HE[v_2]$;
    $HE[v_1] = -1$;
    $HE[v_2] = -1$;

Retour au primal Le but est de calculer un polyèdre tel que si un nœud est à l'intérieur de Ω, alors il est à l'intérieur du polyèdre, et vice versa. Avec cette information, nous pouvons introduire des facettes avec de nouveaux sommets là où les nœuds sont manquants, et connecter le polyèdre en utilisant HE.

Mid-point subdivision Le polyèdre construit a une propriété essentielle : chacun de ses coins a une valence 3. Cela signifie qu'une subdivision standard, qui introduit des sommets au milieu des arêtes, des faces et du volume, aboutira à une configuration entièrement hexaédrique.

Traitement des motifs non valides

[Dhondt, 2001] décrit un ensemble de configurations de nœuds qui conduit à des hexaèdres de mauvaise qualité. En utilisant notre méthode, nous avons remarqué que les problèmes apparaissent lorsque, en tournant autour d'une facette, nous obtenons une configuration in-out-in-out. Intuitivement, on peut voir sur la FIGURE 5.18-**(a)**-**(b)**, que cela introduit une ambiguïté quant au couple à relier à l'intérieur du domaine. Sur ces configurations, notre méthode introduit des hexaèdres qui sont topologiquement invalides.

Heureusement, Dhondt a également remarqué que l'introduction astucieuse de coupes dans la grille pouvait éviter complètement ces cas. Sur l'exemple précédent, cela revient à couper la facette problématique en 2 facettes valides. L'inconvénient de cette approche est que la coupe doit être propagée le long d'une couche entière de la grille, ce qui introduit un nombre important de voxels. Le bon côté des choses est que ces configurations peuvent être facilement évitées par un choix judicieux de grille, et qu'une simple coupe en 3 est suffisante pour garantir la qualité globale du maillage hexaédrique.

Qualité

En partant d'une grille régulière, nous pouvons facilement vérifier la qualité de toutes les configurations possibles (voir FIGURE 5.19). Toutes les configurations contiennent des éléments valides, avec l'élément de plus mauvais *scaled jacobian* de $0,277$. Les mailles intérieures et extérieures sont parfaitement compatibles. Les configurations les plus mauvaises peuvent nécessiter jusqu'à 3 coupes pour s'assurer qu'elles soient valides. Pour générer toutes ces configurations, un binaire *make_example* est disponible dans le code.

Un travail en cours consiste à tester la qualité de la configuration à travers différentes transformations, principalement celles générées par une génération de grille octree.

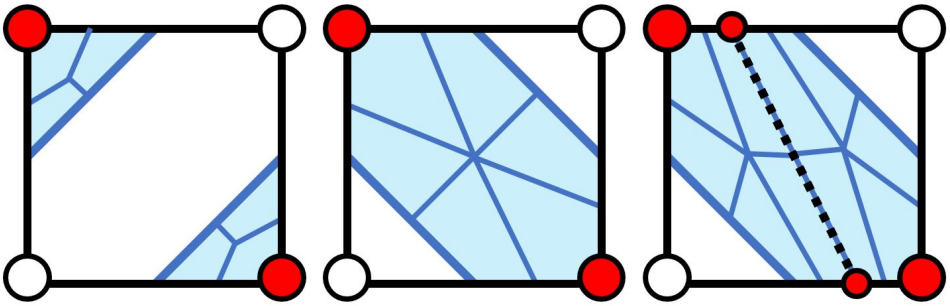
5.3.2 Projection et lissage des frontières

L'extraction du motif repose uniquement sur la grille, et ne correspond en aucun cas à la frontière. Pour obtenir le maillage final, nous déplaçons lentement les sommets vers le bord et les caractéristiques (voir FIGURE 5.20).

Projection sur le bord

Les motifs présentent un clair avantage : chaque sommet introduit a une préférence claire sur le bord. Comme le montre la FIGURE 5.21, si un sommet se trouve sur l'arête d'un voxel, il est clairement candidat à la projection. Une règle standard peut être appliquée pour les points caractéristiques dans le voxel, les arêtes caractéristiques traversant les facettes du voxel, et enfin les triangles coupant le voxel. Dans le code, nous intersectons le domaine, la frontière et les bords des caractéristiques avec la grille, et cela donne des souhaits sur chaque voxel, qui correspondront aux sommets extraits dans le voxel. Ces souhaits correspondent entre voxels voisins.

Ces souhaits sont marqués en fonction du type : sur le bord, sur une arête caractéristique et sur un point caractéristique. Sauf pour ce dernier, nous conservons une petite région de triangles



(a) **(b)** **(c)**

FIGURE 5.18 – **(a)** et **(b)** montrent 2 maillages différents pour une configuration de facettes. Cette ambiguïté conduit à des facettes incompatibles entre voxels et à des éléments mal formés. **(c)** [Dhondt, 2001] propose d'introduire des coupes dans la grille (en pointillées) qui suppriment toutes ces ambiguïtés.

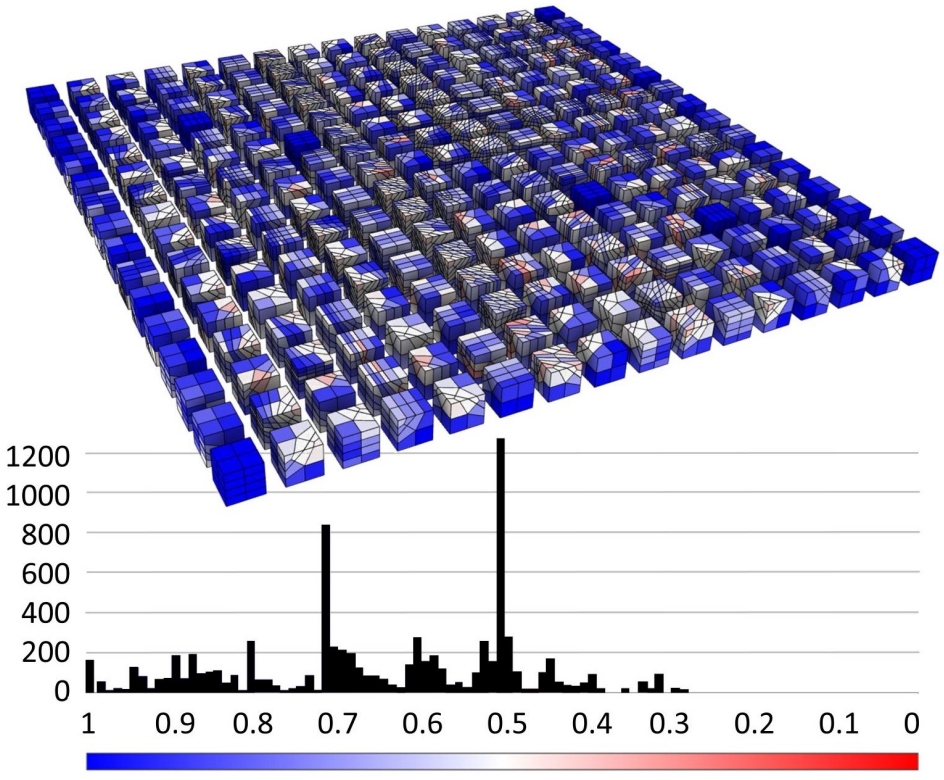


FIGURE 5.19 – Les 256 configurations possibles sont générées sur une grille standard, avec des mailles intérieures et extérieures. Le *scaled jacobian* minimum/moyen est de 0,277/0,643.

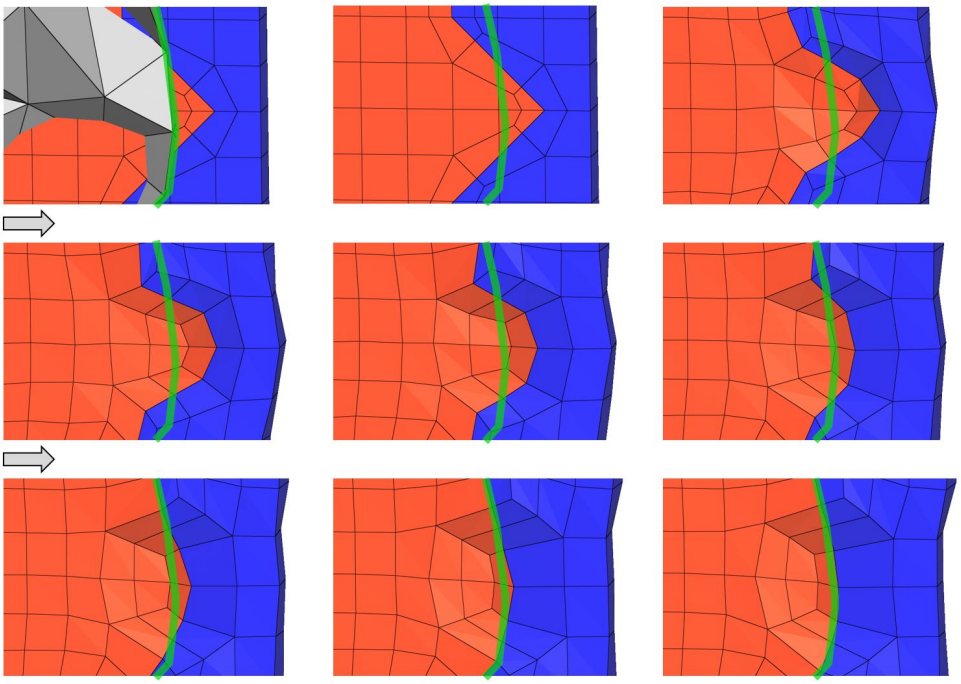


FIGURE 5.20 – Les sommets du maillage sont lentement déplacés afin que le bord du maillage corresponde à celui du domaine.

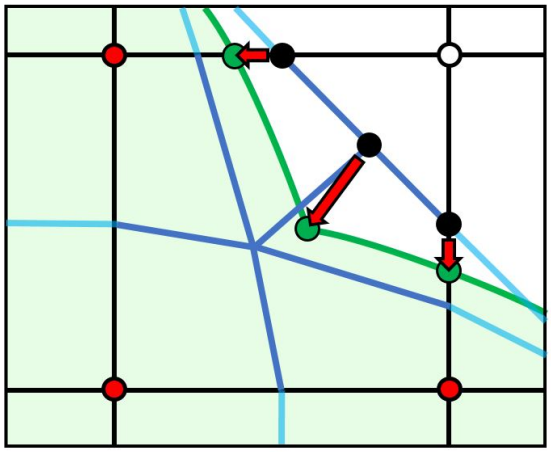


FIGURE 5.21 – Les sommets (en noir) du maillage hexaédrique ont des candidats clairs (en vert) pour la projection.

ou d'arêtes sur laquelle chaque point veut être projeté. Lorsque nous voulons qu'un sommet soit sur le bord, nous trouvons le candidat le plus proche dans cette région.

Il faut noter que pour que le lissage se fasse facilement, les points de projection doivent être aussi éloignés que possible du nœud de la grille (autrement la subdivision introduit 3 sommets très proches du noeud, difficile à lisser). Actuellement, la grille choisie est uniquement régulière, et donc arbitraire, ce qui entraîne beaucoup de problèmes locaux pour converger vers une bonne correspondance des bords. Un bon choix de grille permettrait de déplacer les nœuds pour s'assurer qu'ils ne sont pas trop proches du bord du domaine.

Lissage local

Pour améliorer la fidélité au bord et conserver la qualité en même temps, nous définissons une énergie qui nous donne une idée de la qualité du maillage. Chaque coin d'un hexaèdre définit une base (celle utilisée pour calculer le *scaled jacobian*). Cette base fournit un tétraèdre qui ne doit pas être inversé. Sur cet ensemble d'éléments, nous utilisons l'énergie de [Garanzha *et al.*, 2021a], largement étudiée dans le chapitre 3. Cette énergie a plusieurs propriétés :

— si aucun élément n'est inversé, l'inversion d'un élément donne une énergie infinie ;
— s'il existe des éléments inversés, elle définit une suite d'optimisation qui les démêle ;
— elle est bien adaptée à l'optimisation numérique (matrice hessienne définie positif).

Notre processus est le suivant : nous déplaçons chaque point indépendamment, avec une itération de Newton, avec la contrainte que nous ne nous déplaçons que si nous améliorons la qualité locale. Ensuite, si le point a besoin d'une projection, nous trouvons le candidat le plus proche, et nous le déplaçons le plus loin possible sans inverser un élément.

Si nous partons d'un maillage valide, ce processus est garanti de ne pas inverser d'éléments. Il améliorera tout de même la qualité du maillage s'il commence inversé, comme détaillé dans le chapitre 3. Pour prévenir toute inversion, un ensemble complexe de tétraèdres doit être choisi, ce qui constitue un compromis entre les garanties et les performances. Nous avons trouvé qu'un ensemble de 32 peut empêcher toute inversion en pratique, mais est lent, voir Subsec.5.3.3.

L'utilisation d'une approche locale permet une projection robuste sur le bord. De plus, nous évitons actuellement le parallélisme pour nous assurer que deux points proches ne seront pas déplacés ensemble, introduisant des éléments inversés. Beaucoup d'améliorations pourraient être apportées à ce sujet, mais le code a l'avantage d'être très simple.

5.3.3 Résultats

Temps

Tous les timings sont réalisés à l'aide de `https://github.com/fprotais/marchinghex` sur un i7-8850 à 6 cœurs utilisant MSVC. La correspondance entre le domaine et la grille est entièrement parallélisable, et notre implémentation peut générer plusieurs millions d'hexaèdres par minute. Le goulot d'étranglement évident se situe au niveau du lissage. Actuellement, le code a une moyenne de 5000 sommets par seconde par itération, et pour obtenir un bon résultat, au moins 10 itérations sont nécessaires.

Qualité

La FIGURE 5.22 montre un maillage hexaédrique obtenu sur un domaine lisse avec notre algorithme sur une grille régulière. On peut voir que 20 itérations de lissage suffisent pour obtenir un maillage de bonne qualité. La FIGURE 5.23 montre que notre algorithme fait un bon travail de

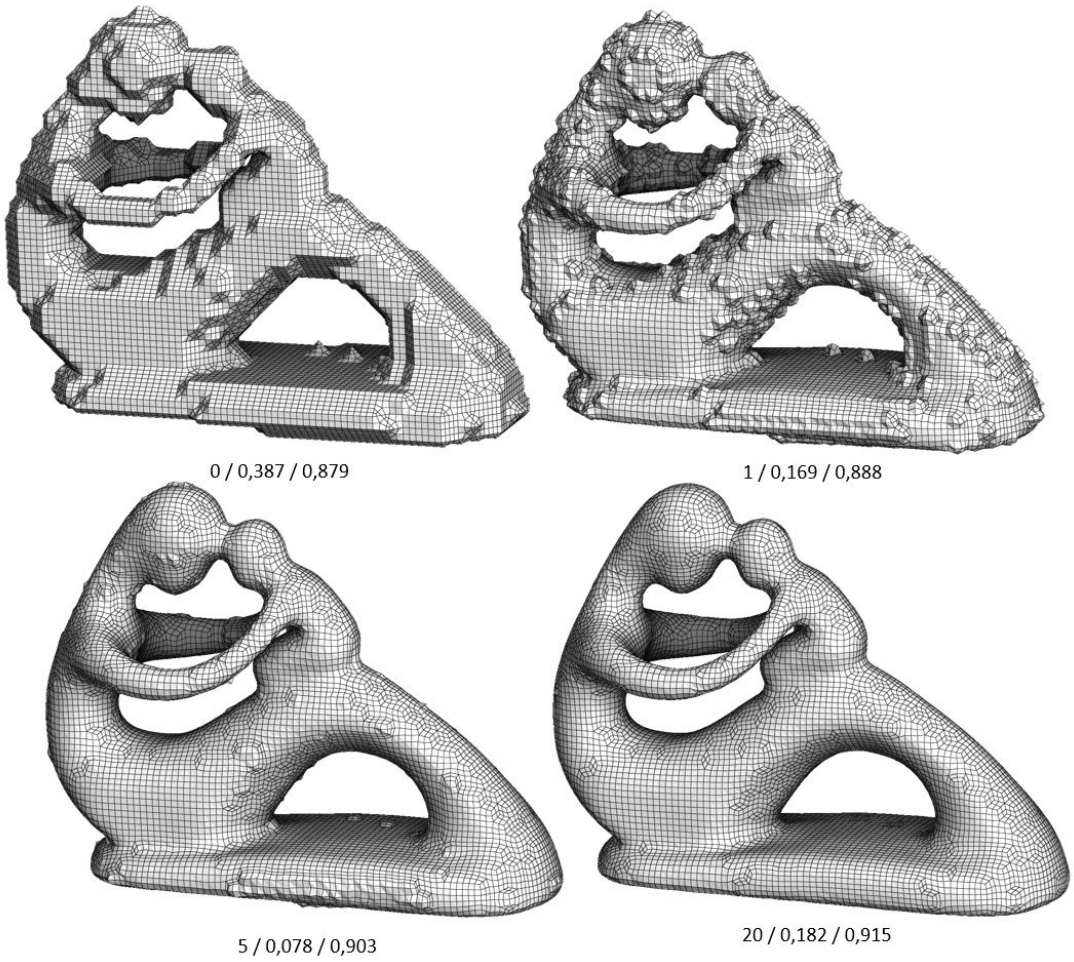


FIGURE 5.22 – Maillage hexaédrique du modèle de *Fertility* à l'aide d'une grille régulière. (nb itération / min SJ / avg SJ)

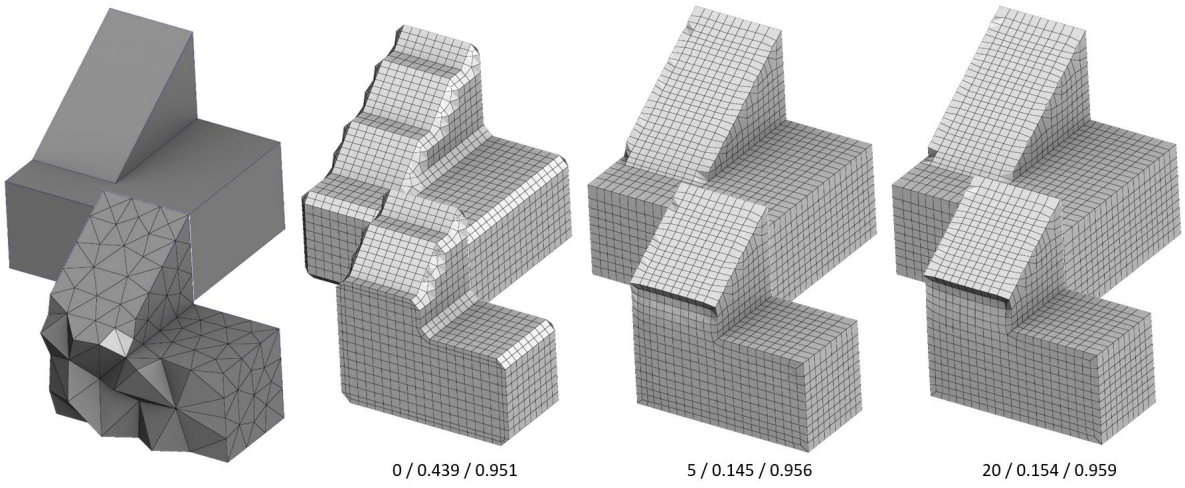


FIGURE 5.23 – Maillage hexaédrique d'un modèle de CAO avec des bords caractéristiques à l'aide d'une grille régulière. (nb itération / min SJ / avg SJ)

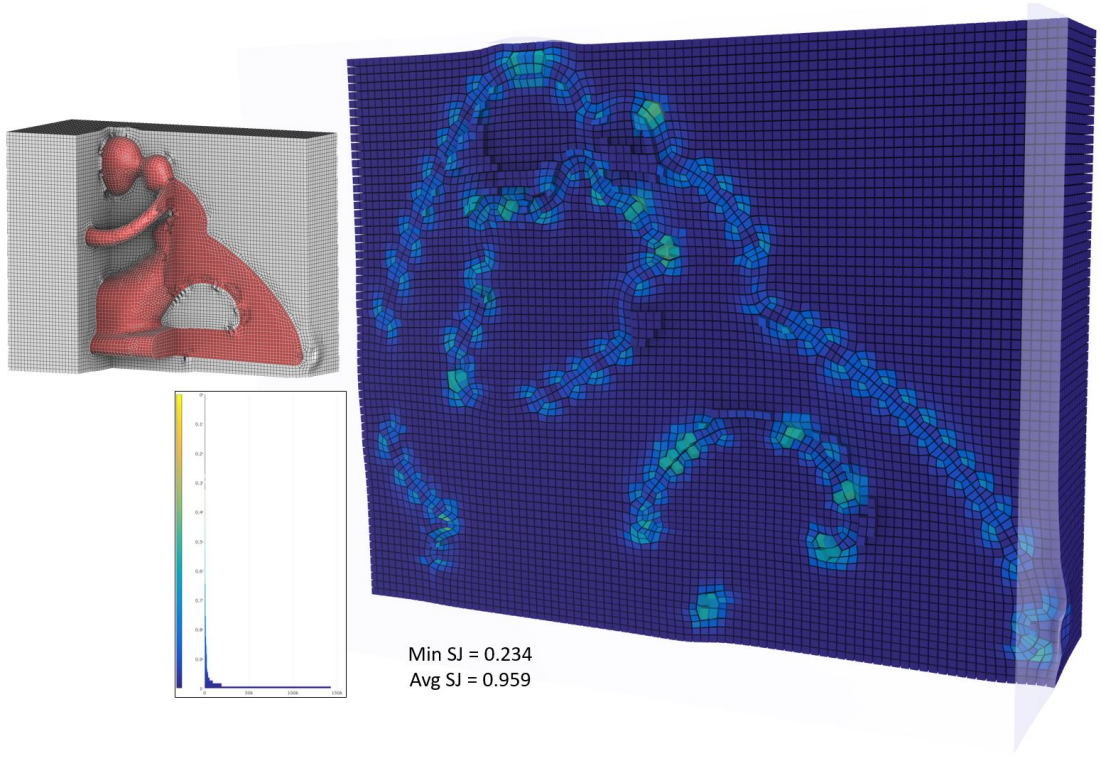


FIGURE 5.24 – Maillage hexaédrique bi-matière du modèle *Fertility* avec la qualité de ses éléments.

préservation des caractéristiques sur les domaines de type CAO. Notez que pour être parfait, le long des courbes caractéristiques, il faudrait introduire des couches d'hexaèdres supplémentaires, comme décrit dans [Maréchal, 2016]. Cette procédure est assez couteuse, et nous espérons trouver une meilleure alternative qui soit globale.

Si le lissage converge très bien dans les cas simples, ce n'est pas le cas pour les cas difficiles. Plus précisément, la coupure ajoutée dans la section précédente peut introduire des configurations très difficiles à lisser, et l'algorithme aura parfois besoin de beaucoup d'itérations très coûteuses. C'est un problème qui reste à résoudre, et nous espérons qu'un meilleur choix de grille permettra de le résoudre.

Variation de la grille

Nous n'avons pas été en mesure de tester la variation de la grille jusqu'à présent. Nous espérons tester sur une grille octree et sur une grille obtenue par une paramétrisation globale qui ne correspond pas complètement au bord.

Maillage multi-matériaux

Un avantage majeur de notre méthode est que, sur la grille, elle donne des maillages intérieur et extérieur compatibles. Ceci est une propriété intéressante pour de nombreuses applications telles que la dynamique des fluides ou la géologie. Nous en donnons un exemple dans la FIGURE 5.24.

Conclusion

Nous proposons une méthode simple et robuste pour extraire un maillage hexaédrique d'une grille. Le maillage correspondra au bord et aux caractéristiques du domaine, et, sous certaines contraintes sur la grille, ne contiendra que des éléments valides. Notre méthode présente encore quelques problèmes tels que la rapidité ou la difficulté à s'adapter à des bords difficiles.

5.4 Perspectives

Les deux méthodes présentées dans ce chapitre illustrent deux angles d'attaque différents pour résoudre le problème de la coloration.

En calculant des champs de croix adaptés pour les modèles de CAO, nous espérons que des travaux futurs produiront des études exhaustives des colorations de polycube, et avec celles-ci, des moyens concrets pour corriger une coloration invalide. Nos champs permettront alors de simplifier l'application de ses correctifs en dessinant, par exemple, des marches de bonne qualité, suivant les caractéristiques clés des modèles de CAO.

Le logiciel *marchinghex*, de son côté, permet d'effectuer une extraction de maillages hexaédriques pour des domaines quelconques. Prendre un domaine, puis le déformer pour que celui-ci soit le plus adapté à la grille semble être un voie de génération robuste de maillage hexaédrique. Cela est, d'une certaine façon, équivalent à produire des déformations partielles en polycube, où les endroits problématiques ont été traités au mieux, mais dont il reste encore des faces non alignées avec les axes. Produire de telles déformations pour maximiser la qualité des hexaèdres finaux reste un problème difficile.

Ces deux approches semblent extensibles au cas des paramétrisations globales. L'obtention de contraintes de bord cohérentes pour le calcul d'un champ de croix 3D nécessite une compréhension poussée du bord du domaine. De la même façon, des déformations partielles sont possible dans l'espace paramétrique, donnant alors la possibilité d'utiliser *marchinghex* pour en extraire des maillages. Bien que, sous sa forme actuelle, cela requiert tout de même des lignes de singularité valides.

Conclusion

Dans ce travail de thèse, nous avons abordé différentes facettes de la méthode des polycubes pour la génération de maillages hexaédriques, pour en combler les lacunes. Chaque étape (coloration, déformation, quantification et inversion) posait initialement des problèmes.

Le chapitre 3 présente notre nouvelle méthode de déformation. Celle-ci représente une contribution importante car elle permet d'efficacement calculer des déformations de bonne qualité, même dans les situations les plus difficiles. Nous illustrons cela sur une importante base de données de problèmes. Au-delà des déformations en Polycube, disposer d'une méthode de ce type est essentiel en Infographie, tant la génération de carte est un outil crucial.

Le chapitre 4 étudie les étapes de quantification et d'inversion à partir d'une carte valide. Nous y proposons une méthode robuste d'extraction d'hexaèdres à partir d'un polycuboid. Notre méthode garantit que le maillage obtenu conserve les propriétés du polycuboid initial, mais nécessite que celui-ci soit bijectif. Des heuristiques proposées permettent de relaxer cette contrainte, mais supportent mal les problèmes de recouvrement globaux du polycuboid (injectif *versus* bijectif).

Pour obtenir une méthode robuste de bout en bout, il reste donc à mentionner la coloration. Le chapitre 5 détaille les problèmes rencontrés à ce sujet. Le problème, assimilable à celui de construire la structure singulière d'une paramétrisation globale, est compliqué. Nous avons privilégié l'étude de solutions pratiques, d'abord en améliorant un outil vital à la compréhension des bords des objets : les champs de croix surfaciques. Nous avons, d'un autre côté, envisagé de relaxer la contrainte "dure" de la coloration, et pour cela nous avons développé le logiciel *marchinghex*.

Le problème théorique reste entier, et rend les méthodes pratiques peu robustes. Une satisfaction tout de même : les résultats de cette thèse permettent de focaliser la recherche en polycube sur la coloration. Les outils produits ont, en effet, une robustesse conséquente, ce qui permet une grande liberté dans les essais de coloration, sans se soucier des étapes suivantes.

Je suis convaincu que le problème de coloration est intrinsèquement lié au problème d'une génération d'un graphe de singularité valide pour une géométrie. Il manque à ce jour des outils mathématiques adéquats pour une bonne compréhension des volumes. Une caractérisation complète des configurations semble difficile pour le moment. Mais cela est aussi le cas en 2D, où l'on est incapable (théoriquement), pour le moment, de produire efficacement des champs de croix dont nous sommes sûrs qu'il sera possible d'en extraire un maillage quadrangulaire. Pour autant, il existe d'excellentes heuristiques pour que cela ne soit pas un problème en pratique. Une meilleure caractérisation des formes usuelles et configurations communes et connues en 3D devrait permettre d'obtenir une robustesse similaire à la 2D.

Bibliographie

[Aigerman et Lipman, 2013] Aigerman, N. et Lipman, Y. (2013). Injective and bounded distortion mappings in 3D. *ACM Trans. Graph.*, 32(4).

[Aigerman et Lipman, 2016] Aigerman, N. et Lipman, Y. (2016). Hyperbolic orbifold tutte embeddings. *ACM Transactions on Graphics*, 35(6):1–14.

[Autodesk, INC., 2019] Autodesk, INC. (2019). Maya.

[Ball, 1976] Ball, J. M. (1976). Convexity conditions and existence theorems in nonlinear elasticity. *Archive for rational mechanics and Analysis*, 63(4):337–403.

[Ball, 1981] Ball, J. M. (1981). Global invertibility of sobolev functions and the interpenetration of matter. *Proceedings of the Royal Society of Edinburgh: Section A Mathematics*, 88(3-4): 315–328.

[Beaufort *et al.*, 2017] Beaufort, P., Lambrechts, J., Henrotte, F., Geuzaine, C. et Remacle, J. (2017). Computing two dimensional cross fields - A PDE approach based on the ginzburg-landau theory. *CoRR*, abs/1706.01344.

[Blender Foundation, 2022] Blender Foundation (2022). Blender.

[Bommes *et al.*, 2013] Bommes, D., Campen, M., Ebke, H.-C., Alliez, P. et Kobbelt, L. (2013). Integer-grid maps for reliable quad meshing. *ACM Transactions on Graphics*, 32(4):1.

[Bommes *et al.*, 2009] Bommes, D., Zimmer, H. et Kobbelt, L. (2009). Mixed-integer quadrangulation. *ACM Trans. Graph.*, 28(3):77:1–77:10.

[Brackbill et Saltzman, 1982] Brackbill, J. et Saltzman, J. (1982). Adaptive zoning for singular problems in two dimensions. *Journal of Computational Physics*, 46(3):342 – 368.

[Branets et Garanzha, 2002] Branets, L. V. et Garanzha, V. A. (2002). Distortion measure of trilinear mapping. Application to 3-D grid generation. *Numerical Linear Algebra with Applications*, 9(6-7):511–526. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nla.302.

[Brückler *et al.*, 2021] Brückler, H., Gupta, O., Mandad, M. et Campen, M. (2021). The 3D motorcycle complex for structured volume decomposition. *CoRR*, abs/2112.05793.

[Bukenberger *et al.*, 2022] Bukenberger, D. R., Tarini, M. et Lensch, H. P. (2022). At-most-hexa meshes. *In Computer Graphics Forum*, volume 41, pages 7–28. Wiley Online Library.

[Campen *et al.*, 2015] Campen, M., Bommes, D. et Kobbelt, L. (2015). Quantized global parametrization. *ACM Transactions on Graphics*, 34(6):1–12.

[Campen *et al.*, 2016] Campen, M., Silva, C. T. et Zorin, D. (2016). Bijective maps from simplicial foliations. *ACM Trans. Graph.*, 35(4).

[Charakhch'yan et Ivanenko, 1997] Charakhch'yan, A. et Ivanenko, S. (1997). A variational form of the winslow grid generator. *Journal of Computational Physics*, 136(2):385–398.

[Chen *et al.*, 2019] Chen, L., Xu, G., Wang, S., Shi, Z. et Huang, J. (2019). Constructing volumetric parameterization based on directed graph simplification of l1 polycube structure from complex shapes. *Computer Methods in Applied Mechanics and Engineering*, 351:422–440.

[Cherchi *et al.*, 2019] Cherchi, G., Alliez, P., Scateni, R., Lyon, M. et Bommes, D. (2019). Selective Padding for Polycube-Based Hexahedral Meshing. *Computer Graphics Forum*, 38(1): 580–591.

[Cherchi *et al.*, 2016] Cherchi, G., Livesu, M. et Scateni, R. (2016). Polycube Simplification for Coarse Layouts of Surfaces and Volumes. *Computer Graphics Forum*, 35(5):11–20.

[Ciarlet et Geymonat, 1982] Ciarlet, P. et Geymonat, G. (1982). Sur les lois de comportement en elasticite non-lineaire compressible. *C.R. Acad.Sci. Paris Ser.II*, 295:423 – 426.

[Coiffier et Corman, 2022] Coiffier, G. et Corman, E. (2022). Seamless Global Parametrization in a Single Optimization. working paper or preprint.

[Collis, 1970] Collis, R. (1970). Lidar. *Applied optics*, 9(8):1782–1788.

[Corman et Crane, 2019] Corman, E. et Crane, K. (2019). Symmetric Moving Frames. *ACM Trans. Graph.*, 38(4):87:1–87:16.

[CPLEX, 2019] CPLEX (2019). ILOG CPLEX Optimization Studio - Overview.

[Crowley, 1962] Crowley, W. (1962). An equipotential zoner on a quadrilateral mesh. *Memo, Lawrence Livermore National Lab*, 5.

[CUBIT, 2022] CUBIT (2022). The CUBIT™ Geometry and Mesh Generation Toolkit.

[Danczyk et Suresh, 2013] Danczyk, J. et Suresh, K. (2013). Finite element analysis over tangled simplicial meshes: Theory and implementation. *Finite Elements in Analysis and Design*, 70-71:57 – 67.

[Dassault Systèmes, 2021] Dassault Systèmes (2021). SolidWorks.

[Dassault Systèmes, 2022] Dassault Systèmes (2022). 3D Precise Mesh.

[De Borst *et al.*, 1988] De Borst, R., Van Den Bogert, P. et Zeilmaker, J. (1988). Modelling and analysis of rubberlike materials. *HERON, 33 (1), 1988.*

[Desobry *et al.*, 2021] Desobry, D., Protais, F., Ray, N., Corman, E. et Sokolov, D. (2021). Frame fields for CAD models. *In International Symposium on Visual Computing*, pages 421–434. Springer.

[Dhondt, 2001] Dhondt, G. (2001). A new automatic hexahedral mesher based on cutting. *International Journal for Numerical Methods in Engineering*, 50(9):2109–2126.

[Diamanti *et al.*, 2014] Diamanti, O., Vaxman, A., Panozzo, D. et Sorkine-Hornung, O. (2014). Designing N-polyvector fields with complex polynomials. *Computer Graphics Forum*, 33(5):1–11.

[Diamanti *et al.*, 2015] Diamanti, O., Vaxman, A., Panozzo, D. et Sorkine-Hornung, O. (2015). Integrable PolyVector Fields. *ACM Trans. Graph.*, 34(4).

[Du *et al.*, 2020] Du, X., Aigerman, N., Zhou, Q., Kovalsky, S. Z., Yan, Y., Kaufman, D. M. et Ju, T. (2020). Lifting simplices to find injectivity. *ACM Transactions on Graphics*, 39(4):120:120:1–120:120:17.

[Dumery *et al.*, 2022] Dumery, C., Protais, F., Mestrallet, S., Bourcier, C. et Ledoux, F. (2022). Evocube: a genetic labeling framework for polycube-maps. *arXiv preprint arXiv:2205.00738.*

[Ebke *et al.*, 2013] Ebke, H.-C., Bommes, D., Campen, M. et Kobbelt, L. (2013). QEx: Robust Quad Mesh Extraction. *ACM Trans. Graph.*, 32(6).

[Eck *et al.*, 1995] Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M. et Stuetzle, W. (1995). Multiresolution analysis of arbitrary meshes. *In Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, page 173–182, New York, NY, USA. Association for Computing Machinery.

[Escobar *et al.*, 2003] Escobar, J. M., Rodriguez, E., Montenegro, R., Montero, G. et González-Yuste, J. M. (2003). Simultaneous untangling and smoothing of tetrahedral meshes. *Computer Methods in Applied Mechanics and Engineering*, 192(25):2775–2787.

[Fang *et al.*, 2018] Fang, X., Bao, H., Tong, Y., Desbrun, M. et Huang, J. (2018). Quadrangulation through morse-parameterization hybridization. *ACM Trans. Graph.*, 37(4).

[Fang *et al.*, 2016] Fang, X., Xu, W., Bao, H. et Huang, J. (2016). All-hex meshing using closed-form induced polycube. *ACM Transactions on Graphics*, 35(4):1–9.

[Floater, 1997] Floater, M. S. (1997). Parametrization and smooth approximation of surface triangulations. *Comput. Aided Geom. Des.*, 14(3):231–250.

[Flory, 1961] Flory, P. J. (1961). Thermodynamic relations for high elastic materials. *Trans. Faraday Soc.*, 57:829–838.

[Folwell et Mitchell, 1999] Folwell, N. T. et Mitchell, S. A. (1999). Reliable Whisker Weaving via Curve Contraction. *Engineering with Computers*, 15(3):292–302.

[Freitag et Plassmann, 2000] Freitag, L. A. et Plassmann, P. (2000). Local optimization-based simplicial mesh untangling and improvement. *International Journal for Numerical Methods in Engineering*, 49(1-2):109–125.

[Fu *et al.*, 2016] Fu, X.-M., Bai, C.-Y. et Liu, Y. (2016). Efficient Volumetric PolyCube-Map Construction. *Computer Graphics Forum*, 35(7):97–106.

[Fu et Liu, 2016] Fu, X.-M. et Liu, Y. (2016). Computing inversion-free mappings by simplex assembly. *ACM Trans. Graph.*, 35(6).

[Gao *et al.*, 2015] Gao, X., Deng, Z. et Chen, G. (2015). Hexahedral Mesh Re-parameterization from Aligned Base-complex. *ACM Trans. Graph.*, 34(4):142:1–142:10.

[Gao *et al.*, 2017a] Gao, X., Jakob, W., Tarini, M. et Panozzo, D. (2017a). Robust hex-dominant mesh generation using field-guided polyhedral agglomeration. *ACM Transactions on Graphics*, 36(4):1–13.

[Gao *et al.*, 2017b] Gao, X., Panozzo, D., Wang, W., Deng, Z. et Chen, G. (2017b). Robust structure simplification for hex re-meshing. *ACM Transactions on Graphics*, 36(6):1–13.

[Gao *et al.*, 2019] Gao, X., Shen, H. et Panozzo, D. (2019). Feature Preserving Octree-Based Hexahedral Meshing. *Computer Graphics Forum*, 38(5):135–149.

[Garanzha, 2000] Garanzha, V. (2000). The barrier method for constructing quasi-isometric grids. *Computational Mathematics and Mathematical Physics*, 40:1617–1637.

[Garanzha et Kaporin, 1999] Garanzha, V. et Kaporin, I. (1999). Regularization of the barrier variational method. *Computational mathematics and mathematical physics*, 39(9):1426–1440.

[Garanzha *et al.*, 2021a] Garanzha, V., Kaporin, I., Kudryavtseva, L., Protais, F., Ray, N. et Sokolov, D. (2021a). Foldover-free maps in 50 lines of code. *ACM Transactions on Graphics*, 40(4).

[Garanzha *et al.*, 2021b] Garanzha, V., Kaporin, I., Kudryavtseva, L., Protais, F., Ray, N. et Sokolov, D. (2021b). On Local Invertibility and Quality of Free-boundary Deformations. *In IMR 2021 - 29th International Meshing Roundtable*, Virtual, United States.

[Garanzha *et al.*, 2022] Garanzha, V., Kaporin, I. E., Kudryavtseva, L. N., Protais, F., Desobry, D. et Sokolov, D. (2022). Practical lowest distortion mapping. *CoRR*, abs/2201.12112.

[Garanzha *et al.*, 2014] Garanzha, V., Kudryavtseva, L. et Utyuzhnikov, S. (2014). Variational method for untangling and optimization of spatial meshes. *Journal of Computational and Applied Mathematics*, 269:24 – 41.

[Garimella, 2002] Garimella, R. V. (2002). Mesh data structure selection for mesh generation and FEA applications. *International journal for numerical methods in engineering*, 55(4):451–478.

[Geuzaine et Remacle, 2009] Geuzaine, C. et Remacle, J.-F. (2009). Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.2579.

[Gregson *et al.*, 2011] Gregson, J., Sheffer, A. et Zhang, E. (2011). All-Hex Mesh Generation via Volumetric PolyCube Deformation. *Computer Graphics Forum*, 30(5):1407–1416.

[Guo *et al.*, 2020] Guo, H.-X., Liu, X., Yan, D.-M. et Liu, Y. (2020). Cut-enhanced PolyCube-maps for feature-aware all-hex meshing. *ACM Transactions on Graphics*, 39(4).

[Gurobi Optimization, LLC, 2022] Gurobi Optimization, LLC (2022). Gurobi Optimizer Reference Manual.

[Hansen et Owen, 2008] Hansen, G. et Owen, S. (2008). Mesh generation technology for nuclear reactor simulation; barriers and opportunities. *Nuclear Engineering and Design*, 238(10):2590–2605.

[He *et al.*, 2009] He, Y., Wang, H., Fu, C.-W. et Qin, H. (2009). A divide-and-conquer approach for automatic polycube map construction. *Computers & Graphics*, 33(3):369–380.

[Hertzmann et Zorin, 2000] Hertzmann, A. et Zorin, D. (2000). Illustrating smooth surfaces. *In PROCEEDINGS OF SIGGRAPH 2000*, pages 517–526.

[Hormann et Greiner, 2000] Hormann, K. et Greiner, G. (2000). MIPS: An Efficient Global Parametrization Method. *In Curve and Surface Design*. Vanderbilt University press.

[Hormann *et al.*, 2007] Hormann, K., Lévy, B. et Sheffer, A. (2007). Mesh parameterization: Theory and practice video files associated with this course are available from the citation page. *In ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, New York, NY, USA. Association for Computing Machinery.

[Hormann *et al.*, 2008] Hormann, K., Polthier, K. et Sheffer, A. (2008). Mesh parameterization: Theory and practice. *In ACM SIGGRAPH ASIA 2008 Courses*, SIGGRAPH Asia '08, New York, NY, USA. Association for Computing Machinery.

[Hu *et al.*, 2018] Hu, Y., Zhou, Q., Gao, X., Jacobson, A., Zorin, D. et Panozzo, D. (2018). Tetrahedral Meshing in the Wild. *ACM Trans. Graph.*, 37(4):60:1–60:14.

[Huang *et al.*, 2014] Huang, J., Jiang, T., Shi, Z., Tong, Y., Bao, H. et Desbrun, M. (2014). l1Basedd Construction of Polycube Maps from Complex Shapes. *ACM Trans. Graph.*, 33(3): 25:1–25:11.

[Huang *et al.*, 2022] Huang, Q., Zhang, W.-X., Wang, Q., Liu, L. et Fu, X.-M. (2022). Untangling all-hex meshes via adaptive boundary optimization. *Graphical Models*, 121:101136.

[Iarussi *et al.*, 2015] Iarussi, E., Bommes, D. et Bousseau, A. (2015). BendFields: Regularized Curvature Fields from Rough Concept Sketches. *ACM Transactions on Graphics*, 34(3).

[Ivanenko, 1988] Ivanenko, S. (1988). Construction of nondegenerate grids. *Zh. Vychisl. Mat. Mat. Fiz*, 28(10):1498.

[Jacquotte, 1988] Jacquotte, O.-P. (1988). A mechanical model for a new grid generation method in computational fluid dynamics. *Computer methods in applied mechanics and engineering*, 66(3):323–338.

[Jakob *et al.*, 2015] Jakob, W., Tarini, M., Panozzo, D. et Sorkine-Hornung, O. (2015). Instant field-aligned meshes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH ASIA)*, 34(6).

[Jiang *et al.*, 2015] Jiang, T., Fang, X., Huang, J., Bao, H., Tong, Y. et Desbrun, M. (2015). Frame field generation through metric customization. *ACM Trans. Graph.*, 34(4).

[Jiang *et al.*, 2017] Jiang, Z., Schaefer, S. et Panozzo, D. (2017). Simplicial complex augmentation framework for bijective maps. *ACM Trans. Graph.*, 36(6).

[Jiang *et al.*, 2020] Jiang, Z., Schneider, T., Zorin, D. et Panozzo, D. (2020). Bijective projection in a shell. *ACM Trans. Graph.*, 39(6).

[Johnen *et al.*, 2017] Johnen, A., Weill, J. C. et Remacle, J. F. (2017). Robust and efficient validation of the linear hexahedral element. *Procedia Engineering*, 203:271–283.

[Kaelberer *et al.*, 2007] Kaelberer, F., Nieser, M. et Polthier, K. (2007). QuadCover - Surface Parameterization using Branched Coverings. *Computer Graphics Forum*.

[Kawamura *et al.*, 2008] Kawamura, Y., Islam, M. S. et Sumi, Y. (2008). A strategy of automatic hexahedral mesh generation by using an improved whisker-weaving method with a surface mesh modification procedure. *Engineering with Computers*, 24(3):215–229.

[Knöppel *et al.*, 2013] Knöppel, F., Crane, K., Pinkall, U. et Schröder, P. (2013). Globally optimal direction fields. *ACM Trans. Graph.*, 32(4).

[Knöppel *et al.*, 2015] Knöppel, F., Crane, K., Pinkall, U. et Schröder, P. (2015). Stripe patterns on surfaces. *ACM Trans. Graph.*, 34(4).

[Knupp, 2000a] Knupp, P. (2000a). Winslow smoothing on two-dimensional unstructured meshes.

[Knupp, 2012] Knupp, P. (2012). Introducing the target-matrix paradigm for mesh optimization via node-movement. *Engineering with Computers*, 28(4):419–429.

[Knupp, 2000b] Knupp, P. M. (2000b). Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part II—a framework for volume mesh optimization and the condition number of the Jacobian matrix. *International Journal for numerical methods in engineering*, 48(8):1165–1185.

[Knupp, 2001] Knupp, P. M. (2001). Hexahedral and tetrahedral mesh untangling. *Engineering with Computers*, 17(3):261–268.

[Knupp, 2003] Knupp, P. M. (2003). A method for hexahedral mesh shape optimization. *International journal for numerical methods in engineering*, 58(2):319–332.

[Koch *et al.*, 2019] Koch, S., Matveev, A., Jiang, Z., Williams, F., Artemov, A., Burnaev, E., Alexa, M., Zorin, D. et Panozzo, D. (2019). ABC: A Big CAD Model Dataset For Geometric Deep Learning. *In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Kovalsky *et al.*, 2015] Kovalsky, S. Z., Aigerman, N., Basri, R. et Lipman, Y. (2015). Large-scale bounded distortion mappings. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH Asia)*, 34(6).

[Kowalski *et al.*, 2014] Kowalski, N., Ledoux, F. et Frey, P. (2014). Block-structured Hexahedral Meshes for CAD Models Using 3D Frame Fields. *Procedia Engineering*, 82:59–71.

[Kowalski *et al.*, 2016] Kowalski, N., Ledoux, F. et Frey, P. (2016). Smoothness driven frame field generation for hexahedral meshing. *Computer-Aided Design*, 72:65–77.

[Kowalski *et al.*, 2012] Kowalski, N., Ledoux, F., Staten, M. L. et Owen, S. J. (2012). Fun sheet matching: towards automatic block decomposition for hexahedral meshes. *Engineering with Computers*, 28(3):241–253.

[Kälberer *et al.*, 2007] Kälberer, F., Nieser, M. et Polthier, K. (2007). QuadCover - Surface Parameterization using Branched Coverings. *Computer Graphics Forum*, 26(3):375–384.

[Li *et al.*, 2013] Li, B., Li, X., Wang, K. et Qin, H. (2013). Surface Mesh to Volumetric Spline Conversion with Generalized Polycubes. *IEEE Transactions on Visualization and Computer Graphics*, 19(9):1539–1551.

[Li *et al.*, 2020] Li, M., Ferguson, Z., Schneider, T., Langlois, T., Zorin, D., Panozzo, D., Jiang, C. et Kaufman, D. M. (2020). Incremental potential contact: Intersection-and inversion-free, large-deformation dynamics. *ACM Trans. Graph.*, 39(4).

[Li *et al.*, 1995] Li, T., McKeag, R. et Armstrong, C. (1995). Hexahedral meshing using midpoint subdivision and integer programming. *Computer methods in applied mechanics and engineering*, 124(1-2):171–193.

[Li *et al.*, 2012] Li, Y., Liu, Y., Xu, W., Wang, W. et Guo, B. (2012). All-hex meshing using singularity-restricted field. *ACM Trans. Graph.*, 31(6).

[Linder, 2009] Linder, W. (2009). *Digital photogrammetry*, volume 1. Springer.

[Lipman, 2012] Lipman, Y. (2012). Bounded distortion mapping spaces for triangular meshes. *ACM Trans. Graph.*, 31(4).

[Liu *et al.*, 2018] Liu, H., Zhang, P., Chien, E., Solomon, J. et Bommes, D. (2018). Singularity-constrained octahedral fields for hexahedral meshing. *ACM Trans. Graph.*, 37(4).

[Liu *et al.*, 2011] Liu, Y., Xu, W., Wang, J., Zhu, L., Guo, B., Chen, F. et Wang, G. (2011). General planar quadrilateral mesh design using conjugate direction field. *ACM Trans. Graph.*, 30(6):1 ?10.

[Livesu *et al.*, 2022] Livesu, M., Pitzalis, L. et Cherchi, G. (2022). Optimal Dual Schemes for Adaptive Grid Based Hexmeshing. *ACM Transactions on Graphics*, 41(2):1–14.

[Livesu *et al.*, 2015] Livesu, M., Sheffer, A., Vining, N. et Tarini, M. (2015). Practical hex-mesh optimization via edge-cone rectification. *ACM Transactions on Graphics*, 34(4):1–11.

[Livesu *et al.*, 2013] Livesu, M., Vining, N., Sheffer, A., Gregson, J. et Scateni, R. (2013). PolyCut: monotone graph-cuts for PolyCube base-complex construction. *ACM Transactions on Graphics*, 32(6):1–12.

[Lorensen et Cline, 1987] Lorensen, W. E. et Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4): 163–169.

[Lyon *et al.*, 2016] Lyon, M., Bommes, D. et Kobbelt, L. (2016). HexEx: robust hexahedral mesh extraction. *ACM Transactions on Graphics*, 35(4):1–11.

[Lyon *et al.*, 2019] Lyon, M., Campen, M., Bommes, D. et Kobbelt, L. (2019). Parametrization quantization with free boundaries for trimmed quad meshing. *ACM Transactions on Graphics*, 38(4):51:1–51:14.

[Lyon *et al.*, 2021] Lyon, M., Campen, M. et Kobbelt, L. (2021). Quad Layouts via Constrained T-Mesh Quantization. page 10.

[Lévy *et al.*, 2002] Lévy, B., Petitjean, S., Ray, N. et Maillot, J. (2002). Least Squares Conformal Maps for Automatic Texture Atlas Generation. *In Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 362–371, New York, NY, USA. ACM. event-place: San Antonio, Texas.

[Marschner *et al.*, 2020] Marschner, Z., Palmer, D., Zhang, P. et Solomon, J. (2020). Hexahedral Mesh Repair via Sum-of-Squares Relaxation. *Computer Graphics Forum*, 39(5): 133–147.

[Maréchal, 2009] Maréchal, L. (2009). Advances in Octree-Based All-Hexahedral Mesh Generation: Handling Sharp Features. *In* Clark, B. W., éditeur : *Proceedings of the 18th International Meshing Roundtable*, pages 65–84. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Maréchal, 2016] Maréchal, L. (2016). All Hexahedral Boundary Layers Generation. *Procedia Engineering*, 163:5–19.

[Moré et Thuente, 1994] Moré, J. J. et Thuente, D. J. (1994). Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Softw.*, 20(3):286–307.

[Myles *et al.*, 2014] Myles, A., Pietroni, N. et Zorin, D. (2014). Robust Field-aligned Global Parametrization. *ACM Trans. Graph.*, 33(4):135:1–135:14.

[Myles et Zorin, 2013] Myles, A. et Zorin, D. (2013). Controlled-distortion Constrained Global Parametrization. *ACM Trans. Graph.*, 32(4):105:1–105:14.

[Naitsat *et al.*, 2020] Naitsat, A., Zhu, Y. et Zeevi, Y. Y. (2020). Adaptive block coordinate descent for distortion optimization. *In Computer Graphics Forum*, volume 39, pages 360–376. Wiley Online Library.

[Nieser *et al.*, 2011] Nieser, M., Reitebuch, U. et Polthier, K. (2011). CubeCover - Parameterization of 3D Volumes. *Computer Graphics Forum*.

[Owen, 2016] Owen, S. (2016). An Introduction to Automatic Mesh Generation Algorithms. *In Short Course*, page 236, Washington, DC.

[Owen *et al.*, 1999] Owen, S., Staten, M., Canann, S. et Saigal, S. (1999). Q-Morph: An indirect approach to advancing front quad meshing. *International Journal for Numerical Methods in Engineering*, 44(9):1317–1340.

[Owen et Saigal, 2000] Owen, S. J. et Saigal, S. (2000). H-Morph: an indirect approach to advancing front hex meshing. *International Journal for Numerical Methods in Engineering*, 49(1-2):289–312. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/1097-0207%2820000910/20%2949%3A1/2%3C289%3A%3AAID-NME934%3E3.0.CO%3B2-L.

[Palacios et Zhang, 2007] Palacios, J. et Zhang, E. (2007). Rotational symmetry field design on surfaces. *ACM Trans. Graph.*, 26(3).

[Panozzo *et al.*, 2014] Panozzo, D., Puppo, E., Tarini, M. et Sorkine-Hornung, O. (2014). Frame fields: Anisotropic and non-orthogonal cross fields. *ACM Trans. Graph.*, 33(4).

[Pellerin *et al.*, 2018] Pellerin, J., Verhetsel, K. et Remacle, J.-F. (2018). There are 174 subdivisions of the hexahedron into tetrahedra. *ACM Transactions on Graphics*, 37(6):266:1–266:9.

[Penn, 1970] Penn, R. W. (1970). Volume changes accompanying the extension of rubber. *Transactions of the Society of Rheology*, 14(4):509–517.

[Pietroni *et al.*, 2022] Pietroni, N., Campen, M., Sheffer, A., Cherchi, G., Bommes, D., Gao, X., Scateni, R., Ledoux, F., Remacle, J.-F. et Livesu, M. (2022). Hex-Mesh Generation and Processing: a Survey. *arXiv:2202.12670 [cs]*. arXiv: 2202.12670.

[Pitzalis *et al.*, 2021] Pitzalis, L., Livesu, M., Cherchi, G., Gobbetti, E. et Scateni, R. (2021). Generalized adaptive refinement for grid-based hexahedral meshing. *ACM Transactions on Graphics*, 40(6):1–13.

[Prokhorova, 2008] Prokhorova, M. F. (2008). Problems of homeomorphism arising in the theory of grid generation. *Proceedings of the Steklov Institute of Mathematics*, 261(1):165–182.

[Protais *et al.*, 2022] Protais, F., Reberol, M., Ray, N., Corman, E., Ledoux, F. et Sokolov, D. (2022). Robust quantization for polycube maps. *Computer-Aided Design (SPM)*, page 103321.

[Rabinovich *et al.*, 2017] Rabinovich, M., Poranne, R., Panozzo, D. et Sorkine-Hornung, O. (2017). Scalable locally injective mappings. *ACM Trans. Graph.*, 36(2).

[Ray *et al.*, 2006] Ray, N., Li, W. C., Lévy, B., Sheffer, A. et Alliez, P. (2006). Periodic Global Parameterization. *ACM Trans. Graph.*, 25(4):1460–1485.

[Ray et Sokolov, 2014] Ray, N. et Sokolov, D. (2014). Robust Polylines Tracing for N-Symmetry Direction Field on Triangulated Surfaces. *ACM Transactions on Graphics*, 33(3): 30:1–30:11.

[Ray et Sokolov, 2015] Ray, N. et Sokolov, D. (2015). On smooth 3D frame field design. *CoRR*, abs/1507.03351.

[Ray *et al.*, 2016] Ray, N., Sokolov, D. et Lévy, B. (2016). Practical 3D frame field generation. *ACM Transactions on Graphics*, 35(6):1–9.

[Ray *et al.*, 2018] Ray, N., Sokolov, D., Reberol, M., Ledoux, F. et Lévy, B. (2018). Hex-dominant meshing: Mind the gap ! *Computer-Aided Design*, 102:94–103.

[Ray *et al.*, 2009] Ray, N., Vallet, B., Alonso, L. et Levy, B. (2009). Geometry-aware direction field processing. *ACM Transactions on Graphics*, 29(1):1–11.

[Ray *et al.*, 2008] Ray, N., Vallet, B., Li, W.-C. et Lévy, B. (2008). N-Symmetry Direction Field Design. *ACM Transactions on Graphics*, 27(2):Article 10.

[Remacle *et al.*, 2012] Remacle, J.-F., Lambrechts, J., Seny, B., Marchandise, E., Johnen, A. et Geuzainet, C. (2012). Blossom-Quad: A non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm. *International Journal for Numerical Methods in Engineering*, 89(9):1102–1119. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.3279.

[Reshetnyak, 1966] Reshetnyak, Y. G. (1966). Bounds on moduli of continuity for certain mappings. *Siberian Mathematical Journal*, 7(5):879–886.

[Ruiz-Gironés *et al.*, 2014] Ruiz-Gironés, E., Roca, X. et Sarrate, J. (2014). Optimizing mesh distortion by hierarchical iteration relocation of the nodes on the CAD entities. *Procedia Engineering*, 82:101–113.

[Schneiders, 1996] Schneiders, R. (1996). A grid-based algorithm for the generation of hexahedral element meshes. *Engineering with Computers*, 12(3):168–177.

[Schüller *et al.*, 2013] Schüller, C., Kavan, L., Panozzo, D. et Sorkine-Hornung, O. (2013). Locally injective mappings. *Computer Graphics Forum (proceedings of Symposium on Geometry Processing)*, 32(5).

[Scott *et al.*, 2006] Scott, M. A., Benzley, S. E. et Owen, S. J. (2006). Improved many-to-one sweeping. *International Journal for Numerical Methods in Engineering*, 65(3):332–348. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.1444.

[Shen *et al.*, 2019] Shen, H., Jiang, Z., Zorin, D. et Panozzo, D. (2019). Progressive embedding. *ACM Trans. Graph.*, 38(4).

[Shepherd *et al.*, 2006] Shepherd, J. F., Tuttle, C. J., Silva, C. et Zhang, Y. (2006). Quality improvement and feature capture in hexahedral meshes. *The University of Utah, Tech. Rep. UUSCI-2006-029*.

[Shtengel *et al.*, 2017] Shtengel, A., Poranne, R., Sorkine-Hornung, O., Kovalsky, S. Z. et Lipman, Y. (2017). Geometric optimization via composite majorization. *ACM Trans. Graph.*, 36(4).

[Si, 2015] Si, H. (2015). TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Trans. Math. Softw.*, 41(2):11:1–11:36.

[Siemens Digital Industries Software, 2021] Siemens Digital Industries Software (Siemens 2021). Simcenter STAR-CCM+ User Guide v. 2021.1.

[Smith *et al.*, 2019] Smith, B., Goes, F. D. et Kim, T. (2019). Analytic eigensystems for isotropic distortion energies. *ACM Trans. Graph.*, 38(1).

[Smith et Schaefer, 2015] Smith, J. et Schaefer, S. (2015). Bijective parameterization with free boundaries. *ACM Trans. Graph.*, 34(4).

[Sokolov et Ray, 2015] Sokolov, D. et Ray, N. (2015). Fixing normal constraints for generation of polycubes. page 19.

[Sokolov *et al.*, 2016] Sokolov, D., Ray, N., Untereiner, L. et Lévy, B. (2016). Hexahedral-Dominant Meshing. *ACM Transactions on Graphics*, 35(5):1–23.

[Sorkine et Alexa, 2007] Sorkine, O. et Alexa, M. (2007). As-rigid-as-possible surface modeling. *In Symposium on Geometry processing*, volume 4, pages 109–116.

[Staten *et al.*, 2010] Staten, M. L., Shepherd, J. F., Ledoux, F. et Shimada, K. (2010). Hexahedral Mesh Matching: Converting non-conforming hexahedral-to-hexahedral interfaces into conforming interfaces: HEXAHEDRAL MESH MATCHING. *International Journal for Numerical Methods in Engineering*, 82(12):1475–1509.

[Su *et al.*, 2019] Su, J.-P., Fu, X.-M. et Liu, L. (2019). Practical foldover-free volumetric mapping construction. *Computer Graphics Forum*, 38(7):287–297.

[Su *et al.*, 2020] Su, J.-P., Ye, C., Liu, L. et Fu, X.-M. (2020). Efficient bijective parameterizations. *ACM Trans. Graph.*, 39(4).

[Surazhsky *et al.*, 2003] Surazhsky, V., Alliez, P. et Gotsman, C. (2003). Isotropic Remeshing of Surfaces: a Local Parameterization Approach. Research Report RR-4967, INRIA.

[Tarini *et al.*, 2004] Tarini, M., Hormann, K., Cignoni, P. et Montani, C. (2004). PolyCube-Maps. *In ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 853–860, New York, NY, USA. ACM. event-place: Los Angeles, California.

[Toulorge *et al.*, 2013] Toulorge, T., Geuzaine, C., Remacle, J.-F. et Lambrechts, J. (2013). Robust untangling of curvilinear meshes. *Journal of Computational Physics*, 254:8–26.

[Tutte, 1963] Tutte, W. T. (1963). How to Draw a Graph. *Proceedings of the London Mathematical Society*, s3-13(1):743–767.

[Vaxman *et al.*, 2017] Vaxman, A., Campen, M., Diamanti, O., Bommes, D., Hildebrandt, K., Technion, M. B.-C. et Panozzo, D. (2017). Directional field synthesis, design, and processing. *In ACM SIGGRAPH 2017 Courses*, SIGGRAPH '17, New York, NY, USA. Association for Computing Machinery.

[Verhetsel *et al.*, 2019a] Verhetsel, K., Pellerin, J. et Remacle, J.-F. (2019a). A 44-Element Mesh of Schneiders' Pyramid. *In* Roca, X. et Loseille, A., éditeurs : *27th International Meshing Roundtable*, Lecture Notes in Computational Science and Engineering, pages 73–87. Springer International Publishing, Cham.

[Verhetsel *et al.*, 2019b] Verhetsel, K., Pellerin, J. et Remacle, J.-F. (2019b). Finding hexahedrizations for small quadrangulations of the sphere. *ACM Transactions on Graphics*, 38(4):53:1–53:13.

[Voss *et al.*, 2021] Voss, J., Ghiba, I.-D., Martin, R. J. et Neff, P. (2021). Sharp rank-one convexity conditions in planar isotropic elasticity for the additive volumetric-isochoric split. *Journal of Elasticity*, 143(2):301–335.

[Wang *et al.*, 2018] Wang, R., Gao, S., Zheng, Z. et Chen, J. (2018). Hex mesh topological improvement based on frame field and sheet adjustment. *Computer-Aided Design*, 103:103–117.

[Wang *et al.*, 2021] Wang, R., Zheng, Z., Yu, W., Shao, Y. et Gao, S. (2021). Structure-aware geometric optimization of hexahedral mesh. *Computer-Aided Design*, 138:103050.

[Weber *et al.*, 2012] Weber, O., Myles, A. et Zorin, D. (2012). Computing extremal quasiconformal maps. *Computer Graphics Forum*, 31(5):1679–1689.

[Weber et Zorin, 2014] Weber, O. et Zorin, D. (2014). Locally injective parametrization with arbitrary fixed boundaries. *ACM Trans. Graph.*, 33(4).

[Weill et Ledoux, 2016] Weill, J. C. et Ledoux, F. (2016). Towards an automatic and reliable hexahedral meshing. Liège.

[Wilson *et al.*, 2012] Wilson, T. J., Sarrate Ramos, J., Roca Ramón, X., Montenegro, R. et Escobar, J. (2012). Untangling and smoothing of quadrilateral and hexahedral meshes. *Civil-Comp Proceedings*.

[Winslow, 1966] Winslow, A. M. (1966). Numerical solution of the quasilinear poisson equation in a nonuniform triangle mesh. *Journal of computational physics*, 1(2):149–172.

[Yamakawa et Shimada, 2010] Yamakawa, S. et Shimada, K. (2010). 88-Element solution to Schneiders' pyramid hex-meshing problem. *International Journal for Numerical Methods in Biomedical Engineering*, 26(12):1700–1712. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cnm.1256.

[Yang *et al.*, 2019] Yang, Y., Fu, X. et Liu, L. (2019). Computing Surface PolyCube-Maps by Constrained Voxelization. *Computer Graphics Forum*, 38(7):299–309.

[Ye *et al.*, 2020] Ye, C., Su, J.-P., Liu, L. et Fu, X.-M. (2020). Memory-efficient bijective parameterizations of very-large-scale models. *Computer Graphics Forum*, 39(7):1–12.

[Yerry et Shephard, 1984] Yerry, M. A. et Shephard, M. S. (1984). Automatic three-dimensional mesh generation by the modified-octree technique. *International Journal for Numerical Methods in Engineering*, 20(11):1965–1990.

[Zhao *et al.*, 2018] Zhao, H., Lei, N., Li, X., Zeng, P., Xu, K. et Gu, X. (2018). Robust edge-preserving surface mesh polycube deformation. *Computational Visual Media*, 4(1):33–42.

[Zhao *et al.*, 2019] Zhao, H., Li, X., Wang, W., Wang, X., Wang, S., Lei, N. et Gu, X. (2019). Polycube Shape Space. *Computer Graphics Forum*, 38(7):311–322.

[Zhou et Jacobson, 2016] Zhou, Q. et Jacobson, A. (2016). Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*.

[Zhu *et al.*, 1997] Zhu, C., Byrd, R. H. et Nocedal, J. (1997). L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560.

[Zhu *et al.*, 2018] Zhu, Y., Bridson, R. et Kaufman, D. M. (2018). Blended cured quasi-newton for distortion optimization. *ACM Trans. Graph.*, 37(4).