Échantillonnage pour l'approximation de fonctions sur des maillages

Vincent Nivoliers

Université de Lorraine
école des Mines de Nancy
école doctorale IAEM

Thèse

pour obtenir le grade de

docteur de l'Université de Lorraine

en informatique

réalisée sous la direction de

Bruno Lévy

par

Vincent Nivoliers

Échantillonnage pour l'approximation de fonctions sur des maillages

---

Thèse soutenue le : 30 novembre 2012

Composition du jury :

M. Bruno Lévy	Inria Nancy Grand Est, directeur de thèse
M. Jean-Michel Dischler	Université de Strasbourg, rapporteur
M. Jean-François Remacle	Université Catholique de Louvain, rapporteur
M. Christophe Geuzaine	Université de Liège, examinateur
Mme. Françoise Simonot-Lion	Université de Lorraine, examinatrice
M. Paul Zimmermann	Inria Nancy Grand Est, examinateur
M. Mark Loriot	Distene SAS, invité

LORIA, 615 Rue du Jardin Botanique, 54600 Villers-lès-Nancy, France

À toi là bas,
à toi ici,
ça va.

Remerciements

Cette thèse doit beaucoup à de nombreuses personnes, qui m'ont entouré et choyé durant les quelques années de cette expérience. Je commencerai tout d'abord par remercier Bruno Lévy, qui a su me faire venir à Nancy, et m'impliquer dans les projets variés et stimulants qui constituent ce document. Merci également de m'avoir encouragé à m'investir dans la médiation scientifique, ce qui m'a beaucoup apporté. Je tiens aussi à remercier tous les membres de mon jury, Jean-Michel Dischler et Jean-François Remacle pour leur relecture, Françoise Simonot-Lion, Christophe Geuzaine, Mark Loriot et Paul Zimmermann en tant qu'examinateurs. Je voudrais notamment souligner le rôle de Paul Zimmermann en tant que référent de thèse, qui a suivi avec soin l'évolution de mes travaux.

Il m'a fallu du temps pour assimiler la Nicolangue, j'ai pu être un peu rétif, mais je sais désormais apprécier le franc-parler de Nicolas Ray, sa sérénité dans son métier, et surtout la qualité de ses avis. Je salue également Laurent Alonso, en particulier pour le travail de relecture attentive qu'il a réalisé sur ce manuscrit. Un clin d'œil aussi à Nicolas Saugnier et Bruno Jobard en souvenir du tiroir à goûter et de nos discussions éclectiques voire électroniques. Merci à Dongming et Sylvain pour les collaborations agréables que nous avons eues, et à Isabelle pour son assistance sans faille. Merci enfin plus généralement à toute l'équipe Alice qui m'a accueilli chaleureusement, je pense notamment aux verres que nous avons partagés, et aux quelques parties de tennis de table que nous avons disputées. Enfin je voudrais saluer Cédric Gérot, Victor Ostromoukhov et Neil Stewart, pour m'avoir lancé sur la voie de la recherche.

La médiation scientifique m'a apporté beaucoup de motivation. Je tiens en particulier à saluer le travail de Véronique Poirel, Olivia Brenner, Sabah Khalfa et Maxime Harquet, qui m'ont proposé de nombreuses interventions. Je voudrais aussi remercier Louisette Hiriart et Christelle Kunc pour m'avoir accueilli au sein du collège Chepfer, et l'ensemble des élèves des ateliers Maths en Jeans auxquels j'ai participé. Merci aussi à Julie Scholler qui m'a donné le coup de pouce nécessaire à cette rencontre.

Dans cette véritable confiture, non pas de rhubarbe, mais de rubriques, j'aimerais que figurassent la compagnie Omnibus, animée par Chantal Puccio, et la compagnie du Nain Jaune animée par Christine Vallon. Merci Chantal pour ta franchise, ton écoute, la bonne humeur et le dynamisme que tu sais insuffler à la compagnie. Merci Christine pour ta gentillesse, ton dévouement et ta ténacité pour mener notre projet au bout. Merci à Mme Smut, ma sœur et son mari, Monsieur le directeur et son gardien, André, la Femelle, Jean-Claude Suco, Aïcha, Martin Bournier, le personnel du centre des chasseurs anonymes de Haute Loire, Madame Gramédoire, Étienne Fallot, les bouchers, les volaillers, les charcutiers, les tripiers, et tous les lapins morts. Merci à Julien, Marie, Laura, au Président, au Docteur S. et son assistante, et au Mage Radjapour pour qui j'ai une pensée émue. Merci à tous leurs

interprètes, pour nos bonnes bases, nos fous rires, pour ces instants fugaces et impérissables passés sur les planches en votre compagnie.

De nombreux noms ont côtoyé le mien sur la sonnette, et je voudrais remercier Martin pour son rire communicatif, Anne-Charlotte d'avoir survécu, Enzo et Léo de m'avoir prouvé qu'il était possible de faire entrer quatre-vingts personnes dans un appartement en une soirée, et de les mettre dehors avant l'arrivée de la cavalerie, Iñaki pour les nombreuses formules de courtoisie qu'il m'a enseignées, Charly pour avoir fait de moi un vandale, Claire pour son sens de l'humour et son aide dans la quête des médailles, et enfin Maxime pour les recettes sobres et diététiques que nous avons confectionnées et affectionnées.

J'oublierai fatalement de saluer certaines des amitiés que j'ai pu lier tout au long de cette thèse, et je vous prie de me pardonner si tel est votre cas. Je n'omettrai cependant pas les habitués des pique-niques du mercredi, Julie, Arnaud, Ghislain, Lucas, Michaël, Armand, Sylvain, et leurs successeurs qui perpétuent aujourd'hui ces moments. Je salue Sonia qui m'a rempli la panse de douceurs non végétariennes et toute la troupe du Porkidor. Merci à Jérémie de m'avoir aidé à éliminer tout ça, et pour les luttes auxquelles nous avons participé ensemble. Merci à Maxime, Nicolas, Aurélia, Mathieu, Isabelle, Lolita, Tif, Claire, Élodie, Jean-Charles, et Pierrot pour m'avoir reçu, m'avoir appris à recevoir, avoir partagé mon goût immodéré des jeux, et pour toutes nos sorties diverses et variées. Merci à ces amis éloignés qui ont su rester proches depuis des années, Damien, Aline, Jean-Cyp, Yona, Julien, Gauthier et Grégoire, en particulier à ceux qui ont pu être présents lors de la soutenance et qui m'ont été d'une aide inestimable.

Rien de tout ceci n'aurait pu avoir lieu sans le soutien et l'amour de mes proches. Je remercie mes parents, pour leur générosité, leur courage, et pour cette curiosité qu'il m'ont transmise. Merci à mon grand frère pour sa droiture, son sens des responsabilités, et pour le soutien régulier qu'il m'a témoigné. Merci à mon petit frère pour sa ténacité, sa joie de vivre, et ses encouragements. Merci à mes tantes, mes oncles, mes grands parents, ma belle sœur et ma belle famille pour toute l'affection dont ils m'entourent, et pour le réconfort qu'il nous ont apporté dans les heures difficiles que nous avons vécues. Je souhaiterais également inclure Aline B., Françoise, Éric, Marie-Jo et Monique à mes remerciements pour les avoir traversées à nos côtés.

Le point final ne saurait être placé sans Aline. Alinéa de ce chapitre amorcé voilà sept ans, ce point que je place aujourd'hui te doit beaucoup. Sept ans que notre histoire s'écrit à deux plumes, et que le tintement de tes voyelles fait sonner juste l'enchevêtrement de mes consonnes. Un nouveau paragraphe débute, à nous de le parer de virgules et de points d'exclamation !

Table des matières

Introduction

Fonctions sur des surfaces

Les travaux décrits dans cette thèse s'articulent autour de l'idée commune de la représentation informatique des objets tridimensionnels. Autrement dit, il s'agit d'enregistrer dans un ordinateur un objet de tous les jours. Les ordinateurs ne peuvent manipuler que des nombres (voire du texte en enregistrant les lettres comme des nombres), c'est donc sous la forme d'une liste de nombres qu'il sera nécessaire d'enregistrer l'objet, d'où le terme *numériser*. Pour enregistrer de tels objets, de nombreuses représentations existent. Le choix de la représentation utilisée dépend de l'application pour laquelle les objets seront utilisés, et du procédé suivi pour numériser l'objet. Dans le cadre de cette thèse, nous nous intéresserons à des *surfaces* représentées par des *maillages* surfaciques. D'une manière générale, ce format est pratique lorsqu'on ne s'intéresse pas à l'intérieur d'un objet. Par exemple, si nous souhaitions numériser la statue du David de Michel-Ange pour pouvoir la restaurer par la suite en cas de dégradation, l'intérieur de la statue est uniformément constitué de marbre et n'aura que peu d'intérêt. Nous pouvons alors nous contenter d'enregistrer la surface de la statue, et de noter par ailleurs qu'elle est en marbre, nous serons alors en mesure de la reproduire. Il n'est en général pas envisageable d'enregistrer les positions de tous les points de la surface d'un objet, nous ne pourrions pas les dénombrer. Un maillage n'enregistre la position que de quelques points judicieusement répartis sur la surface de l'objet. Nous verrons en Chapitre 4 ce que veut dire « judicieusement ». Des triangles (ou plus généralement des polygones) relient ensuite les points entre eux pour former l'objet numérique. Un maillage représentant le David est illustré sur la Figure 1.

Les maillages sont principalement utilisés en synthèse d'images et en ingénierie mécanique. La synthèse d'image consiste à dessiner à l'aide d'un ordinateur. Elle intervient par exemple dans les jeux vidéo pour afficher les personnages manipulés par le joueur, et le monde dans lequel ils évoluent. Ces personnages, lorsqu'ils sont dits « en 3D » sont souvent

FIGURE 1 – Maillage représentant la surface du David de Michel-Ange. Cette statue a été numérisée par Levoy *et al.* [61], en utilisant un scanner 3D. Le maillage est constitué de petits triangles accolés les uns aux autres.

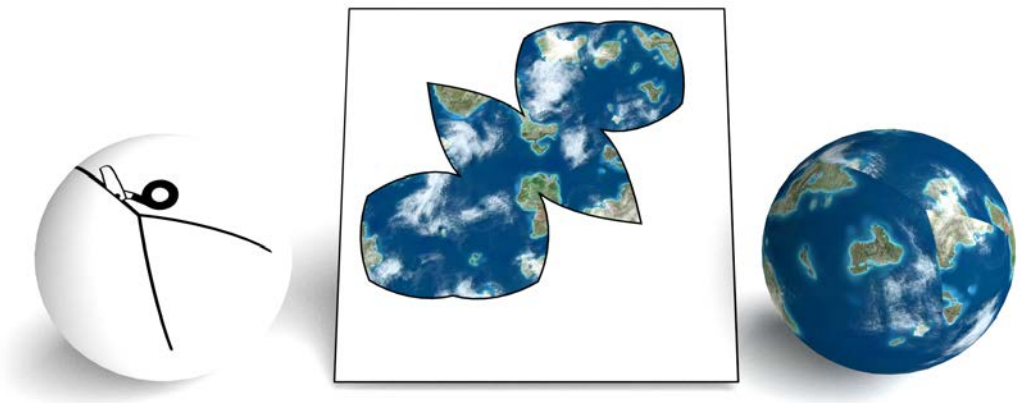


FIGURE 2 – Placage de texture sur une sphère. La sphère est découpée, mise à plat sur une image dans laquelle les couleurs sont enregistrées. Il est ainsi possible de représenter l'ensemble des couleurs de la sphère. Aux endroits où les découpes ont été effectuées, les couleurs de part et d'autre ne se correspondent pas, et les coupures sont visibles dans le résultat. Ce sont les coutures. Nous proposerons en Chapitre 2 une méthode pour remédier à ce problème.

enregistrés sous forme de maillages, et ont été la plupart du temps fabriqués directement sur un ordinateur via un logiciel de modélisation 3D. Dans le domaine de l'ingénierie mécanique, disposer d'objets numériques permet par exemple de déterminer de quelle manière une voiture se pliera lors d'un accident, sans avoir à construire effectivement la voiture pour réaliser le test grandeur nature, et en ayant à disposition autant de voitures que nécessaire car la copie des données ne coûte rien. Dans cette application, pour réaliser le calcul de la déformation de la voiture, les objets sont représentés par des maillages.

Sur ces surfaces, nous chercherons à représenter des fonctions. Une fonction est un objet mathématique défini sur un *domaine*, ici une surface. À chaque point d'une surface, une fonction associe simplement une valeur. Imaginons que ce ne soit plus le David de Michel-Ange qu'il nous faille numériser pour le restaurer, mais la Femme et Oiseau de Joan Miró. Une surface nous permettra certes de restaurer la forme de cette œuvre, mais il lui manquera encore les couleurs. Il nous faut donc associer à chaque point de la surface la couleur qu'elle doit avoir, il s'agit donc d'une fonction. Nous devons ainsi en plus de la surface trouver une manière de numériser une fonction définie sur cette surface. La représentation de couleurs sur une surface est un problème très classique, que nous aborderons dans la Partie I de cette thèse. Ce problème est notamment très présent en synthèse d'image où les personnages manipulés sont également en général colorés. Dans ce domaine, la technique la plus couramment utilisée est celle du *placage de textures*, qui consiste à découper puis déplier la surface pour la mettre à plat sur une image. La couleur d'un point de la surface est enregistrée dans l'image, à l'endroit où le point se trouve une fois la surface dépliée. Sur la Figure 2, nous illustrons ce procédé sur un objet simple. Les endroits où la surface a été découpée sont visibles, car les couleurs correspondant aux points de part et d'autre des coupures ne sont pas voisines dans l'image, et ne se correspondent pas. Il s'agit du problème des *coutures*.

La représentation de fonctions définies sur une surface est également essentielle dans le domaine de l'ingénierie. Dans l'exemple de la simulation d'accident de voiture que nous

avons déjà évoqué, les calculs effectués ont besoin d'un certain nombre de données telles que la rigidité et la robustesse des matériaux utilisés dans la carrosserie, afin de déterminer de quelle manière ils plieront, et à quel moment ils casseront. Le résultat de la simulation se présente également sous la forme de fonctions définies sur la surface, comme la vitesse qu'aura chaque point de la voiture au cours de l'accident ou les forces auxquelles il sera soumis. La méthode utilisée classiquement pour enregistrer ces fonctions est celle des *éléments finis*, qui consiste à ne stocker des valeurs que pour un nombre réduit de points de la surface. La valeur en un point quelconque de la surface sera ensuite extrapolée à partir des valeurs enregistrées aux alentours.

Échantillonnage et approximation

Toutes les méthodes de numérisation d'objets ou de fonctions définies sur ces objets que nous avons présentées sont fondées sur la notion d'*échantillonnage*. Lorsque nous numérisons le David sous forme d'un maillage, nous n'enregistrons la position que d'un nombre réduit de points de sa surface. Ces points sont les *échantillons*. Les triangles qui sont ensuite ajoutés entre ces points permettent d'extrapoler une surface à partir de ces données incomplètes, qui ressemble à la surface originale du David mais n'y correspond pas exactement. Il s'agit d'une *approximation*. D'une manière générale, plus le nombre d'échantillons enregistrés est élevé, plus l'approximation pourra être précise, mais plus le volume des données à enregistrer sera important. La notion de *précision* d'un échantillonnage n'a cependant pas de définition universelle et dépend en général de l'application visée. Typiquement, imaginons que nous souhaitions créer une carte du relief d'un pays. Il sera alors intéressant d'avoir plus d'échantillons enregistrant l'altitude dans les zones montagneuses, où le relief varie beaucoup. Au contraire, dans les plaines, nous utiliserons peu d'échantillons pour limiter le volume des données. Si par contre, notre but était de réaliser une carte démographique, nous aurions plutôt intérêt à concentrer nos échantillons dans les régions où la population varie le plus, et l'échantillonnage précédent ne conviendra pas.

Le problème de la création d'un échantillonnage adapté pour représenter une fonction sur un maillage est central dans ce mémoire. Nous nous intéressons dans la Partie I au placage de textures. Comme nous l'avons dit précédemment, une texture est une image qui sera collée sur la surface sur laquelle nous souhaitons ajouter de la couleur. Le collage est réalisé en dépliant la surface pour la mettre à plat sur l'image. Du point de vue informatique, les images utilisées sont enregistrées sous la forme d'un tableau, dans lequel chaque case contient une couleur. Ces cases sont les *pixels* de l'image. Une fois le placage réalisé, tous ces pixels sont collés sur la surface et sont les échantillons sur lesquels la couleur de la surface est enregistrée. La position de ces échantillons sur la surface est contrôlée par la façon dont la surface a été dépliée. Nous commencerons donc par décrire dans le Chapitre 1 une technique de la littérature pour déplier une surface, qui nous permettra ensuite dans le Chapitre 2 d'élaborer une méthode pour faire en sorte que la fonction représentée sur la surface soit *continue*. Plus prosaïquement, il s'agit comme nous l'avons dit de faire disparaître les coutures apparaissant sur la Figure 2.

Dans la Partie II, nous adoptons une approche différente, qui consiste à répartir les échantillons sur la surface, et à enregistrer une valeur pour chacun d'entre eux. La fonction à numériser est ainsi représentée par la liste des positions et des valeurs des échantillons. À partir de cette liste, nous fournirons une valeur pour chaque point de la surface, simplement en donnant la valeur de l'échantillon le plus proche de ce point. Le Chapitre 3 propose à

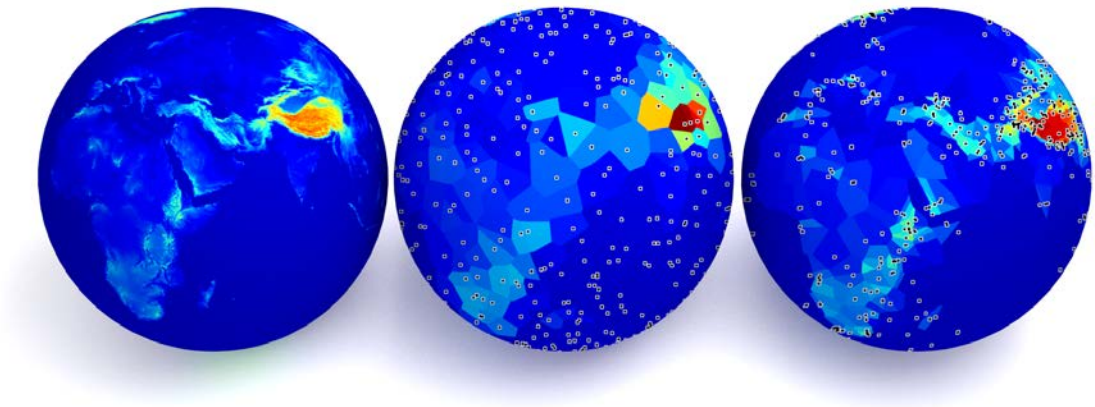


FIGURE 3 – Approximation de la fonction « altitude » sur la surface de la terre. À gauche, la fonction originale, au centre une approximation utilisant un échantillonnage de 1000 points tirés uniformément au hasard, à droite un échantillonnage de 1000 points optimisés.

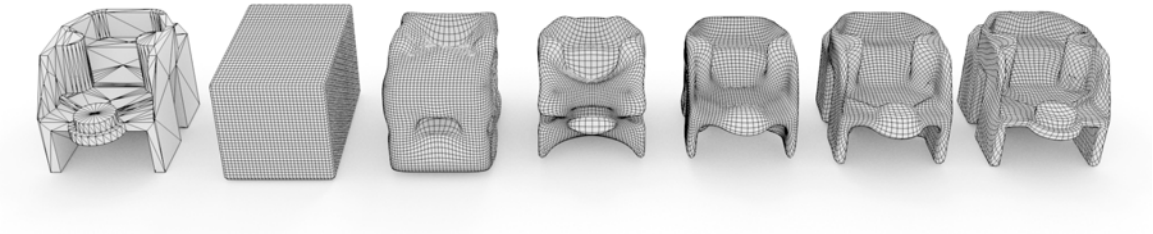


FIGURE 4 – Ajustement de deux surfaces. La surface cible est la pièce mécanique de gauche, et la surface ajustée est le pavé. De gauche à droite, le pavé est déformé pour s'ajuster au maillage cible.

cette fin un outil permettant de calculer efficacement les portions de la surface qui recevront ainsi la valeur de chacun des échantillons. Il s'agit de calculer un *diagramme de Voronoï restreint*. La Figure 3 montre alors que selon la façon dont les échantillons sont répartis sur la surface, l'approximation sera plus ou moins fidèle à la fonction qu'elle représente. Le Chapitre 4 s'intéresse ainsi à mesurer l'erreur entre la fonction initiale et l'approximation qui en est faite pour pouvoir ensuite calculer des positions pour les échantillons de sorte que cette erreur soit minimisée. C'est en utilisant cette méthode que nous avons généré l'échantillonnage optimisé de la Figure 3.

Le Chapitre 5 enfin démontre que notre mesure de la qualité d'un échantillonnage peut également être utilisée pour mesurer à quel point deux surfaces sont semblables. Il suffit en effet de mesurer à quel point un échantillonnage de l'une des surface est de bonne qualité pour l'autre surface et vice versa. Sur cette base, nous aborderons alors le problème de l'ajustement d'une surface sur une autre. Dans ce cadre, deux surfaces sont fournies, l'une fixe (la surface cible) et l'autre mobile (la surface ajustée). Nous appliquerons alors les méthodes d'optimisation d'échantillonnages évoquées dans le Chapitre 4 pour déformer la surface mobile et la rendre la plus semblable possible à la surface fixe, comme illustré sur la Figure 4.

Résumé des contributions

Dans cette thèse, les contributions que nous présentons sont :

- une méthode permettant de plaquer des textures sans coutures apparentes décrite dans le Chapitre 2, et publiée par Ray, Nivoliers, Lefebvre, et Lévy [93] ;
- un algorithme efficace pour le calcul d'un diagramme de Voronoï restreint à une surface, présenté en Chapitre 3. Ces travaux n'ont à l'heure actuelle pas été publiés ;
- une étude sur l'optimisation de fonctions objectif définies sur un diagramme de Voronoï restreint, appliquée au problème de l'optimisation d'un échantillonnage pour l'approximation d'une fonction quelconque sur un maillage. Ces résultats, présentés en Chapitre 4 ne sont pour l'instant pas publiés ;
- une méthode permettant d'ajuster une surface de subdivision à une surface cible (Chapitre 5), publiée par Nivoliers, Yan, et Lévy [74].

Publications associées à cette thèse

[74] V. Nivoliers, D.-M. Yan et B. Lévy : Fitting Polynomial Surfaces to Triangular Meshes with Voronoi Squared Distante Minimization. *Engineering with Computers*, 2012. À paraître. 17, 185

[93] N. Ray, V. Nivoliers, S. Lefebvre et B. Lévy : Invisible seams. *In Computer Graphics Forum*, vol. 29, p. 1489–1496. Wiley Online Library, 2010. 17, 49, 185

Première partie

Échantillonnage par le placage de textures

Chapitre 1

État de l'art sur les maillages et paramétrisations préservant une grille

1.1 Motivations

L'objectif de ce chapitre est tout d'abord d'introduire les maillages ainsi que les notions de géométrie et de topologie de base que nous utiliserons régulièrement au sein de ce mémoire. Nous commencerons ainsi par décrire en Section 1.3 les maillages en tant que structure de données pour enregistrer des objets tridimensionnels, ainsi que les notions de normale, de courbure et de paramétrisation.

Nous aborderons ensuite en Section 1.5 la notion de paramétrisation préservant une grille, sur laquelle reposent les travaux que nous présenterons en Chapitre 2. Le but de cette partie est de présenter l'ensemble des outils nécessaires à l'obtention d'une telle paramétrisation, à partir des différentes méthodes proposées dans la littérature.

1.2 Contributions

Ce chapitre ne contient pas de contribution propre. Nous avons toutefois choisi de décrire de manière précise la chaîne de traitements à réaliser sur un maillage pour l'obtention d'une paramétrisation préservant une grille. Il s'agit en effet d'une étape indispensable pour utiliser la méthode que nous présentons en Chapitre 2. Il nous a été nécessaire d'implémenter les méthodes de l'état de l'art, car il n'existe pas à l'heure actuelle de librairie ouverte et directement utilisable pour obtenir une telle paramétrisation, et nous avons plusieurs fois été sollicités à ce sujet. Ce chapitre tente donc d'apporter une synthèse sur ces méthodes pour les rendre plus accessibles.

1.3 Maillages

1.3.1 Numérisation des objets tridimensionnels

Images

Selon les applications et le mode d'acquisition, il existe de nombreux formats pour numériser les objets tridimensionnels. Avant d'aborder le cas de ces objets, nous ferons toutefois

un détour pour aborder le cas des objets bidimensionnels que nous avons plus l'habitude de manipuler. Il existe plusieurs modèles pour définir la mémoire d'un ordinateur et ce qu'il est possible d'y enregistrer. D'une manière générale, chaque fichier peut être considéré comme une liste de nombres. Le format de fichier définit ensuite de quelle façon ces nombres doivent être interprétés. Pour enregistrer un fichier texte par exemple, le format « txt » consiste simplement à associer à chaque caractère un nombre, et à enregistrer la liste des nombres correspondant aux caractères dans l'ordre où ils apparaissent dans le texte. La norme utilisée pour faire correspondre les caractères aux nombres peut elle-même varier. Pour les objets bidimensionnels, il donc s'agit d'enregistrer une image.

Une méthode très courante pour acquérir une image d'un objet consiste à en prendre une photo. L'objet est alors enregistré sous forme d'une image dite *matricielle*. De manière informatique, ces images sont enregistrées sous forme d'un ensemble de pixels, organisés selon une grille. Chaque pixel correspond à une valeur de couleur enregistrée. Chaque couleur est donc associée à un nombre et il suffit de connaître les dimensions de l'image pour pouvoir afficher la liste de ces nombres sur un écran ou pour l'imprimer. Ces données sont très simples à interpréter et à acquérir. Le revers de la médaille est que les pixels deviennent visibles en regardant l'image de près. Pour lutter contre ce phénomène, il est nécessaire d'augmenter le nombre de pixels utilisés, mais la quantité de pixels et donc de nombre à enregistrer augmente très vite. L'image prend donc beaucoup de place en mémoire. Pour limiter l'espace mémoire, des formats compressés avec ou sans pertes existent, comme les très classiques « jpg » ou « png ». Il est également possible d'utiliser différents modes de *filtrage* pour lire l'image. Nous reviendrons plus en détail sur ces techniques en Section 2.3.3.

L'autre grande famille de formats pour enregistrer des images est celle des formats dits *vectoriels*. Il s'agit ici de représenter des formes sous forme d'équations. Un cercle par exemple peut tout à fait être représenté par la position de son centre et par son rayon. Pour enregistrer un rectangle, une solution consiste à enregistrer la liste des positions de ses sommets et l'ordre dans lequel les relier. C'est alors à l'ordinateur de lire ces données et de réaliser le dessin. Parmi les formats de ce genre, les plus connus sont probablement « pdf », « ps » ou « svg ». Les formats vectoriels étaient en particulier utilisés pour afficher des images sur les premiers écrans d'ordinateur, qui fonctionnaient plutôt sur le principe d'un oscilloscope, et donc d'une sorte de crayon dont le trajet est guidé par une équation (il s'agit en réalité d'un faisceau d'électrons dont la trajectoire est directement contrôlée par un champ magnétique). Le format utilisé par un ordinateur pour communiquer un document à une imprimante est vectoriel, et les contraintes de taille de données ont également rendu ce format très populaire sur la toile pour les applications interactives ou la mise en page des sites.

Objets tridimensionnels

Les objets tridimensionnels s'enregistrent en utilisant des concepts similaires que ceux utilisés pour les images. Le pixel des images matricielles devient le *voxel* pour représenter des données volumiques. Ce format est en particulier utilisé pour enregistrer les données issues d'un appareil d'Imagerie par Résonance Magnétique (IRM). Un tel appareil permet en effet d'acquérir une représentation d'un patient qui ne se limite pas à ce qui est visible de l'extérieur mais permet au contraire de pouvoir l'étudier tranche par tranche. Ces données sont donc enregistrées sous la forme d'une pile d'images, chacune représentant une tranche. Un pixel (« picture element ») devient alors un voxel (« volume element »). Comme leur pendant bidimensionnel, ce type de représentation est très volumineux à stocker.

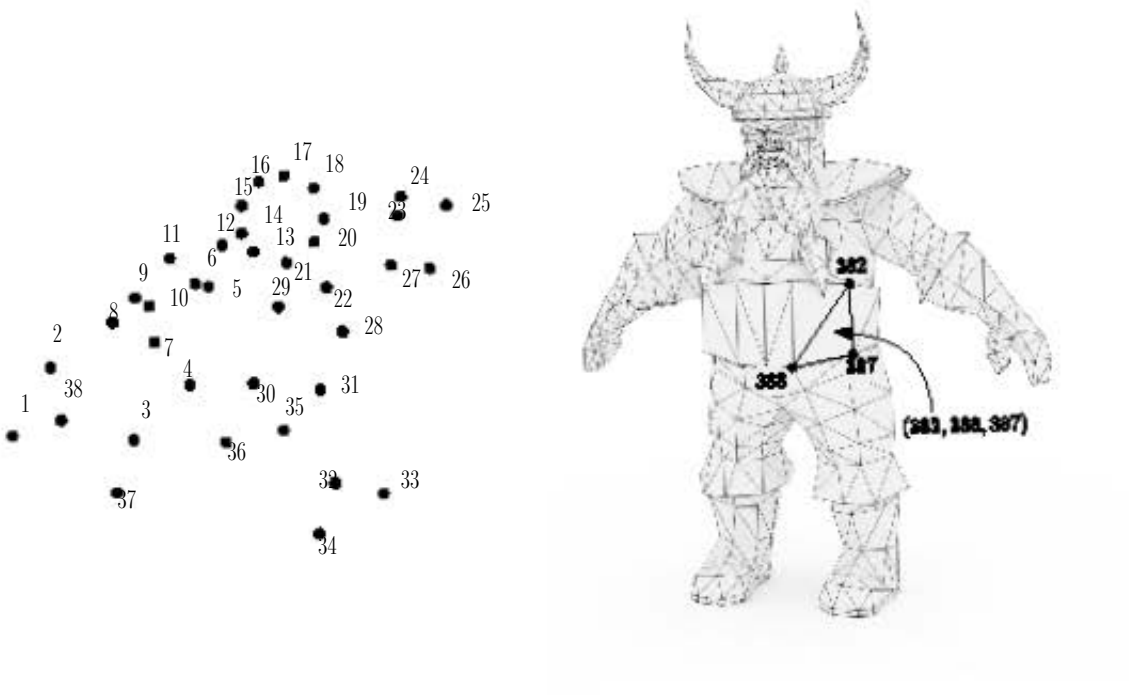


FIGURE 1.1 – Jeu « connectez les points ». Les maillages représentent les surfaces de manière similaire en les décrivant comme un ensemble de points dans l'espace et un ensemble de triangles pour connecter ces points entre eux. Ici la façon de connecter les points est fournie par l'ordre dans lequel ils sont numérotés (leurs indices). Pour les maillages (à droite), chaque triangle est décrit par le triplet des indices de ses trois sommets. Il s'agit ici d'un maillage utilisé pour décrire un personnage de jeu vidéo.

Comme pour les images, il est également possible de décrire les objets par des équations. Ce type de format est en particulier utilisé dans le domaine de l'ingénierie pour concevoir des pièces mécaniques, avec en particulier les formats « step » ou « iges ». Des méthodes ont été mises au point pour permettre de représenter des surfaces complexes et surtout de pouvoir les manipuler simplement. Il s'agit ici de décrire des objets pour lesquels l'intérieur est en général uniformément constitué d'un même matériau. Ainsi seule la surface de ces objets doit être décrite. Le problème de ce type de représentation est que l'interprétation des données peut devenir compliquée à réaliser. Par exemple de nombreuses recherches ont porté sur le l'affichage des courbes et surfaces décrites par ce type de format. Il est en effet essentiel de pouvoir rapidement les dessiner dans le contexte d'un logiciel permettant de les éditer, car l'utilisateur pourra alors visionner immédiatement les modifications qu'il apporte à l'objet. L'écran sur lequel l'objet doit être affiché est composé de pixels. Le problème de l'affichage se résume donc à déterminer pour chaque pixel de l'écran la couleur qu'il doit afficher. Il s'agit du problème du *rendu*. Les équations décrivant les objets doivent rester suffisamment simples pour que ce calcul puisse être réalisé rapidement. Une méthode classique consiste à convertir l'objet en un *maillage*.

1.3.2 Définition d'un maillage

Sommets, triangles, arêtes

Les maillages entrent dans la famille des formats vectoriels pour enregistrer un objet tridimensionnel. Ils décrivent les objets d'une manière similaire à celle des jeux « connectez

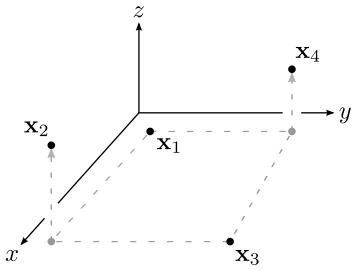


FIGURE 1.2 – Quatre points de $\mathbb{R}^3$ qu'il n'est pas possible de relier par un quadrilatère plan. Ils sont alors dits non coplanaires. Ici, cette configuration est construite en partant d'un rectangle horizontal et en déplaçant deux sommets opposés verticalement. En général, quatre points quelconques de $\mathbb{R}^3$ ne sont pas coplanaires.

les points » que l'on peut trouver dans les magazines pour enfants (Figure 1.1). Ces jeux permettent de dessiner une forme en reliant des points numérotés. L'ordre des numéros permet de déterminer quel point doit être relié à quel autre. De la même façon, un maillage permet de représenter une surface en reliant entre eux des points dans l'espace par des triangles. Chaque triangles est donné par le triplet des numéros des trois sommets qu'il relie.

Plus formellement, un maillage est défini par :

ses sommets que nous noterons $\mathbf{X} = \{\mathbf{x}_i\}_i$, où $\mathbf{x}_i$ est le sommet d'indice i. D'une manière générale, $\mathbf{x}_i$ représentera un sommet quelconque d'un maillage. Lorsque nous utiliserons plusieurs sommets quelconques, nous nous permettrons de noter abusivement ces sommets $\mathbf{x}_1$,$\mathbf{x}_2$,... même s'il ne s'agit pas réellement des sommets d'indice 1,2,... de $\mathbf{X}$. Les coordonnées d'un sommet $\mathbf{x}_i$ seront notées $(x_{i,1},x_{i,2},x_{i,3}) \in \mathbb{R}^3$;

ses triangles , notés $\mathbf{T} = \{T_p\}_p$, où $T_p = (\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3)$ est le triangle d'indice p qui connecte les trois sommets $\mathbf{x}_1$, $\mathbf{x}_2$ et $\mathbf{x}_3$;

ses arêtes notées $\mathbf{E}$, qui se déduisent de l'ensemble des triangles. Une arête est une paire de sommets correspondant à un côté d'un triangle de $\mathbf{T}$. Il s'agit des lignes noires sur le maillage de la Figure 1.1. Par exemple, si $\mathbf{T}$ contient un triangle $T_p = (\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3)$, $\mathbf{E}$ contient les trois arêtes $(\mathbf{x}_1,\mathbf{x}_2)$, $(\mathbf{x}_2,\mathbf{x}_3)$ et $(\mathbf{x}_3,\mathbf{x}_1)$.

Il y a plusieurs raisons pour lesquelles les triangles sont préférés aux polygones quelconques. D'une part, étant donné trois points, il existe toujours un triangle plan (non courbé) passant par ces trois points. Par contre, étant donné quatre points quelconques, il n'existera en général pas de quadrilatère plan reliant ces quatre points (Figure 1.2). Quatre points ne pouvant pas être reliés par un quadrilatère plan sont dits non *coplanaires*. La surface correspondant à un quadrilatère reliant quatre points non coplanaires n'est donc pas proprement définie. D'autre part, un triangle est toujours convexe, c'est à dire qu'étant donné deux points dans un triangle, le segment qui relie ces deux points est entièrement dans le triangle. Cette propriété simplifie de nombreux algorithmes, en particulier l'Algorithme 3.1 que nous verrons en Chapitre 3. Les maillages à faces quadrilatérales, dits maillages quadrangulaires, restent toutefois couramment utilisés dans le domaine de l'ingénierie mécanique et pour modéliser des objets pour les jeux vidéo ou les films d'animation. Au moment de l'affichage, un tel maillage est en général converti automatiquement en maillage triangulaire, en découpant chaque quadrilatère le long d'une de ses diagonales. L'obtention d'un maillage triangulaire à partir d'un maillage quadrangulaire est donc facile. Comme nous le verrons en Section 1.5, la conversion d'un maillage triangulaire en maillage quadrangulaire est par contre un problème difficile.

Pour revenir au problème de l'affichage d'une surface vectorielle qui nous avait permis d'introduire les maillages, les cartes graphiques désormais présentes dans la plupart des

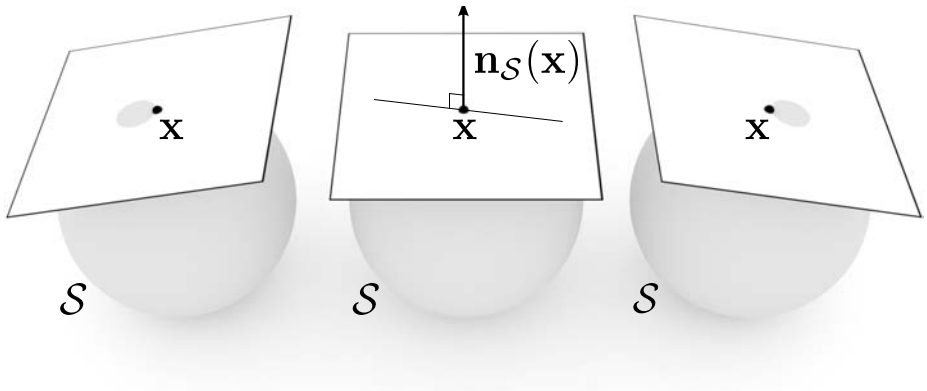


FIGURE 1.3 – Plan tangent au point $\mathbf{x}$ d'une surface $\mathcal{S}$. Le plan tangent est le seul plan que nous pouvons poser sur $\mathcal{S}$ au point $\mathbf{x}$ sans qu'il ne traverse $\mathcal{S}$ (au centre). En pivotant légèrement le plan tangent (à gauche et à droite), $\mathcal{S}$ traverse alors le plan. La normale $\mathbf{n}_{\mathcal{S}}(\mathbf{x})$ de $\mathcal{S}$ au point $\mathbf{x}$ est le vecteur orthogonal au plan tangent, c'est à dire que toute droite contenue dans le plan tangent est perpendiculaire à ce vecteur.

ordinateurs ont justement pour rôle entre autres de dessiner efficacement un ensemble de triangles étant donné un point de vue. Elles permettent en particulier de déterminer efficacement étant donné les paramètre du point de vue choisi quel triangle afficher à quel endroit de l'écran, et quelle portion de quel triangle devra être dessinée devant quelle autre portion de triangle.

1.3.3 Outils de géométrie différentielle

La géométrie différentielle est une branche des mathématiques qui s'intéresse en particulier à l'étude des surfaces du point de vue de l'analyse, c'est à dire à l'étude de surfaces définies par des fonctions. Le lecteur pourra se référer au livre de Do Carmo [24] pour une étude approfondie du sujet. Nous nous contenterons ici d'aborder rapidement et de manière intuitive les deux notions que sont la normale et la courbure d'une surface, et la façon dont ils peuvent être calculés sur un maillage.

Normale

Nous aurons souvent l'occasion dans le présent document d'utiliser la *normale* d'un maillage en un point donné. La normale est intimement liée à la notion de plan tangent. Étant donné un point quelconque $\mathbf{x}$ d'une surface $\mathcal{S}$, déterminer le plan tangent à $\mathcal{S}$ au point $\mathbf{x}$ consiste à trouver le plan que nous pourrions poser sur $\mathcal{S}$ au point $\mathbf{x}$, sans qu'il ne traverse $\mathcal{S}$. Comme nous le montre la Figure 1.3, si la surface $\mathcal{S}$ est lisse au point $\mathbf{x}$, en pivotant ne serait-ce qu'un petit peu le plan tangent posé au point $\mathbf{x}$, la surface $\mathcal{S}$ traverse alors le plan. La normale de $\mathcal{S}$ au point $\mathbf{x}$ est le vecteur orthogonal au plan tangent, c'est à dire que toute droite contenue dans le plan tangent est perpendiculaire à la normale. Nous noterons $\mathbf{n}_{\mathcal{S}}(\mathbf{x})$ la normale de la surface $\mathcal{S}$ au point $\mathbf{x}$. De manière imagée, en considérant un être habitant sur notre surface si petit qu'il aurait l'impression que le sol est plat, il aurait alors l'impression de vivre sur le plan tangent de notre surface.

Les maillages sont des surfaces un peu particulières, car elles sont constituées de triangles plans accolés les uns aux autres. Pour chaque triangle, le plan tangent est alors simplement le plan contenant le triangle. En revanche, au niveau d'une arête ou d'un sommet, le plan tangent n'est pas défini. En effet, un plan posé sur un sommet peut alors pivoter librement

sans traverser le maillage jusqu'à ce qu'il se pose sur une arête ou un triangle, et un plan posé sur une arête peut pivoter de la même manière jusqu'à se poser sur un triangle adjacent. Étant donné un triangle $T = (\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3)$ de notre maillage, la normale de ce triangle se calcule en utilisant les coordonnées des sommets et le *produit vectoriel* $\times$:

$$\mathbf{n}_T = (\mathbf{x}_1 - \mathbf{x}_2) \times (\mathbf{x}_1 - \mathbf{x}_3), \quad \text{où} \quad \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \times \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} y_1 z_2 - y_2 z_1 \\ z_1 x_2 - z_2 x_1 \\ x_1 y_2 - x_2 y_1 \end{pmatrix}. \tag{1.3.1}$$

L'ordre dans lequel les sommets d'un triangle sont donnés a une importance. En effet, la triangle $T' = (\mathbf{x}_1,\mathbf{x}_3,\mathbf{x}_2)$ aura la normale :

$$\mathbf{n}_{T'} = (\mathbf{x}_1 - \mathbf{x}_3) \times (\mathbf{x}_1 - \mathbf{x}_2) = -\mathbf{n}_T.$$

Un moyen mnémotechnique pour déterminer la direction de la normale est la règle dite « de la main droite ». Imaginons que nous posions notre main droite sur le triangle T de telle sorte que $\mathbf{x}_1$ soit au niveau de notre poignet, $\mathbf{x}_2$ au niveau de la base de nos doigts, et $\mathbf{x}_3$ au sommet de nos doigts, alors la direction de notre pouce tendu nous fournit la direction de la normale. Une convention généralement respectée consiste à fixer l'ordre des sommets d'un triangle de façon à orienter la normale de ce triangle vers l'extérieur de la surface (lorsqu'il est défini).

Le vecteur $\mathbf{n}_T$ tel qu'il est défini par l'Équation 1.3.1 n'est pas un vecteur de norme 1. De manière remarquable, la longueur de ce vecteur (sa norme euclidienne), notée $\|\mathbf{n}_T\|$ correspond en réalité à deux fois l'aire du triangle T, que nous noterons $|T|$. Nous utiliserons cette relation dans le Chapitre 4 pour étudier les variations de l'aire d'un triangle en fonction de ses sommets.

Courbure

Reprenons la métaphore de l'être minuscule vivant sur une surface, si petit qu'il la voit plane. Cet être désormais se doute qu'en réalité, la surface n'est pas plane, et suppose alors qu'il s'agit d'une boule. Il pourrait alors se demander quel serait le rayon de cette boule. C'est ce que fit Ératosthène pour notre planète aux alentours de l'an -200. Pour réaliser cette mesure, il fonda ses calculs sur les informations qu'il pouvait trouver dans la région du globe où il vivait. Mesurer la courbure d'une surface correspond à cette idée. Il s'agit à partir des informations dont nous disposons autour d'un point $\mathbf{x}$ de déterminer quelle serait le rayon de la sphère le plus proche de nos observations. Le problème est en réalité plus complexe, car en général, une surface ne sera pas courbée de la même façon dans toutes les directions autour d'un point $\mathbf{x}$, comme le montre la Figure 1.4. Ainsi, calculer la courbure correspond en réalité à calculer la courbure minimale et la courbure maximale de $\mathcal{S}$ autour du point $\mathbf{x}$. La définition que nous donnerons ici de la courbure sera volontairement très simplifiée, car nous n'aurons pas besoin de comprendre cette notion en détail pour la suite. Nous invitons le lecteur à se référer à l'ouvrage de Do Carmo [24] pour une définition rigoureuse de la courbure.

Pour donner une définition grossière de la courbure d'une surface, nous différencierons deux cas :

la surface est en forme de bol éventuellement retourné. Dans ce cas, calculer la courbure revient à déterminer :

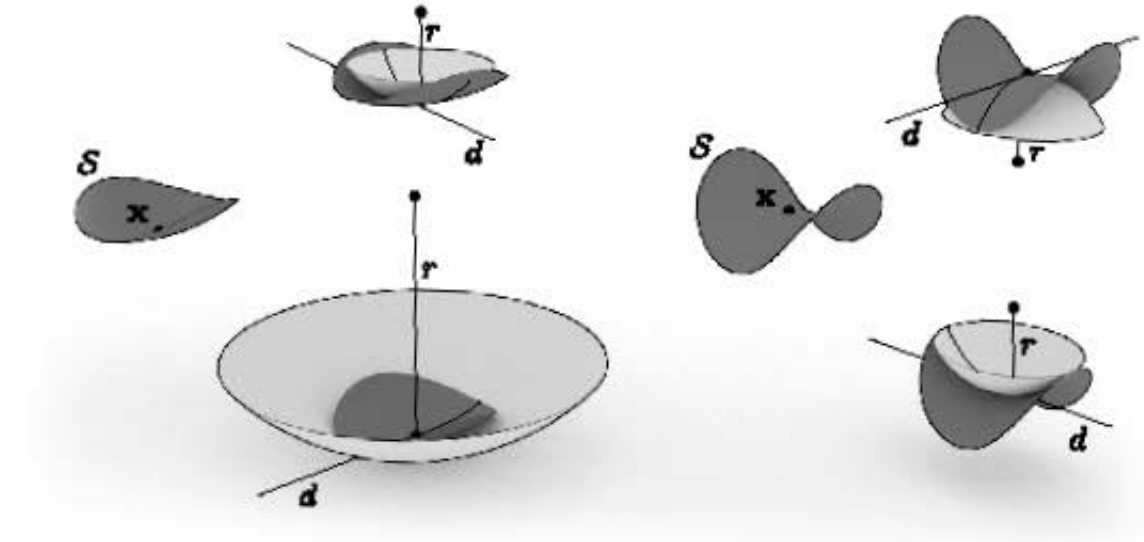


FIGURE 1.4 – Rayons et directions principales de courbure d'une surface. Deux cas de figure sont envisageables : soit la surface est en forme de bol (à gauche), soit elle est en forme de selle (à droite). Dans chaque cas, deux rayons de courbure sont définis, associés à deux directions principales de courbures, ici représentées par les droites d. Dans chaque cas, les deux directions de courbure sont orthogonales.

  - la plus grosse sphère qu'il est possible de faire entrer dans le bol de façon à ce qu'elle en touche le fond (le point $\mathbf{x}$ où la courbure est calculée) ;
  - la plus petite sphère dans laquelle il est possible de faire rentrer le bol, de telle sorte que le fond du bol touche la sphère.

  Ces deux sphères nous fournissent deux rayons de courbure (Figure 1.4 à gauche) ;

la surface est en forme de selle de cheval et n'est donc pas courbée dans le même sens selon la direction considérée. Calculer la courbure revient dans ce cas à chercher de chaque côté de la surface la plus grosse sphère que nous pourrions poser sur le point $\mathbf{x}$ sans la faire décoller. Comme dans le cas précédent, nous obtenons deux rayons de courbure, un pour chaque côté de la surface (Figure 1.4 à droite).

La valeur de la courbure est définie comme l'inverse du rayon de courbure. Dans le cas particulier où la surface est un plan, les sphères que nous venons de décrire ne sont pas définies, car toute sphère, aussi grosse soit-elle peut être en contact avec le point $\mathbf{x}$ sans traverser la surface. Le rayon de courbure est alors infini, et la courbure associée est nulle, ce qui correspond bien à l'intuition qu'un plan n'est pas une surface courbe.

La droite qui passe par le point $\mathbf{x}$ et le centre de la sphère correspondante suit la direction de la normale. Par convention, lorsque le centre de la sphère est dans le sens de la normale par rapport au point $\mathbf{x}$, le signe de la courbure est négatif. À l'inverse, lorsque le centre de la sphère est dans le sens opposé à la normale, la courbure est positive. Ainsi, lorsque la surface est en forme de bol, les deux rayons ont le même signe alors que lorsque la surface est en forme de selle de cheval, les deux rayons sont de signes opposés. Parmi les deux valeurs de courbure que nous obtenons en un point $\mathbf{x}$ de la surface, la valeur maximale de la courbure sera notée κ_1, et la valeur minimale κ_2.

Nous avons défini la courbure en mettant en cherchant à mettre en contact des sphères maximales ou minimales avec la surface. En règle générale, ces sphères ne sont contraintes

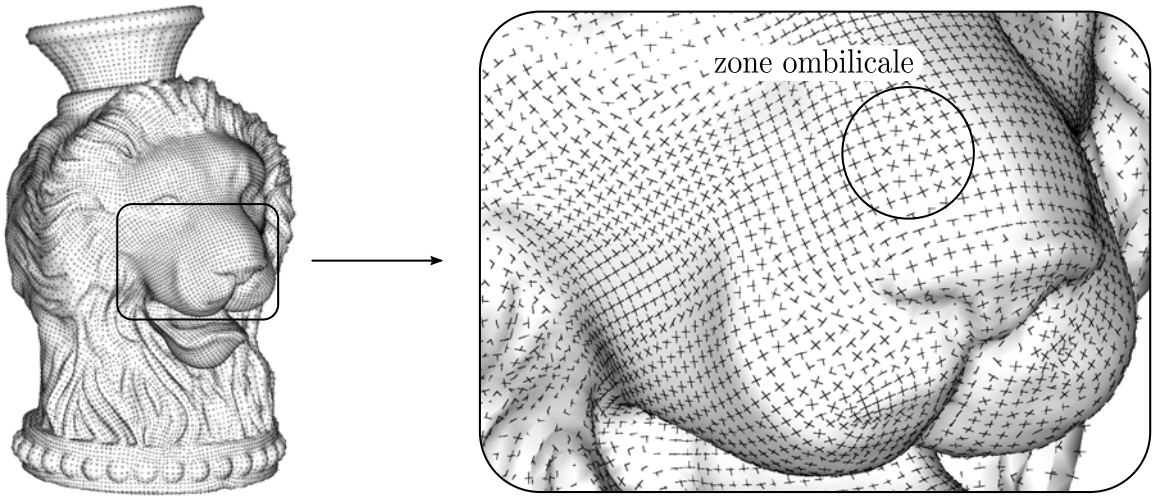


FIGURE 1.5 – Estimation des directions principales de courbure sur un maillage. L'algorithme utilisé ici est celui de Cohen-Steiner et Morvan [19]. Nous voyons apparaître sur le museau des zones ombilicales où la surface est localement sphérique. Dans ce cas, les courbures minimale et maximale sont égales, et n'importe quel direction est une direction de courbure principale. Dans ces zones, les directions de courbures obtenues sont donc incohérentes.

que par une seule direction autour de $\mathbf{x}$, et pourraient continuer à grossir (respectivement rétrécir) dans les autres directions. La direction qui contraint la sphère est appelée la *direction de courbure principale* associée à la courbure définie par la sphère. De manière remarquable, que la surface ait une forme de bol ou de selle, les deux directions principales de courbures associées aux deux sphères calculées sont orthogonales. Il existe cependant un cas où les directions principales de courbure ne sont pas définie : lorsque la surface est exactement une sphère. Dans ce cas, les courbures minimale et maximale sont égale, et les sphères mises en contact avec la surface sont contraintes par toutes les directions à la fois. Ce cas particulier s'étend au cas du plan, qui peut être considéré comme une sphère de rayon infini. Ces zones de la surface où les directions principales de courbures sont dites *ombilicales*.

Un maillage est constitué de faces planes, accolées les unes aux autres. Les faces étant planes, la courbure à l'intérieur des faces est nulle. Au niveau des arête, la surface a des angles, et n'est plus lisse. En ces points, la courbure est infinie. L'application directe de la définition de la courbure sur les maillage n'a donc que peu d'intérêt. Le calcul de la courbure sur un maillage revient en général à supposer que le maillage est une approximation d'une surface lisse. C'est alors la courbure de cette surface lisse qui est estimée à partir de la géométrie du maillage. L'article de synthèse de Petitjean [80] détaille un certain nombre de méthodes pour calculer des courbures sur un maillage. D'autres méthodes ont également été proposées dans des publications plus récentes, comme celles de Cohen-Steiner et Morvan [19] et de Mérigot *et al.* [71]. La méthode de Cohen-Steiner et Morvan [19] est celle que nous avons utilisée ici. Elle estime la courbure en découpant un petit voisinage du maillage autour du point considéré, et en estimant la courbure à partir des angles formés au niveau des arêtes par les faces voisines et de la somme des angles des faces au niveau des sommets de la portion de maillage découpée. La Figure 1.5 fournit un exemple du résultat de cette méthode sur un maillage. La méthode de Mérigot *et al.* [71] aborde le cas des nuages de

FIGURE 1.6 – Transformation d'une tasse en beignet. Le fond de la tasse est tout d'abord relevé, puis l'anse est gonflée alors que le corps est réduit. Le résultat de la déformation est un beignet, ou plus formellement un tore. Cette déformation n'a nécessité ni coupure ni recollement, la surface d'une tasse est donc homéomorphe à un tore.

points, c'est à dire des surfaces qui ne seraient décrite que par un ensemble de points, sans les relier par des triangles. Leur méthode permet d'estimer les directions principales de courbure, et en particulier les arêtes vives des surfaces.

1.3.4 Éléments de topologie

Nous aurons plusieurs fois l'occasion de dire de deux surfaces qu'elles sont *homéomorphes*. Comme précédemment, nous invitons le lecteur à se référer à Do Carmo [24] pour une explication plus formelle de cette notion. En termes communs, deux surfaces $\mathcal{S}$ et $\mathcal{T}$ sont homéomorphes s'il est possible de déformer $\mathcal{S}$ pour obtenir $\mathcal{T}$ sans sans avoir à découper ou recoller la surface. L'exemple classique pour illustrer cette notion est celui de la tasse et du beignet illustré sur la Figure 1.6. Plus formellement, un homéomorphisme entre deux surfaces $\mathcal{S}$ et $\mathcal{T}$ est une fonction $f : \mathcal{S} \to \mathcal{T}$:

- bijective ;
- continue ;
- de réciproque continue.

Tout d'abord, une fonction $f : \mathcal{S} \to \mathcal{T}$ signifie qu'à tout point $\mathbf{x}$ de $\mathcal{S}$, nous allons associer un point de $\mathcal{T}$ que nous appellerons $f(\mathbf{x})$. La fonction f sera bijective si et seulement si pour tout point $\mathbf{y}$ de $\mathcal{T}$, il existe un unique point $\mathbf{x}$ de $\mathcal{S}$ tel que $f(\mathbf{x}) = \mathbf{y}$. Cette propriété nous permet de définir la *réciproque* de f comme la fonction f^{-1} telle que si $f(\mathbf{x}) = \mathbf{y}$ alors $f^{-1}(\mathbf{y}) = \mathbf{x}$. Cette partie de la définition correspond au fait que nous déformons $\mathcal{S}$ pour obtenir $\mathcal{T}$, et que nous obtenons bien $\mathcal{T}$ tout entière.

Une fonction f est continue lorsqu'elle préserve les relations de voisinage. Autrement dit, si deux points $\mathbf{x}$ et $\mathbf{x}'$ sont proches sur $\mathcal{S}$, les points $f(\mathbf{x})$ et $f(\mathbf{x}')$ seront proches sur $\mathcal{T}$. Le fait que f soit continue formalise le fait qu'en déformant $\mathcal{S}$, nous n'avons pas le droit de la découper. En effet, de part et d'autre d'une coupure, des points de $\mathcal{S}$ étaient voisins avant la découpe, et ne le sont plus. Le fait que la réciproque de f soit continue signifie au contraire que nous pouvons pas faire en sorte que deux points qui n'étaient pas voisins le deviennent. Il s'agit donc de ne pas recoller ensembles plusieurs parties de $\mathcal{S}$ lors de la déformation.

Il existe un critère simple à calculer pour déterminer si deux surfaces fermées décrites par des maillages sont homéomorphes. Un maillage représente une surface fermée si chaque arête a deux triangles adjacents. Le critère consiste à déterminer leur caractéristique d'Euler Poincaré. Une expérience pour la mettre en évidence consiste à se munir de ballons de

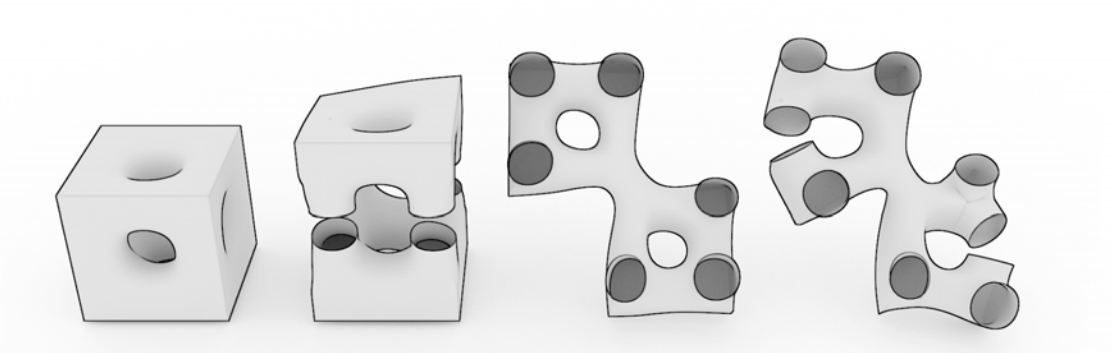


FIGURE 1.7 – Genre d'un cube percé de part en part entre toutes les paires de faces opposées (à gauche). Ici, nous pouvons tout d'abord commencer par couper trois tubes verticaux en laissant la surface connectée par le quatrième (centre gauche). En dépliant la surface autour du tube vertical préservé (centre droit), il apparaît encore deux sortes de tores qu'il est possible de couper. En découpant chacun de ces tores (à droite), il n'est alors plus possible de couper de tube sans déconnecter la surface. Nous avons pu couper cinq tubes, le genre de cette surface vaut donc cinq.

baudruche et d'un feutre. Sur le ballon gonflé, à l'aide du feutre, nous commençons par dessiner un ensemble de points. Ensuite, nous traçons des lignes entre les points sur le ballon, de manière à ce qu'il soit possible de relier toute paire de points en suivant les lignes tracées. Cet ensemble de lignes délimite un certain nombre de faces sur le ballon. Si nous notons n_f le nombre de faces ainsi créées, n_l le nombre de lignes tracées et n_p le nombre de points dessinés, nous aurons alors invariablement

$$n_f - n_l + n_p = 2,$$

quelle que soit la façon dont les points ont été placés, et quel que soit l'ensemble de lignes dessiné pour les relier. Pour un maillage, n_f correspond au nombre de faces $|\mathbf{T}|$, n_l au nombre d'arêtes $|\mathbf{E}|$ et n_p au nombre de sommets $|\mathbf{X}|$. Le nombre χ défini comme

$$\chi = |\mathbf{T}| - |\mathbf{E}| + |\mathbf{X}| \tag{1.3.2}$$

est la *caractéristique d'Euler Poincaré* d'un maillage. Si ce nombre vaut 2, le maillage est alors homéomorphe à une boule. En réalité, de façon plus générale, deux maillages sont homéomorphes si et seulement si leurs caractéristiques d'Euler Poincaré sont égales. Il suffit donc de connaître le nombre de faces, de sommets et d'arêtes de deux maillages pour savoir s'ils sont homéomorphes. Le genre g d'une surface est le nombre $1 - \frac{\chi}{2}$. Sa valeur correspond au nombre de « trous » ou d'*anses* d'une surface. Un beignet a par exemple une anse, alors qu'un bretzel traditionnel en a trois. Plus formellement, le nombre d'anses est le nombre maximal de tubes qu'il est possible de couper sans que la surface ne soit déconnectée en plusieurs morceaux. Couper un tube signifie que la courbe correspondant à la découpe sur la surface est une boucle. La Figure 1.7 fournit un exemple complexe de dénombrement des anses d'une surface.

La caractéristique d'Euler Poincaré permet de déterminer si deux surfaces $\mathcal{S}$ et $\mathcal{T}$ fermées sont homéomorphes. Dans le cas de surface comportant des bords, il est nécessaire de raffiner ce critère pour les prendre en compte. Cette caractéristique ne fournit par contre pas l'homéomorphisme entre $\mathcal{S}$ et $\mathcal{T}$, c'est à dire le point de $\mathcal{T}$ associé à chaque point de $\mathcal{S}$ (et inversement). Chercher un homéomorphisme entre deux surfaces est plutôt un problème de *paramétrisation*.

1.4 Paramétrisation de maillages

Nous aborderons ici le problème de la paramétrisation de maillages. Le but d'une paramétrisation consiste à se doter d'un repère et d'un ensemble de paramètres pour localiser des objets sur une surface. La latitude et la longitude permettant de se repérer sur Terre sont un exemple de paramètres. Plus généralement, un exemple typique de paramétrisation consiste à réaliser une carte plane de la Terre. Il s'agit alors de représenter le globe terrestre tout entier sur la surface plane d'une feuille de papier. De très nombreuses méthodes ont été proposées pour résoudre ce problème, comme par exemple la projection de Mercator, ou celle de Peters. La différence entre ces deux projections est que la projection de Mercator vise à préserver la forme des pays et des continents, parfois au détriment de leur taille. Ainsi sur une carte obtenue par cette projection, le Groenland apparaît avec une taille similaire à celle de l'Afrique, alors qu'il est en réalité de l'ordre de quinze fois plus petit. Sur une carte obtenue par la projection de Peters, l'aire des continents est au contraire préservée, mais pas leur forme. La carte à choisir dépendra de l'utilisation qui en sera faite, selon ce qu'il est le plus important de pouvoir comparer.

1.4.1 Paramétrisation

Cas simple

Paramétrer un maillage consiste à le déplier pour l'étaler à plat sur un plan. Chaque point de la surface doit donc obtenir une position sur le plan. Le plan est appelé le domaine paramétrique. Nous noterons $\mathbf{u} = (u,v)$ un point du domaine paramétrique, avec (u,v) ses coordonnées. La méthode la plus simple pour paramétrer un maillage consiste à associer à chaque sommet $\mathbf{x}_i$ de la surface un point $\mathbf{u}_i$ du domaine paramétrique. Une fois les sommets de la surface placés dans le domaine paramétrique, un triangle $T_p = (\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3)$ du maillage est associé au triangle $C_p = (\mathbf{u}_1,\mathbf{u}_2,\mathbf{u}_3)$ du domaine paramétrique. Étant donné un point $\mathbf{x}$ quelconque sur T_p, il est possible de trouver trois nombres α, β et γ tels que $\alpha + \beta + \gamma = 1$ et

$$\mathbf{x} = \alpha\mathbf{x}_1 + \beta\mathbf{x}_2 + \gamma\mathbf{x}_3.$$

Les nombres α, β et γ sont les *coordonnées barycentriques* de $\mathbf{x}$ par rapport aux trois sommets du triangle T_p qui le contient. En utilisant ces coordonnées barycentriques, nous obtenons le point $\mathbf{u}$ du domaine paramétrique associé à $\mathbf{x}$:

$$\mathbf{u} = \alpha\mathbf{u}_1 + \beta\mathbf{u}_2 + \gamma\mathbf{u}_3.$$

Nous obtenons ainsi des coordonnées paramétriques pour chaque point du maillage, et avons donc une paramétrisation de ce maillage. La paramétrisation f ainsi créée qui à un point de $\mathcal{S}$ associe sa coordonnée paramétrique est dite *linéaire par morceaux*, car pour chaque triangle T_p, nous avons la propriété

$$f(\alpha\mathbf{x} + \beta\mathbf{x}' + \gamma\mathbf{x}'') = \alpha f(\mathbf{x}) + \beta f(\mathbf{x}') + \gamma f(\mathbf{x}''),$$

pour tous triplet $(\mathbf{x},\mathbf{x}',\mathbf{x}'')$ de points de T_p. Cette propriété signifie que f est linéaire. Cependant, elle ne s'applique que quand les trois points sont dans un même triangle, d'où le fait qu'elle n'est linéaire que « par morceaux ».

Il est en général souhaitable d'une paramétrisation qu'elle soit injective, c'est à dire qu'il n'existe pas deux point du maillage ayant les même coordonnées paramétriques. Visuellement, cette propriété se traduit par le fait que deux triangles paramétriques ne doivent

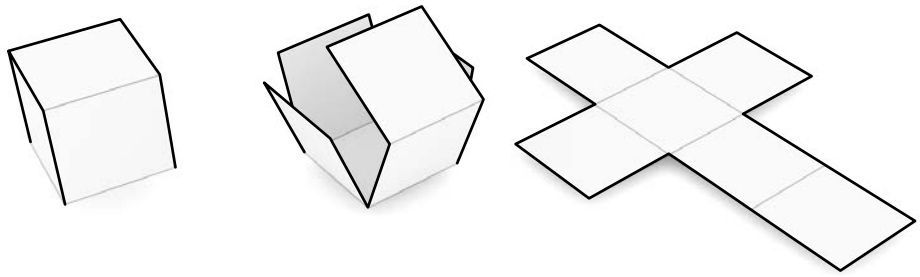


FIGURE 1.8 – Dépliage d'un cube pour en obtenir un patron. Les arêtes tracées en gras sont les *transitions* le long desquelles le cube est découpé pour pouvoir être déplié.

pas se superposer. De plus, par construction, deux triangles voisins sur le maillages seront voisins dans le domaine paramétrique. La paramétrisation est donc continue. Ainsi, elle est en réalité un homéomorphisme entre la surface et la portion de plan occupée par les triangles paramétriques. Pour qu'elle puisse exister il est donc nécessaire que la surface soit homéomorphe à une portion de plan, ce qui n'est souvent pas le cas. Une boule par exemple ne pourra pas être paramétrée de cette manière. Dans la plupart des application, il est même requis que la surface à paramétrer soit homéomorphe à un disque.

Cas général

Pour pouvoir paramétrer de manière injective un maillage quelconque, il est nécessaire de pouvoir le découper pour le rendre homéomorphe à une portion de plan. Il s'agit alors en quelque sorte de fabriquer un « patron » de notre maillage. Un patron bien connu est celui du cube présenté en Figure 1.8. Pour pouvoir déplier la surface et réaliser un tel patron, il est nécessaire de découper la surface le long d'un ensemble d'arêtes bien choisi. Ces arêtes sont les *transitions*. Lorsqu'un sommet a au moins deux transitions incidentes, ce sommet sera présent plusieurs fois dans le domaine paramétrique. Il devient donc nécessaire de pouvoir assigner plusieurs coordonnées paramétriques à un sommet.

La méthode classique consiste alors à différencier les coordonnées paramétriques des sommets pour chaque face adjacente. Ainsi pour la face $T_p = (\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3)$, nous noterons $\mathbf{u}_1^p$, $\mathbf{u}_2^p$ et $\mathbf{u}_3^p$ les coordonnées paramétriques des trois sommets pour cette face. Si deux faces sont adjacentes le long d'une arête $(\mathbf{x}_1,\mathbf{x}_2)$, si cette arête n'est pas une transition, nous aurons alors $\mathbf{u}_1^p = \mathbf{u}_1^q$ et $\mathbf{u}_2^p = \mathbf{u}_2^q$. Lorsque l'arête est une transition, ces coordonnées pourront donc être différentes.

Termes techniques

Nous listons ici quelques termes techniques associés à la paramétrisation de maillages, que nous aurons l'occasion d'utiliser régulièrement par la suite.

Une carte est la paramétrisation d'une portion du maillage délimitée par un ensemble de transitions (voire éventuellement par les bords du maillage). Chaque carte doit donc être homéomorphe à une portion de plan pour pouvoir être paramétrée de manière injective. Pour que la paramétrisation de l'ensemble de la surface soit injective, les cartes ne doivent pas se superposer dans l'espace paramétrique. Nous noterons les cartes avec le symbole C.

Une fonction de transition est une fonction associée à chaque transition. Pour une transition $e_{p,q}$ située entre les faces T_p et T_q du maillage et reliant deux sommets $\mathbf{x}_1$ et $\mathbf{x}_2$, la fonction de transition $\tau_{p,q}$ est l'application linéaire telle que $\tau_{p,q}(\mathbf{u}_1^p) = \mathbf{u}_1^q$, et $\tau_{p,q}(\mathbf{u}_2^p) = \mathbf{u}_2^q$. Elle permet donc de transformer les coordonnées paramétriques d'un point $\mathbf{x}$ de l'arête $e_{p,q}$ calculées par rapport à la face T_p en les coordonnées paramétriques de ce même point calculées pour la face T_q.

1.4.2 Calcul automatique d'une paramétrisation

Nous nous intéresserons ici au problème suivant : étant donné un maillage muni d'un ensemble de transitions, calculer pour chaque carte les coordonnées paramétrique de tous les sommets qu'elle couvre. Il s'agit notamment de s'assurer que la paramétrisation définie sur chaque carte est bien un homéomorphisme.

Calculer un homéomorphisme

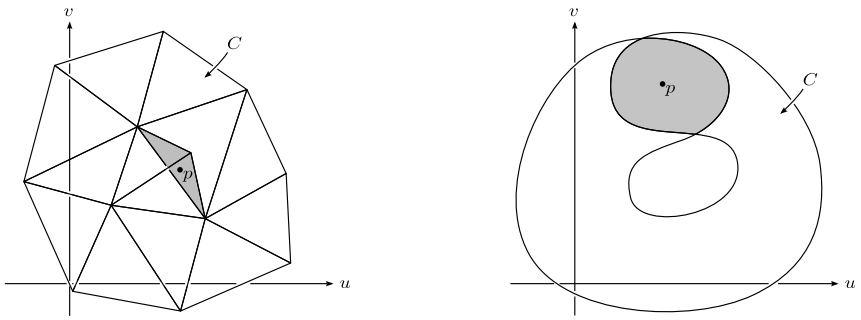
La paramétrisation d'une carte est une fonction linéaire par morceaux, un morceau étant un triangle. Pour que cette paramétrisation soit un homéomorphisme, il faut donc qu'elle respecte trois conditions.

Bijectivité : la paramétrisation ne sera pas bijective lorsque deux triangles d'une même carte se recouvrent au moins partiellement. Deux types de recouvrement sont en général distingués. Un recouvrement local, apparaît à cause du retournement d'un triangle du maillage dans l'espace paramétrique par rapport à ses triangles adjacents. Un recouvrement global correspond à deux portions distantes d'une même carte dont les triangles se recouvrent. Ces deux cas sont illustrés par la figure 1.9. Dans la plupart des applications, les recouvrements sont à proscrire. Dans le cas des paramétrisations préservant une grille que nous verrons pas la suite, les recouvrement globaux ne posent pas de problèmes.

Continuité : par définition, la paramétrisation est linéaire par morceaux et donc continue à l'intérieur de chaque triangle. Du point de vue global, la paramétrisation est discontinue au niveau d'une transition. Au niveau d'une carte, pour que la paramétrisation soit continue, il suffit que les triangles adjacents sur la surface restent adjacents dans la paramétrisation. Cette propriété est assurée si pour chaque arête de la carte (à l'exception des transitions), les cordonnées paramétriques des sommets de l'arête sont égales pour les deux faces adjacentes.

Continuité de l'inverse : si la paramétrisation est bijective, et donc exempte de recouvrements globaux ou locaux, la seule situation où une discontinuité peut apparaître dans la fonction inverse est lorsque deux triangles non adjacents sur la surface sont adjacents dans le domaine paramétrique. Il s'agit d'un cas dégénéré de recouvrement global.

Dans le cas des maillages et des paramétrisations linéaires par morceaux, les travaux de Tutte [111] sur la représentation des graphes planaires permettent d'obtenir une paramétrisation satisfaisant toutes les conditions. La méthode consiste à considérer la portion de maillage à paramétrer comme un système de ressorts en associant à chaque couple de sommets voisins $\mathbf{u}_i$ et $\mathbf{u}_j$ un ressort de longueur à vide nulle et de raideur $\rho_{i,j}$ pour les relier. En épinglant le bord de la région dans le domaine paramétrique, la minimisation de l'énergie de ce système de ressorts fournit une solution unique et sans recouvrement dès lors que les coefficients $\rho_{i,j}$ sont positifs, et que le bord forme un polygone convexe. Cette minimisation



(a) Recouvrement local (b) Recouvrement global

FIGURE 1.9 – Cas de non bijectivité de la paramétrisation d'une carte. Dans le cas local (a) un triangle du maillage a localement été retourné dans la paramétrisation, créant une zone de recouvrement. Dans le cas global (b), deux zones éloignées d'une même carte se recouvrent dans le domaine paramétrique.

se ramène à la résolution d'un système linéaire. Pour chaque sommet $\mathbf{x}_i$ interne de la région à paramétrer, ayant pour voisins $\mathbf{x}_1, \ldots, \mathbf{x}_k$ (appartenant éventuellement au bord), les coordonnées paramétriques $\mathbf{u}_i$ de ce sommet doivent respecter l'équation :

$$\mathbf{u}_i = \sum_{j=1}^{k} \lambda_{i,j} \mathbf{u}_j \tag{1.4.1}$$

$$\text{avec } \lambda_{i,j} = \frac{\rho_{i,j}}{\sum_{j=1}^{k} \rho_{i,j}}$$

$$\text{et } \rho_{i,j} > 0 \text{ pour tout } i,j \tag{1.4.2}$$

Ce système est séparable en u et v, et Tutte a montré qu'il admettait une solution sans recouvrement. Pour plus de détails sur le lien entre le système de ressorts et ces équations, nous invitons le lecteur à se référer au cours de Hormann *et al.* [51]. Pour le cas de la méthode de Tutte, le poids $\lambda_{i,j}$ est uniformément $1/k$ lorsque le sommet $\mathbf{x}_i$ possède k voisins, ce qui revient à placer ses coordonnées paramétriques au barycentre des coordonnées paramétriques de ses voisins. Le résultat de cette méthode est illustré sur la Figure 1.10.

Cette méthode est complètement automatique et fournit un homéomorphisme dans tous les cas. Elle montre donc qu'il est toujours possible de calculer un homéomorphisme linéaire par morceaux lorsque les transitions découpent la surface en une région homéomorphe à un disque. Elle a néanmoins plusieurs limitations. Tout d'abord, elle nécessite de fixer le bord du domaine sur un polygone convexe. Ensuite elle ne préserve pas la forme des faces du maillage. Ce second aspect est crucial pour de nombreuses applications, c'est pourquoi d'autres types de paramétrisation ont été étudiés.

Contraindre une paramétrisation

La qualité d'une paramétrisation est souvent évaluée en fonction de la distorsion qu'elle introduit. La distorsion correspond à l'altération des longueurs, des aires et des angles entre un objet mesuré dans le domaine paramétrique et sur la surface. Autrement dit, une paramétrisation sans distorsion sera telle que la distance euclidienne calculée dans le domaine

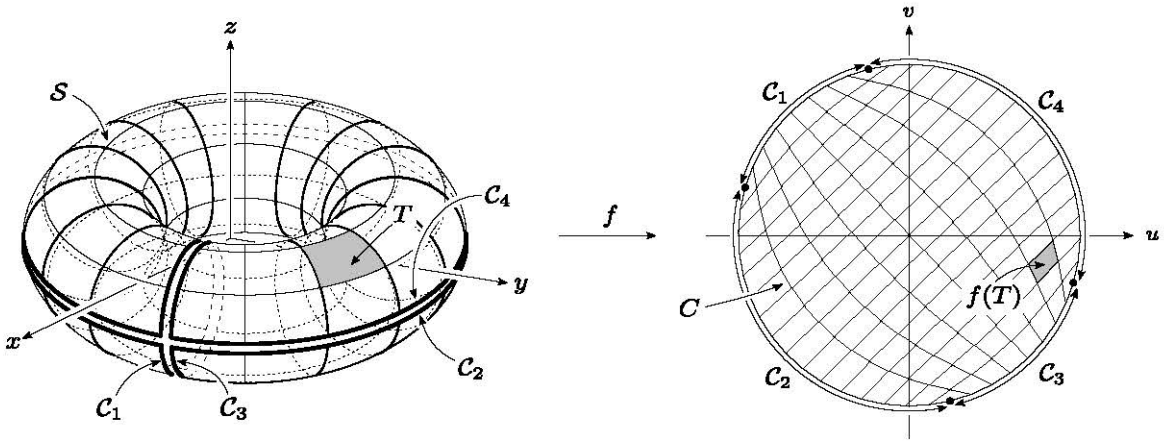
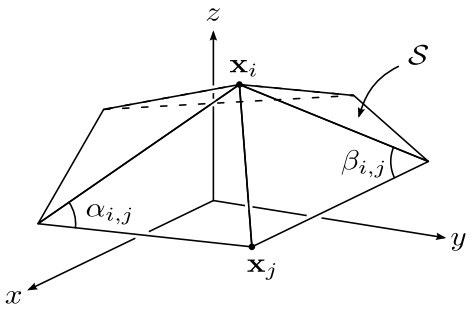


FIGURE 1.10 – Paramétrisation de Tutte pour le cas d'un tore. Sur le schéma, la surface $\mathcal{S}$ a été découpée le long des transitions, pour former quatre courbes circulaires $\mathcal{C}_1, \ldots, \mathcal{C}_4$. Les sommets du bord de l'unique carte C ont été placés sur un cercle pour assurer la convexité. Le résultat est bien une paramétrisation sans recouvrements. Cette méthode fonctionne avec des faces non triangulaires, et nous l'avons ici appliquée à un maillage quadrangulaire.

paramétrique entre deux points $\mathbf{u}_i$ et $\mathbf{u}_j$ soit égale à la distance géodésique entre les points $\mathbf{x}_i$ et $\mathbf{x}_j$ correspondants sur la surface. Une telle paramétrisation est dite *isométrique*. En pratique, seules les surfaces dites *développables* peuvent être paramétrées de façon isométrique. Il s'agit des surfaces pour lesquelles en tout point, l'une des courbures principales est nulle. C'est notamment le cas d'un cylindre ou d'un cône, mais pas le cas d'une sphère. Nous retrouvons ici le problème abordé en introduction de la création d'une carte de la terre sans distorsion. La terre n'étant pas développable, il n'est donc pas possible de calculer une paramétrisation sans distorsion. Les projections de Mercator et Peters fournissent chacune une solution visant à minimiser la distorsion, l'une en limitant l'altération des angles, et l'autre en limitant celle des aires. Une paramétrisation annulant les deux à la fois serait isométrique et donc irréalisable.

Dans le cas des surfaces maillées, il est donc toujours possible en définissant correctement les transitions de calculer une paramétrisation isométrique. Le problème est que le nombre de transitions requises rend en général la paramétrisation inutilisable. En effet, une paramétrisation isométrique préserve également les angles. Pour tout sommet $\mathbf{x}_i$ de la surface, si aucune transition n'est incidente à ce sommet, le sommet paramétrique correspondant $\mathbf{u}_i$ sera entouré de toutes les faces incidentes. La somme de tous les angles incidents au sommet doit donc nécessairement être 2π. Une paramétrisation isométrique préservant les angles, La somme des angles incidents sur la surface doit également être de 2π. Les transitions permettent de séparer le voisinage du sommet en plusieurs morceaux, et donc de s'affranchir de cette contrainte. Dans le cas général, la majorité des sommets d'une surface devront donc avoir une ou plusieurs transitions incidentes. Comme dans le cas où les faces étaient paramétrées indépendamment cette contrainte fait que travailler dans le domaine paramétrique est quasiment équivalent à travailler sur les faces individuelles du maillage.

À défaut de paramétrisation isométrique, il est possible de s'en rapprocher en relâchant la contrainte de préservation des longueurs. Plusieurs types d'approches peuvent grossièrement être définis.



$$\rho_{i,j} = cotan(\alpha_{i,j}) + cotan(\beta_{i,j})$$

FIGURE 1.11 – Poids cotangents pour les paramétrisations de type harmonique. Ces poids sont à insérer dans l'Équation 1.4.1 et la paramétrisation est obtenue en résolvant un système linéaire.

Les systèmes de poids généralisent la méthode de Tutte en modifiant les poids $\rho_{i,j}$ utilisés. Ces nouveaux poids sont souvent définis de telle sorte que si le maillage fourni est déjà plan, les sommets ne soient pas déplacés. Les poids les plus utilisés sont probablement les poids *cotangents* illustrés sur la Figure 1.11 qui ont été introduits par Pinkall et Polthier [83] et utilisés pour la première fois par Eck *et al.* [31] pour la paramétrisation. Ils ont ensuite été étendus par Desbrun *et al.* [23] qui ajoutent une méthode pour gérer automatiquement le bord de la région paramétrée. Ces poids ont l'inconvénient de devenir négatifs lorsque les triangles ont des angles obtus, ce qui n'assure plus la bijectivité de la paramétrisation. D'autres jeux de poids ont été élaborés pour éviter ce problème, comme ceux de Floater [40]. L'avantage de ces approches est qu'elles se ramènent à la résolution d'un système linéaire. La solution s'obtient donc facilement.

Les paramétrisations conformes consistent à essayer de préserver les angles, c'est à dire à faire en sorte que les angles des triangles dans le domaine paramétrique soient les mêmes que les angles des triangles correspondants sur la surface. Ces méthodes sont en particulier couramment utilisées pour le placage de texture que nous aborderons en Chapitre 2. Si les angles sont respectés au mieux, ces méthodes peuvent par contre introduire un facteur d'échelle local altérant la taille globale des triangles. Plusieurs approches visent à calculer une paramétrisation conforme. Lévy *et al.* [63] utilisent une discrétisation des équations de Cauchy-Riemann, ce qui amène à un système de moindres carrés. Cette formulation est en réalité équivalente à celle de Desbrun *et al.* [23]. Sheffer et de Sturler [101] travaillent directement sur les angles et minimisent l'erreur introduite par la paramétrisation, ce qui se ramène à un problème de minimisation non linéaire, traité par un solveur de type Newton. Une autre approche par Hormann et Greiner [50] consiste à minimiser une fonction objectif définie sur la matrice de la première forme fondamentale de la paramétrisation. Kharevych *et al.* [54] formalisent le problème sous forme d'agencement de cercles dans le domaine paramétrique. Une approche globale par Gu et Yau [46] permet de s'abstraire de la position des transitions, en calculant une base de l'ensemble des paramétrisations conformes d'une surface, à partir d'une base d'homologie de la surface. Une approche globale similaire par Ben-Chen *et al.* [6] fait le lien entre la distorsion d'échelle de la paramétrisation et la courbure de la surface. Cette méthode utilise un processus itératif pour calculer à la fois la position des singularités introduites par la paramétrisation, et la paramétrisation qui en découle.

Les méthodes authaliques ont pour objectif de préserver les aires. Contrairement aux paramétrisations conformes, ces méthodes n'introduisent pas de distorsion d'aire,

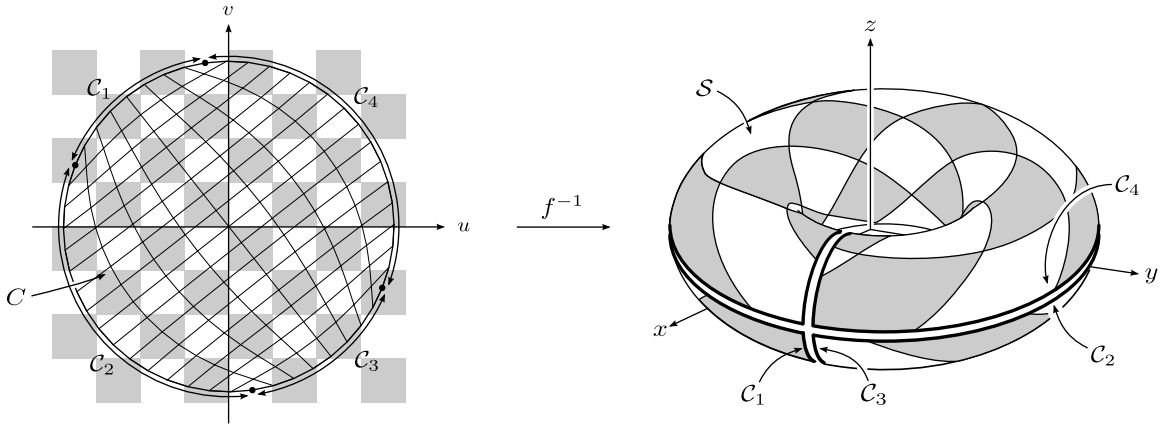


FIGURE 1.12 – Projection naïve de la grille $\mathbb{Z}^2$ sur le tore en utilisant la paramétrisaion de Tutte de la Figure 1.10. Sans contrainte sur la paramétrisation utilisée, les faces du remaillage sont déformées, et ne se recollent pas le long des transitions.

mais se permettent d'introduire une distorsion de type cisaillement. Zou *et al.* [123] proposent par exemple une méthode permettant d'optimiser une paramétrisation conforme pour en améliorer la préservation des aires.

Nous ne détaillerons pas plus ces méthodes ici, car les paramétrisation que nous avons utilisées pour nos travaux sont des paramétrisation préservant une grille, que nous décrirons dans un instant en Section 1.5. Une description plus étoffée de ces méthodes peut être consultée dans les différentes synthèses publiées sur le sujet. [41, 103, 51, 9].

1.5 Paramétrisation préservant une grille

1.5.1 Remaillage paramétrique

Les paramétrisations préservant une grille ont été mises au point pour répondre au problème du remaillage quadrangulaire. Étant donné un maillage quelconque, il s'agit alors de trouver un autre maillage pour représenter la même surface, mais ayant des faces quadrangulaires. Ces maillages sont utiles dans plusieurs cas de figure. Pour réaliser des simulations physiques, les méthodes fondées sur les éléments finis peuvent fournir des résultats plus précis ou plus rapides en utilisant des maillages quadrangulaires. Dans le cadre de cette thèse, nous utiliserons ces paramétrisations en Chapitre 2 pour former sur la surface un échantillonnage avec une structure de grille. La structure de grille nous permettra alors de plaquer des textures sur le maillages sans pour autant faire apparaître de coutures.

L'approche paramétrique du remaillage consiste à construire le nouveau maillage dans le domaine paramétrique, et à ensuite utiliser la paramétrisation pour projeter le maillage ainsi construit dans le domaine de plongement de la surface à remailler. L'avantage de travailler dans le domaine paramétrique est que tout point du domaine correspond à un point sur la surface. Il est ainsi possible de manipuler et déplacer des points sur la surface tout en travaillant dans le plan, et donc en bénéficiant des outils disponibles dans $\mathbb{R}^2$. Si la paramétrisation n'engendre pas trop de distorsion, la distance entre deux points dans le domaine paramétrique pourra être utilisés comme approximation de la distance géodésique entre ces points sur la surface. Il est ainsi envisageable d'utiliser des outils tels que la triangulation de Delaunay, permettant de calculer automatiquement un ensemble de triangles pour relier un

ensemble de points de $\mathbb{R}^2$. La méthode choisie pour le remaillage quadrangulaire en utilisant une paramétrisation consiste à remailler en utilisant la grille $\mathbb{Z}^2$. Les points du domaine paramétrique correspondant aux sommets de cette grille produisent les nouveaux sommets du maillage, reliés entre eux par des faces quadrangulaires selon la connectivité naturelle de la grille. Cette méthode est illustrée sur la Figure 1.12. Deux types de problèmes apparaissent alors : tout d'abord, le recollement des faces au niveau des arêtes n'est absolument pas assuré ; ensuite, l'orientation et la taille des faces du remaillage est entièrement contrôlée par la paramétrisation, qui doit donc être calculée en fonction des souhaits de l'utilisateur pour le remaillage.

Les transitions de la paramétrisation sont le problème majeur du remaillage. Les paramétrisations préservant une grille visent à répondre à ce problème. Elles sont en général réalisées en deux étapes. Tout d'abord un champ de direction est calculé pour contrôler la disposition des faces quadrangulaires du remaillage. Aligner ces faces avec un champ de directions donné consiste ensuite à calculer une paramétrisation dont le gradient s'aligne avec le champ de directions. Nous verrons donc dans un premier temps les procédés utilisés pour générer des champs de directions sur une surface. Ensuite une seconde étape intègre le champ de directions fourni pour obtenir une paramétrisation respectant à la fois les directions données par le champ et les conditions nécessaires au recollement de la grille au niveau des transitions.

Il existe d'autres méthodes pour le remaillage quadrangulaire dans l'espace paramétrique que celle des paramétrisations préservant une grille, notamment les méthodes de *maillage frontal*. Nous nous restreignons ici aux paramétrisations préservant une grille car notre objectif n'est pas le remaillage quadrangulaire, mais le placage de texture.

1.5.2 Champs de directions

Champs de vecteurs et singularités

Nous commencerons ici par définir la notion de *champ de vecteurs*, qui nous sera utile lors du remaillage pour pouvoir contrôler l'orientation des faces du remaillage. Comme précédemment, une description plus formelle et détaillée est donnée dans l'ouvrage de Do Carmo [24]. Un champ de vecteurs sur une surface $\mathcal{S}$ est une application qui a tout point $\mathbf{x}$ de $\mathcal{S}$ associe un vecteur $\mathbf{w}(\mathbf{x})$ du plan tangent $T_{\mathbf{x}}\mathcal{S}$. L'application qui à un point $\mathbf{x}$ associe la direction de courbure maximale $\mathbf{e}_1$ est un exemple de champ de vecteurs. Nous nous intéresserons par la suite au cas des champs de vecteurs continus. Une *singularité* d'un champ de vecteurs est un point $\mathbf{x}$ tel que $\mathbf{w}(\mathbf{x}) = \mathbf{0}$. Nous supposerons également désormais que les singularités des champs de vecteurs que nous utiliserons seront isolées, c'est à dire qu'il existe une petite région (un ouvert) de $\mathcal{S}$ contenant $\mathbf{x}$ tel que $\mathbf{w}$ restreint à cette région n'ait pas d'autre singularité qu'en $\mathbf{x}$. En particulier, cette propriété implique que l'ensemble des singularités est fini.

À chaque singularité, il est possible d'associer un *indice* I_k avec $k \in \{1, \ldots, a\}$. Nous nous contenterons ici d'une définition informelle de cette notion. Étant donné une singularité de $\mathbf{w}$ en un point $\mathbf{x}$ de $\mathcal{S}$, I correspondra au nombre de tour que le champ de vecteurs effectuera autour de votre doigt lorsque vous lui ferez suivre une boucle infinitésimale autour de $\mathbf{x}$ en tournant dans le sens positif par rapport à la normale. Le champ de vecteurs étant continu, l'indice correspondra toujours à un entier relatif. Une singularité d'indice 1 est illustrée sur la Figure 1.13.

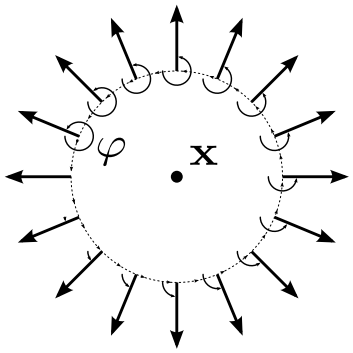


FIGURE 1.13 – Le champ de vecteurs défini dans le plan par $\mathbf{w}(x,y) = (x,y)^t$, a une singularité en $\mathbf{x} = (0,0)$. Après avoir réalisé un tour le long d'une boucle autour de la singularité, le champ de vecteurs aura parcouru un angle $\varphi = 2\pi$ par rapport à sa position initiale. L'indice de cette singularité vaudra donc 1.

Un champ de vecteurs sur une surface $\mathcal{S}$ possède un nombre fini a de singularités. Les indices $\{I_k\}_{k=1}^{a}$ de ces singularités sont liés dans le cas d'une surface sans bords à la caractéristique d'Euler Poincaré $\chi(\mathcal{S})$ de la surface $\mathcal{S}$ sur laquelle est défini le champ de vecteurs. Cette relation constitue le *théorème de Poincaré Hopf* :

$$\sum_{k=1}^{a} I_k = \chi(\mathcal{S}) \tag{1.5.1}$$

Ce résultat signifie que les singularités d'un champ de vecteurs défini sur une surface sont contraintes par sa topologie. En particulier, un champ de vecteurs défini sur une surface de caractéristique d'Euler Poincaré non nulle aura nécessairement des singularités.

Champs de directions

Aligner un remaillage quadrangulaire sur un champ de vecteurs consiste simplement à faire en sorte que les arêtes de la quadrangulation soient soit dans la direction du champ de vecteurs, soit orthogonales. Une quadrangulation alignée sur un champ de vecteurs $\mathbf{w}$ sera donc également alignée sur tout champ de vecteur obtenu en faisant tourner $\mathbf{w}$ d'un angle multiple de $\frac{\pi}{2}$. De plus, la norme de $\mathbf{w}$ n'a pas d'importance. Pour utiliser ces degrés de liberté supplémentaires, il est possible d'utiliser la notion de *champ de directions symétriques* de degré N. Un tel champ associe à tout point $\mathbf{x}$ un vecteur de référence $\mathbf{w}_0(\mathbf{x})$ ainsi que les vecteurs $\{\mathbf{w}_k\}_{k=1}^{N-1}$ obtenus à partir de $\mathbf{w}_0$ par une rotation d'angle $\frac{2k\pi}{N}$.

Un champ de vecteurs continu peut être utilisé comme ensemble de vecteurs de référence $\mathbf{w}_0$ pour définir un champ de directions symétrique continu. L'opération inverse n'est pas contre pas toujours réalisable, comme le montre la Figure 1.14. La notion de singularité définie pour le cas des champs de vecteurs peut également s'étendre pour le cas des champs de directions symétriques. Les singularités apparaissent également aux points $\mathbf{x}$ tels que $\mathbf{w}_0(\mathbf{x}) = \mathbf{0}$. Leurs indices, contrairement aux champs de vecteurs ne sont plus forcément entiers, mais sont des multiples de $\frac{1}{N}$ où N est le degré de symétrie du champ. Comme dans le cas des champs de vecteurs, lorsque la surface n'a pas de bords, le théorème de Poincaré Hopf donné par l'Équation 1.5.1 reste valable avec ces nouveaux indices de singularités.

Génération sur des surfaces maillées

Les champs de vecteurs sont utilisées dans des applications très diverses. Outre le remaillage quadrangulaire, ils sont également utilisés pour guider un processus de synthèse de texture par Praun *et al.* [88], Wei et Levoy [116], ou encore dans le domaine du rendu non photoréaliste par Hertzmann et Zorin [48], Zhang *et al.* [121], Palacios et Zhang [78] pour

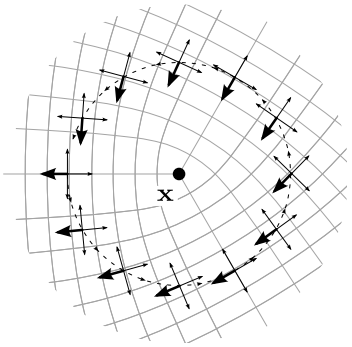


FIGURE 1.14 – Un exemple de champ de directions continu de symétrie 4 à partir duquel il n'est pas possible de produire un champ de vecteurs continu. En suivant une boucle autour du point central **x** comme pour la Figure 1.13, le champ de directions effectue une rotation d'angle $\frac{\pi}{2}$. Le point **x** est donc une singularité d'indice $\frac{1}{4}$

distribuer des primitives – hachures, coups de pinceau – sur la surface. Leur construction a donc fait l'objet de beaucoup d'études, mettant l'accent sur différents aspects :

Continuité : ces approches visent à générer un champ de directions le plus lisse possible. Un certain nombre de méthodes visent à générer un champ de directions à partir de la courbure. À cause des zones ombilicales (voir la Figure 1.5), la courbure ne peut pas directement être utilisée. Deux éléments sont donc optimisés simultanément : la continuité du champ de vecteur et l'alignement avec une direction fournie, comme la courbure. Praun *et al.* [88] utilisent des fonctions à base radiales pour interpoler sur la surface un ensemble épars de contraintes fournies par l'utilisateur. Hertzmann et Zorin [48], Palacios et Zhang [78] échantillonnent le champ de directions sur les sommets du maillage, et utilisent l'arête reliant deux sommets voisins pour définir un repère permettant de mesurer l'angle entre les vecteurs du champ définis au niveau des deux sommets. L'objectif à remplir consiste à minimiser cet angle pour toute paire de sommets voisins. Zhang *et al.* [122, 121] définissent un champ de vecteur comme la somme de champs de vecteurs de base, un par contrainte utilisateur. Le champ obtenu peut ensuite être lissé par un lissage de type Laplacien. Ray *et al.* [95, 94], Bommes *et al.* [10] échantillonnent le champ sur les faces et évaluent la continuité du champ au niveau d'une arête, en dépliant localement les deux triangles voisins le long de cette arête. Une fois les deux triangles dans le même plan, l'angle entre leurs directions respectives peut être mesuré. Ray *et al.* [94], Bommes *et al.* [10] permettent également à l'utilisateur de spécifier des contraintes directionnelles locales sur le champ. Une autre méthode est celle de Gu et Yau [46], qui recherchent une paramétrisation conforme de la surface. Cette approche consiste à calculer la paramétrisation par son gradient, qui est un champ de vecteurs sur la surface. L'intérêt de passer par le gradient est que la paramétrisation est indépendante de la position des transitions. En fonction de la topologie de la surface, une base de l'ensemble des paramétrisations conformes est calculée, puis une paramétrisation est sélectionnée sur cette base pour minimiser la distorsion introduite.

Contrôle de la topologie du champ : il s'agit de contrôler l'emplacement et l'indice des singularités du champ de directions généré. Quelle que soit la méthode employée, la somme des indices de singularités est contrainte par la topologie de la surface via le théorème de Poincaré Hopf. Dans le cas des méthodes générant des champs de vecteurs (Gu et Yau [46], Wei et Levoy [116]), les indices des singularités sont restreints aux valeurs entières. Ces approches permettent de placer manuellement des singularités sur la surface, qui seront prises en compte lors du calcul du champ. Zhang *et al.* [121] étendent la technique de Zhang *et al.* [122] pour manipuler des champs de tenseurs, qui sont des champs de directions de symétrie 2, et permettent donc des singularités d'indices multiples de $\frac{1}{2}$. Leur interface offre à l'utilisateur la possibilité

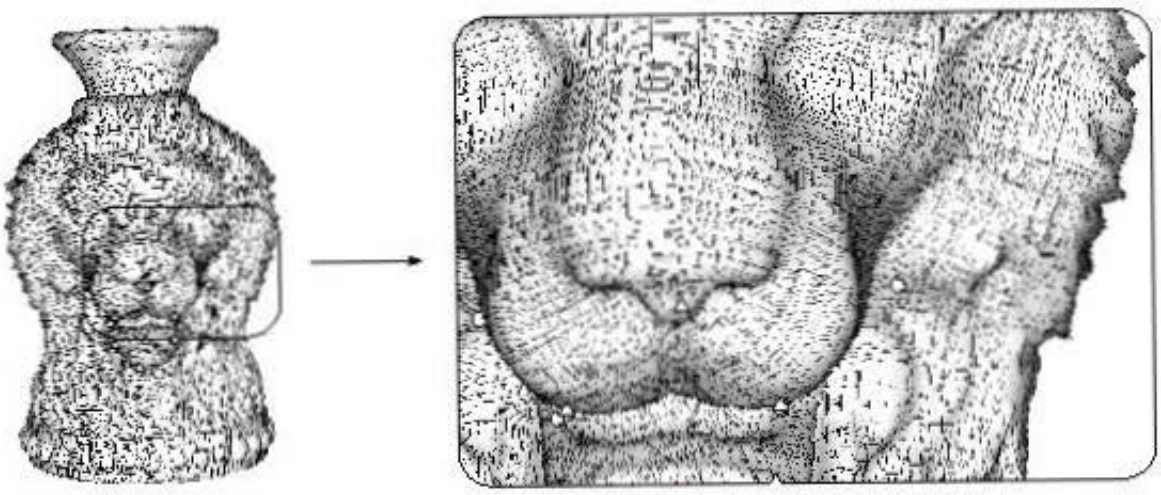


FIGURE 1.15 – Un exemple de champ de directions généré par la méthode de Ray *et al.* [94]. Les directions du champ sont alignées sur les courbures calculées sur la Figure 1.5, mais les incohérences dues aux zones ombilicales ont été supprimées. Les singularités du champ sont symbolisées par les petits polygones. Un triangle est une singularité d'indice $\frac{1}{4}$ et un pentagone une singularité d'indice $-\frac{1}{4}$.

de placer des singularités et des contraintes directionnelles. Une fois le champ généré, il est possible de déplacer des singularités, ou d'annuler des paires de singularités proches ayant des indices opposés. Ray *et al.* [95] théorisent la topologie des champs de directions à symétrie 4 et introduit une méthode permettant à l'utilisateur de fournir exactement les positions des singularités. Hertzmann et Zorin [48] et Ray *et al.* [94] laissent les singularités apparaître automatiquement à partir de la courbure, mais permettent d'éviter l'apparition de trop nombreuses singularités à l'aide d'un lissage, sur le maillage dans les travaux de Hertzmann et Zorin [48], et sur les défauts angulaire dans les travaux de Ray *et al.* [94].

La méthode que nous avons choisi d'utiliser pour calculer les paramétrisation préservant une grille que nous utiliserons en Chapitre 2 est celle de Ray *et al.* [94], car elle permet d'obtenir un champ de directions de bonne qualité, à partir de la courbure, de manière complètement automatique. Un exemple de champ de directions calculé par cette méthode est fourni sur la Figure 1.15. Le résultat est donc un champ de directions stocké sous la forme d'un angle θ_p par face T_p du maillage, défini par rapport à une arête de référence e_p choisie pour la face. Au niveau de chaque arête $e_{p,q}$ reliant les faces T_p et T_q, un *saut de période* $s_{p,q}$ – « period jump » pour Ray *et al.* [94], Bommes *et al.* [10], « layer shift » pour Kälberer *et al.* [53] – est défini comme un entier codant l'angle entre les directions de référence de deux faces adjacentes. Ainsi avec un saut de période $s_{p,q}$, l'angle entre les deux directions de référence est $\frac{\pi}{2} s_{p,q}$ et une direction $\mathbf{w}_k$ avec $k \in \{0,1,2,3\}$ de la face T_p correspond à la direction $\mathbf{w}_{k-s_{p,q}}$ de la face T_q (les indices étant pris module 4). Nous avons donc en particulier $s_{p,q} = -s_{q,p}$. Ces notations sont illustrées sur la Figure 1.16. Lorsque la somme des sauts de période de toutes les arêtes adjacentes à un même un sommet est non nulle, le sommet est une singularité du champ de directions.

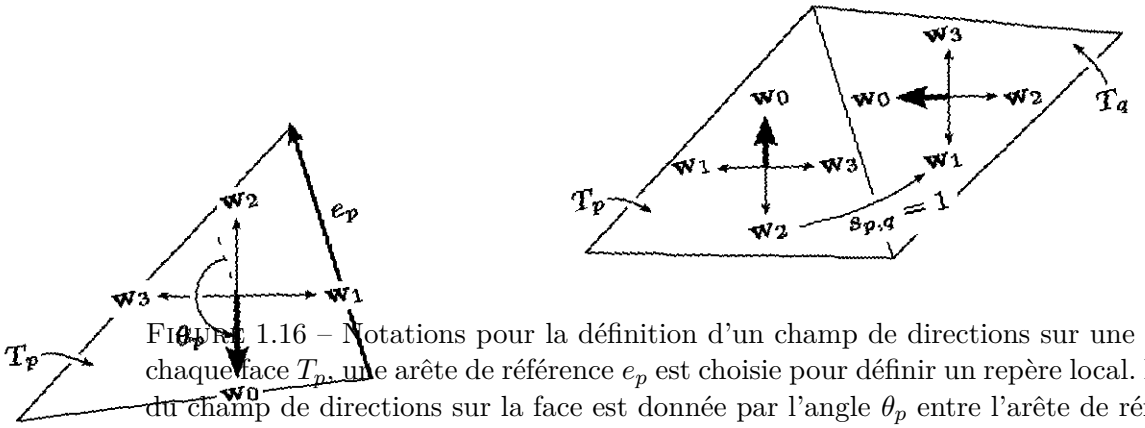


FIGURE 1.16 – Notations pour la définition d'un champ de directions sur une surface. Sur chaque face T_p, une arête de référence e_p est choisie pour définir un repère local. La direction du champ de directions sur la face est donnée par l'angle θ_p entre l'arête de référence et le vecteur de référence $\mathbf{w}_0$ du champ. Entre deux faces T_p et T_q, le saut de période $s_{p,q}$ permet d'indiquer comment les directions du champ doivent être mises en correspondance. Dans le cas de la figure de droite, il vecteur $\mathbf{w}_0$ de T_p est mis en correspondance avec le vecteur $\mathbf{w}_3$ de T_q, d'où un saut de période $s_{p,q} = 1$.

1.5.3 Paramétrisation

Comme mentionné plus tôt, pour aligner les éléments quadrangulaires du remaillage avec un champ de directions fourni, il est nécessaire de faire en sorte que le champ de directions corresponde au gradient de la paramétrisation. Autrement dit, le problème consiste à intégrer le champ de directions. Cette contrainte n'est pas la seule à respecter, car il faut également s'assurer que le recollement au niveau des transitions de la paramétrisation se passe correctement.

Singularités et remaillage quadrangulaire

Les singularités du champ de directions sous-jacent à la paramétrisation ont une importance particulière pour le remaillage. Nous verrons qu'elles introduisent des difficultés pour le placage de textures sans coutures (Chapitre 2). Un tel sommet nécessite également un traitement particulier dans le cadre des surfaces de subdivision (les méthodes les plus courantes sont celles de Catmull et Clark [16] et Doo et Sabin [25]). Pour un remaillage quadrangulaire, une connectivité régulière correspond à faire en sorte que chaque sommet soit adjacent à quatre faces. Dans le cas d'un maillage sans bords, si cette propriété est vérifiée sur tout le maillage, des relations apparaissent entre le nombre de sommets $|\mathbf{X}|$, de faces $|\mathbf{T}|$ et d'arêtes $|\mathbf{E}|$:

$$|\mathbf{E}| = 2|\mathbf{X}| = 2|\mathbf{T}|$$

Ces relations se traduisent par une caractéristique d'Euler Poincaré

$$\chi = |\mathbf{T}| - |\mathbf{E}| + |\mathbf{X}| = 0$$

Ainsi un remaillage quadrangulaire d'une surface sans bord avec une connectivité parfaitement régulière n'est possible que pour les surfaces de caractéristique d'Euler Poincaré nulle, à savoir les tores. Pour les autres surfaces, il est nécessaire d'insérer des sommets de connectivité irrégulière. En notant k_l le nombre de sommets ayant l faces voisines, la

relation entre le nombre de faces et de sommets devient :

$$\begin{aligned} 4|\mathbf{T}| &= \sum_{l=1}^{\infty} l k_l \\ |\mathbf{T}| &= \sum_{l=1}^{\infty} \left(1 + \frac{l-4}{4}\right) k_l \\ |\mathbf{T}| &= |\mathbf{X}| + \sum_{l=1}^{\infty} \frac{l-4}{4} k_l \end{aligned}$$

En remplaçant dans l'expression de la caractéristique d'Euler Poincaré, il vient :

$$\begin{aligned} \chi &= |\mathbf{T}| - |\mathbf{E}| + |\mathbf{X}| \\ &= |\mathbf{T}| - 2|\mathbf{T}| + |\mathbf{X}| \\ &= |\mathbf{X}| - |\mathbf{T}| \\ &= \sum_{l=1}^{\infty} \frac{4-l}{4} k_l. \end{aligned}$$

Un sommet de valence trois contribue donc à la caractéristique d'Euler Poincaré d'un facteur $\frac{1}{4}$ alors qu'un sommet de valence cinq d'un facteur $-\frac{1}{4}$. Ce comportement coïncide parfaitement avec les singularités d'un champ de direction. Dans le cas d'un remaillage quadrangulaire aligné sur un champ de directions, les arêtes du remaillage correspondent aux lignes du champ. Comme le montre la Figure 1.14, une singularité d'indice $\frac{1}{4}$ correspond bien à des lignes de champ formant une sommet de valence trois dans une quadrangulation.

Pour pouvoir faire apparaître un sommet de valence irrégulière dans l'image de la grille $\mathbb{Z}^2$ par la paramétrisation, il est nécessaire que le point $\mathbf{u}$ de la grille qui correspondra au sommet corresponde exactement à un sommet paramétrique $\mathbf{u}_i$ de $\mathcal{S}$. Si ce n'est pas le cas, l'image par la paramétrisation de $\mathbf{u}$ apparaîtra à l'intérieur d'une face de S, et les quatre cellules adjacentes de la grille $\mathbb{Z}^2$ seront disposées autour. De plus, il est nécessaire que le sommet $\mathbf{u}_i$ correspondant soit adjacent à une transition. Sinon, les quatre cellules de la grille $\mathbb{Z}^2$ qui lui sont adjacentes seront toutes envoyées sur les faces adjacentes au sommet $\mathbf{x}_i$ correspondant à $\mathbf{u}_i$, et le sommet du remaillage aura également une valence régulière. Comme indiqué en Section 1.5.2, page 41, les singularités du champ de directions apparaissent au niveau des sommets du maillage pour lesquels la somme des sauts de période des arêtes adjacentes est non nulle. Une singularité affectant un sommet $\mathbf{x}_i$ implique donc que les coordonnées paramétriques $\mathbf{u}_i^p$ pour toute face incidente T_p à ce sommet doivent être entières.

Calcul des transitions

Pour paramétrer la surface, il est nécessaire de définir un ensemble de cartes homéomorphes à des disques. Les paramétrisations préservant une grille sont des paramétrisations dites globales, dans le sens où le gradient de la paramétrisation est défini par le champ de directions, et ne dépend donc pas de l'ensemble de transitions utilisé pour paramétrer la surface. Une méthode simple pour obtenir un ensemble de transitions $\mathbf{E}_\tau$ adéquat consiste à effectuer un parcours en largeur du graphe des faces pour obtenir un arbre couvrant, et à prendre comme transitions l'ensemble des arêtes qui n'ont pas été sélectionnées dans l'arbre couvrant. Cet ensemble initial de transitions peut être ensuite simplifié en supprimant itérativement toutes les transitions qui sont adjacentes à un sommet n'ayant pas

d'autre transition adjacente. Cette passe de simplification supprime toutes les chaînes de transitions qui ne font pas partie d'une boucle. Une exception pour les surfaces homéomorphes à des sphères est nécessaire pour conserver au moins deux arêtes dans l'ensemble de transitions. L'Algorithme 1.1 formalise la construction de l'ensemble initial des transitions. Le résultat de cet algorithme est un ensemble de transition permettant d'obtenir une unique carte homéomorphe à un disque quelque soit le genre de la surface.

En plus de cet ensemble de transitions, il est nécessaire de s'assurer que les sommets sur lesquels des singularités apparaissent ont au moins une transition incidente. L'ensemble initial des transitions est donc étendu en ajoutant itérativement pour chaque singularité la suite d'arêtes la plus petite la connectant aux transitions déjà calculées. L'ensemble de transitions final $\mathbf{E}_\tau$ permet de paramétrer $\mathcal{S}$ en une seule carte, ayant toutes les singularités sur son bord.

Entrées :
– $\mathcal{S}$ une surface polygonale
Sorties :
– $\mathbf{E}_\tau$ un ensemble de transitions.

```
1  Algorithme
2      Soit E_τ un ensemble d'arêtes vide ;
3      Soit File une file vide ;
4      Soit T_init une face de S, ajouter (T_init,T_init) en queue de File ;
5      Marquer T comme traitée;
6      tant que la File n'est pas vide faire
7          Retirer (T_o,T_p) de la tête de File ;
8          // T_o est la face par laquelle T_p a été propagée
9          pour chaque face T_q différente de T_o voisine de T_p faire
10             si T_q est marquée comme traitée alors
11                 Soit e_{p,q} l'arête reliant T_p et T_q ;
12                 E_τ ←+ e_{p,q} ;
13             sinon
14                 Ajouter (T_p,T_q) en queue de File ;
15                 Marquer T_q comme traitée ;
16     tant que E_τ varie et |E_τ| ≥ 2 faire
17         si S possède un sommet n'ayant qu'une unique transition incidente alors
18             Soit e_{p,q} la transition correspondante;
19             E_τ ←− e_{p,q};
20     renvoyer E_τ
```

Algorithme 1.1: `découpe_disque(`$\mathcal{S}$`)` fournit un ensemble de transitions permettant de découper un surface maillée en une unique carte homéomorphe à un disque. Cette algorithme est un simple parcours en largeur du graphe d'adjacence des faces de $\mathcal{S}$.

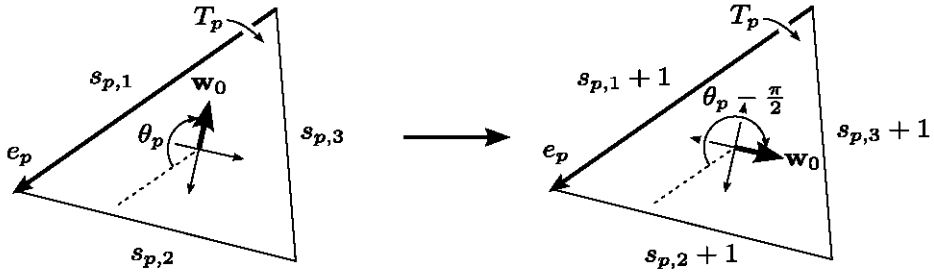


FIGURE 1.17 – Normalisation du champ de directions. En modifiant l'angle θ_p et les sauts de période $s_{p,1}$, $s_{p,2}$, $s_{p,3}$ relatifs à la face T_p, il est possible de changer le vecteur de référence utilisé pour le champ dans cette face.

Alignement de la paramétrisation sur le champ

Pour aligner la paramétrisation avec le champ de directions, il faut pour chaque face T_p faire correspondre le vecteur de référence $\mathbf{w}_0$ de cette face soit avec la direction du paramètre u soit avec la direction du paramètre v de l'espace paramétrique. Ce choix ne peut pas être réalisé indépendamment sur chaque face. Rappelons que le champ de directions est constitué de quatre vecteurs $\mathbf{w}_k, k \in \{0,1,2,3\}$ pour chaque face. Entre deux faces T_p et T_q, les directions sont mises en correspondance avec le saut de période $s_{p,q}$ défini sur l'arête $e_{p,q}$ qui joint les faces (voir Figure 1.16). Ainsi, en choisissant d'aligner le vecteur $\mathbf{w}_0$ de T_p sur la direction du paramètre u, la direction $\mathbf{w}_{-s_{p,q}}$ de la face T_q doit également être alignée sur cette direction, et de proche en proche, le choix est contraint sur l'ensemble des faces de $\mathcal{S}$.

Il est possible de normaliser le champ de directions pour faire en sorte que pour toutes les faces, la direction $\mathbf{w}_0$ soit alignée avec la direction du paramètre u. Cette normalisation se fait en utilisant la même propagation que pour le calcul des transitions présenté dans l'Algorithme 1.1. Au niveau de la ligne 8, une étape de normalisation du champ est insérée pour faire en sorte que les directions $\mathbf{w}_0$ des faces T_o et T_p soient alignées, ce qui se traduit par un saut de période $s_{o,p} = 0$ sur l'arête $e_{o,p}$. Cette normalisation se fait en modifiant le vecteur de référence du champ de la face T_p comme illustré sur la Figure 1.17. Une fois la normalisation effectuée, seules les transitions peuvent avoir un saut de période non nulle. Nous supposerons donc désormais que pour chaque face, la direction à aligner avec la direction du paramètre u est $\mathbf{w}_0$.

Nous allons maintenant voir comment calculer les coordonnées paramétriques des sommets de la surface pour aligner le gradient de la paramétrisation avec le champ de directions. Étant donné un triangle T_p, dont les trois sommets $\mathbf{x}_1$, $\mathbf{x}_2$ et $\mathbf{x}_3$ ont respectivement les coordonnées paramétriques $\mathbf{u}_1^p$, $\mathbf{u}_2^p$, $\mathbf{u}_3^p$, les coordonnées paramétriques $\mathbf{u}$ d'un point $\mathbf{x}$ quelconque de ce triangle sont données par :

$$\mathbf{u} = \alpha_1 \mathbf{u}_1^p + \alpha_2 \mathbf{u}_2^p + \alpha_3 \mathbf{u}_3^p$$

où les α_i sont les coordonnées barycentriques de $\mathbf{x}$ par rapport aux sommets $\mathbf{x}_i$. La coordonnée barycentrique α_1 de $\mathbf{x}$ est donnée par :

$$\alpha_1 = \frac{((\mathbf{x}_2 - \mathbf{x}_3) \times (\mathbf{x}_3 - \mathbf{x})) \cdot \mathbf{n}_{T_p}}{|T_p|}$$

où $\mathbf{n}_{T_p}$ est la normale de T_p (Équation 1.3.1) et $|T_p|$ est son aire Dans la suite nous utiliserons $\mathbf{n} = \mathbf{n}_{T_p}$. La différentielle de α_1 pour un petit déplacement $\mathrm{d}\mathbf{x}$ de $\mathbf{x}$ est :

$$\begin{aligned} \mathrm{d}\alpha_1 &= \frac{((\mathbf{x}_2 - \mathbf{x}_3) \times (-\mathrm{d}\mathbf{x})) \,.\mathbf{n}}{|T_p|} \\ &= \frac{(\mathbf{x}_2 - \mathbf{x}_3) \times \mathbf{n}}{|T_p|}.\mathrm{d}\mathbf{x} \end{aligned}$$

En combinant ces différentielles, celle de $\mathbf{u}$ est

$$\mathrm{d}\mathbf{u} = \left(\frac{(\mathbf{x}_2 - \mathbf{x}_3) \times \mathbf{n}}{|T_p|}\mathbf{u}_1^p + \frac{(\mathbf{x}_3 - \mathbf{x}_1) \times \mathbf{n}}{|T_p|}\mathbf{u}_2^p + \frac{(\mathbf{x}_1 - \mathbf{x}_2) \times \mathbf{n}}{|T_p|}\mathbf{u}_3^p \right).\mathrm{d}\mathbf{x}$$

L'alignement de la paramétrisation sur le champ de directions pour la face T_p se traduit donc par les contraintes :

$$\begin{cases} \mathbf{w}_0 = h\left(\frac{(\mathbf{x}_2 - \mathbf{x}_3) \times \mathbf{n}}{|T_p|}u_1^p + \frac{(\mathbf{x}_3 - \mathbf{x}_1) \times \mathbf{n}}{|T_p|}u_2^p + \frac{(\mathbf{x}_1 - \mathbf{x}_2) \times \mathbf{n}}{|T_p|}u_3^p \right) \\ \mathbf{w}_1 = h\left(\frac{(\mathbf{x}_2 - \mathbf{x}_3) \times \mathbf{n}}{|T_p|}v_1^p + \frac{(\mathbf{x}_3 - \mathbf{x}_1) \times \mathbf{n}}{|T_p|}v_2^p + \frac{(\mathbf{x}_1 - \mathbf{x}_2) \times \mathbf{n}}{|T_p|}v_3^p \right) \end{cases}$$

où h est un paramètre inséré pour permettre de contrôler l'échelle de la paramétrisation, et donc le nombre d'éléments du remaillage. Nous obtenons donc un système pour le paramètre u et un pour le paramètre v. Même si chacun de ces systèmes semble offrir trois équations pour trois inconnues, ils sont en réalité de rang 2, et un degré de liberté subsiste. Ce degré de liberté vient du fait que le gradient de la paramétrisation est inchangé lorsque les coordonnées paramétriques sont uniformément translatées. Cette translation est la constante qui apparaît lors de l'intégration du champ de directions.

Ce système peut être simplifié en remarquant que :

$$\begin{cases} \mathbf{w}_0.(\mathbf{x}_1 - \mathbf{x}_2) = h(u_1^p - u_2^p) \\ \mathbf{w}_0.(\mathbf{x}_2 - \mathbf{x}_3) = h(u_2^p - u_3^p) \\ \mathbf{w}_0.(\mathbf{x}_3 - \mathbf{x}_1) = h(u_3^p - u_1^p) \end{cases} \quad \text{et} \quad \begin{cases} \mathbf{w}_1.(\mathbf{x}_1 - \mathbf{x}_2) = h(v_1^p - v_2^p) \\ \mathbf{w}_1.(\mathbf{x}_2 - \mathbf{x}_3) = h(v_2^p - v_3^p) \\ \mathbf{w}_1.(\mathbf{x}_3 - \mathbf{x}_1) = h(v_3^p - v_1^p) \end{cases} \tag{1.5.2}$$

Ce sont ces équations que nous intégrerons dans le système global calculant la paramétrisation.

Fonctions de transition

Les coordonnées paramétriques des sommets du maillage ne peuvent pas être calculées indépendamment. Soit $e_{p,q}$ l'arête reliant les faces T_p et T_q, et soient $\mathbf{x}_1$ et $\mathbf{x}_2$ les deux sommets aux extrémités de cette arête. Si $e_{p,q}$ n'est pas une transition, alors les coordonnées paramétriques des sommets $\mathbf{x}_1$ et $\mathbf{x}_2$ doivent être identiques pour les faces T_p et T_q :

$$\begin{cases} \mathbf{u}_1^p = \mathbf{u}_1^q \\ \mathbf{u}_2^p = \mathbf{u}_2^q \end{cases} \tag{1.5.3}$$

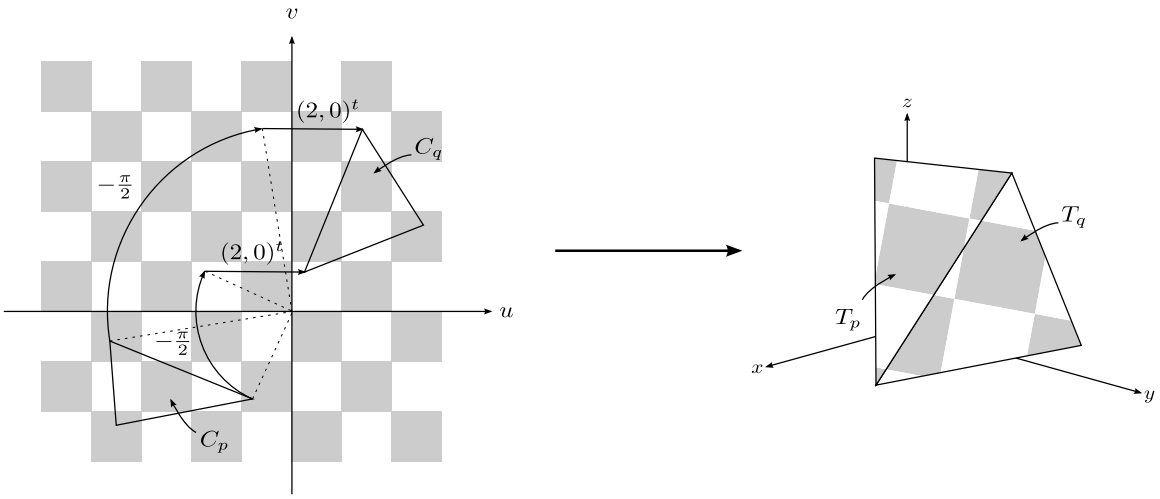


FIGURE 1.18 – Fonction de transition au niveau de l'arête $e_{p,q}$ reliant les faces T_p et T_q. $\tau_{p,q}$ est la composition d'une rotation d'angle $-\frac{\pi}{2}$ et d'une translation à coordonnées entières, de vecteur $(2,0)^t$. À droite, l'image de la grille $\mathbb{Z}^2$ sur la surface se recolle correctement au niveau de la transition.

Lorsque l'arête $e_{p,q}$ est une transition, il faut s'assurer que la quadrangulation se recolle correctement. Les contraintes permettant de s'en assurer consistent à restreindre les fonctions de transitions de la paramétrisation (Section 1.4.1). Les éléments quadrangulaires pourront être mis en correspondance si les fonctions de transition préservent une grille. Plus spécifiquement, la fonction de transition $\tau_{p,q}$ entre les faces paramétriques C_p et C_q correspondant respectivement à T_p et T_q doit être une application linéaire directe (préservant l'orientation) telle que :

$$\begin{cases} \tau_{p,q}(\mathbf{u}_1^p) = \mathbf{u}_1^q\,; \\ \tau_{p,q}(\mathbf{u}_2^p) = \mathbf{u}_2^q\,; \\ \tau_{p,q} \text{ est une bijection de } \mathbb{Z}^2 \text{ (elle préserve la grille } \mathbb{Z}^2). \end{cases}$$

Les applications linéaires répondant à ces critères sont les translations de vecteur à coordonnées entières, et les rotations d'angle multiple de $\frac{\pi}{2}$. La rotation est donnée par le saut de période $s_{p,q}$. Les contraintes sur les coordonnées paramétriques des sommets $\mathbf{x}_1$ et $\mathbf{x}_2$ de l'arête sont donc :

$$\begin{cases} \mathbf{u}_1^q = R_{p,q}\mathbf{u}_1^p + \mathbf{t}_{p,q}\,; \\ \mathbf{u}_2^q = R_{p,q}\mathbf{u}_2^p + \mathbf{t}_{p,q}. \end{cases} \tag{1.5.4}$$

où $R_{p,q}$ est la matrice de rotation d'angle $\frac{\pi}{2}s_{p,q}$ et $t_{p,q}$ est la translation à coordonnées entières associée à l'arête $e_{p,q}$. Notez que $\mathbf{t}_{q,p} = -t_{p,q}$ et $R_{q,p} = R_{p,q}^{-1}$. Lorsque $s_{p,q}$ vaut 1 ou 3, ces équations relient entre elles les variables du paramètre u et celles du paramètre v. Ces contraintes ont été formalisées par Ray *et al.* [92] et sont utilisées dans toutes les méthodes de paramétrisation préservant une grille (Tong *et al.* [110],Kälberer *et al.* [53] et Bommes *et al.* [10]). La Figure 1.18 montre l'effet de ces contraintes sur les fonctions de transitions sur le recollement de l'image de la grille sur la surface.

Calcul de la paramétrisation

Le calcul de la paramétrisation consiste à déterminer l'ensemble des inconnues, à savoir :

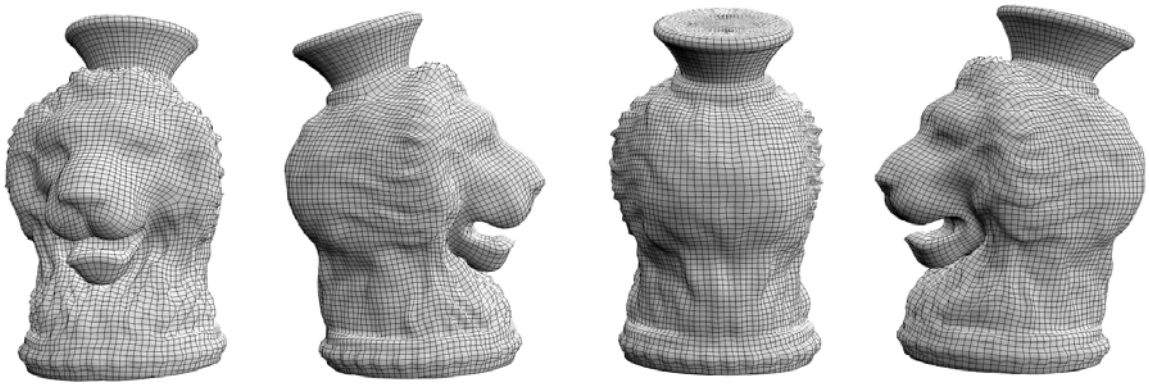


FIGURE 1.19 – Remaillage quadrangulaire d'une surface en utilisant une paramétrisation préservant une grille. Ici, le champ de directions a été obtenu par la méthode de Ray *et al.* [94], puis la paramétrisation calculée selon Bommes *et al.* [10].

- les coordonnées paramétriques $\mathbf{u}_i^p$ pour tout sommet $\mathbf{x}_i$ de chaque face T_p ;
- les translations $\mathbf{t}_{p,q}$ de chaque transition.

Les contraintes à respecter définies par les Équations 1.5.2, 1.5.3 et 1.5.4 ne peuvent en général pas être toutes satisfaites. Le calcul de la paramétrisation se fait donc en cherchant à respecter au mieux toutes les contraintes définies dans l'Équation 1.5.2 au sens des moindres carrés, ce qui ramène le problème à la résolution d'un système linéaire. Les contraintes de l'Équation 1.5.3 et de l'Équation 1.5.4 sont imposées par élimination de Gauß dans ce système linéaire. La résolution du système linéaire qui en découle permet d'obtenir des valeurs réelles pour toutes les variables. Cette solution n'est en général pas satisfaisante car un certain nombre de variables doivent en réalité être à valeur entière :

- les coordonnées paramétriques $\mathbf{u}_i^p$ des singularités (voir Section 1.5.3, page 43) ;
- les translation $\mathbf{t}_{p,q}$ des transitions.

L'approche de Kälberer *et al.* [53] consiste à résoudre le problème continu, puis à arrondir toutes les variables entières à l'entier le plus proche, alors que Bommes *et al.* [10] mettent en place un solveur mixte à variables entières et réelles, qui arrondit itérativement chaque variable entière, recalculant les autres variables encore réelles à chaque itération. Un remaillage quadrangulaire obtenu par une paramétrisation préservant une grille calculée par la méthode de Bommes *et al.* [10] est illustrée en Figure 1.19. Pour le remaillage, les recouvrement globaux ne sont pas gênant. L'absence de retournements locaux n'est pas contre pas garantie. Bommes *et al.* [10] rajoutent une phase de rigidification locale du maillage pour les éliminer lorsqu'ils apparaissent.

1.6 Conclusion

Dans ce chapitre, nous avons abordé les notions de base sur la paramétrisation de maillages, et détaillé plus particulièrement les paramétrisations préservant une grille. Ces paramétrisation permettent d'échantillonner un maillage par une grille, et ont à l'origine été pensées pour le remaillage. Nous allons désormais dans le Chapitre 2 aborder une autre application de ces paramétrisations pour le placage de textures. La structure de grille nous permet de faire disparaître l'artéfact des coutures inhérent à cette technique, et déjà évoqué en introduction.

Chapitre 2

Textures sans coutures

2.1 Motivations

Le placage de texture est l'une des applications majeures de la paramétrisation de surfaces dans le domaine de la synthèse d'image. Cette technique très classique, introduite par Catmull [15] consiste à déplier à plat un maillage pour le mettre en correspondance avec une image et y appliquer des couleurs. L'algorithme permettant, étant donné une surface et l'ensemble des coordonnées paramétriques de ses sommets, de calculer la couleur associée à tout point de la surface est implanté de façon matérielle dans toutes les cartes graphiques, pour le rendre le plus rapide possible. Un exemple de placage de texture sur un personnage de jeu vidéo est donné sur la Figure 2.1.

Si à l'origine le placage de texture a été conçu pour appliquer des couleurs sur une surface, son utilisation s'est désormais étendue à d'autres attributs que la couleur, comme dans le cas de la perturbation de normale introduite par Blinn [8]. L'utilisation des coordonnées paramétriques pour récupérer des attributs de la surface en mémoire est souvent une composante majeure des méthodes d'ombrage de surfaces. La rapidité de l'accès à la mémoire texture est alors critique pour les performances du rendu.

Le principal artéfact dû à l'utilisation du placage de texture correspond aux coutures. Comme vu dans le Chapitre 1, dans le cas général, la paramétrisation d'une surface requiert de la segmenter en un ensemble de régions homéomorphes à des disques. Les coutures apparaissent là où les différentes régions se recollent, sous forme de discontinuités au niveau de l'attribut plaqué sur la surface. Ces discontinuités sont illustrées sur la Figure 2.1.

2.2 Contributions

Ce chapitre introduit une méthode pour le placage de textures sans coutures. Ces travaux, réalisés en collaboration avec Nicolas Ray, Sylvain Lefebre et Bruno Lévy ont été publiés dans [93]. Cette méthode se fonde sur la génération d'un atlas de texture en utilisant une paramétrisation préservant une grille (voir Chapitre 1). L'avantage de cette méthode par rapport à l'état de l'art est tout d'abord qu'elle ne requiert pas l'introduction de code spécifique pour l'accès à la texture dans la chaîne de rendu de la carte graphique. Il est donc possible de récupérer les attributs à plaquer sur la surface par un accès générique à la texture, ce qui rend notre approche très rapide. De plus l'utilisation d'un atlas de texture rend cette méthode rétro-compatible avec tous les moteurs de rendu utilisant le placage de texture. La contrepartie est que l'atlas de texture utilisé pour le placage doit

(a) texture (b) objet texturé (c) couture

FIGURE 2.1 – Exemple de placage de texture. Les couleurs sont stockées dans l'image (a), et mises en correspondance avec l'objet 3D (b) par un atlas de texture. Des coutures apparaissent au niveau des transitions de la paramétrisation (c). Ce modèle est fourni gracieusement par Psionic3d

être généré automatiquement, par l'utilisation d'une paramétrisation préservant la grille. Ces paramétrisations introduisent en général plus de distorsion qu'une paramétrisation classique. De plus, il n'existe actuellement pas de méthode permettant de modifier une paramétrisation donnée pour lui faire préserver la grille. Il n'est donc pas possible de modifier un atlas de texture existant pour lui permettre de plaquer des textures sans couture visible.

Nos contributions sont les suivantes :

- Une méthode pour construire un atlas permettant le placage de texture à partir d'une paramétrisation préservant une grille. Cette construction utilise un procédé pour déterminer les ensembles de texels d'une texture devant avoir la même valeur pour qu'aucune couture ne soit apparente lors du placage en utilisant notre atlas. Le placage peut être réalisé en utilisant la plupart des modes classiques d'accès à la texture, à savoir les filtrages constant, bilinéaire, biquadratique, et tous les degrés supérieurs. Ces modes de filtrages sont détaillés en Section 2.3.3. Notre approche est décrite en Section 2.5.
- Une modification de l'atlas pour permettre de choisir localement le niveau de détail de la texture. Le nouvel atlas permet donc d'augmenter localement la résolution de la texture sur le modèle, et donc d'économiser l'espace pour les zones du maillage qui en ont réellement besoin. Cette méthode est présentée en Section 2.6.
- Un procédé pour gérer le mipmapping, qui est un mode d'accès à la texture particulier, largement utilisé, et permettant de diminuer l'effet de crénelage résultant d'une résolution trop importante de la texture par rapport à la résolution de l'image rendue. Une rapide explication de ce mode d'accès, est donnée en Section 2.3.3. Notre façon de l'aborder est décrite en Section 2.7.

2.3 Fondements

2.3.1 Notations

Nous rappelons tout d'abord les notations introduites dans le Chapitre 1 pour décrire un maillage et une paramétrisation de ce maillage.

Surface manipulée

$\mathcal{S}$	surface ajustée, décrite par un maillage
$\mathbf{X}$	ensemble des sommets de $\mathcal{S}$
$\mathbf{x}_i$	sommet de $\mathcal{S}$
$\mathbf{x}$	point quelconque de $\mathcal{S}$
$\mathbf{T}$	ensemble des faces de $\mathcal{S}$
T_p	une face de $\mathcal{S}$
$\mathbf{E}$	ensemble des arêtes de $\mathcal{S}$
$e_{p,q}$	arête adjacentes aux faces T_p et T_q

Atlas de texture

$\mathbf{E}_\tau$	ensemble des transitions de $\mathcal{S}$
$\tau_{p,q}$	fonction de transition associée à l'arête $e_{p,q}$
$\mathbf{u}$	point quelconque du domaine paramétrique
$\mathbf{u}_i^p$	coordonnées paramétriques du sommet $\mathbf{x}_i$ pour la face T_p
C_p	face paramétrique correspondant à T_p

Nous utiliserons également un certain nombre de symboles pour décrire les textures, et l'accès à ces textures. Ces notions seront introduites progressivement dans ce chapitre.

Texels et filtrage

$\mathbb{T}$	ensemble des texels
t	texel
t^p	texel utile pour la face paramétrique C_p
$[t]$	zone d'influence d'un texel t
$\mathbb{T}_\mathbf{u}$	ensemble des texels utiles en un point paramétrique $\mathbf{u}$
$\mathbb{T}_p$	ensemble des texels utiles pour une face paramétrique C_p
$h(t)$	valeur associée au texel t
$h(\mathbf{u})$	valeur obtenue par filtrage au point $\mathbf{u}$ à partir des valeurs des texels
$\mathbf{p}(t)$	position d'un texel dans le domaine paramétrique
$c(t)$	classe d'équivalence d'un texel t

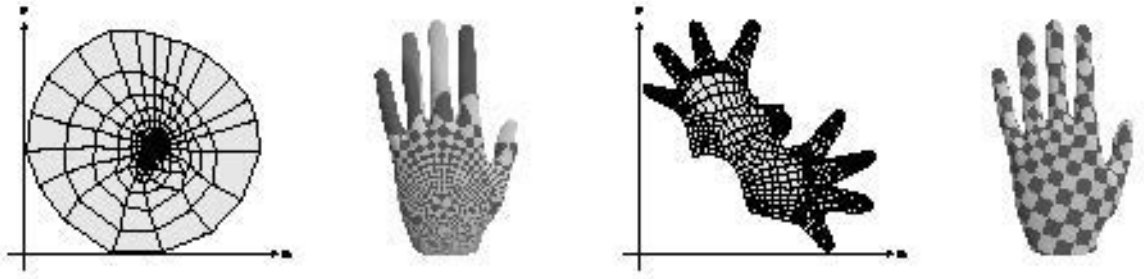
Résolution adaptative

ρ_p	résolution d'une face T_p dans l'atlas de texture
$\rho_\star$	résolution courante considérée
$\mathbb{T}_p^q$	texels utiles pour C_p à la résolution ρ_q
$t^{p,q}$	texel utile pour la face paramétrique C_p à la résolution ρ_q

2.3.2 Atlas de texture

Les paramétrisations utilisées pour le placage de texture correspondent parfaitement aux paramétrisations de surfaces maillées définies en Chapitre 1. Étant donné un maillage $\mathcal{S}$, constitué de sommets $\mathbf{X} = \{\mathbf{x}_i\}_i$ et de triangles $\mathbf{T} = \{T_p\}_p$, une paramétrisation est représentée par un ensemble de coordonnées paramétriques $\{\mathbf{u}_i^p\}_{i,p}$ associées à chaque sommet de chaque face du maillage. À chaque face T_p de sommets $\mathbf{x}_1$, $\mathbf{x}_2$ et $\mathbf{x}_3$ correspond une face paramétrique C_p de sommets $\mathbf{u}_1^p$, $\mathbf{u}_2^p$ et $\mathbf{u}_3^p$. Un sous-ensemble $\mathbf{E}_\tau$ des arêtes de la surfaces (les transitions) découpe la surface en un ensemble de régions homéomorphes à des disques.

Le placage de texture utilise les coordonnées paramétriques pour définir la position dans l'image correspondant à la valeur associée à un point de la surface. La même image étant



(a) Paramétrisation sans transitions (b) Paramétrisation avec transitions

FIGURE 2.2 – Réduction de la distorsion de la paramétrisation par l'introduction de transitions. Le modèle est homéomorphes à un disque et peut donc être paramétré sans transitions (a). Une paramétrisation conforme du modèle sans transitions aura une forte distorsion d'échelle, ici illustrée par la taille des carrés d'une grille régulière de l'espace paramétrique plaquée sur la surface. L'introduction de transitions permet de paramétrer la surface en réduisant la distorsion (b).

utilisée pour toute la surface, il est nécessaire de faire en sorte que deux régions différentes de l'image soient associées à des portions différentes de l'image. La phase d'*agencement* consiste à modifier les paramétrisations des différentes régions – les cartes – pour faire en sorte qu'elles ne se recouvrent pas. L'ensemble des coordonnées paramétriques ainsi obtenues correspondent à l'*atlas de texture*. Les travaux concernant la segmentation et l'agencement sont résumés par Hormann *et al.* [51]. Nous nous contenterons ici d'en esquisser les principes.

Segmentation

La segmentation est une étape souvent réalisée à la main, car le positionnement des transitions peut dépendre de contraintes sémantiques, propres à l'objet à texturer. Typiquement il peut s'agir de séparer différentes portions d'un objet n'ayant pas le même matériau, comme les vêtements du personnage présenté en Figure 2.1. En dehors de ces contraintes liées à l'utilisateur, les transitions ont également plusieurs utilités :

Rendre la surface homéomorphe à un disque : comme vu en Section 1.4.1, la paramétrisation étant réalisée sur une portion du plan, il est nécessaire que l'ensemble des régions définies sur la surface soient homéomorphes à des disques. Selon la topologie de la surface, un certain nombre de transitions seront donc requises.

Minimisation de la distorsion : les transitions, en plus de rendre la surface homéomorphe à un disque, permettent également d'agir sur la distorsion de la paramétrisation. La Figure 2.2 montre qu'il est nécessaire d'introduire des transitions pour des surfaces complexes, pour améliorer la qualité de la paramétrisation. En terme de placage de texture, la distorsion est directement liée à la résolution de la texture sur la surface.

Optimisation de l'agencement : le rôle de l'agencement est de limiter l'espace texture inutilisé, pour limiter les dimensions de la texture. Les transitions peuvent donc être utilisées pour faciliter l'agencement, en faisant en sorte que les cartes aient des formes faciles à ranger.

L'Algorithme 1.1 définit en Section 1.5.3 fournit un ensemble de coutures permettant de rendre toute surface, quelle qu'elle, soit homéomorphe à un disque. Le problème général

de trouver le plus petit ensemble de transitions possible, en terme de nombre d'arêtes ou de leurs longueurs, est un problème NP-difficile comme l'ont montré Erickson et Har-Peled [37].

Pour pouvoir paramétrer une surface sans distorsion, il est nécessaire qu'elle soit développable, c'est à dire qu'en tout point, l'une des courbures principales est nulle. Il existe donc deux approches principales fondées sur cette idée. L'une consiste à regrouper les faces de la surface pour former des régions les plus planes possibles, l'autre détecte les zones de forte courbure gaussienne, et en déduit un ensemble de transitions passant par ces zones.

Le critère généralement utilisé pour limiter l'espace perdu lors de la phase d'agencement consiste à limiter la longueur du bord des différentes régions. Les méthodes de segmentation fondées sur le regroupement de faces du maillage pour former des régions plane sont ici mieux adaptées que les méthodes cherchant à tracer l'ensemble des transitions entre les régions de forte courbure gaussienne, qui ont tendance à générer des formes de cartes complexes. Pour le cas des atlas que nous produirons dans ce chapitre, c'est ce point qui nous intéressera, car la distorsion sera de toute façon définie par la paramétrisation préservant une grille utilisée pour initialiser notre méthode.

Agencement

Les images utilisées comme textures sont stockées sous forme de tableaux bidimensionnel, donc rectangulaires, voire carrés selon les spécifications de la carte graphique. La plupart des méthodes d'agencement considèrent le cas carré pour plus de généricité. Calculer un agencement d'un ensemble de cartes consiste à modifier les paramétrisations des cartes pour faire en sorte qu'elles ne se recouvrent pas mutuellement dans l'espace paramétrique. Une fois un tel agencement calculé, le carré paramétrique correspondant à la texture devra être un carré englobant toutes les cartes. La qualité d'un agencement est évaluée en calculant le taux de remplissage de la texture, à savoir le rapport entre la somme des aires des cartes, et l'aire totale du carré de la texture.

Le problème du calcul d'un agencement optimal est également un problème NP-difficile. Pour le cas de l'agencement d'un atlas de texture, la plupart des méthodes sont fondées sur la méthode de l'agencement « Tetris » élaborée par Lévy *et al.* [63], qui consiste à introduire les cartes une par une verticalement ou horizontalement, en faisant en sorte à chaque insertion de minimiser la croissance de la boîte englobante. C'est cette méthode que nous utiliserons dans ce chapitre.

2.3.3 Filtrage de la texture

Le filtrage de la texture correspond à la façon dont les données enregistrées dans la texture sont associées à une positions paramétrique $\mathbf{u}$ donnée. Les cartes graphiques proposent en général deux modes de filtrage, le mode *au plus proche* et le mode *bilinéaire*. Ces deux modes peuvent être combinés avec le mipmapping qui permet d'éviter les artéfacts liés à un échantillonnage trop fin de la texture par rapport à l'affichage. Nous détailleront ici les différentes notions et notations associées à l'accès à la texture.

Texels

Le mot *texel* (« texture element ») est un dérivé du mot *pixel* signifiant « picture element ». Historiquement, les textures sont apparues pour plaquer de la couleur sur un objet 3D. Une texture correspond donc la plupart du temps à une image. En terme de structure de données, une image est un tableau bidimensionnel contenant dans chaque case une couleur. Chacune de ces cases est un pixel. Les textures sont désormais utilisées de manières plus générale pour associer des données quelconques à une surface, et donc d'autres données que de la couleur. C'est la raison pour laquelle le terme *texel* « texture element » est employé pour désigner une case du tableau d'une texture, ne contenant donc pas nécessairement une couleur. Nous utiliserons la terminologie suivante :

un texel noté t est une case du tableau bidimensionnel $\mathbb{T}$ représentant une texture ;

la valeur notée $h(t)$ d'un texel est la donnée qu'il contient.

Filtres

Le filtre correspond à la fonction associant à un point paramétrique $\mathbf{u}$ une valeur. Cette valeur est calculée en fonction des valeurs des texels de la texture. Un filtre h associe donc une valeur à tout point de l'espace paramétrique :

$$\begin{array}{rccc} h: & \mathbb{R}^2 & \to & \mathbb{R}^k \\ & \mathbf{u} & \mapsto & h(\mathbf{u}) \end{array}$$

Dans le cadre de nos travaux, nous avons étudié les modes de filtrage *au plus proche*, *bilinéaire*, *biquadratique* et *bicubique*, qui sont les plus utilisés dans le domaine du placage de texture. Ces modes de filtrages sont construits à base de fonctions B-splines. Nous décrirons ces modes de filtrage dans les prochaines sections, et invitons le lecteur à se référer à la synthèse de Prautzsch *et al.* [89] pour plus de détails sur les fonctions B-splines. Dans ces quatre modes de filtrage, la valeur associée au paramètre $\mathbf{u}$ est une combinaison linéaire des valeurs des texels de la texture :

$$h(\mathbf{u}) = \sum_{t \in \mathbb{T}} w_t(\mathbf{u}) h(t). \tag{2.3.1}$$

où la fonction w_t est une fonction B-spline, définissant l'influence de la valeur du texel t. Nous utiliserons la terminologie suivante :

Les texels utiles d'un point paramétrique $\mathbf{u}$ sont les texels dont la fonction d'influence w_t est non nulle en $\mathbf{u}$. Nous les noterons $\mathbb{T}_u$. Pour les modes de filtrage étudiés, cet ensemble est toujours fini et contient toujours le même nombre de texels.

Une cellule de filtrage est une région paramétrique où l'ensemble des texels utiles est constant. Dans le cadre des modes de filtrage que nous avons choisis, ces cellules de filtrage forment une grille de l'espace paramétrique.

La zone d'influence d'un texel t correspond à la région paramétrique où w_t est non nulle. Il s'agit donc des zones dans lesquelles la couleur donnée par le filtrage dépendra de la valeur du texel. La zone d'influence est l'union des cellules de filtrages pour lesquelles le texel est utile. Nous noterons $[t]$ la zone d'influence d'un texel t.

Ces notions sont illustrées en Figures 2.3, 2.4, 2.5 et 2.6 pour les modes de filtrage traités. Dans la suite, nous ferons également toujours en sorte de positionner le repère paramétrique pour faire en sorte que la grille formée par les cellules de filtrage corresponde à la grille $\mathbb{Z}^2$. Cette convention sera prise en choisissant pour chaque texel une *position* dans l'espace paramétrique, que nous noterons $\mathbf{p}(t)$.

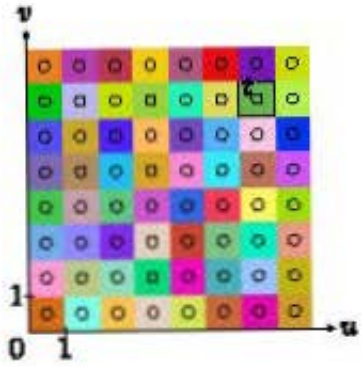


FIGURE 2.3 – Un exemple de filtrage au plus proche. Les cellules de filtrage correspondent aux cellules de la grille $\mathbb{Z}^2$, et la zone d'influence du texel t ne contient qu'une cellule de filtrage.

Filtrage au plus proche (constant)

Ce mode est le plus simple, mais fournit un champ de valeurs discontinu. Pour ce mode, les positions des texels sont définies pour placer les texels sur la grille $\mathbb{Z}^2 + (\frac{1}{2},\frac{1}{2})^t$. Étant donné un point $\mathbf{u}$ de l'espace paramétrique, la valeur associée est celle du texel t_0 dont la position est la plus proche :

$$h(\mathbf{u}) = h(t_0) \tag{2.3.2}$$

Ici, les cellules de filtrages correspondent bien aux cellules de la grille $\mathbb{Z}^2$, et dans chacune de ces cellules, la valeur est uniformément celle du texel positionné en son centre. Inversement, la zone d'influence d'un texel est l'unique cellule de filtrage pour laquelle il est utile, et donc correspond à la cellule de la grille $\mathbb{Z}^2$ centrée sur la position du texel. Ces éléments sont illustrés en Figure 2.3.

Ce mode de filtrage fait partie de la norme OpenGL et peut être utilisé comme mode de filtrage en utilisant le nom `GL_NEAREST`. Il est donc supporté nativement par toutes les cartes graphiques actuelles.

Filtrage Bilinéaire

Ce mode de filtrage est le plus utilisé. Il permet d'obtenir un champ de valeurs continu. Dans ce mode, les texels sont positionnés sur la grille $\mathbb{Z}^2$. La valeur du filtrage pour un point $\mathbf{u} = (u,v)^t$ est calculée en utilisant les valeurs des quatre texels dont les positions sont aux coins de la cellule de $\mathbb{Z}^2$ contenant $\mathbf{u}$. En posant :

$$\begin{cases} u_0 = \lfloor u \rfloor \\ v_0 = \lfloor v \rfloor \end{cases} \quad \text{et} \quad \begin{cases} \alpha = u - u_0 \\ \beta = v - v_0 \end{cases} \tag{2.3.3}$$

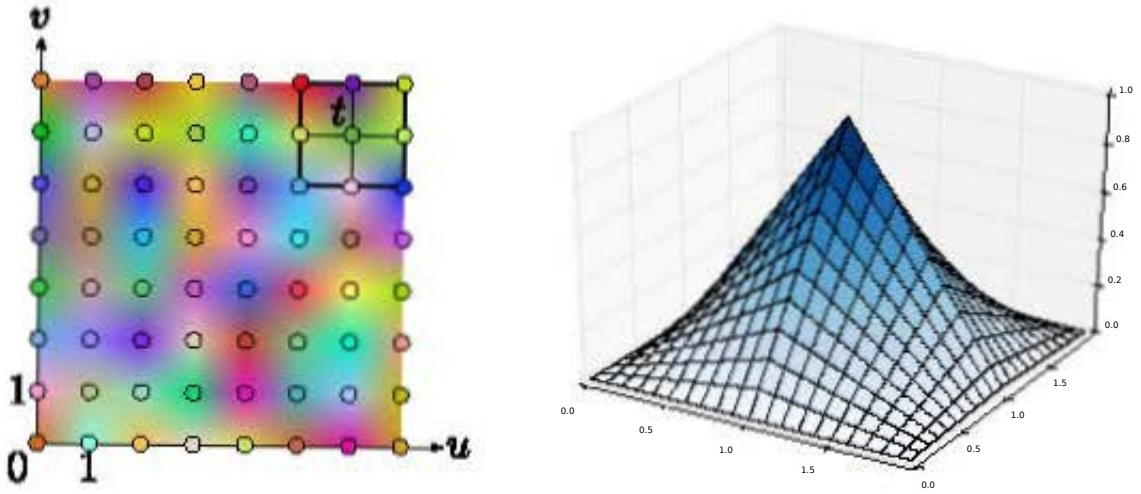
les positions des texels utiles $t_{i,j}$ avec $(i,j) \in \{0,1\}^2$ sont :

$$\mathbf{p}(t_{i,j}) = (u_0 + i, v_0 + j)^t$$

et leur influence est donnée par :

$$w_{t_{i,j}}(\mathbf{u}) = \alpha_i \beta_j \quad \text{où} \quad \begin{cases} \alpha_0 = (1-\alpha) \\ \alpha_1 = \alpha \end{cases} \quad \text{et} \quad \begin{cases} \beta_0 = (1-\beta) \\ \beta_1 = \beta \end{cases}$$

Le filtrage est enfin obtenu à partir des valeurs de ces texels en utilisant l'Équation 2.3.1.



(a) filtrage (b) influence d'un texel

FIGURE 2.4 – Un exemple de filtrage bilinéaire est donné en (a). Ici, la zone d'influence d'un texel est composée de quatre cellules de filtrage. La fonction d'influence d'un texel est illustrée en (b), où les courbes en gras correspondent aux limites des cellules de filtrage.

Le résultat de ce mode de filtrage est illustré sur la Figure 2.4. Comme le filtrage au plus proche, ce mode de filtrage fait partie de la norme OpenGL ; il est activé en utilisant le nom `GL_LINEAR`. Attention cependant, le repère que nous définissons pour la texture ne correspond pas au repère classique défini par OpenGL, qui positionne toujours les texels comme dans le cas du filtrage au plus proche.

Filtrage Biquadratique

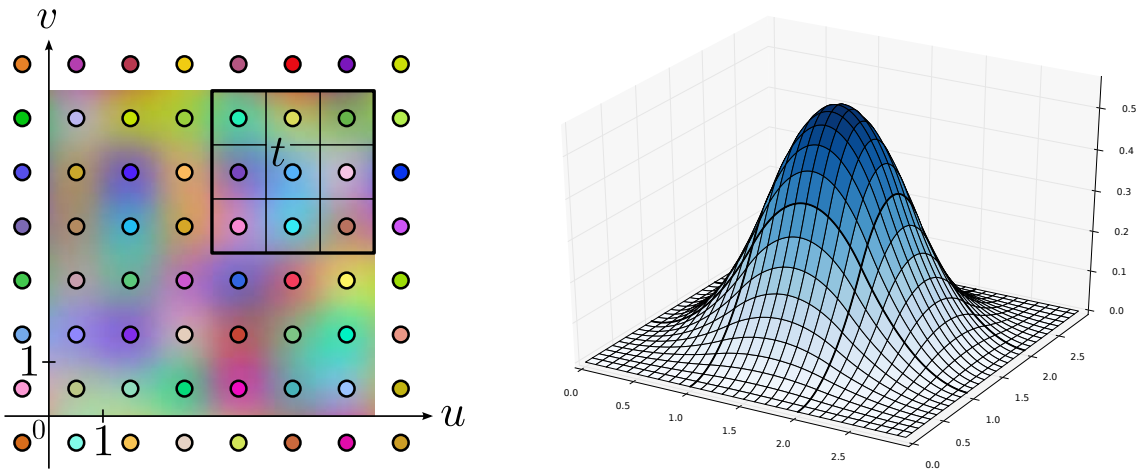
Ce mode de filtrage est moins utilisé que les précédents, car il n'est pas supporté nativement par la carte graphique ; il ne peut être obtenu qu'en programmant la carte graphique, ce qui ralentit les accès à la texture. C'est aussi le cas du filtrage bicubique, mais ces deux filtrages ayant sensiblement le même coût en terme d'accès à la texture, le filtrage bicubique est généralement préféré car il offre une meilleure continuité du champ de valeurs. Pour ce filtrage, les texels sont positionnés comme dans le cas du filtrage au plus proche, sur la grille $\mathbb{Z}^2 + (\frac{1}{2},\frac{1}{2})^t$. En chaque point $\mathbf{u} = (u,v)^t$, le filtrage se fait à partir de neuf texels. En utilisant les notations de l'Équation 2.3.3, en un point $\mathbf{u}$, les neuf texels utiles $\{t_{i,j}\}$ avec $(i,j) \in \{0,1,2\}^2$ sont aux positions :

$$\mathbf{p}(t_{i,j}) = \left(u_0 + i - \tfrac{1}{2}, v_0 + j - \tfrac{1}{2}\right)^t$$

et leur influence est donnée par :

$$w_{t_{i,j}}(\mathbf{u}) = \alpha_i \beta_j \quad \text{avec} \quad \begin{cases} \alpha_0 = \frac{1}{2}(1-\alpha)^2 \\ \alpha_1 = -\alpha^2 + \alpha + \frac{1}{2} \\ \alpha_2 = \frac{1}{2}\alpha^2 \end{cases} \quad \text{et} \quad \begin{cases} \beta_0 = \frac{1}{2}(\beta-1)^2 \\ \beta_1 = -\beta^2 + \beta + \frac{1}{2} \\ \beta_2 = \frac{1}{2}\beta^2 \end{cases}$$

Le résultat de ce filtrage est illustré en Figure 2.5.



(a) filtrage (b) influence d'un texel

FIGURE 2.5 – Un exemple de filtrage biquadratique est donné en (a). Ici, la zone d'influence d'un texel est composée de neuf cellules de filtrage. La fonction d'influence d'un texel est illustrée en (b), où les courbes en gras correspondent aux limites des cellules de filtrage.

Filtrage Bicubique

De même que le filtrage biquadratique, le filtrage bicubique n'est pas supporté nativement par l'architectures des cartes graphiques. Il est donc nécessaire de programmer l'accès à la texture pour l'obtenir. Il est possible de programmer ce filtrage en n'utilisant que quatre accès texture en mode bilinéaire, en utilisant l'implémentation proposée par Pharr et Fernando [82, chapitre 20]. Pour ce filtrage, les texels sont positionnés comme dans le cas bilinéaire, sur la grille $\mathbb{Z}^2$. Pour un point $\mathbf{u}$, en utilisant les notations de l'Équation 2.3.3, les seize texels utiles $\{t_{i,j}\}$ où $(i,j) \in \{0,1,2,3\}^2$ sont aux positions :

$$\mathbf{p}(t_{i,j}) = (u_0 + i - 1, v_0 + j - 1)^t$$

et leur influence est donnée par :

$$w_{t_{i,j}}(\mathbf{u}) = \alpha_i \beta_j$$

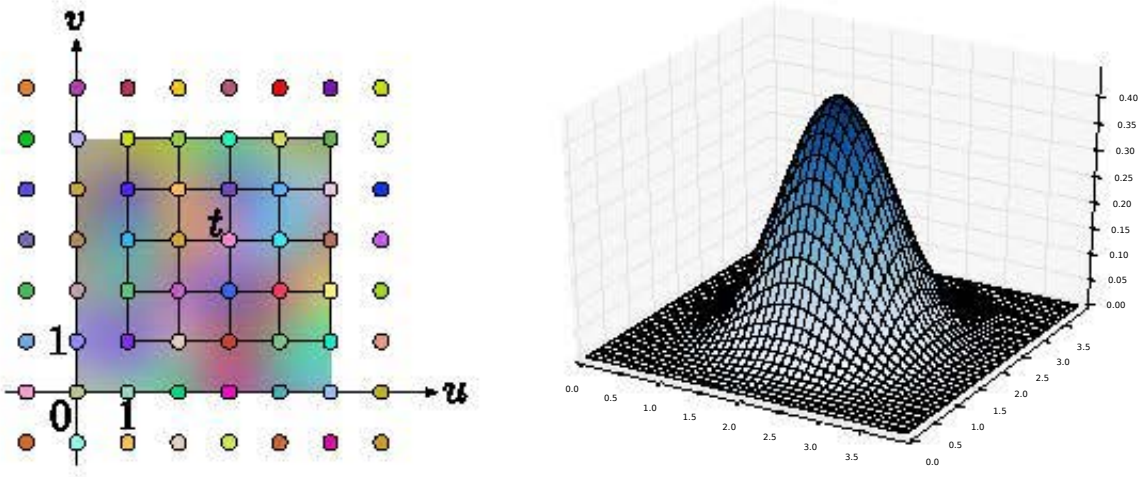
où

$$\begin{cases} \alpha_0 = \frac{1}{6}(1-\alpha)^3 \\ \alpha_1 = \frac{1}{6}(3\alpha^3 - 6\alpha^2 + 4) \\ \alpha_2 = \frac{1}{6}(-3\alpha^3 + 3\alpha^2 + 3\alpha + 1) \\ \alpha_3 = \frac{1}{6}\alpha^3 \end{cases} \quad \text{et} \quad \begin{cases} \beta_0 = \frac{1}{6}(1-\beta)^3 \\ \beta_1 = \frac{1}{6}(3\beta^3 - 6\beta^2 + 4) \\ \beta_2 = \frac{1}{6}(-3\beta^3 + 3\beta^2 + 3\beta + 1) \\ \beta_3 = \frac{1}{6}\beta^3 \end{cases}$$

Ce mode de filtrage est illustré sur la Figure 2.6.

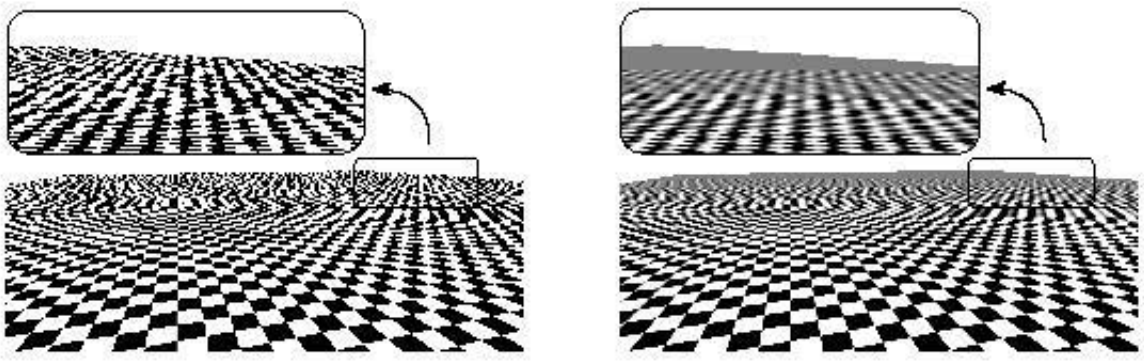
Mipmapping et filtrage anisotrope

Le *mipmapping* est une technique de filtrage introduite par Williams [117] qui vise à limiter les effets de crénelage qui apparaissent lorsque la texture a une résolution plus élevée que l'affichage. Lorsque c'est le cas, deux pixels voisins dans l'image affichée peuvent correspondre à des texels de la texture éloignés, ce qui crée des artéfacts à l'affichage. Pour résoudre ce problème, le mipmapping consiste à précalculer des versions réduites de la texture initiale, et à choisir la résolution de la texture utilisée en fonction de la résolution de



(a) filtrage (b) influence d'un texel

FIGURE 2.6 – Un exemple de filtrage bicubique est donné en (a). Ici, la zone d'influence d'un texel est composée de seize cellules de filtrage. La fonction d'influence d'un texel est illustrée en (b), où les courbes en gras correspondent aux limites des cellules de filtrage.



(a) Sans mipmapping (b) Avec mipmapping

FIGURE 2.7 – Effet du mipmapping pour diminuer l'effet de crénelage lorsque la texture a une résolution plus forte que l'affichage.

l'affichage. Cette technique permet d'améliorer la qualité de l'affichage sans trop dégrader les performances. En mémoire, l'espace supplémentaire requis correspond au tiers de l'espace occupé par la texture seule. La différence entre l'affichage avec et sans mipmapping est illustré sur la Figure 2.7.

La *pyramide* de mipmaps est une suite de textures de résolution de plus en plus grossière. Cette pyramide est initialisée avec la texture fournie par l'utilisateur, et chaque niveau a une résolution deux fois plus faible que le niveau précédent. Le niveau le plus grossier est une texture n'ayant qu'un seul texel. Les cartes graphiques possèdent des mécanismes automatiques pour calculer ces niveaux itérativement à partir de la texture initiale, simplement en remplaçant les carrés de quatre texels par un seul texel ayant pour valeur la moyenne des valeurs des texels qu'il remplace. Ce procédé peut cependant altérer certains détails de la texture qu'il serait souhaitable de préserver au mieux. L'utilisateur peut donc également fournir lui-même les différents niveaux de la pyramide.

Pour afficher un objet texturé, l'objet est tout d'abord morcelé en petites portions, chacune correspondant à un pixel de l'écran. Pour chacune de ces portions peut par contre couvrir une zone de l'atlas de texture contenant de nombreux texels. En fonction de cette différence de résolution, deux méthodes peuvent être utilisées pour récupérer une valeur dans la texture. Le mode au plus proche choisit le niveau de la pyramide de mipmaps dont la résolution est la plus proche de la résolution d'affichage, et récupère la valeur dans ce niveau en utilisant le filtrage choisi. Le mode linéaire choisit les deux niveaux ayant une résolution respectivement immédiatement supérieure et immédiatement inférieure à la résolution d'affichage, récupère une valeur dans ces deux niveaux en utilisant le filtrage normal, puis interpole linéairement une valeur à partir de ces deux valeurs pour la résolution d'affichage. Ce second mode évite de voir apparaître des discontinuités entre deux fragments voisins, utilisant des niveaux différents de la pyramide.

Il existe également un autre mode de filtrage qui améliore encore la qualité du rendu, lorsque la résolution de la texture à afficher est différente verticalement et horizontalement : le *filtrage anisotrope*. Pour réaliser ce filtrage, la pyramide de mipmaps est étendue en intégrant toutes les paires de résolutions possibles verticalement et horizontalement. L'espace de stockage supplémentaire requis correspond au plus à trois fois l'espace occupé par la texture initiale.

2.4 État de l'art

L'idée du placage de texture sans couture apparente n'est pas neuve, et de nombreuses méthodes ont déjà été élaborées pour s'attaquer à ce problème. L'une des méthodes les plus récentes est celle élaborée par Burley et Lacewell [11], sur laquelle nous reviendrons, qui a entre autres été intégrée dans le moteur de rendu commercial Renderman. Cette méthode n'a pas été initialement développée pour le rendu en temps réel, mais la puissance des cartes graphiques actuelles permet désormais de l'envisager. Dans ces travaux les auteurs utilisent les critères suivants pour évaluer les méthodes existantes :

Qualité : la méthode doit supporter les standards de l'industrie audiovisuelle, notamment le mipmapping et le filtrage anisotrope, et permettre le placage de textures sans couture visible.

Généricité : il doit être possible de traiter tout type de maillage, quelle que soit la complexité de la géométrie ou de la topologie du maillage.

Efficacité : le coût en stockage, accès à la mémoire, et la quantité de calculs doivent être les plus limités possibles.

Mise en place : les prétraitements à appliquer sur le maillage à texturer doivent être les plus simples possibles. Il doit être envisageable de modifier la surface ou les textures facilement au niveau de la chaîne de production.

Nous tenterons d'évaluer notre méthode et les approches de la littérature objectivement par rapport à ces critères.

2.4.1 Placage sans atlas

Une première idée pour supprimer les coutures consiste simplement à se passer d'atlas de texture. Un grand nombre de méthodes définissent des procédés alternatifs pour plaquer des couleurs sur une surface sans avoir à paramétrer la surface, et donc sans transitions.

Méthodes volumiques

Les cartes graphiques supportent également l'utilisation de textures volumiques. Il s'agit de définir une texture plane, mais cubique, et d'utiliser les coordonnées 3D des points de la surface pour assigner à chaque point de l'objet une donnée dans le cube. La texture est donc un tableau tridimensionnel, et les cases de ce tableau sont des *voxels*. Cette approche est exempte de transitions, et ne fait donc pas apparaître de discontinuités sur la surface. En terme de qualité, le résultat est donc tout à fait satisfaisant, même si le mipmapping est difficile à généraliser. En terme de généricité, il est simplement nécessaire de faire en sorte qu'un même voxel ne soit pas utilisé par plusieurs portions non adjacentes de la surface à texturer, et donc la grille de voxels doit simplement être suffisamment fine. Au niveau de la mise en place, aucune paramétrisation n'est nécessaire, les coordonnées 3D de l'objet sont directement utilisées. Le problème réside au niveau de l'espace mémoire occupé, puisqu'un cube entier de voxels est stocké, alors que seule une minorité d'entre eux est nécessaire pour texturer la surface. L'édition d'une texture en 3D est également un procédé fastidieux ; la texture doit être modifiée si la surface est déformée, car l'adressage de la texture est fondé sur les coordonnées 3D de l'objet. L'accès reste cependant très rapide, car ce procédé est nativement supporté par la carte graphique.

Pour limiter l'espace mémoire nécessaire Benson et Davis [7] remplacent la texture volumique par un arbre octal. L'utilisation d'un tel arbre est une technique classique pour stocker des données volumiques creuses. La racine d'un tel arbre correspond au cube englobant tout l'objet. La subdivision d'un nœud consiste à découper le cube auquel il correspond en huit cubes par bissection. Récursivement, le cube englobant est subdivisé, et seuls les nœuds intersectant la surface sont subdivisés. La subdivision s'arrête quand la résolution désirée est atteinte. Les auteur discutent du stockage de cette structure dans une texture, accédée en reprogrammant l'accès à la texture. Le mipmapping se généralise en choisissant la profondeur utilisée dans l'arbre en fonction de la résolution d'affichage. Ce procédé reste cependant limité car la profondeur doit être au moins suffisante pour que deux portions différentes de la surface correspondant à un même nœud puissent avoir des valeurs différentes.

Une alternative également fondée sur les arbres octaux est proposée par Lefebvre et Dachsbacher [59]. L'idée de cette approche consiste à stocker l'attribut à plaquer sur l'objet dans des images correspondant aux faces des nœuds – cubiques – de l'arbre octal plutôt que dans les nœuds eux-mêmes. La face du nœud utilisée pour les portions de surface qu'il englobe est définie par rapport à la normale de la surface. La profondeur de l'arbre octal nécessaire est donc plus faible, ce qui améliore les performances d'accès. Pour éliminer les discontinuités apparaissant lorsque différentes faces du nœud sont utilisées pour deux points voisins de la surface, l'attribut stockant la face dans laquelle récupérer la valeur est associé aux sommets du maillage. Il est interpolé linéairement sur les triangles, ce qui permet un mélange continu des textures des différentes faces lors des transitions.

Maillages de texture

Tarini *et al.* [107] proposent d'utiliser un *polycube* plutôt qu'un arbre octal. Un polycube est construit en accolant un a un des cubes d'égale dimension pour former une sous-partie de la grille $\mathbb{Z}^3$. La méthode de Tarini *et al.* [107] consiste à mettre en correspondance la surface à texturer avec la surface d'un polycube fourni par l'utilisateur. Cette correspondance est calculée en assignant à chaque sommet du maillage à texturer des coordonnées

sur la surface du polycube par un procédé d'ajustement de surface similaire à celui défini en Chapitre 5. Ces coordonnées sont interpolées linéairement, ce qui assure que les valeurs stockées sur la surface du polycube sont plaquées continument sur la surface. Le fait que les faces du polycube soient carrées permet de définir aisément un champ de valeurs continu par interpolation bilinéaire, ce qui permet donc d'assurer que le champ de valeurs plaqué sur la surface n'a pas de discontinuités.

Le calcul automatique d'un polycube n'est pas un problème trivial, de même que la paramétrisation d'une surface sur un polycube. Lorsque la surface à texturer est déjà munie d'un maillage quadrangulaire, Burley et Lacewell [11] proposent de stocker la texture directement sur les faces du maillage sans passer par le polycube, en associant une texture à chaque face. Cette idée simplifie énormément la mise en place, car les maillages quadrangulaires sont très souvent déjà un standard de l'industrie audiovisuelle, et la qualité de l'échantillonnage de la texture est liée à la qualité des éléments quadrangulaires, qui fait déjà partie des contraintes de modélisation. Burley et Lacewell [11] généralisent aussi les résultats de Tarini *et al.* [107] pour permettre un niveau de résolution adaptatif selon les faces du maillage. Les auteurs abordent également le problème du filtrage bicubique. Ils permettent le placage sans couture visible lorsque les résolutions de deux faces voisines sont identiques et en l'absence de singularités (les sommets de valence autre que quatre, voir Section 1.5.3). Lorsque les résolutions des faces adjacentes sont différentes, ou au niveau des singularités, leur approche permet de réduire l'aspect visuel des coutures, sans toutefois parvenir à les dissimuler complètement.

Pour traiter les maillages quelconques Yuksel *et al.* [120] proposent de stocker les texels directement sur le maillage. Ils ne se restreignent par contre plus à un agencement des texels sur une grille carrée, mais utilisent au contraire une grille triangulaire, en définissant une méthode de filtrage linéaire pour obtenir un champ de valeurs continu à partir des valeurs des texels, et en généralisant le mipmapping à ce nouveau schéma d'accès. Pour les faces carrées, ils utilisent la disposition carrée usuelle.

2.4.2 Dissimulation des coutures

Il n'est pas nécessaire de se passer des atlas de texture pour pouvoir définir un champ de valeurs continu sur la surface. La technique naturelle utilisée pour dissimuler les coutures consiste à s'assurer que les valeurs des texels de part et d'autre se correspondent. Cette précaution ne suffit en général pas à supprimer complètement les discontinuités de valeur au niveau des coutures.

Positionnement des coutures

Une première idée pour diminuer l'impact des coutures consiste à faire en sorte qu'elles soient les moins visibles possible. C'est l'approche choisie par Sheffer et Hart [102], qui choisisse l'ensemble des transitions en fonction de trois critères. Tout d'abord, les transitions doivent rendre la surface homéomorphe à un disque. Ensuite, les transitions doivent passer par un certain nombre de points de la surface de forte courbure gaussienne, pour limiter la déformation de la paramétrisation. Enfin, la visibilité des arêtes de la surface est évaluée depuis un ensemble de points de vue régulièrement répartis sur une sphère – ou un hémisphère selon la configuration de la scène. À partir de ces échantillons, la probabilité de visibilité d'une arête est calculée ; les arêtes les moins visibles sont utilisées pour placer

les transitions. Cette méthode ne supprime donc pas les transitions, mais les rend moins visibles.

Traitement spécial au voisinage des transitions

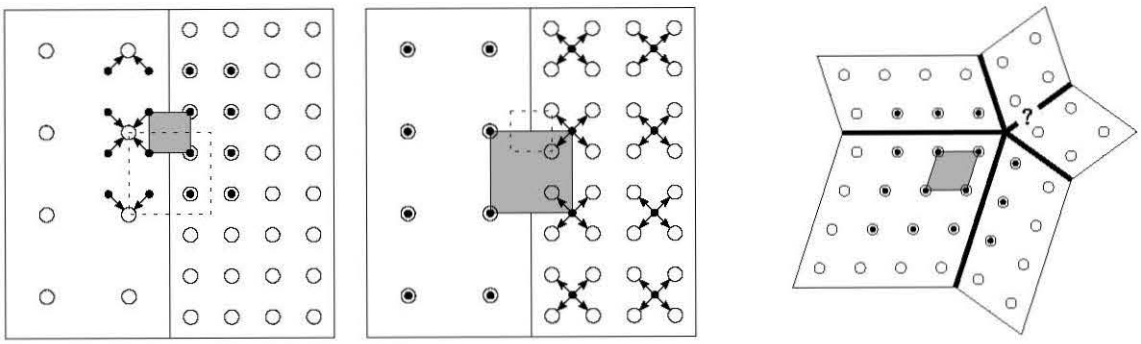
Une approche par Piponi et Borshukov [85] consiste à utiliser un atlas classique, en rajoutant une carte pour paramétrer le voisinage des transitions. La portion du maillage correspondant à cette carte est donc définie comme l'ensemble des faces adjacentes aux transitions, ou à distance deux. Les faces concernées ont donc deux jeux de coordonnées paramétriques. Lors de l'accès à la texture, les valeurs des deux sont mêlées. La fonction définissant le mélange est calculée sur la carte graphique lors du rendu, de telle sorte qu'au niveau de la transition, seule la carte de son voisinage soit influente, et qu'au niveau des bords de la carte couvrant les transitions, seule la carte usuelle soit visible. Ces contraintes assurent donc un champ de valeurs continu sur la surface.

À plus faible grain, González et Patow [45] réalisent une opération similaire à l'échelle des texels. Leur approche consiste à détecter les texels proches des transitions et à générer une triangulation pour les relier entre eux de part et d'autre de la transition. Cette triangulation est stockée dans une texture ; au moment du rendu, l'accès à la texture est reprogrammé pour que les valeurs soient interpolées sur les triangles générés au voisinage des transitions. Le gros avantage de cette méthode est qu'elle préserve à la fois la surface, l'atlas de texture et la texture fournie par l'utilisateur, pour qui l'utilisation est complètement transparente.

Génération d'atlas spécifiques

Plusieurs méthodes génèrent des atlas de texture garantissant directement le placage de textures sans couture apparente. Ces méthodes sont fondées sur la génération de cartes de formes rectangulaires, permettant d'assurer le recollement de part et d'autre des transitions. Pour générer l'ensemble des cartes, elle utilisent des méthodes itératives pour regrouper les faces, introduites par Eck *et al.* [32], consistant à définir un ensemble de triangles sources, puis à faire grandir les régions à partir de ces triangles de manière gloutonne, en ajoutant des triangles dans un ordre défini par une fonction de coût.

Carr et Hart [13] ont tout d'abord cherché à créer des cartes carrées pour créer une paramétrisation dynamique, adaptant sa résolution en fonction de la demande de l'utilisateur. Les cartes carrées étaient nécessaires pour réaliser l'agencement de l'atlas à l'aide d'un arbre quaternaire – la version bidimensionnelle des arbres octaux. La profondeur de l'arbre augmentant, les cellules carrées deviennent de plus en plus petites et la résolution diminue. Les portions du maillages nécessitant une forte résolution sont ainsi placées le plus haut possible dans l'arbre. Pour former les cartes carrées, le maillage est segmenté en répartissant ses triangles entre les différentes cartes. Un traitement spécial doit être réalisé pour limiter les triangles ayant deux de leurs côtés sur le bord, qui deviennent dégénérés lors de la paramétrisation. Carr *et al.* [14] étendent cette idée spécifiquement pour créer des atlas permettant de texturer sans couture visible. La segmentation du maillage est modifiée pour utiliser la norme L_∞ et encourager les régions carrées. Une heuristique est ensuite appliquée pour déterminer le nombre de texels le long des transitions séparant deux cartes. En alignant les texels de part et d'autre des transitions, le placage sans couture apparente devient possible.



(a) Saut de résolution (b) Singularité

FIGURE 2.8 – Difficultés pour réaliser un filtrage bicubique sans couture apparente. Lorsque les résolutions des textures de part et d'autre d'une transition sont différentes (a), les cellules de filtrage – en gris – n'utilisent pas le même ensemble de texels selon le côté de la transition. Au voisinage d'une singularité (b), l'ensemble des des texels utiles pour calculer la valeur d'un point de la cellule de filtrage est mal défini.

Une approche développée parallèlement par Purnomo *et al.* [90] construit également des cartes par agglomération de faces, mais commence par créer des régions globalement circulaire en minimisant le rapport périmètre sur aire des régions créées. Une fois les régions construites, elles sont subdivisées à partir du centre pour former des régions quadrangulaires, paramétrées sur des carrés. Différentes méthodes d'agencement sont ensuite étudiées, notamment pour permettre le placage de textures sans couture apparente, même dans le cas du mipmapping.

2.4.3 Bilan

Qualité

En termes de qualité, la plupart des méthodes proposées permettent de définir un champ de valeurs continu sur la surface, en utilisant un filtrage linéaire. Ce mode simplifie grandement le problème, car les cellules de filtrage ne dépendent que d'un voisinage restreint de texels. En plaçant des texels exactement sur les transitions, aucune cellule de filtrage ne se retrouve à cheval de part et d'autre des transitions, et il suffit de s'assurer que le champ de valeur au niveau des transitions est identique de chaque côté. Cette condition est facilement remplie par un jeu de contraintes simples sur les valeurs des texels placés sur les transitions.

Pour le filtrage d'ordre supérieur, aucune méthode actuelle ne parvient à proposer un champ de valeurs parfaitement continu, même si Burley et Lacewell [11] proposent des solutions pour limiter le problème. Notez qu'il est illusoire de chercher un champ de valeurs d'ordre C^1 et plus, tant que la paramétrisation utilisée reste linéaire par morceaux. La difficulté supplémentaire que ces modes de filtrage apportent est que l'ensemble des texels utiles pour une cellule de filtrage est plus étendu. Le jeu de contraintes classiques consistant à récupérer les texels nécessaires dans les cartes voisines permet à Burley et Lacewell [11] de proposer un champ continu avec le filtrage bicubique sauf dans deux situations : lorsque les résolutions de la texture de part et d'autre d'une transition ne sont pas identiques, et en présence d'une singularité comme le montre la Figure 2.8.

Notre approche en comparaison analyse précisément les contraintes nécessaires à l'obtention d'un filtrage continu, même bicubique, à la fois dans le cas de résolutions différentes de part et d'autre des transitions, et dans le cas des singularités. Elle permet de gérer le mipmapping aussi bien que les méthodes de la littérature, à savoir sur un nombre limité de niveaux au delà duquel les discontinuités sont de toute façons de l'ordre de la taille des pixels de l'affichage, et n'ont donc plus d'importance.

Généricité

Toutes les méthodes présentées permettent de traiter tout type de maillage. Les méthodes volumiques permettent même de traiter des surfaces implicites, car elles n'utilisent que la position 3D des points, voire la normale. C'est au niveau du rapport entre la résolution des texels et la finesse du maillage que les autres méthodes se différencient.

Les méthodes stockant les texels directement sur les faces du maillage placent au moins un texel par face, et ne peuvent donc pas avoir une résolution de texture plus faible que celle du maillage. Elles s'adaptent très bien au maillages actuellement utilisés dans le domaine de l'industrie audio visuelle ou des jeux vidéos où les maillages sont en règle générale les plus légers possibles. La résolution des maillages a toutefois actuellement tendance à augmenter avec l'émergence des outils de sculpture de maillages.

À l'inverse, les méthodes utilisant des atlas créent des cartes par regroupement des faces du maillage, et la distorsion liée à la paramétrisation de ces cartes sera d'autant plus faible que le nombre de faces de la carte sera important. Ces méthodes requièrent donc que le maillage soit en général bien plus fin que la texture à plaquer dessus. Carr *et al.* [14] utilisent d'ailleurs leur atlas dans un but de compression, en créant des *cartes de géométrie* pour stocker la géométrie dans une texture et pouvoir simplifier le maillage.

En contraste, notre approche utilise un atlas de texture, permettant donc d'avoir une résolution de texture plus faible que la résolution du maillage. Nos atlas ne sont par contre pas générés par un processus de regroupement de faces, mais à l'aide d'une paramétrisation préservant une grille. La distorsion résultant, même si elle reste un peu plus importante que la distorsion issue d'une paramétrisation classique, est bien plus faible que pour les méthodes proposées par la littérature. Notre approche est donc à la fois appropriée pour les maillages légers et pour les maillages fins.

Efficacité

Étant donné la puissance des cartes graphiques actuelles, la plupart des méthodes proposées peuvent être mises en place en temps réel. À part dans le cas de l'utilisation d'un atlas de texture classique, l'accès à la texture doit cependant toujours être reprogrammé, et un accès texture coûtera en général bien plus cher que l'accès classique. Les systèmes temps réel, comme les jeux vidéo, sont en général contraints de fournir un certain nombre d'images par secondes. C'est à partir de cette contrainte qu'est évaluée le temps que pourra mettre la carte graphique pour dessiner une image, et donc le temps qu'auront les programmes spécialisés pour s'exécuter. Les accès à des texture sont actuellement le goulet d'étranglement de ces programmes spécialisés ; ils doivent donc être les plus rapides possibles.

Notre méthode utilise un atlas de texture classique. Elle ne nécessite pas de reprogrammer l'accès à la texture – sauf pour l'utilisation des filtrages biquadratique et bicubique qui ne sont pas supportés nativement. L'avantage est ici double, car d'une part, l'accès est extrêmement efficace, et d'autre part, il n'est pas nécessaire de modifier les moteurs de rendu actuels, qui la supportent nativement.

En terme d'espace mémoire, l'utilisation d'un atlas de texture rend par contre notre approche légèrement moins efficace. D'une part, l'utilisation d'un atlas engendre la perte d'une portion des texels de l'image, à moins que l'agencement ne soit parfait, ce qui n'arrive en pratique que très rarement. Les méthodes présentées ici ont la particularité de générer des cartes rectangulaires, ce qui les rend particulièrement faciles à agencer, et limite beaucoup l'espace texture perdu. Cet espace est cependant gagné au prix d'une distorsion plus grande de la paramétrisation, qui ne peut être compensée qu'en augmentant la résolution de la texture. D'autre part, notre méthode utilise une paramétrisation préservant une grille ce qui contraint l'agencement de l'atlas pour préserver cette propriété. La taille de nos atlas ne peut en particulier pas être augmentée facilement pour augmenter la résolution de la texture, il est en général nécessaire de le reconstruire en grande partie.

Mise en place

Les méthodes nécessitant le moins de travail préalable à l'édition de la texture sont probablement celles qui stockent les texels sur les faces du maillage, car l'échantillonnage des faces est complètement automatique. La seule contrainte est que si la connectivité du maillage est modifiée il peut-être nécessaire de transférer la texture entre les anciennes faces et les nouvelles, ce qui reste meilleur que la plupart des méthodes actuelles. La méthode de González et Patow [45] offre l'avantage de s'appliquer sur un atlas de texture classique, et est donc compatible avec tous les modèles conçus à base d'atlas de texture, qui constituent la vaste majorité des modèles utilisés jusqu'à présent.

Dans notre approche, il est nécessaire de calculer au préalable une paramétrisation préservant une grille. Une telle paramétrisation peut-être calculée de manière automatique comme expliqué en Chapitre 1, même si une intervention de l'utilisateur peut grandement améliorer le résultat au moment de la génération du champ de directions. À partir de cette paramétrisation, l'atlas est calculé de manière totalement automatique. Dans le cas particulier des maillages quadrangulaires, une paramétrisation préservant une grille triviale peut-être calculée simplement en paramétrant chaque face du maillage sur le carré unité, et dans ce cas, notre méthode devient quasiment équivalente à celle de Burley et Lacewell [11]. Si l'utilisateur décide d'augmenter localement la résolution de la texture, seul l'atlas doit être recalculé, et non la paramétrisation préservant une grille, ce qui reste un processus complètement automatique.

Nous allons maintenant aborder le détail de notre contribution, en commençant par le cas le plus simple, c'est à dire la placage de texture utilisant le filtrage bilinéaire, sans différence de résolution de part et d'autre d'une couture, et sans mipmapping. Nous aborderons ces deux extensions par la suite dans les Sections 2.6 et 2.7.

2.5 Contribution : génération d'un atlas spécifique

Cette section détaille la construction d'un atlas de texture pour le placage de textures sans couture visible, à partir d'une paramétrisation préservant une grille. La construction de ce type de paramétrisation est détaillée en Chapitre 1. Cette construction suit les étapes suivantes :

Paramétrisation préservant la grille $\mathbb{Z}^2$: il s'agit de la base de notre méthode. La paramétrisation préservant $\mathbb{Z}^2$ permet de définir une grille sur la surface qui correspondra aux cellules de filtrage.

Contraintes entre texels : à partir de la paramétrisation, chaque triangle de la surface est considéré comme appartenant à une carte indépendante, ce qui revient à considérer toutes les arêtes comme des transitions. Selon le mode de filtrage, la position des texels est déterminée ; chaque carte se voit associer l'ensemble des texels utiles au placage de texture sur le triangle associé. Pour que les transitions ne donnent pas lieu à des coutures visible, un ensemble de contraintes est calculé entre les texels utiles des différentes cartes. Ces contraintes sont représentées par des classes d'équivalence de texels.

Agencement de l'atlas : une fois l'ensemble des contraintes déterminé, les cartes correspondant à des triangles individuels sont agglomérées pour former des cartes plus vastes et limiter l'espace texture perdu. Pour ne pas introduire de contraintes inutiles, cette agglomération doit prendre en compte les classes d'équivalence, pour ne pas fusionner deux texels de deux classes d'équivalence différentes. Les cartes ainsi produites sont enfin agencées pour former un atlas de texture.

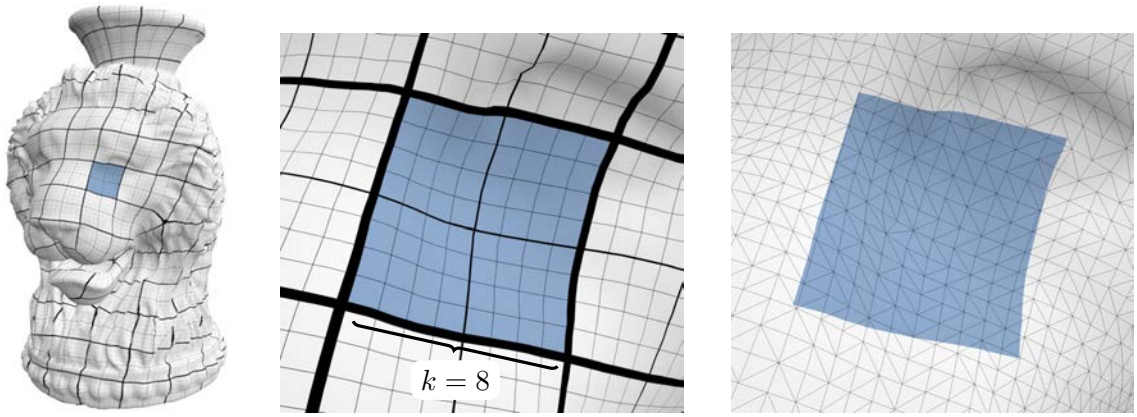
2.5.1 Agencement des texels sur la surface

L'idée d'utiliser une grille rectangulaire pour permettre de masquer les coutures au niveau des transitions est une idée récurrente dans la littérature. Notre originalité par rapport aux méthodes précédentes est que nous utilisons une grille directement pour disposer les texels sur la surface, et non pour créer des cartes paramétrées sur des rectangles du domaine paramétrique.

Utilisation d'une paramétrisation préservant une grille

La méthode la plus proche de notre approche est probablement celle de Burley et Lacewell [11]. Dans ces travaux, les auteurs requièrent que le maillage à texturer soit un maillage quadrangulaire. Une texture est alors alignée sur chaque face du maillage. La grille naturellement définie par le maillage est donc en quelque sorte raffinée pour placer les texels et donc créer la grille des cellules de filtrage. Pour utiliser cette méthode sur un maillage quelconque, il suffirait donc de réaliser un remaillage quadrangulaire de cette surface, et c'est là que les paramétrisations préservant une grille entrent en jeu, en tant que méthodes automatiques pour réaliser cette opération.

Il n'est en réalité pas nécessaire de remailler la surface. Une fois la paramétrisation préservant une grille calculée, les faces du remaillage sont définies via la paramétrisation par les cellules de la grille $\mathbb{Z}^2$. En utilisant une paramétrisation préservant une grille pour réaliser le placage de texture et en faisant correspondre les cellules de filtrage avec la grille $\mathbb{Z}^2$, les cellules de filtrage s'agenceront donc naturellement sur la surface pour former une grille. Cette convention sur la disposition des cellules de filtrage correspond à celle que nous avions prise en Section 2.3.3, page 54. Pour augmenter la résolution de la texture,



(a) (b) (c)

FIGURE 2.9 – Une paramétrisation préservant une grille fournit un ensemble de coordonnées paramétriques telles que l'image de la grille $\mathbb{Z}^2$ soit une grille sur la surface paramétrée (a). Cette grille initiale peut être raffinée en multipliant les coordonnées paramétriques par un facteur k entier (b). Contrairement aux méthodes précédentes fondées sur des atlas, les cellules de notre grille ne sont pas définies en regroupant des triangles, et un triangle peut être couvert par plusieurs cellules (c).

deux solutions sont possibles. Tout d'abord, nous pouvons multiplier les coordonnées paramétriques de la paramétrisation préservant une grille par un entier k, ce qui conserve la propriété de préservation de la grille et permet de placer k^2 cellules de filtrage au lieu d'une seule. Il est également possible de calculer une nouvelle paramétrisation préservant une grille en modifiant le paramètre définissant le nombre de faces souhaitées dans le remaillage. Il s'agit du paramètre h de l'Équation 1.5.2. Une paramétrisation préservant une grille et son raffinement sont illustrés sur la Figure 2.9

Notez que par cette méthode, les cellules de filtrage peuvent s'étendre de part et d'autre des transitions. Les paramétrisations préservant une grille ne sont pas conçues pour permettre le placage de textures. En particulier, la paramétrisation n'est pas bijective, ce qui se traduit par des recouvrements dans l'espace paramétrique.

Cartes

Notre méthode commence par considérer chaque triangle comme faisant partie d'une carte indépendante. Les cartes seront ensuite agglomérées pour former des cartes de domaines plus vastes. Pour un triangle T_p, nous noterons C_p le triangle paramétrique correspondant à T_p. Une *carte* correspond à une paire (T_p,C_p). À chaque carte est ensuite associé un ensemble de *texels utiles*, correspondant à l'ensemble des texels pour lesquels il existe un point de la carte où l'influence du texel est non nulle. Notez que l'influence des texels – et donc l'ensemble des texels utiles d'une carte – dépend du mode de filtrage utilisé, comme illustré en Figure 2.10. Nous noterons t^p un texel utile en C_p et $\mathbb{T}_p$ l'ensemble de ces texels. $\mathbb{T}_p$ peut être calculé en utilisant un algorithme de *matricialisation* (en anglais « rasterisation ») du triangle C_p. Notez que l'ensemble des texels utiles est plus vaste que l'ensemble des texels dont la position est à l'intérieur de C_p. Chacun des texels utiles est pour l'instant considéré comme un degré de liberté auquel il est possible d'associer n'importe quelle valeur. Nous verrons par la suite que pour assurer le placage sans coutures, il est nécessaire de réduire ces degrés de liberté.

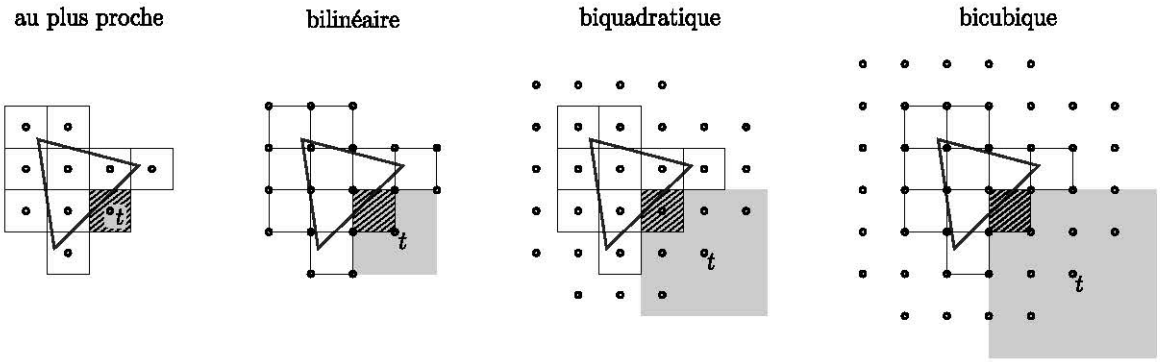


FIGURE 2.10 – Texels utiles pour une même carte, selon le mode de filtrage. Quelque soit le mode de filtrage, le même ensemble de cellules de filtrage est utilisé pour texturer le triangle. Étant donné un texel t, la zone d'influence de ce texel (grisée) s'élargit lorsque le degré du filtrage croît ; une même cellule de filtrage (hachurée) est sous l'influence de texels de plus en plus éloignés.

Nous choisissons de traiter chaque triangle indépendamment car en définissant les cartes avant de déterminer l'ensemble des texels utiles, il est peut arriver de restreindre les degrés de liberté normalement disponibles, comme le montre la Figure 2.11.

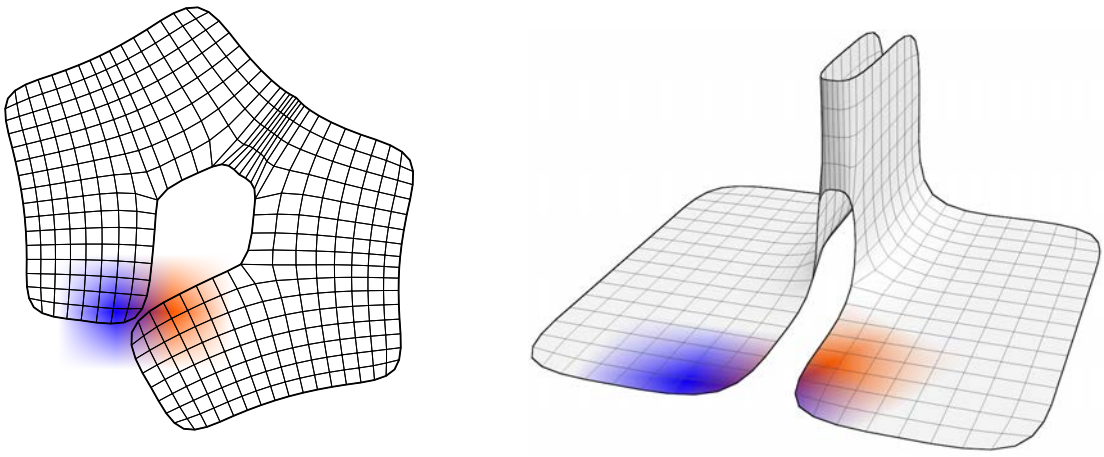
2.5.2 Classes d'équivalence

Nous détaillerons ici les contraintes nécessaires entre les texels utiles des différentes cartes pour assurer un placage sans couture apparente.

Construction

Notre approche est fondée sur une paramétrisation préservant une grille $\mathbb{Z}^2$. Une telle paramétrisation est définie par le fait que pour chaque arête $e_{p,q}$ la fonction de transition $\tau_{p,q} : C_p \to C_q$ est une bijection linéaire de $\mathbb{Z}^2$. Dans notre cas particulier, cette propriété fait que l'image d'une cellule de filtrage de C_p est une cellule de filtrage de C_q et que l'image de la position d'un texel $t^p \in \mathbb{T}_p$ est la position d'un texel $t^q \in \mathbb{T}_q$. Soit $\mathbf{x}$ un point sur $e_{p,q}$. L'arête $e_{p,q}$ étant une transition, à $\mathbf{x}$ correspondent deux jeux de coordonnées paramétriques, $\mathbf{u}^p$ et $\mathbf{u}^q$ respectivement dans C_p et C_q, tels que $\tau_{p,q}(\mathbf{u}^p) = \mathbf{u}^q$. Pour qu'aucune couture n'apparaisse lors du placage au niveau de $e_{p,q}$, il faut que les valeurs $h(\mathbf{u}^p)$ et $h(\mathbf{u}^q)$ calculées respectivement en utilisant les valeurs des texels de $\mathbb{T}_p$ et $\mathbb{T}_q$ soient identiques. Pour un t^p utile en $\mathbf{u}^p$, il existe un texel t^q tel que $\mathbf{p}(t^q) = \tau_{p,q}(\mathbf{p}(t^p))$. En s'assurant que $h(t^p) = h(t^q)$ pour tout texel t^p, le filtrage $h(\mathbf{u}^p)$ sera égal au filtrage $h(\mathbf{u}^q)$. Ce procédé est illustré sur la Figure 2.12.

Pour traiter l'ensemble de ces contraintes, nous avons choisi d'utiliser une structure de données pour gérer des ensembles de classes d'équivalence [20, Section 21.3]. À tout texel t^p nous associons une classe $c(t^p)$. Notre structure permet en temps quasi constant de déterminer la classe d'un texel et de réaliser la fusion de plusieurs classes d'équivalences. Initialement, chaque texel est dans sa propre classe d'équivalence. Nous considérons ensuite itérativement chaque arête $e_{p,q}$ du maillage et déterminons l'ensemble $\mathbb{T}_{p,q}$ des texels dont l'influence contient les coordonnées paramétriques d'un point de $e_{p,q}$. Via la fonction de transition $\tau_{p,q}$ nous déterminons les texels correspondants $\mathbb{T}_{q,p}$ pour la carte C_q, et fusionnons les classes d'équivalence des paires de texels en correspondance.



(a) (b)

FIGURE 2.11 – Lorsque l'atlas est construit avant l'étude des contraintes et des degrés de liberté du placage de texture, un atlas mal conçu (a) peut amener deux portions éloignées de la surface à utiliser le même ensemble de texels (ici en bleu et orange) lors du placage (b) et ainsi réduire les degrés de liberté normalement disponibles. En traitant chaque face comme une carte indépendante et disposant d'un ensemble de texels indépendant, nous nous assurons de préserver le maximum de degrés de liberté.

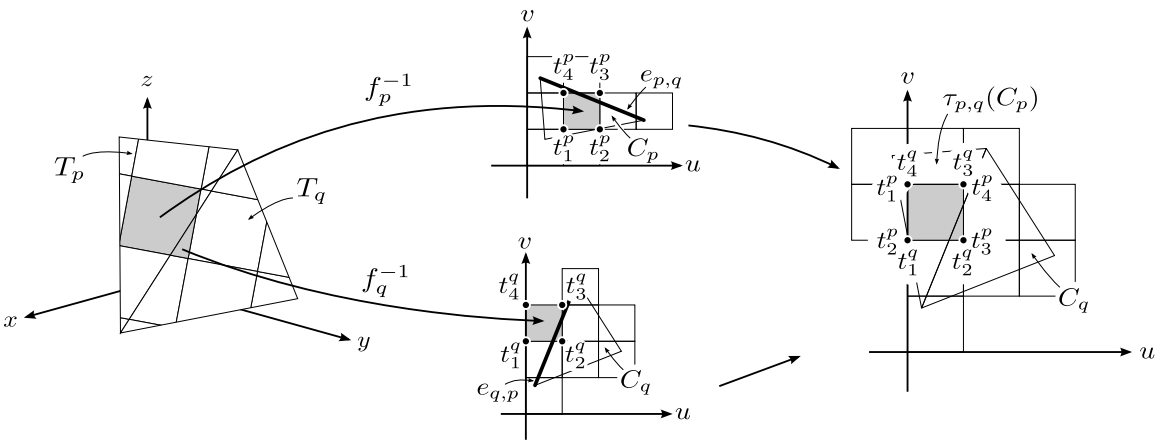


FIGURE 2.12 – Fusion de classes d'équivalence de texels. Une cellule de filtrage sur le maillage correspond à plusieurs cellules de filtrage dans le domaine paramétrique, une dans le domaine C_p de la paramétrisation du triangle T_p et une autre dans le domaine C_q de la paramétrisation du triangle T_q. Dans C_p, avec le filtrage bilinéaire, une valeur dans cette cellule est calculée à partir des texels $\{t_1^p, t_2^p, t_3^p, t_4^p\}$. Dans C_q, les texels utilisés sont $\{t_1^p, t_2^p, t_3^p, t_4^p\}$. Via la fonction de transition $\tau_{p,q}$ entre les domaines des deux cartes, les classes d'équivalence de ces deux ensembles de texels sont deux à deux fusionnées, et les couples de texels (t_1^p, t_4^q), (t_2^p, t_1^q), (t_3^p, t_2^q) et (t_4^p, t_3^q) seront contraints à partager la même valeur. Ainsi, quelle que soit la carte utilisée, les cellules de filtrage fourniront la même valeur et aucune discontinuité n'apparaîtra le long de l'arête $e_{p,q}$ lors du placage.

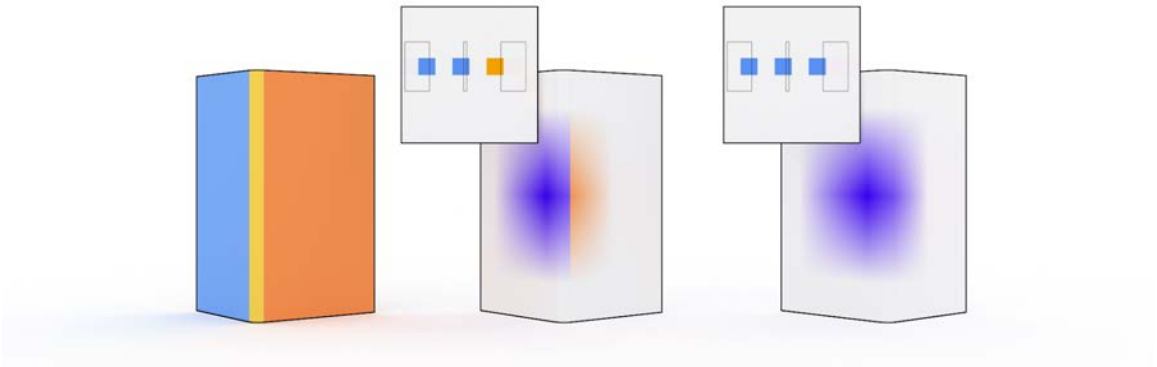


FIGURE 2.13 – Couture en cas de correction naïve des valeurs des texels. Ici, l'atlas est composé de trois cartes (colorées à gauche). En utilisant une correction naïve se contentant pour chaque transition de mettre les texels correspondant à la même valeur, certains texels sont édités plusieurs fois et des coutures apparaissent (au centre). Les classes d'équivalence permettent de s'assurer que tous les texels devant avoir la même valeur seront traités ensemble et qu'aucune couture ne sera visible (à droite).

Définition 2.1. *Deux texels* $t^p \in \mathbb{T}_p$ *et* $t^q \in \mathbb{T}_q$ *seront équivalents si et seulement s'il existe une suite de texels* $(t^0, \ldots, t^n) \in \mathbb{T}_0 \times \ldots \times \mathbb{T}_n$ *tels que :*

- $t^0 = t^p$ *et* $t^n = t^q$
- *pour tout* $k \in [0,n-1]$, $t^k \in \mathbb{T}_{k,k+1}$ *et* $\mathbf{p}(t^{k+1}) = \tau_{k,k+1}(\mathbf{p}(t^k))$

Cette méthode fondée sur un ensemble de classes d'équivalence nous permet, en comparaison des méthodes de l'état de l'art, de gérer simplement à la fois les modes de filtrage d'ordre supérieur, et les singularités. Elle permet également de traiter correctement des situations où une méthode naïve ne fonctionne pas, comme dans le cas de la Figure 2.13. Pour gérer un filtrage d'ordre supérieur, il suffit de prendre en compte l'élargissement de l'influence des texels pour déterminer les texels utiles ; dans le cas des singularités, aucun traitement supplémentaire n'est requis, même si les relations naturellement issues des classes d'équivalences entre les texels au voisinage d'une singularité ne sont pas triviales, comme le montre la Figure 2.14. Les classes d'équivalence permettent également de prendre en compte des situations où des texels doivent avoir la même valeur alors que leurs cartes ne sont pas adjacentes, ce qui n'est en général pas considéré par les méthodes existantes.

Position d'une classe d'équivalence

Une fois les classes d'équivalence construites, il suffit de s'assurer que tous les texels d'une même classe aient la même valeur pour qu'aucune couture ne soit visible lors du placage. Nous choisissons à cette fin parmi les texels de la classe un texel particulier, que nous appellerons le texel de référence de la classe. Tous les texels de la classe se verront associer la valeur de ce texel. Pour sélectionner ce texel, nous calculons pour chaque texel t^p la distance signée de sa position $\mathbf{p}(t^p)$ au triangle C_p qui la contient. Notez que cette distance est négative pour un texel dont la position est à l'intérieur du triangle, nulle pour un texel sur une arête, et positive pour un texel à l'extérieur du triangle. Nous sélectionnons ensuite le texel dont la distance est la plus faible. Dans la plupart des cas, une classe aura un et un seul texel pour lequel la distance signée sera négative. Ce texel se trouvant à l'intérieur du triangle, sa position correspond aux coordonnées paramétriques d'un point sur la surface, et il pourra donc être édité par un outil de peinture de maillage. À l'inverse, un texel ayant une

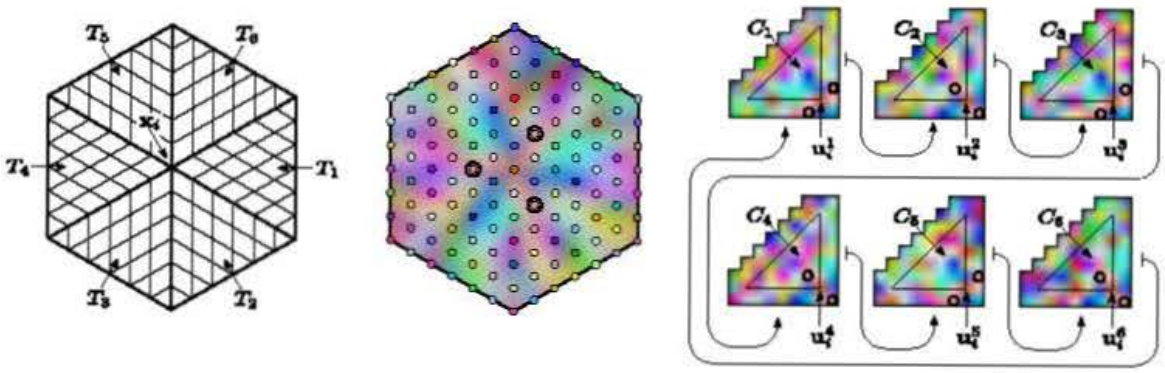


FIGURE 2.14 – Classes d'équivalence au voisinage d'une singularité d'indice $-\frac{1}{2}$, dans le cadre du filtrage bicubique. À gauche, les cellules de filtrages de six triangles $\{T_k\}_{k=1}^{6}$ s'agencent pour former une singularité d'indice $-\frac{1}{2}$ au sommet $\mathbf{x}_i$. Au centre une texture aléatoire est plaquée sans couture visible, avec un filtrage bicubique. Les points de la surface dont les coordonnées correspondent aux positions de texels sont marqués avec la couleur des texels correspondants. Nous voyons apparaître des points de la surface distincts correspondant à des texels contraints à avoir la même valeur. À droite, les versions paramétriques $\{C_k\}_{k=1}^{6}$ des triangles, avec les texels utiles de chaque carte. Les flèches indiquent comment les cartes se recollent. La classe correspondant aux trois positions marquées sur la surface au centre contient deux texels par carte, centrés sur les coordonnées paramétriques $\mathbf{u}_i^k$ de la singularité. Sur trois des cartes, l'un de ces texels est à l'intérieur du triangle paramétrique, ces trois texels correspondent à ceux marqués sur la surface au centre.

distance positive correspond à un texel dont la valeur ne pourra pas être éditée directement. Ce choix du texel de référence fait donc que dans la plupart des cas, aucun des texels édité par un artiste ne verra sa valeur modifiée. La seule situation où plusieurs texels ont une valeur négative correspond au voisinage d'une singularité, comme le montre la Figure 2.14.

Pour repérer le texel de référence, nous enregistrons pour chaque classe l'indice p de la carte contenant le texel de référence, et la position de ce texel dans le triangle C_p, en coordonnées barycentriques. Nous appellerons cette position la *position de classe*, et pour un texel t^p, nous noterons la position de sa classe $\mathbf{p}(c(t^p))$. Cette encodage nous permet de retrouver le texel de référence après la construction de l'atlas, où les cartes seront agglomérées en cartes plus importantes, puis agencées dans l'espace texture.

2.5.3 Construction d'un atlas

Pour pouvoir réaliser le placage, il est nécessaire de construire un atlas de texture, et donc d'associer à la surface un nouveau jeu de coordonnées paramétriques. La contrainte principale de cette étape est que la nouvelle paramétrisation doit toujours préserver la grille pour pouvoir toujours assurer un placage sans couture apparente.

Regroupement des cartes

Jusqu'à présent, nous considérions chaque face T_p de la surface comme faisant partie d'une carte indépendante. Pour économiser de l'espace texture, il est possible d'agglomérer ces petites cartes en cartes plus vastes. Notez cependant que cette étape est facultative, et qu'il est tout à fait possible d'agencer directement ces cartes à une seule face.

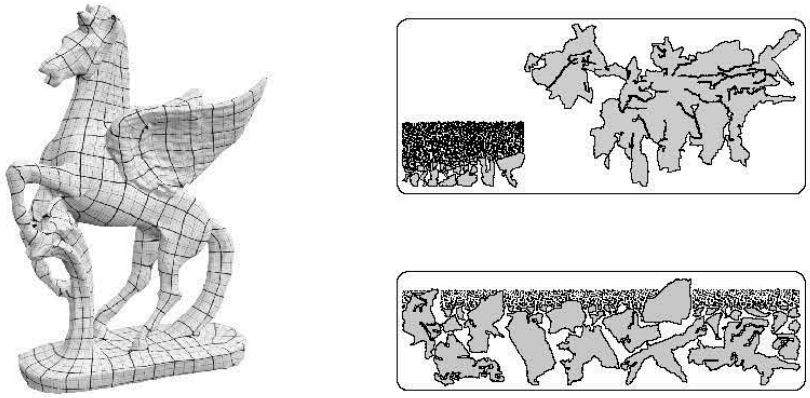


FIGURE 2.15 – Agglomération des cartes individuelles pour former des cartes plus vastes. À gauche, la surface, et la paramétrisation préservant une grille utilisée. À droite, deux atlas utilisant respectivement l'agglomération gloutonne – en haut – et une segmentation préalable – en bas. Lorsqu'une segmentation est calculée, le cartes sont plus faciles à agencer, et permettent de gagner de l'espace texture.

Nous réalisons l'agglomération des cartes de manière gloutonne en considérant itérativement chaque arête $e_{p,q}$ du maillage, et en fusionnant les cartes courantes des triangles T_p et T_q de part et d'autre de cette arête lorsque cette opération est possible. Pour réaliser cette fusion, nous appliquons la fonction de transition $\tau_{p,q}$ à la face paramétrique C_q correspondant à T_q pour amener les deux cartes dans le même repère, de telle sorte que les faces paramétriques C_p et C_q soient adjacentes. Cette opération amène un certain nombre de texels utiles de $\mathbb{T}_q$ à partager la même position que certains texels utiles de $\mathbb{T}_p$. La fusion sera considérée comme possible lorsque les couples de texels utiles des deux cartes partageant la même position dans la carte fusionnée appartiennent à la même classe d'équivalence.

Cet algorithme glouton peut fournir des cartes de formes compliquées à agencer par la suite. Pour contrôler la forme des cartes, nous effectuons au préalable une segmentation du maillage par la méthode de Eck *et al.* [32]. Les arêtes délimitant les différentes régions du maillage ainsi construites ne seront pas utilisées pour fusionner deux cartes ensembles, et resteront des transitions. La Figure 2.15 montre sur un exemple l'espace texture que cette simple heuristique permet de gagner.

Agencement des cartes

Pour agencer les cartes, notre méthode utilise l'approche classique de l'agencement Tetris, introduite par Lévy *et al.* [63]. L'idée de cette méthode consiste à placer les cartes en utilisant une stratégie gloutonne qu'un joueur pourrait utiliser au jeu de Tetris. À chaque nouvelle pièce, le joueur teste toutes les positions possibles de la pièce, et choisit celle qui laisse le moins de cases vides en dessous de la pièce une fois qu'elle est posée. Pour mettre en œuvre cette technique, un tableau stocke pour chaque colonne la hauteur courante des pièces. Pour chaque position de la pièce, la hauteur est mise à jour, et l'aire entre les deux hauteurs est calculée. La position optimale selon le coût que nous nous somme définis est celle qui minimise l'aire entre la courbe des hauteurs avant la pose de la nouvelle pièce, et après.

Pour appliquer cette idée de base à notre problème, pour chaque carte, l'ensemble des texels utiles forme un assemblage de carrés qu'il est possible d'assimiler à une pièce de Tetris un peu sordide (un polyomino). Nous avons toutefois l'avantage de pouvoir choisir l'ordre des pièces. Les cartes sont ainsi traitées de la plus grosse à la plus petite, dans l'idée que plus une pièce est petite, moins il sera difficile de la loger. Nous nous fixons une largeur initiale l correspondant au nombre de colonnes autorisées, et lançons l'algorithme décrit précédemment, en mettant à jour la hauteur pour chaque colonne. Lorsque l'ajout d'une nouvelle carte nous fait dépasser la hauteur l, nous doublons la valeur de l et relançons l'algorithme, jusqu'à réussite. Nous nous sommes contentés de cette approche, même s'il est possible d'améliorer le résultat en ajoutant les cartes par plusieurs directions. Nous disposons ainsi d'un atlas rendant possible le placage de textures, car les cartes n'ont pas de recouvrement local si la paramétrisation préservant une grille n'en a pas, et les recouvrement globaux sont évités par le procédé d'agglomération des cartes individuelles, et par le processus d'agencement de l'atlas.

Texture de correspondances

À partir de notre atlas, il est possible pour l'utilisateur de plaquer n'importe quelle texture via le placage de textures naturellement supporté par le matériel graphique. Le placage n'exhibera pas de coutures tant que les contraintes définies par les classes d'équivalence sont respectées. Nous fournissons donc cette ensemble de contraintes sous forme d'une *texture de correspondances*, qui permettra de corriger n'importe quelle texture fournie pour s'assurer que les classes d'équivalence partagent la même valeur. Notre texture de correspondances correspond exactement aux *textures d'indirection* décrites par Lefebvre et Hoppe [60]. Nous n'utilisons pas ce terme, car nous stockerons par la suite plus d'informations dans nos textures de correspondances.

La texture de correspondances se présente sous la forme d'une texture – un tableau 2D de la taille de l'atlas – stockant pour chaque texel la position à laquelle la valeur doit être récupérée. Dans le cas le plus courant, le texel est seul dans sa classe d'équivalence, et la valeur stockée correspond à la propre position du texel. Lorsque plusieurs texels partagent une classe d'équivalence, nous stockons pour tous les texels la position de la classe, correspondant à la position du texel de référence de la classe définie en Section 2.5.2, page 70. L'intérêt de stocker ces positions dans une texture est qu'il est alors possible de stocker cette texture sur la carte graphique et en modifiant légèrement l'accès à la texture d'appliquer dynamiquement les contraintes, notamment lors du processus d'édition de la texture. Notez cependant que la texture de correspondances n'est utilisée que pour corriger une texture fournie, et qu'une fois la texture corrigée, il n'est plus nécessaire de l'utiliser. Le modèle muni de l'atlas et de la texture corrigée peut être intégré à n'importe quelle système de rendu, et le placage n'aura pas de coutures visibles.

2.5.4 Résultats

Nous fournissons tout d'abord un exemple simple de nos résultats pour différents modes de filtrage. Les modes au plus proche, bilinéaire et bicubique sont illustrés en Figure 2.16. Une application pour laquelle les coutures sont particulièrement gênantes est le placage de relief. Le but de cette technique est d'ajouter des détails issus d'une texture à un objet décrit par un maillage grossier. Le maillage est alors envoyé à la carte graphique pour le rendu, qui va automatiquement subdiviser les triangles fournis pour augmenter la résolution du maillage, et la position de chacun des sommets ainsi créés est modifiée en utilisant

FIGURE 2.16 – Placage sans couture visible pour différents modes de filtrages. De gauche à droite, les différentes cartes et les filtrages au plus proche bilinéaire et bicubique. Ici, la paramétrisation contient quatre singularités d'indice $\frac{2}{2}$ visibles en haut et en bas de la sphère, en particulier sur le filtrage au plus proche. Aucune couture n'est apparente, même autour des singularités.

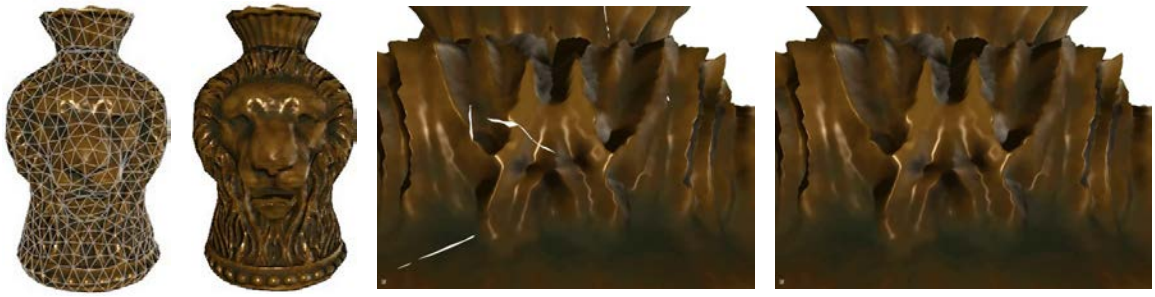


FIGURE 2.17 – Placage de relief pour ajouter du détail à un modèle grossier. À gauche, le maillage utilisé. Au centre, au niveau des coutures, les déplacement utilisés pour perturber la surface ne se correspondent pas, créant des fentes au niveau des transitions. En utilisant notre méthode, ces artéfacts disparaissent.

une valeur donnée par une texture. Cette technique permet ainsi de donner une apparence détaillée à un objet décrit par un nombre réduit de polygones. Un exemple de cette technique est donné en Figure 2.17. Les discontinuités au niveau des transitions, dans cet exemple, produisent des fentes très visibles, que notre méthode permet de supprimer. Dans une application interactive de type sculpture où l'utilisateur édite le contenu de la texture fournissant les déplacement à appliquer, il est possible de fournir à la carte graphique la texture de correspondances, et de l'appliquer dynamiquement. L'utilisateur peut ainsi éditer le relief de la surface sans voir apparaître de fentes.

L'application des contraintes de la texture de correspondance permet également de fournir une couleur aux texels qui ne peuvent pas être directement peints dans une application de peinture sur surface, comme illustré sur la Figure 2.18. Par comparaison, lorsqu'un atlas classique de texture est utilisé, il est nécessaire d'étendre automatiquement les couleurs vers l'extérieur des cartes, pour faire en sorte que tous les texels utiles aient une couleur, y compris ceux dont la position est à l'extérieur d'une carte. En utilisant notre méthode, à part au niveau des singularités, ces texels sont les seuls pour lesquels la couleur sera déterminée par les contraintes de la texture de correspondances.

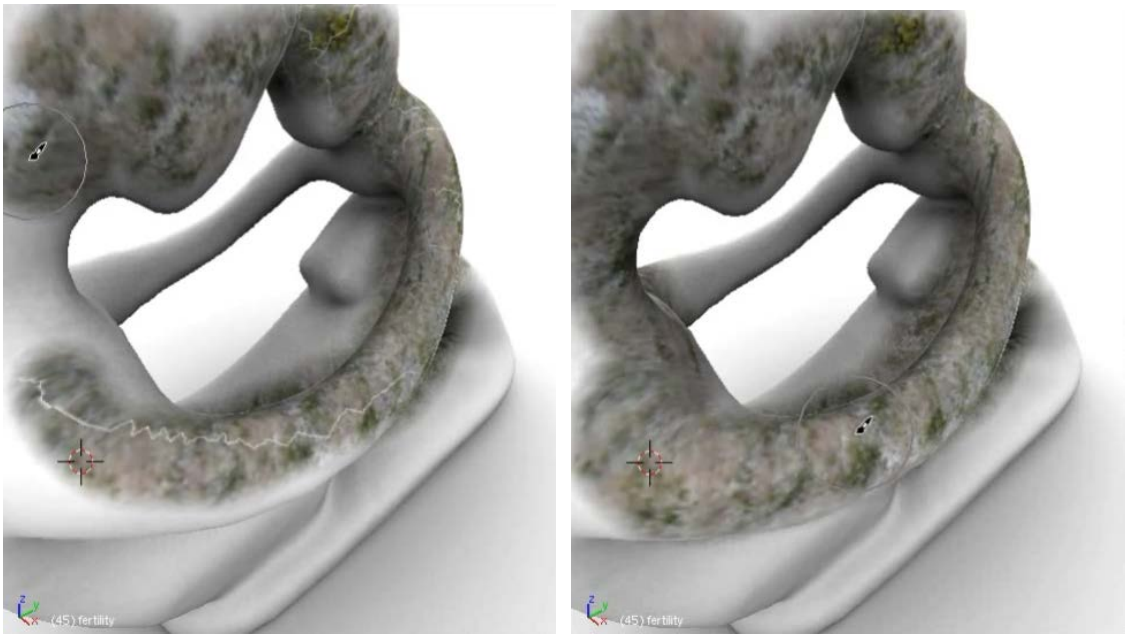


FIGURE 2.18 – Correction dynamique de la texture. Dans une application de peinture sur surface, certains texels utiles ne peuvent pas être peints directement sans utiliser de technique attribuer une couleur aux texels bordant les cartes (à gauche). En appliquant les corrections indiquées par la texture de correspondance, ces texels obtiennent une couleur et les coutures sont invisibles.

Dans notre implémentation, nous utilisons une structure de matrice creuse pour enregistrer l'ensemble des texels utiles au niveau des transitions. La majorité du calcul est investie dans le calcul de la paramétrisation préservant la grille $\mathbb{Z}^2$. Cette opération n'a jamais demandé plus d'une minute dans tous les résultats que nous présenterons. Le calcul des contraintes et de la carte de correspondance associée est de l'ordre de la seconde. Au niveau de l'espace texture, notre algorithme d'agencement des cartes est très similaires aux algorithmes classiques, et l'espace texture perdu est du même ordre qu'avec un atlas classique.

2.6 Contribution : résolution adaptative

Pour économiser de l'espace texture, une technique classique consiste à ajuster localement la résolution nécessaire, en jouant sur la taille des cartes dans l'espace texture. Plus une carte est grande, plus la résolution sera élevée. Les paramétrisations préservant une grille ne permettent pas de changer indépendamment la taille des cartes, car au niveau d'une transition, les arêtes de part et d'autres n'auraient plus la même taille dans le domaine paramétrique, et la fonction de transition ne préserverait donc plus la grille. En s'assurant que le facteur d'échelle de part et d'autre de la transition est une puissance de deux, et en alignant correctement les cellules d'interpolation de part et d'autre des transitions, il est toutefois possible de formuler les contraintes nécessaire pour obtenir un placage sans couture visible. Ces contraintes sont fondées sur les propriétés de subdivision des modes de filtrage auxquels nous nous intéressons.

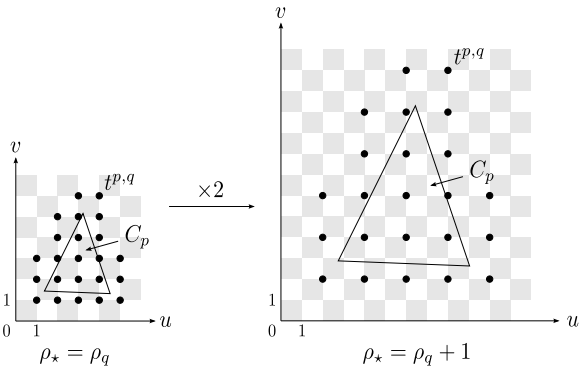


FIGURE 2.19 – Coordonnées paramétriques et position des texels par rapport à la résolution active $\rho_\star$. Nous nous plaçons ici dans le cadre du filtrage bilinéaire. Un texel $t^{p,q}$ utile à la résolution ρ_q n'est positionné sur la grille des texels – $\mathbb{Z}^2$ en mode bilinéaire – que lorsque $\rho_\star = \rho_q$. En augmentant de 1 la résolution active, les coordonnées des sommets de C_p ainsi que les positions des texels de $\mathbb{T}_p^q$ sont multipliées par 2.

2.6.1 Pyramide de classes d'équivalence

Résolution

À chaque carte (T_p,C_p), nous associons une résolution que nous noterons ρ_p. Une carte ayant une résolution ρ_p verra les coordonnées des sommets de C_p multipliées par 2^{ρ_p} lors de la création de l'atlas final. Avant cette dernière phase de création de l'atlas, nous considérerons toujours toutes les cartes à la même résolution que nous appellerons la *résolution active*, que nous noterons $\rho_\star$. À cette résolution, le même facteur multiplicatif $2^{\rho_\star}$ est appliqué à toutes les coordonnées paramétriques de la surface.

Une fois déterminée la résolution active, et appliqué le facteur d'échelle sur les coordonnées paramétriques, les cellules de filtrage sont comme précédemment définies comme les cellules de la grille $\mathbb{Z}^2$, et les texels utiles d'une cartes sont déterminés comme précédemment, selon le mode de filtrage choisi par l'utilisateur. Nous noterons $\mathbb{T}_p^q$ l'ensemble des texels utiles en C_p, pour la résolution active $\rho_\star = \rho_q$. Un texel de $\mathbb{T}_p^q$ sera noté $t^{p,q}$. Les coordonnées des sommets de C_p étant variables, nous enregistrons la position $\mathbf{p}(t^{p,q})$ d'un texel utile à la résolution ρ_q en coordonnées barycentriques par rapport à ce triangle. Les coordonnées paramétriques des texels varient donc également avec la résolution active, et correspondront effectivement à la position de texels lorsque la résolution active correspondra à leur résolution. Ces notations sont illustrées en Figure 2.19. Pour une face T_p, après la construction de l'atlas final, seuls les texels de $\mathbb{T}_p^p$ correspondront effectivement à des texels pour lesquels l'utilisateur fournira une valeur.

Classes d'équivalence

Pour chaque résolution demandée pour au moins une face T_p, nous fixons la résolution active à $\rho_\star = \rho_p$, et construisons les classes d'équivalence entre texels comme précédemment, en considérant chaque arête, et en fusionnant les classes d'équivalence des texels de part et d'autre de l'arête. Tous les texels utilisés sont donc les texels de la résolution ρ_p. Seules les classes d'équivalence contenant au moins un texel $t^{q,p}$ tel que $\rho_q = \rho_p$ nous intéressent, car elles contiennent au moins un texels qui sera effectivement présent pour l'atlas final. Pour déterminer le texel de référence et la position d'une telle classe, deux cas de figure sont possible :

pour tout $t^{q,p}$ de la classe, $\rho_q \geq \rho_p$: dans ce cas, le texel de référence est déterminé comme précédemment, en calculant pour chaque texel de la classe la distance signée de sa position au triangle C_q de sa carte, et en choisissant le texel de distance minimale. Il

peut en particulier arriver que le texel $t^{q,p}$ ainsi choisi soit tel que $\rho_q > \rho_p$. Nous verrons comment gérer ce cas de figure lors de la correction de la texture de l'utilisateur.

il existe un texel $t^{q,p}$ de la classe tel que $\rho_q < \rho_p$: cette propriété signifie qu'une des arêtes utilisées pour propager les classes d'équivalence est adjacente à un triangle T_q de résolution $\rho_q < \rho_p$. Au niveau d'une telle transition, pour assurer un placage sans coutures, les texels des faces de résolution ρ_p de la classe doivent adapter leur valeur par rapport aux texels plus grossiers des faces voisines pour assurer la continuité. Nous choisissons donc un texel $t^{q,p}$ tel que $\rho_q < \rho_p$ comme texel de référence. Nous verrons que lorsque plusieurs texels de la classe ont cette propriété, le choix du texel n'a pas d'importance, car la correction de la texture de l'utilisateur fournira toujours la même valeur.

Une fois le texel de référence déterminé, nous enregistrons comme précédemment sa position comme position de classe, en coordonnées barycentriques par rapport au triangle de sa carte, ce qui nous permet d'être indépendant à la fois de la transformation que subira la carte pour être mise à la résolution demandée par l'utilisateur, et de la transformation que subira la carte lors du regroupement des cartes et de l'agencement.

Construction de l'atlas et texture de correspondance

Pour construire l'atlas, toutes les cartes sont mises à la résolution demandée, et les classes d'équivalences utilisées pour l'agglomération des cartes et l'agencement sont celle construites pour cette résolution. L'agglomération des cartes ne regroupe que des cartes qui ont la même résolution. Les cartes ainsi construites sont enfin agencées pour former l'atlas.

Dans la texture de correspondances, nous enregistrons comme précédemment la position de la classe du texel de l'atlas correspondant, mais stockons en plus la résolution de la carte qui le contient, et la résolution de la carte qui contient la position de sa classe. Ces deux résolutions seront utilisées lors de la correction de la texture de l'utilisateur.

2.6.2 Subdivision et valeur de classe

Subdivision

La subdivision des courbes splines est un procédé élaboré à l'origine pour fournir une méthode efficace pour les dessiner, et pour les évaluer en un point quelconque. Dans notre cas, la courbe correspond au champ de couleur sur la surface, et les points de contrôle sont les texels. Dans ce cadre, la subdivision consiste à doubler le nombre de texels, tout en faisant en sorte que le champ de couleur obtenu par le filtrage reste inchangé. Au niveau des cellules de filtrage, chaque cellule de filtrage est découpée en quatre cellules de filtrages plus petites, fournissant la même valeur de filtrage que la cellule de filtrage originale. Cette méthode nous permet donc d'augmenter virtuellement la résolution d'une carte, pour l'amener à la même résolution qu'une carte voisine. La subdivision d'un ensemble de texels est illustrée sur la Figure 2.20. Une fois les deux cartes à la même résolution, les mêmes contraintes que dans le cas de résolutions identiques sont appliquées, sauf que seuls les texels de résolution élevée sont des degrés de liberté. C'est la raison pour laquelle lors de la construction des cartes d'équivalence, le texel de référence est choisi au sein d'une carte de résolution plus grossière s'il en existe un. Après la subdivision, la position de classe correspond effectivement à un texel, et sa valeur peut être assignée à la classe.

FIGURE 2.20 – Subdivision bicubique des texels d'une texture aléatoire. Chaque subdivision préserve le champ de valeurs issu du filtrage, qui correspond également à la limite vers laquelle tendent les valeurs des texels. La subdivision est ainsi une méthode rapide pour calculer ce champ de valeur

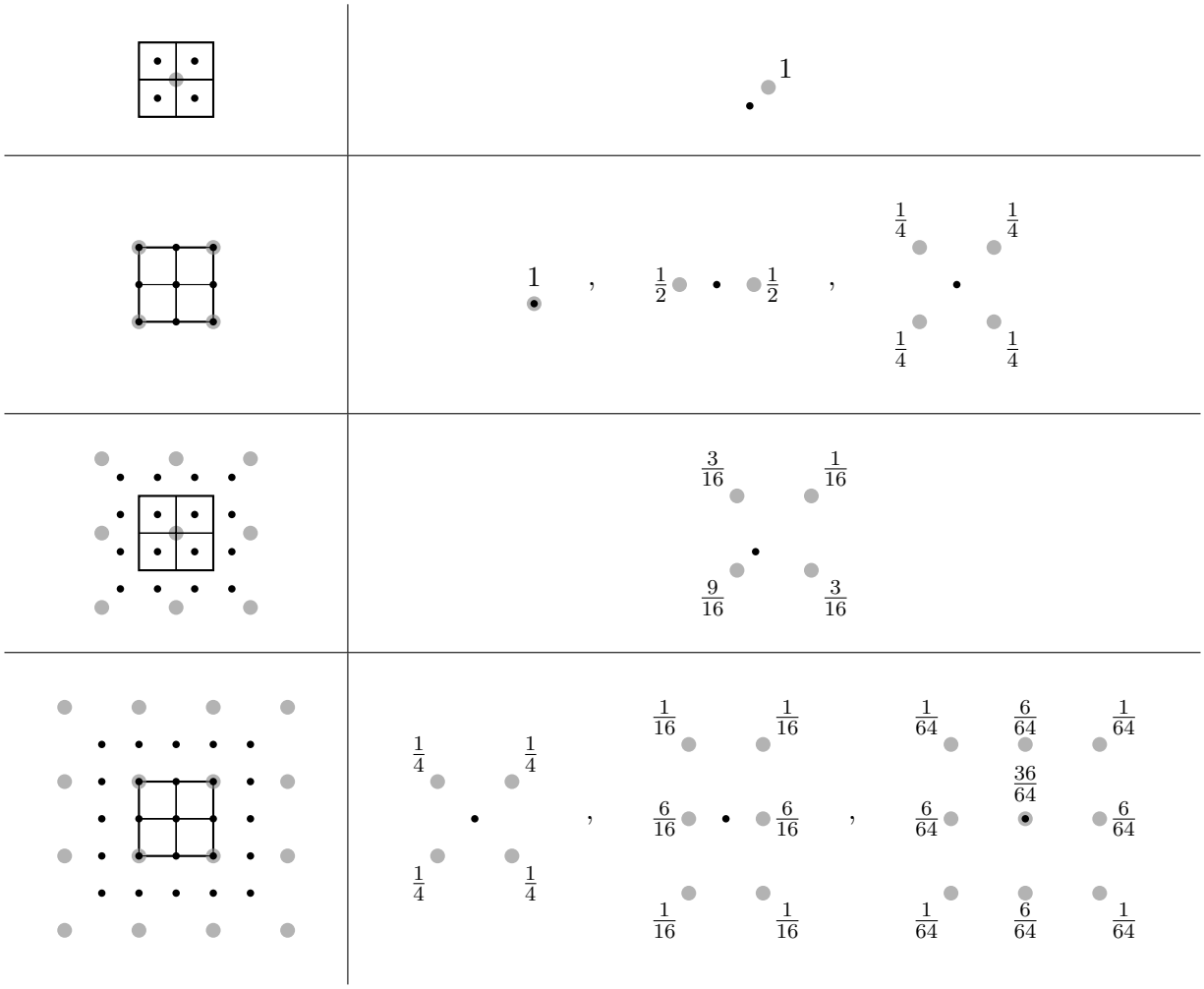


FIGURE 2.21 – Stencils de subdivision des modes de filtrage classiques, du filtrage au plus proche en haut au filtrage bicubique en bas. À gauche, la subdivision d'une cellule de filtrage donne quatre cellules de filtrage. Les texels utiles de la cellule de filtrage initiale (en gris) sont utilisés pour calculer les valeurs des texels utiles des cellules de filtrage subdivisées (en noir). À droite, la valeur de chaque nouveau texel est une combinaison linéaire des valeurs des texels initiaux. Les différents cas en fonction de la position du nouveau texel par rapport aux anciens texels sont énumérés, aux symétries près.

Pour réaliser cette opération, un stencil de subdivision est utilisé pour calculer les valeurs des texels subdivisés en fonctions des valeurs des texels originaux. Si les stencils pour le filtrage au plus proche et le filtrage bilinéaire sont triviaux car ces filtrages sont des interpolations, ceux des filtrages biquadratique et bicubique sont également bien connus, car ils sont utilisés pour définir des surfaces de subdivision, respectivement par la méthode de Doo et Sabin [25] et celle de Catmull et Clark [16]. Tous ces stencils sont fournis sur la Figure 2.21. Dans le cas des filtrages bilinéaire et bicubique, la subdivision est dite *primale*, car les texels issus de la subdivisions réutilisent les positions des texels originaux. Dans le cas contraire, pour les modes au plus proche et biquadratique, la subdivision est dite *duale*. Par la subdivision, nous pouvons donc associer une valeur $h(t^{p,q})$ au texel $t^{p,q}$ dès que $\rho_q \geq \rho_p$.

Un texel $t^{p,p}$ sera utilisé pour déterminer la valeur d'un texel $t^{p,q}$ avec $\rho_q > \rho_p$ lors de la subdivision si leurs zones d'influence sont telles que $[t^{p,q}] \subset [t^{p,p}]$. En effet, imaginons qu'il existe un point $\mathbf{u} \in [t^{p,q}]$ tel que $\mathbf{u} \notin [t^{p,p}]$. Mettons nous dans la situation où tous les texels de la résolution ρ_p ont la valeur 0, à l'exception de $t^{p,p}$. Par définition de la zone d'influence, nous savons que pour tout $\mathbf{u}' \notin [t^{p,p}]$, $h_p(\mathbf{u}') = 0$ où h_p est le filtrage utilisant les texels de résolution ρ_p. Inversement pour tout $\mathbf{u}' \in [t^{p,p}]$, $h_p(\mathbf{u}') \neq 0$. Nous avons donc $h_p(\mathbf{u}) = 0$. Si $t^{p,p}$ est utilisé pour déterminer la valeur de $t^{p,q}$, nous aurons donc $h(t^{p,q}) \neq 0$, et donc $h_q(\mathbf{u}) \neq 0$. La subdivision préserve le filtrage, donc $h_p(\mathbf{u}) = h_q(\mathbf{u})$. Nous avons donc une contradiction et $\mathbf{u}$ ne peut pas exister.

Utilisation de la texture de correspondance

Étant donné une texture fournie par l'utilisateur pour notre atlas, la texture de correspondances indique comment la corriger pour garantir un placage sans couture apparente. Pour un texel $t^{p,p}$, la carte fournit la résolution ρ_p du texel, la position $\mathbf{p}$ de sa classe d'équivalence, et la résolution ρ_q de la carte contenant cette position. Plusieurs cas de figure sont possibles :

$\rho_p = \rho_q$ **:** dans ce cas, le triangle C_q contenant la position de classe est à la même résolution que le texel, et $\mathbf{p}$ correspond à la position d'un texel $t^{q,q}$. Dans ce cas, la valeur $h(t^{q,q})$ est assignée à $t^{p,p}$.

$\rho_p < \rho_q$ **:** ici, la résolution du triangle C_q contenant $\mathbf{p}$ est plus élevée et le texel de référence de la classe a donc été remplacé par des texels de plus haute résolution. Dans ce cas, nous récupérons la valeur $h(\mathbf{p})$ calculée par le mode de filtrage choisi à la position de classe puis assignons cette valeur à $t^{p,p}$. Il s'agit ici d'un choix de notre part : n'importe quelle valeur peut être assignée à la classe sans que des coutures apparaissent. Ce cas est un peu particulier, car dans la plupart des cas, aucun des autres texels de la classe $c(t^{p,p})$ n'est positionné à l'intérieur d'une carte, il n'est donc pas possible d'éditer la valeurs des texels de cette classe par un outil de peinture classique, car aucun point de la surface n'a des coordonnées paramétriques correspondant à la position d'un texel de la classe.

$\rho_p > \rho_q$ **:** enfin, lorsque la résolution du triangle C_p contenant le texel est plus fine que celle du triangle C_q contenant la position de sa classe, l'ensemble des texels utiles au point $\mathbf{p}$ est déterminé, et nous appliquons $\rho_p - \rho_q$ étapes de subdivision sur cet ensemble de texels. Parmi les texels issus de la subdivision, un texel $t^{q,p}$ aura la position de la classe et sa valeur sera donnée à $t^{p,p}$.

Nous voyons ici que l'ordre dans lequel les contraintes sont appliquées a une incidence sur le résultat. En effet, dans le cas $\rho_p < \rho_q$, la valeur est récupérée à partir de texels de résolution plus faibles. A contrario, dans le cas $\rho_p > \rho_q$, la valeur de texels fins est calculée à partir de texels grossiers. Pour garantir un placage sans couture visible, il est nécessaire de corriger les texels par ordre de résolution croissante. De cette façon, les texels grossiers obtiennent une valeur qui élimine les discontinuités entre cartes de même résolution, puis les texels plus fins déduisent leur valeur de ces texels grossiers pour assurer la continuité sur les arêtes entre résolutions différentes.

Indépendance du choix du texel de référence dans une carte plus grossière

Lors de la construction des classes d'équivalences, lorsqu'une classe contient au moins un texel $t^{q,p}$ tel que $\rho_q < \rho_p$, nous choisissons parmi les texels vérifiant cette propriété un texel quelconque comme texel de référence pour la classe. Nous allons désormais montrer qu'étant donné le processus de correction de la texture que nous appliquons, le choix de ce texel n'a pas d'incidence sur le résultat.

Théorème 2.1. *Soient $t^{q,p}$ et $t^{r,p}$ deux texels équivalents, tels que $\rho_q < \rho_p$ et $\rho_r < \rho_p$. Nous avons :*

$$h(t^{q,p}) = h(t^{r,p})$$

Démonstration. Sans perte de généralité, nous supposerons que $\rho_r \leq \rho_q$. Nous allons montrer ce résultat par récurrence forte sur la résolution ρ_p. Supposons que pour toute résolution $\rho < \rho_p$, le théorème soit vrai.

Par définition de l'équivalence de $t^{q,p}$ et $t^{r,p}$, il existe une suite de texels $t^{0,p},\ldots,t^{n,p} \in \mathbb{T}_0^p \times \ldots \times \mathbb{T}_n^p$ tels que $t^{0,p} = t^{q,p}$, $t^{n,p} = t^{r,p}$ et pour tout $k \in [0,n-1]$, nous avons $t^{k,p} \in \mathbb{T}_{k,k+1}^p$ et $\mathbf{p}(t^{k+1,p}) = \tau_{k,k+1}(\mathbf{p}(t^{k,p}))$ (voir la Définition 2.1). Soit $t^{q,q}$ un texel utilisé pour le calcul de $h(t^{q,p})$. Nous avons donc $[t^{q,p}] \subset [t^{q,q}]$, et ainsi $t^{q,q} \in \mathbb{T}_{0,1}^q$. Soit $t^{1,q}$ tel que $\mathbf{p}(t^{1,q}) = \tau_{0,1}(\mathbf{p}(t^{q,q}))$. Étant donné que $\mathbf{p}(t^{1,p}) = \tau_{0,1}(\mathbf{p}(t^{q,p}))$, nous avons $\left[t^{1,p}\right] \subset \left[t^{1,q}\right]$. En itérant ce procédé, nous pouvons construire une suite de texels $t^{1,q},\ldots,t^{n,q} \in \mathbb{T}_0^q \times \ldots \times \mathbb{T}_n^q$ telle que $t^{n,q} \in \mathbb{T}_r^q$. Posons donc $t^{r,q} = t^{n,q}$. Plus particulièrement, nous avons également $[t^{r,p}] \subset [t^{r,q}]$, et par définition des classes d'équivalence, $t^{q,q}$ et $t^{r,q}$ sont équivalents. Nous avons alors deux cas :

$\rho_q = \rho_r$: dans ce cas, $t^{r,q}$ correspond également à un texel fourni par l'utilisateur, et comme $[t^{r,p}] \subset [t^{r,q}]$, sa valeur est utilisée pour calculer $h(t^{r,p})$. Les texels subdivisés pour calculer $h(t^{q,p})$ et $h(t^{r,p})$ sont donc deux à deux équivalents, et leur résolution étant plus faible que ρ_p, leurs valeurs seront donc égales au moment du calcul de $h(t^{q,p})$ et $h(t^{r,p})$. Ces deux valeurs sont donc calculées en subdivisant des texels de même valeur, elles sont alors égales ;

$\rho_q > \rho_r$: ici, $t^{r,q}$ est donc un texel de résolution plus forte que la résolution ρ_r de sa carte, et sa valeur est donc calculée par subdivision à partir des texels de $\mathbb{T}_r^r$. Soit $t^{s,q}$ le texel de référence choisi pour la classe de $t^{q,q}$ et $t^{r,q}$. Comme $\rho_q < \rho_r$, le procédé de sélection du texel de référence a choisi un texel de résolution plus faible que ρ_q, et donc $\rho_s < \rho_q$. Nous avons $\rho_q < \rho_p$ donc par hypothèse de récurrence, $h(t^{s,q}) = h(t^{r,q})$. Étant donné que la texture est corrigée par résolution croissante, lors du calcul de $h(t^{q,p})$ et $h(t^{r,p})$, nous aurons donc $h(t^{q,q}) = h(t^{r,q})$. Les texels utilisés pour calculer $h(t^{q,p})$ et les texels issus de $\rho_q - \rho_r$ subdivisions des texels de $\mathbb{T}_r^r$ utilisés pour calculer $h(t^{r,p})$ sont donc deux à deux équivalents et donc de même valeur, et en appliquant

les $\rho_p - \rho_q$ étapes de subdivisions restantes, nous obtenons les mêmes valeurs pour $h(t^{q,p})$ et $h(t^{r,p})$.

La récurrence peut être initialisée au rang $\rho_p = 1$, car dans ce cas, la résolution étant minorée par zéro, nous avons $\rho_q = \rho_r = 0$, et ce cas n'utilise pas l'hypothèse de récurrence dans la démonstration précédente. Ainsi, nous avons bien $h(t^{q,p}) = h(t^{r,p})$

□

2.6.3 Résultats

La première utilité de la résolution adaptative est de pouvoir augmenter localement la résolution de la texture comme réalisé sur la Figure 2.22. Les portions pour lesquelles plus de résolution est demandée occuperont ainsi plus de place dans l'atlas et se verront donc allouer plus de texels. Il est également possible d'utiliser la résolution adaptative pour compenser la distorsion d'échelle présente dans la paramétrisation préservant une grille fournie. La Figure 2.23 en fournit un exemple. Il est cependant nécessaire que la distorsion soit isotrope pour pouvoir la compenser. En effet, l'augmentation de résolution se fait en multipliant les coordonnées paramétriques de la carte par deux. La résolution est donc amplifiée uniformément le long des deux axes des paramètres u et v. Si la distorsion est anisotrope, l'anisotropie reste donc préservée.

Le temps de calcul de la paramétrisation préservant la grille $\mathbb{Z}^2$ reste le même que précédemment. En ce qui concerne le calcul des classes d'équivalence, le calcul est réalise une fois pas niveau de résolution. Ainsi le modèle de grenouille présenté sur la Figure 2.22 n'a que quelques niveaux et nécessite 1.7 secondes. Le modèle de la main (Figure 2.23) contient beaucoup plus de niveaux, et demande 8.3 secondes.

2.7 Contribution : gestion du mipmapping

Notre approche permet d'assurer la continuité du placage à travers le filtrage trilinéaire du mipmapping pour tous les cas pratiques. Comme Burley et Lacewell [11], nous n'assurons pas la continuité pour les niveaux les plus grossiers. En pratique, ces niveaux ne sont utilisés que lorsque l'objet texturé est très loin du point de vue, et n'occupe que quelques pixels du rendu. À ce niveau, assurer la continuité du placage n'a donc que peu d'intérêt, car ces discontinuités sont de l'ordre de la taille d'un pixel, et donc du même ordre que l'échantillonnage de l'image, qui de toute façon est discontinu.

2.7.1 Prise en compte du mipmapping pour générer l'atlas

La difficulté du mipmapping est qu'il est nécessaire de pouvoir assurer la continuité sur des textures de résolutions différentes, tout en gardant le même atlas. Comme dans le cas de la résolution adaptative, chaque niveau de résolution introduit un facteur deux sur les coordonnées paramétriques. Dans le cas du mipmapping, comme expliqué en Section 2.3.3, le but est de diminuer la résolution de la texture à chaque niveau jusqu'à atteindre un unique pixel pour toute la surface.

Diminution de la résolution et préservation de la grille

Pour diminuer la résolution de l'atlas, il est nécessaire de diviser par deux les coordonnées paramétriques de tous les sommets de la surface. Malheureusement, si multiplier les coordonnées d'une paramétrisation préservant la grille $\mathbb{Z}^2$ donne une nouvelle paramétrisation

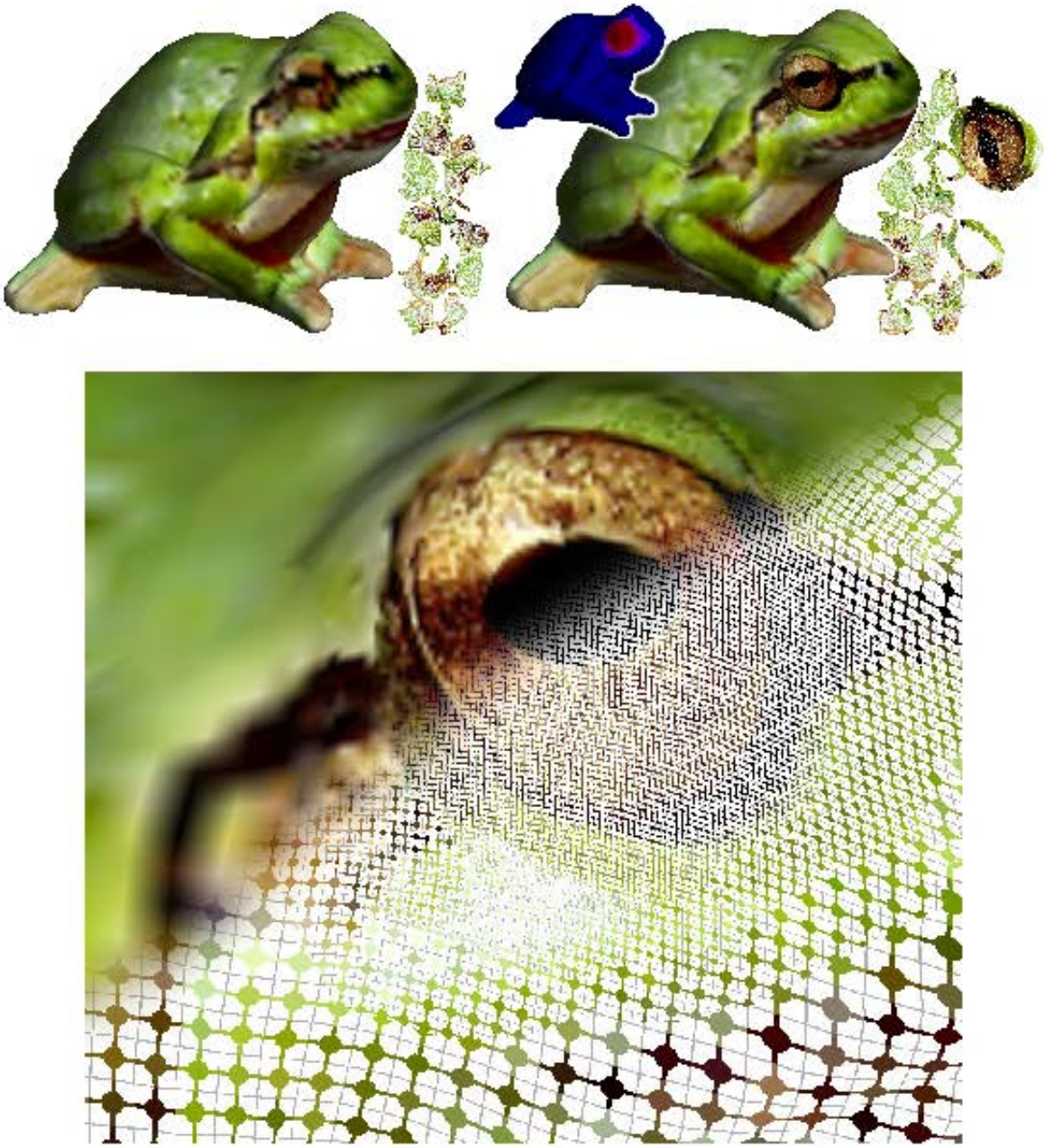


FIGURE 2.22 – Placage à résolution adaptative. Au centre, la résolution de la texture a été localement augmentée au niveau de l'œil de la grenouille pour augmenter la résolution par rapport à la résolution uniforme (à gauche).

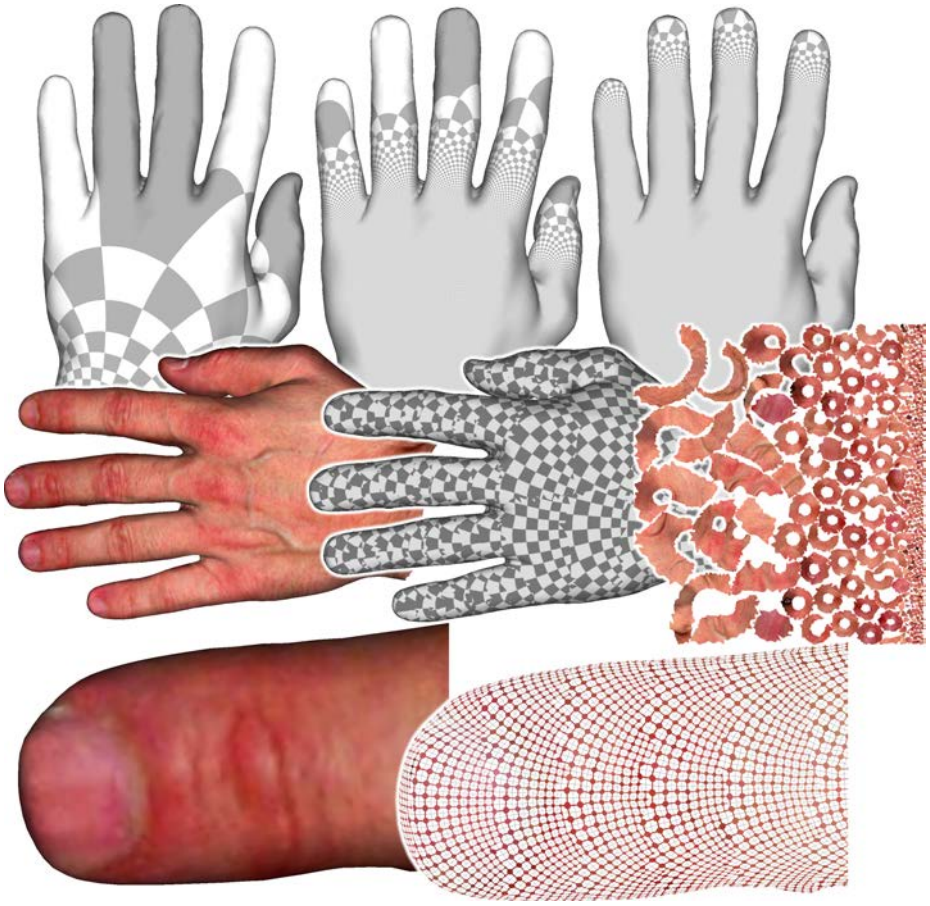


FIGURE 2.23 – Compensation de la distorsion d'échelle de la paramétrisation par un placage de texture à résolution adaptative. Ici, aucune transition n'est introduite dans la paramétrisation préservant une grille produite, qui considère la main comme un disque, et est dont très distendue au niveau des doigts (en haut). Notre technique permet de créer un atlas ayant une résolution adaptative, permettant de compenser la distorsion de la paramétrisation et d'avoir une résolution constante sur la surface, sans que des discontinuités apparaissent entre les différentes résolutions.

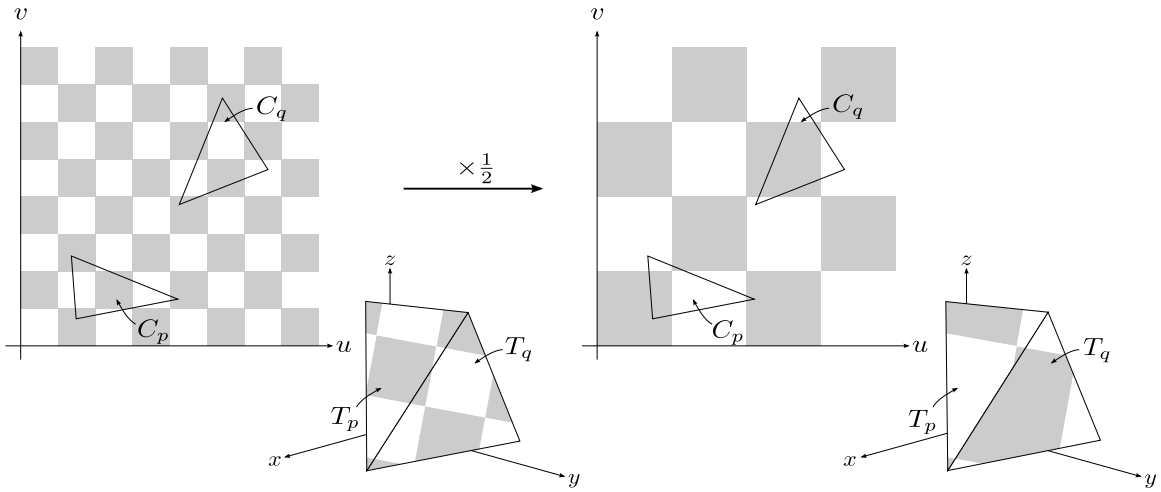


FIGURE 2.24 – Division des coordonnées paramétriques d'une paramétrisation préservant la grille $\mathbb{Z}^2$ initiale (à gauche). Le résultat (à droite) ne préserve plus $\mathbb{Z}^2$.

préservant cette grille, ce n'est plus vrai en divisant les coordonnées, comme le montre la Figure 2.24. C'est pour cette raison que nous sommes incapables d'assurer la continuité du placage pour les niveaux les plus bas.

La paramétrisation préservant une grille régit donc le niveau le plus bas pour lequel nous assurerons la continuité. En dessous de ce niveau, l'interpolation fonctionnera toujours, mais elle aura des discontinuités. Nous générons donc une paramétrisation de faible résolution préservant la grille $\mathbb{Z}^2$, et générerons les niveaux plus fins en multipliant à chaque niveau les coordonnées paramétrique de cette paramétrisation par 2.

Construction de l'atlas et correction d'une texture donnée

Le même atlas doit être utilisé pour tous les niveaux. Nous traitons donc d'abord la résolution la plus faible gérée, en construisant l'atlas et la texture de correspondance comme dans le cas classique. Pour les niveaux de résolution plus élevés par contre, nous ne repartons pas d'une carte individuelle par triangle de la surface, mais utilisons les cartes générées pour l'atlas de la résolution la plus faible. La seule différence est que dans ce cas, toutes les arêtes ne sont pas des transitions, et que seules les transitions restantes sont considérées pour fusionner les classes d'équivalence des texels.

Nous construisons ainsi un ensemble de classes d'équivalence par niveau, et une texture de correspondance pour donner les corrections à effectuer sur une texture de la résolution appropriés. Étant donné la texture fournie par l'utilisateur, la pyramide de textures plus faible est construite, en divisant par deux à chaque niveau la résolution de la texture. Chaque niveau est ensuite corrigé à l'aide de la texture de correspondance appropriée, et ces textures sont prêtes à être fournies comme les différents niveaux de mipmap de la texture initiale. Comme précédemment, il n'est pas nécessaire de reprogrammer l'accès à la texture, et l'interpolation trilinéaire n'a pas de discontinuités tant que les niveaux grossiers non gérés ne sont pas utilisés. Lorsque ces niveaux apparaissent, des discontinuités seront présentes, mais elles ne seront pas perceptibles en pratique, comme le montre la Figure 2.25.

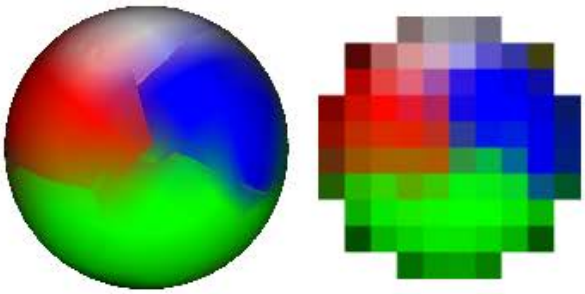


FIGURE 2.25 – Des discontinuités apparaissant (à gauche) dans le cas du mipmapping pour les niveaux de faible résolution. Cependant, la résolution à laquelle l'objet est dessiné lorsque ce niveau est accédé rend ces discontinuités négligeables (à droite).

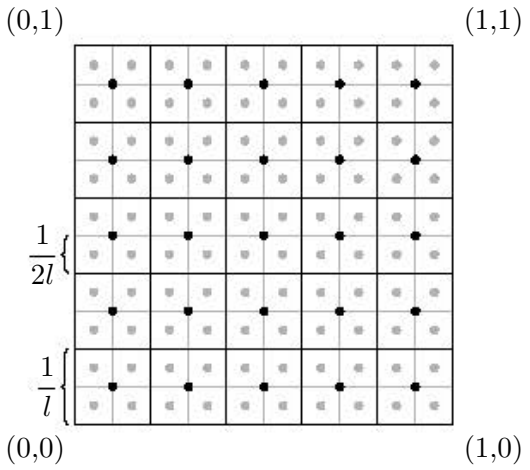


FIGURE 2.26 – Subdivision duale des grilles de texels entre deux niveaux de résolution successifs de la pyramide de mipmaps. Les positions des texels de la résolution la plus fine (en gris) ne réutilisent pas les positions des texels de la résolution la plus grossière (en noir), d'où la dualité de la subdivision.

2.7.2 Incompatibilité avec la résolution adaptative

Nous expliquons ici les raisons qui font qu'il n'est pas possible d'utiliser à la fois la résolution adaptative en filtrage bilinéaire (le plus courant) et le mipmapping sans avoir à reprogrammer l'accès à la texture. Comme nous le verrons, il est possible de rendre les deux mécanismes compatibles en ajoutant un décalage au coordonnées paramétriques auxquelles un accès texture est réalisé. Nous nous restreignons ici au filtrage bilinéaire, car c'est le plus courant. Les filtrage biquadratique et bicubique requièrent de toute façon de reprogrammer l'accès car ils ne sont pas nativement supportés par les cartes graphiques.

Subdivision duale du mipmapping

Lorsque le placage de texture est réalisé sur la carte graphique, les coordonnées paramétriques sont normalisées. Cette normalisation ramène toutes les coordonnées paramétriques dans le domaine $[0,1]^2$. Étant donné une texture carrée de l texels de côté, la norme openGL précise que ces texels seront positionnés sur la grille :

$$\frac{1}{l}\left(\mathbb{Z}^2 + \left(\tfrac{1}{2}, \tfrac{1}{2}\right)\right).$$

Cette convention implique que si nous considérons deux textures correspondant à deux niveaux successifs de la pyramide de mipmaps, si l est la largeur en texels de la plus grossière, la plus fine a donc $2l$ texels de large. En examinant les deux grilles de texels conformes à la norme openGL dans le carré $[0,1]^2$, la subdivision des texels du niveau grossier pour produire les texels du niveau fin est duale, comme le montre la Figure 2.26.

Incompatibilité entre subdivisions duale et primale

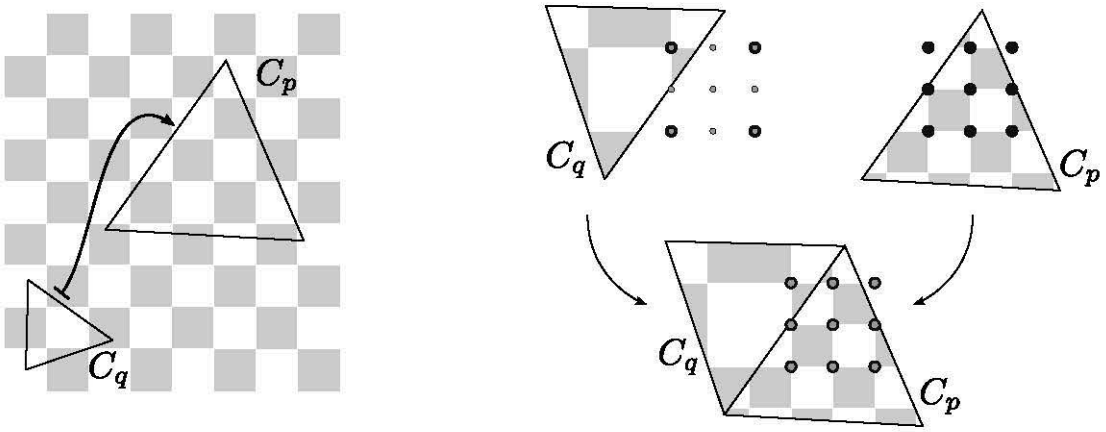
Pour assurer un recollement sans transitions entre deux faces T_p et T_q de résolutions $\rho_p < \rho_q$, il est nécessaire que la subdivision des texels de T_p fournisse une grille dont les texels s'alignent avec ceux de T_q. La subdivision utilisée est celle du filtrage, qui dans le cas du filtrage bilinéaire est une subdivision primale. Ainsi, si les texels sont alignés correctement entre deux cartes de résolutions différentes pour un niveau de mipmap, la subdivision duale du mipmapping pour obtenir le niveau suivant n'est pas compatible avec la subdivision primale utilisée pour le filtrage bilinéaire, et au niveau de mipmap suivant, les texels ne sont plus alignés, comme le montre la Figure 2.27.

Ainsi, pour pouvoir assurer un placage sans couture visible lors du mipmapping lorsque l'atlas comporte des cartes de résolutions différentes, il est nécessaire de modifier l'accès à la texture pour que la subdivision effectuée entre les différents niveaux de mipmaps corresponde à la subdivision du mode de filtrage désiré. Dans le cas du filtrage bilinéaire, il suffit donc de décaler les coordonnées paramétriques d'un demi texel à chaque niveau par rapport au précédent, comme illustré en Figure 2.27. Cette correction est assez classique, et a déjà été proposée par Purnomo *et al.* [90]. La partie compliquée de cette modification réside dans le fait que l'amplitude du décalage dépend du niveau dans lequel l'accès est réalisé, car les coordonnées sont normalisées dans l'intervalle [0,1]. Ainsi une cellule de filtrage n'a pas la même taille dans ce repère selon le niveau de résolution.

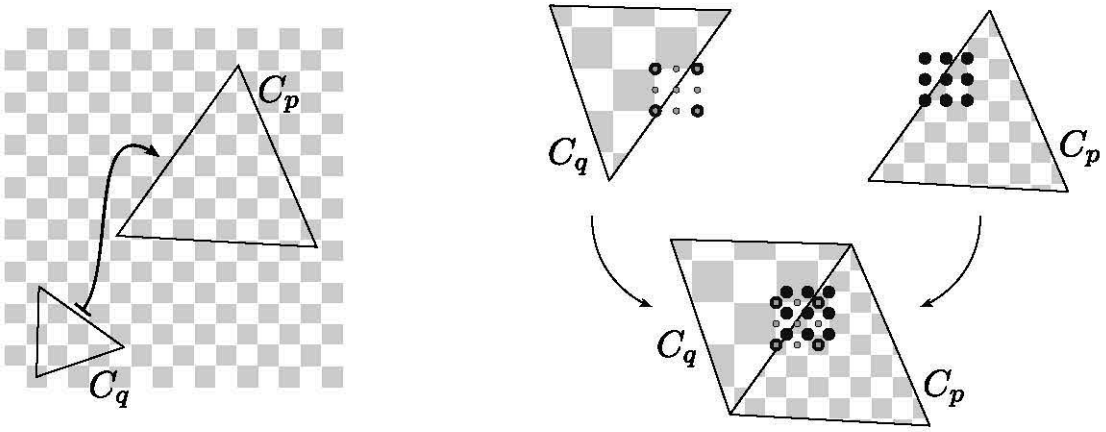
2.7.3 Résultats

Nous montrons tout d'abord sur la Figure 2.28 quatre niveaux de la pyramide de mipmaps plaqué sans couture apparente. Ces quatre niveaux n'ayant pas de coutures apparentes, l'interpolation trilinéaire réalisée par le mipmapping n'introduit pas de discontinuités supplémentaires. Comme le montre la Figure 2.29, il s'agit bien du même atlas qui est utilisé pour tous les niveaux, seule la résolution de la texture change. Au niveau de la résolution la plus forte, l'espace inutilisé dans l'atlas entre les cartes est légèrement plus important que pour un atlas à la même résolution ne gérant pas le mipmapping. Il n'est pas possible de diminuer cet espace sans contraindre des texels supplémentaires à partager leur valeur à la résolution la plus faible, alors que les cartes correspondantes ne sont pas forcément adjacentes sur la surface.

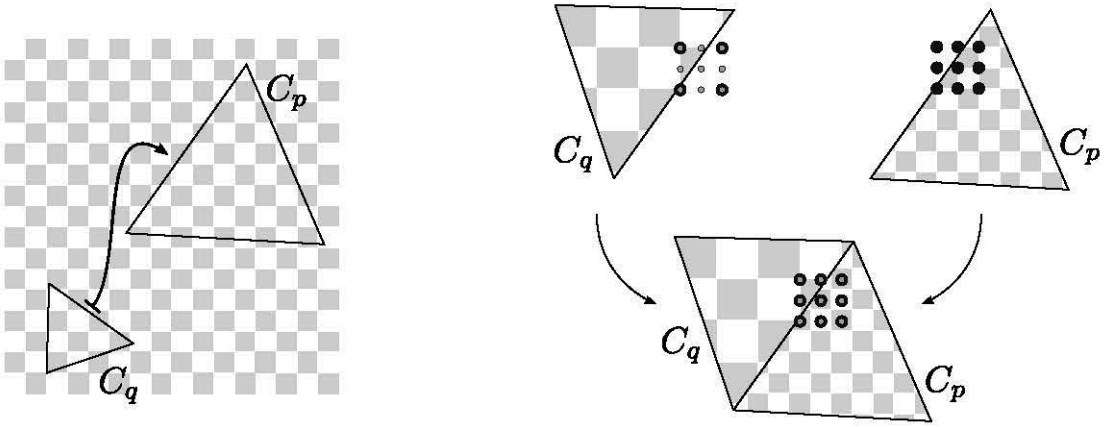
Du point de vue de l'espace texture utilisé, l'atlas est construit au niveau de la résolution de mipmap la plus grossière supportée. L'espace entre les cartes est donc calculé par rapport à la taille des cellules de filtrage à cette résolution. Ainsi, à une résolution plus élevée, l'espace vide entre les cartes est légèrement plus important que pour un atlas généré pour cette résolution sans se soucier du mipmapping. Cette espace reste néanmoins l'espace minimal requis pour que des artéfacts du type de celui illustré en Figure 2.11 ne surviennent à la résolution plus grossière. Ainsi, n'importe quel atlas, même un atlas classique, voulant éviter ce genre d'artéfact devra introduire un espacement similaire entre les cartes dans l'atlas.



(a) Atlas initial



(b) Subdivision duale des texels



(c) Décalage d'un demi texel pour que la subdivision soit primale

FIGURE 2.27 – Relation entre subdivision du mipmapping et subdivision du mode de filtrage. L'atlas initial (a) est généré pour assurer un placage sans couture apparente au niveau de mipmap le plus grossier supporté. Ici, deux triangles C_p et C_q adjacents sont de résolution différentes, et en subdivisant les texels de C_q, nous pouvons les aligner avec ceux de C_p en collant bord à bord les triangles. Au niveau de mipmap de résolution immédiatement supérieur (b), les texels sont subdivisés de manière duale et les texels des triangles C_p et C_q ne peuvent plus être alignés. En décalant les coordonnées paramétriques d'un demi texel (c), il est de nouveau possible d'aligner les texels de C_p et C_q.

FIGURE 2.28 – Placage sans couture apparente de différents niveaux de mipmaps. En utilisant le même atlas, il est possible de plaquer tous les niveaux de mipmaps sans couture apparente. Les résolutions plus grossières laisseront par contre les coutures apparaître. Ici les textures sont de résolution 32×32 jusqu'à 2048×2048, de gauche à droite. Il reste donc quatre niveaux plus grossiers qui ne sont pas gérés.

FIGURE 2.29 – Trois niveaux de Mipmaps, du plus fin à gauche au plus grossier à droite. L'atlas reste le même pour les trois niveaux, mais la résolution de la texture change.

2.8 Conclusion et perspectives

2.8.1 Contributions

La méthode que nous avons présentée dans ce chapitre permet de réaliser un placage de texture sans couture apparente. Elle permet d'augmenter localement la résolution de la texture et supporte la technique du mipmapping. Elle nécessite de fournir en entrée une paramétrisation préservant une grille, ce qui peut être compliqué à calculer, et fait que l'utilisateur ne peut pas éditer à la main l'atlas produit, ni fournir lui-même un atlas. En contrepartie, l'utilisation d'une telle paramétrisation permet le placage sans coutures avec un simple atlas. N'importe quel moteur de rendu existant et supportant le placage de textures peut donc en bénéficier sans la moindre modification. De plus l'accès à la texture reste donc le plus simple possible, et n'introduit pas de coût supplémentaire, ce qui permet d'être moins limité par les accès à la texture dans les programmes spécifiques lancés sur la carte graphique. Une partie des méthodes de placage de textures sans couture visible repose en réalité sur une paramétrisation préservant une grille. C'est notamment le cas de la méthode de Burley et Lacewell [11], qui utilisent une texture par face. La paramétrisation préservant une grille correspondante consiste simplement à paramétrer chaque face quadrangulaire sur le carré unité. Par rapport à ces méthodes, l'utilisation d'une paramétrisation préservant une grille plus générique permet de limiter la distorsion introduite par la paramétrisation.

Il reste pour l'instant impossible d'utiliser la résolution adaptative tout en prévenant les discontinuités issues du mipmapping. Cet inconvénient n'empêche pas d'utiliser le mipmapping, mais le résultat comportera alors des discontinuités. Pour supporter à la fois la résolution adaptative et le mipmapping, il devient alors nécessaire de modifier l'accès à la texture, et donc de reprogrammer la carte, comme le proposent Purnomo *et al.* [90].

2.8.2 Perspectives

Nous avons utilisé une paramétrisation préservant une grille pour définir une grille de texels propre sur la surface, et identifier des texels entre différentes cartes. Il est également possible d'utiliser la structure de grille pour définir des voisinages entre texels. En utilisant ces voisinages, il serait envisageable de réaliser des simulations de type automates cellulaires, qui peuvent par exemple servir à simuler des impacts ou des coulures sur une surface dans les jeux vidéo.

Une autre application de ces voisinages entre texels serait de généraliser les méthodes de génération procédurales de texture à partir d'exemples, pour pouvoir directement générer une texture sans couture dans notre atlas, sans que les transitions ne perturbent la cohérence de la texture.

Enfin, il semble qu'il soit possible de gérer d'autres modes de filtrage, tant que le motif correspondant à l'ensemble des texels utiles en un point reste le même partout. Il serait donc en particulier envisageable de traiter le filtrage anisotrope de la même façon que le mipmapping. La différence est que cette fois, lorsque les deux arêtes correspondant à une transition ne sont pas parallèles dans le domaine paramétrique, le filtrage anisotrope ne récupèrera pas forcément la valeur dans la même texture pour les deux triangles adjacents, car une anisotropie en u pour un triangle correspond à une anisotropie en v pour le triangle voisin. Les classes d'équivalence entre texels devraient donc s'étendre sur plusieurs textures de niveaux et anisotropies différents.

Deuxième partie

Échantillonnage par un diagramme de Voronoï restreint

Chapitre 3

Diagramme de Voronoï restreint

3.1 Motivations

Les diagrammes de Voronoï sont une structure fondamentale en géométrie, que l'on retrouve dans de très nombreuses applications. Étant donnée une ville par exemple, munie d'un certain nombre de casernes de pompiers, le problème consiste à déterminer, lors d'un incendie, quelle caserne pourra envoyer le plus rapidement possible des agents sur les lieux pour intervenir. Il s'agit donc d'attribuer à chaque caserne un secteur dont elle sera responsable. Cette répartition constitue une *partition*, car tout point de la ville doit être couvert par une caserne. Cette caserne est déterminée comme étant la caserne la plus proche. La partition utilise donc une notion de *distance* entre deux points dans la ville. Plus formellement, étant donné un domaine E (la ville) muni d'une distance $d : E \times E \to \mathbb{R}$ qui pour deux points $\mathbf{v}_1$ et $\mathbf{v}_2$ du domaine (221b Baker Street, 10 Downing Street) fournit la distance à parcourir $d(\mathbf{v}_1,\mathbf{v}_2)$ pour se rendre de l'un à l'autre (3,9 km), étant donné un ensemble de *sites* $\mathbf{V} = \{\mathbf{v}_k\}_{k=1}^n$ (les casernes), le diagramme de Voronoï associe à chaque site $\mathbf{v}_k$ une région Ω_k (la zone de responsabilité de la caserne) telle que :

$$\Omega_k = \{\mathbf{x} \in E, d(\mathbf{v}_k,\mathbf{x}) \leq d(\mathbf{v}_\ell,\mathbf{x}) \text{ pour tout } \ell \neq k\}$$

autrement dit l'ensemble des points $\mathbf{x}$ de la ville pour lesquels la caserne $\mathbf{v}_k$ est la plus proche. Un exemple de diagramme de Voronoï pour $E = \mathbb{R}^2$ avec d la distance euclidienne usuelle est donné en Figure 3.1. Avec cette distance, les cellules de Voronoï sont des polygones convexes.

Les diagrammes de Voronoï apparaissent spontanément dans de nombreux problèmes. Okabe *et al.* [76], Fortune [42] dénombrent un grand nombre d'applications dans lesquelles

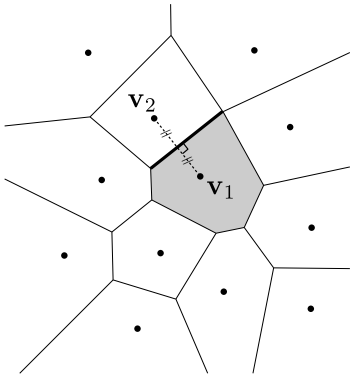
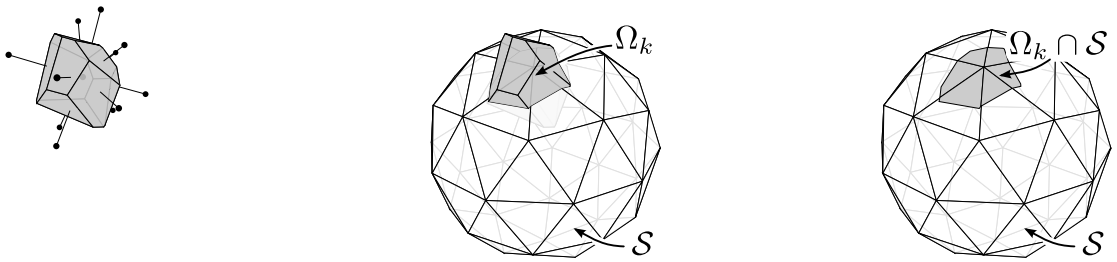


FIGURE 3.1 – Diagramme de Voronoï dans le plan en utilisant la distance euclidienne. La zone grisée est la cellule de Voronoï du site $\mathbf{v}_1$. Les cellules de Voronoï dans ce cas sont des polygones convexes, dont les côtés sont les médiatrices de couples de sites.



(a) cellule de Voronoï (b) cellule et surface (c) cellule restreinte

FIGURE 3.2 – Diagramme de Voronoï utilisant la distance euclidienne dans $\mathbb{R}^3$ restreint à une surface $\mathcal{S}$. Dans $\mathbb{R}^3$, les cellules de Voronoï sont des polyèdres dont les faces sont les plans bissecteurs de segments reliant des couples de sites.

ils interviennent. Dans le cadre de cette thèse, nous nous intéresserons en Chapitre 4 à des problèmes d'optimisation formulés grâce à ces diagrammes. Par rapport à l'exemple précédent des casernes et des villes, un tel problème d'optimisation correspondrait par exemple à chercher à disposer les casernes afin de minimiser le temps d'intervention moyen (ou maximal) en cas d'incendie. Ce problème se ramène donc à la minimisation d'une fonction objectif $\mathscr{F}$, qui a comme variables les positions de tous les sites, et qui à partir de ces positions calcule le temps d'intervention moyen, à partir du diagramme de Voronoï des sites. Pour résoudre un tel problème, plusieurs facteurs sont déterminants. Tout d'abord, les méthodes les plus efficaces requièrent que la fonction objectif $\mathscr{F}$ soit la plus lisse possible, c'est-à-dire que sa valeur ne variera pas brusquement entre deux ensembles de sites semblables. Ensuite, ces méthodes demandent d'être capable d'évaluer un grand nombre de fois $\mathscr{F}$, et donc que le calcul du diagramme de Voronoï des sites soit rapide. Toujours dans le cadre de notre exemple de la répartition des casernes, le calcul de la distance entre deux points d'une ville est un problème complexe. En effet le véhicule des pompiers doit emprunter les rues, qui sont parfois en sens unique, et seront plus ou moins encombrées selon l'heure de l'intervention. Pour obtenir tout de même un résultat, il est alors possible de simplifier le problème, par exemple en considérant que les distances sont calculées à vol d'oiseau. C'est ce que nous ferons dans ce chapitre.

Nous nous intéressons ici au cas où le domaine sur lequel nous voulons construire un diagramme de Voronoï est une surface $\mathcal{S}$, définie par un maillage, constitué de triangles $\mathbf{T} = \{T_p\}_p$ reliant entre eux des sommets $\mathbf{X} = \{\mathbf{x}_i\}_i$. Les positions de ces points sont définies dans $\mathbb{R}^3$. Étant donné un ensemble de sites disposés sur cette surface, nous cherchons à partitionner la surface en régions de Voronoï associées à ces différents sites. La distance naturelle sur la surface correspond à la distance *géodésique*, à savoir la longueur du plus court chemin sur la surface entre les deux points fournis. Le calcul d'un tel chemin est un problème complexe, c'est pourquoi nous remplaçons dans nos travaux cette distance par la distance euclidienne entre deux points de $\mathbb{R}^3$. Cette simplification revient à considérer qu'il nous est possible de voler (ou creuser) en ligne droite pour relier n'importe quelle paire de points. Dans ce cas, la région $\Omega_{k|\mathcal{S}}$ de la surface associée à un site correspond à l'intersection entre la région de Voronoï Ω_k de ce site dans le domaine $\mathbb{R}^3$ complet, et la surface :

$$\Omega_{k|\mathcal{S}} = \Omega_k \cap \mathcal{S}$$

Un exemple d'une telle cellule restreinte est donné sur la Figure 3.2, et un diagramme de Voronoï restreint est illustré en Figure 3.3.

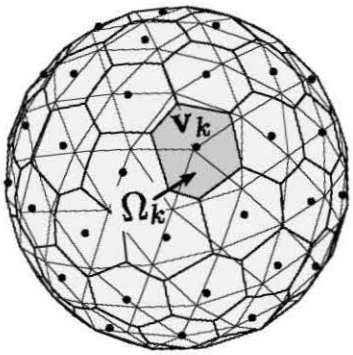


FIGURE 3.3 – Exemple de diagramme de Voronoï restreint d'un ensemble de sites sur une sphère maillée. À chaque site $\mathbf{v}_k$ correspond une région Ω_k.

3.2 Contributions

Dans ce chapitre, nous introduirons deux variations d'un algorithme existant pour le calcul d'un diagramme de Voronoï restreint proposé par Yan *et al.* [118] :

Propagation par les triangles : donnée en Section 3.4, cette variante propose de réordonner les calculs de l'algorithme, pour en améliorer la complexité en temps et en espace lorsque le nombre de sites est important devant le nombre de faces de la surface. Cette première version de l'algorithme permet de calculer le diagramme de Voronoï restreint en considérant les faces de $\mathcal{S}$ une par une, et en les partitionnant entre les différents sites à proximité.

Propagation par les sites : plutôt que d'itérer sur les triangles de la surface, il est possible de se concentrer sur les sites du diagramme de Voronoï, et de calculer les cellules restreintes une par une. Cette variante est donnée en Section 3.5, et permet d'améliorer la complexité en espace nécessaire pour le calcul de certaines fonctions sur les cellules de Voronoï restreintes. Cette seconde variante reste en pratique moins performante que la précédente.

Le diagramme de Voronoï restreint est calculé sous la forme d'un ensemble de faces polygonales, où chaque face est l'intersection entre une face de $\mathcal{S}$ et une cellule de Voronoï. Ces deux algorithmes permettent de l'obtenir avec une complexité $O(mn^2 \log n)$ dans le pire des cas, où m est le nombre de faces de $\mathcal{S}$ et n le nombre de sites. Dans cette complexité, un premier facteur mn correspond au pire cas du nombre de faces du diagramme de Voronoï restreint calculé, et ne peut donc pas être évité. Le facteur $n \log n$ restant correspond à la complexité du calcul de l'intersection entre une face du maillage et une cellule de Voronoï. Dans toutes les expérimentations que nous avons réalisées en utilisant ces algorithmes, cette complexité dans le pire des cas n'est jamais atteinte, et nous avons plutôt observé une complexité de $O(m + n)$.

Ces travaux n'ont à l'heure actuelle pas encore fait l'objet d'une publication.

3.3 Fondements et état de l'art

Nous commencerons par étudier plus en détails la structure d'un diagramme de Voronoï, et comment ces diagrammes sont calculés. Nous avons choisi de travailler avec un diagramme de Voronoï restreint, qui utilise donc la distance euclidienne de $\mathbb{R}^3$ plutôt que la distance géodésique sur la surface. Il existe cependant des méthodes pour calculer des diagrammes de Voronoï en utilisant cette distance géodésique. Nous détaillerons donc ces méthodes, avant d'aborder l'algorithme de Yan *et al.* [118] sur lequel notre approche est fondée.

3.3.1 Notations

Nous récapitulons ici les notations utilisées dans ce chapitre.

$\mathbf{V}$	ensemble des sites du diagramme de Voronoï restreint
$\mathbf{v}_k$	site quelconque de $\mathbf{V}$
Ω_k	cellule de Voronoï de $\mathbf{v}_k$ dans $\mathbb{R}^3$
$\mathcal{S}$	surface sur laquelle calculer le diagramme, décrite par un maillage
$\Omega_{k\|\mathcal{S}}$	cellule de Voronoï de $\mathbf{v}_k$ restreinte à $\mathcal{S}$
$\mathbf{X}$	ensemble des sommets de $\mathcal{S}$
$\mathbf{x}_i$	sommet de $\mathcal{S}$
$\mathbf{T}$	ensemble des faces de $\mathcal{S}$
T_p	face de $\mathcal{S}$
$\mathbf{x}$	point quelconque de $\mathcal{S}$ ou du domaine dans lequel le diagramme est calculé

3.3.2 Diagramme de Voronoï

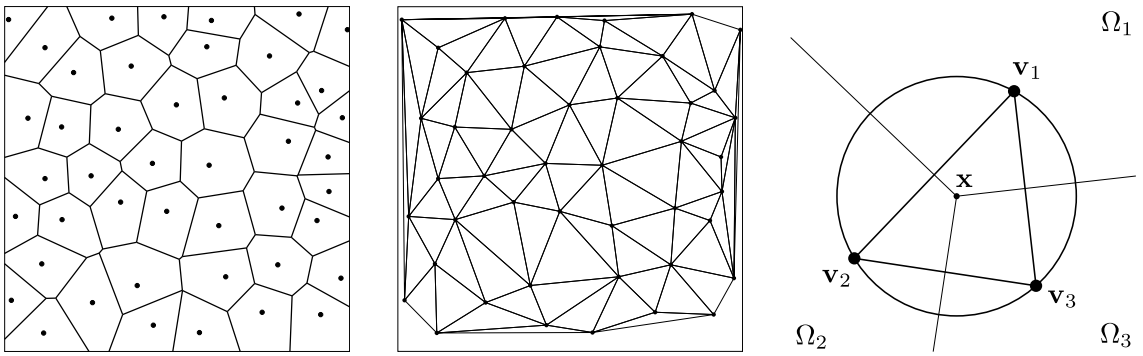
Diagramme de Voronoï et triangulation de Delaunay

Les diagrammes de Voronoï sont une structure géométrique fondamentale pour de très nombreux problèmes ; Aurenhammer [4] en énumère en particulier un certain nombre, et George et Borouchaki [44] détaillent leurs application au problème de la génération de maillages. Dans nos travaux, ils nous intéressent pour évaluer la qualité d'un échantillonnage, en permettant automatiquement d'associer à un échantillon une portion du domaine à échantillonner. Pour définir simplement un diagramme de Voronoï, nous commencerons par nous placer dans le cas où le domaine est $\mathbb{R}^2$ et la distance d est la distance euclidienne. Étant donné un ensemble de sites $\mathbf{V} = \{\mathbf{v}_k\}_{k=1}^n$, le diagramme de Voronoï pour cet ensemble de sites associe à chaque $\mathbf{v}_k$ une région Ω_k définie par :

$$\Omega_k = \{\mathbf{x} \in \mathbb{R}^2, d(\mathbf{v}_k,\mathbf{x}) \leq d(\mathbf{v}_\ell,\mathbf{x}) \text{ pour tout } \ell \neq k\}. \tag{3.3.1}$$

Étant donnée une paire de sites $(\mathbf{v}_1,\mathbf{v}_2) \in \mathbf{V}^2$, l'ensemble des points $\mathbf{x} \in E$ tels que $d(\mathbf{v}_1,\mathbf{x}) \leq d(\mathbf{v}_2,\mathbf{x})$ est un demi-plan délimité par une droite appelée la *médiatrice* du segment $[\mathbf{v}_1,\mathbf{v}_2]$. Cette droite contient l'ensemble des points de $\mathbb{R}^2$ tels que $d(\mathbf{v}_1,\mathbf{x}) = d(\mathbf{v}_2,\mathbf{x})$, et est la droite perpendiculaire à $[\mathbf{v}_1,\mathbf{v}_2]$ passant par le milieu de ce segment. Une méthode simple pour construire la cellule de Voronoï d'un site $\mathbf{v}_k$ consiste donc simplement à tracer les médiatrices de ce site par rapport à tous les autres sites. À la fin de ce procédé, la cellule de Voronoï Ω_k est la portion du plan contenant $\mathbf{v}_k$ délimitée par ces médiatrices. En général, toutes les médiatrices ne seront pas utiles pour définir la cellule de Voronoï. L'ensemble des sites $\mathbf{v}_\ell$ tels que la médiatrice de $[\mathbf{v}_k,\mathbf{v}_\ell]$ borde Ω_k sera appelé l'ensemble des voisins de $\mathbf{v}_k$, et noté $\mathcal{N}(\mathbf{v}_k)$.

Nous l'avons dit, chaque arête de Ω_k correspond à la médiatrice entre $\mathbf{v}_k$ et un site $\mathbf{v}_\ell$ voisin. Chaque sommet $\mathbf{x}$ du polygone correspondant à Ω_k est à l'intersection de deux arêtes et donc deux médiatrices, associées respectivement à deux sites voisins $\mathbf{v}_{\ell_1}$ et $\mathbf{v}_{\ell_2}$. Par définition, ce sommet est donc à égale distance des trois sites $\mathbf{v}_k$, $\mathbf{v}_{\ell_1}$ et $\mathbf{v}_{\ell_2}$. Dans le cas général, un sommet de Voronoï est donc un point où trois cellules se rejoignent. Le triangle formé par trois sites ayant un sommet de Voronoï en commun est appelé un triangle de *Delaunay*. Un diagramme de Voronoï, ainsi que la triangulation de Delaunay associée est illustré sur la Figure 3.4. La triangulation de Delaunay est dite *duale* au diagramme de Voronoï car à chaque région de Voronoï Ω_k correspond un sommet de la triangulation : le



(a) Diagramme de Voronoï (b) Triangulation de Delaunay (c) Dualité

FIGURE 3.4 – Un diagramme de Voronoï et la triangulation de Delaunay associée. La triangulation de Delaunay est duale au diagramme de Voronoï : à la région Ω_1 correspond le sommet $\mathbf{v}_1$, à l'arête entre Ω_1 et Ω_2 correspond le segment $[\mathbf{v}_1,\mathbf{v}_2]$, et au sommet $\mathbf{x}$ commun à Ω_1, Ω_2 et Ω_3 correspond le triangle $(\mathbf{v}_1,\mathbf{v}_2,\mathbf{v}_3)$.

site $\mathbf{v}_k$; à chaque arête de Voronoï correspond une arête de la triangulation : le segment $[\mathbf{v}_k,\mathbf{v}_\ell]$ dont l'arête est la médiatrice ; à chaque sommet de Voronoï correspond un triangle de Delaunay, reliant entre eux les trois sites équidistants à ce sommet.

Un sommet de Voronoï $\mathbf{x}$ est à égale distance de trois sites $\mathbf{v}_1$, $\mathbf{v}_2$ et $\mathbf{v}_3$. Pour que ce sommet existe, aucun autre site $\mathbf{v}_4$ ne doit être plus proche de $\mathbf{x}$ que $\mathbf{v}_1$, $\mathbf{v}_2$ et $\mathbf{v}_3$. Cette condition se traduit par le fait que le cercle centré sur $\mathbf{x}$ et passant par $\mathbf{v}_1$, $\mathbf{v}_2$ et $\mathbf{v}_3$ ne doit pas contenir de site. Cette notion permet de définit la triangulation de Delaunay sans avoir à passer par le diagramme de Voronoï. En effet, à chaque sommet de Voronoï correspond un triangle de Delaunay. Ainsi, un triangle de Delaunay reliera les sites $\mathbf{v}_1$, $\mathbf{v}_2$ et $\mathbf{v}_3$ si et seulement si le cercle circonscrit à ce triangle ne contient pas d'autre site (Figure 3.5). Cette nouvelle définition nous permet également de faire apparaître un cas particulier dans un diagramme de Voronoï. En effet, lorsque quatre sites sont situés sur un même cercle ne contenant pas d'autre site, le quadrilatère qu'ils forment pourrait être triangulé de deux façons, et il ne correspond qu'à un seul sommet de Voronoï. Dans le diagramme de Voronoï, ce sommet est adjacent à quatre cellules de Voronoï. Cette situation est illustrée sur la Figure 3.5.

Ces notions présentées dans le cas du domaine $\mathbb{R}^2$ se généralisent en dimension supérieure. Pour le cas qui nous intéresse, lorsque le domaine est $\mathbb{R}^3$, les médiatrices deviennent les plans *bissecteurs* de couples de sites, et les cellules de Voronoï sont des polyèdres convexes. La triangulation de Delaunay devient une tétraédrisation ; les ambiguïtés surviennent lorsque cinq sites ou plus sont *cosphériques*, ou lorsque quatre sites ou plus sont coplanaires. Le plan bissecteur entre deux sites ainsi qu'une cellule de Voronoï dans $\mathbb{R}^3$ sont illustrés en Figure 3.6.

Calcul d'un diagramme de Voronoï

Pour calculer un diagramme de Voronoï, selon l'application, plusieurs méthodes peuvent être utilisées. Tout d'abord, il est possible de calculer le diagramme de manière approchée sur une grille discrète du domaine, dans $\mathbb{R}^2$ ou $\mathbb{R}^3$. Hoff III *et al.* [49] et Sud *et al.* [105] proposent des méthodes pour calculer le diagramme de Voronoï d'un ensemble de primitives

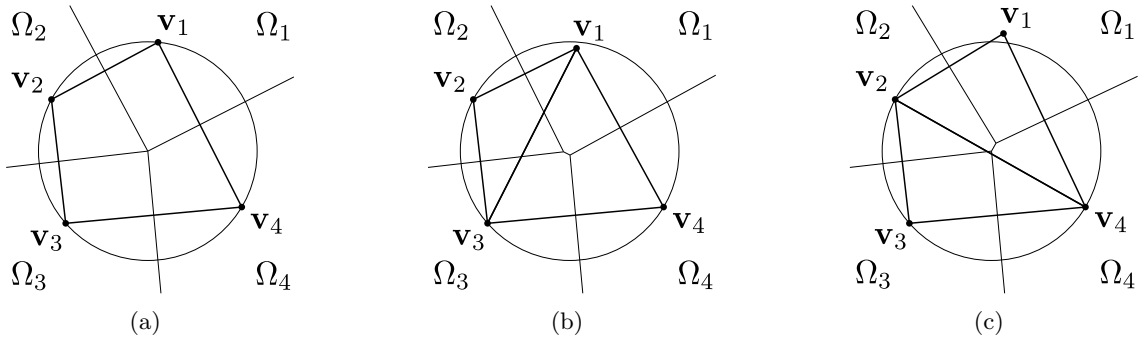


FIGURE 3.5 – Triangulation de Delaunay dans le cas de point cocycliques. Lorsque quatre points sont cocycliques, un quadrilatère apparaît dans la triangulation de Delaunay (a). Une infime perturbation de la position d'un des sites permet d'obtenir deux triangulations différentes, selon que la perturbation amène le site à l'intérieur ou à l'extérieur du cercle circonscrit des trois autres sites. Lorsque $\mathbf{v}_1$ pénètre le cercle circonscrit à $\mathbf{v}_2$, $\mathbf{v}_3$ et $\mathbf{v}_4$ (b), les cellules Ω_2 et Ω_4 ne se touchent plus, contrairement à Ω_1 et Ω_3, et une arête de Delaunay joint les sommets $\mathbf{v}_1$ et $\mathbf{v}_3$. Lorsqu'au contraire $\mathbf{v}_1$ s'éloigne du cercle (c), Ω_1 et Ω_3 ne sont plus en contact, Ω_2 et Ω_4 le sont, et une arête de Delaunay apparaît entre $\mathbf{v}_2$ et $\mathbf{v}_4$.

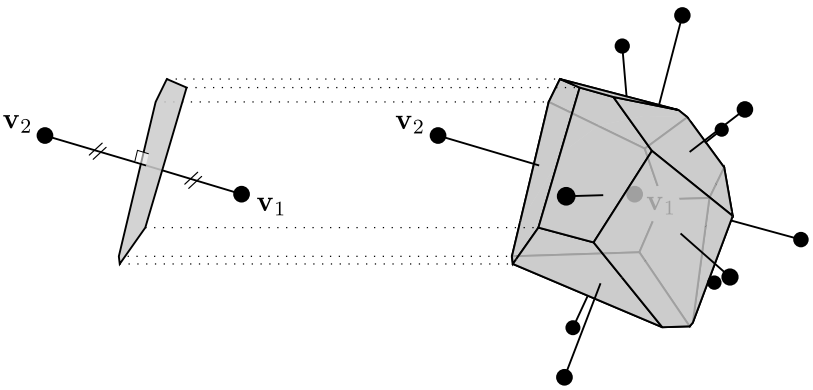


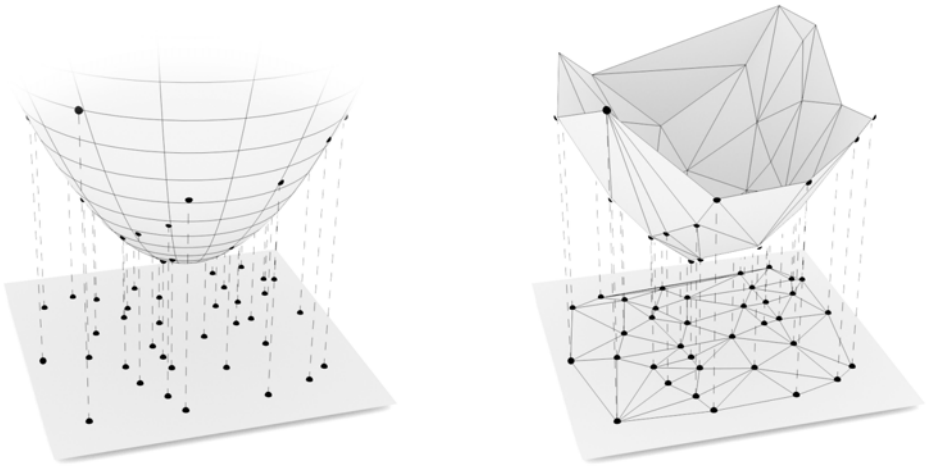
FIGURE 3.6 – Cellule de Voronoï dans $\mathbb{R}^3$. Dans ce cas, les cellules sont des polyèdres dont chaque face est le plan bissecteur d'une paire de sites. Étant donnés deux sites $\mathbf{v}_1$ et $\mathbf{v}_2$, leur plan bissecteur est le plan perpendiculaire au segment $[\mathbf{v}_1,\mathbf{v}_2]$, passant par son milieu.

(points, segments ou polygones) en utilisant une carte graphique. Dans $\mathbb{R}^2$, ces méthodes génèrent un maillage dans $\mathbb{R}^3$ pour représenter le graphe de la fonction $d : \mathbb{R}^2 \to \mathbb{R}$ donnant la distance par rapport à une primitive. À chaque primitive est associé un tel maillage, ainsi qu'une couleur. Un point de vue est ensuite déterminé de telle sorte que pour chaque maillage généré, la profondeur corresponde à la distance à la primitive dont il est issu. En effectuant un rendu selon ce point de vue de tous les maillages simultanément, les couleurs de l'image générée fournissent le diagramme de Voronoï, et le tampon de profondeur contient la distance au site le plus proche. Pour généraliser cette approche à des diagrammes dans $\mathbb{R}^3$, le diagramme est calculé tranche par tranche, en utilisant l'approche dans $\mathbb{R}^2$ pour chaque tranche.

Le calcul exact du diagramme de Voronoï d'un ensemble de sites a également fait l'objet de beaucoup de travaux. Pour les calculer, il est possible d'utiliser les bibliothèques fournies par CGAL [109] et Barber *et al.* [5]. Plusieurs synthèses des algorithmes utilisés existent, nous invitons le lecteur à se référer aux articles de synthèse de Fortune [42] et Okabe *et al.* [76] pour obtenir plus de détails sur les méthodes dont nous parlerons ici. Ce sont ces méthodes qui sont nécessaires dans le cadre de nos travaux.

Tout d'abord, il existe une correspondance forte entre la notion de triangulation de Delaunay, et celle d'enveloppe convexe. Nous rappelons qu'un ensemble est dit convexe si étant donnés deux points quelconques de l'ensemble, le segment les joignant est intégralement inclus dans l'ensemble. L'enveloppe convexe d'un ensemble de points est la plus petite partie convexe – au sens de l'inclusion – contenant l'ensemble des points. De façon remarquable, la triangulation de Delaunay dans $\mathbb{R}^d$ (les « triangles » sont alors des simplexes de dimension d) d'un ensemble de sites $\mathbf{v}_k = (v_{k,1}, \ldots, v_{k,d}) \in \mathbb{R}^d$ se déduit de la projection orthogonale sur $\mathbb{R}^d$ (en ne gardant que les d première coordonnées) de la partie inférieure de l'enveloppe convexe de l'ensemble de points $\mathbf{w}_k = (v_{k,1}, \ldots, v_{k,d}, v_{k,1}^2 + \ldots + v_{k,d}^2) \in \mathbb{R}^{d+1}$. Un point de l'enveloppe convexe est sur la partie inférieure de l'enveloppe si le segment qui le relie à son projeté sur $\mathbb{R}^d$ ne coupe pas l'enveloppe convexe en un autre point. À chaque face de l'enveloppe convexe inférieure correspond un « triangle » de Delaunay. Ce procédé est illustré sur la Figure 3.7, pour calculer une triangulation de Delaunay dans $\mathbb{R}^2$. C'est sur ce principe que fonctionnent les algorithmes de Hermeline [47] et de Barber *et al.* [5]. Les points sont ajoutés un par un, et à chaque nouveau point l'enveloppe convexe est mise à jour. Pour déterminer étant donné un nouveau point si une face de l'enveloppe convexe reste valide, il suffit de calculer de quel côté le point se trouve de l'hyperplan contenant la face. Une face reste valide si tous les points $\mathbf{w}_k$ sont du même côté de cet hyperplan. Pour chaque arête de l'enveloppe séparant une face valide et une face invalide, une nouvelle face est formée avec le point inséré. L'ensemble des faces invalides est quant à lui supprimé.

Cette approche nécessite de travailler en dimension $d+1$ pour calculer une triangulation de Delaunay en dimension d. Le test permettant de déterminer si une face de l'enveloppe convexe est valide ou non étant donné un nouveau point est en réalité équivalent au test réalisé pour déterminer si la sphère circonscrite à un tétraèdre en dimension d contient le site correspondant. L'approche de CGAL [109] consiste donc également à insérer les sites un par un dans la triangulation de Delaunay, et à chaque nouveau site à déterminer l'ensemble des tétraèdres dont la sphère circonscrite contient le point. Ces tétraèdres sont supprimés, ce qui crée une cavité dans la triangulation de Delaunay, et de nouveaux tétraèdres sont formés entre les faces du bord de la cavité et le site inséré. L'avantage de cette approche est qu'elle travaille en dimension d ce qui économise beaucoup d'espace mémoire. L'efficacité



(a) Relèvement sur un paraboloïde (b) Enveloppe convexe inférieure

FIGURE 3.7 – Lien entre le calcul d'une triangulation de Delaunay dans $\mathbb{R}^d$ et celui d'une enveloppe convexe dans $\mathbb{R}^{d+1}$. Ici pour calculer une triangulation de Delaunay dans $\mathbb{R}^2$, les sites sont tout d'abord relevés sur un paraboloïde de $\mathbb{R}^3$ en leur faisant subir la transformation $(x,y) \mapsto (x,y,x^2+y^2)$ (a). En projetant les triangles de l'enveloppe convexe inférieure des sites relevés, la triangulation de Delaunay est obtenue (b).

de ces méthodes peut être optimisée en utilisant une méthode de tri spatial, permettant de faire en sorte que les points insérés soient proches, et ainsi que la recherche des tétraèdres invalides soit plus rapide.

La complexité d'une triangulation de Delaunay, en terme de nombre d'arêtes, est dans le pire des cas de $O(n^{\lceil d/2 \rceil})$. Dans $\mathbb{R}^3$, lorsque la distribution des points est uniforme, la complexité en moyenne tombe à $O(n)$.

3.3.3 Diagramme de Voronoï géodésique

Il existe des méthodes pour approximer ou calculer un diagramme de Voronoï sur une surface utilisant la distance *géodésique*. Cette distance consiste étant donnés deux points $\mathbf{x}_1$ et $\mathbf{x}_2$ de $\mathcal{S}$ à chercher une courbe sur $\mathcal{S}$ de longueur minimale, ayant $\mathbf{x}_1$ et $\mathbf{x}_2$ comme extrémités. La distance entre $\mathbf{x}_1$ et $\mathbf{x}_2$ est alors la longueur de cette courbe. Muni de cette distance, la cellule de Voronoï d'un site $\mathbf{v}_k$ est définie comme précédemment comme l'ensemble des points de $\mathcal{S}$ pour lesquels $\mathbf{v}_k$ est le site le plus proche. Les bissecteurs ne sont par contre plus aussi simples, car selon la géométrie de $\mathcal{S}$, il peut arriver que des régions entières de $\mathcal{S}$ soient équidistantes de deux sites.

Calcul approché

Un diagramme de Voronoï sur une surface est une manière de calculer une partition de cette surface en autant de régions que de sites placés. Nous avons déjà vu en Section 2.4.2 une application de ce problème à la paramétrisation de surfaces, pour découper une surface en un ensemble de régions homéomorphes à des disques, afin de paramétrer ces régions et ainsi créer un atlas de texture. La méthode introduite par Eck *et al.* [32] pour créer un ensemble de régions est une forme d'approximation d'un diagramme de Voronoï géodésique.

Le procédé consiste à partir d'un ensemble de sites, à attribuer itérativement les triangles de la surface à la région de l'un ou l'autre des sites, par un processus de propagation.

Une autre approche pour calculer un diagramme de Voronoï géodésique utilise un processus de propagation sur le maillage, pour calculer le site le plus proche. Peyré et Cohen [81] échantillonnent ainsi la fonction distance au site le plus proche sur les sommets du maillage. Étant donnée une vitesse de propagation définie également pour chaque sommet du maillage, ils établissent des équations en chaque sommet pour faire en sorte que la norme du gradient de la fonction distance corresponde à la vitesse de propagation souhaitée, en approximant le gradient par la différence des valeurs de la distance entre deux sommets voisins. Le diagramme de Voronoï est ainsi construit de façon incrémentale, en introduisant les sites un par un, et en mettant à jour les distances stockées en chaque sommet.

Ces deux méthodes ont l'avantage d'être rapides, mais nécessitent que le maillage ait une résolution bien supérieure à la densité des sites sur la surface pour que l'approximation ne soit pas trop grossière. De plus, par rapport à nos applications, il sera ici difficile d'optimiser le diagramme de Voronoï par rapport à une fonction objectif, car si la fonction objectif peut être rapidement calculée, son gradient par contre sera beaucoup plus dur à déterminer.

Formes closes

Il est également possible de calculer le diagramme de Voronoï de manière exacte sur un maillage, par la méthode de Liu *et al.* [66]. Cette méthode calcule intégralement la combinatoire de la fonction distance à un site donné. Étant donné un site, et un point du maillage où l'on souhaite connaître la distance au site, un chemin pour relier ces deux points passera par une suite de triangles de la surface. En prenant le problème à l'envers, étant donnée une suite de triangles connectés deux à deux, partant du triangle contenant le site, et rejoignant le triangle contenant le point, il est alors possible de déplier ces triangles pour les mettre à plat, et de calculer la distance entre le point et le site dans le polygone obtenu. Pour éviter d'avoir à énumérer toutes les suites de triangles possibles pour relier deux faces du maillage, l'algorithme utilise alors un processus de propagation initialisé à la face contenant le site. Il encode sur les arêtes du maillage une structure enregistrant la distance au site de manière exacte, et les informations nécessaires pour propager ce calcul aux triangles voisins. Il est difficile d'évaluer la complexité d'un tel algorithme, car elle dépend fortement de la combinatoire et de la géométrie du problème. Cependant, les mesures des temps d'exécution fournies dans ces travaux indiquent que pour une trentaine de sites il faut plusieurs dizaines de secondes pour calculer un diagramme de Voronoï sur une surface ayant quelques milliers de faces, ce qui est trop lent pour nos applications. En comparaison, nous obtenons des temps d'exécution similaires pour des maillages et des ensembles de sites de l'ordre du million d'éléments.

Dans le cas où la surface est définie de manière paramétrique, par une fonction de classe C^2, Kunze *et al.* [56] dérivent les équations différentielles permettant d'obtenir sous forme close les bissecteurs du diagramme de Voronoï géodésique d'un ensemble de sites. Ces équations permettent ainsi d'obtenir sous forme close le diagramme de Voronoï, lorsqu'il est possible de résoudre ces équations différentielles sous forme close. Cette méthode ne peut cependant s'appliquer qu'aux surfaces homéomorphes à un disque, pour lesquelles nous possédons une paramétrisation suffisamment lisse, ce qui n'est en général pas le cas pour les problèmes que nous visons.

3.3.4 Diagramme de Voronoï restreint

Nous détaillerons ici l'algorithme de Yan *et al.* [118], à partir duquel nous avons élaboré les algorithmes que nous proposons dans ce chapitre.

Intersection face – cellule

La brique de base du calcul d'un diagramme de Voronoï restreint est le calcul de l'intersection entre une face T_p de la surface $\mathcal{S}$, et la cellule de Voronoï Ω_k d'un site $\mathbf{v}_k$. Comme mentionné en Section 3.3.2, une cellule de Voronoï dans $\mathbb{R}^3$ est un polyèdre délimité par un ensemble de bissecteurs avec des sites voisins. Pour calculer l'intersection entre T_p et Ω_k, il suffit de découper successivement T_p par tous ces bissecteurs. C'est ce que fait l'Algorithme 3.1, et son exécution est illustrée sur la Figure 3.8. Le cœur de cet algorithme consiste à calculer l'intersection entre un polygone et un demi-espace, délimité par un bissecteur entre deux sites. La procédure `intersection_polygone_plan` réalise cette intersection. Ce problème a été très étudié dans la littérature : étant donné un polygone quelconque à n sommets, l'algorithme de Sutherland et Hodgman [106] permet de réaliser cette intersection avec la complexité $O(n)$. Dans notre cas, les faces de $\mathcal{S}$ sont des triangles, et donc des polygones convexes. La convexité signifie que si deux points $\mathbf{x}_1$ et $\mathbf{x}_2$ sont à l'intérieur du polygone, alors le segment $[\mathbf{x}_1,\mathbf{x}_2]$ est également entièrement contenu dans le polygone. Cette propriété signifie qu'un plan ne pourra intersecter le polygone au maximum qu'en deux points. Il est possible de démontrer que l'intersection entre un demi-plan et un polygone convexe reste un polygone convexe. Ainsi au fur et à mesure de l'Algorithme 3.1, les polygones à découper par des bissecteurs sont tous convexes. Dans le cas d'un polygone convexe, Rappoport [91] fournit un algorithme de complexité $\log n$ pour rechercher les points d'intersection. Cette complexité ne prend par contre pas en compte le temps de lecture du polygone, ni la construction du polygone découpé.

Entrées :
– T_p une face de $\mathcal{S}$
– $\mathbf{v}_k$ un site
Sorties :
– $\Omega_k \cap T_p$

```
1 Algorithme
2     Inter ← T_p ;
3     pour chaque site v_ℓ ∈ N(v_k) faire
4         Soit B_{k,ℓ} le bissecteur de [v_k,v_ℓ] ;
5         Inter ← intersection_polygone_plan(Inter,B_{k,ℓ}) ;
6     renvoyer Inter ;
```

Algorithme 3.1: `intersection_face_cellule`$(T_p,\mathbf{v}_k)$: calcul de l'intersection entre une face T_p et la cellule de Voronoï Ω_k de $\mathbf{v}_k$.

Un polygone sera découpé par un plan si et seulement si au moins un de ses sommets n'est pas du même côté du plan que les autres. Une découpe supprime donc au moins un des sommets du polygone à l'étape précédente. Les sommets rajoutés au polygone par la

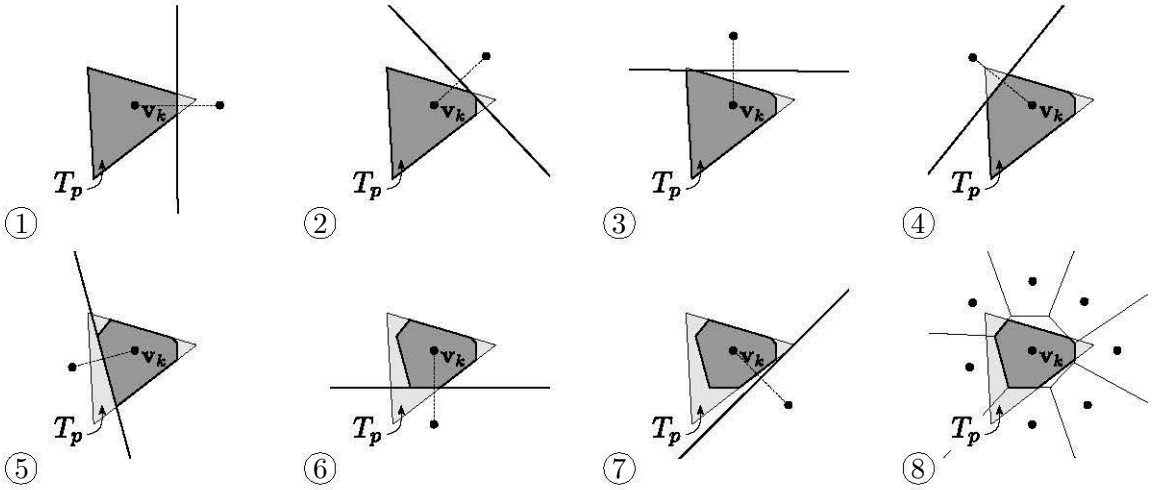


FIGURE 3.8 – Calcul itératif de l'intersection entre une face T_p et la cellule de Voronoï Ω_k, ici dans $\mathbb{R}^2$.

découpe sont les points d'intersection entre le plan et les arêtes du polygone. Lorsque le polygone est convexe, il ne peut pas y avoir plus de deux intersections de cette sorte. Ainsi, la procédure `intersection_polygone_plan` rajoute au maximum un sommet au polygone. Dans le pire des cas, un site est voisin de tous les autres sites, soit n sites. L'intersection entre une cellule de Voronoï et un triangle nécessitera donc n découpes par des plans. La complexité totale de cette opération est donc :

$$O(\log 3) + O(\log 4) + \ldots + O(\log (3+n)) = O(\log ((n+3)!)) \approx O(n \log n). \qquad (3.3.2)$$

Ce pire des cas décrit mal la plupart des situations réelles. En pratique, dans notre implémentation de cet algorithme, nous avons utilisé l'algorithme de Sutherland et Hodgman [106], qui s'est avéré plus efficace étant donné que les sites ont en général un nombre de voisins relativement faible.

Propagation

L'objet de base manipulé par l'algorithme est un *couple*, constitué d'un site et d'une face. Étant donné un tel couple, il est possible de calculer l'intersection entre la face et la cellule de Voronoï du site en utilisant l'Algorithme 3.1. Une fois ce calcul réalisé, de nouveaux couples peuvent être propagés à l'aide d'une file, en examinant le bord de cette intersection, comme illustré sur la Figure 3.9. Sur le bord de l'intersection entre une face T_p et une cellule Ω_k, une arête peut provenir soit :

d'une arête de T_p : dans ce cas, l'arête est contenue dans une arête initiale de T_p. Si cette arête n'est pas un bord de la surface $\mathcal{S}$, elle relie T_p à une face voisine T_q. Dans ce cas, un nouveau couple $(T_q,\mathbf{v}_k)$ est ajouté à la file.

d'un bissecteur : dans ce cas, l'arête est contenue dans la droite correspondant à l'intersection entre le plan de la face et le plan d'un bissecteur. En notant $\mathbf{v}_\ell$ le site de l'autre côté de ce bissecteur, le couple $(T_p,\mathbf{v}_\ell)$ est alors ajouté à la file.

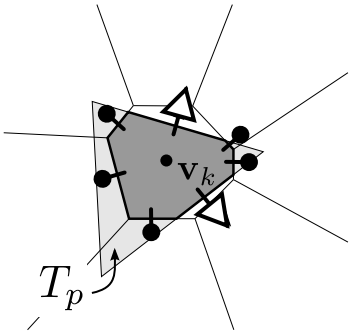


FIGURE 3.9 – Propagation selon les arêtes de l'intersection entre une face T_p et une cellule de Voronoï d'un site $\mathbf{v}_k$. Les arêtes marquées par le symbole ⍋ sont issues des arêtes originales de T_p, et entraînent la propagation du couple $(T_q,\mathbf{v}_k)$ où T_q est la face adjacente à T_p suivant l'arête donnée. Les arêtes portant le symbole ⍊ sont issues d'un bissecteur avec un autre site $\mathbf{v}_\ell$, et le couple $(T_p,\mathbf{v}_\ell)$ est propagé.

Le but de l'algorithme est donc de traiter tous les couples $(T_p,\mathbf{v}_k)$ tels que $T_p \cap \Omega_k \neq \emptyset$ et de ne les traiter qu'une fois. L'Algorithme 3.2 élaboré par Yan *et al.* [118] s'assure de ces deux conditions. Les auteurs utilisent le diagramme de Voronoï restreint pour le remaillage, en particulier pour simplifier un maillage fourni. Dans cette application le nombre de sites est en général inférieur au nombre de faces du maillage. Pour s'assurer de ne pas traiter deux fois le même couple, chaque face enregistre l'ensemble des sites avec lesquels elle a déjà été traitée. Chaque couple face – site obtenu suite à l'intersection entre une face et une cellule de Voronoï est ainsi testé avant d'être ajouté à la file pour savoir s'il n'a pas déjà été inséré. Ce test réalise une recherche parmi les sites stockés pour la face. Dans le pire des cas, le stockage pour chaque face est de $O(n)$, ce qui amène donc un stockage total de $O(mn)$. Le test prend dans le pire des cas $O(\log n)$ opérations pour chaque couple produit à la suite d'une intersection entre le polygone et un bissecteur. Le polygone final, suite à toutes les intersections pouvant avoir $O(n)$ arêtes, la complexité totale de la phase de propagation est donc de $O(n \log n)$ pour un couple face – site, ce qui n'empire pas la complexité dans le pire des cas de la phase d'intersection de la face avec la cellule de Voronoï du site.

La propagation assure qu'étant donné un couple initial $(T_p, \mathbf{v}_k)$, si $T_p \cap \Omega_k \neq \emptyset$, alors toute la composante connexe de $\mathcal{S}$ contenant T_p sera traitée. L'algorithme est initialisé à partir d'une face T_0 quelconque, et le site $\mathbf{v}_0$ le plus proche de son barycentre est recherché. Toutes les faces atteintes lors de la propagation sont marquées. Lorsque la file assurant la propagation est vide, l'algorithme recherche une autre face de $\mathcal{S}$ qui ne serait pas marquée, et relance une propagation s'il en trouve une. Dans le cas contraire, il s'arrête. Sur la totalité de l'algorithme, la recherche de faces non marquées ne demandera pas plus de m opérations. La recherche du site le plus proche de cette face peut être effectuée avec la complexité en temps moyenne de $O(\log n)$ en utilisant une structure de données adaptée. En utilisant une recherche linéaire sur l'ensemble des sites, cette complexité est dans le pire des cas $O(n)$.

Complexité totale en temps et en espace

La complexité est mesurée en fonction du nombre de faces m du maillage, et du nombre de sites n. L'Algorithme 3.2 traite tous les couples $(T_p,\mathbf{v}_k)$ tels que $T_p \cap \Omega_k \neq \emptyset$ et uniquement ceux là. Dans le pire des cas, il est possible d'avoir à traiter $O(mn)$ couples. Pour chaque couple, l'intersection entre la cellule de Voronoï et la face est réalisée en $O(n \log n)$, et la propagation en $O(n \log n)$. La complexité en temps totale de cet algorithme dans le pire des cas est donc de $O(mn^2 \log n)$. Le coût de la recherche du site le plus proche d'une face lors de l'initialisation est absorbé dans cette complexité.

En terme d'espace, sans compter l'espace nécessaire au stockage du diagramme restreint, deux structures sont construites. Tout d'abord la file servant à la propagation, qui dans le

Entrées :
– $\mathbf{V} = \{\mathbf{v}_k\}_{k=1}^n$ l'ensemble des sites
– $\mathcal{S}$ la surface restreignant le diagramme
Sorties :
– Diagramme de Voronoï de $\mathbf{V}$ restreint à $\mathcal{S}$

```
1  Algorithme
2      RVD ← ∅ ;                          // Diagramme de Voronoï Restreint
3      File ← ∅ ;
4      pour chaque face T_p de S faire
5          SitesTraités[T_p] ← ∅ ;        // enregistrement des couples traités
6      pour chaque face T_0 de S faire
7          si SitesTraités[T_0] est vide alors
8              Soit v_0 le site la plus proche du barycentre de T_0 ;
9              Ajouter (T_0,v_0) dans la File ;
10             tant que la File n'est pas vide faire
11                 Extraire (T_p,v_k) de la tête de File ;
12                 // Algorithme 3.1
13                 T_p ∩ Ω_k ← intersection_face_cellule(T_p,v_k) ;
14                 RVD ← RVD ∪ (T_p ∩ Ω_k) ;
15                 pour chaque arête e de T_p ∩ Ω_k faire
16                     si e est une portion d'une arête de T_p alors
17                         Soit T_q la face adjacente à T_p par e ;
18                         si v_k ∉ SitesTraités[T_q] alors
19                             SitesTraités[T_q] ← SitesTraités[T_q] ∪ {v_k} ;
20                             Ajouter (T_q,v_k) dans la File ;
21                     sinon // e est contenue dans un bissecteur
22                         Soit v_ℓ telle que le bissecteur de [v_k,v_ℓ] contient e;
23                         si v_ℓ ∉ SitesTraités[T_p] alors
24                             SitesTraités[T_p] ← SitesTraités[T_p] ∪ {v_ℓ} ;
25                             Ajouter (T_p,v_ℓ) dans la File ;
26     renvoyer RVD;
```

Algorithme 3.2: `voronoï_restreint(`$\mathbf{V}$,$\mathcal{S}$`)` : Calcul du diagramme de Voronoï de $\mathbf{V} = \{\mathbf{v}_k\}_{k=1}^n$ restreint à la surface $\mathcal{S}$.

pire des cas contient $O(mn)$ couples, et la structure enregistrant pour chaque face les sites ayant été traités pour elle, qui peut également atteindre une taille de $O(mn)$.

Nous introduisons maintenant notre contribution, qui consiste à supprimer cette dernière structure, et nous permet donc d'améliorer nettement l'espace mémoire utilisé par l'algorithme. Nous améliorons également le test permettant de déterminer si un couple $(T_p,\mathbf{v}_k)$ a déjà été examiné. Même si cette amélioration est invisible dans la complexité dans le pire des cas, nous montrerons sur la Figure 3.11 que le temps de calcul est également plus faible.

3.4 Contribution : améliorer la propagation

Nous détaillerons ici la première variante de l'Algorithme 3.2, qui a été élaborée pour le cas où le nombre de sites est élevé par rapport au nombre de faces. Il s'agit d'une idée très simple consistant à réordonner les calculs, pour pouvoir simplifier le test vérifiant si un couple $(T_p,\mathbf{v}_k)$ a déjà été traité auparavant. Cette variante n'améliore pas la complexité en temps dans le pire des cas qui reste $O(mn^2 \log n)$. En espace, par contre, l'organisation des calculs permet de s'assurer que le nombre de couples stockés pendant la propagation ne dépasse jamais $n + m$. De plus, il n'est plus nécessaire d'enregistrer pour les faces les sites traités, ce qui occupait un espace de $O(mn)$. Cette structure est remplacée par un tableau de taille n et un autre de taille m, ce qui donne au total un espace nécessaire de $O(m + n)$. Comme nous le verrons dans un instant, le test permettant de s'assurer de ne pas traiter deux fois le même couple est réalisé en $O(1)$ avec cette nouvelle organisation, ce qui n'améliore pas la complexité asymptotique, mais a un réel impact sur les performances pratiques de l'algorithme lorsque le nombre de cellules intersectant une même face grandit.

3.4.1 Priorité à la propagation au sein d'une face

Nouvelles files pour la propagation

Pour réordonner les calculs, nous séparons la file utilisée dans l'Algorithme 3.2 en deux files distinctes, et scindons la boucle globale sur cette file en une boucle interne, et une boucle externe, chacune travaillant sur l'une des deux files. Cette séparation est fondée sur le fait qu'une face T_p est connexe. Ainsi, si nous possédons un couple $(T_p,\mathbf{v}_k)$ constitué de cette face et d'un site $\mathbf{v}_k$ tel que $\Omega_k \cap T_p \neq \emptyset$, en n'utilisant que la propagation par les arêtes de $T_p \cap \Omega_k$ qui sont issues des bissecteurs avec les sites voisins de $\mathbf{v}_k$, il est possible de parcourir l'ensemble des sites $\mathbf{v}_\ell$ tels que $T_p \cap \Omega_\ell \neq \emptyset$. Nous utiliserons le terme *couple valide* pour définir les couples $(T_p,\mathbf{v}_k)$ tels que $T_p \cap \Omega_k \neq \emptyset$.

Le nouvel algorithme contient donc une boucle externe, sélectionnant les triangles de T_p un par un et pour chacun d'entre eux un site $\mathbf{v}_k$ tel que le couple $(T_p,\mathbf{v}_k)$ soit valide. À partir de ce couple, une propagation uniquement sur les sites est initiée pour partitionner T_p entre les différentes cellules de Voronoï qu'elle intersecte. Ces deux boucles sont illustrées en Figure 3.10. Nous utiliserons donc deux files :

une file de faces contenant des couples $(T_p,\mathbf{v}_k)$ valides. Cette file sert à initier le traitement des faces de $\mathcal{S}$. Une boucle externe extrait les couples de cette file, fixe la face en cours de traitement à T_p, et initialise la file interne avec $\mathbf{v}_k$.

une file de sites contenant des sites $\mathbf{v}_\ell$ tels que le couple qu'ils forment avec la face T_p en cours de traitement soit valide. Cette file est gérée par une boucle interne, qui s'arrêtera lorsque la face courante aura été intégralement partitionnée.

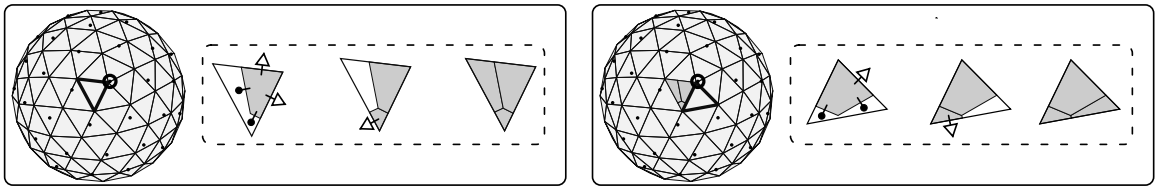


FIGURE 3.10 – Boucles imbriquées dans le calcul du diagramme de Voronoï restreint, avec priorité sur la propagation au sein d'une face. Ici deux itérations de la boucle externe sur la file de faces sont encadrées d'un trait continu. À gauche de ces cadres, la surface, les sites, et le couple triangle – site utilisé pour initialiser la boucle interne. La boucle interne, encadrée en pointillés partitionne le triangle entre les différentes cellules de Voronoï. Le symbole ⍋ symbolise la propagation d'un couple face – site dans la file de faces, et le symbole ↑ correspond à la propagation d'un site dans la file de sites.

Tester la propagation

Muni de ces deux files, il devient plus simple de tester si un couple $(T_p,\mathbf{v}_k)$ a déjà été traité auparavant. Tout d'abord, à partir d'un couple valide, la boucle interne assure que la face du couple sera intégralement partitionnée. Il ne sera donc jamais plus nécessaire de revenir à cette face. Nous enregistrons donc pour chaque face un marqueur indiquant si elle a déjà été ajoutée à la file de faces, et nous assurons que cette face ne passera qu'une seule fois par cette file. Ce test est réalisé en temps $O(1)$, et l'ensemble des marqueurs nécessaires occupe donc un espace de $O(m)$.

Au niveau de la boucle interne, la convexité des cellules de Voronoï fait que l'intersection entre une face T_p et une cellule de Voronoï Ω_k est un polygone connexe. Un site n'aura donc pas besoin d'être traité plus d'une fois pour une face. Pour nous en assurer, nous enregistrons pour chaque site $\mathbf{v}_k$ l'indice p de la dernière face T_p avec laquelle l'intersection avec Ω_k a été calculée. Les faces étant partitionnées d'un coup, il suffit alors de tester si la face en cours de traitement correspond à la dernière face traitée pour le site $\mathbf{v}_\ell$ candidat à la propagation. Lorsque c'est le cas, le site a donc déjà été traité pour la face, sinon, nous mettons à jour la dernière face traitée de $\mathbf{v}_\ell$, et le propageons dans la file de sites. Ce test s'effectue donc en un temps $O(1)$. L'ensemble des indices stockés pour les sites nécessite un espace de $O(n)$.

3.4.2 Analyse

Complexité dans le pire des cas

Comme pour l'Algorithme 3.2, l'Algorithme 3.3 traitera une fois et une seule chacune des paires $(T_p,\mathbf{v}_k)$ valides. Dans le pire des cas, ces paires sont au nombre de $O(mn)$, et il s'agit de la complexité du résultat, qui ne peut pas être évitée. Pour chacune de ces paires, l'Algorithme 3.1 calcule l'intersection entre une face et une cellule de Voronoï, en temps $O(n \log n)$ dans le pire des cas. La propagation traitera chaque arête de cette intersection, et réalisera un test de complexité $O(1)$ pour s'assurer que la propagation est nécessaire. La phase de propagation pour un couple valide est donc réalisée avec une complexité de $O(n)$, absorbée par la complexité du calcul de l'intersection. La complexité totale est donc de $O(mn^2 \log n)$.

Entrées :
– $\mathbf{V} = \{\mathbf{v}_k\}_{k=1}^n$ l'ensemble des sites
– $\mathcal{S}$ la surface restreignant le diagramme
Sorties :
– Diagramme de Voronoï de $\mathbf{V}$ restreint à $\mathcal{S}$

Algorithme
  RVD $\leftarrow \emptyset$; `// Diagramme de Voronoï Restreint`
  FileDeFaces $\leftarrow \emptyset$;
  FileDeSites $\leftarrow \emptyset$;
  pour chaque *face* T_p *de* $\mathcal{S}$ **faire**
    FaceTraitée$[T_p] \leftarrow$ faux ; `// enregistrement des faces traitées`
  pour chaque *site* $\mathbf{v}_k$ *de* $\mathbf{V}$ **faire**
    FaceDuSite$[\mathbf{v}_k] \leftarrow -1$; `// dernière face traitée pour un site`
  pour chaque *face* T_{p_0} *de* $\mathcal{S}$ **faire**
    si FaceTraitée$[T_{p_0}]$ *est faux* **alors**
      FaceTraitée$[T_{p_0}] \leftarrow$ vrai ;
      Soit $\mathbf{v}_{k_0}$ le site le plus proche du barycentre de T_{p_0} ;
      Ajouter $(T_{p_0},\mathbf{v}_{k_0})$ dans la FileDeFaces ;
      tant que *la* FileDeFaces *n'est pas vide* **faire**
        Extraire $(T_p,\mathbf{v}_{k_p})$ de la tête de la FileDeFaces ;
        FaceDuSite$[\mathbf{v}_{k_p}] \leftarrow p$;
        Ajouter $\mathbf{v}_{k_p}$ à la FileDeSites ;
        tant que *la* FileDeSites *n'est pas vide* **faire**
          Extraire $\mathbf{v}_k$ de la FileDeSites ;
          `// Algorithme 3.1`
          $T_p \cap \Omega_k \leftarrow$ `intersection_face_cellule(`T_p,$\mathbf{v}_k$`)` ;
          RVD $\leftarrow$ RVD $\cup$ $(T_p \cap \Omega_k)$;
          pour chaque *arête* e *de* $T_p \cap \Omega_k$ **faire**
            si e *est une portion d'une arête de* T_p **alors**
              Soit T_q la face adjacente à T_p par e ;
              si FaceTraitée$[T_q]$ *est faux* **alors**
                FaceTraitée$[T_q] \leftarrow$ vrai ;
                Ajouter $(T_q,\mathbf{v}_k)$ dans la FileDeFaces ;
            sinon `//` e `est contenue dans un bissecteur`
              Soit $\mathbf{v}_\ell$ telle que le bissecteur de $[\mathbf{v}_k,\mathbf{v}_\ell]$ contient e ;
              si FaceDuSite$[\mathbf{v}_\ell] \neq p$ **alors**
                FaceDuSite$[\mathbf{v}_\ell] \leftarrow p$;
                Ajouter $\mathbf{v}_\ell$ dans la FileDeSites ;
  renvoyer RVD;

Algorithme 3.3: `voronoï_restreint(`$\mathbf{V}$,$\mathcal{S}$`)` : Calcul du diagramme de Voronoï de $\mathbf{V} = \{\mathbf{v}_k\}_{k=1}^n$ restreint à la surface $\mathcal{S}$, en propageant en priorité à l'intérieur des faces de $\mathcal{S}$.

Cette complexité dans le pire des cas n'améliore pas la complexité de l'algorithme naïf consistant à calculer pour tous les couples possibles l'intersection, éventuellement vide, entre la face et la cellule de Voronoï de du site qu'il contient. Elle ne reflète cependant pas la réalité du temps nécessaire pour les cas pratiques sur lesquels nous avons travaillé comme nous le montrerons plus loin.

Complexité en espace

Au niveau de l'espace nécessaire, l'Algorithme 3.3 enregistre pour chaque face un marqueur pour vérifier si elle a été traitée, et pour chaque site l'indice de la dernière face traitée. La file de faces ne peut pas contenir deux couples comprenant le même triangle, et sa taille est donc limitée à m. La file de site ne contient pas deux fois le même site, et sa taille est donc au pire de n. Ainsi l'espace mémoire total occupé par cet algorithme est de $O(m+n)$.

Performances sur des cas pratiques

La Figure 3.12 donne le temps nécessaire au calcul d'un diagramme de Voronoï restreint en fonction du nombre de sites et du nombre de faces. La forme du graphe de cette fonction semble indiquer que la complexité de $O(mn^2 \log n)$ dans le pire des cas décrit assez mal les performances de notre algorithme en pratique, qui semble avoir une comportement plus proche d'une fonction linéaire en m et n, même si les tests sont ici trop marginaux pour une étude précise de cette complexité pratique. Il apparaît également que pour les exemples que nous avons choisis, le temps de calcul soit peu dépendant de la forme de $\mathcal{S}$. La différence entre la complexité constatée et la complexité dans le pire des cas s'explique par le fait qu'en réalité, la taille des voisinages des cellules de Voronoï est relativement faible, et qu'en tirant aléatoirement les sites sur la surface, leur répartition fait que le nombre de couples valides est plutôt de l'ordre de $m+n$ que de mn. Il ne s'agit là néanmoins que d'une conjecture, et non d'une analyse en moyenne de la complexité.

Par rapport à l'Algorithme 3.2, nous constatons effectivement une amélioration des performances lorsque le nombre de sites croît au détriment du nombre de faces, comme le montre la Figure 3.11. Cette amélioration est due à l'amélioration du test permettant de déterminer si un couple face – site a déjà été traité auparavant. Dans notre cas, ce test est effectué en $O(1)$, comparé à $O(\log s)$ précédemment avec s le nombre de sites stockés pour la face.

3.5 Contribution : calcul par composante connexe

Nous présentons ici une autre variante de l'Algorithme 3.2. Par rapport à l'Algorithme 3.3, il s'agit d'inverser la boucle externe sur les faces, et la boucle interne sur les sites par face, dans le but de calculer le diagramme de Voronoï restreint cellule par cellule, plutôt que face par face. Contrairement aux faces, les cellules de Voronoï restreintes ne sont pas connexes. Il ne sera donc pas possible de calculer de manière efficace les cellules restreintes une par une. Par contre, la propagation permet de calculer les *composantes connexes* des cellules restreintes une par une.

3.5.1 Cas d'application

Changer l'ordre dans lequel le calcul est réalisé n'est avantageux que lorsque le stockage du diagramme de Voronoï restreint n'est pas envisageable. Dans ce cas, pour obtenir la

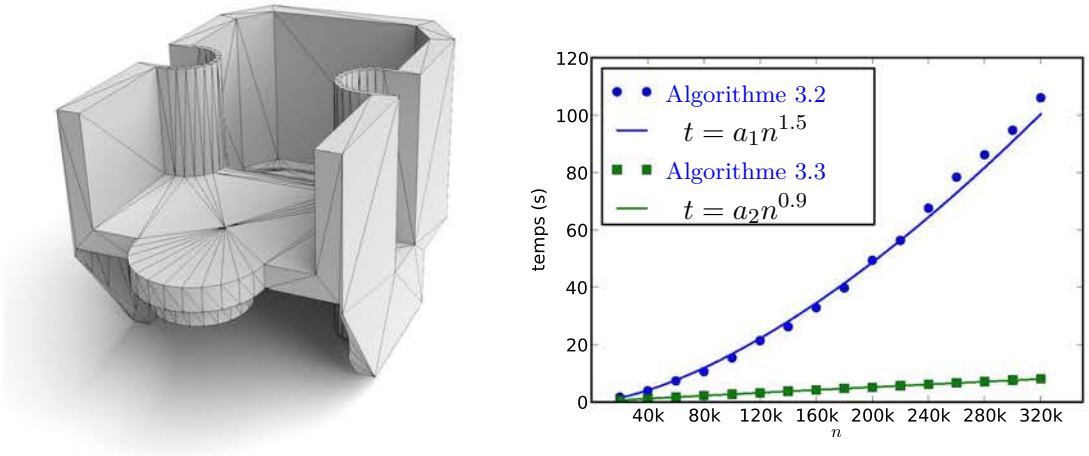


FIGURE 3.11 – Comparaison des performances entre l'Algorithme 3.2 et l'Algorithme 3.3 dans le cas où le nombre de couples valides pour une même face devient important. Le modèle choisi (à gauche) comporte 1000 faces, et nous avons testé pour un ensemble de sites variant de 20000 à 320000. Les courbes associées aux données (à droite) ont été obtenue par régression, en déterminant les meilleurs paramètres a et b (au sens des moindres carrés) tels que $t = an^b$, où t est le temps.

valeur d'une fonction définie sur ce diagramme, le calcul doit se faire au fur et à mesure de sa construction et les éléments produits doivent être supprimés après avoir ajouté leur contribution à la valeur de la fonction. L'application particulière que nous visons pour cet algorithme est le remaillage.

Sur le même principe que la triangulation de Delaunay peut être construite à partir d'un diagramme de Voronoï, il est possible de construire une triangulation de la surface à partir d'un diagramme de Voronoï restreint. Cette méthode a été formalisée par Edelsbrunner et Shah [36]. Le diagramme de Voronoï restreint forme une partition de $\mathcal{S}$. Lorsque les cellules restreintes de deux sites sont adjacentes sur la surface, une arête est créée entre ces deux sites. Lorsque les cellules de trois sites se rencontrent en un point, un triangle est créé entre les trois sites. Comme dans le cas d'un diagramme de Voronoï de $\mathbb{R}^2$, il n'existe en général pas de point de $\mathcal{S}$ équidistant de quatre sites. Pour que le remaillage soit correct, il est nécessaire que le nouveau maillage de $\mathcal{S}$ lui soit *homéomorphe*. Un homéomorphisme entre deux surfaces signifie qu'il est possible de déformer l'une pour obtenir l'autre sans découpage ni collage. Edelsbrunner et Shah [36] énoncent des conditions pour s'assurer que la triangulation de Delaunay restreinte d'un ensemble de sites est homéomorphe à la surface restreignant le diagramme. Dans notre cas, $\mathcal{S}$ étant une surface de dimension 2 plongée dans $\mathbb{R}^3$, il s'agit de vérifier que :

- une cellule restreinte est homéomorphe à un disque ;
- l'intersection de deux cellules restreintes est vide ou homéomorphe à un segment ;
- l'intersection de trois cellules restreintes est vide ou homéomorphe à un point.

Pour vérifier ces trois propriétés en utilisant l'Algorithme 3.3, il est alors nécessaire d'enregistrer le diagramme de Voronoï restreint intégralement. À partir de ce diagramme, les différentes composantes connexes composantes connexes des cellules peuvent être reconstruites pour vérifier les différentes propriétés précédentes. L'Algorithme 3.8 que nous proposons permet en revanche de vérifier ces propriétés sans avoir à stocker le diagramme de

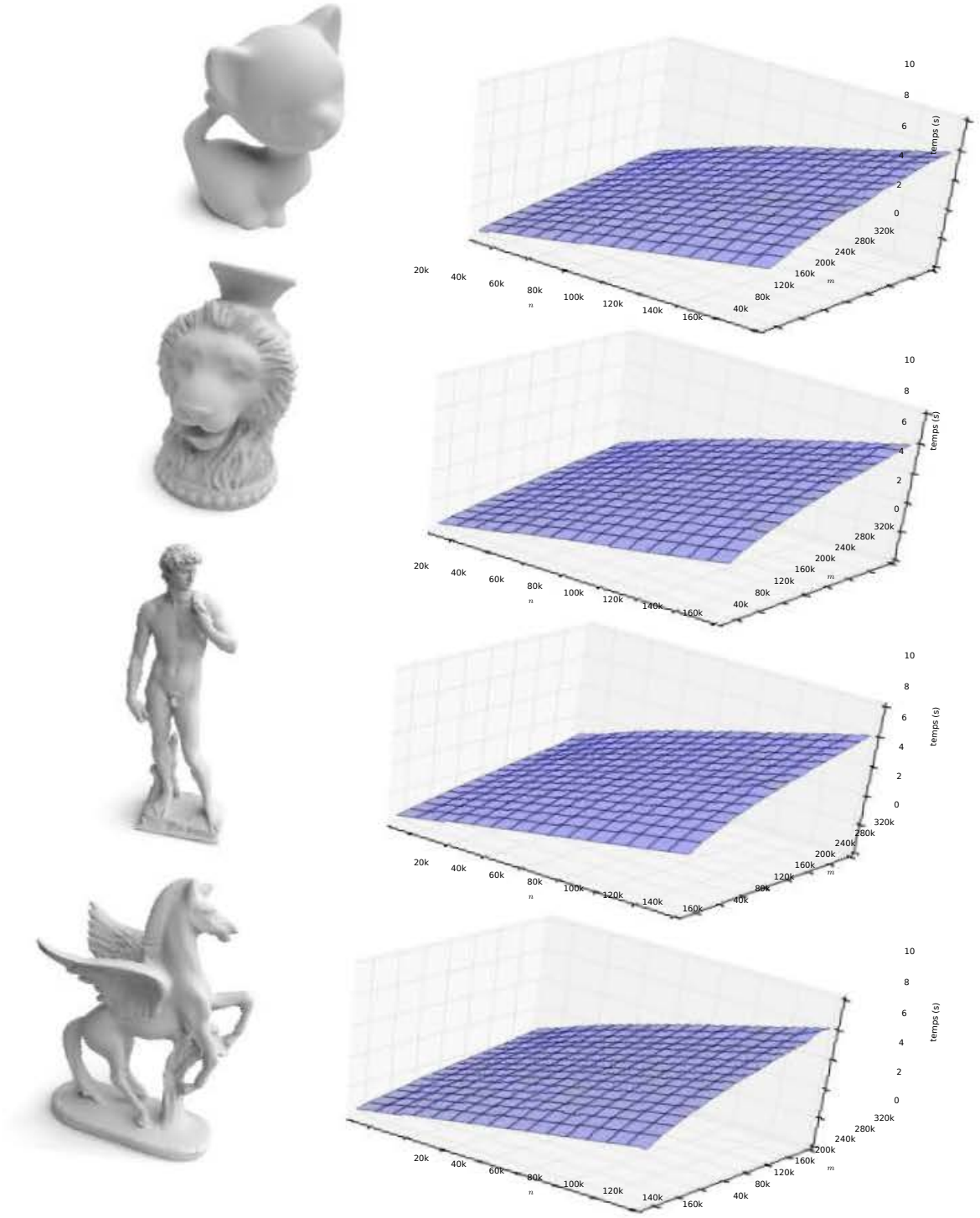


FIGURE 3.12 – Performance de l'Algorithme 3.3, en faisant varier la résolution du maillage m de 20000 à 320000 faces et celle de l'ensemble des sites n de 10000 à 160000. Les sites sont tirés aléatoirement sur la surface pour chaque paire (m,n) testée. Ces performances ont été obtenues en utilisant une implémentation mono processus, utilisant un processeur Intel Core i7 2620M cadencé à 2.7Ghz. Dans chaque cas, la forme des graphes montre que le temps d'exécution est plus proche d'un comportement linéaire en m et n que de la complexité théorique dans le pire des cas.

Voronoï restreint, tout au long de son calcul.

3.5.2 Priorité à la propagation au sein d'une cellule

La propagation proposée au niveau de l'Algorithme 3.3 repose sur le fait que les triangles de la surface sont connexes pour assurer qu'un triangle sera entièrement partitionné. Les cellules restreintes par contre ne sont pas nécessairement connexes. Propager prioritairement au sein d'une cellule n'assure donc que le traitement complet d'une seule composante connexe de la cellule restreinte. Notre algorithme comportera donc une boucle externe, qui lancera le calcul de nouvelles composantes connexes, à partir d'un couple valide, et une boucle interne qui propagera à partir de la face du couple, de face en face, le calcul de la composante connexe de la cellule restreinte.

Propagation interne sur les faces

Au sein d'une composante connexe, nous utilisons la même stratégie que pour partitionner une face, afin de ne pas traiter deux fois le même couple (T_p,$\mathbf{v}_k$). Pour chaque face, nous enregistrons le dernier site ayant été traité pour cette face. Les cellules de Voronoï étant convexes, il n'est pas possible qu'une face intervienne dans deux composantes connexes différentes d'une même cellule. Ainsi, même en traitant successivement deux composantes connexes de la même cellule, ce marquage reste suffisant. L'Algorithme 3.4 réalise la propagation utilisant ce test pour calculer une composante connexe d'une cellule restreinte.

Entrées :
– $T_p \cap \Omega_k$
Données :
– FileDeFaces la file dans laquelle les faces sont propagées
– SiteDeLaFace le dernier site traité pour la face
Sorties :
– Mise à jour de la FileDeFaces et du SiteDeLaFace

```
1 Algorithme
2     pour chaque arête e de T_p ∩ Ω_k faire
3         si e est une portion d'une arête de T_p alors
4             Soit T_q la face adjacente à T_p par e;
5             si SiteDeLaFace[T_q] ≠ v_k alors
6                 SiteDeLaFace[T_q] ← v_k ;
7                 Marquer T_q comme traitée ;          // voir l'Algorithme 3.8
8                 Ajouter T_q dans la FileDeFaces ;
```

Algorithme 3.4: `propagation_triangles`($T_p \cap \Omega_k$) : Propagation sur les faces pour le calcul d'une composante connexe du diagramme de Voronoï restreint.

Propagation externe sur les sites

Il n'est par contre plus possible de simplement marquer un site comme traité ou non, car il peut rester des composantes connexes à calculer pour ce site. Une autre idée pratique

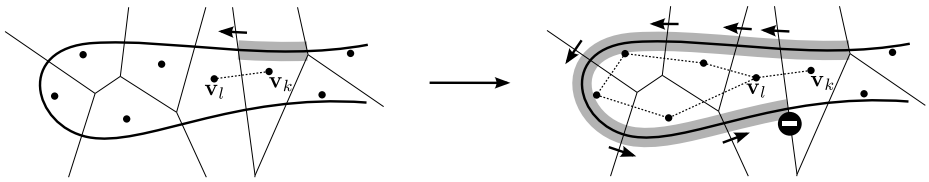


FIGURE 3.13 – Échec du calcul avec un marquage sur les arêtes de Delaunay. À gauche, en initialisant la propagation par une composante connexe du site $\mathbf{v}_k$, le calcul se propage au site $\mathbf{v}_\ell$ voisin, et l'arête de Delaunay les reliant est marquée, interdisant toute propagation de $\mathbf{v}_\ell$ vers $\mathbf{v}_k$. À droite, après avoir calculé quelques nouvelles composantes connexes, le calcul échoue à se propager pour calculer une composante connexe de $\mathbf{v}_k$, car l'arête marquée entre $\mathbf{v}_k$ et $\mathbf{v}_\ell$ l'en empêche. Ce cas dans $\mathbb{R}^2$ se généralise facilement à $\mathbb{R}^3$.

aurait été de marquer les arêtes du graphe des voisinages entre sites. Après le calcul d'une composante, les arêtes vers tous les sites adjacents vers lesquels le calcul a été propagé sont marquées, pour interdire une propagation dans l'autre sens. De cette façon, nous ne recalculerons jamais une composante connexe déjà calculée. Cette condition est néanmoins encore trop forte, car comme l'illustre la Figure 3.13 il peut arriver que deux composantes connexes d'une même cellule soient adjacentes à une ou plusieurs composantes connexes d'une même autre cellule. Ce marquage interdirait donc la propagation du calcul à une composante connexe d'une cellule n'ayant pas encore été calculée.

La solution que nous avons choisie consiste à encoder le bord de l'ensemble des composantes connexes déjà calculées. Dès qu'une nouvelle composante connexe est calculée, nous mettons à jour ce bord, sélectionnons une nouvelle composante à calculer au delà de ce bord, et lançons le calcul de cette composante. Pour cela, nous n'avons besoin que de l'ensemble des sites dont la cellule de Voronoï a une composante connexe qui borde la zone traitée, avec une face pour lancer le calcul pour cette composante. Pour ce site adjacent, nous devons pouvoir connaître quelles composantes connexes voisines ont été calculées auparavant. Nous enregistrons ces informations en conservant pour chaque site un ensemble de *coins*.

Un coin est un point de la surface où trois cellules de Voronoï se rejoignent. Il peut donc être représenté par la face T_p dans laquelle cette intersection se produit, et les trois sites $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$ dont les cellules se rejoignent en ce point. Étant donné un ensemble de composantes connexes de cellules restreintes déjà calculées, le bord de la région de $\mathcal{S}$ couverte par ces composantes est un ensemble de courbes polygonales sur la surface. Parmi les sommets de ces courbes, un certain nombre correspondent à des coins. En partant d'un coin, et en parcourant la courbe du bord jusqu'au coin suivant, les sites correspondant aux composantes de part et d'autre du bord ne varient pas. Il n'est pas nécessaire de stocker l'ensemble de la courbe polygonale reliant deux coins car il est possible de déterminer toutes les composantes connexes adjacentes à la zone calculée uniquement à partir des coins du bord. Pour chaque site dont la cellule de Voronoï restreinte comporte une ou plusieurs composantes connexes n'ayant pas encore été calculées jouxtant la zone déjà calculée, nous enregistrons l'ensemble des coins du bord correspondant à ces composantes connexes, comme illustré sur la Figure 3.14. Lorsqu'une nouvelle composante connexe est calculée pour un site $\mathbf{v}_k$, nous mettons à jour les listes de coins en examinant chacun des coins $(T_p,\mathbf{v}_k,\mathbf{v}_{l_1},\mathbf{v}_{l_2})$ de la composante calculée. Lorsqu'un coin apparaît déjà dans la liste de coins de $\mathbf{v}_k$, nous le supprimons de cette liste. Lorsqu'il n'apparaît pas dans cette liste, nous l'ajoutons aux listes des sites $\mathbf{v}_{l_1}$ et $\mathbf{v}_{l_2}$, qui ont une composante adjacente à ce coin. Cette propagation

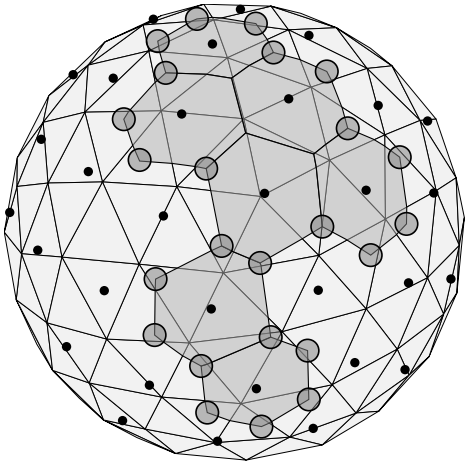


FIGURE 3.14 – Ensemble des coins stockés à un instant donné du calcul du diagramme de Voronoï restreint par composantes connexes. Les cellules restreintes calculées correspondent aux zones grisées de la surface, et les coins enregistrés à cette étape du calcul sont représentés par les cercles sur le bord de la zone grisée. Chaque coin est stocké dans la liste de coins de *chaque* site dont la cellule restreinte a une composante connexe adjacente à ce coin.

est formalisée par l'Algorithme 3.5.

Cas des bords sans coins

Nous avons dit que nous encodions le bord de la zone de $\mathcal{S}$ traitée par l'entremise de l'ensemble des coins le long de ce bord. Il peut arriver cependant qu'une des courbes polygonales appartenant au bord n'ait pas de coins. Cette situation peut arriver dans le cas illustré en Figure 3.15, au niveau des zones tubulaires de $\mathcal{S}$. Dans un tel cas, la courbe du bord n'ayant pas de coins sépare deux composantes connexes de cellules restreintes, correspondant à deux sites que nous nommerons $\mathbf{v}_k$ et $\mathbf{v}_\ell$. Pour gérer ce cas, nous déterminons la face T_p d'indice minimal le long du bord incriminé, et générons un faux coin $(T_p,\mathbf{v}_k,\mathbf{v}_\ell,0)$, que nous propageons de la même façon que les coins classiques.

Pour détecter ce cas, il ne suffit pas de compter le nombre de coins le long du bord d'une composante connexe calculée. En effet, il est possible que le bord de la composante connexe ne soit pas lui-même connexe, et même si une des composantes connexes du bord comporte un coin, il peut exister une ou plusieurs autres composantes du bord n'en ayant pas. Nous reconstruisons donc intégralement le bord d'une composante connexe au fur et à mesure de son calcul, et une fois la composante terminée, nous parcourons toutes les portions connexes du bord obtenues pour créer les faux coins de celle n'en ayant pas.

Afin de reconstruire le bord, nous enregistrons pour chaque face T_p l'ensemble des portions de bord qu'elle contient. Nous appellerons *extrémité* l'intersection d'un bord de $\Omega_{k|\mathcal{S}}$ avec une arête de T_p. Ce point est donc l'intersection entre un bissecteur de $\mathbf{v}_k$ avec un autre site $\mathbf{v}_\ell$ et une arête joignant deux sommets $\mathbf{x}_i$ et $\mathbf{x}_j$ de $\mathcal{S}$. Nous encoderons donc une telle extrémité par un quadruplet $(\mathbf{x}_i,\mathbf{x}_j,\mathbf{v}_k,\mathbf{v}_\ell)$. Un exemple est fourni sur la Figure 3.15.

Après avoir calculé l'intersection entre une face T_p et une cellule de Voronoï Ω_k, nous déterminons l'ensemble des portions de bord que la face T_p contient. Si la composante connexe de $\Omega_{k|\mathcal{S}}$ est entièrement contenue dans la face, rien ne se passe. Par contre, lorsque nous découvrons une portion du bord reliant deux extrémités entre elles, nous examinons si cette portion de bord comporte un coin, et l'encodons par ses deux extrémités. Pour chaque face T_q adjacente aux arêtes correspondant à ces extrémités, si la face a été calculée précédemment, nous recollons la portion de bord que nous possédons avec celles de T_q ayant les mêmes extrémités. Pour effectuer ces recollements, nous utilisons une structure de

Entrées :
– $T_p \cap \Omega_k$
Données :
– FileDeSites la file dans laquelle les sites sont propagés
– CoinsDuSite un ensemble de coins pour chaque site
Sorties :
– Mise à jour de la FileDeSites et des CoinsDuSite

```
1  Algorithme
2      pour chaque coin c = (T_p,v_k,v_{l_1},v_{l_2}) sur le bord de T_p ∩ Ω_k faire
3          si c ∈ CoinsDuSite[v_k] alors
4              CoinsDuSite[v_k] ← CoinsDuSite[v_k] \ c ;
5          sinon
6              si CoinsDuSite[v_{l_1}] est vide alors
7                  Ajouter v_{l_1} dans la FileDeSites ;
8              CoinsDuSite[v_{l_1}] ← CoinsDuSite[v_{l_1}] ∪ c ;
9              si CoinsDuSite[v_{l_2}] est vide alors
10                 Ajouter v_{l_2} dans la FileDeSites ;
11             CoinsDuSite[v_{l_2}] ← CoinsDuSite[v_{l_2}] ∪ c ;
```

Algorithme 3.5: `propagation_coins`$(T_p \cap \Omega_k)$: propagation des coins pour lancer le calcul de nouvelles composantes connexes du diagramme de Voronoï restreint.

(a) Bord sans coins (b) Extrémités

FIGURE 3.15 – Reconstruction des bords pour la détection des bords sans coins. Une composante connexe du bord de $\Omega_{k|\mathcal{S}}$ peut ne pas avoir de coins (a). Il est donc nécessaire de reconstruire le bord. (b) La face T_p contient une portion du bord d'une composante connexe de $\Omega_{k|\mathcal{S}}$, qui sera encodée par ses deux extrémités $\mathbf{b}_1$ et $\mathbf{b}_2$. Chaque extrémité est à l'intersection entre une arête de T_p et un bissecteur. Une extrémité est encodée par les deux sommets de l'arête et les deux sites définissant le bissecteur. Ici $\mathbf{b}_1 = (\mathbf{x}_1,\mathbf{x}_3,\mathbf{v}_1,\mathbf{v}_2)$ et $\mathbf{b}_2 = (\mathbf{x}_2,\mathbf{x}_3,\mathbf{v}_1,\mathbf{v}_3)$.

données permettant de gérer un ensemble de classes d'équivalences, détaillée dans Cormen *et al.* [20, Section 21.3]. Lorsque nous fusionnons deux portions de bord, nous calculons également la face d'indice minimal pour la classe, et si cette classe contient un coin. Ce traitement est synthétisé dans l'Algorithme 3.6.

Entrées :
– $T_p \cap \Omega_k$
Données :
– BordsDeLaFace un ensemble de portions de bord par face
– SiteDeLaFace le dernier site traité pour une face
– Classe gestionnaire des classes d'équivalence du bord
Sorties :
– Mise à jour des BordsDeLaFace

```
1  Algorithme
2      pour chaque portion de bord (b1,b2) sur le bord de Tp ∩ Ωk faire
3          BordsDeLaFace[Tp] ← BordsDeLaFace[Tp] ∪ (b1,b2) ;
4          Soient e1 et e2 les arêtes de Tp contenant respectivement b1 et b2 ;
5          si une face T1 jouxte Tp par e1 et SiteDeLaFace[T1] = vk alors
6              pour chaque portion de bord (b3,b4) ∈ BordsDeLaFace[T1] faire
7                  si b1 = b3 ou b1 = b4 alors
8                      Fusionner Classe[(b1,b2)] et Classe[(b3,b4)] ;
9          si une face T2 jouxte Tp par e2 et SiteDeLaFace[T2] = vk alors
10             pour chaque portion de bord (b3,b4) ∈ BordsDeLaFace[T2] faire
11                 si b2 = b3 ou b2 = b4 alors
12                     Fusionner Classe[(b1,b2)] et Classe[(b3,b4)] ;
```

Algorithme 3.6: `reconstruire_bords`$(T_p \cap \Omega_k)$: reconstruction des bords d'une composante connexe.

Une fois la composante connexe de $\mathbf{v}_k$ calculée, pour chaque composante connexe du bord n'ayant pas de coin nous créons un faux coin que nous propageons. Cette propagation est donnée par l'Algorithme 3.7. Les faux coins sont traités exactement de la même façon que les vrais coins. Étant donné que la cellule propagée aura également un bord sans coin, elle créera également un faux coin, et l'utilisation de la face d'indice minimal assure qu'elle trouvera le même que celui que $\mathbf{v}_k$ aura propagé, ce qui empêchera la propagation de revenir sur ses pas.

Algorithme global

En combinant les différents algorithmes, il devient possible de calculer le diagramme de Voronoï restreint complet, composante connexe par composante connexe. L'Algorithme 3.8 initialise toutes les structures de données utilisées, et utilise ces différents algorithmes pour effectuer ce calcul.

Entrées :
– $\mathbf{v}_k$ le site dont la composante connexe est calculée
Données :
– Classes gestionnaire des classes d'équivalence du bord
– CoinsDuSite un ensemble de coins par site
– FileDeSites la file des sites ayant des coins à traiter
Sorties :
– Mise à jour des CoinsDuSite et de la FileDeSites

```
1  Algorithme
2      pour chaque classe d'équivalence C de Classes faire
3          si les portions de bord associées à C ne contiennent pas de coin alors
4              Soit T_p la face d'indice minimal ayant une portion de bord dans C ;
5              Soit v_ℓ tel que Ω_{ℓ|S} est adjacente à Ω_{k|S} par ce bord ;
6              Soit c le faux coin (T_p,v_k,v_ℓ,0) ;
7              si c ∈ CoinsDuSite[v_k] alors
8                  CoinsDuSite[v_k] ← CoinsDuSite[v_k] \ c ;
9              sinon
10                 si CoinsDuSite[v_ℓ] est vide alors
11                     Ajouter v_ℓ à la FileDeSites ;
12                 CoinsDuSite[v_ℓ] ← CoinsDuSite[v_ℓ] ∪ c ;
13     Réinitialiser Classes pour le calcul de la prochaine composante connexe ;
```

Algorithme 3.7: `propagation_bords_sans_coins(`$\mathbf{v}_k$`)` : propagation de faux coins lorsqu'une boucle du bord de la composante connexe calculée pour $\mathbf{v}_k$ n'a pas de coins.

Entrées :
- $\mathbf{V} = \{\mathbf{v}_k\}_k$ un ensemble de sites
- $\mathcal{S}$ une surface

Sorties :
- Diagramme de Voronoï de $\{\mathbf{v}_k\}_k$ restreint à $\mathcal{S}$

Algorithme
RVD $\leftarrow \emptyset$; // Diagramme de Voronoï Restreint
FileDeFaces $\leftarrow \emptyset$;
FileDeSites $\leftarrow \emptyset$;
pour chaque *face* T_p *de* $\mathcal{S}$ **faire**
  SiteDeLaFace$[T_p] \leftarrow 0$; // dernier site traité pour une face
  BordsDeLaFace$[T_p] \leftarrow \emptyset$; // un ensemble de bords par face
pour chaque *site* $\mathbf{v}_k$ *de* $\mathbf{V}$ **faire**
  CoinsDuSite$[\mathbf{v}_k] \leftarrow \emptyset$; // un ensemble de coins par site
pour chaque *face* T_0 *de* $\mathcal{S}$ **faire**
  si T_p *n'a pas été marquée comme traitée* **alors**
    Soit $\mathbf{v}_0$ le site le plus proche du barycentre de T_0 ;
    CoinsDuSite$[\mathbf{v}_0] = \{(T_0,\mathbf{v}_0,0,0)\}$;
    Ajouter $\mathbf{v}_0$ dans la FileDeSites ;
    tant que *la* FileDeSites *n'est pas vide* **faire**
      Extraire $\mathbf{v}_k$ de la FileDeSites ;
      tant que CoinsDuSite$[\mathbf{v}_k]$ *n'est pas vide* **faire**
        Extraire $c = (T_{init},\mathbf{v}_k,\mathbf{v}_{l_1},\mathbf{v}_{l_2})$ de CoinsDuSite$[\mathbf{v}_k]$;
        SiteDeLaFace$[T_{init}] \leftarrow \mathbf{v}_k$;
        BordsDeLaFace$[T_{init}] \leftarrow \emptyset$;
        Ajouter T_{init} dans la FileDeFaces ;
        tant que *la* FileDeFaces *n'est pas vide* **faire**
          Extraire T_p de la FileDeFaces ;
          // Algorithme 3.1
          $T_p \cap \Omega_k \leftarrow$ `intersection_face_cellule`$(T_p,\mathbf{v}_k)$;
          RVD $\leftarrow$ RVD $\cup (T_p \cap \Omega_k)$;
          `propagation_triangles`$(T_p \cap \Omega_k)$; // Algorithme 3.4
          `propagation_coins`$(T_p \cap \Omega_k)$; // Algorithme 3.5
          `reconstruire_bords`$(T_p \cap \Omega_k)$; // Algorithme 3.6
          si $(T_p,\mathbf{v}_k,0,0) \in$ CoinsDuSite$[\mathbf{v}_k]$ **alors**
            CoinsDuSite$[\mathbf{v}_k] =$ CoinsDuSite$[\mathbf{v}_k] \setminus (T_p,\mathbf{v}_k,0,0)$;
      `propagation_bords_sans_coins`$(\mathbf{v}_k)$; // Algorithme 3.7

Algorithme 3.8: Calcul du diagramme de Voronoï restreint par composantes connexes des cellules restreintes.

3.5.3 Complexité et performances

Analyse dans le pire des cas

Tout d'abord l'Algorithme 3.4 réalise la propagation sur les triangles. Il doit traiter chaque arête de l'intersection réalisée pour chaque couple valide, en réalisant un test en $O(1)$ sur chacune d'entre elles. Ainsi, cet algorithme est donc dans le pire des cas en $O(n)$. Le stockage réalisé pour cet algorithme correspond à un site par face, il est donc en $O(m)$.

Pour la gestion des coins par l'Algorithme 3.5 et l'Algorithme 3.7, le nombre de coins stockés dans le pire des cas est $O(mn)$. En effet, en utilisant la caractéristique d'Euler d'une partition du plan sur la partition d'une face réalisée par le diagramme de Voronoï restreint, il est possible de démontrer que le nombre de coins est alors dans le pire cas $O(n)$ par face. Les faux coins traités par l'Algorithme 3.7 ne peuvent également pas être plus de n par face. Chaque coin est traité trois fois au cours de l'algorithme global, une fois pour chaque composante connexe d'une cellule adjacente. Les faux coins sont traités deux fois. Pour chaque coin, il est nécessaire de tester s'il fait partie de l'ensemble de coins d'un site, ce qui demande au pire un temps $O(\log(mn)) = O(\log m + \log n)$ dans le cas où le site possède $O(mn)$ coins dans son ensemble. Effacer un coin d'un ensemble ou l'ajouter requiert également ce temps. La gestion des coins prend donc au total sur le calcul du diagramme de Voronoï restreint un temps $O(mn(\log m + \log n))$. En espace, au pire, l'espace nécessaire est de $O(mn)$. Cette complexité dans le pire des cas exagère fortement la complexité constatée empiriquement.

Enfin la reconstruction des bords effectuée par l'Algorithme 3.6 sera effectuée pour chaque couple T_p,$\mathbf{v}_k$ valide. Pour déterminer l'ensemble des extrémités de bord, une recherche linéaire est effectuée sur chaque sommet de $T_p \cap \Omega_k$, ce qui prend au pire un temps $O(n)$. Une fois les extrémités déterminées, les portions de bord sont encodées par leurs deux extrémités. Une cellule de Voronoï étant convexe, une arête de T_p ne peut donc pas contenir plus de deux extrémités, car sinon, deux de ces extrémités seraient reliées par un segment qui ne serait pas contenu dans la cellule. Il ne peut donc y avoir au maximum que six extrémités pour une face T_p, et donc uniquement trois portions de bord. Le stockage des portions de bord pour l'ensemble des faces de $\mathcal{S}$ occupe donc un espace de $O(m)$. Pour chaque couple valide, nous réalisons donc au pire six recollements de bords. Ce recollement, comme décrit par Cormen *et al.* [20, Section 21.3] est effectué en un temps $O(\alpha(b))$ où b est le nombre de portions du bord de la classe, et α est l'inverse de la fonction d'Ackermann. Cette fonction a une croissante si lente que pour $b = 10^{80}$, $\alpha(b) = 4$. Dans le pire des cas, nous avons $b = m$. Ainsi, pour chaque couple, la reconstruction des bords nécessite un temps $O(\alpha(m))$.

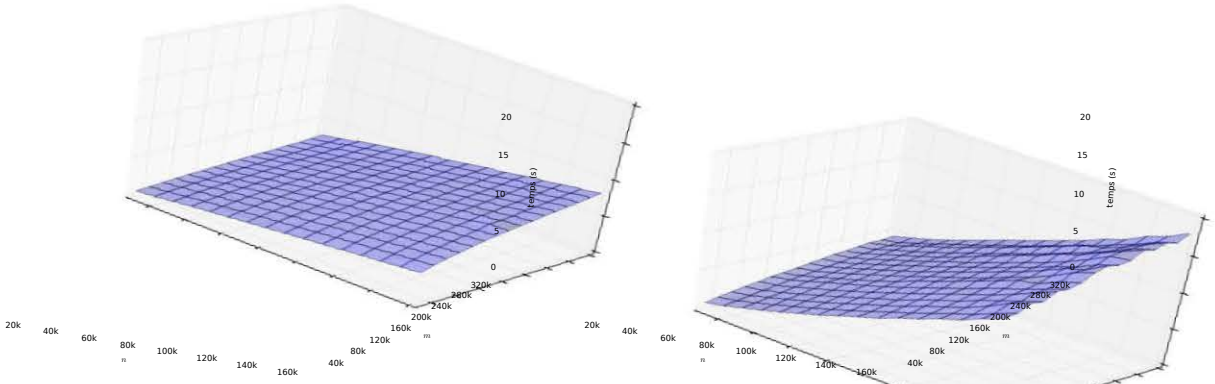
La complexité en temps de l'Algorithme 3.8 est donc de

$$O(mn^2 \log n) + O(mn(\log m + \log n) + O(mn\alpha(m)) = O(mn(n \log n + \log m)) \quad (3.5.1)$$

En terme d'espace la structure prenant le plus de place dans le pire des cas est l'ensemble de coins stockés par site, qui peut dans le pire des cas être de $O(mn)$.

Performances pratiques

Dans la pratique, de même que pour l'Algorithme 3.3, le graphe donné en Figure 3.16 pour l'Algorithme 3.8 semble toujours indiquer des performances meilleures que la complexité



(a) priorité aux triangles (b) priorité aux sites

FIGURE 3.16 – Comparaison des performances entre l'Algorithme 3.3 et l'Algorithme 3.8. Les tests ont été réalisés sur le modèle du David pour tous les couples (m,n) où le nombre de faces m varie de 20k à 320k par pas de 20k et le nombre de sites n varie de 10k à 160k par pas de 10000. L'Algorithme 3.8 semble avoir une complexité similaire à celle de l'Algorithme 3.3 , même si la vitesse d'exécution est de deux à quatre fois plus faible.

théorique dans le pire des cas. Le calcul par composante connexe des cellules restreintes reste cependant plus lent que le calcul par triangle, d'un facteur entre deux et quatre dans nos expérimentations.

Pour ce qui est de la complexité en espace, la structure enregistrant l'ensemble des coins du bord de la zone calculée est une structure dont la taille varie dynamiquement au cours de l'algorithme. Dans le pire des cas, nous avions évalué sa taille comme étant en $O(mn)$. Empiriquement, cependant, la taille de cette structure semble être relativement indépendante du nombre de faces du maillage, comme le montre la Figure 3.17.

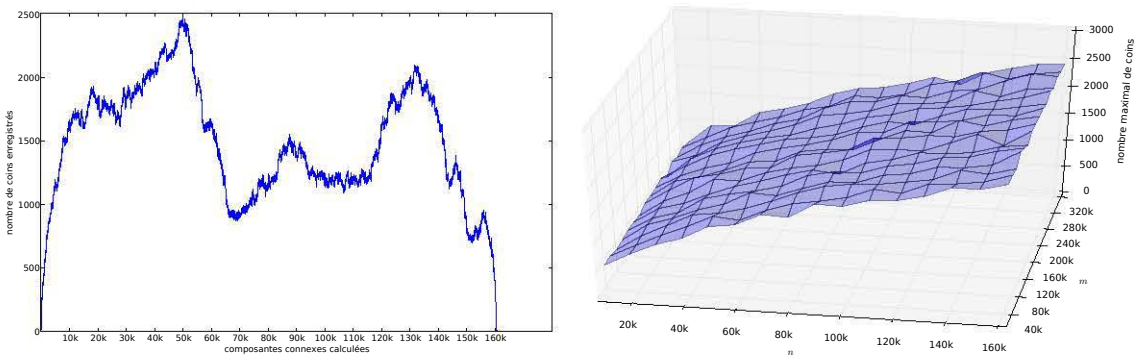
3.6 Conclusion et perspectives

3.6.1 Contributions

Dans ce chapitre, nous avons élaboré deux algorithmes pour le calcul du diagramme de Voronoï d'un ensemble de sites restreint à une surface maillée. Ces deux algorithmes diffèrent par l'ordre dans lequel les cellules de Voronoï restreintes sont calculées. Le premier algorithme, décrit en Section 3.4, améliore l'algorithme proposé par Yan *et al.* [118] en supprimant la nécessité de stocker pour chaque face de la surface l'ensemble des sites déjà traités pour cette face. Cette modification permet à la fois de limiter l'espace mémoire utilisé et d'accélérer le calcul lorsque le nombre de sites croît par rapport au nombre de faces de la surface. Le second algorithme, proposé en Section 3.5, réordonne le calcul pour permettre d'extraire directement au cours du calcul les différentes composantes connexes des cellules de Voronoï restreintes. Cette variante est plus coûteuse à la fois en espace mémoire et en temps que le premier algorithme, mais permet d'éviter d'avoir à enregistrer le diagramme de Voronoï restreint complet pour en extraire les différentes composantes connexes des cellules de Voronoï restreintes.

3.6.2 Perspectives

Nos algorithmes ne proposent pas pour l'instant de calculer en précision exacte un diagramme de Voronoï restreint. En cas d'erreurs numériques, il peut arriver que la combina-



(a) (b)

FIGURE 3.17 – Étude empirique de la taille de l'ensemble des coins stockés. Au cours du calcul, la taille de l'ensemble de coins évolue dynamiquement. En (a), nous avons relevé la taille de l'ensemble de coins après chaque composante connexe calculée, sur le modèle du David, avec 320k faces, et 160k sites. En (b), nous avons mesuré la taille maximale de l'ensemble de coins au cours du calcul, en faisant varier la résolution du modèle du David de 20k à 320k par palier de 20k, et le nombre de sites de 10k à 160k par palier de 10k.

toire obtenue lors du calcul d'une cellule de Voronoï restreinte soit fausse et que des couples face – site invalides soient propagés. La propagation de l'Algorithme 3.3 est assez simple, et ce genre de situation peut au pire faire que quelques faces de la surface ne seront pas traitées. Pour l'Algorithme 3.8, la propagation est fondée sur la combinatoire du diagramme et les coins. En cas d'erreur numérique, lorsque la combinatoire est faussée, il se peut que la face enregistrée pour représenter un coin soit mauvaise, lorsque le coin est proche d'une arête de la surface. Dans cette situation, deux composantes connexes voisines ne trouvent pas la même face pour un même coin, ce qui peut alors bloquer la propagation dans une boucle infinie. Pour réaliser le calcul d'un diagramme de Voronoï restreint, il est nécessaire d'utiliser des outils permettant de réaliser le calcul de certains prédicats en précision arbitraire, comme ceux proposés par Pion [84]. En utilisant ces méthodes, il est alors possible de s'assurer que la combinatoire calculée est correcte. Il reste alors à gérer les cas particuliers, comme lorsqu'une arête de la surface est inclue dans la face de Voronoï séparant deux sites. Ces cas particuliers doivent pouvoir être traités en utilisant une méthode de perturbation symbolique similaire à celle proposée par Edelsbrunner et Mücke [35].

L'Algorithme 3.8 permet de construire les composantes connexes des cellules de Voronoï restreintes ainsi que leur bord. Il est alors possible de déterminer si les conditions sont respectées pour s'assurer que la triangulation de Delaunay restreinte correspondant à un diagramme de Voronoï restreint est homéomorphe à la surface segmentée par le diagramme. Si nous pouvons détecter les erreurs, nous ne proposons pas de solution pour y remédier. Il serait alors envisageable d'élaborer un procédé d'insertion automatique de sites en fonction des anomalies détectées pour s'assurer que les conditions nécessaires sont respectées, comme le font Yan *et al.* [118].

Chapitre 4

Optimisation d'un diagramme de Voronoï restreint

4.1 Motivations

Nous aborderons ici la question de savoir comment optimiser un diagramme de Voronoï restreint. Cette notion apparaît naturellement dans un grand nombre de problèmes de classification en statistiques, en traitement d'images, et pour des problèmes de remaillage que nous approfondirons par la suite. De nombreuses applications sont données par Du *et al.* [28]. Dans le Chapitre 3, nous avions utilisé la métaphore d'un ensemble de casernes de pompiers et des zones d'une ville dont elles étaient responsables pour décrire un diagramme de Voronoï. Nous nous plaçons désormais dans la peau d'une personne cherchant à déterminer la meilleure répartition possible des casernes, pour minimiser le temps d'arrivée sur les lieux d'un sinistre. Il s'agit bien là d'un problème d'optimisation. Le problème est tout d'abord défini par un objectif : minimiser la distance entre une caserne, et tous les points de la zone dont elle est responsable. Pour atteindre cet objectif, nous avons le droit de jouer sur un certain nombre de variables, qui correspondent ici aux positions des casernes. Nous n'avons cependant pas le choix du nombre de casernes à placer, ni de la géométrie des rues.

Chaque adresse $\mathbf{x}_i$ de la ville est placée sous la responsabilité de la caserne $\mathbf{v}_k$ la plus proche, par rapport à une distance que nous noterons $d(\mathbf{x}_i,\mathbf{v}_k)$. L'ensemble des adresses dont une caserne est responsable est sa région de Voronoï notée Ω_k. Imaginons que nous commencions par un ensemble de casernes réparties au hasard. Si nous déplaçons une caserne, ce déplacement aura deux effets. Tout d'abord, les distances entre la caserne et les adresses dont elle est responsable vont être modifiées. Il est donc nécessaire de trouver une direction dans laquelle déplacer la caserne pour faire en sorte que ces distances diminuent. Le second effet sera que le diagramme de Voronoï se verra lui-même modifié, ce qui fera que certaines adresses dont la caserne était responsable seront attribuées à une autre caserne, et inversement que la caserne deviendra responsable d'un nouvel ensemble d'adresses. Ce deuxième effet est plus compliqué à prédire et à prendre en compte dans le processus d'optimisation. Il est en effet nécessaire de déterminer de quelle façon variera le diagramme de Voronoï en fonction de la direction dans laquelle la caserne sera déplacée. Intuitivement, ce deuxième effet n'affectera que les adresses situées au bord de la région de Voronoï de la caserne, car ce sont elles qui risquent de passer sous la responsabilité de cette caserne ou d'une autre caserne.

De nombreux travaux ont déjà été réalisés pour aborder ce problème, et nous commencerons pas les détailler, avant d'aborder un problème plus général. Imaginons désormais que nous cherchions à placer des députés plutôt que des casernes. Comme précédemment, chaque citoyen sera représenté par le député le plus proche de son adresse. Par contre, le mécontentement d'un citoyen ne sera plus mesuré par la distance qui le sépare de son député, mais par la différence d'opinion politique entre lui et son député. Nous chercherons alors à optimiser à la fois, pour chaque député, sa position et son opinion politique.

Ce chapitre aborde le cas d'un diagramme de Voronoï restreint à une surface. La surface est définie par l'ensemble des positions de ses sommets $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{l}$ dans $\mathbb{R}^3$ et par les triangles $\mathbf{T} = \{T_p\}_{p=1}^{m}$ reliant ces sommets entre eux. Le diagramme de Voronoï est défini par l'ensemble des positions des sites $\mathbf{V} = \{\mathbf{v}_k\}_{k=1}^{n}$. Les sites sont positionnés dans $\mathbb{R}^3$ également, et ne sont pas nécessairement placés sur la surface $\mathcal{S}$. Le problème d'optimisation est formulé sous la forme de la recherche du minimum d'une fonction. Dans le premier exemple, la fonction objectif minimisée est :

$$\mathscr{F}(\mathbf{V}) = \sum_{\mathbf{v}_k \in \mathbf{V}} \int_{\Omega_{k|\mathcal{S}}} \|\mathbf{v}_k - \mathbf{x}\|^2 \mathrm{d}\mathbf{x},$$

où $\Omega_{k|\mathcal{S}}$ est la cellule de Voronoï de $\mathbf{v}_k$ restreinte à $\mathcal{S}$. Cette fonction mesure la somme pour chaque caserne $\mathbf{v}_k$ de la somme pour chaque citoyen $\mathbf{x}$ sous sa responsabilité de la distance au carré à la position de la caserne. Dans le second exemple, nous ajoutons une fonction f mesurant l'opinion politique d'un citoyen, et un ensemble de variables $\tilde{\mathbf{F}} = \{\tilde{f}_k\}_{k=1}^{n}$. $\tilde{f}_k$ correspond à l'opinion politique du député placé à la position $\mathbf{v}_k$. La fonction objectif minimisée est alors :

$$\mathscr{F}(\mathbf{V},\tilde{\mathbf{F}}) = \sum_{\mathbf{v}_k \in \mathbf{V}} \int_{\Omega_{k|\mathcal{S}}} \|\tilde{f}_k - f(\mathbf{x})\|^2 \mathrm{d}\mathbf{x}.$$

Pour pouvoir minimiser $\mathscr{F}$, il est utile de pouvoir calculer comment $\mathscr{F}$ varie lorsque les variables $\mathbf{V}$ ou $\mathbf{X}$ varient. Il s'agit donc de calculer son *gradient*. Ce gradient, lorsqu'il existe, donne la direction dans laquelle les variables doivent être modifiées pour faire diminuer $\mathscr{F}$ le plus possible. Cette direction peut être améliorée en calculant également comment le gradient varie, c'est à dire la *Hessienne*, correspondant aux dérivées secondes. Nous ne nous intéresserons cependant pas ici à ce calcul.

4.2 Contributions

La Section 4.3 ne contient pas de contribution propre. Les résultats qui y sont présentés seront néanmoins utiles par la suite à la fois car ces travaux sont très similaires à la contribution que nous réalisons dans ce chapitre, et parce qu'ils seront directement appliqués dans le Chapitre 5 à un problème d'ajustement de surfaces.

Notre contribution ici consiste à proposer une méthode pour approximer une fonction quelconque $f : \mathcal{S} \to \mathbb{R}$ définie sur une surface. L'approximation est réalisée en utilisant un ensemble de sites $\mathbf{V} = \{\mathbf{v}_k\}_k$, et en approximant f par une valeur constante $\tilde{f}_k$ à l'intérieur de la cellule de Voronoï $\Omega_{k|\mathcal{S}}$ de $\mathbf{v}_k$ restreinte à $\mathcal{S}$. Nous optimisons alors les positions des sites et l'ensemble $\tilde{\mathbf{F}} = \{\tilde{f}_k\}_k$ des valeurs d'approximation de leurs cellules restreintes en minimisant la fonction objectif

$$\mathscr{F}(\mathbf{V},\tilde{\mathbf{F}}) = \sum_{\mathbf{v}_k \in \mathbf{V}} \int_{\Omega_{k|\mathcal{S}}} \|f(\mathbf{x}) - \tilde{f}_k\|^2 \mathrm{d}\mathbf{x}. \tag{4.2.1}$$

Nous montrerons ainsi en Section 4.4 comment calculer le gradient de cette fonction objectif, et proposerons une heuristique proche d'une descente de gradient pour trouver un minimum local de $\mathscr{F}$. Ces résultats n'ont pas à l'heure actuelle fait l'objet d'une publication.

4.3 Fondements et état de l'art

Nous évoquerons ici les avancées existantes dans le domaine de l'optimisation de diagrammes de Voronoï. La plupart de ces avancées ont été réalisées pour obtenir des *diagrammes de Voronoï barycentriques*, qui apparaissent dans de très nombreuses applications, comme le montrent Du *et al.* [28]. Nous commencerons donc par définir un diagramme de Voronoï barycentrique, et les approches mises au point pour en obtenir un. Nous aborderons ensuite les applications plus spécifiques de ce type de diagramme de Voronoï au cas du remaillage, à l'aide des diagrammes de Voronoï restreints définis en Chapitre 3.

4.3.1 Notations

Nous utiliserons les même notations que dans le Chapitre 3 pour décrire un diagramme de Voronoï restreint.

Diagramme de Voronoï restreint

$\mathbf{V}$	ensemble des sites du diagramme de Voronoï restreint
$\mathbf{v}_k$	site quelconque de $\mathbf{V}$
Ω_k	cellule de Voronoï de $\mathbf{v}_k$ dans $\mathbb{R}^3$
$\mathcal{S}$	surface sur laquelle calculer le diagramme, décrite par un maillage
$\Omega_{k\|\mathcal{S}}$	cellule de Voronoï de $\mathbf{v}_k$ restreinte à $\mathcal{S}$
$\mathbf{X}$	ensemble des sommets de $\mathcal{S}$
$\mathbf{x}_i$	sommet de $\mathcal{S}$
$\mathbf{T}$	ensemble des faces de $\mathcal{S}$
T_p	face de $\mathcal{S}$
$\mathbf{x}$	point quelconque de $\mathcal{S}$ ou du domaine dans lequel le diagramme est calculé
$\mathbf{c}$	sommet d'un diagramme de Voronoï restreint

Nous introduisons également quelques notations supplémentaires pour la Section 4.4, où nous optimisation un diagramme de Voronoï restreint pour l'échantillonnage d'une fonction.

Échantillonnage d'une fonction

$\tilde{\mathbf{F}}$	ensemble des valeurs d'approximation associées aux sites
$\tilde{f}_k$	valeur d'approximation du site $\mathbf{v}_k$
$\delta^s_{\mathbf{v}_k}$	pas d'optimisation du site $\mathbf{v}_k$ à l'itération s
$\delta^s_{\mathbf{v}_k}$	pas d'optimisation de la valeur d'approximation $\tilde{f}_k$ à l'itération s

4.3.2 Diagrammes de Voronoï barycentriques

Définition

La plupart des travaux sur l'optimisation de diagrammes de Voronoï ont été réalisés dans le but d'obtenir des diagrammes de Voronoï barycentriques. Lorsque les sites évoluent dans $\mathbb{R}^3$ et que le domaine est également une portion de $\mathbb{R}^3$, le barycentre d'une région V_k peut

être défini comme le point $\mathbf{g}_k$ tel que :

$$\mathbf{g}_k = \frac{\int_{V_k} \mathbf{x}\mathrm{d}\mathbf{x}}{\int_{V_k} \mathrm{d}\mathbf{x}}. \tag{4.3.1}$$

Du *et al.* [28] prouvent que la fonction

$$\mathscr{F}(\mathbf{V},\{V_k\}) = \sum_{k=1}^{n} \int_{V_k} \|\mathbf{v}_k - \mathbf{x}\|^2 \mathrm{d}\mathbf{x},$$

où $\{V_k\}_{k=1}^n$ est un ensemble de régions et $\mathbf{V} = \{\mathbf{v}_k\}_{k=1}^n$ un ensemble de sites, ne peut être minimisée que lorsque pour tout k, la région V_k correspond à la cellule de Voronoï Ω_k de $\mathbf{v}_k$, et le site $\mathbf{v}_k$ est situé au barycentre $\mathbf{g}_k$ de V_k. La propriété du barycentre qui sous-tend cette dernière condition est qu'il vérifie également la relation :

$$\mathbf{g}_k = \underset{\mathbf{g}\in V_k}{\operatorname{argmin}} \int_{V_k} \|\mathbf{g} - \mathbf{x}\|^2 \mathrm{d}\mathbf{x}. \tag{4.3.2}$$

Par rapport au problème du placement des casernes évoqué en introduction, ce résultat signifie donc que si nous avons le choix à la fois de la position des casernes, et de la portion de la ville associée à chaque caserne, le placement optimal ne pourra être réalisé que lorsque la portion de la ville correspond à la cellule de Voronoï du site correspondant à la caserne, et lorsque ce site est situé au barycentre de sa cellule de Voronoï. Il s'agit donc bien là d'un diagramme de Voronoï barycentrique. Nous utiliserons dans la suite la fonction objectif simplifiée :

$$\mathscr{F}(\mathbf{V}) = \sum_{\mathbf{v}_k\in\mathbf{V}} \int_{\Omega_k} \|\mathbf{x} - \mathbf{v}_k\|^2 \mathrm{d}\mathbf{x}. \tag{4.3.3}$$

À partir de cette fonction objectif, le problème de l'obtention d'un diagramme de Voronoï barycentrique se résume à la recherche de l'ensemble de sites $\mathbf{V}$ tel que $\mathscr{F}$ soit minimale. En pratique, l'obtention d'un minimum global est très difficile. De plus, un tel minimum n'est pas unique, étant donné qu'en intervertissant les positions de deux sites, la valeur de la fonction objectif ne change pas. Il y a donc au moins autant de minima globaux que de permutations des sites. La plupart des travaux se sont donc concentrés sur l'obtention de minima locaux. Lorsque nous parlerons de minimiser une fonction objectif, il s'agira donc d'en trouver un minimum local. Plusieurs méthodes existent pour la résolution de ce problème de minimisation. La plus couramment utilisée est probablement la méthode de Lloyd [68].

Algorithme de Lloyd

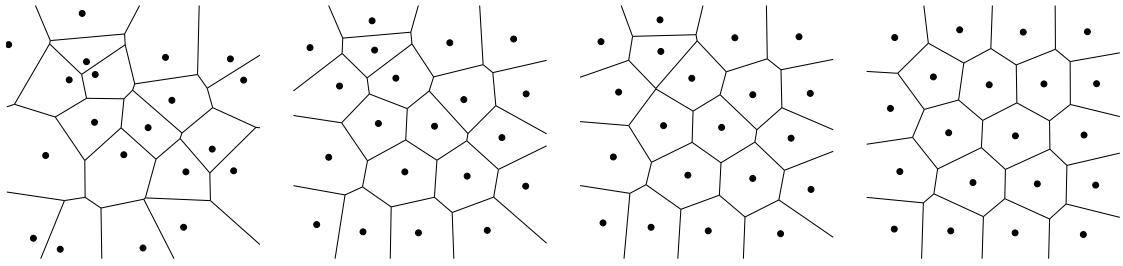
Cet algorithme est initialisé à un ensemble quelconque de sites, qui sont ensuite itérativement repositionnés au barycentre de leurs cellules de Voronoï. Son fonctionnement est synthétisé dans l'Algorithme 4.1. Sa convergence vers un ensemble de sites minimisant l'Équation 4.3.3 a été étudiée par Du *et al.* [27]. La Figure 4.1 illustre ce procédé sur un domaine carré de $\mathbb{R}^2$.

Cet algorithme réalise en réalité une descente de gradient, car comme nous le montrerons dans un instant, le vecteur $\mathbf{v}_k - \mathbf{g}_k$ où $\mathbf{g}_k$ est le barycentre de la cellule de Voronoï restreinte de $\mathbf{v}_k$ a la même direction que le gradient de $\mathscr{F}$ par rapport à $\mathbf{v}_k$. Nocedal et Wright [75, Chapitre 3] explique qu'une descente de gradient converge au pire linéairement lorsque la

Entrées :
– Ω le domaine dans lequel le diagramme est calculé
– $\mathbf{V}$ les positions initiales des sites du diagramme
Sorties :
– $\mathbf{V}^*$ l'ensemble des sites optimisés

Algorithme
  $\mathbf{V}^* \leftarrow \mathbf{V}$;
  tant que *le critère de convergence n'est pas satisfait* **faire**
    $\{\Omega_k^*\}_{k=0}^n \leftarrow$ `diagramme_voronoï(`Ω`,`$\mathbf{V}^*$`)` ;
    pour chaque *cellule de Voronoï* Ω_k^* *d'un site* $\mathbf{v}_k^* \in \mathbf{V}^*$ **faire**
      $\mathbf{v}_k^* \leftarrow$ `barycentre(`Ω_k`)` ;
  renvoyer $\mathbf{V}^*$;

Algorithme 4.1: `Lloyd(`Ω`,`$\mathbf{V}$`)` : optimisation de l'ensemble de sites $\mathbf{V}$ pour obtenir un diagramme de Voronoï barycentrique du domaine Ω.



(a) Initialisation (b) Itération 1 (c) Itération 2 (d) Itération 50

FIGURE 4.1 – Itérations de Lloyd pour faire évoluer un diagramme de Voronoï quelconque de 20 sites vers un diagramme de Voronoï barycentrique. Ici l'état initial ainsi que le diagramme après quelques itérations sont illustrés.

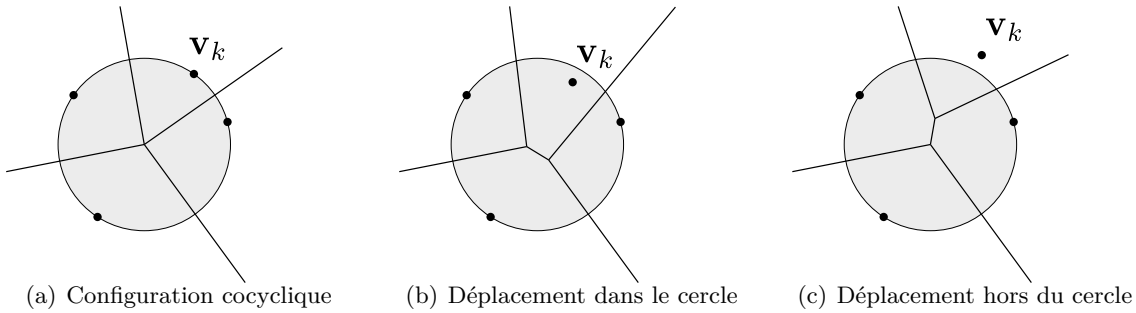
fonction objectif est deux fois dérivable, et que les pas sont bien choisis. Ces conditions ne sont pas respectées ici, car la fonction $\mathscr{F}$ n'est pas tout à fait partout deux fois dérivable, en particulier lorsque deux sites coïncident, et que le diagramme de Voronoï est mal défini. De plus une convergence linéaire reste relativement lente dans le domaine de l'optimisation numérique.

Gradient de $\mathscr{F}$

Nous utiliserons les notations proposées par Minka [73] pour les dérivées, qui permettent de travailler plus facilement avec des fonctions multivariées à valeurs vectorielles. Selon ces notations, lorsque f est une fonction scalaire, sa dérivée selon la position d'un site $\mathbf{v}_k$ est le vecteur ligne :

$$\frac{\partial f}{\partial \mathbf{v}_k} = \left(\frac{\partial f}{\partial v_{k,1}}, \frac{\partial f}{\partial v_{k,2}}, \frac{\partial f}{\partial v_{k,3}} \right)$$

où les $v_{k,l}$ sont les coordonnée du site $\mathbf{v}_k$. Pour une fonction $\mathbf{f}$ à valeur dans $\mathbb{R}^d$, de fonctions partielles $\{\mathbf{f_i}\}_{i=1}^d$, la dérivée selon le site $\mathbf{v}_k$ est la matrice dont chaque ligne correspond à



(a) Configuration cocyclique (b) Déplacement dans le cercle (c) Déplacement hors du cercle

FIGURE 4.2 – Changement de la combinatoire du diagramme de Voronoï au niveau d'une configuration cocyclique. Selon que le site $\mathbf{v}_k$ est déplacé à l'intérieur du cercle ou à l'extérieur, sa cellule de Voronoï n'a pas le même nombre d'arêtes.

la dérivée de l'une des fonctions partielles par rapport à $\mathbf{v}_k$. Il s'agit donc de la matrice jacobienne de $\mathbf{f}$ par rapport aux coordonnées de $\mathbf{v}_k$.

$\mathscr{F}$ a longtemps été considérée comme une fonction continue, et dérivable uniquement lorsque les positions des sites sont telles qu'une modification infinitésimale de leur position ne change pas la combinatoire du diagramme [52]. Dans le cas de diagrammes de Voronoï de $\mathbb{R}^2$, les changements de combinatoire apparaissent lorsque quatre sites sont cocycliques. Dans ce cas, comme le montre la Figure 4.2, la cellule de Voronoï d'un site $\mathbf{v}_k$ parmi les sites cocycliques n'aura pas le même nombre d'arêtes et de sommets selon que le site $\mathbf{v}_k$ est déplacé vers l'intérieur ou l'extérieur du cercle contenant les quatre sites. Ainsi, la courbe correspondant au bord de la cellule de $\mathbf{v}_k$ n'est pas une fonction C^1 de la position des sites. Malgré ce phénomène, Cortes *et al.* [21] ont montré que $\mathscr{F}$ était tout de même une fonction C^1, et Liu *et al.* [67] qu'elle était même C^2 lorsque le domaine dans lequel le diagramme est défini est convexe et compact.

Le gradient de $\mathscr{F}$ a été étudié par Iri *et al.* [52]. Les calculs nécessaires pour arriver à ce résultat sont détaillés par Asami [2]. Ces calculs peuvent en réalité être simplifiés en utilisant une généralisation au cas multivarié de la règle de Leibniz permettant de calculer la dérivée d'une intégrale dans laquelle le terme intégré et le domaine de l'intégrale varient en fonction des variables utilisées pour dériver. Nous reviendrons plus en détails en Section 4.4.1 sur cette règle, également appelée le *théorème du transport de Reynolds*. Le gradient de $\mathscr{F}$, lorsqu'il existe, est donc :

$$\begin{aligned}\frac{\partial \mathscr{F}}{\partial \mathbf{V}}(\mathbf{V}) &= \frac{\partial}{\partial \mathbf{V}}\left[\sum_{\mathbf{v}_k\in\mathbf{V}}\int_{\Omega_k}\|\mathbf{x}-\mathbf{v}_k\|^2\mathrm{d}\mathbf{x}\right]\\ &= \sum_{\mathbf{v}_k\in\mathbf{V}}\frac{\partial}{\partial \mathbf{V}}\left[\int_{\Omega_k}\|\mathbf{x}-\mathbf{v}_k\|^2\mathrm{d}\mathbf{x}\right]\\ &= \sum_{\mathbf{v}_k\in\mathbf{V}}\left[\int_{\Omega_k}\frac{\partial}{\partial \mathbf{V}}(\|\mathbf{x}-\mathbf{v}_k\|^2)\mathrm{d}\mathbf{x}+\int_{\partial\Omega_k}\|\mathbf{x}-\mathbf{v}_k\|^2\mathbf{n}_{\partial\Omega_k}(\mathbf{x})^t\left(\frac{\partial \mathbf{x}}{\partial \mathbf{V}}\right)\mathrm{d}\mathbf{x}\right]\end{aligned} \tag{4.3.4}$$

où $\partial\Omega_k$ correspond au bord de Ω_k, $\frac{\partial \mathbf{x}}{\partial \mathbf{V}}$ à la jacobienne du point $\mathbf{x}$ de $\partial\Omega_k$ par rapport à $\mathbf{V}$ et $\mathbf{n}_{\partial\Omega_k}(\mathbf{x})$ la normale de $\partial\Omega_k$ en $\mathbf{x}$. Ici, le gradient pour une cellule donnée contient deux termes. Un terme interne correspondant à une intégrale sur l'intérieur de la cellule Ω_k et un terme externe correspondant à une intégrale sur le bord $\partial\Omega_k$. Étant donné un

couple de sites adjacents $\mathbf{v}_1$ et $\mathbf{v}_2$, nous noterons $\partial\Omega_{1,2}$ l'intersection $\partial\Omega_1 \cap \partial\Omega_2$ des bords de ces deux cellules, orientée dans le sens positif selon Ω_1 et respectivement $\partial\Omega_{2,1}$ cette même intersection orientée dans le sens positif selon Ω_2 et donc dans le sens opposé à $\partial\Omega_{1,2}$. Nous avons ainsi $\mathbf{n}_{\partial\Omega_{1,2}}(\mathbf{x}) = -\mathbf{n}_{\partial\Omega_{2,1}}(\mathbf{x})$. Un point $\mathbf{x}$ appartenant à $\partial\Omega_1 \cap \partial\Omega_2$ est à égale distance de $\mathbf{v}_1$ et $\mathbf{v}_2$. Nous avons donc :

$$\int_{\partial\Omega_{1,2}} \|\mathbf{x}-\mathbf{v}_1\|^2 \mathbf{n}_{\partial\Omega_{1,2}}(\mathbf{x})^t \left(\frac{\partial \mathbf{x}}{\partial \mathbf{V}}\right) \mathrm{d}\mathbf{x} = -\int_{\partial\Omega_{2,1}} \|\mathbf{x}-\mathbf{v}_2\|^2 \mathbf{n}_{\partial\Omega_{2,1}}(\mathbf{x})^t \left(\frac{\partial \mathbf{x}}{\partial \mathbf{V}}\right) \mathrm{d}\mathbf{x} \tag{4.3.5}$$

Lorsqu'un bord d'une cellule de Voronoï correspond à un bord du domaine, les variations des points du bord sont alors orthogonales à la normale du bord, et la portion correspondante du terme externe s'annule. Nous détaillerons ce point en Section 4.4.2. Ainsi, pour le gradient de $\mathscr{F}$, l'Équation 4.3.5 et cette dernière remarque font que la somme du terme externe pour toute les cellules est nulle, car les termes apparaissant pour une portion de bord commune à deux cellules s'annulent. Nous avons donc :

$$\frac{\partial \mathscr{F}}{\partial \mathbf{V}}(\mathbf{V}) = \sum_{\mathbf{v}_k \in \mathbf{V}} \int_{\Omega_k} \frac{\partial}{\partial \mathbf{V}}(\|\mathbf{x}-\mathbf{v}_k\|^2)\mathrm{d}\mathbf{x}, \tag{4.3.6}$$

et le gradient par rapport à un site $\mathbf{v}_l$ devient :

$$\begin{aligned}\frac{\partial \mathscr{F}}{\partial \mathbf{v}_l}(\mathbf{V}) &= \sum_{\mathbf{v}_k \in \mathbf{V}} \int_{\Omega_k} \frac{\partial}{\partial \mathbf{v}_l}(\|\mathbf{x}-\mathbf{v}_k\|^2)\mathrm{d}\mathbf{x} \\ &= \int_{\Omega_k} -2(\mathbf{x}-\mathbf{v}_l)^t \mathrm{d}\mathbf{x} \\ &= 2m_l(\mathbf{v}_l - \mathbf{g}_l)^t,\end{aligned} \tag{4.3.7}$$

où m_l est l'aire ou le volume de Ω_l selon la dimension et $\mathbf{g}_l$ son barycentre. Descendre le gradient de $\mathscr{F}$ correspond donc bien à déplacer les sites dans la direction de leur barycentre, ce qui correspond bien à l'algorithme de Lloyd.

Optimisation par une méthode de Newton ou quasi-Newton

Lorsque $\mathscr{F}$ est deux fois dérivable, il existe des algorithmes de recherche de minimum qui ont une convergence quadratique. Plusieurs pistes ont été étudiées pour l'obtention d'un diagramme de Voronoï barycentrique.

Du et Emelianenko [26] proposent l'utilisation de la méthode de Newton pour la résolution du système d'équations non linéaires :

$$\frac{\partial \mathscr{F}}{\partial \mathbf{V}} = 0 \Leftrightarrow \forall k, \mathbf{v}_k = \mathbf{g}_k.$$

Pour mettre en œuvre cette méthode, il est nécessaire de calculer la dérivée seconde de $\mathscr{F}$, c'est à dire la dérivée pour tout k de $\mathbf{v}_k - \mathbf{g}_k$ par rapport à l'ensemble des sites. Un système linéaire faisant intervenir la hessienne ainsi obtenue est ensuite utilisé pour mettre à jour la position des sites. Cette méthode est expliquée plus en détails par Nocedal et Wright [75, Chapitre 3]. Si n est le nombre de sites, la hessienne est une matrice de taille $O(n^2)$. Même si cette matrice est en général creuse, elle peut être pleine lorsque tous les sites sont mutuellement voisins. Le stockage nécessaire à cette méthode peut donc devenir quadratique en n. La convergence de ce procédé est prouvée et quadratique [75, Théorème 3.5] lorsque l'ensemble initial de sites est proche de la configuration optimale et que $\mathscr{F}$ est suffisamment lisse. L'algorithme de Lloyd est donc utilisé pour se rapprocher de cette

configuration, avant de lancer la méthode de Newton. En effet, cette méthode cherche un point où le gradient de $\mathscr{F}$ s'annule, mais cette condition n'est pas suffisante pour obtenir un minimum local. Il est également possible de converger vers des points selle par exemple, qui fourniraient donc un mauvais résultat. L'aspect délicat de ce procédé est d'effectuer le passage entre l'algorithme de Lloyd et la méthode de Newton au bon moment. En lançant trop tôt la méthode de Newton la solution obtenue risque d'être instable, et en la lançant trop tard le bénéfice de l'accélération qu'elle induit est perdu.

Liu *et al.* [67] proposent l'utilisation d'une méthode quasi-Newton : L-BFGS [65]. Cette méthode a également une convergence quadratique, mais a l'avantage de ne pas stocker la matrice hessienne complète, dont la taille augmente de manière quadratique avec le nombre de sites. L-BFGS estime la matrice hessienne en utilisant les valeurs successives du gradient, et ne requiert qu'un stockage linéaire pour cette estimation. Ce procédé a été appliqué avec succès au remaillage de surfaces [118] ou de volumes [62]. C'est sur cette approche que se fonde la méthode présentée en Chapitre 5. Dans ce cadre, il est donc nécessaire de fournir au solveur la valeur de la fonction objectif et son gradient, dont l'expression est donnée par l'Équation 4.3.7.

4.3.3 Diagrammes de Voronoï restreints pour le remaillage

Nous avons abordé les diagrammes de Voronoï restreints en Chapitre 3. Nous verrons ici qu'ils peuvent être optimisés de la même façon que les diagrammes de Voronoï classiques, pour obtenir un diagramme de Voronoï barycentrique. Un tel diagramme est particulièrement intéressant pour le remaillage via la *triangulation de Delaunay restreinte*. Pour l'application spécifique du remaillage, certaines modifications de la fonction objectif classique ont été proposées. Ces modifications, ont permis d'améliorer la qualité des remaillages obtenus, mais ont rendu plus complexe le calcul du gradient de la fonction objectif. Nous détaillerons la méthode utilisée pour ces calculs, car il s'agit de celle que nous utiliserons en Chapitre 5 et qu'est une alternative à la méthode que nous proposerons en Section 4.4.

La définition d'un diagramme de Voronoï barycentrique est plus subtile dans le cas restreint. En effet, pour une portion V_k de $\mathcal{S}$ l'Équation 4.3.1 ne fournit en général pas un point de $\mathcal{S}$. D'autre part, la définition proposée par l'Équation 4.3.2 ne garantit ni l'existence ou l'unicité du point de V_k obtenu. Dans nos travaux, et ceux que nous allons présenter, les sites ne sont pas obligatoirement des points de $\mathcal{S}$, mais plus généralement des points de $\mathbb{R}^3$. Nous conservons cependant la définition du barycentre fournie par l'Équation 4.3.1. Un diagramme de Voronoï restreint barycentrique dans ce contexte n'aura donc en général pas ses sites situés sur $\mathcal{S}$, et certains sites peuvent éventuellement avoir une cellule de Voronoï restreinte vide. Ces sites ne servent alors plus dans l'optimisation et deviennent inutiles.

Triangulation de Delaunay restreinte

De même que la triangulation de Delaunay est duale au diagramme de Voronoï restreint, il est possible de définir le dual d'un diagramme de Voronoï restreint : la triangulation de Delaunay restreinte. L'utilisation de cette structure pour le remaillage a été introduite et étudiée par Edelsbrunner et Shah [36]. La triangulation de Delaunay restreinte associe à chaque cellule de Voronoï restreinte Ω_k un sommet, situé au niveau du site $\mathbf{v}_k$. Lorsque deux cellules de Voronoï restreintes Ω_1 et Ω_2 sont adjacentes, une arête est ajoutée entre les sommets $\mathbf{v}_1$ et $\mathbf{v}_2$. Enfin lorsque trois cellules de Voronoï restreintes Ω_1, Ω_2 et Ω_3 se rencontrent en un point de la surface, un triangle est ajouté entre les sites $\mathbf{v}_1$, $\mathbf{v}_2$ et $\mathbf{v}_3$.

(a) Initialisation (b) Diagramme optimisé (c) Triangulation associée

FIGURE 4.3 – Triangulation de Delaunay restreinte issue d'un diagramme de Voronoï restreint barycentrique. Le résultat a été obtenu en utilisant la méthode proposée par Liu *et al.* [67]. La régularité de l'échantillonnage dû à un diagramme de Voronoï barycentrique fait que les triangles sont presque équilatéraux

Un diagramme de Voronoï barycentrique tend à répartir équitablement les sites dans le domaine. Cette répartition fait qu'au niveau de la triangulation de Delaunay associée, les triangles sont proches de triangles équilatéraux. Cette caractéristique est souvent souhaitable pour le remaillage, en particulier pour utiliser le maillage obtenu pour simuler un phénomène physique par éléments finis. Dans ce cas, l'erreur de la simulation est liée à la forme des triangles comme le montre Shewchuk [104]. La notion de diagrammes de Voronoï barycentriques s'étend au diagrammes de Voronoï restreints. Dans ce cas, les sites doivent être situés au barycentre de leur cellule de Voronoï restreinte. Dans la plupart des cas, ce barycentre n'est pas situé exactement sur la surface. Un tel diagramme répartit également les sites régulièrement sur la surface, et la triangulation de Delaunay restreinte associée exhibe également des triangles de forme régulière.

Fonction objectif et continuité

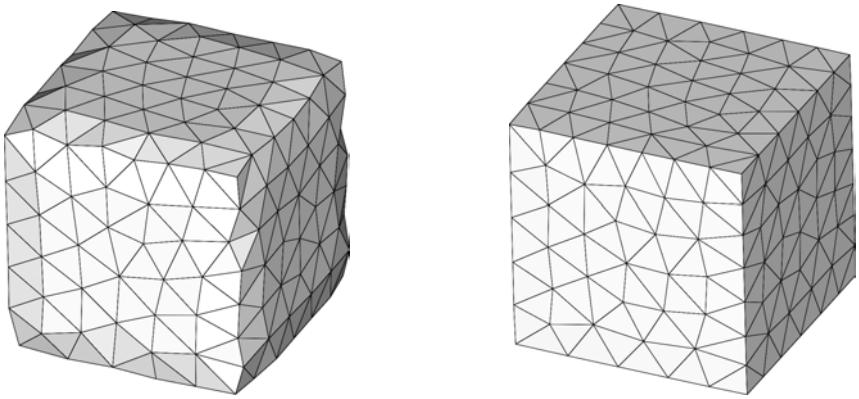
La fonction objectif utilisée pour optimiser un diagramme de Voronoï restreint est la même que pour un diagramme de Voronoï classique, en remplaçant les cellules de Voronoï classiques par des cellules de Voronoï restreintes. Elle devient donc :

$$\mathscr{F}_{|\mathcal{S}}(\mathbf{V}) = \sum_{\mathbf{v}_k \in \mathbf{V}} \int_{\Omega_{k|\mathcal{S}}} \|\mathbf{x} - \mathbf{v}_k\|^2 \mathrm{d}\mathbf{x}. \tag{4.3.8}$$

À partir de cette formule, le gradient se décompose également en un terme interne, et un terme externe comme pour l'Équation 4.3.4, et les mêmes simplifications opèrent pour obtenir la même expression du gradient par rapport à un site $\mathbf{v}_k$:

$$\frac{\partial \mathscr{F}_{|\mathcal{S}}}{\partial \mathbf{v}_k}(\mathbf{V}) = 2m_k(\mathbf{v}_k - \mathbf{g}_k)^t, \tag{4.3.9}$$

où m_k est l'aire de la portion de surface correspondant à $\Omega_{k|\mathcal{S}}$ et $\mathbf{g}_k$ son barycentre.



(a) sans anisotropie normale (b) avec anisotropie normale

FIGURE 4.4 – Introduction de l'anisotropie normale dans la fonction objectif $\mathscr{F}_{|\mathcal{S}}$ pour préserver les arêtes vives du maillage lors d'un remaillage.

La continuité de $\mathscr{F}_{|\mathcal{S}}$ est par contre plus délicate à étudier. Dans $\mathbb{R}^2$ la combinatoire du diagramme change lorsque quatre sites ou plus sont cocycliques. Dans le cas d'un digramme de Voronoï restreint, la combinatoire change par exemple lorsqu'un sommet d'une cellule de Voronoï de $\mathbb{R}^3$ est contenu dans une face de $\mathcal{S}$, ou lorsqu'un sommet de $\mathcal{S}$ est contenu dans une face d'une cellule de Voronoï. Aucune étude de continuité n'a à notre connaissance été réalisée pour ce type de diagrammes, même si les résultats de Liu *et al.* [67] se généralisent en grande partie à ce cas. Empiriquement, les résultats obtenus par l'utilisation d'une méthode quasi-Newton par Liu *et al.* [67] semblent confirmer l'hypothèse que $\mathscr{F}_{|\mathcal{S}}$ est bien C^2 presque partout.

Anisotropie normale

[62] introduisent deux modifications dans la fonction objectif $\mathscr{F}_{|\mathcal{S}}$, pour les applications spécifiques au remaillage. La première est l'anisotropie normale, mise au point pour éviter que le remaillage ne rogne les arêtes franches du maillage initial. Cette modification est réalisée sur la façon dont la distance entre un site $\mathbf{v}_k$ et un point $\mathbf{x}$ de $\mathcal{S}$ est calculée. Elle consiste à séparer dans cette distance le terme correspondant à la distance dans la direction de la normale $\mathbf{n}(\mathbf{x})$ à $\mathcal{S}$ en $\mathbf{x}$, et à pénaliser la distance dans cette direction. En pénalisant la distance normale par rapport à la distance tangentielle, de meilleurs résultats sont obtenus au niveau des arêtes franches du maillage, comme le montre la Figure 4.4. L'intérêt principal de cette méthode est qu'elle ne nécessite pas de détecter les arêtes vives et d'y appliquer un traitement particulier.

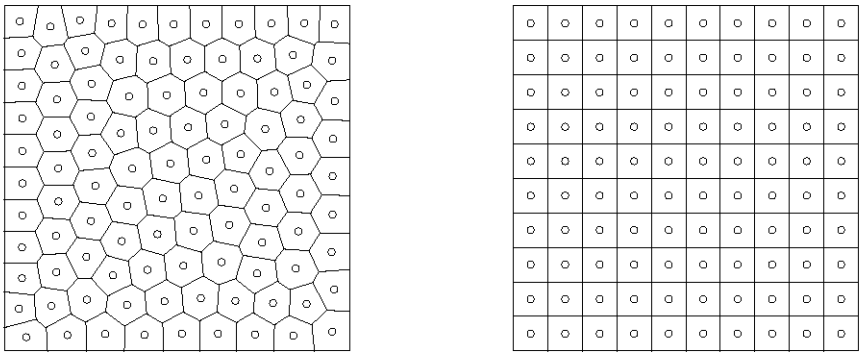
La composante du vecteur $\mathbf{x} - \mathbf{v}_k$ le long de la normale de $\mathcal{S}$ est donnée par le vecteur :

$$\left[(\mathbf{x} - \mathbf{v}_k)\,.\mathbf{n}(\mathbf{x})\right]\mathbf{n}(\mathbf{x}) = \mathbf{n}(\mathbf{x})\left[\mathbf{n}(\mathbf{x})^t(\mathbf{x} - \mathbf{v}_k)\right].$$

Pour pouvoir contrôler la pénalisation de la distance le long de cette direction dans $\mathscr{F}_{|\mathcal{S}}$, Lévy et Liu [62] définissent au point $\mathbf{x}$ la matrice $M(\mathbf{x})$:

$$M(\mathbf{x},s) = (s-1)\mathbf{n}(\mathbf{x})\mathbf{n}(\mathbf{x})^t + I_{3\times 3},$$

où $I_{3\times 3}$ est la matrice identité de $\mathbb{R}^3$, et s est le paramètre contrôlant la pénalisation, valant 1 dans le cas non pénalisé. En augmentant la valeur de s, la distance le long de la normale à



(a) stable (b) instable

FIGURE 4.5 – Stabilité de deux diagrammes de Voronoï barycentriques sous la norme L_2.

$\mathcal{S}$ prend donc plus d'importance. La fonction objectif intégrant ce paramètre devient ainsi :

$$\mathscr{F}_{|\mathcal{S}}(\mathbf{V}) = \sum_{\mathbf{v}_k \in \mathbf{V}} \int_{\Omega_{k|\mathcal{S}}} \|M(\mathbf{x},s)(\mathbf{x} - \mathbf{v}_k)\|^2 \mathrm{d}\mathbf{x}. \tag{4.3.10}$$

Les diagrammes de Voronoï anisotropes ont par ailleurs également été étudiés par Labelle et Shewchuk [57] et Du et Wang [29], qui modifient la manière de calculer les distances en y introduisant un tenseur métrique. À partir de cette distance, un algorithme est proposé pour calculer le barycentre des cellules de Voronoï définies par cette distance, et l'algorithme de Lloyd est utilisé avec ces barycentres pour obtenir un diagramme de Voronoï barycentrique tenant compte du tenseur métrique. Valette *et al.* [112] utilisent une fonction objectif proche de celle de l'Équation 4.3.10, Mais utilisent une partition des faces de $\mathcal{S}$ comme approximation d'un diagramme de Voronoï. Un algorithme glouton est mis en place pour transférer des faces d'une cellule à l'autre et mettre à jour le barycentre de ces cellules, afin d'optimiser cette partition.

Norme L_p

Un diagramme de Voronoï peut être barycentrique, sans être stable. À partir d'un tel diagramme, en réalisant une petite perturbation, et en optimisant $\mathscr{F}_{|\mathcal{S}}$ sur l'ensemble de sites obtenu, l'optimisation ne reviendra pas à la configuration initiale, mais convergera vers une configuration stable. Les configurations stables correspondent aux minima locaux de $\mathscr{F}_{|\mathcal{S}}$ alors que les configurations instables sont des situations où le gradient de $\mathscr{F}_{|\mathcal{S}}$ s'annule sans qu'il ne s'agisse d'un minimum. Il peut alors s'agir d'un maximum local ou d'un point selle. Deux configurations sont données sur la Figure 4.5, l'une stable, l'autre non. La configuration instable sur cette figure peut cependant être souhaitable dans le but de réaliser un remaillage quadrangulaire. C'est à partir de ce constat que Lévy et Liu [62] ont proposé une seconde modification de $\mathscr{F}_{|\mathcal{S}}$ pour rendre stable la configuration instable de la Figure 4.5.

Cette modification est fondée sur l'idée que l'optimisation, tend à rendre les cellules les plus circulaires possibles, et que le pavage hexagonal est le pavage régulier dont les cellules sont le plus proches du cercle. Pour rendre la grille régulière stable, [62] ont modifié la norme utilisée pour mesurer la distance. Rappelons que la norme L_p d'un vecteur $\mathbf{w}$ de

coordonnées $(w_1, \dots w_d)$ d'un vecteur est définie comme :

$$\|\mathbf{w}\|_p = \left(\sum_{l=1}^{d} |w_l|^p \right)^{\frac{1}{p}}.$$

En remplaçant la norme L_2 usuelle par la norme L_p avec p assez grand, la boule unité de la distance se rapproche d'un carré, ce qui rend stable la configuration instable présentée en Figure 4.5. La fonction objectif modifiée en utilisant cette norme ainsi que l'anisotropie normale devient donc :

$$\mathscr{F}_{|\mathcal{S}}(\mathbf{V}) = \sum_{\mathbf{v}_k \in \mathbf{V}} \int_{\Omega_{k|\mathcal{S}}} (\|M(\mathbf{x},s)(\mathbf{x} - \mathbf{v}_k)\|_p)^p \, \mathrm{d}\mathbf{x}. \tag{4.3.11}$$

Dans cette formule, les cellules de Voronoï restent cependant calculées en utilisant la norme L_2, car il n'existe pas à notre connaissance d'algorithmes pour le calcul de diagrammes de Voronoï restreints en norme Lp, et qu'un tel calcul serait probablement plus complexe.

Calcul de $\mathscr{F}_{|\mathcal{S}}$ en norme L_p avec anisotropie normale

L'Équation 4.3.9 utilisait le fait que dans le calcul de la dérivée, le terme externe était annulé du fait qu'à l'interface entre deux cellules restreintes Ω_1 et Ω_2, nous avions $\|\mathbf{x} - \mathbf{v}_1\|_2 = \|\mathbf{x} - \mathbf{v}_2\|_2$. En utilisant la norme L_p, ce résultat n'est plus vrai, car le diagramme de Voronoï est calculé en norme L_2. De même, en utilisant l'anisotropie normale, le terme externe ne s'annule plus. Il devient donc nécessaire de prendre en compte le terme externe lors du calcul du gradient.

La surface $\mathcal{S}$ est un maillage, et les cellules restreintes le sont donc également. Une cellule restreinte $\Omega_{k|\mathcal{S}}$ est donc constituée d'un ensemble de triangles et l'Équation 4.3.11 peut être réécrite :

$$\mathscr{F}_{|\mathcal{S}}(\mathbf{V}) = \sum_{\mathbf{v}_k \in \mathbf{V}} \sum_{T \in \Omega_{k|\mathcal{S}}} \int_T \|M_T(s)(\mathbf{x} - \mathbf{v}_k)\|_p^p \mathrm{d}\mathbf{x}, \tag{4.3.12}$$

où T est un triangle de $\Omega_{k|\mathcal{S}}$, et $M_T(s)$ est la matrice de l'anisotropie normale utilisant la normale $\mathbf{n}_T$ du triangle T. Par un changement de variable, et en utilisant les résultats de Lasserre et Avrachenkov [58], l'expression de l'intégrale sur T peut être obtenue sous forme close, en fonction des sommets $\mathbf{c}_1$, $\mathbf{c}_2$ et $\mathbf{c}_3$ de T, et de $\mathbf{v}_k$:

$$\mathscr{F}_{|\mathcal{S}}(\mathbf{V}) = \sum_{\mathbf{v}_k \in \mathbf{V}} \sum_{T \in \Omega_{k|\mathcal{S}}} \left[\frac{2|T|}{(p+1)(p+2)} \sum_{\alpha+\beta+\gamma=p} \mathbf{1}^t \left(\mathbf{p}_1^\alpha * \mathbf{p}_2^\beta * \mathbf{p}_3^\gamma \right) \right], \tag{4.3.13}$$

où $\mathbf{p}_i$ correspond au vecteur $M_T(s)(\mathbf{c}_i - \mathbf{v}_k)$, $\mathbf{1}$ est le vecteur de coordonnées 1 et, en notant $p_{i,l}$ les coordonnées de $\mathbf{p}_i$,

$$\begin{cases} \mathbf{p}_1 * \mathbf{p}_2 & = [p_{1,1}p_{2,1} \ , \ p_{1,2}p_{2,2} \ , \ p_{1,3}p_{2,3}]^t \\ \mathbf{p}^\alpha & = \mathbf{p} * \dots * \mathbf{p} \ (\alpha \text{ fois}) \end{cases}$$

Les détails de l'obtention de cette expression sont donnés dans [62]. À partir de cette expression, il est possible de calculer la valeur de $\mathscr{F}_{|\mathcal{S}}$ à partir des positions de $\mathbf{c}_1$, $\mathbf{c}_2$ et $\mathbf{c}_3$, données par le diagramme de Voronoï restreint.

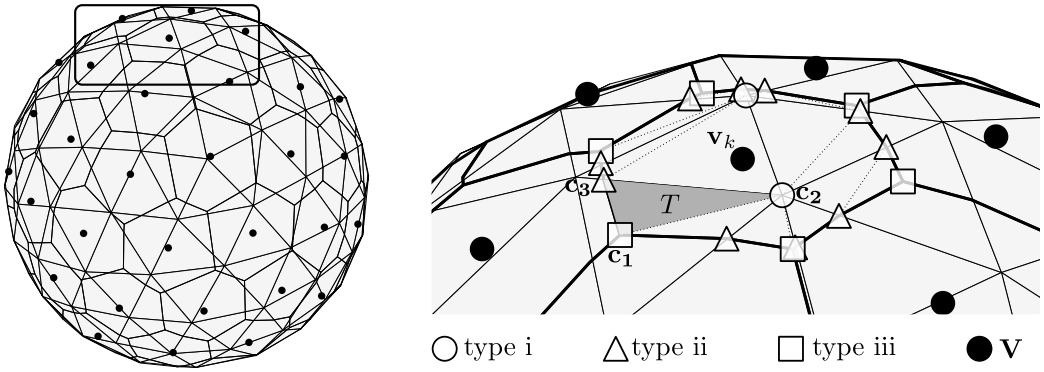


FIGURE 4.6 – Classification des types de points issus d'un diagramme de Voronoï restreint. Les points peuvent être ○ un sommet de $\mathcal{S}$, △ l'intersection entre le bissecteur de deux sites et une arête de $\mathcal{S}$, □ l'intersection entre une arête de Voronoï entre trois sites et une face T de $\mathcal{S}$.

Le gradient de $\mathscr{F}_{|\mathcal{S}}$ est obtenu en dérivant l'Équation 4.3.13. Il est alors nécessaire de calculer les dérivées des points $\mathbf{c}_1$, $\mathbf{c}_2$ et $\mathbf{c}_3$ en fonction des positions des sites. Notez qu'en calculant le gradient de cette manière, nous supposons que la combinatoire du diagramme de Voronoï restreint ne varie pas, en supposant qu'une variation suffisamment petite de l'ensemble des sites n'ajoute ou n'enlève aucun triangle dans la somme sur $T \in \Omega_k$.

Dérivée des sommets du diagramme restreint par rapport aux sites

Combinatoirement, un diagramme de Voronoï restreint réutilise tous les sommets de la surface $\mathcal{S}$ par laquelle il est restreint, et rajoute de nouveaux sommets pour matérialiser les bords des cellules restreintes. Les sommets d'un triangle T d'une cellule restreinte $\Omega_{k|\mathcal{S}}$ peuvent donc être de trois types :

○ **type i :** un tel sommet correspond à un sommet initial de $\mathcal{S}$. Ce sommet ne dépend donc pas de la positions des sites.

△ **type ii :** il s'agit ici de l'intersection entre une arête de $\mathcal{S}$ et un bissecteur défini par $\mathbf{v}_k$ et un site voisin. Il dépend donc de deux sites.

□ **type iii :** ce sommet est à l'intersection entre une face de $\mathcal{S}$ et deux bissecteurs, entre $\mathbf{v}_k$ et deux de ses sites voisins. Il dépend alors de trois sites.

Ces trois types de sommets sont illustrés sur la Figure 4.6. Les sommets de type i sont de dérivée nulle, car ils ne dépendent pas des sites. Pour les deux autres configurations, la même technique est utilisée pour calculer leurs dérivées : les sommets sont exprimés sous la forme d'un système linéaire :

$$A\mathbf{c} = \mathbf{b}.$$

À partir de cette expression, en utilisant les formules pour la dérivation de fonctions vectorielles et matricielles données par Minka [73], la différentielle $\mathrm{d}\mathbf{c}$ de $\mathbf{c}$ est donnée par :

$$\begin{aligned} \mathrm{d}\mathbf{c} &= \mathrm{d}(A^{-1}\mathbf{b}) = (\mathrm{d}A^{-1})\mathbf{b} + A^{-1}\mathrm{d}\mathbf{b} \\ &= -A^{-1}(\mathrm{d}A)A^{-1}\mathbf{b} + A^{-1}\mathrm{d}\mathbf{b} \\ &= A^{-1}(\mathrm{d}\mathbf{b} - (\mathrm{d}A)\mathbf{c}). \end{aligned} \tag{4.3.14}$$

Selon le type de point, les quantités A^{-1}, $\mathrm{d}A$ et $\mathrm{d}\mathbf{b}$ sont calculées et injectées dans cette équation pour obtenir la dérivée du point en fonction des sites. Pour formuler le système

linéaire, le point $\mathbf{c}$ est défini comme l'intersection de trois plans. Étant donné un plan $\mathcal{P}$, $\mathbf{n}_\mathcal{P}$ sa normale et un point $\mathbf{p}_\mathcal{P}$ de $\mathcal{P}$, le point $\mathbf{c}$ est dans $\mathcal{P}$ si et seulement si :

$$\mathbf{n}_\mathcal{P}.\mathbf{c} = \mathbf{n}_\mathcal{P}.\mathbf{p}_\mathcal{P}.$$

Étant donné trois plan $\mathcal{P}_1$, $\mathcal{P}_2$ et $\mathcal{P}_3$ contenant $\mathbf{c}$, le système linéaire correspondant est donc :

$$\begin{pmatrix} \mathbf{n}_{\mathcal{P}_1}^t \\ \mathbf{n}_{\mathcal{P}_2}^t \\ \mathbf{n}_{\mathcal{P}_3}^t \end{pmatrix} \mathbf{c} = \begin{pmatrix} \mathbf{n}_{\mathcal{P}_1}.\mathbf{p}_{\mathcal{P}_1} \\ \mathbf{n}_{\mathcal{P}_2}.\mathbf{p}_{\mathcal{P}_2} \\ \mathbf{n}_{\mathcal{P}_3}.\mathbf{p}_{\mathcal{P}_3} \end{pmatrix}$$

Il suffit donc simplement de fournir pour chaque plan une normale et un point, définis à partir des sommets $\{\mathbf{x}_i\}_{i=1}^m$ et des sites $\{\mathbf{v}_k\}_{k=1}^n$:

type ii : l'arête contenant le point $\mathbf{c}$ relie deux sommets $\mathbf{x}_1$ et $\mathbf{x}_2$ de $\mathcal{S}$. Cette arête est au moins adjacente à une face T, de normale $\mathbf{n}_T$. Un premier plan est donc défini par $\mathbf{n}_T$ et $\mathbf{x}_1$. Pour définir un second plan contenant l'arête ($\mathbf{x}_1$,$\mathbf{x}_2$), la normale $\mathbf{n}_T \times (\mathbf{x}_1 - \mathbf{x}_2)$ est choisie, toujours avec le point $\mathbf{x}_1$. Enfin le bissecteur contenant $\mathbf{c}$ est défini par deux sites $\mathbf{v}_1$ et $\mathbf{v}_2$. Sa normale est $\mathbf{v}_1 - \mathbf{v}_2$, et il passe par le point $\frac{1}{2}(\mathbf{v}_1 + \mathbf{v}_2)$ correspondant au centre de ce segment. Nous avons donc :

$$A = \begin{pmatrix} \mathbf{n}_T^t \\ (\mathbf{n}_T \times (\mathbf{x}_1 - \mathbf{x}_2))^t \\ \mathbf{v}_1^t - \mathbf{v}_2^t \end{pmatrix} \quad , \quad \mathrm{d}A = \begin{pmatrix} \mathbf{0}^t \\ \mathbf{0}^t \\ \mathrm{d}\mathbf{v}_1^t - \mathrm{d}\mathbf{v}_2^t \end{pmatrix},$$

et

$$\mathbf{b} = \begin{pmatrix} \mathbf{x}_1.\mathbf{n}_T \\ \mathbf{x}_1.(\mathbf{n}_T \times (\mathbf{x}_1 - \mathbf{x}_2)) \\ \frac{1}{2}(\|\mathbf{v}_1\|^2 - \|\mathbf{v}2\|^2) \end{pmatrix} \quad , \quad \mathrm{d}\mathbf{b} = \begin{pmatrix} 0 \\ 0 \\ \mathbf{v}_1.\mathrm{d}\mathbf{v}_1 - \mathbf{v}2.\mathrm{d}\mathbf{v}_2 \end{pmatrix}.$$

En remplaçant dans l'Équation 4.3.14, nous obtenons donc :

$$\begin{aligned} \mathrm{d}\mathbf{c} &= A^{-1} \begin{pmatrix} 0 \\ 0 \\ (\mathbf{v}_1 - \mathbf{c}).\mathrm{d}\mathbf{v}_1 + (\mathbf{c} - \mathbf{v}_2).\mathrm{d}\mathbf{v}_2 \end{pmatrix} \\ &= A^{-1} \begin{pmatrix} \mathbf{0}^t \\ \mathbf{0}^t \\ (\mathbf{v}_1 - \mathbf{c})^t \end{pmatrix} \mathrm{d}\mathbf{v}_1 + A^{-1} \begin{pmatrix} \mathbf{0}^t \\ \mathbf{0}^t \\ (\mathbf{c} - \mathbf{v}_2)^t \end{pmatrix} \mathrm{d}\mathbf{v}_2 \\ &= \frac{\partial \mathbf{c}}{\partial \mathbf{v}_1} \mathrm{d}\mathbf{v}_1 + \frac{\partial \mathbf{c}}{\partial \mathbf{v}_2} \mathrm{d}\mathbf{v}_2 \end{aligned} \tag{4.3.15}$$

type iii : le point $\mathbf{c}$ est cette fois contenu dans la face T de sommets $\mathbf{x}_1$, $\mathbf{x}_2$ et $\mathbf{x}_3$. Le plan correspondant a pour normale $\mathbf{n}$, et passe par $\mathbf{x}_1$. Le point est également contenu dans deux bissecteurs, correspondant à trois sites $\mathbf{v}_1$, $\mathbf{v}_2$ et $\mathbf{v}_3$. Le premier a pour normale $(\mathbf{v}_1 - \mathbf{v}_2)$ et passe par $\frac{1}{2}(\mathbf{v}_1 + \mathbf{v}_2)$, le second a la normale $(\mathbf{v}_1 - \mathbf{v}_3)$ et passe par $\frac{1}{2}(\mathbf{v}_1 + \mathbf{v}_3)$. Nous avons donc :

$$A = \begin{pmatrix} \mathbf{n}_T^t \\ \mathbf{v}_1^t - \mathbf{v}_2^t \\ \mathbf{v}_1^t - \mathbf{v}_3^t \end{pmatrix} \quad , \quad \mathrm{d}A = \begin{pmatrix} \mathbf{0}^t \\ \mathrm{d}\mathbf{v}_1^t - \mathrm{d}\mathbf{v}_2^t \\ \mathrm{d}\mathbf{v}_1^t - \mathrm{d}\mathbf{v}_3^t \end{pmatrix},$$

et

$$\mathbf{b} = \begin{pmatrix} \mathbf{x}_1.\mathbf{n}_T \\ \frac{1}{2}(\|\mathbf{v}_1\|^2 - \|\mathbf{v}2\|^2) \\ \frac{1}{2}(\|\mathbf{v}_1\|^2 - \|\mathbf{v}3\|^2) \end{pmatrix} \quad , \quad \mathrm{d}\mathbf{b} = \begin{pmatrix} 0 \\ \mathbf{v}_1.\mathrm{d}\mathbf{v}_1 - \mathbf{v}2.\mathrm{d}\mathbf{v}_2 \\ \mathbf{v}_1.\mathrm{d}\mathbf{v}_1 - \mathbf{v}3.\mathrm{d}\mathbf{v}_3 \end{pmatrix}.$$

En remplaçant dans l'Équation 4.3.14, nous obtenons donc :

$$\begin{aligned}
\mathrm{d}\mathbf{c} &= A^{-1}\begin{pmatrix} 0 \\ (\mathbf{v}_1-\mathbf{c}).\mathrm{d}\mathbf{v}_1 + (\mathbf{c}-\mathbf{v}_2).\mathrm{d}\mathbf{v}_2 \\ (\mathbf{v}_1-\mathbf{c}).\mathrm{d}\mathbf{v}_1 + (\mathbf{c}-\mathbf{v}_3).\mathrm{d}\mathbf{v}_3 \end{pmatrix} \\
&= A^{-1}\begin{pmatrix} \mathbf{0}^t \\ (\mathbf{v}_1-\mathbf{c})^t \\ (\mathbf{v}_1-\mathbf{c})^t \end{pmatrix}\mathrm{d}\mathbf{v}_1 + A^{-1}\begin{pmatrix} \mathbf{0}^t \\ (\mathbf{c}-\mathbf{v}_2)^t \\ \mathbf{0}^t \end{pmatrix}\mathrm{d}\mathbf{v}_2 + A^{-1}\begin{pmatrix} \mathbf{0}^t \\ \mathbf{0}^t \\ (\mathbf{c}-\mathbf{v}_3)^t \end{pmatrix}\mathrm{d}\mathbf{v}_3 \\
&= \frac{\partial \mathbf{c}}{\partial \mathbf{v}_1}\mathrm{d}\mathbf{v}_1 + \frac{\partial \mathbf{c}}{\partial \mathbf{v}_2}\mathrm{d}\mathbf{v}_2 + \frac{\partial \mathbf{c}}{\partial \mathbf{v}_3}\mathrm{d}\mathbf{v}_3
\end{aligned} \tag{4.3.16}$$

En utilisant ces dérivées, il devient alors possible d'obtenir le gradient de $\mathscr{F}_{|\mathcal{S}}$ par rapport à l'ensemble des sites. C'est ce gradient qui est ensuite injecté dans le solveur L-BFGS pour optimiser un diagramme de Voronoï restreint par Lévy et Liu [62] pour remailler une surface.

4.4 Contribution : échantillonnage d'une fonction quelconque

Nous étudierons ici une variante de la fonction objectif optimisée pour obtenir un diagramme de Voronoï restreint. Notre objectif ici est l'approximation d'une fonction définie sur une surface. Nous disposons donc d'une fonction $f : \mathcal{S} \to \mathbb{R}$ définie en tout point de $\mathcal{S}$, et nous souhaitons l'approximer par un diagramme de Voronoï restreint en utilisant une valeur constante dans chaque cellule de Voronoï. D'autres variantes ont déjà été étudiées, en particulier pour les problèmes de *localisation géométrique*, où étant donné une fonction $f : \mathbb{R} \to \mathbb{R}$, la fonction objectif minimisée est :

$$\mathscr{F}(\mathbf{V}) = \sum_{\mathbf{v}_k \in \mathbf{V}} \int_{\Omega_k} f(\|\mathbf{x}-\mathbf{v}_k\|^2)\mathrm{d}\mathbf{x}.$$

Au niveau du bord d'une cellule de Voronoï, cette variante a l'avantage de ne pas dépendre du site $\mathbf{v}_k$ choisi pour évaluer sa valeur, ce qui se traduit de nouveau par le fait que le terme externe de l'Équation 4.3.4 correspondant à l'intégrale sur $\partial\Omega_k$ s'annule toujours. Cette propriété permet notamment de simplifier le calcul du gradient de $\mathscr{F}$.

Dans leur étude sur les diagrammes de Voronoï anisotropes, Du et Wang [29, Section 2.6] mentionnent comme application l'optimisation de l'échantillonnage d'une fonction, par la minimisation de l'équation :

$$\mathscr{F}(\mathbf{V}) = \sum_{\mathbf{v}_k \in \mathbf{V}} \int_{\Omega_k} \|f(\mathbf{x}) - f(\mathbf{v}_k)\|^2 \mathrm{d}\mathbf{x}, \tag{4.4.1}$$

où f est une fonction quelconque. En effet, lorsque $\mathbf{x}$ et $\mathbf{v}_k$ sont proches, cette fonction objectif peut être approximée par :

$$\mathscr{F}(\mathbf{V}) \simeq \sum_{\mathbf{v}_k \in \mathbf{V}} \int_{\Omega_k} \|\frac{\partial f}{\partial \mathbf{x}}(\mathbf{x})(\mathbf{x}-\mathbf{v}_k)\|^2 \mathrm{d}\mathbf{x},$$

ce qui se ramène au calcul d'un diagramme de Voronoï barycentrique avec une anisotropie issue du gradient de f.

Nous nous intéresserons ici à la minimisation d'une fonction objectif très similaire à celle de l'Équation 4.4.1, avec la seule différence que nous ne considérerons pas que la valeur de l'approximation de la fonction f dans une cellule de Voronoï d'un site $\mathbf{v}_k$ est nécessairement donnée par $f(\mathbf{v}_k)$. Nous définissons ainsi un nouvel ensemble de variables $\tilde{\mathbf{F}} = \{\tilde{f}_k\}_{k=1}^n$ associées à chaque site et donnant la valeur de l'approximation de f dans la cellule Ω_k. La fonction objectif que nous chercherons à minimiser est ainsi la fonction :

$$\mathscr{F}(\mathbf{V},\tilde{\mathbf{F}}) = \sum_{\mathbf{v}_k \in \mathbf{V}} \int_{\Omega_{k|\mathcal{S}}} \|f(\mathbf{x}) - \tilde{f}_k\|^2 \mathrm{d}\mathbf{x}. \tag{4.4.2}$$

Pour optimiser un ensemble de sites par rapport à cette fonction objectif, nous allons donc voir comment calculer son gradient, afin d'utiliser une méthode de type descente de gradient. Pour réaliser ce calcul, nous utiliserons le théorème du transport de Reynolds, que nous allons donc commencer par présenter.

4.4.1 Théorème du transport de Reynolds

Énoncé

La difficulté du calcul du gradient de $\mathscr{F}$ réside dans le fait que le variables correspondent à la position des sites, et qu'en modifiant ces variables, le diagramme de Voronoï est modifié. Ainsi les cellules de Voronoï sont variables, et il est donc nécessaire de calculer des dérivées de la forme :

$$\frac{\partial}{\partial t}\left[\int_{\Omega(t)} f(\mathbf{x},t)\mathrm{d}\mathbf{x}\right],$$

où à la fois le domaine d'intégration et la fonction intégrée varient en fonction de la variable par rapport à laquelle la dérivée est calculée. Pour réaliser ce calcul, nous utiliserons le théorème du transport de Reynolds. Reynolds [97] énonce ce théorème de la façon suivante :

> Tout changement quel qu'il soit dans la quantité d'une entité quelconque contenue à l'intérieur d'une surface fermée ne peut survenir que par l'une ou l'autre des deux causes disjointes suivantes :
>
> 1. il peut être causé par la création ou la disparition de l'entité à l'intérieur de la surface, ou
> 2. par le passage de l'entité au travers de la surface.

Ce théorème correspond en réalité à la généralisation au cas multivarié de la règle de dérivation sous le signe somme de Leibnitz :

$$\frac{\partial}{\partial t}\left[\int_{\Omega(t)} f(\mathbf{x},t)\mathrm{d}\mathbf{x}\right] = \int_{\Omega(t)} \frac{\partial f}{\partial t}(\mathbf{x},t)\mathrm{d}\mathbf{x} + \int_{\partial\Omega(t)} f(\mathbf{x},t)\frac{\partial \mathbf{x}}{\partial t}(t).\mathbf{n}_{\partial\Omega(t)}(\mathbf{x})\mathrm{d}\mathbf{x}, \tag{4.4.3}$$

où $\partial\Omega(t)$ est le bord de $\Omega(t)$, et $\mathbf{n}_{\partial\Omega(t)}(\mathbf{x})$ est la normale de ce bord au point $\mathbf{x}$. Ce théorème apparaît de façon récurrente dans les différents travaux concernant le calcul du gradient de la fonction objectif définie par l'Équation 4.3.3. Asami [2] démontrent cette formule en détaillant les calculs menant à l'expression du gradient fournie par Iri *et al.* [52], et Cortes *et al.* [21] la démontrent également pour prouver la continuité C^1 de cette fonction objectif. Différentes preuves de ce résultat selon la dimension sont également proposées par Flanders [38].

Hypothèses nécessaires

Les hypothèses nécessaires pour la fonction f et le domaine $\Omega(t)$ ne sont par contre pas toujours mentionnées clairement dans les sources énonçant ce théorème. Cortes *et al.* [21] mentionnent précisément les hypothèses prises pour leur démonstration :

hypothèses sur f : il est nécessaire que les intégrales apparaissant dans la partie droite de l'Équation 4.4.3 soient bien définies. Il est ici notamment nécessaire que f soit intégrable sur $\partial\Omega(t)$. De plus $\frac{\partial f}{\partial t}$ doit également être intégrable sur $\Omega(t)$. Dans la preuve de Cortes *et al.* [21], il est également nécessaire que f soit continue pour assurer un passage à la limite.

hypothèses sur Ω : deux termes de l'Équation 4.4.3 apparaissent et dépendent de $\Omega(t)$: la normale $\mathbf{n}_{\partial\Omega(t)}(\mathbf{x})$ du bord au point $\mathbf{x}$ et la dérivée $\frac{\partial \mathbf{x}}{\partial t}$ du point $\mathbf{x}$ du bord en fonction de t. Pour assurer que ces deux termes ne posent pas de problèmes, il est nécessaire qu'il existe une paramétrisation $\gamma(\theta,t) : \mathbb{S}^1 \times \mathbb{R} \to \partial\Omega(t)$ de $\partial\Omega(t)$, où $\mathbb{S}^1$ est le cercle unité. Pour pouvoir définir $\mathbf{n}_{\partial\Omega(t)}(\mathbf{x})$, cette paramétrisation doit être différentiable par morceaux en fonction de θ. Dans la preuve de Cortes *et al.* [21], pour assurer un passage à la limite, γ doit également être C^1 par rapport à t, ce qui assure également que $\frac{\partial \mathbf{x}}{\partial t}$ est une fonction continue sur $\partial\Omega$. Enfin, cette preuve est réalisée dans le cas où Ω est un domaine étoilé, mais cette condition, de même que celle imposant que $\partial\Omega(t)$ soit paramétrable sur $\mathbb{S}^1$ peuvent être relâchées tant que Ω peut être décomposé comme somme ou différence d'ensembles étoilés, et que les bords de chacune de ses composantes connexes sont des courbes fermées.

Dans notre cas, nous souhaiterions utiliser le théorème du transport de Reynolds pour calculer le gradient de $\mathscr{F}$. Les domaines sont donc des cellules de Voronoï restreintes. Il est ainsi nécessaire d'évaluer dans quelles circonstances ce théorème est applicable. Nous supposerons également que f remplit toutes les conditions nécessaires.

Paramétrisation du bord

Dans la plupart des cas, chaque sommet $\mathbf{c}$ du diagramme de Voronoï restreint peut être défini sans ambiguïté comme étant du type i, ii ou iii, comme défini en Section 4.3.3. La position d'un sommet non ambigu $\mathbf{c}$ est alors la solution d'un système linéaire et ses coordonnées s'expriment alors comme des fractions rationnelles des coordonnées des sites. Dans un voisinage autour d'une configuration sans ambiguïtés des sites, le sommet $\mathbf{c}$ varie donc de façon C^∞ par rapport aux positions des sites. Pour certaines configurations des sites, il n'est cependant pas possible de définir de manière unique la combinatoire d'un sommet du diagramme de Voronoï restreint. Une telle situation survient par exemple lorsqu'un sommet de Voronoï est contenu dans une face de $\mathcal{S}$. Dans un tel cas une variation, si petite soit-elle, de la position des sites peut amener le sommet de Voronoï d'un côté ou de l'autre de la face de $\mathcal{S}$, et ainsi modifier la combinatoire du diagramme de Voronoï restreint. Cet exemple est illustré sur la Figure 4.7. Nous dirons dans ce cas que le diagramme de Voronoï comporte un sommet *ambigu*.

Lorsque la configuration des sites ne donne pas lieu à des sommets ambigus, il est alors possible de paramétrer trivialement chaque bord d'une cellule de Voronoï restreinte $\Omega_{k|\mathcal{S}}$ sur le cercle unité $\mathbb{S}^1$ de façon à respecter les conditions d'application du théorème du transport de Reynolds évoquées en Section 4.4.1. Un bord de $\Omega_{k|\mathcal{S}}$ est une courbe polygonale, dont les sommets sont de type ii ou iii. Soit $\{\mathbf{c}_j\}_{j=1}^s$ l'ensemble ordonné des sommets le long du

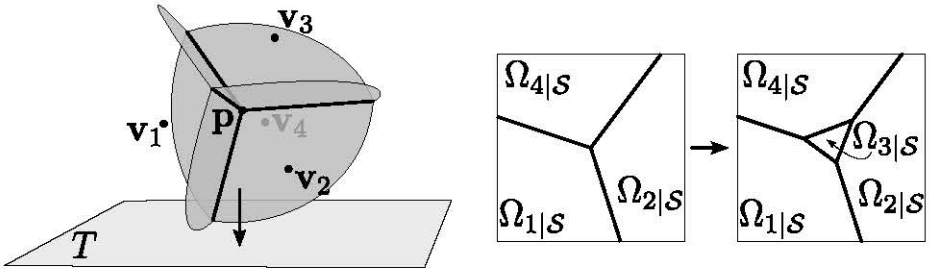


FIGURE 4.7 – Changement de combinatoire lorsqu'un sommet de Voronoï $\mathbf{p}$ traverse une face T de $\mathcal{S}$. Lorsque $\mathbf{p}$ est au dessus de T, seules les cellules de Voronoï des sites $\mathbf{v}_1$, $\mathbf{v}_2$ et $\mathbf{v}_4$ intersectent T. Lorsque $\mathbf{p}$ passe en dessous de $\mathcal{S}$, la cellule de Voronoï de $\mathbf{v}_3$ intersecte également T, ce qui modifie la combinatoire du diagramme de Voronoï restreint à l'intérieur de T.

bord, à chaque sommet $\mathbf{c}_j$, nous associons l'angle θ_j du cercle unité $\mathbb{S}^1$. Chaque segment $[\mathbf{c}_j,\mathbf{c}_{j+1}]$ est ensuite paramétré linéairement sur l'intervalle $[\theta_j,\theta_{j+1}]$. Cette paramétrisation que nous noterons $\gamma(\theta,\mathbf{V})$ est linéaire par morceaux, elle est donc bien C^1 par morceaux par rapport à θ. De plus les sommets $\{\mathbf{c}_j\}_j$ varient de manière C^∞ par rapport à $\mathbf{V}$, cette paramétrisation est donc bien dérivable par rapport à $\mathbf{V}$, et cette dérivée est continue. La paramétrisation γ respecte donc bien les hypothèses nécessaires à l'application du théorème du transport de Reynolds.

Lorsque par contre un bord d'une cellule de Voronoï restreinte contient un sommet $\mathbf{c}$ ambigu, une variation infinitésimale de la position des sites peut alors modifier le nombre de sommets le long du bord. Si θ est le paramètre correspondant à $\mathbf{c}$, γ est alors C^0 en θ par rapport aux positions des sites, mais n'est plus C^1, car différentes dérivées sont possibles pour le point $\mathbf{c}$ selon la direction de la variation infinitésimale des sites.

4.4.2 Calcul du gradient

Nous nous plaçons ici dans le cas de calcul du gradient pour une configuration des sites sans sommet ambigu. Il est probable que ces résultats se généralisent lorsque le diagramme de Voronoï restreint contient un sommet ambigu, mais nous ne disposons pas à l'heure actuelle d'une preuve satisfaisante de ce résultat.

Application du théorème du transport de Reynolds

Notre fonction objectif a l'expression :

$$\mathscr{F}(\mathbf{V},\tilde{\mathbf{F}}) = \sum_{\mathbf{v}_k\in\mathbf{V}} \int_{\Omega_{k|\mathcal{S}}} \|f(\mathbf{x}) - \tilde{f}_k\|^2 \mathrm{d}\mathbf{x}, \tag{4.4.4}$$

où $\tilde{\mathbf{F}} = \{\tilde{f}_k\}_{k=1}^n$ correspond à l'ensemble des valeurs d'approximation associées à chaque site. Tout d'abord, le gradient de $\mathscr{F}$ par rapport aux valeurs d'approximation est simple à calculer car les cellules de Voronoï ne varient pas avec ces variables. Ainsi, nous avons

$$\frac{\partial\mathscr{F}}{\partial\tilde{f}_k} = -2\int_{\Omega_{k|\mathcal{S}}} f(\mathbf{x}) - \tilde{f}_k \mathrm{d}\mathbf{x} = 2\left(\tilde{f}_k \int_{\Omega_{k|\mathcal{S}}} \mathrm{d}\mathbf{x} - \int_{\Omega_{k|\mathcal{S}}} f(\mathbf{x})\mathrm{d}\mathbf{x}\right). \tag{4.4.5}$$

Pour calculer le gradient par rapport aux sites, nous utilisons le théorème du transport de Reynolds :

$$\begin{aligned}\frac{\partial \mathscr{F}}{\partial \mathbf{V}} &= \sum_{\mathbf{v}_k \in \mathbf{V}} \frac{\partial}{\partial \mathbf{V}} \left[\int_{\Omega_{k|\mathcal{S}}} \|f(\mathbf{x}) - \tilde{f}_k\|^2 \mathrm{d}\mathbf{x} \right] \\ &= \sum_{\mathbf{v}_k \in \mathbf{V}} \underbrace{\int_{\Omega_{k|\mathcal{S}}} \frac{\partial}{\partial \mathbf{V}} \left(\|f(\mathbf{x}) - \tilde{f}_k\|^2 \right) \mathrm{d}\mathbf{x}}_{=0} + \int_{\partial\Omega_{k|\mathcal{S}}} \|f(\mathbf{x}) - \tilde{f}_k\|^2 \left(\frac{\partial \mathbf{x}}{\partial \mathbf{V}} . \mathbf{n}_{\partial\Omega_{k|\mathcal{S}}}(\mathbf{x}) \right) \mathrm{d}\mathbf{x}.\end{aligned} \tag{4.4.6}$$

Dans cette expression, le premier terme s'annule, car la fonction f ainsi que la valeur d'approximation $\tilde{f}_k$ ne dépendent pas de la position des sites, et que pour cette intégrale, le domaine est supposé constant, et donc $\mathbf{x}$ ne varie pas. Dans le second terme, $\frac{\partial \mathbf{x}}{\partial \mathbf{V}}$ correspond aux variations d'un sommet du bord par rapport aux sites, et $\mathbf{n}_{\partial\Omega_{k|\mathcal{S}}}$ correspond à la normale du $\partial\Omega_{k|\mathcal{S}}$. Ici, $\frac{\partial \mathbf{x}}{\partial \mathbf{V}}$ n'est pas un vecteur mais une matrice de hauteur 3 et de largeur $3n$, avec n le nombre de sites. Par extension, nous notons

$$\frac{\partial \mathbf{x}}{\partial \mathbf{V}} . \mathbf{n}_{\partial\Omega_{k|\mathcal{S}}}(\mathbf{x}) \coloneqq \left(\frac{\partial \mathbf{x}}{\partial \mathbf{V}} \right)^t \mathbf{n}_{\partial\Omega_{k|\mathcal{S}}}(\mathbf{x})$$

Variations du bord d'une cellule de Voronoï restreinte

Le théorème du transport de Reynolds fait apparaître le terme $\frac{\partial \mathbf{x}}{\partial \mathbf{V}}$ correspondant au gradient d'un point $\mathbf{x}$ du bord d'une cellule de Voronoï restreinte $\Omega_{k|\mathcal{S}}$ par rapport aux positions des sites $\mathbf{V}$. L'expression que nous fournirons ici est en réalité une généralisation du résultat obtenu par Asami [2] au cas des diagrammes de Voronoï restreint.

Soient $\mathbf{v}_1$ et $\mathbf{v}_2$ deux sites. Soient $\mathbf{p}$ et $\mathbf{w}$ respectivement un point et un vecteur unitaire de $\mathbb{R}^3$ tels que la droite de direction $\mathbf{w}$ passant par $\mathbf{p}$ coupe le bissecteur de $\mathbf{v}_1$ et $\mathbf{v}_2$ en un point $\mathbf{p}_i$. Nous allons étudier les variations du point $\mathbf{p}_i$ en fonction de $\mathbf{v}_1$ et $\mathbf{v}_2$. Tout d'abord, $\mathbf{p}_i$ est un point de la droite de direction $\mathbf{w}$ passant par $\mathbf{p}$, ce qui se traduit par le fait qu'il existe un réel u tel que

$$\mathbf{p}_i = \mathbf{p} + u\mathbf{w}.$$

De plus $\mathbf{p}_i$ est dans le bissecteur de $\mathbf{v}_1$ et $\mathbf{v}_2$. Ce plan a pour normale le vecteur $\mathbf{v}_1 - \mathbf{v}_2$ et passe par le point $\frac{1}{2}(\mathbf{v}_1 + \mathbf{v}_2)$. Nous avons donc

$$(\mathbf{p}_i - \frac{\mathbf{v}_1 + \mathbf{v}_2}{2}).(\mathbf{v}_1 - \mathbf{v}_2) = 0.$$

En combinant les deux équations précédentes, nous obtenons que

$$u = \frac{\left(\mathbf{p} - \frac{\mathbf{v}_1 + \mathbf{v}_2}{2}\right).(\mathbf{v}_1 - \mathbf{v}_2)}{\mathbf{w}.(\mathbf{v}_1 - \mathbf{v}_2)}$$

En développant les calculs, la différentielle de $\mathbf{p}_i$ vaut

$$\begin{aligned}\mathrm{d}\mathbf{p}_i &= \mathrm{d}(\mathbf{p} + u\mathbf{w}) = (\mathrm{d}u)\mathbf{w} \\ &= \frac{\mathrm{d}\mathbf{v}_1.(\mathbf{v}_1 - \mathbf{p}_i) - \mathrm{d}\mathbf{v}_2.(\mathbf{v}_2 - \mathbf{p}_i)}{\mathbf{w}.(\mathbf{v}_1 - \mathbf{v}_2)}\mathbf{w}.\end{aligned} \tag{4.4.7}$$

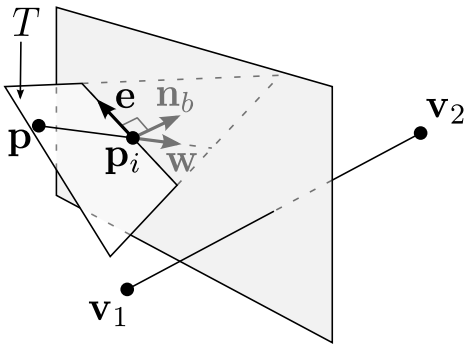


FIGURE 4.8 – Notations pour le calcul des variations de l'intersection $\mathbf{p}_i$ entre une droite de direction $\mathbf{w}$ passant par un point $\mathbf{p}$ et le bissecteur de deux sites $\mathbf{v}_1$ et $\mathbf{v}_2$. Le point $\mathbf{p}$ et la direction $\mathbf{w}$ sont pris dans une face T de $\mathcal{S}$. Le vecteur $\mathbf{e}$ correspond à la direction de l'intersection entre T et le bissecteur, $\mathbf{n}_b$ est le vecteur perpendiculaire à $\mathbf{e}$ dans T. Ce vecteur correspond à la normale du bord de la cellule restreinte $\Omega_{1|\mathcal{S}}$ au point $\mathbf{p}_i$.

Dans l'Équation 4.4.3, seul le produit scalaire du gradient d'un point $\mathbf{x}$ du bord avec la normale du bord est utilisé. Imaginons donc désormais que la droite passant par $\mathbf{p}$ de direction $\mathbf{w}$ sont contenue dans le plan d'une face T de $\mathcal{S}$. Soit $\mathbf{n}_T$ la normale de T, et $\mathbf{e}$ un vecteur directeur unitaire de la droite correspondant à l'intersection entre T et le bissecteur de $\mathbf{v}_1$ et $\mathbf{v}_2$. La normale du bord $\mathbf{n}_b$ correspond alors au signe près au vecteur $\mathbf{e}\times\mathbf{n}_T$, le signe dépendant du sens choisi pour $\mathbf{e}$, et de la cellule choisie entre $\Omega_{1|\mathcal{S}}$ et $\Omega_{2|\mathcal{S}}$. Ces notations sont illustrées en Figure 4.8. Les vecteurs $\mathbf{e}$ et $\mathbf{n}_b$ forment une base orthogonale du plan de la face T. Le vecteur $\mathbf{w}$ peut donc s'écrire

$$\mathbf{w} = (\mathbf{w}.\mathbf{e})\mathbf{e} + (\mathbf{w}.\mathbf{n}_b)\mathbf{n}_b.$$

$\mathbf{e}$ est un vecteur contenu dans le bissecteur de $\mathbf{v}_1$ et $\mathbf{v}_2$, nous avons donc $\mathbf{e}.(\mathbf{v}_1-\mathbf{v}_2)=0$, et donc $\mathbf{w}.(\mathbf{v}_1-\mathbf{v}_2) = (\mathbf{w}.\mathbf{n}_b)(\mathbf{n}_b.(\mathbf{v}_1-\mathbf{v}_2))$. À partir de l'Équation 4.4.7, nous obtenons donc

$$\begin{aligned}
\mathrm{d}\mathbf{p}_i.\mathbf{n}_b &= \frac{\mathrm{d}\mathbf{v}_1.(\mathbf{v}_1-\mathbf{p}_i) - \mathrm{d}\mathbf{v}_2.(\mathbf{v}_2-\mathbf{p}_i)}{(\mathbf{w}.\mathbf{n}_b)(\mathbf{n}_b.(\mathbf{v}_1-\mathbf{v}_2))}\mathbf{w}.\mathbf{n}_b \\
&= \frac{\mathrm{d}\mathbf{v}_1.(\mathbf{v}_1-\mathbf{p}_i) - \mathrm{d}\mathbf{v}_2.(\mathbf{v}_2-\mathbf{p}_i)}{\mathbf{n}_b.(\mathbf{v}_1-\mathbf{v}_2)} \\
&= \mathbf{n}_b^t\frac{\partial\mathbf{p}_i}{\partial\mathbf{v}_1}\mathrm{d}\mathbf{v}_1 + \mathbf{n}_b^t\frac{\partial\mathbf{p}_i}{\partial\mathbf{v}_2}\mathrm{d}\mathbf{v}_2.
\end{aligned} \tag{4.4.8}$$

De manière remarquable, l'Équation 4.4.8 ne fait plus apparaître la direction $\mathbf{w}$, et seule la position $\mathbf{p}_i$ du point d'intersection entre la droite et le bissecteur est utile. Lorsque la normale $\mathbf{n}_T$ de T est contenue dans le plan du bissecteur, nous avons alors $\mathbf{n}_b.(\mathbf{v}_1-\mathbf{v}_2) = \|\mathbf{v}_1-\mathbf{v}_2\|$, et nous retrouvons le résultat d'Asami [2, Équation 34].

Soit $[\mathbf{c}_1,\mathbf{c}_2]$ une arête du bord d'une cellule de Voronoï restreinte $\Omega_{1|\mathcal{S}}$ d'un site $\mathbf{v}_1$. Cette arête peut être de deux sortes :

une portion d'arête du bord de $\mathcal{S}$, dans ce cas, les sommets $\mathbf{c}_1$ et $\mathbf{c}_2$ sont de type i ou ii. Tout d'abord si $\mathbf{c}_1$ est de type i, il ne varie pas en fonction des sites. S'il est de type ii, il s'agit de l'intersection entre l'arête du bord et un bissecteur entre $\mathbf{v}_1$ et un autre site $\mathbf{v}_2$. Nous pouvons appliquer ici l'Équation 4.4.7, pour montrer que les variations de $\mathbf{c}_1$ par rapport à $\mathbf{v}_1$ ou $\mathbf{v}_2$ sont dans la direction de l'arête $[\mathbf{c}_1,\mathbf{c}_2]$, et le produit scalaire avec la normale du bord est donc nul. Le même raisonnement s'applique sur $\mathbf{c}_2$, ainsi pour tout point $\mathbf{x} = u\mathbf{c}_1 + (1-u)\mathbf{c}_2$ avec $u\in[0,1]$, les variations de $\mathbf{x}$ sont orthogonales à la normale du bord, et nous avons

$$\frac{\partial\mathbf{x}}{\partial\mathbf{V}}.\mathbf{n}_{[\mathbf{c}_1,\mathbf{c}_2]} = \mathbf{0}^t,$$

où $\mathbf{n}_{[\mathbf{c}_1,\mathbf{c}_2]}$ est la normale de l'arête $[\mathbf{c}_1,\mathbf{c}_2]$ dans le plan de T, orientée vers l'extérieur de $\Omega_{1|\mathcal{S}}$.

l'intersection entre une face de Voronoï et une face T de $\mathcal{S}$ et les sommets $\mathbf{c}_1$ et $\mathbf{c}_2$ sont de type ii ou iii. Si $\mathbf{c}_1$ est de type ii, il s'agit donc de l'intersection entre une arête de $\mathcal{S}$ et le bissecteur. D'après l'Équation 4.4.8, nous avons alors

$$\frac{\partial \mathbf{x}}{\partial \mathbf{v}_1}.\mathbf{n}_{[\mathbf{c}_1,\mathbf{c}_2]} = \left(\frac{\mathbf{v}_1 - \mathbf{c}_1}{\mathbf{n}_{[\mathbf{c}_1,\mathbf{c}_2]}.(\mathbf{v}_1 - \mathbf{v}_2)}\right)^t \quad \text{et} \quad \frac{\partial \mathbf{x}}{\partial \mathbf{v}_2}.\mathbf{n}_{[\mathbf{c}_1,\mathbf{c}_2]} = \left(\frac{\mathbf{v}_2 - \mathbf{c}_1}{\mathbf{n}_{[\mathbf{c}_1,\mathbf{c}_2]}.(\mathbf{v}_2 - \mathbf{v}_1)}\right)^t.$$

Si $\mathbf{c}_1$ est de type iii, il est alors à l'intersection d'une arête de Voronoï correspondant à $\mathbf{v}_1$, $\mathbf{v}_2$ et un troisième site $\mathbf{v}_3$ avec T. $\mathbf{c}_1$ varie donc en fonction de trois sites. Nous noterons $\mathbf{w}_{k,l}$ le vecteur directeur de la droite correspondant à l'intersection entre T et le bissecteur de $\mathbf{v}_k$ et $\mathbf{v}_l$. Pour calculer la dérivée de $\mathbf{c}_1$ par rapport à $\mathbf{v}_1$, nous définissons $\mathbf{c}_1$ comme l'intersection entre le bissecteur de $\mathbf{v}_1$ et $\mathbf{v}_2$ et la droite de direction $\mathbf{w}_{2,3}$ passant par $\mathbf{c}_1$. La direction de cette droite ne varie pas en fonction de $\mathbf{v}_1$, et en appliquant l'Équation 4.4.8, nous retrouvons l'expression précédente. Nous appliquons le même raisonnement pour la dérivée par rapport à $\mathbf{v}_2$ en utilisant la direction $\mathbf{w}_{1,3}$, qui nous permet également de retrouver l'expression précédente. Enfin pour la dérivée par rapport à $\mathbf{v}_3$, nous définissons $\mathbf{c}_1$ comme l'intersection entre $\mathbf{w}_{1,2}$ et le bissecteur de $\mathbf{v}_1$ et $\mathbf{v}_3$. D'après l'Équation 4.4.7, les variations de $\mathbf{c}_1$ sont alors colinéaires à $\mathbf{w}_{1,2}$ qui correspond à la direction de notre arête du bord, et est donc orthogonal à sa normale. Finalement, en appliquant le même procédé sur $\mathbf{c}_2$, pour un point $\mathbf{x} = u\mathbf{c}_1 + (1-u)\mathbf{c}_2$ avec $u \in [0,1]$, nous avons

$$\frac{\partial \mathbf{x}}{\partial \mathbf{v}_1}.\mathbf{n}_{[\mathbf{c}_1,\mathbf{c}_2]} = \left(\frac{\mathbf{v}_1 - \mathbf{x}}{\mathbf{n}_{[\mathbf{c}_1,\mathbf{c}_2]}.(\mathbf{v}_1 - \mathbf{v}_2)}\right)^t \quad \text{et} \quad \frac{\partial \mathbf{x}}{\partial \mathbf{v}_2}.\mathbf{n}_{[\mathbf{c}_1,\mathbf{c}_2]} = \left(\frac{\mathbf{v}_2 - \mathbf{x}}{\mathbf{n}_{[\mathbf{c}_1,\mathbf{c}_2]}.(\mathbf{v}_2 - \mathbf{v}_1)}\right)^t. \tag{4.4.9}$$

De façon remarquable, cette équation ne dépend pas du type des sommets $\mathbf{c}_1$ et $\mathbf{c}_2$.

Expression finale du gradient par rapport aux sites

À partir de l'Équation 4.4.6, en séparant le bord arête par arête, la différentielle de $\mathscr{F}$ par rapport aux positions des sites est alors

$$\mathrm{d}\mathscr{F} = \sum_{\mathbf{v}_k} \sum_{[\mathbf{c}_1,\mathbf{c}_2]} \int_{[\mathbf{c}_1,\mathbf{c}_2]} \|f(\mathbf{x}) - \tilde{f}_k\|^2 \left(\frac{\mathbf{v}_k - \mathbf{x}}{\mathbf{n}_{[\mathbf{c}_1,\mathbf{c}_2]}.(\mathbf{v}_k - \mathbf{v}_l)}.\mathrm{d}\mathbf{v}_k - \frac{\mathbf{v}_l - \mathbf{x}}{\mathbf{n}_{[\mathbf{c}_1,\mathbf{c}_2]}.(\mathbf{v}_k - \mathbf{v}_l)}.\mathrm{d}\mathbf{v}_l\right) \mathrm{d}\mathbf{x},$$

où $\mathbf{v}_l$ est le site dont la cellule de Voronoï restreinte est adjacente à celle de $\mathbf{v}_k$ le long de $[\mathbf{c}_1,\mathbf{c}_2]$. Chaque arête apparaît en réalité deux fois dans cette somme, une fois pour chaque site qu'elle borde. Ces deux apparitions se font avec des normales opposées, car les normales sont orientées vers l'extérieur de la cellule $\Omega_{k|\mathcal{S}}$ dans laquelle l'arête apparaît. En regroupant les termes faisant apparaître les différentielles des mêmes sites, nous obtenons

$$\begin{aligned}
\mathrm{d}\mathscr{F} &= \sum_{\mathbf{v}_k} \sum_{[\mathbf{c}_1,\mathbf{c}_2]} \int_{[\mathbf{c}_1,\mathbf{c}_2]} \left(\|f(\mathbf{x}) - \tilde{f}_k\|^2 - \|f(\mathbf{x}) - \tilde{f}_l\|^2\right) \left(\frac{\mathbf{v}_k - \mathbf{x}}{\mathbf{n}_{[\mathbf{c}_1,\mathbf{c}_2]}.(\mathbf{v}_k - \mathbf{v}_l)}.\mathrm{d}\mathbf{v}_k\right) \mathrm{d}\mathbf{x} \\
&= \sum_{\mathbf{v}_k} \left[\sum_{[\mathbf{c}_1,\mathbf{c}_2]} \int_{[\mathbf{c}_1,\mathbf{c}_2]} \left(\|f(\mathbf{x}) - \tilde{f}_k\|^2 - \|f(\mathbf{x}) - \tilde{f}_l\|^2\right) \frac{\mathbf{v}_k - \mathbf{x}}{\mathbf{n}_{[\mathbf{c}_1,\mathbf{c}_2]}.(\mathbf{v}_k - \mathbf{v}_l)} \mathrm{d}\mathbf{x}\right].\mathrm{d}\mathbf{v}_k \\
&= \sum_{\mathbf{v}_k} \frac{\partial \mathscr{F}}{\partial \mathbf{v}_k}.\mathrm{d}\mathbf{v}_k,
\end{aligned} \tag{4.4.10}$$

où $\tilde{f}_l$ est la valeur d'approximation associée au site $\mathbf{v}_l$ dont la cellule restreinte jouxte celle de $\mathbf{v}_k$ le long de l'arête $[\mathbf{c}_1,\mathbf{c}_2]$.

4.4.3 Calcul et optimisation

Nous détaillerons ici le procédé suivi pour obtenir une configuration des sites minimisant localement $\mathscr{F}$.

Conjectures sur la continuité de la fonction objectif

Nous ne disposons pas à l'heure actuelle de résultats satisfaisants sur la continuité de $\mathscr{F}$. Il est clair que $\mathscr{F}$ est discontinue lorsqu'une face T de $\mathcal{S}$ et une face de Voronoï ont une intersection dégénérée de dimension 2. Dans une telle configuration, selon que la portion de T est attribuée à l'une ou l'autre des cellules de Voronoï adjacentes, si ces deux cellules n'ont pas la même valeur d'approximation, la valeur de $\mathscr{F}$ sera différente. $\mathscr{F}$ est également discontinue lorsque deux sites ont la même position. Dans ce cas, la région de Voronoï des deux sites est ambiguë et une portion de $\mathcal{S}$ contenue dans cette région peut être attribuée à l'un ou l'autre des sites. En dehors de ces cas, $\mathscr{F}$ est continue. Il est par contre plus difficile d'évaluer la continuité du gradient de $\mathscr{F}$. En l'absence d'ambiguïté, le gradient est continu. Notre conjecture est que le gradient reste continu en cas d'ambiguïtés ponctuelles, mais que des discontinuités apparaissent pour des ambiguïtés de dimension 1, c'est à dire lorsqu'une arête de Voronoï est comprise dans une face de $\mathcal{S}$ ou une arête de $\mathcal{S}$ est comprise dans une face de Voronoï. Enfin la dérivée seconde serait discontinue au niveau des configurations comportant ne serait-ce qu'un sommet ambigu.

Les discontinuités en cas d'ambiguïté ponctuelles sont réellement gênantes. En effet, pour modifier la combinatoire d'un diagramme de Voronoï restreint, il est nécessaire de passer par une situation ambiguë. Autrement dit, si la configuration initiale des sites $\mathbf{V}_0$ et la configuration optimale selon $\mathscr{F}$ n'ont pas la même combinatoire, le chemin suivi par l'optimisation devra nécessairement passer par une configuration comportant une ambiguïté.

Calcul approché des intégrales

Pour calculer $\mathscr{F}$ et son gradient, il nous est en particulier nécessaire de pouvoir calculer les termes

$$\int_{\Omega_{k|\mathcal{S}}} f(\mathbf{x})\mathrm{d}\mathbf{x} \quad \text{et} \quad \int_{[\mathbf{c}_1,\mathbf{c}_2]} f(\mathbf{x})\mathrm{d}\mathbf{x},$$

correspondant respectivement à l'intégrale de la fonction approximée f sur une cellule de Voronoï restreinte ou une arête du bord d'une telle cellule. Pour rester le plus générique possible, nous avons supposé que la seule chose que nous puissions connaître était la valeur de $f(\mathbf{x})$ pour un point $\mathbf{x}$ quelconque. Nous avons donc utilisé les méthodes classiques de quadrature pour évaluer les intégrales. Pour les intégrales sur les surfaces, nous avons utilisé la méthode de Dunavant [30], fournissant des points de quadrature et des poids associés sur des triangles, et pour les intégrales sur les segments du bord, nous avons utilisé les points de quadrature de Gauß-Legendre. Nous avons également essayé un échantillonnage aléatoire uniforme, tiré à chaque itération, mais les résultats se sont avérés moins bons, comme le montre la Figure 4.9.

Descente de gradient normalisée

Il est possible en ne disposant que de la valeur et du gradient d'une fonction objectif d'utiliser la méthode LBFGS pour en chercher un minimum. Cette méthode a notamment été appliquée avec succès pour obtenir des diagrammes de Voronoï barycentriques. Pour que cette méthode fonctionne correctement, il est nécessaire que la fonction minimisée soit C^2, ce qui n'est pas partout le cas pour notre fonction objectif, en particulier en cas d'ambiguïté. Dans notre cas, le solveur se bloque après quelques itérations dans une configuration clairement médiocre, ce qui tend à confirmer notre hypothèse selon laquelle $\mathscr{F}$ est discontinue en cas d'ambiguïté dans le diagramme de Voronoï restreint. Nous nous sommes donc plutôt orienté vers une méthode de descente de gradient, qui s'est révélée suffisamment robuste pour obtenir un résultat satisfaisant.

Entrées :
- $\mathcal{S}$ la surface échantillonnée
- f une fonction définie sur $\mathcal{S}$
- $\mathbf{V}$ un ensemble initial de sites
- $\tilde{\mathbf{F}}$ un ensemble initial de valeurs d'approximations

Sorties :
- $\mathbf{V}^*$ l'ensemble optimisé des sites
- $\tilde{\mathbf{F}}^*$ l'ensemble optimisé des valeurs d'approximation

Données :
- PasDuSite un pas d'optimisation par site
- PasDeLApprox un pas d'optimisation par valeur d'approximation

Algorithme
- $\mathbf{V}^* \leftarrow \mathbf{V}$;
- $\tilde{\mathbf{F}}^* \leftarrow \tilde{\mathbf{F}}$;
- **tant que** *le nombre maximal d'itérations n'est pas atteint* **faire**
  - Calculer le diagramme de Voronoï de $\mathbf{V}^*$ restreint par $\mathcal{S}$;
  - **pour chaque** *site* $\mathbf{v}_k^*$ *de* $\mathbf{V}^*$ **faire**
    - `// En utilisant l'Équation 4.4.5 et l'Équation 4.4.10`
    - $\mathbf{g}_{\mathbf{v}_k} \leftarrow \dfrac{\partial \mathscr{F}}{\partial \mathbf{v}_k}, \quad g_{\tilde{f}_k} \leftarrow \dfrac{\partial \mathscr{F}}{\partial \tilde{f}_k}$;
    - `// Le gradient par rapport aux sites est normalisé`
    - $\mathbf{g}_{\mathbf{v}_k} \leftarrow \dfrac{\mathbf{g}_{\mathbf{v}_k}}{\|\mathbf{g}_{\mathbf{v}_k}\|}$;
    - `// Descente de gradient`
    - $\mathbf{v}_k^* \leftarrow \mathbf{v}_k^* - \mathsf{PasDuSite}[\mathbf{v}_k] \times \mathbf{g}_{\mathbf{v}_k}$;
    - $\tilde{f}_k^* \leftarrow \tilde{f}_k^* - \mathsf{PasDeLApprox}[\mathbf{v}_k] \times g_{\tilde{f}_k}$;
- **renvoyer** $\mathbf{V}^*$ *et* $\tilde{\mathbf{F}}^*$;

Algorithme 4.2: `optimise_échantillonnage(`$\mathcal{S}$,f,$\mathbf{V}$,$\tilde{\mathbf{F}}$`)` : optimisation d'un ensemble de sites $\mathbf{V}$ et d'un ensemble de variables d'approximation $\tilde{\mathbf{F}}$ pour l'échantillonnage d'une fonction f définie sur une surface $\mathcal{S}$.

Pour optimiser la position des sites, nous suivons l'Algorithme 4.2. Le gradient fournit pour chaque site la direction dans laquelle le déplacer pour faire diminuer la fonction objectif. Dans l'Équation 4.4.10, le vecteur $(\mathbf{v}_k - \mathbf{v}_l)$ entre deux sites voisins apparaît au dénominateur. Ainsi, lorsque deux sites voisins se rapprochent, la norme de ce vecteur devient arbitrairement petite, et la norme du gradient tend vers l'infini. Comme nous l'avons mentionné plus haut, lorsque deux sites coïncident, $\mathscr{F}$ est discontinue. C'est la raison pour laquelle dans l'Algorithme 4.2, la portion du gradient correspondant aux sites est normalisée pour ne garder que la direction. L'amplitude du déplacement des sites est ensuite contrôlé par les pas attribués à chaque site.

La gestion du pas dans la mise à jour des variables dans une descente de gradient est en général un problème délicat. Nous noterons $\delta^s_{\mathbf{v}_k}$ le pas d'un site $\mathbf{v}_k$ après s itérations, et $\delta^s_{\tilde{f}_k}$ celui de la variable d'approximation associée à ce site. Ici, nous avons choisi d'utiliser des pas différents pour le gradient en fonction des sites et le gradient en fonction des valeurs d'approximation. L'utilisation de pas différents permet de tenir compte du fait que les variables spatiales correspondant aux positions des sites ne sont en général pas de la même grandeur que les variables d'approximation. Si la surface $\mathcal{S}$ est comprise dans la boîte $[0,1]^3$, un pas de taille 1 pour les sites sera trop grand, et la descente de gradient ne convergera pas. En revanche, si la fonction f fournit des valeurs dans l'intervalle $[-10^3,10^3]$, un pas de 1 peut être trop faible. Il n'est donc en général pas souhaitable d'utiliser le même pas pour faire varier les sites et les valeurs d'approximation.

Lorsque l'optimisation approche d'un minimum, le fait de normaliser le gradient fait que les sites se mettent à osciller autour de leur position optimale. Pour éviter ce problème, nous utilisons un mécanisme qui diminue le pas d'optimisation lorsque la direction du gradient varie trop pour un site. Au contraire, lorsque le gradient persiste dans la même direction, le pas est augmenté. Pour détecter les oscillations, nous utilisons les directions du gradient par rapport à la position d'un site pour deux itérations successives. Le produit scalaire de ces deux directions, lorsqu'il est positif, signifie que les deux directions ne forment pas un angle de plus de $\frac{\pi}{2}$. Lorsqu'il est négatif, l'angle est alors supérieur à $\frac{\pi}{2}$, et nous considérons que le site oscille. Ainsi si $\delta^s_{\mathbf{v}_k}$ est le pas d'un site à une itération s donnée, et que $\mathbf{g}^s_{\mathbf{v}_k}$ est le gradient normalisé de $\mathscr{F}$ par rapport à ce site à cette même itération, alors à l'itération $s+1$ nous mettons à jour le pas en utilisant la règle

$$\delta^{s+1}_{\mathbf{v}_k} = \begin{cases} \sigma\delta^s_{\mathbf{v}_k} & \text{si } \mathbf{g}^s_{\mathbf{v}_k}.\mathbf{g}^{s+1}_{\mathbf{v}_k} < -\tau \\ \frac{1}{\sigma}\delta^s_{\mathbf{v}_k} & \text{si } \mathbf{g}^s_{\mathbf{v}_k}.\mathbf{g}^{s+1}_{\mathbf{v}_k} > \tau \\ \delta^s_{\mathbf{v}_k} & \text{sinon} \end{cases}$$

où σ est un paramètre réglant la force de l'atténuation ou de l'amplification selon que respectivement un cas d'oscillation ou de persistance est détectée. Le paramètre τ règle le seuil de détection des oscillations et de la persistance. Pour tous les résultats que nous présenterons, nous avons utilisé $\sigma = 0{,}9$ et $\tau = 0{,}3$.

Les seuls paramètres restant donc à fournir sont le pas initial pour les sites, et celui pour les valeurs d'approximation. Empiriquement, nous avons constaté que si la valeur d'approximation n'est pas rapidement mise à jour par rapport aux déplacements des sites, l'optimisation diverge. Il est donc nécessaire de veiller à ce que les sites varient suffisamment lentement pour permettre aux variables d'approximation de se mettre à jour. Pour une surface $\mathcal{S}$ dont la boîte englobante a son plus grand côté de longueur 1, nous avons utilisé

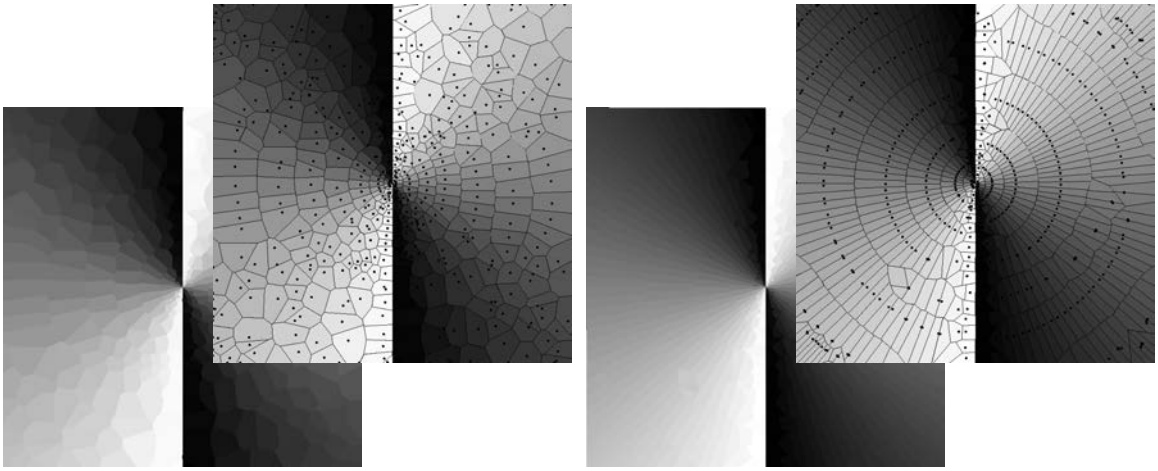


FIGURE 4.9 – Approximation de la fonction $f(x,y) = \arctan\left(\frac{y}{x}\right)$. À gauche, les intégrales sont approximées en utilisant un échantillonnage aléatoire (10 points par triangle ou par arête, générés à chaque intégrale). À droite les points de quadrature de Dunavant [30] sont utilisés pour les intégrales surfaciques (trois points par triangle), et ceux de GaußLegendre pour les intégrales sur les arêtes (deux points par arête). Le résultat de notre optimisation est bien meilleur en utilisant les quadratures. Nous pouvons constater que les sites s'alignent naturellement pour former des courbes suivant le gradient de la fonction approximée, et ainsi faire en sorte que leurs cellules soient fines dans la direction du gradient, où f varie le plus, et au contraire allongées dans la direction orthogonale où f varie peu. De plus les discontinuités de la fonction sont très bien capturées par des paires de sites alignés de part et d'autre.

un pas initial pour les sites de $\delta^0_{\mathbf{v}_k} = 10^{-2}$. Pour les valeurs d'approximation, le fait de ne pas normaliser le gradient par rapport à ces variables nous rend moins sensibles à la grandeur des valeurs de f. Par contre, dans l'Équation 4.4.5, l'aire de la cellule de Voronoï restreinte de $\mathbf{v}_k$ intervient, ce qui se traduit dans nos expérience par le fait que l'optimisation des variables $\{\tilde{f}_k\}_k$ est sensible à la fois à l'aire de la surface $\mathcal{S}$ et au nombre de points répartis dessus. Dans tous nos résultats, nous avons utilisé un pas

$$\delta^0_{\tilde{f}_k} = \frac{n}{4|\mathcal{S}|},$$

où n est le nombre de sites, et $|\mathcal{S}|$ l'aire de $\mathcal{S}$.

Résultats

Pour illustrer le comportement général de notre méthode, nous considérons tout d'abord le cas de l'approximation de la fonction $f(x,y) = \arctan\left(\frac{y}{x}\right)$ (Figure 4.9). Cette fonction varie sans oscillations, mais comporte une discontinuité au niveau de la droite $x = 0$. Ici, la surface $\mathcal{S}$ utilisée est le carré $\{(x,y,z), (x,y) \in [-1,1]^2$ et $z = 0$, et les sites ne sont pas contraints à rester sur le plan $z = 0$, mais peuvent s'éloigner de la surface. De manière générale, ils se répartissent de telle sorte à allonger leurs cellules dans la direction où f varie le moins en se répartissant le long de courbes suivant la direction du gradient de f. Au niveau de la discontinuité les sites se placent par paires de part et d'autre pour que le bissecteur les séparant s'ajuste sur la discontinuité. Ils minimisent ainsi l'erreur d'approximation. Dans une telle situation le bord d'une cellule restreinte coïncide avec une discontinuité de f. Sur ce bord l'intégrale de l'Équation 4.4.10 n'est pas définie et donc le gradient de $\mathscr{F}$ non plus.

Notre méthode converge tout de même dans cette situation grâce au fait que, pour chaque site, la composante du gradient de $\mathscr{F}$ par rapport à ses coordonnées est normalisée et le pas $\delta^s_{\mathbf{v}_k}$ de mise à jour est contrôlé pour limiter les oscillations.

La Figure 4.10 montre l'approximation d'une fonction ayant des variations moins régulières, fournie par une image. Selon le nombre de sites utilisés, certains détails ne pourront pas être capturés. Nous montrons enfin l'approximation de l'illumination directe d'une surface sur la Figure 4.11. Lorsque $\mathcal{S}$ comporte des zones fine, les cellules restreintes peuvent facilement avoir plusieurs composantes connexes ayant des illuminations très différentes. Typiquement lorsqu'une composante connexe est éclairée, il est probable que les autres soient à l'ombre. Dans ce cas, les sites doivent donc se positionner pour éviter d'avoir plusieurs composantes connexes dans leurs cellules restreintes. Inversement, lorsque toutes les composantes connexes sont à l'ombre, un seul site peut être utilisé pour approximer plusieurs couches de $\mathcal{S}$. En pratique, pour notre méthode, cette situation peut rendre difficile la convergence de l'optimisation, car les sites se mettent à osciller malgré la gestion du pas de mise à jour des sites.

4.4.4 Valeur moyenne comme variable d'approximation

Modification de la fonction objectif

Nous proposons ici une modification de la fonction objectif $\mathscr{F}$, permettant de rendre l'optimisation plus stable et du supprimer le paramètre correspondant au pas de mise à jour des valeurs d'approximation. Cette évolution est fondée sur le constat que pour que l'optimisation converge il est nécessaire que les valeurs d'approximation s'adaptent rapidement à une nouvelle position d'un site. Une méthode classique pour optimiser un problème ayant des variables de grandeurs différentes, comme ici les variables de position et les variables d'approximation, consiste à optimiser ces variables séparément dans des passes séparées. Dans notre cas, cette méthode reviendrait à successivement optimiser la position des sites en supposant les variables d'approximation constantes, puis à optimiser les variables d'approximation en considérant les sites constants.

Pour une position fixe des sites, l'obtention de la valeur d'approximation optimale pour un site ne requiert en réalité pas d'effectuer une descente de gradient. En effet la valeur d'approximation $\tilde{f}_k$ d'un site $\mathbf{v}_k$ doit minimiser l'expression

$$\int_{\Omega_{k|\mathcal{S}}} \|f(\mathbf{x}) - \tilde{f}_k\|^2 \mathrm{d}\mathbf{x} = \tilde{f}_k^2 \int_{\Omega_{k|\mathcal{S}}} \mathrm{d}\mathbf{x} - 2\tilde{f}_k \int_{\Omega_{k|\mathcal{S}}} f(\mathbf{x})\mathrm{d}\mathbf{x} + \int_{\Omega_{k|\mathcal{S}}} f(\mathbf{x})^2 \mathrm{d}\mathbf{x},$$

qui est un simple polynôme du second degré en $\tilde{f}_k$. Il est donc minimisé pour

$$\tilde{f}_k = \frac{\int_{\Omega_{k|\mathcal{S}}} f(\mathbf{x})\mathrm{d}\mathbf{x}}{\int_{\Omega_{k|\mathcal{S}}} \mathrm{d}\mathbf{x}} = \tilde{f}_k(\mathbf{V}), \tag{4.4.11}$$

c'est à dire que $\tilde{f}_k$ doit correspondre à la valeur moyenne de f dans la cellule de Voronoï restreinte $\Omega_{k|\mathcal{S}}$. Cette expression nous décrit donc $\tilde{f}_k$ comme une fonction de l'ensemble des sites, que nous noterons par extension $\tilde{f}_k(\mathbf{V})$. Il nous est alors possible de remplacer dans l'expression de $\mathscr{F}$ fournie en Équation 4.4.2 pour obtenir la nouvelle fonction objectif

$$\mathscr{G}(\mathbf{V}) = \sum_{\mathbf{v}_k \in \mathbf{V}} \int_{\Omega_{k|\mathcal{S}}} \|f(\mathbf{x}) - \tilde{f}_k(\mathbf{V})\|^2 \mathrm{d}\mathbf{x},$$

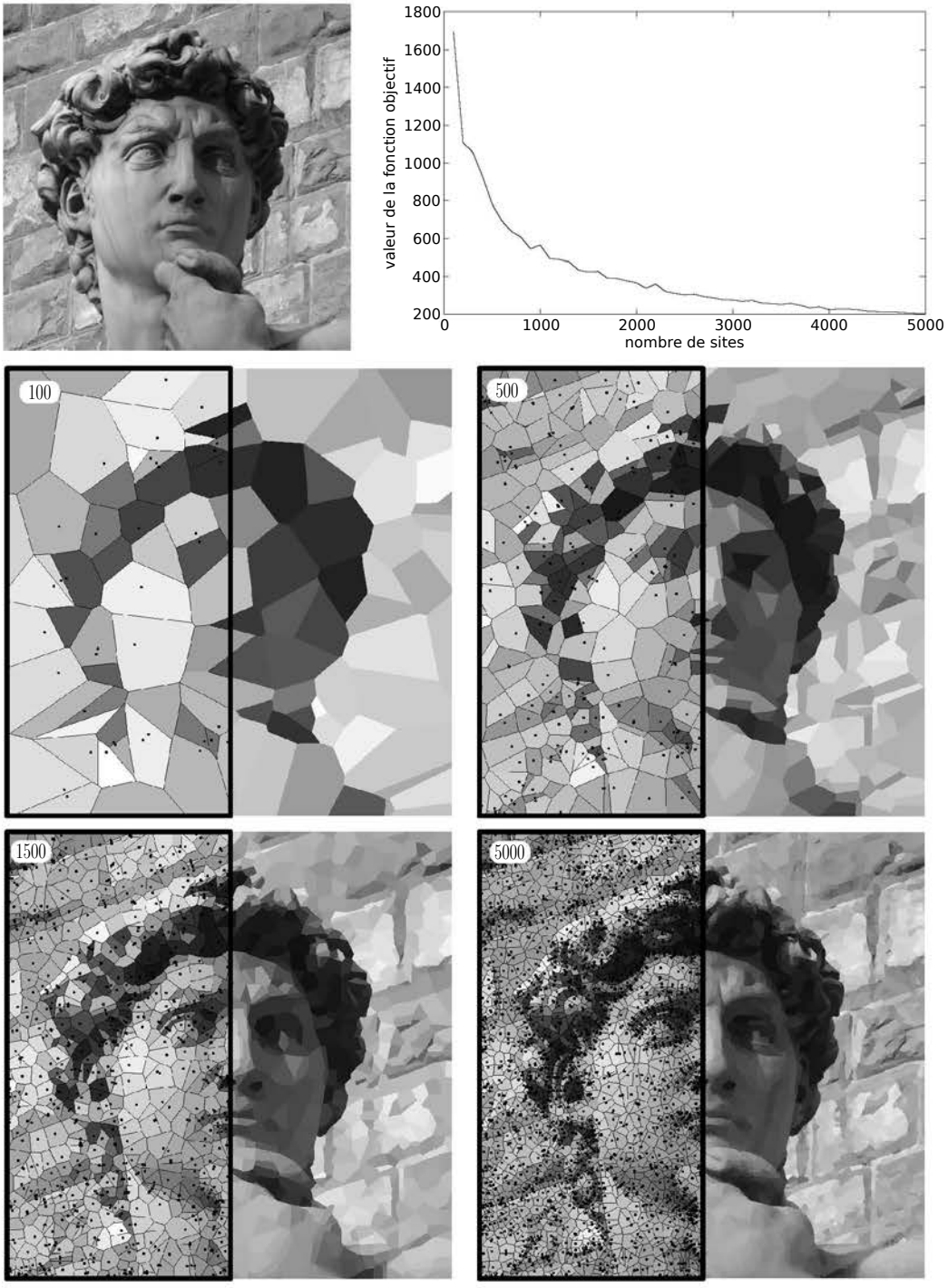


FIGURE 4.10 – Évolution de l'approximation en fonction du nombre de sites alloués pour l'approximation. Ici la fonction approximée est une photo en niveaux de gris de la statue de David. De manière prévisible, la valeur finale obtenue pour notre fonction objectif décroît rapidement lorsque le nombre de sites augmente. La fonction comporte ici de nombreuses discontinuités, qui seront plus ou moins bien capturées en fonction du nombre de sites. Les résultats obtenus pour 100, 500, 1500 et 5000 sites sont illustrés.

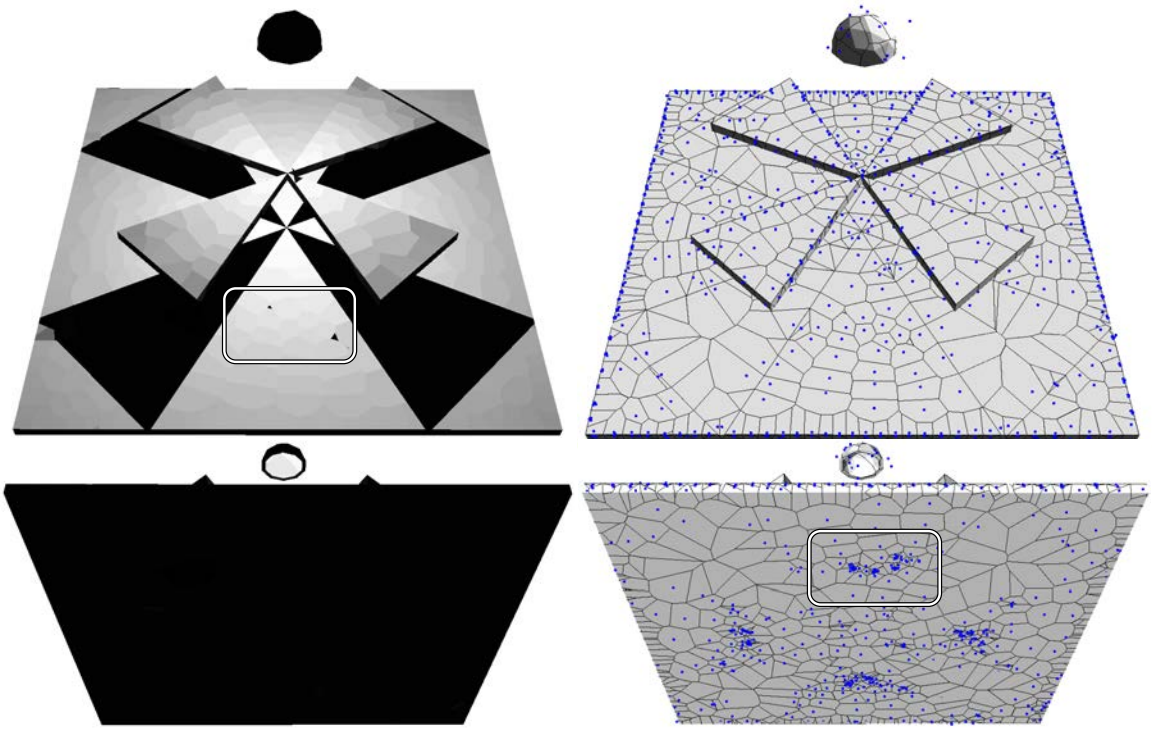


FIGURE 4.11 – Approximation de l'illumination directe sur une surface avec 1500 sites. La source est une lampe ponctuelle placée dans le petit bol en haut de la scène. Ici, une hélice jette une ombre sur la plaque située en dessous. La complexité de cette scène réside dans le fait que pour les surfaces, l'écart entre la face éclairée et la face opposée à l'ombre est faible, et que lorsqu'une cellule restreinte a une composante connexe sur le côté éclairé et une sur le côté à l'ombre, l'approximation est mauvaise. Les sites à l'ombre, bien que la fonction à approximer dans leurs cellules restreintes soit constante, doivent s'aligner sur les sites voisins du côté éclairé pour empêcher les cellules restreintes d'avoir des composantes de part et d'autre. Ici, la convergence de l'optimisation est difficile, et le résultat contient toujours quelques artéfacts dûs à des grappes de sites plus denses du côté ombragé que du côté éclairé (encadré).

qui ne dépend plus que des positions des sites. Cette fonction objectif correspond en réalité à la *variance* de f au sein de chaque cellule de Voronoï restreinte. En développant dans l'intégrale, nous retrouvons alors une formulation classique de la variance :

$$\begin{aligned}\mathscr{G}(\mathbf{V}) &= \sum_{\mathbf{v}_k \in \mathbf{V}} \int_{\Omega_{k|\mathcal{S}}} f(\mathbf{x})^2 \mathrm{d}\mathbf{x} - \int_{\Omega_{k|\mathcal{S}}} \tilde{f}_k(\mathbf{V})^2 \mathrm{d}\mathbf{x} \\ &= \int_{\mathcal{S}} f(\mathbf{x})^2 \mathrm{d}\mathbf{x} - \sum_{\mathbf{v}_k \in \mathbf{V}} \tilde{f}_k(\mathbf{V})^2 |\Omega_{k|\mathcal{S}}|, \end{aligned} \tag{4.4.12}$$

où $|\Omega_{k|\mathcal{S}}|$ est l'aire de la cellule restreinte $\Omega_{k|\mathcal{S}}$. Dans cette formulation il ne serait en réalité pas nécessaire de conserver le premier terme, qui ne dépend en réalité pas des positions des sites.

Nouvelle expression du gradient

À partir de l'Équation 4.4.12, la différentielle de $\mathscr{G}$ est

$$\begin{aligned}\mathrm{d}\mathscr{G} &= -\sum_{\mathbf{v}_k \in \mathbf{V}} \tilde{f}_k^2 \mathrm{d}|\Omega_{k|\mathcal{S}}| + 2\tilde{f}_k |\Omega_{k|\mathcal{S}}| \mathrm{d}\tilde{f}_k \\ &= -\sum_{\mathbf{v}_k \in \mathbf{V}} \tilde{f}_k^2 \mathrm{d}|\Omega_{k|\mathcal{S}}| + 2\tilde{f}_k |\Omega_{k|\mathcal{S}}| \frac{\mathrm{d}\left[\int_{\Omega_{k|\mathcal{S}}} f(\mathbf{x})\mathrm{d}\mathbf{x}\right] |\Omega_{k|\mathcal{S}}| - \int_{\Omega_{k|\mathcal{S}}} f(\mathbf{x})\mathrm{d}\mathbf{x}\, \mathrm{d}|\Omega_{k|\mathcal{S}}|}{|\Omega_{k|\mathcal{S}}|^2} \\ &= \sum_{\mathbf{v}_k \in \mathbf{V}} \tilde{f}_k^2 \mathrm{d}|\Omega_{k|\mathcal{S}}| - 2\tilde{f}_k \mathrm{d}\left[\int_{\Omega_{k|\mathcal{S}}} f(\mathbf{x})\mathrm{d}\mathbf{x}\right]. \end{aligned}$$

En appliquant le théorème du transport de Reynolds sur les deux termes, et en utilisant l'Équation 4.4.8, nous obtenons

$$\mathrm{d}\mathscr{G} = \sum_{\mathbf{v}_k} \sum_{[\mathbf{c}_1,\mathbf{c}_2]} \int_{[\mathbf{c}_1,\mathbf{c}_2]} (\tilde{f}_k^2 - 2\tilde{f}_k f(\mathbf{x})) \left[\frac{\mathbf{v}_k - \mathbf{x}}{\mathbf{n}_{[\mathbf{c}_1,\mathbf{c}_2]} \cdot (\mathbf{v}_k - \mathbf{v}_l)} \mathrm{d}\mathbf{v}_k - \frac{\mathbf{v}_l - \mathbf{x}}{\mathbf{n}_{[\mathbf{c}_1,\mathbf{c}_2]} \cdot (\mathbf{v}_k - \mathbf{v}_l)} \mathrm{d}\mathbf{v}_l \right] \mathrm{d}\mathbf{x},$$

où $[\mathbf{c}_1,\mathbf{c}_2]$ est une arête du bord de $\Omega_{k|\mathcal{S}}$ et $\mathbf{v}_l$ le site dont la cellule de Voronoï restreinte jouxte celle de $\mathbf{v}_k$ le long de cette arête. En regroupant comme précédemment les termes faisant intervenir la même différentielle nous obtenons

$$\begin{aligned}\mathrm{d}\mathscr{G} &= \sum_{\mathbf{v}_k} \left[\sum_{[\mathbf{c}_1,\mathbf{c}_2]} \int_{[\mathbf{c}_1,\mathbf{c}_2]} (\tilde{f}_k^2 - 2\tilde{f}_k f(\mathbf{x}) - \tilde{f}_l^2 + 2\tilde{f}_l f(\mathbf{x})) \frac{\mathbf{v}_k - \mathbf{x}}{\mathbf{n}_{[\mathbf{c}_1,\mathbf{c}_2]} \cdot (\mathbf{v}_k - \mathbf{v}_l)} \mathrm{d}\mathbf{x} \right] \mathrm{d}\mathbf{v}_k \\ &= \sum_{\mathbf{v}_k} \left[\sum_{[\mathbf{c}_1,\mathbf{c}_2]} \int_{[\mathbf{c}_1,\mathbf{c}_2]} \left((\tilde{f}_k - f(\mathbf{x}))^2 - (\tilde{f}_l - f(\mathbf{x}))^2 \right) \frac{\mathbf{v}_k - \mathbf{x}}{\mathbf{n}_{[\mathbf{c}_1,\mathbf{c}_2]} \cdot (\mathbf{v}_k - \mathbf{v}_l)} \mathrm{d}\mathbf{x} \right] \mathrm{d}\mathbf{v}_k \\ &= \sum_{\mathbf{v}_k} \frac{\partial \mathscr{G}}{\partial \mathbf{v}_k} \mathrm{d}\mathbf{v}_k = \sum_{\mathbf{v}_k} \frac{\partial \mathscr{F}}{\partial \mathbf{v}_k} \mathrm{d}\mathbf{v}_k \end{aligned} \tag{4.4.13}$$

où nous retrouvons le résultat de l'Équation 4.4.10.

Pour optimiser $\mathscr{G}$, il n'est ainsi pas nécessaire de modifier la mise à jour des sites dans l'Algorithme 4.2, le seul changement consiste donc, au lieu d'effectuer également une descente de gradient sur $\tilde{f}_k$, à simplement mettre à jour à chaque itération $\tilde{f}_k$ en lui assignant

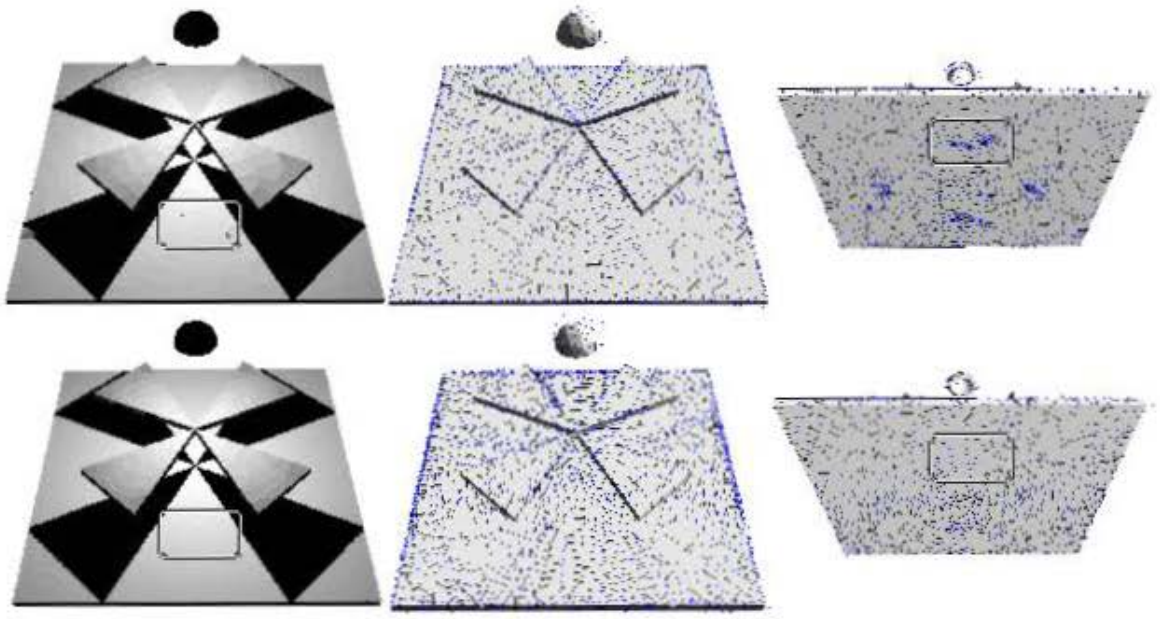


FIGURE 4.12 – Comparaisons entre la méthode précédente utilisant des variables d'approximation (en haut) et la méthode utilisant la moyenne des valeurs de f dans la cellule restreinte comme approximation (en bas). Dans les deux cas 1500 sites sont utilisés. Avec la nouvelle méthode, la convergence est plus stable, et les artéfacts correspondant à des composantes connexes multiples pour une cellule restreinte sont éliminés. Les grappes de sites présentes en haut à droite disparaissent en bas, ce qui laisse plus de sites pour réaliser l'approximation sur la face illuminée (en bas au centre).

la valeur moyenne de f dans la cellule de Voronoï restreinte de $\mathbf{v}_k$, selon l'Équation 4.4.11. Cette modification revient à utiliser le pas

$$\delta^s_{\tilde{f}_k} = \frac{1}{2|\Omega_{k|\mathcal{S}}|}$$

dans la méthode précédente, car avec ce pas, d'après l'Équation 4.4.5, la valeur d'approximation $\tilde{f}_k$ est amenée exactement à la valeur moyenne de f sur $\Omega_{k|\mathcal{S}}$. Cette valeur est cohérente avec le pas de $\frac{n}{4|\mathcal{S}|}$ précédemment choisi avec n le nombre de sites, car $\frac{|\mathcal{S}|}{n}$ est une approximation de l'aire d'une cellule de Voronoï restreinte.

Résultats

Cette évolution de la méthode de base rend l'optimisation plus stable, et permet en particulier de fournir de meilleurs résultats au niveau des surfaces ayant des parties fines. Une comparaison entre les deux méthodes dans ce cas est illustrée en Figure 4.12. D'une manière générale, dans les cas moins pathologiques, les résultats obtenus par les deux méthodes sont similaires, sauf lorsque certaines cellules de Voronoï restreintes deviennent très grandes. En effet, dans ce cas le pas d'optimisation $\delta^s_{\tilde{f}_k}$ utilisé dans la méthode de base devient inadapté et trop grand, ce qui fait osciller les valeurs d'approximation. De telles cellules apparaissent notamment ici dans les zones d'ombre où la valeur de f est constante, et peut donc être approximée par de grandes cellules. Notre méthode est également applicable pour des modèles plus complexes, comme celui illustré sur la Figure 4.13.

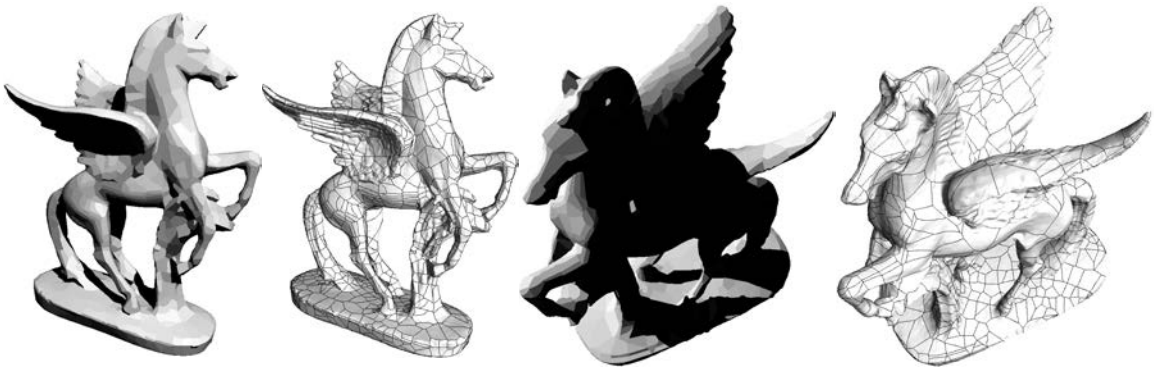


FIGURE 4.13 – Approximation de l'illumination sur un modèle usuel à l'aide de 1500 sites. Le modèle comporte des parties fines au niveau des ailes et des pattes. 85k faces, durée, nombre d'itérations.

4.5 Conclusion et perspectives

4.5.1 Contributions

Dans ce chapitre, nous avons étudié l'optimisation de la fonction objectif

$$\mathscr{G}(\mathbf{V}) = \sum_{\mathbf{v}_k \in \mathbf{V}} \int_{\Omega_{k|\mathcal{S}}} \|f(\mathbf{x}) - \tilde{f}_k(\mathbf{V})\|^2 \mathrm{d}\mathbf{x}.$$

Lorsque $f(\mathbf{x}) = \mathbf{x}$ et $\tilde{f}_k(\mathbf{V}) = \mathbf{v}_k$, nous retrouvons la fonction objectif minimisée pour obtenir un diagramme de Voronoï restreint. Pour l'optimisation, nous avons fourni l'expression du gradient de cette fonction objectif. Cette expression généralise les calculs réalisés par Asami [2] à la fois au cas d'un diagramme de Voronoï restreint, et en intégrant la fonction f. Par comparaison à la méthode de Lévy et Liu [62] pour le calcul du gradient de leur fonction objectif, il ne nous est pas nécessaire d'évaluer le gradient des sommets du diagramme de Voronoï restreint. De plus, notre méthode ne requiert pas l'évaluation de la dérivée de f. Il est par contre nécessaire d'en calculer l'intégrale sur des cellules de Voronoï restreintes. Nous approximons ces intégrales en utilisant des points de quadrature.

Nous avons ensuite appliqué ces résultats à l'optimisation de l'approximation d'une fonction quelconque définie sur une surface par un diagramme de Voronoï restreint. L'approximation est réalisée en assignant à chaque cellule restreinte une valeur d'approximation constante, qui peut soit être définie par une variable à optimiser également, soit comme la moyenne des valeurs de la fonction à approximer dans la cellule de Voronoï restreinte. Cette dernière configuration permet d'améliorer la stabilité de l'optimisation, et d'améliorer les résultats obtenus. Pour réaliser l'optimisation, nous avons mis en place un algorithme de type descente de gradient. Notre approche permet d'approximer des fonctions comportant des discontinuités. Dans ce cas, des paires de sites alignent leur bissecteur le long de la discontinuité pour minimiser l'erreur d'approximation.

4.5.2 Perspectives

Il serait tout d'abord nécessaire de travailler sur le solveur utilisé pour minimiser notre fonction objectif. Sa définition reste en effet pour le moment très heuristique et un certain nombre de paramètres subsistent, en particulier les paramètres τ et σ guidant l'ajustement

automatique du pas d'optimisation des sites. Notre méthode d'optimisation permet en général d'obtenir un minimum local de la fonction objectif. Lorsque l'initialisation des sites est défavorable, il est possible que ce minimum soit médiocre et que l'erreur d'approximation soit mal répartie sur le domaine. Dans ce cas il serait envisageable d'ajouter une procédure prélevant des sites dans les régions où l'erreur d'approximation est faible pour les insérer là où elle est la plus forte. Il serait également possible d'adapter la méthode du recuit simulé pour perturber la solution obtenue et tenter d'obtenir une solution voisine de meilleure qualité.

Une extension de nos travaux serait de remplacer l'approximation constante par cellule par une approximation linéaire ou d'ordre supérieur. L'utilisation d'éléments linéaires par cellule ne supprime pas les discontinuités de l'approximation au niveau du bord des cellules restreintes, mais permettrait d'utiliser moins de sites pour approximer une fonction dont le gradient est localement constant. Dans le cas linéaire, il serait alors nécessaire d'utiliser trois paramètres par cellule restreinte pour y définir l'approximation. Une généralisation directe de notre méthode consisterait donc à définir ces trois paramètres comme variables d'approximation, et de calculer le gradient de $\mathscr{F}$ par rapport à ces trois variables. Les travaux dans le domaine de l'approximation d'intégrales par des points de quadrature pourraient également définir les valeurs optimales de ces paramètres sous forme close. L'utilisation de ces formes closes serait alors similaire à l'utilisation de la valeur moyenne comme valeur d'approximation que nous avons présentée ici.

Chapitre 5

Ajustement de surfaces

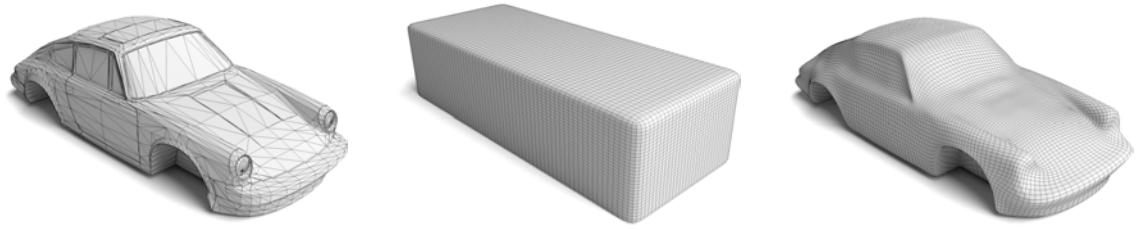
5.1 Motivations

Dans ce chapitre, nous appliquerons les méthodes d'optimisation de diagrammes de Voronoï restreints à l'ajustement de surfaces. Pour ce problème, nous disposons de deux maillages $\mathcal{S}$ et $\mathcal{T}$. $\mathcal{S}$ décrit la surface que nous allons ajuster. Il s'agit donc de la variable de notre problème. $\mathcal{T}$ reste fixe et correspond à la surface que nous chercherons à obtenir en déformant $\mathcal{S}$. Pour réaliser cette déformation, nous agissons sur les sommets de $\mathcal{S}$ en modifiant leurs positions. Le problème consiste ainsi à trouver la meilleure position possible pour les sommets de $\mathcal{S}$ afin que $\mathcal{S}$ soit la plus ressemblante possible à la surface cible $\mathcal{T}$. Il s'agit une fois de plus d'un problème de minimisation. Les variables sont les sommets de $\mathcal{S}$, et nous cherchons à minimiser l'écart entre les surfaces $\mathcal{S}$ et $\mathcal{T}$ (Figure 5.1).

Il existe déjà un certain nombre de méthodes pour ajuster deux surfaces. Ces méthodes diffèrent en général sur la façon dont l'écart entre les deux surfaces $\mathcal{S}$ et $\mathcal{T}$ est mesuré. Notre contribution majeure ici consiste à remarquer que la fonction objectif utilisée pour obtenir un diagramme de Voronoï barycentrique (Chapitre 4) peut également être utilisée pour définir cet écart. Étant donné un ensemble de sites $\mathbf{V} = \{\mathbf{v}_k\}_k$ et un maillage $\mathcal{T}$, nous obtenions un diagramme de Voronoï barycentrique en minimisant la fonction :

$$\mathscr{F}_{|\mathcal{T}}(\mathbf{V}) = \sum_{\mathbf{v}_k \in \mathbf{V}} \int_{\Omega_{k|\mathcal{T}}} \|\mathbf{y} - \mathbf{v}_k\|^2 \mathrm{d}\mathbf{y}$$

où $\Omega_{k|\mathcal{T}}$ est la cellule de Voronoï de $\mathbf{v}_i$ restreinte à $\mathcal{T}$, autrement dit l'ensemble des points de $\mathcal{T}$ pour lesquels le site $\mathbf{v}_k$ est le plus proche parmi l'ensemble des sites $\mathbf{V}$. Cette fonction objectif calcule la somme pour chaque point $\mathbf{y}$ de $\mathcal{T}$ de la distance entre $\mathbf{y}$ et le site le plus



(a) Maillage cible $\mathcal{T}$ (b) Initialisation de $\mathcal{S}$ (c) $\mathcal{S}$ ajustée à $\mathcal{T}$

FIGURE 5.1 – Ajustement d'un maillage $\mathcal{S}$ à un autre maillage $\mathcal{T}$.

proche de lui au sein de $\mathbf{V}$. Il est alors possible de la voir comme une forme d'écart entre la surface $\mathcal{T}$ et l'ensemble des sites $\mathbf{V}$. En considérant que les sites sont en réalité répartis sur la surface $\mathcal{S}$, nous montrerons alors que cette fonction peut être utilisée pour mesurer l'écart entre $\mathcal{S}$ et $\mathcal{T}$ à minimiser.

5.2 Contributions

Notre contribution majeure consiste ici à établir un parallèle entre la fonction objectif minimisée pour l'obtention d'un diagramme de Voronoï barycentrique, et la distance au carré entre surfaces, minimisée dans de nombreuses méthodes de la littérature pour l'ajustement de surfaces. Plus spécifiquement, nous proposons :

la définition d'une nouvelle fonction objectif pour l'ajustement de surfaces. Cette fonction objectif peut être minimisée proprement et rapidement comme détaillé en Chapitre 4. À l'aide de cette fonction objectif, nous ajusterons alors un maillage $\mathcal{S}$ à un autre maillage $\mathcal{T}$. Nous étendrons également ce procédé à l'ajustement de surfaces de subdivision,

une formulation symétrique de notre fonction objectif, permettant d'assurer que tous les points des surfaces $\mathcal{S}$ et $\mathcal{T}$ sont pris en compte. Cette formulation est réalisée en ajoutant un terme à la fonction objectif précédente. Ce terme consiste simplement à considérer un échantillonnage de $\mathcal{T}$ et à calculer cette fois l'écart entre cet échantillonnage et $\mathcal{S}$. Par rapport au terme précédent, les variables et les paramètres sont alors inversés. Nous fournirons le gradient de ce nouveau terme, afin de pouvoir minimiser la nouvelle fonction objectif obtenue,

une étude théorique démontrant que cette fonction objectif peut en effet être considérée comme une approximation de la distance au carré entre surfaces. Nous montrerons ainsi que quand les échantillonnages de $\mathcal{S}$ et $\mathcal{T}$ deviennent de plus en plus denses, notre fonction objectif converge vers cette distance.

5.3 Fondements et état de l'art

5.3.1 Notations

Dans ce chapitre, nous aurons l'occasion de manipuler deux surfaces : la surface $\mathcal{S}$ à ajuster et la surface $\mathcal{T}$ correspondant à l'objectif. Nous récapitulons ici les différentes notations associées à ces surfaces.

Surface ajustée

$\mathcal{S}$	surface ajustée, décrite par un maillage
$\mathbf{X}$	ensemble des sommets de $\mathcal{S}$
$\mathbf{x}_i$	sommet de $\mathcal{S}$
$\tilde{\mathbf{X}}$	ensemble des échantillons de $\mathcal{S}$
$\tilde{\mathbf{x}}_i$	échantillon de $\mathcal{S}$
$\mathbf{x}$	point quelconque de $\mathcal{S}$

Surface cible

$\mathcal{T}$	surface cible, décrite par un maillage
$\mathbf{Y}$	ensemble des sommets de $\mathcal{T}$
$\mathbf{y}_j$	sommet de $\mathcal{T}$
$\tilde{\mathbf{Y}}$	ensemble des échantillons de $\mathcal{T}$
$\tilde{\mathbf{y}}_j$	échantillon de $\mathcal{T}$
$\mathbf{y}$	point quelconque de $\mathcal{T}$

Nous réutiliserons également certaines des notations déjà utilisées précédemment pour décrire les diagrammes de Voronoï restreints.

Diagrammes de Voronoï restreints

Ω_i	la cellule de Voronoï tridimensionnelle de l'échantillon $\tilde{\mathbf{x}}_i$
$\Omega_{i\|\mathcal{T}}$	la cellule de Voronoï de $\tilde{\mathbf{x}}_i$ restreinte à $\mathcal{T}$
$\mathbf{c}_k$	un sommet d'un diagramme de Voronoï restreint

5.3.2 État de l'art pour l'ajustement de surfaces

Le problème de l'ajustement de surface a été étudié principalement pour deux applications. La première est le remaillage, qui consiste étant donné un maillage d'un objet à en chercher un autre, ayant des caractéristiques désirables pour une application donnée. Nous avons par exemple déjà évoqué en Section 1.5.1 le cas des éléments finis. Pour le remaillage, la surface $\mathcal{S}$ à ajuster n'est alors pas une donnée du problème, mais doit également être construite. La seconde application est la paramétrisation d'une surface sur une autre. Cette fois ci, les deux surfaces sont données, et le problème consiste à trouver pour chaque point de $\mathcal{S}$ un point lui correspondant sur $\mathcal{T}$, et inversement, de façon bijective et continue – il s'agit donc d'un homéomorphisme, défini en Section 1.3.4. Pour ces deux applications, l'ajustement de surfaces est l'une des approches possibles.

Nous distinguerons trois problèmes au sein de l'ajustement de surfaces, qui sont en général traités différemment selon l'application :

L'initialisation de $\mathcal{S}$ est une étape essentielle pour le remaillage, qui ne suppose pas que la surface $\mathcal{S}$ est donnée. Dans ce cas, cette surface est générée à partir de du maillage $\mathcal{T}$ à remailler, et pourra évoluer au fur et à mesure de l'optimisation ;

La mise en correspondance de $\mathcal{S}$ et $\mathcal{T}$, qui permet d'évaluer l'écart entre $\mathcal{S}$ et $\mathcal{T}$ en associant les points de $\mathcal{S}$ et $\mathcal{T}$ qui devront être proches dans le résultat final. Pour le remaillage, cette phase s'appuie en général sur la construction de $\mathcal{S}$. Dans le cas de la paramétrisation en revanche, il s'agit du cœur du problème, et les correspondances entre $\mathcal{S}$ et $\mathcal{T}$ pourront être amenées à varier au cours de l'optimisation ;

L'optimisation de $\mathcal{S}$, pour l'ajuster à $\mathcal{T}$. Dans le cas où $\mathcal{S}$ est un maillage, il s'agit donc de calculer une nouvelle position pour chacun des sommets de $\mathcal{S}$ pour minimiser l'écart entre $\mathcal{S}$ et $\mathcal{T}$. Dans le cas d'une surface spline – carreaux de Bézier ou surface de subdivision –, ce sont les positions des points de contrôle qui sont optimisées.

L'approche choisie pour aborder ces problèmes dépend fortement des hypothèses choisies. Dans notre cas, nous utiliserons l'hypothèse assez forte que la surface $\mathcal{S}$ à ajuster nous est fournie. Vis à vis de la surface $\mathcal{T}$ correspondant à l'objectif à atteindre, nos hypothèses sont en revanche très faibles, car nous utiliserons uniquement le fait qu'elle est définie par un ensemble de triangles. Les relations d'adjacentes entre ces triangles, lorsqu'elles sont

fournies, accélèrent légèrement la méthode. Elles ne sont toutefois pas nécessaire, et ne changent en rien le résultat obtenu.

Initialisation de $\mathcal{S}$

Une première approche pour l'initialisation de $\mathcal{S}$ consiste à utiliser un maillage canonique simple. Floater [39] se restreint ainsi aux surfaces topologiquement équivalentes à un disque en paramétrant $\mathcal{T}$ sur le carré unité afin d'ajuster un carreau de spline. Kobbelt *et al.* [55] s'intéressent aux surfaces de genre zéro, qu'ils commencent par paramétrer sur la sphère unité. L'image par la paramétrisation d'un icosaèdre est ensuite utilisée comme initialisation. De nouveaux sommets sont ensuite ajoutés sur ce maillage de base en scindant des arêtes, pour assurer un échantillonnage régulier de $\mathcal{T}$. L'utilisation d'une paramétrisation pour ces deux méthodes les rend particulièrement sensibles à la qualité du maillage de $\mathcal{T}$. En particulier, en cas de problème topologique – trous, ou anses indésirables provenant de l'acquisition de ${}_T s$ – il ne sera pas possible d'appliquer ces résultats. Delingette [22] proposent à l'utilisateur d'initialiser leur méthode en utilisant un plan, une sphère, ou un cylindre, mais $\mathcal{T}$ peut être de genre quelconque. Ils alternent ensuite une phase d'optimisation, et une phase d'édition automatique de $\mathcal{S}$, qui permet d'augmenter localement la résolution, ou de percer des trous. Cette phase d'édition permet a priori de traiter n'importe quelle genre pour $\mathcal{T}$.

Dans le cas où $\mathcal{T}$ est fournie sous la forme d'un maillage de forte résolution, il est possible d'utiliser une méthode de *décimation* pour obtenir un maillage similaire, mais ayant un nombre de sommets bien plus faibles. Une méthode classique consiste à contracter une à une les arêtes de $\mathcal{T}$ [100, 43]. Ma *et al.* [70] utilisent ensuite ce maillage simplifié en tant que maillage de contrôle d'une surface de subdivision, qu'il ajustent ensuite à $\mathcal{T}$. Pour obtenir un maillage initial quadrangulaire, Eck et Hoppe [33] utilisent la décimation, puis fusionnent des paires de triangles adjacents pour former des faces quadrangulaires. La fusion de paires de triangles voisins peut être vue comme un problème de couplage optimal dans un graphe. Remacle *et al.* [96] utilisent ainsi l'algorithme « Blossom » pour déterminer un ensemble de couples de triangles à fusionner. La décimation de maillages quadrangulaire est plus compliquée à réaliser, car il faut s'assurer que le maillage reste quadrangulaire. Certaines méthodes commencent toutefois à voir le jour, ainsi Panozzo *et al.* [79] étendent les travaux de Tarini *et al.* [108] sur la décimation quadrangulaire, dans le but précis de l'ajustement d'une surface de subdivision. Cette fois, le maillage cible $\mathcal{T}$ est directement converti en un maillage quadrangulaire, qui est ensuite décimé pour finalement aboutir à une surface $\mathcal{S}$ initiale. Les méthodes utilisant la décimation ont l'énorme avantage de pouvoir s'appliquer à des surfaces de genre quelconque. Le maillage cible doit cependant de nouveau être de qualité raisonnable, car par exemple tous les trous indésirables seront préservés dans le maillage produit.

Il est également envisageable d'utiliser les méthodes présentées en Chapitre 1 pour effectuer un remaillage quadrangulaire grossier, utilisé ensuite comme maillage de contrôle. Bommes *et al.* [9] proposent un état de l'art plus détaillé sur ces approches. En particulier, les travaux de Campen *et al.* [12] se concentrent sur la génération d'un maillage quadrangulaire grossier à partir d'un champ de directions défini sur la surface cible $\mathcal{T}$. Cette construction est réalisée en déterminant l'ensemble des singularités du champ de directions, puis en cherchant un ensemble de boucles sur $\mathcal{T}$ pour séparer les paires de singularités, c'est à dire qu'un chemin reliant deux singularités devra nécessairement croiser une de ces boucles. Le résultat est une segmentation de $\mathcal{T}$ en régions délimitées par ces boucles. Un maillage

quadrangulaire est enfin extrait en prenant le dual de cette segmentation. Pour cette méthode, les boucles utilisées sont définies par des ensembles d'arêtes de $\mathcal{T}$, et la finesse et la régularité de la triangulation de $\mathcal{T}$ peuvent donc fortement influencer le résultat.

Les méthodes mentionnées précédemment sont en grande partie dépendantes de la qualité de $\mathcal{T}$. Il sera difficile de les appliquer lorsque la triangulation sera très irrégulière, qu'elle contiendra des triangles très allongés, voire que la connectivité entre les triangles sera erronée ou absente. Dans cette situation, il n'existe pas à notre connaissance de méthode complètement automatique pour initialiser $\mathcal{S}$. Yao *et al.* [119] proposent cependant une méthode assistée où l'utilisateur fournit un squelette de $\mathcal{T}$, à partir duquel le maillage de base est construit en assemblant des tubes à section carrée pour former un maillage quadrangulaire.

Mise en correspondance de $\mathcal{S}$ et $\mathcal{T}$

Lorsque $\mathcal{S}$ est initialisée via une paramétrisation, la phase de mise en correspondance est triviale, étant donné que la paramétrisation fournit les correspondance entre $\mathcal{S}$ et $\mathcal{T}$. Lorsqu'aucune paramétrisation n'est disponible, la méthode la plus classique consiste à projeter les sommets de $\mathcal{S}$ sur $\mathcal{T}$. Eck et Hoppe [33] et Ma *et al.* [70] construisent leur maillage $\mathcal{S}$ initial par une méthode de décimation. Dans ce cas, $\mathcal{S}$ et $\mathcal{T}$ sont déjà proches, et la projection utilisée consiste simplement à associer à chaque point de $\mathcal{S}$ le point le plus proche de $\mathcal{T}$. Dans des situations où $\mathcal{S}$ est plus éloignée de $\mathcal{T}$, cette projection ignore les zones concaves de $\mathcal{T}$ pour lesquelles aucun point n'est le plus proche voisin d'un point de $\mathcal{S}$. Ainsi Kobbelt *et al.* [55] et Tarini *et al.* [107], pour qui $\mathcal{S}$ n'est pas nécessairement très proche de $\mathcal{T}$, projettent dans la direction de la normale de $\mathcal{S}$. À partir d'un point $\mathbf{x}$ de $\mathcal{S}$ un rayon est tiré dans la direction de la normale de $\mathcal{S}$ en $\mathbf{x}$, et le premier point rencontré sur $\mathcal{T}$ est associé à $\mathbf{x}$. Un traitement spécial devient par contre nécessaire lorsque le rayon tiré depuis $\mathbf{x}$ ne coupe pas $\mathcal{T}$.

Schreiner *et al.* [99] proposent également une formulation symétrique, consistant à paramétrer $\mathcal{S}$ et $\mathcal{T}$ sur un domaine commun. Dans cette approche, les deux surfaces $\mathcal{S}$ et $\mathcal{T}$ sont décimées jusqu'à obtenir deux maillages grossiers de connectivité identique. En suivant le procédé inverse de la décimations, les sommets des deux surfaces à la fois sont ensuite réinsérés sur ce domaine de base. Une fois cette réinsertion réalisée, tous les sommets de $\mathcal{S}$ et $\mathcal{T}$ sont paramétrés sur le domaine commun, et en composant les paramétrisations, $\mathcal{S}$ est paramétrée sur $\mathcal{T}$ et inversement. Dans cette méthode, la méthode pour déterminer le maillage grossier qui servira de cible commune aux décimations reste manuelle. Wang *et al.* [114] utilisent un domaine commun dépendant du genre des deux surfaces. Pour les genre 0 et 1, la paramétrisation est réalisée respectivement sur une sphère et sur un plan. Pour les surfaces de genre supérieur, la paramétrisation est réalisée sur le disque hyperbolique de Poincaré. Une fois les deux surfaces paramétrées, une transformation est réalisée dans le domaine paramétrique pour faire coïncider les domaines des deux paramétrisations. Comme précédemment, en composant toutes ces fonctions dans le bon sens, $\mathcal{S}$ est paramétrée sur $\mathcal{T}$. Ici, la transformation calculée pour mettre en correspondance les deux domaines paramétriques requiert également une intervention de l'utilisateur.

Ajustement de $\mathcal{S}$ à $\mathcal{T}$

Lorsque les correspondances calculées entre $\mathcal{S}$ et $\mathcal{T}$ sont de qualité suffisante, et qu'il n'est pas nécessaire de les faire évoluer au cours de l'ajustement, plusieurs méthodes permettent

directement d'obtenir la position de $\mathcal{S}$. Floater [39], après avoir paramétré $\mathcal{T}$ sur le carré unité échantillonnent le domaine paramétrique par une grille, et calculent l'unique carreau de spline passant par tous les points de $\mathcal{T}$ correspondant aux échantillons. Ma *et al.* [70] et Eck et Hoppe [33], une fois les correspondances déterminées, expriment la position optimale de $\mathcal{S}$ comme la solution d'un système de moindres carrés. Le système mis en place revient simplement à demander à ce que chaque échantillon pris sur $\mathcal{S}$ soit le plus proche possible du point de $\mathcal{T}$ auquel il a été associé. De même, la solution d'un tel système existe, et est unique. Une fois l'ajustement réalisé, il est possible d'améliorer le résultat de ces méthodes en itérant le procédé. Chaque itération commence par déterminer un ensemble de correspondances entre $\mathcal{S}$ et $\mathcal{T}$, puis optimise $\mathcal{S}$ pour minimiser les distances entre les points correspondants. Il s'agit alors de la méthode *ICP* pour « Iterative Closest Point ». Différentes variantes autour de ce principe sont étudiées par Rusinkiewicz et Levoy [98]. Tarini *et al.* [107] utilisent une variante de cette méthode pour améliorer la régularité de l'ajustement obtenu. Leurs itérations consistent donc à alterner une étape de lissage de $\mathcal{S}$ et une étape de projection sur $\mathcal{T}$.

Delingette [22] définissent deux forces agissant sur $\mathcal{S}$. La force interne ne dépend pas de $\mathcal{T}$, et vise simplement à lisser $\mathcal{S}$. La force externe attire chaque point de $\mathcal{S}$ vers le point de $\mathcal{T}$ lui correspondant. L'optimisation est ensuite réalisée par une méthode explicite, consistant à déplacer les sommets de $\mathcal{S}$ petit à petit sous l'influence de ces deux forces. Une approche plus haut niveau consiste plutôt à définir le problème de l'ajustement de surface comme le problème de la minimisation d'une fonction objectif. Cette fonction objectif quantifie l'écart séparant $\mathcal{S}$ et $\mathcal{T}$. Eckstein *et al.* [34] définissent la fonction objectif comme étant la distance de Hausdorff. Cette distance étant néanmoins compliquée à minimiser du fait que son gradient n'est pas continu, une approximation continue est proposée. La minimisation est ensuite réalisée comme précédemment par une méthode explicite. Pour améliorer le résultat obtenu, et surtout éviter d'aboutir à un mauvais minimum local de la fonction objectif, des hypothèses sont ajoutées sur les déplacement autorisés de $\mathcal{S}$.

5.3.3 Distance au carré entre surfaces

Dans nos travaux, nous nous intéressons au problème de l'ajustement d'une surface donnée. Nous ne considérons donc pas l'initialisation de cette surface. L'approche que nous avons choisie est celle de la minimisation de la distance au carré entre deux surfaces. Entre une surface $\mathcal{S}$ et une surface $\mathcal{T}$, cette distance est donnée par :

$$\mathscr{F}_{\mathcal{S}\to\mathcal{T}} = \int_{\mathcal{S}} \|\mathbf{x} - \Pi_{\mathcal{T}}(\mathbf{x})\|^2 \mathrm{d}\mathbf{x}, \tag{5.3.1}$$

où $\Pi_{\mathcal{T}}(\mathbf{x})$ est le point de $\mathcal{T}$ le plus proche de $\mathbf{x}$. Par rapport à la définition formelle d'une distance, cette équation n'est tout d'abord symétrique, car $\mathscr{F}_{\mathcal{S}\to\mathcal{T}} \neq \mathscr{F}_{\mathcal{T}\to\mathcal{S}}$. Ce problème peut être aisément corrigé en utilisant la somme des deux distances orientées $\mathscr{F}_{\mathcal{S}\to\mathcal{T}}+\mathscr{F}_{\mathcal{T}\to\mathcal{S}}$. Cette nouvelle distance est donc symétrique et ne s'annule que lorsque les deux surfaces $\mathcal{S}$ et $\mathcal{T}$ coïncident. La dénomination « distance » reste cependant un abus de langage, car la distance au carré entre surfaces ne respecte pas l'inégalité triangulaire, et n'est donc pas à proprement parler une distance. De vraies distances entre surfaces existent, et Veltkamp [113] en proposent un inventaire.

Cette distance a tout d'abord été introduite pour l'ajustement de surfaces par Pottmann et Leopoldseder [87], qui proposent de minimiser la fonction objectif :

$$\mathscr{G}(\mathbf{X}) = \mathscr{F}_{\mathcal{S}\to\mathcal{T}}(\mathbf{X}) + \lambda\mathscr{R}(\mathbf{X}) \qquad (5.3.2)$$
$$\text{où} \quad \mathscr{R}(\mathbf{X}) = \|\mathbf{L}\mathbf{X}\|^2.$$

Dans cette équation, $\mathbf{X} = \{\mathbf{x}_i\}_i$ est l'ensemble des sommets de $\mathcal{S}$, et correspond aux variables du problème. Le terme $\mathscr{R}(\mathbf{X})$ a pour but de préserver la qualité de la surface $\mathcal{S}$ au cours de l'ajustement. $\mathbf{L}$ est une matrice qui dans le cas le plus simple correspond à la matrice de coefficients :

$$l_{i,j} = \begin{cases} 0 & \text{si } \mathbf{x}_i \text{ et } \mathbf{x}_j \text{ ne sont pas voisins ;} \\ \frac{1}{k} & \text{où } k \text{ est la valence de } \mathbf{x}_i \text{ sinon.} \end{cases} \qquad (5.3.3)$$

Minimiser $\|\mathbf{L}\mathbf{X}\|^2$ correspond avec ces coefficients à minimiser au sens des moindres carrés l'écart entre les sommets de $\mathcal{S}$ et le barycentre de leurs voisins. De façon plus générale, $\mathbf{L}\mathbf{X}$ peut être considéré comme une approximation du Laplacien de $\mathcal{S}$ à chacun de ses sommets. D'autres formes pour $\mathbf{L}$ sont donc envisageables, en particulier en utilisant les poids contangents évoqués en Section 1.4.2. Le facteur de régularisation λ permet à l'utilisateur de choisir un compromis entre la régularité de $\mathcal{S}$ et l'ajustement aux données. Cette distance a été utilisée dans divers travaux pour l'ajustement de surfaces de surfaces B-spline [87], de surfaces de subdivision [17] ou de courbes B-spline [115].

Deux éléments sont difficiles à calculer dans le cas général dans cette équation : obtenir la forme close de l'intégrale sur $\mathcal{S}$, et déterminer de manière exacte $\Pi_{\mathcal{T}}(\mathbf{x})$ pour tout point $\mathbf{x}$. Pour ces raisons, ces méthodes utilisent deux échantillonnages :

un échantillonnage $\tilde{\mathbf{X}} = \{\tilde{\mathbf{x}}_i\}_i$ de $\mathcal{S}$ permet d'approximer l'intégrale sur $\mathcal{S}$ par la somme des distances au carrés pour chaque échantillon :

$$\int_{\mathcal{S}} \|\mathbf{x} - \Pi_{\mathcal{T}}(\mathbf{x})\|^2 \mathrm{d}\mathbf{x} \approx \sum_{\tilde{\mathbf{x}}_i \in \tilde{\mathbf{X}}} \|\tilde{\mathbf{x}}_i - \Pi_{\mathcal{T}}(\tilde{\mathbf{x}}_i)\|^2;$$

un échantillonnage $\tilde{\mathbf{Y}} = \{\tilde{\mathbf{y}}_i\}_i$ de $\mathcal{T}$ permet d'approximer le point le plus proche sur $\mathcal{T}$ par l'échantillon le plus proche dans $\tilde{\mathbf{Y}}$:

$$\Pi_{\mathcal{T}}(\mathbf{x}) \approx \Pi_{\tilde{\mathbf{Y}}}(\mathbf{x}).$$

Avec ces simplifications la fonction objectif de l'Équation 5.3.2 est alors approximée par :

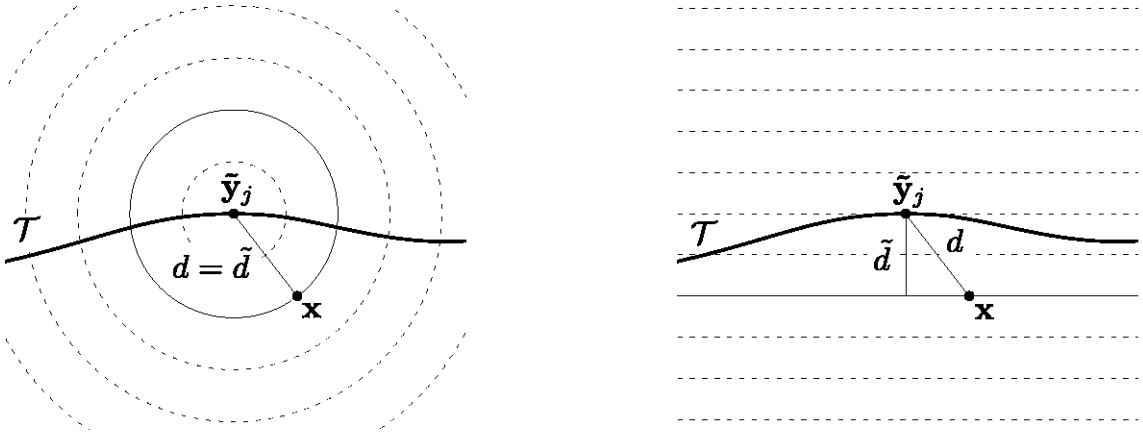
$$\tilde{\mathscr{G}}(\mathbf{X}) = \sum_{\tilde{\mathbf{x}}_i \in \tilde{\mathbf{X}}} \|\tilde{\mathbf{x}}_i - \Pi_{\tilde{\mathbf{Y}}}(\tilde{\mathbf{x}}_i)\|^2_{SD} + \lambda\mathscr{R}(\mathbf{X}), \qquad (5.3.4)$$

où $\|.\|_{SD}$ est un développement limité de la distance au carré à $\mathcal{T}$ au voisinage de $\mathbf{x}$. Ce développement limité permet de limiter l'erreur due aux échantillonnages. Dans la littérature, les développement utilisés sont réalisés :

à l'ordre 0 :

$$\|\mathbf{x} - \tilde{\mathbf{y}}_j\|_{SD_0} = \|\mathbf{x} - \tilde{\mathbf{y}}_j\|^2$$

la distance au carré au point le plus proche est directement utilisée. Les isovaleurs de cette approximation sont montrées sur la Figure 5.2. La fonction objectif obtenue en utilisant cette distance dans l'Équation 5.3.4 peut être minimisée par un processus itératif alternant deux étapes. La première considère les $\Pi_{\tilde{\mathbf{Y}}}(\mathbf{x})$ comme des constantes,



(a) ordre 0 (b) ordre 1

FIGURE 5.2 – Isovaleurs de l'approximation à l'ordre 0 (a) et à l'ordre 1 (b) de la distance au carré à $\mathcal{T}$. d correspond à la distance euclidienne entre $\mathbf{x}$ et $\tilde{\mathbf{y}}_j$, et $\tilde{d}$ à la distance dont le carré est utilisé pour l'approximation de la distance au carré à $\mathcal{T}$ en $\mathbf{x}$.

et la minimisation se ramène à un problème de moindres carrée. La seconde étape recalcule les projetés $\Pi_{\tilde{\mathbf{Y}}_j}$. Wang *et al.* [115] montrent que ce procédé converge linéairement ;

à l'ordre 1 :

$$\|\mathbf{x} - \tilde{\mathbf{y}}_j\|_{SD_1} = \|(\mathbf{x} - \tilde{\mathbf{y}}_j).\mathbf{n}_j\|^2$$

où $\mathbf{n}_j$ est la normale à $\mathcal{T}$ au point $\tilde{\mathbf{y}}_j$. Cette distance revient à calculer la distance entre $\mathbf{x}$ et le plan tangent à $\mathcal{T}$ en $\tilde{\mathbf{y}}_j$. Ses isovaleurs sont illustrées sur la Figure 5.2. Wang *et al.* [115] montrent que la minimisation de l'erreur en utilisant cette distance est équivalente à une méthode de Gauss Newton. Cette méthode a une convergence quadratique lorsque la fonction objectif tend vers 0. Cependant si cette condition n'est pas vérifiée, la convergence n'est plus forcément garantie :

à l'ordre 2 : cette approximation a été définie par Pottmann et Hofer [86].

$$\|\mathbf{x} - \tilde{\mathbf{y}}_j\|_{SD_2} = \alpha_1\|(\mathbf{x} - \tilde{\mathbf{y}}_j).\mathbf{e}_1\|^2 + \alpha_2\|(\mathbf{x} - \tilde{\mathbf{y}}_j).\mathbf{e}_2\|^2 + \|(\mathbf{x} - \tilde{\mathbf{y}}_j).\mathbf{n}_j\|^2$$

où :

- $\mathbf{e}_1$ et $\mathbf{e}_2$ est la base du plan tangent à $\mathcal{T}$ en $\tilde{\mathbf{y}}_j$ formée par les directions de courbures principales, respectivement associées aux rayons de courbure κ_1 et κ_2 ;
- α_1 et α_2 sont des coefficients calculés à partir des rayons de courbure κ_1 et κ_2. En posant d comme étant la distance signée telle que $|d| = \|\mathbf{x} - \tilde{\mathbf{y}}_j\|$, nous avons :

$$\alpha_k = \frac{d}{d - \kappa_k}.$$

Lorsque $\mathbf{x}$ s'éloigne de $\mathcal{T}$, cette approximation devient équivalente à celle à l'ordre 0. Lorsqu'au contraire, $\mathbf{x}$ tend vers $\tilde{\mathbf{y}}_j$, nous retrouvons l'approximation à l'ordre 1. Les isovaleurs de cette distance sont illustrées sur la Figure 5.3. Wang *et al.* [115] montrent que la minimisation du terme d'erreur avec cette distance correspond à une méthode de Newton, et a donc une convergence quadratique.

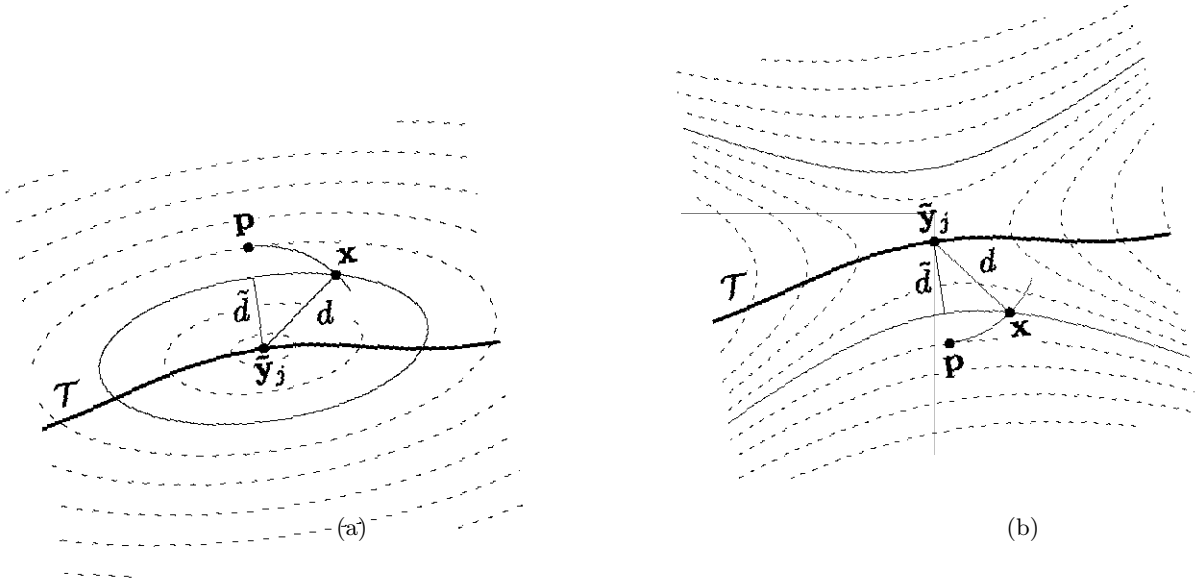


FIGURE 5.3 – En pointillés, les isovaleurs de l'approximation au second ordre de la distance au carré à $\mathcal{T}$ en $\tilde{\mathbf{y}}_j$. Le développement limité au second ordre est pris au point $\mathbf{p}$, à la même distance d de $\tilde{\mathbf{y}}_j$ que $\tilde{\mathbf{x}}_i$ dans la direction de la normale en $\tilde{\mathbf{y}}_j$. C'est au point $\mathbf{p}$ que l'approximation a un contact d'ordre 2 avec la distance au carré à $\mathcal{T}$. Deux cas sont possibles selon que $\mathbf{x}$ se trouve du côté opposé au cercle osculateur en $\tilde{\mathbf{y}}_j$ (a) ou du côté opposé (b). L'isovaleur correspondant au point $\tilde{\mathbf{x}}_i$ est celle qui n'est pas en pointillés. $\tilde{d}$ est la distance mise au carré pour approximer la distance entre $\tilde{\mathbf{x}}_i$ et $\mathcal{T}$. Dans le cas (b) où $\tilde{\mathbf{x}}_i$ est du même côté de $\mathcal{T}$ que le cercle osculateur en $\tilde{\mathbf{y}}_j$, l'approximation de la distance au carré n'est plus une fonction positive. Pottmann et Hofer [86] proposent une modification de l'approximation pour corriger ce problème.

Nos travaux améliorent ces méthodes sur deux points. Tout d'abord, nous utilisons la formulation symétrique $\mathscr{F}_{\mathcal{S}\to\mathcal{T}} + \mathscr{F}_{\mathcal{T}\to\mathcal{S}}$ de la distance au carré, qui n'est annulée que lorsque les deux surfaces coïncident. De plus, nous montrons qu'il est possible de se passer de l'un des deux échantillonnages. En effet, en utilisant simplement $\Pi_{\mathcal{T}}(\mathbf{x}) \approx \Pi_{\tilde{\mathbf{Y}}}(\mathbf{x})$, lorsque $\mathcal{S}$ est une surface polygonale, il est alors possible de calculer sous forme close l'intégrale

$$\int_{\mathcal{S}} \|\mathbf{x} - \Pi_{\tilde{\mathbf{Y}}}(\mathbf{x})\|^2 \mathrm{d}\mathbf{x}.$$

Notre fonction objectif, définie sur la base de cette formulation peut être minimisée en utilisant une méthode quasi-Newton, ayant une convergence rapide, sans pour autant devoir utiliser de développement limité à l'ordre 2 sur la distance au carré. Il ne nous est donc pas nécessaire de calculer la courbure de $\mathcal{T}$, ce qui rend notre approche plus robuste lorsque $\mathcal{T}$ possède des triangles dégénérés, voire lorsque la connectivité entre les triangles n'est pas disponible.

5.4 Contribution : distance au carré de Voronoï

5.4.1 Définition

Nous proposons de minimiser une fonction objectif fondée sur la formulation symétrique de la distance au carré de Voronoï :

$$\mathscr{F}(\mathbf{X}) = \mathscr{F}_{\mathcal{S}\to\mathcal{T}}(\mathbf{X}) + \mathscr{F}_{\mathcal{T}\to\mathcal{S}}(\mathbf{X}) + \lambda\mathscr{R}(\mathbf{X}). \tag{5.4.1}$$

Nous approximerons $\mathscr{F}_{\mathcal{S}\to\mathcal{T}}$ en utilisant un échantillonnage de $\tilde{\mathbf{Y}}$ de $\mathcal{T}$, et $\mathscr{F}_{\mathcal{T}\to\mathcal{S}}$ en utilisant un échantillonnage $\tilde{\mathbf{X}}$ de $\mathcal{S}$. Ces deux termes étant symétriques, nous nous focaliserons sur le terme $\mathscr{F}_{\mathcal{T}\to\mathcal{S}}$. En utilisant l'échantillonnage $\tilde{\mathbf{X}}$ de $\mathcal{S}$, nous réalisons l'approximation

$$\|\mathbf{y} - \Pi_{\mathcal{S}}(\mathbf{y})\| \approx \min_{\tilde{\mathbf{x}}_i \in \tilde{\mathbf{X}}} \|\mathbf{y} - \tilde{\mathbf{x}}_i\|. \tag{5.4.2}$$

Nous obtenons alors une approximation de $\mathscr{F}_{\mathcal{T}\to\mathcal{S}}$:

$$\begin{aligned}
\mathscr{F}_{\mathcal{T}\to\mathcal{S}} &= \int_{\mathcal{T}} \|\mathbf{y} - \Pi_{\mathcal{S}}(\mathbf{y})|^2 \mathrm{d}\mathbf{y} \\
&\approx \int_{\mathcal{T}} \min_i \|\mathbf{y} - \tilde{\mathbf{x}}_i\|^2 \mathrm{d}\mathbf{y} \\
\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}} &= \sum_{\tilde{\mathbf{x}}_i \in \tilde{\mathbf{X}}} \int_{\Omega_{i|\mathcal{T}}} \|\mathbf{y} - \tilde{\mathbf{x}}_i\|^2 \mathrm{d}\mathbf{y},
\end{aligned} \tag{5.4.3}$$

où $\Omega_{i|\mathcal{T}}$ est la cellule de Voronoï de $\tilde{\mathbf{x}}_i$ restreinte à $\mathcal{T}$, autrement dit l'ensemble des points de $\mathcal{T}$ pour lesquels l'échantillon $\tilde{\mathbf{x}}_i$ est le plus proche. En effectuant la même approximation pour $\mathscr{F}_{\mathcal{S}\to\mathcal{T}}$ la formulation de la fonction objectif minimisée par notre approche devient :

$$\tilde{\mathscr{F}}(\mathbf{X}) = \tilde{\mathscr{F}}_{\mathcal{S}\to\mathcal{T}}(\mathbf{X}) + \tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}(\mathbf{X}) + \lambda \underbrace{\|\mathbf{L}\mathbf{X}\|^2}_{\mathscr{R}(\mathbf{X})} \tag{5.4.4}$$

$$\text{où} \quad \tilde{\mathscr{F}}_{\mathcal{S}\to\mathcal{T}}(\mathbf{X}) = \sum_{\tilde{\mathbf{y}}_j \in \tilde{\mathbf{Y}}} \int_{\Omega_{j|\mathcal{S}}} \|\mathbf{x} - \tilde{\mathbf{y}}_j\|^2 \mathrm{d}\mathbf{x}.$$

$$\text{et} \quad \tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}(\mathbf{X}) = \sum_{\tilde{\mathbf{x}}_i \in \tilde{\mathbf{X}}} \int_{\Omega_{i|\mathcal{T}}} \|\mathbf{y} - \tilde{\mathbf{x}}_i\|^2 \mathrm{d}\mathbf{y}$$

La matrice $\mathbf{L}$ est définie de la même manière que précédemment, ses coefficients sont fournis par l'Équation 5.3.3. L'influence du facteur de régularisation λ est illustrée sur la Figure 5.4. $\Omega_{i|\mathcal{T}}$ et $\Omega_{j|\mathcal{S}}$ sont respectivement les cellules de Voronoï de $\tilde{\mathbf{x}}_i$ et$\tilde{\mathbf{y}}_j$, restreintes par $\mathcal{T}$ pour $\tilde{\mathbf{x}}_i$ et $\mathcal{S}$ pour $\tilde{\mathbf{y}}_j$. Les deux termes $\tilde{\mathscr{F}}_{\mathcal{S}\to\mathcal{T}}$ et $\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}$ sont illustrés sur la figure Figure 5.5. De manière remarquable, nous retrouvons la fonction objectif utilisée pour obtenir un diagramme de Voronoï barycentrique au Chapitre 4, définie dans l'Équation 4.3.8. Nous expliquerons en Section 5.4.4 comment appliquer ces méthodes à notre cas particulier.

5.4.2 Qualité de l'approximation

Nous étudions ici la qualité de l'approximation réalisée en remplaçant le terme $\mathscr{F}_{\mathcal{T}\to\mathcal{S}}$ par $\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}$, en fonction de la qualité de l'échantillonnage $\tilde{\mathbf{X}}$. Les propriétés qui suivent s'appliquent également à $\mathscr{F}_{\mathcal{S}\to\mathcal{T}}$ et $\tilde{\mathscr{F}}_{\mathcal{S}\to\mathcal{T}}$, il suffit pour cela d'intervertir les surfaces utilisées. Nous commencerons par aborder le cas où $\mathcal{S}$ et $\mathcal{T}$ sont des surfaces lisses, avant d'étendre ces résultats au cas des surfaces polygonales qui nous concerne.

Cas des surfaces lisses

Pour mesurer la qualité d'un échantillonnage $\mathbf{X}$ d'une surface $\mathcal{S}$, Amenta et Bern [1] définissent la notion d'ε-échantillonnage comme suit :

Définition 5.1 (ε-échantillonnage)**.** *Un ensemble de points $\tilde{\mathbf{X}}$ est dit être un ε-échantillonnage d'une surface $\mathcal{S}$ si pour tout point $\mathbf{x}$ de $\mathcal{S}$, il existe un point $\tilde{\mathbf{x}}_i$ de $\tilde{\mathbf{X}}$ tel que :*

$$\|\mathbf{x} - \tilde{\mathbf{x}}_i\| < \varepsilon \mathrm{lfs}\,(\mathbf{x})\,,$$

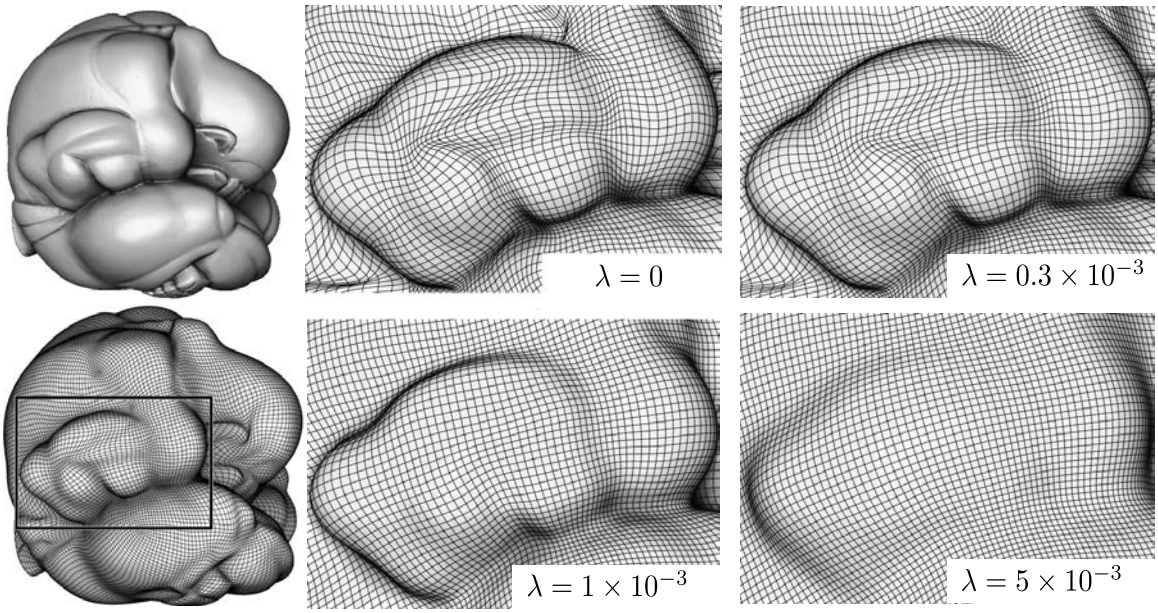
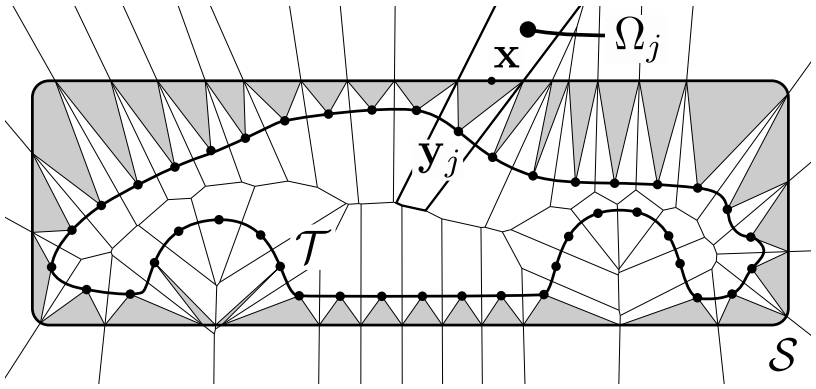
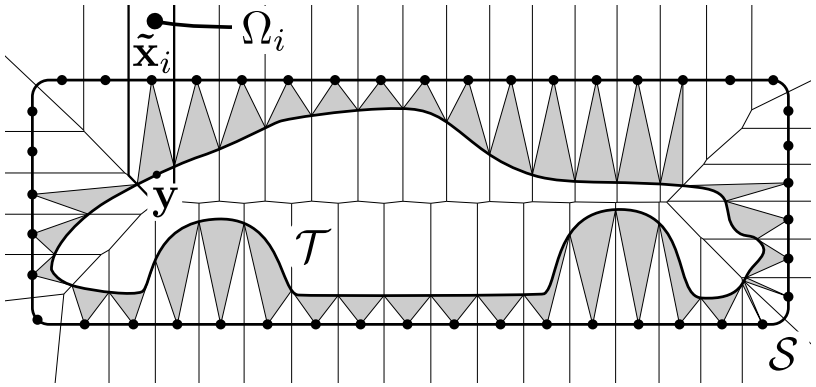


FIGURE 5.4 – Influence du facteur de régularisation λ sur l'ajustement d'une surface de subdivision.



(a) $\tilde{\mathscr{F}}_{\mathcal{S}\to\mathcal{T}}$



(b) $\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}$

FIGURE 5.5 – Distance au carré de Voronoï. Les régions grisées représentent, pour chaque échantillon $\tilde{\mathbf{x}}_i$ (resp. $\tilde{\mathbf{y}}_j$) la portion de $\mathcal{T}$ (resp. $\mathcal{S}$) pour laquelle $\tilde{\mathbf{x}}_i$ sera le plus proche.

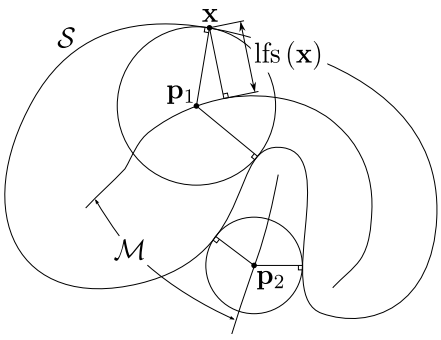


FIGURE 5.6 – L'axe médian $\mathcal{M}$ de $\mathcal{S}$ est l'ensemble des points de l'espace pour lesquels le point le plus proche sur $\mathcal{S}$ n'est pas unique. Ici, $\mathbf{p}_1$ et $\mathbf{p}_2$ sont deux points de l'axe médian, et pour chacun de ces points, $\mathcal{S}$ contient deux points les plus proches. La distance à l'axe médian d'un point $\mathbf{x}$ de $\mathcal{S}$, notée $\mathrm{lfs}\,(()\,\mathbf{x})$ pour « local feature size » est la distance entre $\mathbf{x}$ et le point le plus proche de $\mathcal{M}$.

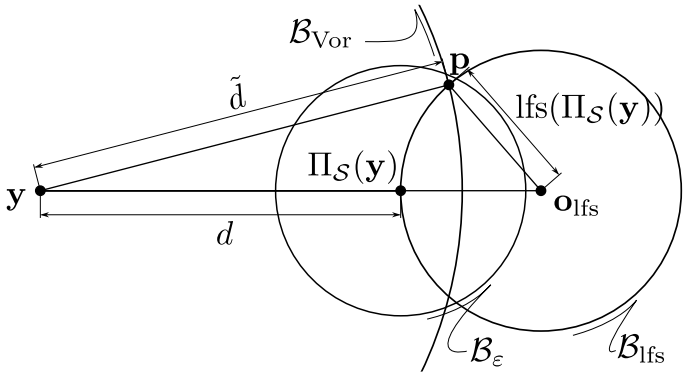


FIGURE 5.7 – Étant donné un point $\mathbf{y}$ sur $\mathcal{T}$, étude de l'erreur commise en approximant d, la distance à $\Pi_{\mathcal{S}}(\mathbf{y})$, son point le plus proche sur $\mathcal{S}$ par $\tilde{d}$ la distance au point le plus proche dans un ε-échantillonnage de $\mathcal{S}$

où $\mathrm{lfs}\,(\mathbf{x})$ *pour « local feature size » correspond à la distance entre le point* $\mathbf{x}$ *et le point le plus proche de l'axe médian de* $\mathcal{S}$.

La distance à l'axe médian est un outil introduit pour la reconstruction de surfaces. Grossièrement, elle permet de mesurer l'épaisseur d'une surface en un point donné. Un exemple est fourni sur la Figure 5.6. Elle permet de s'assurer que l'échantillonnage de la surface est plus fin lorsque la surface est fine ou fortement courbée. Cette définition d'un ε-échantillonnage ne s'applique que pour les surfaces lisses qui ne s'intersectent pas elles-mêmes. Dans ces situations, l'axe médian passerait par la surface, et la distance à l'axe médian s'annule. L'échantillonnage devrait donc contenir une infinité de points.

Nous allons montrer ici que lorsque ε diminue, $\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}$ converge de manière quadratique vers $\mathscr{F}_{\mathcal{T}\to\mathcal{S}}$. Une convergence quadratique signifie que lorsque ε est divisé par deux, l'erreur commise entre $\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}$ et $\mathscr{F}_{\mathcal{T}\to\mathcal{S}}$ est divisée par quatre.

Lemme 5.1 (Qualité de l'approximation en un point de $\mathcal{T}$)**.** *Soit* $\tilde{\mathbf{X}}$ *un* ε*-échantillonnage de* $\mathcal{S}$*. Soit* $\mathbf{y}$ *un point de* $\mathcal{T}$ *et soit* $\tilde{\mathbf{x}}_i$ *son échantillon le plus proche dans* $\tilde{\mathbf{X}}$ *(voir la Figure 5.7). Soit* $d = \|\mathbf{y} - \Pi_{\mathcal{S}}(\mathbf{y})\|$ *et* $\tilde{d} = \|\mathbf{y} - \tilde{\mathbf{x}}_i\|$*. Alors pour* $\varepsilon < 2$ *la borne suivante est optimale :*

$$\tilde{d}^2 - d^2 \leq \varepsilon^2 \mathrm{lfs}\,(\Pi_{\mathcal{S}}(\mathbf{y}))\,(\mathrm{lfs}\,(\Pi_{\mathcal{S}}(\mathcal{T})) - d)$$

Démonstration. Soient :

– $\mathcal{B}_{\mathrm{Vor}}$ la boule ouverte centrée en $\mathbf{y}$ et dont le bord passe par $\tilde{\mathbf{x}}_i$. Cette boule ne contient pas de point de $\tilde{\mathbf{X}}$ car $\tilde{\mathbf{x}}_i$ est l'échantillon le plus proche de $\mathbf{y}$;

- $\mathcal{B}_{\text{lfs}}$ la boule ouverte tangente à $\mathcal{S}$ en $\Pi_{\mathcal{S}}(\mathbf{y})$ du côté opposé à $\mathbf{y}$, et de rayon $\text{lfs}\,(\Pi_{\mathcal{S}}(\mathbf{y}))$. Soit $\mathbf{o}_{\text{lfs}}$ son centre. Comme expliqué par Amenta et Bern [1], cette boule ne contient pas de point de $\mathcal{S}$. Elle ne contient donc en particulier pas d'échantillon de $\tilde{\mathbf{X}}$;
- $\mathcal{B}_{\varepsilon}$ la boule ouverte centrée en $\Pi_{\mathcal{S}}(\mathbf{y})$ de rayon $\varepsilon\text{lfs}\,(\Pi_{\mathcal{S}}(\mathbf{y}))$. Cette boule doit donc contenir un échantillon de $\tilde{\mathbf{X}}$, car $\tilde{\mathbf{X}}$ est un ε-échantillonnage.

Nous noterons $\partial\mathcal{B}$ le bord d'une boule $\mathcal{B}$ et $\overline{\mathcal{B}}$ la clôture de $\mathcal{B}$, c'est à dire $\overline{\mathcal{B}} = \mathcal{B} \cup \partial\mathcal{B}$. Il découle de ces définitions que :

$$\mathcal{B}_{\varepsilon} \not\subset (\mathcal{B}_{\text{lfs}} \cup \mathcal{B}_{\text{Vor}}). \tag{5.4.5}$$

C'est sur cette propriété que la preuve s'appuie. Comme $\Pi_{\mathcal{S}}(\mathbf{y})$ est le point le plus proche de $\mathbf{y}$ sur $\mathcal{S}$, la droite $(\mathbf{y},\Pi_{\mathcal{S}}(\mathbf{y}))$ est orthogonale à $\mathcal{S}$ en $\Pi_{\mathcal{S}}(\mathbf{y})$. $\mathcal{B}_{\text{lfs}}$ est tangente à $\mathcal{S}$, son centre se trouve donc également sur cette droite. Le problème est ainsi complètement symétrique par rotation autour de la droite $(\mathbf{y},\Pi_{\mathcal{S}}(\mathbf{y}))$, car les centres de $\mathcal{B}_{\text{Vor}}$, $\mathcal{B}_{\text{lfs}}$ et $\mathcal{B}_{\varepsilon}$ s'y trouvent. La Figure 5.7 représente tous ces éléments en coupe.

$\Pi_{\mathcal{S}}(\mathbf{y})$ est le point le plus proche de $\mathbf{y}$ sur $\mathcal{S}$. Nous avons donc nécessairement $\Pi_{\mathcal{S}}(\mathbf{x}) \in \overline{\mathcal{B}_{\text{Vor}}}$, et donc $\mathcal{B}_{\text{Vor}} \cap \mathcal{B}_{\text{lfs}} \neq \emptyset$. Comme $\varepsilon < 2$, si $\mathcal{B}_{\text{Vor}}$ contenait $\mathcal{B}_{\text{lfs}}$ alors $\mathcal{B}_{\text{Vor}}$ contiendrait $\mathcal{B}_{\varepsilon}$, ce qui contredit l'Équation 5.4.5. $\mathcal{B}_{\text{Vor}}$ ne contient donc pas complètement $\mathcal{B}_{\text{lfs}}$, et $\partial\mathcal{B}_{\text{Vor}} \cap \partial\mathcal{B}_{\text{lfs}} \neq \emptyset$. Soit $\mathbf{p} \in \partial\mathcal{B}_{\text{Vor}} \cap \partial\mathcal{B}_{\text{lfs}}$. Nous avons alors :

$$\mathcal{B}_{\varepsilon} \not\subset (\mathcal{B}_{\text{lfs}} \cup \mathcal{B}_{\text{Vor}}) \Leftrightarrow \mathbf{p} \in \overline{\mathcal{B}_{\varepsilon}}.$$

En utilisant les relation élémentaires du triangle $(\Pi_{\mathcal{S}}(\mathbf{y}),\mathbf{o}_{\text{lfs}},\mathbf{p})$, nous avons :

$$\| \Pi_{\mathcal{S}}(\mathbf{y}) - \mathbf{p} \|^2 = 2\text{lfs}\,(\Pi_{\mathcal{S}}(\mathbf{y}))^2\,(1 - \cos\alpha), \tag{5.4.6}$$

où α correspond à l'angle $\angle(\mathbf{p},\mathbf{o}_{\text{lfs}},\Pi_{\mathcal{S}}(\mathbf{y}))$. En utilisant les même relations dans le triangle $(\mathbf{y},\mathbf{o}_{\text{lfs}},\mathbf{p})$, nous obtenons :

$$\begin{aligned}\tilde{d}^2 &= (d + \text{lfs}\,(\Pi_{\mathcal{S}}(\mathbf{y})))^2 + \text{lfs}\,(\Pi_{\mathcal{S}}(\mathbf{y}))^2 \\ &\quad - 2(d + \text{lfs}\,(\Pi_{\mathcal{S}}(\mathbf{y})))\text{lfs}\,(\Pi_{\mathcal{S}}(\mathbf{y}))\cos\alpha \\ \tilde{d}^2 - d^2 &= 2(d + \text{lfs}\,(\Pi_{\mathcal{S}}(\mathbf{y})))\text{lfs}\,(\Pi_{\mathcal{S}}(\mathbf{y}))\,(1 - \cos\alpha).\end{aligned}$$

En utilisant l'Équation 5.4.6, $(1 - \cos\alpha)$ peut être remplacé pour donner

$$\tilde{d}^2 - d^2 = (d + \text{lfs}\,(\Pi_{\mathcal{S}}(\mathbf{y})))\frac{\| \Pi_{\mathcal{S}}(\mathbf{y}) - \mathbf{p} \|^2}{\text{lfs}\,(\Pi_{\mathcal{S}}(\mathbf{y}))}.$$

Enfin comme $\mathbf{p} \in \overline{\mathcal{B}_{\varepsilon}}$, nous avons

$$\| \Pi_{\mathcal{S}}(\mathbf{y}) - \mathbf{p} \| \leq \varepsilon\text{lfs}\,(\Pi_{\mathcal{S}}(\mathbf{y}))\,,$$

ce qui fournit le résultat :

$$\tilde{d}^2 - d^2 \leq \varepsilon^2\text{lfs}\,(\Pi_{\mathcal{S}}(\mathbf{y}))\,(\text{lfs}\,(\Pi_{\mathcal{S}}(\mathbf{y})) + d). \tag{5.4.7}$$

Cette borne est optimale car elle est atteinte lorsque $\mathcal{S}$ est exactement $\partial\mathcal{B}_{\text{lfs}}$ et $\tilde{\mathbf{x}}_i$ et $\mathbf{p}$ sont confondus.

□

Ce lemme conduit à une borne globale de l'erreur d'approximation de la distance au carré de Voronoï.

Théorème 5.1 (Borne quadratique sur $\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}} - \mathscr{F}_{\mathcal{T}\to\mathcal{S}}$, avec $\mathcal{S}$ et $\mathcal{T}$ lisses)**.** *Si $\mathcal{S}$ est bornée et différente d'un plan, alors :*

$$\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}(\mathbf{X}) - \mathscr{F}_{\mathcal{T}\to\mathcal{S}}(\mathbf{X}) \leq \varepsilon^2 |\mathcal{T}| \sigma_{\mathcal{S}} (\sigma_{\mathcal{S}} + d_{\mathcal{H}}(\mathcal{T},\mathcal{S}))$$

où $\sigma_{\mathcal{S}} = \sup_{\mathbf{x}\in\mathcal{S}} (\mathrm{lfs}\,(\mathbf{x}))$, $d_{\mathcal{H}}$ est la distance de Hausdorff entre deux surfaces et $|\mathcal{T}|$ est l'aire de $\mathcal{T}$.

Démonstration. $\mathcal{S}$ étant bornée et différente d'un plan, $\sigma_{\mathcal{S}}$ existe. Par définition de la distance de Hausdorff, pour tout $\mathbf{y} \in \mathcal{T}$, $\|\mathbf{y} - \Pi_{\mathcal{S}}(\mathbf{x})\| \leq d_{\mathcal{H}}(\mathcal{S},\mathcal{T})$.

$$\begin{aligned}
\tilde{\mathscr{F}}_{\mathcal{S}\to\mathcal{T}}(\mathbf{X}) - \mathscr{F}_{\mathcal{S}\to\mathcal{T}}(\mathbf{X}) &= \int_{\mathcal{T}} \min_{\mathbf{x}_i\in\mathbf{X}} \|\mathbf{y} - \mathbf{x}_i\|^2 \mathrm{d}\mathbf{y} - \int_{\mathcal{T}} \|\mathbf{y} - \Pi_S s(\mathbf{y})\|^2 \mathrm{d}\mathbf{y} \\
&= \int_{\mathcal{T}} \min_{\mathbf{x}_i\in\mathbf{X}} \|\mathbf{y} - \mathbf{x}_i\|^2 - \|\mathbf{y} - \Pi_S s(\mathbf{y})\|^2 \mathrm{d}\mathbf{y} \\
&\leq \int_{\mathcal{T}} \varepsilon^2 \sigma_{\mathcal{S}} (\sigma_{\mathcal{S}} + d_{\mathcal{H}}(\mathcal{T},\mathcal{S})) \mathrm{d}\mathbf{y} \\
&\leq \varepsilon^2 |\mathcal{T}| \sigma_{\mathcal{S}} (\sigma_{\mathcal{S}} + d_{\mathcal{H}}(\mathcal{T},\mathcal{S}))
\end{aligned}$$

□

Cette borne, rappelons le, est valable pour le cas où $\mathcal{S}$ est une surface lisse. Dans notre cas, cependant, $\mathcal{S}$ et $\mathcal{T}$ sont des maillages, et chaque arête d'un maillage constitue une discontinuité. Il n'est donc pas envisageable de produire un ε-échantillonnage d'une telle surface, et il est donc nécessaire de trouver d'autres outils pour mesurer la qualité d'un échantillonnage.

Cas des surfaces polygonales

Pour évaluer la densité d'un échantillonnage sur une surface polygonale, Attali et Boissonnat [3] utilisent la notion d'(ε,κ)-échantillonnage.

Définition 5.2 ((ε,κ)-échantillonnage)**.** *Un échantillonnage $\tilde{\mathbf{X}}$ d'une surface polygonale $\mathcal{S}$ est un (ε,κ)-échantillonnage si pour tout point $\mathbf{x}$ de $\mathcal{S}$ et toute face T de $\mathcal{S}$ contenant $\mathbf{x}$:*

- *il existe un échantillon $\tilde{\mathbf{x}}_i \in T$ tel que $\|\mathbf{x} - \tilde{\mathbf{x}}_i\| \leq \varepsilon$;*
- *il n'existe pas plus de κ échantillons $\tilde{\mathbf{x}}_i \in T$ tels que $\|\mathbf{x} - \tilde{\mathbf{x}}_i\| \leq 2\varepsilon$.*

Dans notre cas, nous n'avons pas besoin des conditions sur la densité maximale de l'échantillonnage, utilisant le paramètre κ. Muni d'un tel échantillonnage, une première borne d'approximation peut trivialement être déterminée.

Propriété 5.1 (Borne linéaire pour $\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}} - \mathscr{F}_{\mathcal{T}\to\mathcal{S}}$)**.** *Soit $\tilde{\mathbf{X}}$ un (ε,κ)-échantillonnage de $\mathcal{S}$, alors*

$$\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}} - \mathscr{F}_{\mathcal{S}\to\mathcal{T}} \geq 2|\mathcal{T}|\varepsilon(d_{\mathcal{H}}(\mathcal{T},\mathcal{S}) + \varepsilon),$$

où $d_{\mathcal{H}}(\mathcal{T},\mathcal{S})$ est la distance de Hausdorff et $|\mathcal{T}|$ l'aire de $\mathcal{T}$.

Démonstration. Par hypothèse, pour tout point $\mathbf{x}$ de $\mathcal{S}$, il existe $\tilde{\mathbf{x}}_i \in \tilde{\mathbf{X}}$ tel que $\|\mathbf{x} - \tilde{\mathbf{x}}\| \leq \varepsilon$. L'inégalité triangulaire nous donne donc trivialement que $\|\mathbf{y} - \tilde{\mathbf{x}}_i\| \leq |\mathbf{y} - \Pi_{\mathcal{S}}(\mathbf{x})\| + \varepsilon$, et donc

$$\|\mathbf{y} - \tilde{\mathbf{x}}_i\|^2 - |\mathbf{y} - \Pi_{\mathcal{S}}(\mathbf{x})\|^2 \leq 2\|\mathbf{y} - \Pi_{\mathcal{S}}(\mathbf{x})\|\varepsilon + \varepsilon^2 \leq 2\varepsilon(d_{\mathcal{H}}(\mathcal{T},\mathcal{S}) + \varepsilon).$$

Ainsi

$$\begin{aligned}\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}} - \mathscr{F}_{\mathcal{T}\to\mathcal{S}} &= \int_{\mathcal{T}} \min_i \|\mathbf{y}-\tilde{\mathbf{x}}_i\|^2 - \|\mathbf{y}-\Pi_{\mathcal{S}}(\mathbf{y})\|^2 \mathrm{d}\mathbf{y} \\ &\leq \int_{\mathcal{T}} 2\varepsilon(d_{\mathcal{H}}(\mathcal{T},\mathcal{S})+\varepsilon)\mathrm{d}\mathbf{y} \\ &\leq 2|\mathcal{T}|\varepsilon(d_{\mathcal{H}}(\mathcal{T},\mathcal{S})+\varepsilon).\end{aligned}$$

□

Pour obtenir une borne quadratique en ε, il est cependant nécessaire de rajouter des conditions supplémentaires sur l'échantillonnage.

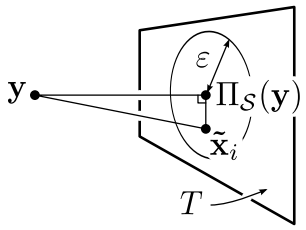
Théorème 5.2 (Borne quadratique sur $\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}} - \mathscr{F}_{\mathcal{T}\to\mathcal{S}}$ avec $\mathcal{S}$ et $\mathcal{T}$ polygonales)**.** *Soit* $\tilde{\mathbf{X}}$ *un échantillonnage de* $\mathcal{S}$ *tel que*

- *pour toute face* T *de* $\mathcal{S}$ *et tout point* $\mathbf{x}$ *de* T*, il existe un échantillon* $\tilde{\mathbf{x}}_i \in T$ *tel que* $\|\mathbf{x}-\tilde{\mathbf{x}}_i\| \leq \varepsilon$ *;*
- *pour toute arête* e *de* $\mathcal{S}$ *et tout point* $\mathbf{x}$ *de* e*, il existe un échantillon* $\tilde{\mathbf{x}}_i \in e$ *tel que* $\|\mathbf{x}-\tilde{\mathbf{x}}_i\| \leq \varepsilon$ *;*
- *pour tout sommet* $\mathbf{x}_i$ *de* $\mathcal{S}$*,* $\mathbf{x}_i \in \tilde{\mathbf{X}}$*.*

$$\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}} - \mathscr{F}_{\mathcal{T}\to\mathcal{S}} \leq |\mathcal{T}|\varepsilon^2.$$

Démonstration. Étant donné un point $\mathbf{y}$ de $\mathcal{T}$, nous procéderons par cas selon que le point le plus proche $\Pi_{\mathcal{S}}(\mathbf{y})$ de $\mathbf{y}$ sur $\mathcal{S}$ est situé dans une face, sur une arête ou sur un sommet de $\mathcal{S}$.

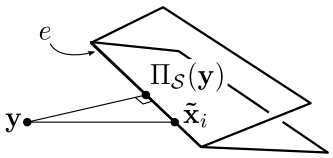
$\Pi_{\mathcal{S}}(\mathbf{y})$ est dans une face T :



dans ce cas, la droite $(\mathbf{y},\Pi_{\mathcal{S}}(\mathbf{y}))$ est orthogonale à T. Un échantillons $\tilde{\mathbf{x}}_i \in T$ est tel que $\|\Pi_{\mathcal{S}}(\mathbf{y}) - \tilde{\mathbf{x}}_i\| \leq \varepsilon$. Le triangle $(\mathbf{y},\Pi_{\mathcal{S}}(\mathbf{y}),\tilde{\mathbf{x}}_i)$ est rectangle, le théorème de Pythagore nous donne donc

$$\|\mathbf{y}-\tilde{\mathbf{x}}_i\|^2 - \|\mathbf{y}-\Pi_{\mathcal{S}}(\mathbf{y})\|^2 \leq \varepsilon^2.$$

$\Pi_{\mathcal{S}}(\mathbf{y})$ est dans une arête e :



ici, la droite $(\mathbf{y},\Pi_{\mathcal{S}}(\mathbf{y}))$ est perpendiculaire à e. Un échantillons $\tilde{\mathbf{x}}_i \in e$ est tel que $\|\Pi_{\mathcal{S}}(\mathbf{y}) - \tilde{\mathbf{x}}_i\| \leq \varepsilon$. Le triangle $(\mathbf{y},\Pi_{\mathcal{S}}(\mathbf{y}),\tilde{\mathbf{x}}_i)$ est rectangle, le théorème de Pythagore nous donne donc

$$\|\mathbf{y}-\tilde{\mathbf{x}}_i\|^2 - \|\mathbf{y}-\Pi_{\mathcal{S}}(\mathbf{y})\|^2 \leq \varepsilon^2.$$

$\Pi_{\mathcal{S}}(\mathbf{y})$ est sur un sommet $\mathbf{x}_i$: nous avons alors $\mathbf{x}_i \in \tilde{\mathbf{X}}$ et donc l'échantillon $\tilde{\mathbf{x}}_i \in \tilde{\mathbf{X}}$ le plus proche de $\mathbf{y}$ coïncide avec $\mathbf{x}_i$, d'où

$$\|\mathbf{y}-\tilde{\mathbf{x}}_i\|^2 = \|\mathbf{y}-\Pi_{\mathcal{S}}(\mathbf{y})\|^2.$$

Nous obtenons donc finalement

$$\begin{aligned}\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}} - \mathscr{F}_{\mathcal{T}\to\mathcal{S}} &= \int_{\mathcal{T}} \min_i \|\mathbf{y}-\tilde{\mathbf{x}}_i\|^2 - \|\mathbf{y}-\Pi_{\mathcal{S}}(\mathbf{y})\|^2 \mathrm{d}\mathbf{y} \\ &\leq \int_{\mathcal{T}} \varepsilon^2 \mathrm{d}\mathbf{y} = |\mathcal{T}|\varepsilon^2.\end{aligned}$$

□

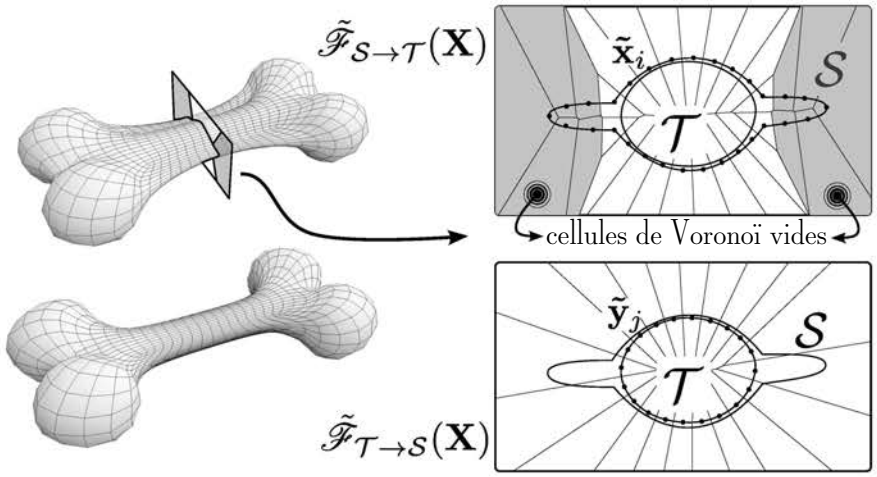


FIGURE 5.8 – La surface minimisant $\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}$ a des « ailettes » qui ne peuvent pas être éliminées, car les cellules de Voronoï des échantillons qui les approximent n'intersectent pas la surface cible $\mathcal{T}$. Le terme symétrique – en bas – permet de considérer ces portions de la surface $\mathcal{S}$ à ajuster, car les cellules de Voronoï des échantillons de $\mathcal{T}$ partitionnent entièrement l'espace. L'intégralité de $\mathcal{S}$ est donc ainsi prise en compte.

Ce théorème montre donc que notre fonction objectif $\tilde{\mathscr{F}}$ est une bonne approximation de $\mathscr{F}$. Il faut néanmoins rester conscient que la surface $\mathcal{S}$ est une variable, et que nous ne fournissons pas de garanties sur le fait qu'au cours de l'optimisation, le paramètre ε correspondant à son échantillonnage ne croîtra pas. Il ne faut donc pas voir ce paramètre comme fixé a priori au moment de la création de l'échantillonnage $\tilde{\mathbf{X}}$. Par contre, à chaque itération, il est toujours possible étant donné l'échantillonnage de déterminer le paramètre ε qu'il respecte, et ainsi d'avoir une borne sur l'erreur commise pour le calcul effectué à cette itération.

5.4.3 Nécessité du terme symétrique

Les méthodes de l'état de l'art n'utilisent pas de formulation symétrique, et n'utilisent que le terme $\mathscr{F}_{\mathcal{S}\to\mathcal{T}}$ pour définir leur fonction objectif. Lorsque $\mathcal{T}$ contient $\mathcal{S}$, alors pour chaque échantillon de $\mathcal{S}$, le point le plus proche sur $\mathcal{T}$ est à distance nulle. Ainsi nous avons

$$\mathscr{F}_{\mathcal{S}\to\mathcal{T}} = \int_{\mathcal{S}} \|\mathbf{x} - \Pi_{\mathcal{T}}(\mathbf{x})\|^2 \mathrm{d}\mathbf{x} = 0,$$

et la fonction objectif est minimisée. En n'utilisant que $\mathscr{F}_{\mathcal{S}\to\mathcal{T}}$, les points de $\mathcal{T}$ qui ne sont les plus proches pour aucun point de $\mathcal{S}$ ne sont pas pris en compte. L'utilisation de la formulation symétrique permet de s'assurer que la fonction objectif n'est minimisée que lorsque les deux surfaces coïncident, et de prendre en compte chaque point de chaque surface durant le processus de minimisation.

Dans notre cas, le terme le plus simple à utiliser est le terme $\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}$, car il correspond directement à la fonction objectif utilisée pour faire en sorte que le diagramme de Voronoï des échantillons $\tilde{\mathbf{X}}$ de $\mathcal{S}$ soit barycentrique. Nous avons vu en Section 4.3.3 et nous reverrons dans un instant que le gradient de cette fonction objectif a une expression très simple. La Figure 5.8 montre qu'en se restreignant à ce terme même pour des objets relativement simples, le résultat obtenu ne prend pas en compte l'intégralité de $\mathcal{S}$. La position des

échantillons

$$\{\tilde{\mathbf{x}}_i \in \tilde{\mathbf{X}}, \Omega_{i|\mathcal{T}} = \emptyset\}$$

n'a aucune influence sur la valeur de $\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}$, et les portions correspondantes de $\mathcal{S}$ ne sont soumises qu'au terme de lissage $\mathscr{R}$. L'utilisation de la formulation symétrique permet en revanche d'obtenir un résultat satisfaisant. L'utilisation seule de $\tilde{\mathscr{F}}_{\mathcal{S}\to\mathcal{T}}$ donne également lieu à des effets indésirables, car en contractant $\mathcal{S}$ en un point, nous obtenons $\tilde{\mathscr{F}}_{\mathcal{S}\to\mathcal{T}} = \mathscr{R} = 0$, alors que le résultat obtenu est mauvais.

5.4.4 Procédé de minimisation

Pour minimiser notre fonction objectif donnée par l'Équation 5.4.4, nous utilisons la méthode L-BFGS [65], de la même manière que Lévy et Liu [62] pour l'obtention d'un diagramme de Voronoï barycentrique. Cette méthode est présentée en Section 4.3.3. Il s'agit d'une méthode de type Newton, qui requiert de connaître la valeur et le gradient de la fonction objectif. Au cours de l'optimisation, les valeurs successives du gradient sont utilisée pour obtenir une approximation de la Hessienne, et accélérer la minimisation. Bien que seul le gradient soit requis pour les calculs, il est nécessaire que la fonction objectif soit C^2 pour que l'algorithme converge vers une solution acceptable. Dans le cas de $\tilde{\mathscr{F}}$, nous avons :

$\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}$ est le *bruit de quantification*, qui est la fonction objectif minimisée pour obtenir un diagramme de Voronoï barycentrique. Elle est de classe C^2 presque partout, comme évoqué en Section 4.3.3 ;

$\tilde{\mathscr{F}}_{\mathcal{S}\to\mathcal{T}}$ est obtenu en inversant les rôles des variables et des paramètres dans $\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}$. Sa continuité n'a pas encore été étudiée et fera l'objet de travaux futurs. Dans nos travaux, nous avons observé un comportement stable du solveur L-BFGS, ce qui est encourageant ;

$\mathscr{R}$ est une forme quadratique. Il est donc C^∞.

La minimisation est réalisée en suivant les étapes de l'Algorithme 5.1.

Valeur de $\tilde{\mathscr{F}}$

Pour calculer la valeur de $\tilde{\mathscr{F}}$ à chaque itération (ligne 8), nous utilisons la méthode décrite par Lévy et Liu [62], et donnée en Section 4.3.3. Les deux termes $\tilde{\mathscr{F}}_{\mathcal{S}\to\mathcal{T}}$ et $\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}$ se calculent de la même façon, il suffit d'inverser l'échantillonnage et la surface utilisés. Dans le cas de $\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}$ la surface est $\mathcal{T}$ et les échantillons sons $\tilde{\mathbf{X}}$. Chaque cellule de Voronoï restreinte $\Omega_{i|\mathcal{T}}$ du diagramme de Voronoï de $\tilde{\mathbf{X}}$ restreint par $\mathcal{T}$ est décomposée en un ensemble de triangles. La contribution au gradient d'un triangle $T = (\mathbf{c}_1,\mathbf{c}_2,\mathbf{c}_3)$ de la cellule restreinte de $\tilde{\mathbf{x}}_i$ est donnée par l'Équation 4.3.13 que nous spécialisons ici au cas où $p = 2$ qui nous concerne, étant donné que nous utilisons la distance euclidienne :

$$\tilde{\mathscr{F}}^T_{\mathcal{S}\to\mathcal{T}} = \frac{|T|}{6} \sum_{1\le k\le l\le 3} (\mathbf{c}_k - \tilde{\mathbf{x}}_i)\cdot(\mathbf{c}_l - \tilde{\mathbf{x}}_i), \tag{5.4.8}$$

où $|T|$ est l'aire de $\mathcal{S}$. Le calcul est donc réalisé en itérant sur chaque triangle de chaque cellule de Voronoï restreinte.

La surface $\mathcal{S}$ varie à chaque itération, et son échantillonnage $\tilde{\mathbf{X}}$ suit ces variations. Il n'est donc pas possible de conserver les diagrammes de Voronoï restreints d'une itération sur l'autre. En ce qui concerne les diagrammes de Voronoï, seul celui de $\tilde{\mathbf{X}}$ doit être recalculé à chaque itération, l'échantillonnage $\tilde{\mathbf{Y}}$ étant constant.

Entrées :
– $\mathcal{S}^{(0)}$ la surface source à ajuster ;
– $\mathcal{T}$ la surface cible ;
– λ le facteur d'influence du terme de régularisation.
Sorties :
– $\mathcal{S}^{(*)}$ la surface source ajustée à $\mathcal{T}$.
Données :
– $\mathcal{D}_{\tilde{\mathbf{X}}}$ le diagramme de Voronoï de $\tilde{\mathbf{X}}$;
– $\mathcal{D}_{\tilde{\mathbf{Y}}}$ le diagramme de Voronoï de $\tilde{\mathbf{Y}}$;
– $\mathcal{S}^{(\mathsf{k})}$ la surface optimisée à l'itération k ;
– $\mathbf{X}^{(\mathsf{k})}$ les sommets de $\mathcal{S}^{(\mathsf{k})}$.

Algorithme
- $\mathbf{L}^2 \leftarrow$ carré du laplacien de graphe pour $\mathcal{S}^{(0)}$;
- $\tilde{\mathbf{X}} \leftarrow$ échantillonnage de $\mathcal{S}$; $\tilde{\mathbf{Y}} \leftarrow$ échantillonnage de $\mathcal{T}$;
- $\mathcal{D}_{\tilde{\mathbf{Y}}} \leftarrow$ diagramme de Voronoï de $\tilde{\mathbf{Y}}$;
- $\mathsf{k} \leftarrow 0$;
- **tant que** *le minimum n'est pas atteint* **faire**
  - $\mathcal{D}_{\tilde{\mathbf{X}}} \leftarrow$ diagramme de Voronoï de $\tilde{\mathbf{X}}$;
  - $\tilde{\mathscr{F}} \leftarrow \tilde{\mathscr{F}}_{\mathcal{S}\to\mathcal{T}} + \tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}$; // Équation 5.4.8
  - $\tilde{\mathscr{F}} \leftarrow (1-\lambda)\tilde{\mathscr{F}} + \lambda\|\mathbf{L}\mathbf{X}^{(\mathsf{k})}\|^2$;
  - $\frac{\partial\tilde{\mathscr{F}}}{\partial\mathbf{X}} \leftarrow$ `gradient_Fts`$(\mathcal{D}_{\tilde{\mathbf{X}}},\mathcal{T})$; // Équation 5.4.10
  - $\frac{\partial\tilde{\mathscr{F}}}{\partial\mathbf{X}} \overset{+}{\leftarrow}$ `gradient_Fst`$(\mathcal{D}_{\tilde{\mathbf{Y}}},\mathcal{S})$; // Équation 5.4.12
  - $\frac{\partial\tilde{\mathscr{F}}}{\partial\mathbf{X}} \leftarrow (1-\lambda)\frac{\partial\tilde{\mathscr{F}}}{\partial\mathbf{X}} + 2\lambda\mathbf{X}^{(\mathsf{k})^t}\mathbf{L}^t\mathbf{L}$; // Équation 5.4.9
  - $\mathcal{S}^{(\mathsf{k}+1)} \leftarrow$ `minimisation_LBFGS`$(\tilde{\mathscr{F}}, \frac{\partial\tilde{\mathscr{F}}}{\partial\mathbf{X}})$;
  - $\mathsf{k} \overset{+}{\leftarrow} 1$;
- **renvoyer** $\mathcal{S}^{(\mathsf{k})}$;

Algorithme 5.1: `ajuste`$(\mathcal{S}^{(0)},\mathcal{T},\lambda)$ – minimisation de VSDM

Gradient de $\tilde{\mathscr{F}}$

Comme précédemment au Chapitre 4, nous utilisons les notations de Minka [73] pour la dérivations de fonctions matricielles multivariées. En particulier pour une fonction $\mathscr{F}$ sur un ensemble de variables $\mathbf{X} = \{x_i\}_{i=1}^n$, nous noterons

$$\frac{\partial \mathscr{F}}{\partial \mathbf{X}} = \left(\frac{\partial \mathscr{F}}{\partial x_1}, \dots, \frac{\partial \mathscr{F}}{\partial x_n}\right) = (\nabla_{\mathbf{X}} \mathscr{F})^t,$$

où $\nabla_{\mathbf{X}}$ est le gradient de $\mathscr{F}$ par rapport à $\mathbf{X}$. Cette notation permet de manipuler plus facilement les fonctions vectorielles, et d'appliquer naturellement la règle de dérivation des fonctions composées.

Le gradient de $\tilde{\mathscr{F}}$ est constitué du gradient de ses trois terme :

$\mathscr{R}$ est une forme quadratique, et son gradient est simplement donné par

$$\frac{\partial \mathscr{R}}{\partial \mathbf{X}} = \frac{\partial \|\mathbf{L}\mathbf{X}\|^2}{\partial \mathbf{X}} = 2\mathbf{X}^t\mathbf{L}^t\mathbf{L} \tag{5.4.9}$$

$\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}$ est la fonction objectif minimisée par présentée en Section 4.3.2 pour obtenir un diagramme de Voronoï barycentrique. La différence est qu'ici les variables sont les sommets de $\mathcal{S}$ et non les sites du diagramme de Voronoï, qui sont les échantillons $\tilde{\mathbf{X}}$. D'après l'Équation 4.3.7, nous avons

$$\frac{\partial \tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}}{\partial \tilde{\mathbf{x}}_i} = 2m_i(\tilde{\mathbf{x}}_i - \mathbf{g}_i)^t,$$

où m_i est l'aire de la cellule de Voronoï restreinte $\Omega_{i|\mathcal{T}}$ et $\mathbf{g}_i$ son barycentre. Chaque échantillon $\tilde{\mathbf{x}}_i$ s'exprime en coordonnées barycentriques par rapport aux trois sommets du triangle de $\mathcal{S}$ qui le contient. En organisant ces coefficients dans une matrice $M_{\tilde{\mathbf{X}}}$ telle que $\tilde{\mathbf{X}} = M_{\tilde{\mathbf{X}}}\mathbf{X}$, la règle de dérivation des fonctions composées nous donne alors

$$\frac{\partial \tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}}{\partial \mathbf{X}} = \frac{\partial \tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}}{\partial \tilde{\mathbf{X}}}\frac{\partial \tilde{\mathbf{X}}}{\partial \mathbf{X}} = \frac{\partial \tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}}{\partial \tilde{\mathbf{X}}} M_{\tilde{\mathbf{X}}}. \tag{5.4.10}$$

$\tilde{\mathscr{F}}_{\mathcal{S}\to\mathcal{T}}$ est identique à $\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}$, mais les variables et les paramètres sont inversés. Ici, ce ne sont plus les sites du diagramme de Voronoï restreint qui sont les variables mais les sommets de $\mathcal{S}$. Nous calculons son gradient à partir de l'Équation 5.4.8. Étant donné un triangle $T = (\mathbf{c}_1,\mathbf{c}_2,\mathbf{c}_3)$ de la cellule de Voronoï restreinte d'un échantillon $\tilde{\mathbf{y}}_j$, posons

$$f = \sum_{1\le k\le l\le 3} (\mathbf{c}_k - \tilde{\mathbf{y}}_j)\cdot(\mathbf{c}_l - \tilde{\mathbf{y}}_j) \qquad \text{et} \qquad \mathbf{n}_T = (\mathbf{c}_1 - \mathbf{c}_2)\times(\mathbf{c}_1 - \mathbf{c}_3).$$

Le vecteur $\mathbf{n}_T$ est la normale non normalisée de T. En particulier, l'aire de T s'exprime comme $2|T| = \|\mathbf{n}_T\|$. La différentielle de la contribution $\tilde{\mathscr{F}}^T_{\mathcal{S}\to\mathcal{T}}$ de T au gradient vaut

$$\mathrm{d}\tilde{\mathscr{F}}^T_{\mathcal{S}\to\mathcal{T}} = \mathrm{d}\left[\frac{\|\mathbf{n}_T\| f}{12}\right] = \frac{1}{12}(\mathrm{d}\|\mathbf{n}_T\| f + \|\mathbf{n}_T\|\mathrm{d}f),$$

avec

$$\mathrm{d}f = \sum_{k=1}^{3} \mathrm{d}\mathbf{c}_k\cdot(2\mathbf{c}_k+\mathbf{c}_{k+1}+\mathbf{c}_{k+2}-3\tilde{\mathbf{y}}_j) \quad \text{et} \quad \mathrm{d}\|\mathbf{n}_T\| = \frac{1}{\|\mathbf{n}_T\|}\sum_{k=1}^{3} \mathrm{d}\mathbf{c}_k\cdot((\mathbf{c}_{k+1}-\mathbf{c}_{k+2})\times\mathbf{n}_T),$$

en considérant les indices des sommets de T modulo 3. En combinant ces expression, nous obtenons finalement

$$\begin{aligned} \mathrm{d}\tilde{\mathscr{F}}^{T}_{\mathcal{S}\to\mathcal{T}} &= \frac{1}{12}\sum_{k=1}^{3}\left[(2\mathbf{c}_k+\mathbf{c}_{k+1}+\mathbf{c}_{k+2}-3\tilde{\mathbf{y}}_j)^t\|\mathbf{n}_T\| + f((\mathbf{c}_{k+1}-\mathbf{c}_{k+2})\times\frac{\mathbf{n}_T}{\|\mathbf{n}_T\|})\right]^t \mathrm{d}\mathbf{c}_k \\ &= \sum_{k=1}^{3}\frac{\partial\tilde{\mathscr{F}}^{T}_{\mathcal{S}\to\mathcal{T}}}{\partial\mathbf{c}_k}\mathrm{d}\mathbf{c}_k. \end{aligned} \tag{5.4.11}$$

Cette expression nous donne le gradient de $\tilde{\mathscr{F}}_{\mathcal{S}\to\mathcal{T}}$ par rapport aux sommets des triangles du diagramme de Voronoï de $\tilde{\mathbf{Y}}$ restreint à $\mathcal{S}$. Pour obtenir enfin le gradient par rapport à $\mathbf{X}$ il est nécessaire de composer ce gradient avec celui des points $\mathbf{c}_0$, $\mathbf{c}_1$ et $\mathbf{c}_2$, donnés par les Équations 5.4.13 à 5.4.15 :

$$\frac{\partial\tilde{\mathscr{F}}_{\mathcal{S}\to\mathcal{T}}}{\partial\mathbf{X}} = \sum_{\tilde{\mathbf{y}}_j\in\tilde{\mathbf{Y}}}\ \sum_{T\in\Omega_{j|\mathcal{S}}}\ \sum_{k=1}^{3}\frac{\partial\tilde{\mathscr{F}}^{T}_{\mathcal{S}\to\mathcal{T}}}{\partial\mathbf{c}_k}\frac{\partial\mathbf{c}_k}{\partial\mathbf{X}} \tag{5.4.12}$$

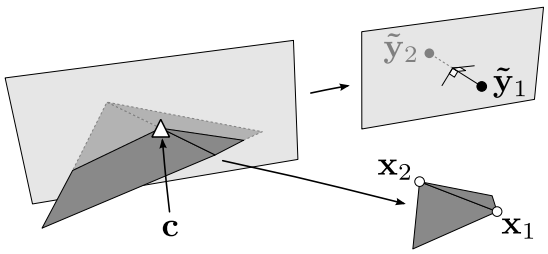
Gradient des sommets de diagramme de $\tilde{\mathbf{Y}}$ restreint à $\mathcal{S}$

Comme dans la Section 4.3.3 nous distinguons trois types de sommets différents au sein d'un diagramme de Voronoï restreint (Figure 4.6) :

○ **type i :** $\mathbf{c}$ est un sommet $\mathbf{x}_i$ de $\mathcal{S}$. Dans ce cas, nous avons :

$$\frac{\partial\mathbf{c}}{\partial\mathbf{x}_i} = I_{3\times3} \tag{5.4.13}$$

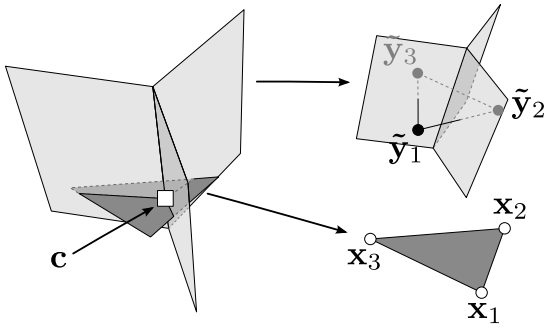
△ **type ii :**



$\mathbf{c}$ est l'intersection entre le bissecteur de deux échantillons $\tilde{\mathbf{y}}_1$ et $\tilde{\mathbf{y}}_2$ de $\mathcal{T}$, et l'arête de $\mathcal{S}$ reliant les sommets $\mathbf{x}_1$ et $\mathbf{x}_2$. Les Jacobiennes de $\mathbf{c}$ par rapport aux points $\mathbf{x}_1$ et $\mathbf{x}_2$ sont données par :

$$\begin{aligned} \frac{\partial\mathbf{c}}{\partial\mathbf{x}_1} &= \mathbf{e}\frac{\partial u}{\partial\mathbf{x}1} + (1-u)I_{3\times3} \\ \frac{\partial\mathbf{c}}{\partial\mathbf{x}_2} &= \mathbf{e}\frac{\partial u}{\partial\mathbf{x}_2} + uI_{3\times3} \end{aligned} \qquad \text{où :} \begin{cases} \mathbf{e} &= (\mathbf{x}_2-\mathbf{x}_1) \\ \mathbf{n} &= (\tilde{\mathbf{y}}_2-\tilde{\mathbf{y}}_1) \\ k &= \mathbf{n}\cdot\mathbf{e} \\ h &= \mathbf{n}\cdot(\tilde{\mathbf{y}}_1+\tilde{\mathbf{y}}_2)/2 \\ u &= (h-\mathbf{n}\cdot\mathbf{x}_1)/k \\ \partial u/\partial\mathbf{x}_1 &= (h-\mathbf{n}\cdot\mathbf{x}_2)\mathbf{n}^t/k^2 \\ \partial u/\partial\mathbf{x}_2 &= -(h-\mathbf{n}\cdot\mathbf{x}_1)\mathbf{n}^t/k^2 \end{cases} \tag{5.4.14}$$

□ **type iii :**



$\mathbf{c}$ est l'intersection entre les trois bissecteurs des segments $[\tilde{\mathbf{y}}_1,\tilde{\mathbf{y}}_2]$, $[\tilde{\mathbf{y}}_2,\tilde{\mathbf{y}}_3]$ et $[\tilde{\mathbf{y}}_3,\tilde{\mathbf{y}}_1]$, et un triangle de $\mathcal{S}$ défini par ses trois sommets $\mathbf{x}_1$, $\mathbf{x}_2$ et $\mathbf{x}_3$. Les Jacobiennes de $\mathbf{c}$ par rapport à ces trois points sont données par :

$$\begin{array}{l} \dfrac{\partial \mathbf{c}}{\partial \mathbf{x}_1} = \mathbf{e}\dfrac{\partial u}{\partial \mathbf{x}_1} \\ \dfrac{\partial \mathbf{c}}{\partial \mathbf{x}_2} = \mathbf{e}\dfrac{\partial u}{\partial \mathbf{x}_2} \\ \dfrac{\partial \mathbf{c}}{\partial \mathbf{x}_3} = \mathbf{e}\dfrac{\partial u}{\partial \mathbf{x}_3} \end{array} \quad \text{où : } \left\{ \begin{array}{rl} \mathbf{e} & = (\tilde{\mathbf{y}}_1 - \tilde{\mathbf{y}}_2) \times (\tilde{\mathbf{y}}_1 - \tilde{\mathbf{y}}_3) \\ \mathbf{n} & = (\mathbf{x}_1 - \mathbf{x}_2) \times (\mathbf{x}_1 - \mathbf{x}_3) \\ k & = \mathbf{n} \cdot \mathbf{e} \\ \partial u/\partial \mathbf{x}_1 & = ((\mathbf{x}_2 - \mathbf{x}_3) \times (\mathbf{x}_1 - \mathbf{c}) + \mathbf{n})^t/k \\ \partial u/\partial \mathbf{x}_2 & = ((\mathbf{x}_3 - \mathbf{x}_1) \times (\mathbf{x}_1 - \mathbf{c}))^t/k \\ \partial u/\partial \mathbf{x}_3 & = ((\mathbf{x}_1 - \mathbf{x}_2) \times (\mathbf{x}_1 - \mathbf{c}))^t/k \end{array} \right. \tag{5.4.15}$$

Ces formules ont été obtenues en exprimant dans chaque cas le sommet à dériver comme l'intersection entre un plan $\mathcal{P}$ et une droite $\mathcal{L}$:

- $\mathcal{P}$ est défini par un point $\mathbf{p}$ et une normale $\mathbf{n}$;
- $\mathcal{L}$ est définie par un point $\mathbf{v}$ et une direction $\mathbf{w}$.

L'intersection $\mathbf{c}$ de $\mathcal{P}$ et $\mathcal{L}$ est donnée par

$$\mathbf{c} = \mathbf{v} + u\mathbf{w} \quad \text{avec} \quad u = \frac{(\mathbf{p} - \mathbf{v}).\mathbf{n}}{\mathbf{w}.\mathbf{n}}. \tag{5.4.16}$$

Il suffit ensuite d'exprimer ces différents éléments en fonction des variables $\mathbf{x}$ et des paramètres $\tilde{\mathbf{Y}}$ selon le type de $\mathbf{c}$:

type ii : le plan est le bissecteur de $\tilde{\mathbf{y}}_1$ et $\tilde{\mathbf{y}}_2$ et la droite l'arête de $\mathcal{S}$ joignant $\mathbf{x}_1$ et $\mathbf{x}_2$.

$$\left\{ \begin{array}{l} \mathbf{p} = \dfrac{\tilde{\mathbf{y}}_1 + \tilde{\mathbf{y}}2}{2} \\ \mathbf{n} = \tilde{\mathbf{y}}_1 - \tilde{\mathbf{y}}2 \end{array} \right. \quad \text{et} \quad \left\{ \begin{array}{l} \mathbf{v} = \mathbf{x}_1 \\ \mathbf{w} = \mathbf{x}_2 - \mathbf{x}_1 \end{array} \right.$$

La différentielle de $\mathbf{c}$ vaut

$$\mathrm{d}\mathbf{c} = \mathrm{d}\mathbf{x}_1 + u\mathrm{d}\mathbf{w} + \mathrm{d}\mathbf{w}u = u\mathrm{d}\mathbf{x}_2 + (1-u)\mathrm{d}\mathbf{x}_1 + \mathrm{d}u\mathbf{w},$$

avec

$$\begin{aligned} \mathrm{d}u &= \frac{1}{(\mathbf{w}.\mathbf{n})^2}\left[-(\mathrm{d}\mathbf{x}_1.\mathbf{n})(\mathbf{w}.\mathbf{n}) - ((\mathbf{p}-\mathbf{v}).\mathbf{n})((\mathrm{d}\mathbf{x}_2 - \mathrm{d}\mathbf{x}_1).\mathbf{n})\right] \\ &= \frac{1}{(\mathbf{w}.\mathbf{n})^2}\left[((\mathbf{p}-\mathbf{x}_2).\mathbf{n})\mathbf{n}^t\mathrm{d}\mathbf{x}_1 - ((\mathbf{p}-\mathbf{x}_1).\mathbf{n})\mathbf{n}^t\mathrm{d}\mathbf{x}_2\right] \end{aligned}$$

Nous obtenons finalement les Jacobiennes

$$\frac{\partial \mathbf{c}}{\partial \mathbf{x}_1} = \frac{(\mathbf{p}-\mathbf{x}_2).\mathbf{n}}{\mathbf{w}.\mathbf{n}}\mathbf{n}^t + I_{3\times 3} \quad \text{et} \quad \frac{\partial \mathbf{c}}{\partial \mathbf{x}_2} = -\frac{(\mathbf{p}-\mathbf{x}_1).\mathbf{n}}{\mathbf{w}.\mathbf{n}}\mathbf{n}^t + (1-u)I_{3\times 3} \tag{5.4.17}$$

type iii : le plan est la face de T reliant $\mathbf{x}_1$, $\mathbf{x}_2$ et $\mathbf{x}_3$, et l'arête est à l'intersection des bissecteurs de $\tilde{\mathbf{y}}_1$, $\tilde{\mathbf{y}}_2$ et $\tilde{\mathbf{y}}_3$.

$$\begin{cases} \mathbf{p} = \mathbf{x}_1 \\ \mathbf{n} = (\mathbf{x}_1 - \mathbf{x}_2) \times (\mathbf{x}1 - \mathbf{x}_3) \end{cases} \quad \text{et} \quad \begin{cases} \mathbf{v} = \mathbf{c} \\ \mathbf{w} = (\tilde{\mathbf{y}}_1 - \tilde{\mathbf{y}}_2) \times (\tilde{\mathbf{y}}_1 - \tilde{\mathbf{y}}_3) \end{cases}$$

Nous pouvons ici nous permettre de définir la droite $\mathcal{L}$ par le point $\mathbf{c}$ étant donné que cette droite ne varie pas. Nous avons

$$\mathrm{d}\mathbf{c} = \mathrm{d}u\mathbf{w} \quad \text{et} \quad \mathrm{d}\mathbf{n} = \mathrm{d}\mathbf{x}_1 \times (\mathbf{x}_2 - \mathbf{x}_3) + \mathrm{d}\mathbf{x}_2 \times (\mathbf{x}_3 - \mathbf{x}_1) + \mathrm{d}\mathbf{x}_3 \times (\mathbf{x}_1 - \mathbf{x}_2),$$

avec

$$\mathrm{d}u = \frac{1}{(\mathbf{w}.\mathbf{n})^2}[((\mathrm{d}\mathbf{p}.\mathbf{n}) + (\mathbf{p} - \mathbf{v}).\mathrm{d}\mathbf{n})(\mathbf{w}.\mathbf{n}) - \underbrace{((\mathbf{p} - \mathbf{v}).\mathbf{n})}_{=0}\mathrm{d}(\mathbf{w}.\mathbf{n})],$$

où $(\mathbf{p} - \mathbf{v}).\mathbf{n} = 0$ car $\mathbf{p} = \mathbf{x}_1$ et $\mathbf{v} = \mathbf{c}$ sont tous les deux contenus dans $\mathcal{P}$. En développant $\mathrm{d}\mathbf{n}$ et en simplifiant, nous obtenons finalement

$$\mathrm{d}u = \frac{1}{\mathbf{w}.\mathbf{n}}\left[\mathbf{n}.\mathrm{d}\mathbf{x}_1 + \sum_{i=1}^{3}(\mathbf{x}_{i+1} - \mathbf{x}_{i+2}) \times (\mathbf{x}_1 - \mathbf{c}).\mathrm{d}\mathbf{x}_i\right],$$

où les indices i sont pris modulo 3. Nous obtenons enfin les jacobiennes

$$\begin{aligned} \frac{\partial \mathbf{c}}{\partial \mathbf{x}_1} &= \frac{((\mathbf{x}_2 - \mathbf{x}_3) \times (\mathbf{x}_1 - \mathbf{c}) + \mathbf{n})^t}{\mathbf{w}.\mathbf{n}} \\ \frac{\partial \mathbf{c}}{\partial \mathbf{x}_2} &= \frac{((\mathbf{x}_3 - \mathbf{x}_1) \times (\mathbf{x}_1 - \mathbf{c}))^t}{\mathbf{w}.\mathbf{n}} \\ \frac{\partial \mathbf{c}}{\partial \mathbf{x}_3} &= \frac{((\mathbf{x}_1 - \mathbf{x}_2) \times (\mathbf{x}_1 - \mathbf{c}))^t}{\mathbf{w}.\mathbf{n}} \end{aligned} \tag{5.4.18}$$

5.4.5 Ajustement de surfaces de subdivision

La gestion des surfaces se fait très naturellement sans changement majeur. Dans ce cas, les variables $\mathbf{X}$ sont les sommets du maillage de contrôle de $\mathcal{S}$. Une surface de subdivision est générée à partir d'un maillage de contrôle par un processus de raffinement qui subdivise récursivement le maillage de contrôle et un processus de lissage qui fournit une nouvelle position pour les sommets du maillage obtenu à chaque itération. Ce processus converge vers une surface lisse. Un exemple est fourni en Figure 5.9. Les schémas de subdivision les plus couramment utilisés sont ceux de Catmull et Clark [16] pour les maillages quadrangulaires et de Loop [69] pour les maillages triangulaires.

Le point important est qu'à chaque niveau de subdivision, tous les sommets du maillage s'expriment sous forme d'une combinaison linéaire des sommets du maillage au niveau précédent. En notant $\mathcal{S}_k$ le maillage obtenu après k subdivisions et $\mathbf{X}_k$ ses sommets, cette relation s'exprime sous forme matricielle en introduisant la matrice M_k par

$$\mathbf{X}_k = M_k\mathbf{X}_{k-1} = M_kM_{k-1}\dots M_1\mathbf{X}.$$

Pour ajuster le résultat de k subdivision à une surface $\mathcal{T}$, nous calculons la valeur de $\tilde{\mathscr{F}}$ comme précédemment en utilisant l'Équation 5.4.8, et les Équations 5.4.9 à 5.4.12 nous fournissent le gradient par rapport à $\mathbf{X}_k$. Le gradient par rapport à $\mathbf{X}$ est finalement obtenu en utilisant la règle de dérivation des fonctions composées

$$\frac{\partial \tilde{\mathscr{F}}}{\partial \mathbf{X}} = \frac{\partial \tilde{\mathscr{F}}}{\partial \mathbf{X}_k}\frac{\partial \mathbf{X}_k}{\partial \mathbf{X}} = \frac{\partial \tilde{\mathscr{F}}}{\partial \mathbf{X}_k}M_k\dots M_1.$$

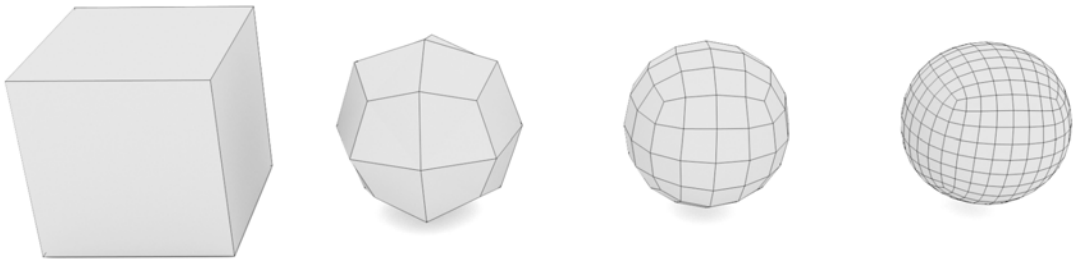


FIGURE 5.9 – Surface de subdivision générée en utilisant le schéma de Catmull et Clark [16]. À chaque étape, chaque face est subdivisée en 4 et tous les sommets sont replacés. Ce procédé tend vers une surface lisse.

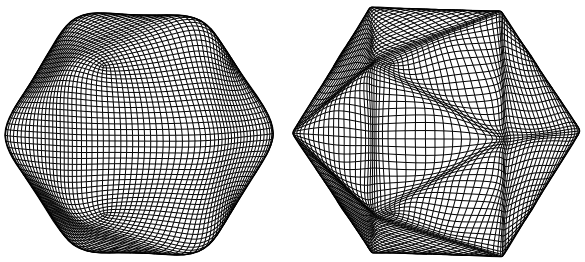


FIGURE 5.10 – Ajustement d'un cube sur un icosaèdre. Le résultat de la méthode sans l'anisotropie normale a tendance à lisser les arêtes franches (au centre). L'introduction de l'anisotropie dans la direction normale à la surface permet de rendre l'ajustement sensible aux aspérités (à gauche).

5.4.6 Anisotropie normale

Il est possible que le résultat du processus de minimisation ne respecte pas précisément les arêtes franches du maillage cible, à cause d'un lissage excessif. Ce comportement peut être amélioré en introduisant une distance anisotrope par rapport à la surface, comme réalisé précédemment dans Lévy et Liu [62] et décrit en Section 4.3.3. Cette modification affecte les termes $\tilde{\mathscr{F}}_{\mathcal{S}\to\mathcal{T}}$ et $\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}}$, qui deviennent :

$$\tilde{\mathscr{F}}_{\mathcal{S}\to\mathcal{T}} = \sum_{\tilde{\mathbf{y}}_j\in\tilde{\mathbf{Y}}} \sum_{T\in\Omega_j\cap\mathcal{S}} \int_T \|M_T(s)(\mathbf{x}-\tilde{\mathbf{y}}_j)\|^2 \mathrm{d}\mathbf{x} \tag{5.4.19}$$

$$\tilde{\mathscr{F}}_{\mathcal{T}\to\mathcal{S}} = \sum_{\tilde{\mathbf{x}}_i\in\tilde{\mathbf{X}}} \sum_{T\in\Omega_i\cap\mathcal{T}} \int_T \|M_T(s)(\mathbf{y}-\tilde{\mathbf{x}}_i)\|^2 \mathrm{d}\mathbf{y} \tag{5.4.20}$$

avec

$$M_T(s) = (s-1)\mathbf{n}_T\mathbf{n}_T^t + I_{3\times 3}$$

où le paramètre $s \in [0,\infty)$ règle l'importance de l'anisotropie. Pour les deux termes, la normale utilisée est celle de la surface cible $\mathcal{T}$, $\mathbf{n}_T$ étant la normale au triangle T et $\mathbf{n}_j$ la normale à $\mathcal{T}$ au point d'échantillonnage $\tilde{\mathbf{y}}_j$, sans interpolation. Avec l'augmentation de s, la distance entre deux points dans la direction tangente à $\mathcal{T}$ devient négligeable devant la distance dans la direction de la normale à $\mathcal{T}$. Le gradient de la nouvelle fonction objectif obtenue est calculé en suivant la méthode décrite en Section 4.3.3. Dans tous les exemples présentés dans la suite, cette anisotropie a été utilisée. Un exemple simple de son effet est donné sur la Figure 5.10.

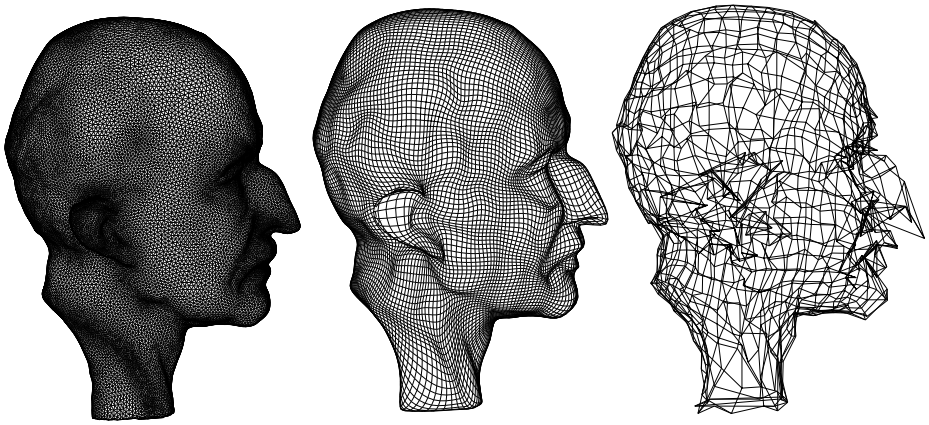


FIGURE 5.11 – Ajustement d'une surface de subdivision au modèle de Max Planck (52k sommets) à partir de la boîte englobante (6257 sommets). Le maillage de contrôle obtenu est donné à droite, et le résultat au centre. La distance de Hausdorff obtenue vaut 0.386% de la diagonale de la boîte englobante de $\mathcal{T}$.

5.5 Résultats

Pour mesurer la qualité de nos résultats, nous mesurons le rapport entre la distance de Hausdorff entre $\mathcal{S}$ et $\mathcal{T}$, et la longueur de la diagonale de la boîte englobante de $\mathcal{T}$. Cette mesure est réalisée en utilisant la méthode de Cignoni *et al.* [18].

5.5.1 Gestion des paramètres

Pour les surfaces de subdivision, utilisées ici pour tous nos ajustement des Figures 5.12 à 5.17, nous avons utilisé deux étapes de subdivision. Au niveau des paramètres, nous utilisons au départ un fort paramètre de lissage λ, de l'ordre de 1×10^{-3}, que nous faisons ensuite diminuer progressivement jusqu'à 0 au fur et à mesure de l'ajustement. Le paramètre s de l'anisotropie normale n'est activé qu'en fin d'optimisation à l'approche de la solution, et est fixé à 10 pour touts nos ajustements.

Pour initialiser $\mathcal{S}$, lorsque $\mathcal{T}$ est topologiquement équivalente à une sphère, et n'a pas de longues protubérances, nous utilisons la boîte englobante de $\mathcal{T}$. La Figure 5.19 montre cependant que cette initialisation n'est pas toujours possible. À part lorsque le temps d'exécution est mentionné, nos résultats ont tous été obtenus en moins de cinq minutes. La Figure 5.11 donne une première idée des résultats obtenus. Nous ne montrerons le maillage de contrôle que pour cette figure, dans les autres cas, nous ne montrons que la version subdivisée de $\mathcal{S}$. Les Figures 5.12 et 5.13 montrent que même lorsque les triangles sont très irréguliers et que $\mathcal{T}$ comporte des arêtes franches, le résultat reste satisfaisant.

5.5.2 Initialisation automatique

Nous avons évoqué en Section 5.3.2 un certain nombre de méthodes proposant la génération automatique d'une surface initiale $\mathcal{S}$. La Figure 5.14 propose in ajustement initialisé par la méthode de Yao *et al.* [119]. Cette méthode n'est pas complètement automatique, mais reste indépendante de la qualité de $\mathcal{S}$.

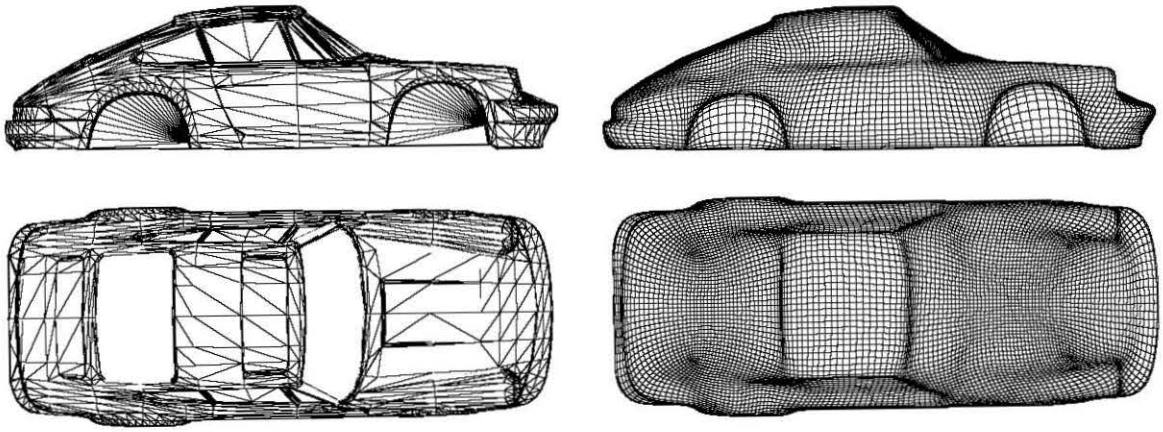


FIGURE 5.12 – Ajustement d'un modèle CAO à partir de la boîte englobante. La particularité de ce type modèle est que la triangulation a été générée uniquement dans le but de l'affichage de l'objet, et la qualité des triangles est très mauvaise. Ici la distance de Hausdorff mesurée vaut 0.554% de la diagonale de la boîte englobante de $\mathcal{T}$.

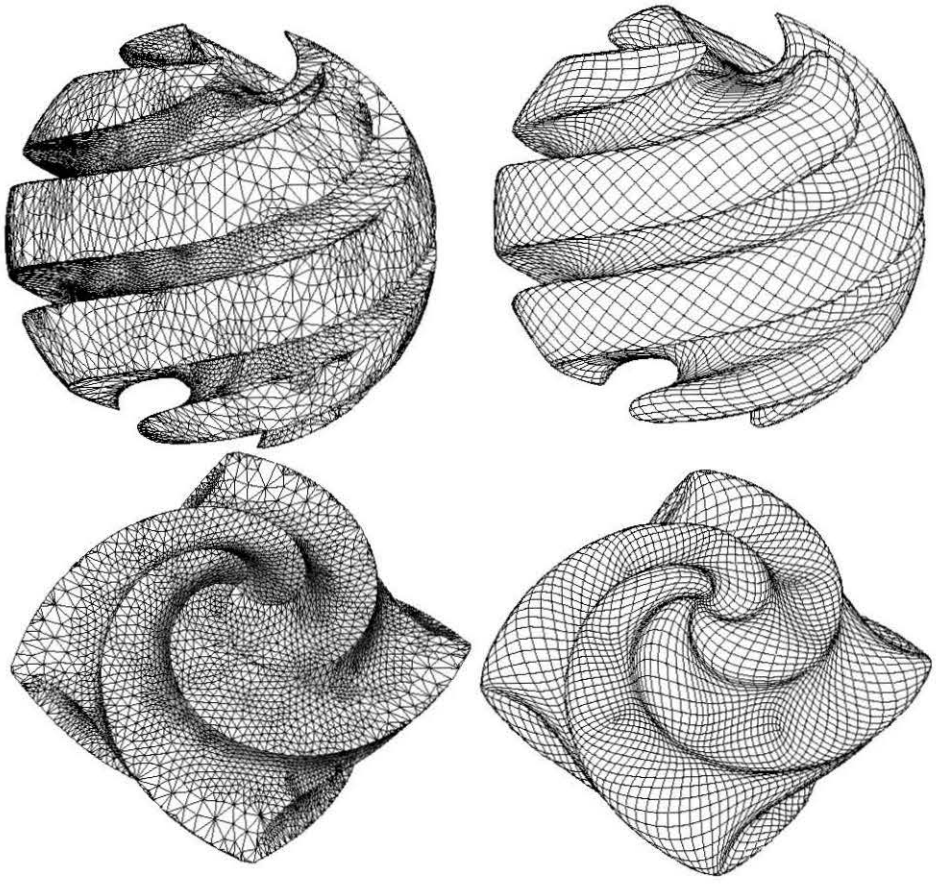


FIGURE 5.13 – Ajustement de modèle comportant des arêtes franches. Ces deux ajustements sont réalisés à partir de la boîte englobante (386 sommets). Les rapports des distances de Hausdorff à la diagonale de la boîte englobante valent respectivement 1.072% pour la sphère creusée (10k sommets) et 0.706% pour l'octaèdre pivoté.

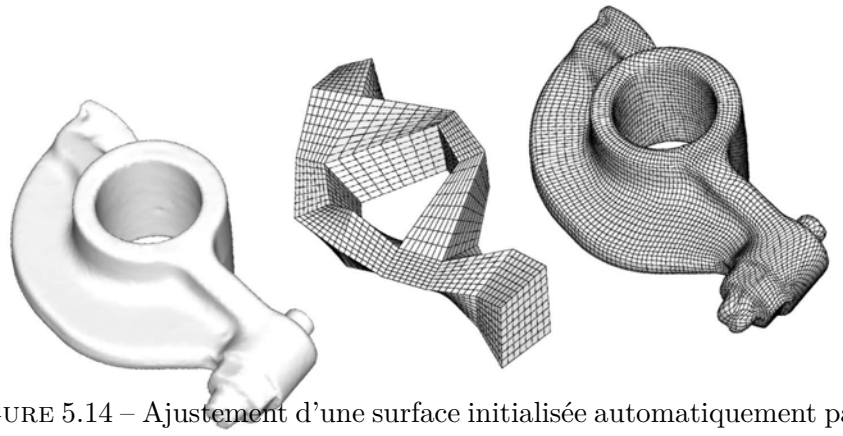


FIGURE 5.14 – Ajustement d'une surface initialisée automatiquement par la méthode de Yao *et al.* [119]. Les données nous ont été fournies par les auteurs. Le modèle initial comporte 20k sommets, le maillage de contrôle en a 4k, et le rapport distance/diagonale obtenu est de 0.44%.

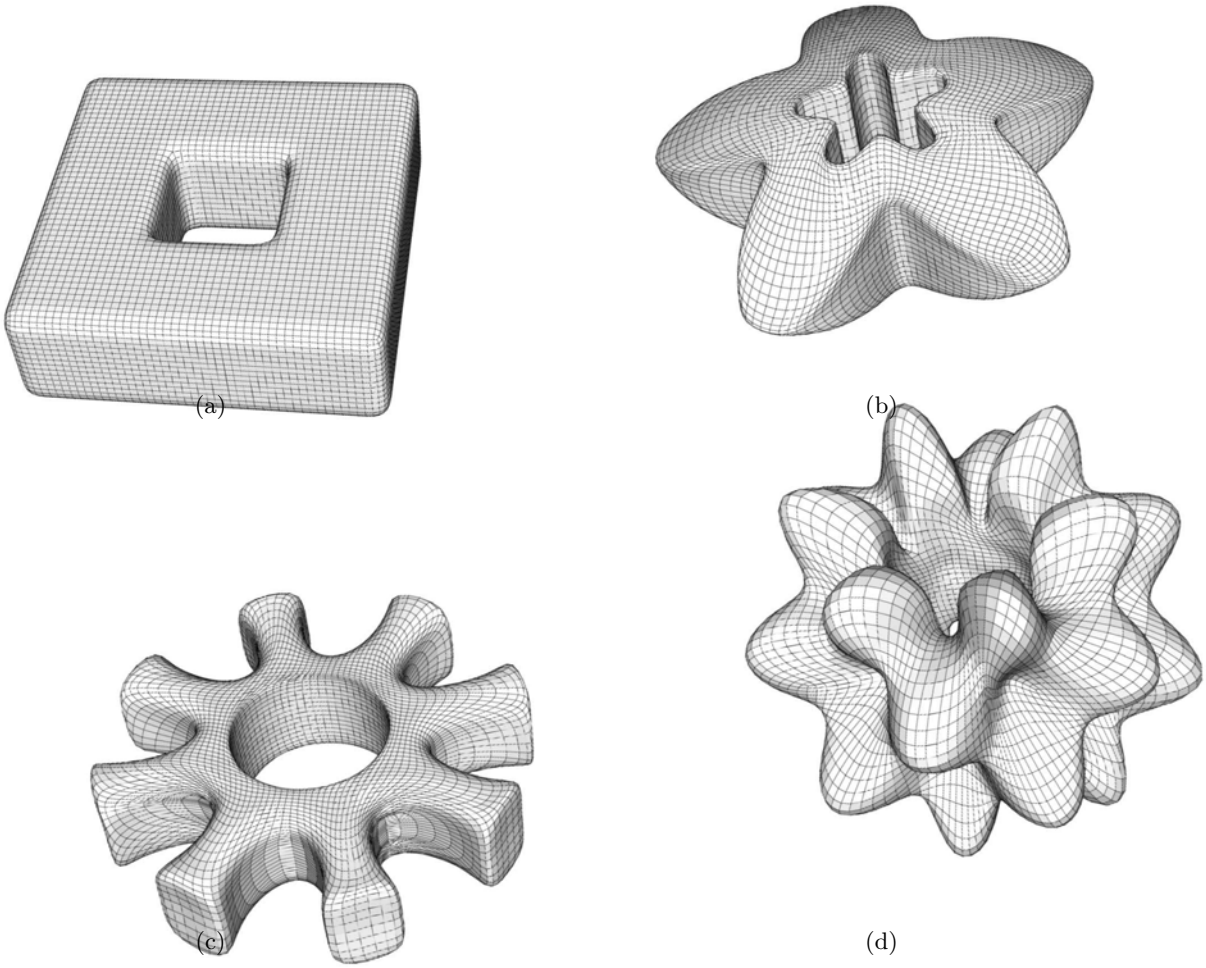


FIGURE 5.15 – Robustesse à la différence entre l'initialisation et la cible. Le maillage initial (a) – 512 sommets de contrôle – est tout d'abord ajusté à une première surface pour donner le maillage (b). Le résultat de cet ajustement est ajusté à une nouvelle surface pour donner le maillage (c), puis de nouveau à une dernière surface pour donner le maillage (d).

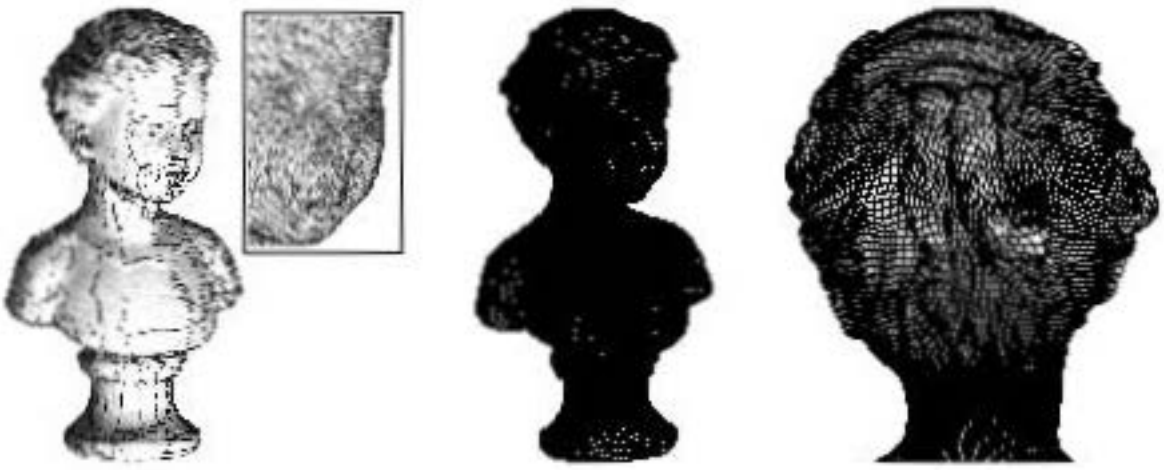


FIGURE 5.16 – Ajustement d'un ensemble de vues brutes obtenues par un scanner 3D, pour un total de 320k sommets et 630k faces. Chaque vue est un maillage de la portion de l'objet visible au moment de la prise de vue. Toutes les vues sont ici simplement alignées, et aucun traitement supplémentaire n'a été réalisé. La connectivité des triangles est donc partielle, et les différentes vues se recouvrent voire s'intersectent. L'initialisation est réalisée par la boîte englobante, et nous obtenons un rapport distance/diagonale de 1.43%. Ce résultat a été généré en 7 minutes.

5.5.3 Robustesse

Pour tester la robustesse de notre approche lorsque $\mathcal{S}$ sont de formes relativement éloignées, à partir d'une surface $\mathcal{S}$ torique, nous avons utilisé trois maillages différents, auxquels nous avons ajusté successivement $\mathcal{S}$, l'ajustement du second maillage étant donc réalisé à partir du résultat du premier ajustement (Figure 5.15).

Nous avons également testé notre méthode à l'ajustement pour des surfaces $\mathcal{T}$ de très mauvaise qualité. Les Figures 5.16 et 5.17 montrent ainsi le résultat obtenu pour l'ajustement de données brutes obtenues par un scanner 3D. Ces données sont issues du dépôt AIM@SHAPE. Ces données se présentent sous la forme de plusieurs maillages partiels, correspondant à différentes vues de l'objet scanné. Les paramètres permettant l'alignement de ces différentes vues sont obtenus à l'aide de la position et de l'orientation du scanner. Nous utilisons uniquement ces données de recalage pour amalgamer les différentes vues ensembles en une soupe de portions de maillage, aucune reconstruction ou réparation supplémentaire n'est effectuée. Notre méthode nous permet d'obtenir une surface propre de l'objet scanné. Enfin nous avons construit un exemple synthétique présenté en Figure 5.18 pour dans lequel la cible est une soupe de triangles perturbés. Le résultat de l'ajustement permet de fournir un résultat proche de la surface initiale avant la perturbation.

5.5.4 Cas d'échec

Notre approche peut échouer dans un certain nombre de cas. Tout d'abord, nous ne modifions pas la combinatoire de la surface $\mathcal{S}$ ajustée, et ne pouvons donc pas ajouter d'anses lorsque $\mathcal{S}$ en est dépourvue. L'ajustement sera donc mauvais lorsque les deux surfaces $\mathcal{S}$ et $\mathcal{T}$ sont de genres différents, même si notre algorithme ne subira pas d'erreur fatale. De plus, notre méthode sera handicapée lorsque

$\mathcal{S}$ est trop différente de $\mathcal{T}$: la combinatoire de $\mathcal{S}$ reste inchangée au cours de l'optimi-

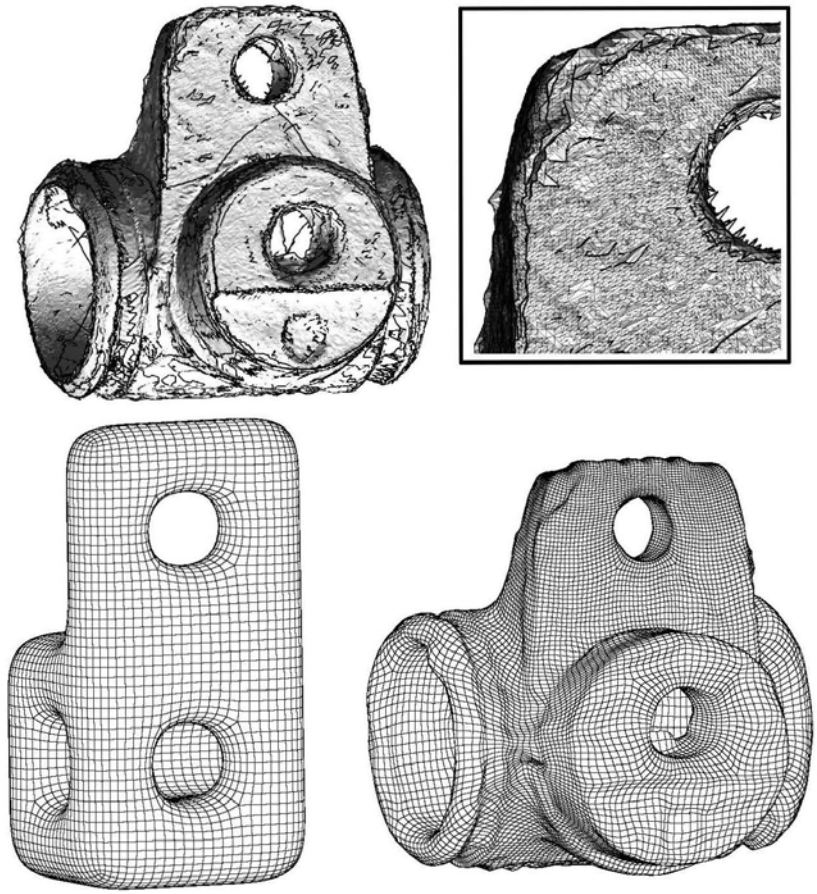


FIGURE 5.17 – Ajustement d'un ensemble de vues brutes obtenues par un scanner 3D, pour un total de 750k sommets et 1.43M faces. Le rapport distance/diagonale obtenu vaut 7.66% pour 15 minutes d'optimisation. Cette forte valeur est due au fait que des données manquent à l'intérieur de l'objet, correspondant à une portion invisible au scanner.

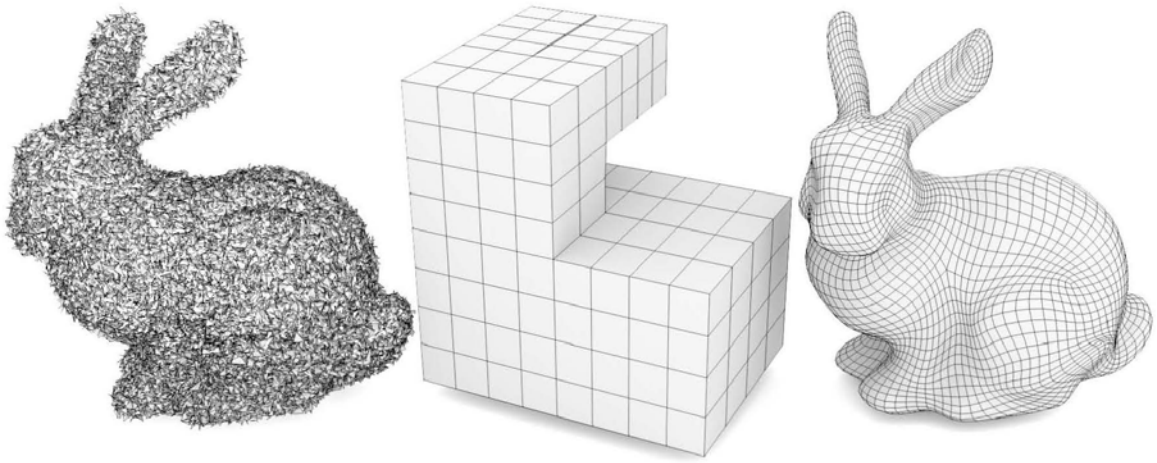


FIGURE 5.18 – Ajustement à un maillage dont la connectivité des triangles a été retirée, et dont les sommets ont subi une petite perturbation. À partir d'une surface initiale créée manuellement, il est possible d'obtenir un résultat dont le rapport distance/diagonale par rapport à la cible avant la perturbation vaut 1.447%.

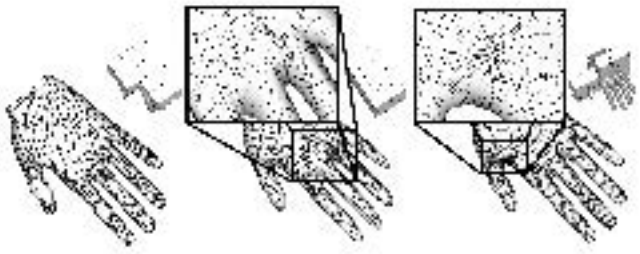


FIGURE 5.19 – Échec de l'ajustement en cas de mauvaise initialisation. Dans de tels cas, le résultat peut comportement des pincements ou une densité localement excessive.

FIGURE 5.20 – Échec de l'ajustement en cas de détails fins. Ici certaines portions de $\mathcal{T}$ sont approximées par des portions de $\mathcal{S}$ ayant une orientation inverse.

sation, si bien que dans ce cas, notre algorithme étirera ou pincera la surface pour compenser cette initialisation (Figure 5.19) ;

$\mathcal{S}$ est mal positionnée par rapport à $\mathcal{T}$: le résultat de notre approche est un minimum local de la fonction objectif. Le minimum atteint dépend des positions relatives de $\mathcal{S}$ et $\mathcal{T}$. L'ajustement automatique de l'orientation de $\mathcal{S}$ à $\mathcal{T}$ est un problème qui n'a pour l'instant pas de solution simple et rapide. Dans ce cas également, le minimum local atteint peut amener à un fort étirement ou à des pincements (Figure 5.19) ;

$\mathcal{T}$ comporte des parties fines : même lorsque l'anisotropie normale est utilisée, seule la direction de la normal est utilisée, et non son orientation. Dans les parties fines de $\mathcal{T}$, plusieurs couches peuvent donc être approximées par un seul feuillet de $\mathcal{S}$, parfois d'orientation opposée (Figure 5.20).

5.5.5 Performances

La Figure 5.21 donne un exemple des performances de notre méthode. Expérimentalement, le temps d'exécution semble évoluer linéairement par rapport au nombre de sommets, voire au pire avoir une complexité de $O(n \log n)$ en fonction du nombre n de sommets de $\mathcal{S}$. La distance de Hausdorff décroît très rapidement avec le nombre de sommets, même si cette décroissance n'est pas stricte.

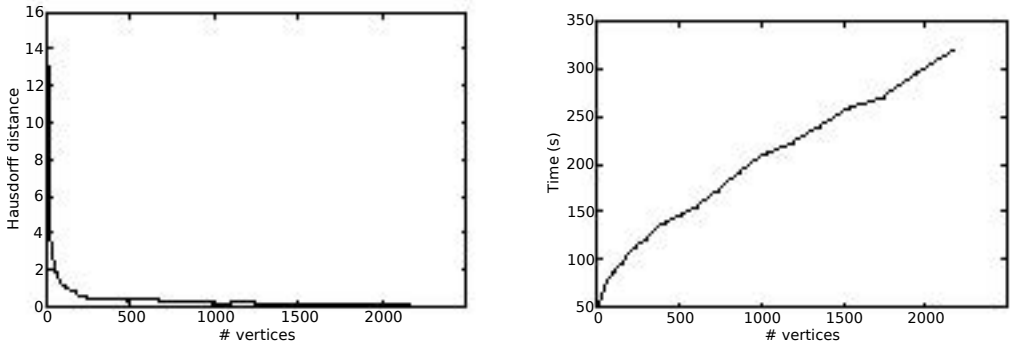


FIGURE 5.21 – Performance de notre procédé d'ajustement de surface. À gauche, l'évolution de la distance de Hausdorff entre $\mathcal{S}$ et $\mathcal{T}$ en fonction du nombre de sommets de $\mathcal{S}$. À gauche, le temps d'exécution nécessaire, également en fonction du nombre de sommets de $\mathcal{S}$. Le modèle utilisé pour ces tests est celui de l'octaèdre pivoté de la Figure 5.13.

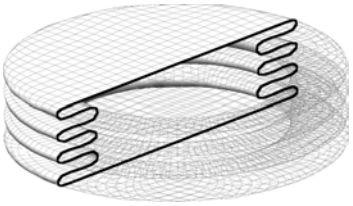


FIGURE 5.22 – Premier résultat d'une fonction objectif modifiée mesurant l'écart de normale entre un échantillon et les points de sa cellule restreinte. La fonction objectif classique échoue dans ce cas, comme le montre la Figure 5.19.

5.6 Conclusion et perspectives

5.6.1 Contributions

La méthode que nous avons présentée dans ce chapitre offre les avantages suivants :

robustesse : notre méthode peut être appliquée dans des situations où d'autres méthodes faisant appel à des outils plus complexes tels qu'une paramétrisation de la surface cible ou l'approximation de sa courbure ne sont pas envisageable, tout en proposant des résultat de qualité comparable. Les seules conditions nécessaires sur la surface $\mathcal{T}$ est qu'elle soit constituée d'un ensemble de triangles ;

fondements : nous avons montré que notre fonction objectif est une bonne approximation de la distance au carré entre deux surfaces. Par rapport aux autres approximations de cette distance existant dans la littérature, notre approximation peut être minimisée au moins aussi rapidement, sans faire appel à la courbure de $\mathcal{T}$;

gestion des surfaces de subdivision : notre approche permet de gérer les surfaces de subdivision de manière naturelle, sans rendre plus complexe le procédé d'optimisation.

Dans les circonstances où la qualité de $\mathcal{T}$ le permet, notre approche ne sera pas forcément la plus appropriée pour obtenir un remaillage quadrangulaire, en particulier parce que nous ne fournissons pas de procédé automatique pour initialiser $\mathcal{S}$. Il reste toutefois envisageable d'utiliser notre méthode pour ajuster une surface de subdivision contrôlée par un maillage quadrangulaire grossier obtenu par une autre méthode.

5.6.2 Perspectives

L'un des cas posant problème à notre méthode est celui des surfaces ayant des parties fines. Comme montré en Figure 5.20, notre fonction objectif ne rend pas en compte l'alignement des normales, si bien qu'il est possible de d'ajuster à une portion de $\mathcal{T}$ une portion de $\mathcal{S}$ d'orientation opposée. Pour s'attaquer à ce problème, notre idée consisterait à ajouter un terme à la fonction objectif pour faire en sorte qu'au sein d'une cellule de Voronoï restreinte, les points de la portion de surface aient une normale similaire à celle de l'échantillon correspondant à la cellule. Nos premiers résultats en appliquant cette idée semblent prometteur, et permettent d'obtenir un résultat satisfaisant au cas problématique de la Figure 5.20, donné en Figure 5.22.

À plus long terme, nous envisageons d'étudier le problème de la modification de la combinatoire du maillage ajusté au cours de l'optimisation, pour pouvoir percer des anses, ou contracter des portions inutiles au niveau des pincements. Des opérations existent déjà pour la modification de maillages quadrangulaires, le problème majeur consiste plutôt à détecter de manière robuste les portions de $\mathcal{S}$ pouvant bénéficier d'une modification du maillage, et de déterminer les opérations à effectuer en fonction de l'anomalie détectée.

Conclusion

Bilan des contributions

Nous avons abordé dans cette thèse deux approches différentes pour l'échantillonnage de surfaces. La première, décrite dans la Partie I, repose sur une paramétrisation de la surface. La seconde, détaillée en Partie II, utilise un diagramme de Voronoï restreint. Nous résumons ici les principales contributions de cette thèse.

Une méthode pour le placage de textures sans coutures qui propose de construire un atlas de texture spécifique pour s'assurer de bonnes conditions de recollement d'une texture au niveau des coutures lors du placage. Cette méthode sera adaptée si l'utilisateur ne souhaite pas modifier le moteur de rendu utilisé pour réaliser le placage de texture, ou si les accès à la texture doivent rester le plus rapide possible. Lorsque par contre l'utilisateur souhaite préserver les modèles et textures déjà créés, il sera préférable d'utiliser une autre méthode de l'état de l'art réalisant la correction sur la carte graphique au moment de l'accès à la texture. Ces travaux sont décrits dans le Chapitre 2 et sont publiés dans [93].

Un algorithme pour le calcul d'un diagramme de Voronoï restreint qui reste efficace à la fois lorsqu'une cellule de Voronoï restreinte couvre de nombreuses faces de la surface et lorsqu'une face est segmentée en de multiples cellules de Voronoï restreintes. Nous proposons également une variante de cet algorithme permettant de fournir les différentes composantes connexes d'une cellule de Voronoï restreinte. Ces travaux sont décrits dans le Chapitre 3 et ne sont pour l'instant pas publiés ;

Une étude d'une classe de fonctions objectif définies sur un diagramme de Voronoï restreint et permettant de mesurer la qualité d'un échantillonnage d'une surface pour l'approximation d'une fonction quelconque définie sur cette surface. Nous fournissons également un procédé d'optimisation d'un échantillonnage pour minimiser une telle fonction objectif. Pour réaliser cette optimisation il n'est nécessaire que de pouvoir évaluer la valeur de la fonction approximée en tout point de la surface. Cette méthode peut en particulier s'avérer utile lorsqu'il n'est pas envisageable de paramétrer la surface ou si l'espace de stockage disponible est très réduit. Ces travaux sont décrits dans le Chapitre 4 et ne sont pour l'instant pas publiés.

Une méthode d'ajustement de surfaces minimisant une fonction objectif calculée sur des diagrammes de Voronoï restreints. Nous prouvons que la fonction objectif minimisée est une bonne approximation de la distance carrée entre deux surfaces, et fournissons un mécanisme pour ajuster des surfaces de subdivision. Cette méthode est particulièrement adaptée à l'ajustement de ce type de représentation car elle ne suppose pas de correspondance a priori entre les deux surfaces. Elle permet également de traiter des surfaces de très mauvaise qualité, en particulier du fait qu'elle n'a pas besoin d'une estimation de la courbure des surfaces fournies. Ces travaux sont décrits dans le Chapitre 5 et publiés dans [74].

Les deux approches que nous avons proposées sont complémentaires. En effet, l'approche par le placage de texture offre une structure de grille entre les échantillons, qui permet de définir des splines. Les splines permettent alors à partir des échantillons de reconstruire une fonction ayant un degré de continuité aussi haut que désiré. La structure de grille sous-jacente est issue d'une paramétrisation préservant la grille $\mathbb{Z}^2$ qui peut en revanche s'avérer difficile à calculer en particulier lorsque la surface donnée est de mauvaise qualité. La seconde approche approxime une fonction par un diagramme de Voronoï restreint en assignant une valeur à chaque cellule de Voronoï restreinte. L'absence de structure entre les échantillons ne permet pas de reconstruire de manière triviale des fonctions continues à partir des valeurs des échantillons, et des discontinuités apparaissent aux frontières des cellules de Voronoï restreintes. En revanche, cette méthode ne requiert de la surface que d'être constituée de triangles et s'applique donc sur tout maillage quelle que soit s qualité. La méthode d'ajustement de surface que nous avons proposée réalise un pont entre ces deux approches en ajustant à une surface donnée un échantillonnage muni d'une structure de maillage. Il nous est alors possible d'approximer la surface cible par une surface de subdivision (une surface spline) déduite à la fois des positions des échantillons et de la structure de maillage qui les relie.

Publications associées à cette thèse

[74] V. Nivoliers, D.-M. Yan et B. Lévy : Fitting Polynomial Surfaces to Triangular Meshes with Voronoi Squared Distante Minimization. *Engineering with Computers*, 2012. À paraître. 17, 185

[93] N. Ray, V. Nivoliers, S. Lefebvre et B. Lévy : Invisible seams. *In Computer Graphics Forum*, vol. 29, p. 1489–1496. Wiley Online Library, 2010. 17, 49, 185

Perspectives

Une première piste pour poursuivre nos travaux consiste à aborder le problème de la génération de maillages multirésolution. De tels maillages sont très utilisés car ils permettent de représenter de manière compacte des maillages très fins, en se fondant sur des schémas de subdivision. Ces maillages sont en particulier utilisés par les artistes lors de la création d'un modèle, conjointement à un outil de sculpture virtuel. Ce mode d'édition de maillages est relativement récent, et a été rendu possible grâce à la rapidité des cartes graphiques et aux nouveaux périphériques de type tablette graphique. Ce procédé d'édition fait appel à de nombreux outils différents de génération de maillage en lien avec nos travaux. Tout d'abord, la forme générale de la surface peut être amenée à évoluer fortement au cours de l'édition, ce qui requiert au maillage de s'adapter et de rester propre. David Lopez étudie actuellement durant sa thèse un procédé de suivi de surfaces fondé sur les diagrammes de Voronoï barycentriques. La méthode que nous avons présentée en Chapitre 5 pourrait ici être modifiée pour convertir la surface ainsi obtenue en une surface multirésolution. Il faudrait à cette fin modifier notre algorithme pour pouvoir déterminer automatiquement le niveau de résolution local nécessaire à l'approximation d'un maillage, et intégrer les relations entre les variables des différents niveaux de résolution.

Une autre suite possible à nos travaux serait de les confronter au domaine des éléments finis et de la résolution d'équations aux dérivées partielles pour la simulation de phénomènes physiques. Comme nous l'avions évoqué en introduction, ce domaine est aussi demandeur

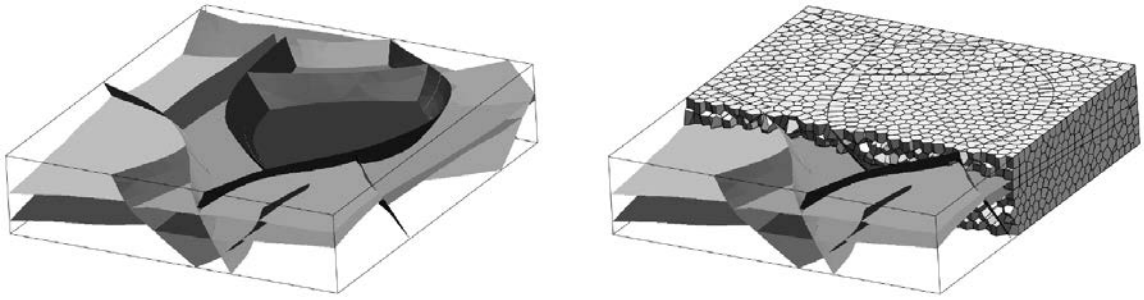


FIGURE 5.23 – Remaillage polyédral d'une portion de sol, prenant en compte un ensemble de failles et d'horizons entre les couches géologiques. Cette méthode, développée par Merland *et al.* [72] utilise le solveur décrit en Chapitre 4 pour optimiser le diagramme de Voronoï.

de maillages pour représenter les objets sur lesquelles les simulations sont effectuées. Les grandeurs physiques données ou calculées sont alors échantillonnées sur ces maillages. Dans ce cadre, nos travaux pourraient intervenir de deux manières. Tout d'abord, de nouvelles méthodes de simulation dites « mimétiques » ont été développées pour étudier certains problème physiques sur des maillages polyédraux quelconques [64], en particulier pour la simulation d'écoulements dans les sols en géologie. Merland *et al.* [72] se sont intéressés sur cette base à la génération de maillages polyédraux qui respectent la structure du sol pour ce type de simulation (Figure 5.23), en utilisant des méthodes similaires à celles présentées dans ce mémoire en Chapitre 4. Dans ce cadre, notre méthode pourrait permettre de générer un maillage polyédral optimisé pour l'approximation d'une grandeur particulière sur un domaine, telle que la porosité ou la conductivité thermique d'un matériau. D'autre part, le problème de la génération de maillages hexaèdriques pour les éléments finis reste un problème très actif en recherche, car il n'existe que très peu de méthodes complètement automatiques pour générer ces maillages. Les travaux de Lévy et Liu [62] proposent une approche fondées sur les diagrammes de Voronoï restreints pour ce problème, qui pourrait être étendue en utilisant nos travaux du Chapitre 5, avec un maillage généré progressivement par une méthode frontale. Paillé et Poulin [77] proposent également un procédé d'ajustement volumique de maillages hexaèdriques qui nécessite d'ajuster le bord du maillage ajusté avec le bord du domaine à mailler.

Bibliographie

[1] N. Amenta et M. Bern : Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry*, 22(4):481–504, 1999. 164, 167

[2] Y. Asami : A note on the derivation of the first and second derivative of objective functions in geographical optimization problems. *Journal of the Faculty of Engineering, The University of Tokyo (B)*, 41(1):1–13, 1991. 128, 138, 141, 142, 153

[3] D. Attali et J. Boissonnat : A linear bound on the complexity of the delaunay triangulation of points on polyhedral surfaces. *Discrete & Computational Geometry*, 31(3):369–384, 2004. 168

[4] F. Aurenhammer : Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991. 96

[5] C. Barber, D. Dobkin et H. Huhdanpaa : The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483, 1996. 99

[6] M. Ben-Chen, C. Gotsman et G. Bunin : Conformal flattening by curvature prescription and metric scaling. *In Computer Graphics Forum*, vol. 27, p. 449–458. Wiley Online Library, 2008. 36

[7] D. Benson et J. Davis : Octree textures. *In ACM Transactions on Graphics (TOG)*, vol. 21, p. 785–790. ACM, 2002. 60

[8] J. Blinn : Simulation of wrinkled surfaces. *In ACM SIGGRAPH Computer Graphics*, vol. 12, p. 286–292. ACM, 1978. 49

[9] D. Bommes, B. Lévy, N. Pietroni, E. Puppo, C. S. a, M. Tarini et D. Zorin : State of the art in quad meshing. *In Eurographics STARS*, 2012. 37, 158

[10] D. Bommes, H. Zimmer et L. Kobbelt : Mixed-integer quadrangulation. *In ACM Transactions on Graphics (TOG)*, vol. 28, p. 77. ACM, 2009. 40, 41, 47, 48

[11] B. Burley et D. Lacewell : Ptex : Per-face texture mapping for production rendering. *In Computer Graphics Forum*, vol. 27, p. 1155–1164. Wiley Online Library, 2008. 59, 61, 63, 65, 66, 81, 89

[12] M. Campen, D. Bommes et L. Kobbelt : Dual loops meshing : quality quad layouts on manifolds. *ACM Transactions on Graphics (TOG)*, 31(4):110, 2012. 158

[13] N. Carr et J. Hart : Painting detail. *In ACM Transactions on Graphics (TOG)*, vol. 23, p. 845–852. ACM, 2004. 62

[14] N. Carr, J. Hoberock, K. Crane et J. Hart : Rectangular multi-chart geometry images. *In Proceedings of the fourth Eurographics symposium on Geometry processing*, p. 181–190. Eurographics Association, 2006. 62, 64

[15] E. Catmull : *A Subdivision Algorithm for Computer Display of Curved Splines*. Thèse de doctorat, PhD thesis, Computer Science Department, University of Utah, Salt Lake City, UT, 1974. 49

[16] E. Catmull et J. Clark : Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350 – 355, 1978. 42, 79, 176, 177

[17] K. Cheng, W. Wang, H. Qin, K. Wong, H. Yang et Y. Liu : Fitting subdivision surfaces to unorganized point data using sdm. *In Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on*, p. 16–24. IEEE, 2004. 161

[18] P. Cignoni, C. Rocchini et R. Scopigno : Metro : measuring error on simplified surfaces. 17(2):167–174, 1998. 178

[19] D. Cohen-Steiner et J. Morvan : Restricted delaunay triangulations and normal cycle. *In Proceedings of the nineteenth annual symposium on Computational geometry*, p. 312–321. ACM, 2003. 28

[20] T. Cormen, C. Leiserson, R. Rivest et C. Stein : *Introduction to algorithms*. The MIT press, 2001. 68, 116, 119

[21] J. Cortes, S. Martinez et F. Bullo : Spatially-distributed coverage optimization and control with limited-range interactions. *ESAIM contrôle optimisation et calcul des variations*, 11:691, 2005. 128, 138, 139

[22] H. Delingette : General object reconstruction based on simplex meshes. *International Journal of Computer Vision*, 32(2):111–146, 1999. 158, 160

[23] M. Desbrun, M. Meyer et P. Alliez : Intrinsic parameterizations of surface meshes. *In Computer Graphics Forum*, vol. 21, p. 209–218. Wiley Online Library, 2002. 36

[24] M. Do Carmo : *Differential geometry of curves and surfaces*, vol. 1. Prentice-Hall Englewood Cliffs, NJ, 1976. 25, 26, 29, 38

[25] D. Doo et M. Sabin : Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10(6):356 – 360, 1978. 42, 79

[26] Q. Du et M. Emelianenko : Acceleration schemes for computing centroidal voronoi tessellations. *Numerical Linear Algebra with Applications*, 13(2-3):173–192, 2006. 129

[27] Q. Du, M. Emelianenko et L. Ju : Convergence of the lloyd algorithm for computing centroidal voronoi tessellations. *SIAM journal on numerical analysis*, p. 102–119, 2006. 126

[28] Q. Du, V. Faber et M. Gunzburger : Centroidal voronoi tessellations : Applications and algorithms. *SIAM review*, p. 637–676, 1999. 123, 125, 126

[29] Q. Du et D. Wang : Anisotropic centroidal voronoi tessellations and their applications. *SIAM Journal on Scientific Computing*, 26(3):737–761, 2005. 133, 137

[30] D. Dunavant : High degree efficient symmetrical gaussian quadrature rules for the triangle. *International journal for numerical methods in engineering*, 21(6):1129–1148, 1985. 144, 147

[31] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery et W. Stuetzle : Multiresolution analysis of arbitrary meshes. *In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, p. 173–182. ACM, 1995. 36

[32] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery et W. Stuetzle : Multiresolution analysis of arbitrary meshes. *In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, p. 173–182. ACM, 1995. 62, 72, 100

[33] M. Eck et H. Hoppe : Automatic reconstruction of b-spline surfaces of arbitrary topological type. *In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, p. 325–334. ACM, 1996. 158, 159, 160

[34] I. Eckstein, J. Pons, Y. Tong, C. Kuo et M. Desbrun : Generalized surface flows for mesh processing. *In Proceedings of the fifth Eurographics symposium on Geometry processing*, p. 183–192. Eurographics Association, 2007. 160

[35] H. Edelsbrunner et E. Mücke : Simulation of simplicity : a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics (TOG)*, 9 (1):66–104, 1990. 121

[36] H. Edelsbrunner et N. Shah : Triangulating topological spaces. *International Journal of Computational Geometry and Applications*, 7(4):365–378, 1997. 110, 130

[37] J. Erickson et S. Har-Peled : Optimally cutting a surface into a disk. *Discrete & Computational Geometry*, 31(1):37–59, 2004. 53

[38] H. Flanders : Differentiation under the integral sign. *The American Mathematical Monthly*, 80(6):615–627, 1973. 138

[39] M. Floater : Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997. 158, 160

[40] M. Floater : Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003. 36

[41] M. Floater et K. Hormann : Surface parameterization : a tutorial and survey. *Advances in multiresolution for geometric modelling*, p. 157–186, 2005. 37

[42] S. Fortune : Voronoi diagrams and delaunay triangulations. *Computing in Euclidean geometry*, 1(193-233):2, 1992. 93, 99

[43] M. Garland et P. Heckbert : Surface simplification using quadric error metrics. *In Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, p. 209–216. ACM Press/Addison-Wesley Publishing Co., 1997. 158

[44] P. George et H. Borouchaki : *Delaunay triangulation and meshing : application to finite elements*. 1998. 96

[45] F. González et G. Patow : Continuity mapping for multi-chart textures. *In ACM Transactions on Graphics (TOG)*, vol. 28, p. 109. ACM, 2009. 62, 65

[46] X. Gu et S. Yau : Global conformal surface parameterization. *In Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, p. 127–137. Eurographics Association, 2003. 36, 40

[47] F. Hermeline : Triangulation automatique d'un poly edre en dimension n. *RAIRO numerical analysis*, 16(3), 1982. 99

[48] A. Hertzmann et D. Zorin : Illustrating smooth surfaces. *In Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, p. 517–526. ACM Press/Addison-Wesley Publishing Co., 2000. 39, 40, 41

[49] K. Hoff III, J. Keyser, M. Lin, D. Manocha et T. Culver : Fast computation of generalized voronoi diagrams using graphics hardware. *In Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, p. 277–286. ACM Press/Addison-Wesley Publishing Co., 1999. 97

[50] K. Hormann et G. Greiner : Mips : An efficient global parametrization method. *Curve and Surface Design : Saint-Malo 99*, p. 153, 2000. 36

[51] K. Hormann, B. Lévy, A. Sheffer *et al.* : Mesh parameterization : Theory and practice. *SIGGRAPH Course Notes*, 2007. 34, 37, 52

[52] M. Iri, K. Murota et T. Ohya : A fast voronoi-diagram algorithm with applications to geographical optimization problems. *System Modelling and Optimization*, p. 273–288, 1984. 128, 138

[53] F. Kälberer, M. Nieser et K. Polthier : Quadcover-surface parameterization using branched coverings. *In Computer Graphics Forum*, vol. 26, p. 375–384. Wiley Online Library, 2007. 41, 47, 48

[54] L. Kharevych, B. Springborn et P. Schröder : Discrete conformal mappings via circle patterns. *ACM Transactions on Graphics (TOG)*, 25(2):412–438, 2006. 36

[55] L. Kobbelt, J. Vorsatz et U. Labsik : A shrink wrapping approach to remeshing polygonal surfaces. *Computer Graphics Forum*, 18(3):119–130, 1999. 158, 159

[56] R. Kunze, F. Wolter et T. Rausch : Geodesic voronoi diagrams on parametric surfaces. *In Computer Graphics International, 1997. Proceedings*, p. 230–237. IEEE, 1997. 101

[57] F. Labelle et J. Shewchuk : Anisotropic voronoi diagrams and guaranteed-quality anisotropic mesh generation. *In Annual Symposium on Computational Geometry : Proceedings of the nineteenth annual symposium on Computational geometry*, vol. 8, p. 191–200, 2003. 133

[58] J. Lasserre et K. Avrachenkov : The multi-dimensional version of $\int_a^b x^p dx$. *The American Mathematical Monthly*, 108(2):151–154, 2001. 134

[59] S. Lefebvre et C. Dachsbacher : Tiletrees. *In Proceedings of the 2007 symposium on Interactive 3D graphics and games*, p. 25–31. ACM, 2007. 60

[60] S. Lefebvre et H. Hoppe : Appearance-space texture synthesis. *In ACM Transactions on Graphics (TOG)*, vol. 25, p. 541–548. ACM, 2006. 73

[61] M. Levoy, S. Rusinkiewicz, B. Curless, M. Ginzton, J. Ginsberg, K. Pulli, D. Koller, S. Anderson, J. Shade, L. Pereira, J. Davis et D. Fulk : The digital michelangelo project : 3d scanning of large statues. p. 131–144, 2000. 13

[62] B. Lévy et Y. Liu : Lp centroidal voronoi tesselation and its applications. *ACM Transactions on Graphics*, 29:4, 2010. 130, 132, 133, 134, 137, 153, 171, 177, 187

[63] B. Lévy, S. Petitjean, N. Ray, J. Maillot *et al.* : Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics*, 21(3):362–371, 2002. 36, 53, 72

[64] K. Lipnikov, M. Shashkov et D. Svyatskiy : The mimetic finite difference discretization of diffusion problem on unstructured polyhedral meshes. *Journal of Computational Physics*, 211(2):473–491, 2006. 187

[65] D. Liu et J. Nocedal : On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989. 130, 171

[66] Y. Liu, Z. Chen et K. Tang : Construction of iso-contours, bisectors and voronoi diagrams on triangulated surfaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (99):1–1, 2011. 101

[67] Y. Liu, W. Wang, B. Lévy, F. Sun, D. Yan, L. Lu et C. Yang : On centroidal voronoi tessellation – energy smoothness and fast computation. *ACM Transactions on Graphics (ToG)*, 28(4):101, 2009. 128, 130, 131, 132

[68] S. Lloyd : Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. 126

[69] C. Loop : *Smooth Subdivision Surfaces Based on Triangles*. Thèse de doctorat, University of Utah, 1987. 176

[70] W. Ma, X. Ma, S. Tso et Z. Pan : A direct approach for subdivision surface fitting from a dense triangle mesh. *Computer-Aided Design*, 36(6):525–536, 2004. 158, 159, 160

[71] Q. Mérigot, M. Ovsjanikov et L. Guibas : Voronoi-based curvature and feature estimation from point clouds. *Visualization and Computer Graphics, IEEE Transactions on*, 17(6):743–756, 2011. 28

[72] R. Merland, B. Levy et G. Caumon : Voronoi grids conformal to 3d structural features. *In 13th European Conference on the Mathematics of Oil Recovery (ECMOR)*, Biarritz, France, September 2012. 187

[73] T. Minka : Old and new matrix algebra useful for statistics. 1997. 127, 135, 173

[74] V. Nivoliers, D.-M. Yan et B. Lévy : Fitting Polynomial Surfaces to Triangular Meshes with Voronoi Squared Distante Minimization. *Engineering with Computers*, 2012. À paraître. 17, 185

[75] J. Nocedal et S. Wright : *Numerical optimization*. Springer verlag, 1999. 126, 129

[76] A. Okabe, B. Boots, K. Sugihara et S. Chiu : *Spatial tessellations : concepts and applications of Voronoi diagrams*. Wiley & Sons Chichester., 1992. 93, 99

[77] G.-P. Paillé et P. Poulin : As-conformal-as-possible discrete volumetric mapping. *Computers & Graphics*, 36(5):427–433, 2012. 187

[78] J. Palacios et E. Zhang : Rotational symmetry field design on surfaces. *In ACM Transactions on Graphics (TOG)*, vol. 26, p. 55. ACM, 2007. 39, 40

[79] D. Panozzo, E. Puppo, M. Tarini, N. Pietroni et P. Cignoni : Automatic construction of quad-based subdivision surfaces using fitmaps. *IEEE Transactions on Visualization and Computer Graphics*, 17(10):1510–1520, 2011. 158

[80] S. Petitjean : A survey of methods for recovering quadrics in triangle meshes. *ACM Computing Surveys (CSUR)*, 34(2):211–262, 2002. 28

[81] G. Peyré et L. Cohen : Geodesic remeshing using front propagation. *International Journal of Computer Vision*, 69(1):145–156, 2006. 101

[82] M. Pharr et R. Fernando : *Gpu gems 2 : programming techniques for high-performance graphics and general-purpose computation*. Addison-Wesley Professional, first édn, 2005. ISBN 9780321545411. 57

[83] U. Pinkall et K. Polthier : Computing discrete minimal surfaces and their conjugates. *Experimental mathematics*, 2(1):15–36, 1993. 36

[84] S. Pion : *De la géométrie algorithmique au calcul géométrique*. Thèse de doctorat, Université de Nice-Sophia Antipolis, France, 1999. Thèse de doctorat. 121

[85] D. Piponi et G. Borshukov : Seamless texture mapping of subdivision surfaces by model pelting and texture blending. *In Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, p. 471–478. ACM Press/Addison-Wesley Publishing Co., 2000. 62

[86] H. Pottmann et M. Hofer : Geometry of the squared distance function to curves and surfaces. *Visualization and Mathematics III, Springer*, p. 221–242, 2003. 162, 163

[87] H. Pottmann et S. Leopoldseder : A concept for parametric surface fitting which avoids the parametrization problem. *Computer Aided Geometric Design*, 20(6):343–362, 2003. 160, 161

[88] E. Praun, A. Finkelstein et H. Hoppe : Lapped textures. *In Proceedings of SIGGRAPH 2000*, p. 465–470. New York, NY, USA, 2000. 39, 40

[89] H. Prautzsch, W. Boehm et M. Paluszny : *Bézier and B-spline techniques*. Springer Verlag, 2002. 54

[90] B. Purnomo, J. Cohen et S. Kumar : Seamless texture atlases. *In Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, p. 65–74. ACM, 2004. 63, 86, 89

[91] A. Rappoport : An efficient algorithm for line and polygon clipping. *The Visual Computer*, 7(1):19–28, 1991. 102

[92] N. Ray, W. Li, B. Lévy, A. Sheffer et P. Alliez : Periodic global parameterization. *ACM Transactions on Graphics (TOG)*, 25(4):1460–1485, 2006. 47

[93] N. Ray, V. Nivoliers, S. Lefebvre et B. Lévy : Invisible seams. *In Computer Graphics Forum*, vol. 29, p. 1489–1496. Wiley Online Library, 2010. 17, 49, 185

[94] N. Ray, B. Vallet, L. Alonso et B. Lévy : Geometry-aware direction field processing. *ACM Transactions on Graphics (TOG)*, 29(1):1, 2009. 40, 41, 48

[95] N. Ray, B. Vallet, W. Li et B. Lévy : N-symmetry direction field design. *ACM Transactions on Graphics (TOG)*, 27(2):10, 2008. 40, 41

[96] J.-F. Remacle, J. Lambrechts, B. Seny, E. Marchandise, A. Johnen et C. Geuzaine : 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, 2012. 158

[97] O. Reynolds : *Papers on Mechanical and Physical Subjects*, vol. 3. The University Press, 1903. 138

[98] S. Rusinkiewicz et M. Levoy : Efficient variants of the icp algorithm. *In 3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, p. 145–152. IEEE, 2001. 160

[99] J. Schreiner, A. Asirvatham, E. Praun et H. Hoppe : Inter-surface mapping. *ACM Transactions on Graphics (TOG)*, 23(3):870–877, 2004. 159

[100] W. Schroeder, J. Zarge et W. Lorensen : Decimation of triangle meshes. *In ACM SIGGRAPH Computer Graphics*, vol. 26, p. 65–70. ACM, 1992. 158

[101] A. Sheffer et E. de Sturler : Parameterization of faceted surfaces for meshing using angle-based flattening. *Engineering with Computers*, 17(3):326–337, 2001. 36

[102] A. Sheffer et J. Hart : Seamster : inconspicuous low-distortion texture seam layout. *In Visualization, 2002. VIS 2002. IEEE*, p. 291–298. IEEE, 2002. 61

[103] A. Sheffer, E. Praun et K. Rose : Mesh parameterization methods and their applications. *Foundations and Trends in Computer Graphics and Vision*, 2(2):105–171, 2006. 37

[104] J. Shewchuk : What is a good linear element ? interpolation, conditioning, and quality measures. *In International Meshing Roundtable conference proceedings*, p. 115–126, 2002. 131

[105] A. Sud, N. Govindaraju, R. Gayle, I. Kabul et D. Manocha : Fast proximity computation among deformable models using discrete voronoi diagrams. *In ACM Transactions on Graphics (TOG)*, vol. 25, p. 1144–1153. ACM, 2006. 97

[106] I. Sutherland et G. Hodgman : Reentrant polygon clipping. *Communications of the ACM*, 17(1):32–42, 1974. 102, 103

[107] M. Tarini, K. Hormann, P. Cignoni et C. Montani : Polycube-maps. *ACM Transactions on Graphics (TOG)*, 23(3):853–860, 2004. 60, 61, 159, 160

[108] M. Tarini, N. Pietroni, P. Cignoni, D. Panozzo et E. Puppo : Practical quad mesh simplification. *Computer Graphics Forum*, 29(2):407–418, 2010. 158

[109] Cgal : Computational Geometry Algorithms Library. http ://www.cgal.org. 99

[110] Y. Tong, P. Alliez, D. Cohen-Steiner et M. Desbrun : Designing quadrangulations with discrete harmonic forms. *In Proceedings of the fourth Eurographics symposium on Geometry processing*, p. 201–210. Eurographics Association, 2006. 47

[111] W. Tutte : How to draw a graph. *Proc. London Math. Soc*, 13(3):743–768, 1963. 33

[112] S. Valette, J. Chassery et R. Prost : Generic remeshing of 3d triangular meshes with metric-dependent discrete voronoi diagrams. *Visualization and Computer Graphics, IEEE Transactions on*, 14(2):369–381, 2008. 133

[113] R. Veltkamp : Shape matching : Similarity measures and algorithms. *In Shape Modeling and Applications, SMI 2001 International Conference on.*, p. 188–197. IEEE, 2001. 160

[114] H. Wang, Y. He, X. Li, X. Gu et H. Qin : Polycube splines. *Computer-Aided Design*, 40(6):721–733, 2008. 159

[115] W. Wang, H. Pottmann et Y. Liu : Fitting b-spline curves to point clouds by curvature-based squared distance minimization. *ACM Transactions on Graphics (ToG)*, 25(2):214–238, 2006. 161, 162

[116] L. Wei et M. Levoy : Texture synthesis over arbitrary manifold surfaces. *In Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, p. 355–360. ACM, 2001. 39, 40

[117] L. Williams : Pyramidal parametrics. *In ACM Siggraph Computer Graphics*, vol. 17, p. 1–11. ACM, 1983. 57

[118] D. Yan, B. Lévy, Y. Liu, F. Sun et W. Wang : Isotropic remeshing with fast and exact computation of restricted voronoi diagram. *In Computer graphics forum*, vol. 28, p. 1445–1454. Wiley Online Library, 2009. 95, 102, 104, 120, 121, 130

[119] C. Yao, H. Chu, T. Ju et T. Lee : Compatible quadrangulation by sketching. *Computer Animation and Virtual Worlds*, 20(2-3):101–109, 2009. 159, 178, 180

[120] C. Yuksel, J. Keyser et D. House : Mesh colors. *ACM Transactions on Graphics (TOG)*, 29(2):15, 2010. 61

[121] E. Zhang, J. Hays et G. Turk : Interactive tensor field design and visualization on surfaces. *Visualization and Computer Graphics, IEEE Transactions on*, 13(1):94–107, 2007. 39, 40

[122] E. Zhang, K. Mischaikow et G. Turk : Vector field design on surfaces. *ACM Transactions on Graphics (TOG)*, 25(4):1294–1326, 2006. 40

[123] G. Zou, J. Hu, X. Gu et J. Hua : Authalic parameterization of general surfaces using lie advection. *Visualization and Computer Graphics, IEEE Transactions on*, 17 (12):2005–2014, 2011. 37

Échantillonnage pour l'approximation de fonctions sur des maillages

La numérisation est un procédé qui consiste à enregistrer un objet dans un ordinateur pour pouvoir ensuite le manipuler à l'aide d'outils informatiques. Nous nous intéressons dans ce manuscrit à la numérisation d'objets tridimensionnels. Il s'agit tout d'abord d'enregistrer leur forme. De nombreuses méthodes ont été développées pour répondre à ce problème, et nous nous concentrerons sur les objets représentés par des *maillages*. Sur ces objets, il est alors utile de pouvoir représenter des attributs tels que la couleur, la température ou la charge électrique, selon l'application. Nous proposons deux approches complémentaires pour aborder ce problème. La première est fondée sur le placage de textures. Cette technique consiste à déplier (paramétrer) le maillage à plat sur une image dans laquelle l'attribut est stocké. Une valeur récupérée dans l'image est ainsi associée à chaque point de l'objet. Nous proposerons une méthode permettant de masquer l'artéfact des *coutures* qui est inhérent à cette technique. Déplier le maillage nécessite qu'il soit de bonne qualité, ce qui n'est pas toujours le cas. Nous décrivons donc également dans un second temps une approche de l'échantillonnage d'une surface via un *diagramme de Voronoï restreint*. Nous expliquons en particulier comment calculer efficacement un tel objet et comment l'optimiser par rapport à un critère de qualité. Ces résultats sont ensuite appliqués au problème de l'ajustement de surfaces.

Mots-clés

modélisation 3D, maillage, géométrie, échantillonnage, approximation, optimisation, placage de textures, Voronoï, Delaunay, ajustement, remaillage

Function approximation on meshes by sampling

Digitalisation is an operation which consists in storing an object in a computer for further manipulation using data processing tools. In this document, we are interested in the digitalisation of three-dimensional objects. It is first a matter of recording the shape of the object. Many methods have been developed to address this problem, and we will focus on objects described as *meshes*. On such objects the storage of attributes like colour, temperature or electrical charge is often useful, depending on the application. We will describe two complementary approaches to deal with this issue. The first one relies on texture mapping. This technique consists in unfolding – parametrising – the mesh on a flat image in which the attribute is stored. A value recovered from the image can therefore be associated with each point of the object. We will describe a method which hides the seam artifact, commonly encountered using this technique. Unfolding the mesh demands that its quality be good, which is not always the case. We thus secondly describe a surface sampling method based on a *restricted Voronoï diagram*. We especially detail how to efficiently compute such an object and how to optimise it with respect to some quality measure. These results are then applied to the surface fitting problem.

Keywords

3D modelling, mesh, geometry, sampling, approximation, optimisation, texture mapping, Voronoï, Delaunay, fitting, remeshing