N° d'ordre : 4688

THÈSE

présentée à

L'UNIVERSITÉ BORDEAUX 1

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

par **Simon BOYÉ**

Pour obtenir le grade de

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Représentation hybride pour la modélisation géométrique interactive

---

Thèse soutenue le : 12/12/2012

Devant la commission d'examen composée de :

Christophe SCHLICK	Professeur ..	Directeur de Thèse
Gaël GUENNEBAUD	CR Inria ...	Co-Directeur de Thèse
Bruno LÉVY	DR Inria ...	Rapporteur
Loïc BARTHE	MdC (HDR)	Rapporteur
Victor OSTROMOUKHOV	Professeur ..	Examinateur (président du jury)
Pascal BARLA	CR Inria ...	Examinateur

– 2012 –

Remerciements

Tout d'abord, je remercie Loïc Barthe et Bruno Levy pour avoir accepté de relire mon travail et d'avoir fait partie de mon jury de thèse.

Je remercie mes directeurs de thèse, Christophe et Gaël, pour m'avoir suivi et accompagné depuis le master 2, et m'avoir permis de réaliser mon rêve. Leur enseignement a su me guider et me redonner espoir lorsque c'était nécessaire.

Je tiens aussi à remercier tous les membres de l'ex-équipe Iparla, pour les nombreuses discussions enrichissantes que nous avons tenu le long de ma thèse, ainsi que pour leur contribution à l'agréable ambiance des nos locaux. Je tiens tout particulièrement à remercier Pascal pour l'aide qu'il m'a apporté sur la deuxième partie de ma thèse.

À mes bétalectrices, qui ont pris de leur temps pour effectuer plusieurs passes de correction sur ce document, et m'ont grandement soutenu pendant toute ma période de rédaction et avant, j'adresse de nombreux bisous. Elles ont vécu cette thèse avec moi – surtout la fin – et je sais qu'elles sont aussi soulagées que moi d'être arrivé jusqu'au rendu de ce manuscrit.

Je remercie mes proches, amis et famille, qui m'ont tous aidé à aller jusqu'au bout à leur façon, que ce soit par leur soutien direct ou en m'aidant à relâcher la pression par moment. Je remercie tout particulièrement Noémie-Fleur, sans laquelle ma vie serait bien plus terne, et qui a beaucoup fait pour moi durant la rédaction.

Et pour finir sur une note plus légère, il me semble important de préciser que cette thèse à été effectuée et rédigée sans ingérer une goutte de café.

Résumé

De nos jours, les objets virtuels sont devenus omniprésents. On les trouve dans de nombreux domaines comme le divertissement (cinéma, jeux vidéo, etc.), la conception assistée par ordinateur ou encore la réalité virtuelle. Nous nous intéressons en particulier à la modélisation d'objets 3D dans le domaine de la création artistique. Ici, la création d'images riches nécessite de faire appel à des modèles très détaillés et donc extrêmement complexes.

Les surfaces de subdivision, traditionnellement utilisées dans ces domaines, voient leur complexité croître rapidement lorsqu'on ajoute des détails, et la gestion de la connectivité du maillage de contrôle devient trop contraignante. Une approche standard pour gérer la complexité de tels modèles est d'utiliser des représentations différentes pour la forme générale de la surface et les détails. Cependant, ces détails sont représentés par des cartes matricielles qui ne possèdent pas la plupart des avantages des représentations vectorielles, et cela complexifie certaines tâches, comme par exemple l'animation.

Dans cette thèse, nous proposons deux nouvelles représentations *vectorielles*, la première pour les surfaces de base, la deuxième pour les détails. Nous utilisons pour cette dernière une représentation vectorielle appelée *images de diffusion* permettant de créer des variations lisses à l'aide d'un ensemble réduit de contraintes. Cela nous permet de représenter aussi bien la géométrie que la couleur ou d'autres paramètres nécessaires au rendu de façon purement vectoriel, en conservant des contrôles de haut niveau.

Notre première contribution est une représentation de surfaces, baptisée LS^3, issue de la combinaison entre surfaces de subdivision et *point set surfaces*. Cette approche réduit notablement les artefacts des surfaces de subdivision aux alentours de sommets dits *extraordinaires*, qui sont connus pour poser problème. Nous présentons une analyse numérique des propriétés de ces surfaces, qui tend à montrer que du point de vue de la continuité elles se comportent au moins aussi bien que les schémas de subdivision linéaires traditionnels.

Notre deuxième contribution est un solveur pour les images de diffusion dont le principal avantage est de produire en sortie une autre représentation vectorielle légère et très rapide à évaluer. Nous illustrons la force de note solveur sur de nombreux exemples difficiles ou impossibles à réaliser avec les méthodes précédentes.

Pour conclure, nous montrons comment combiner nos deux contributions pour obtenir une représentation de surface entièrement vectorielle capable de représenter des détails sans avoir à manipuler la connectivité d'un maillage.

Mots clefs : Modélisation géométrique, surfaces de sudivision, dessin vectoriel

LaBRI
(UMR CNRS 5800)
351, cours de la Libération
33405 Talence cedex
France

Centre de Recherche
Inria Bordeaux
200 av. de la Vielle Tour
33405 Talence cedex
France

Hybrid representation for interactive geometric modeling

Abstract

Nowadays, virtual objects have become omnipresent. We can find them in various domains such as entertainment (movies, video games, etc.), computer-aided design or virtual reality. Our main focus in this document is the modeling of 3D objects in the domain of artistic creation, where rich images creation requires highly detailed and complex models.

Subdivision surfaces, the most used surface representation in this domain, quickly become very dense as the user add details, and manual handling of the connectivity becomes too cumbersome. A standard approach to handle the complexity of such models is to separate the overall shape of the surface and the details. Although, these detail maps are often stored in bitmap images that does not provide the advantages of vectorial representation, which complicate some tasks, like animation.

In this document, we present two new *vectorial* representations : the first one for the base surface, the second one for the detail maps. For the later, we use a vectorial representation called *diffusion images* that allow to create smooth or sharp variations from a small set of constraints. This enables us to represent geometry as well as color or any other parameter required for rendering, while keeping high-level controls.

Our first contribution is a surface representation, called LS^3, based on the combination of subdivision surfaces and *point set surfaces*. This approach reduces notably artifacts that subdivision surfaces produce around so called extraordinary vertices. We also present a numerical analysis of the mathematical properties of these surfaces, that show that they behave at least as well as classical subdivision schemes.

Our second contribution is a solver for diffusion images that has the particularity to produce as output a denser vectorial representation which is light and fast to evaluate. We show the advantages of this approach on several examples that would be hard or impossible to produce with former methods.

To conclude, we show how these two contributions can be used together to obtain a fully vectorial surface representation able to produce detailed surfaces without needing to deal with complex connectivity.

Keywords : Geometric modeling, subdivision surfaces, vector graphics

Table des matières

Introduction

Contexte et motivations

De nos jours, les objets virtuels sont devenus omniprésents. Largement utilisés pour le divertissement, par exemple pour la création d'effets spéciaux, de films d'animations ou de jeux vidéo, ils sont aussi utilisés dans d'autres domaines comme la conception assistée par ordinateur (CAO), l'architecture, l'archéologie, la réalité virtuelle et la publicité. Avec la montée en puissance des ordinateurs, les objets virtuels deviennent de plus en plus complexes. Cela pose de nombreuses questions : comment acquérir de tels objets, les éditer, les transmettre ou encore les restituer ?

Nous nous intéressons en particulier à la création et à l'édition d'objets 3D qui trouvent de nombreuses applications, comme la conception d'objets complexes (carrosseries, fuselages), la création d'images de synthèse de haute qualité ou la conception de mondes virtuels. Dans toutes ces applications, se pose la question fondamentale de la représentation numériques de ces objets 3D. D'une manière générale, il n'y a pas de représentation idéale convenant à toutes les situations, et des compromis doivent être faits entre précision, qualité et facilité de manipulation. Pour la création de formes 3D, les objets sont le plus souvent représentés par leur surface définissant l'interface entre le volume de l'objet et l'extérieur. Dans le monde de la CAO, il existe généralement de fortes contraintes sur la qualité des surfaces générées qui doivent présenter un haut degré de lisseur, être manufacturables, satisfaire certaines contraintes mécaniques, tout en permettant la représentation exacte de certaines formes comme des portions cylindriques ou sphériques.

Dans le domaine de la création artistique, les contraintes qualitatives sont moins fortes : les modèles doivent seulement permettre de synthétiser des images de haute qualité sans artefact visible. En revanche, afin de produire des images riches, il est nécessaire de faire appel à des modèles très détaillés et donc extrêmement complexes. Par exemple, il est possible de trouver des modèles de personnages modélisés intégralement jusqu'à des détails extrêmement fins, comme des rides ou le grain de la peau. Créer et éditer des surfaces d'une telle complexité nécessite des représentations et des outils adaptés. En pratique, la facilité d'utilisation des outils permettant de manipuler les surfaces est ici primordiale, car cela permet aux artistes de créer des modèles 3D plus complexes tout en étant plus productifs, ce qui a un impact direct sur la qualité et le coût des productions.

Dans cette thèse, nous nous intéressons particulièrement aux représentations dédiées à la création artistique permettant de modéliser des objets riches.

Représentations des objets 3D

Les grilles de voxels font partie des représentations d'objets 3D les plus simples. Elles sont similaires aux images matricielles mais en trois dimen-

sions : il s'agit de grilles régulières dont chacune des cellules, appelées *voxels*, contiennent des informations comme une couleur, un booléen ou autre suivant l'application. Si ce type de représentations fonctionne bien en 2D, car il s'agit du format natif de la plupart des dispositifs de restitution (écrans, imprimantes, etc.) et que le coût en mémoire reste raisonnable, ce n'est pas le cas pour la 3D.

Des représentations continues et *vectorielles* sont donc largement préférables, notamment parce qu'il est possible de les visualiser de différents points de vue et à différents niveaux de zooms sans artefact. Nous qualifions de *vectorielles*, des représentations légères en mémoire, indépendantes de la résolution du dispositif sur lequel elle sont restituées et éditables directement. Ces représentations sont basées sur des primitives définissant des surfaces à l'aide de modèles mathématiques. Ces primitives sont le plus souvent elles-mêmes contrôlées par un ensemble de *points de contrôle* connectés les uns aux autres. Là encore, un parallèle peut être fait avec les représentations d'images 2D, puisque cette façon de représenter des surfaces est très similaire au dessin vectoriel où des formes sont décrites à l'aide de courbes les délimitant et de primitives de remplissage. Cette façon de représenter des images se retrouve dans des nombreuses applications graphiques 2D à but artistique comme Adobe Illustrator, Adobe Flash ou Inkscape, dans les suites de bureautique comme OpenOffice, ou tout simplement dans les formats de ficher décrivant des documents tel que PDF, Postscript ou SVG.

Les maillages polygonaux sont des représentations vectorielles très simples composées d'un ensemble de sommets reliés par des faces. Par exemple, avec cinq sommets, quatre faces triangulaires et une carrée, il est possible de représenter exactement une pyramide. Cependant, cette méthode n'approche des surfaces lisses qu'au coût d'un nombre élevé de polygones. Les surfaces paramétriques résolvent ce problème en associant à un point dans un espace *paramétrique* de dimension deux une position dans l'espace. Mathématiquement, ces représentations sont similaires aux représentations des courbes utilisées en dessin vectoriel. Les plus connues sont les surfaces de Bézier, les B-splines ou les NURBS [Farin, 2002]. Cependant, de par leur paramétrisation, ces représentations ne sont capables de représenter que des surfaces homéomorphes à un plan, ce qui s'avère très contraignant en pratique d'autant plus que la structure des points de contrôle de la surface est régulière. Il est possible d'assembler plusieurs patchs paramétriques pour représenter des topologies arbitraires, mais assurer la continuité à leurs jonctions s'avère particulièrement difficile.

Ces limitations ont conduit à la naissance de représentations plus souples. Notamment, les surfaces de subdivision permettent de créer des surfaces lisses à partir de maillages polygonaux avec une connectivité arbitraire. Ces représentations font partie des plus utilisées de nos jours grâce à leur flexibilité [DeRose *et al.*, 1998].

Découplage surface-détails. Avec les approches précédentes, le nombre de points de contrôle du maillage reste directement corrélé avec la quantité de détails à représenter. Aujourd'hui, il est possible d'afficher en temps réel des modèles composés de plusieurs millions de points de contrôle. À ce niveau de détails, les informations chromatiques peuvent généralement être corrélées avec la géométrie ce qui a motivé des représentations sans connectivité s'appuyant sur un simple nuage de points [Grossman et Dally, 1998; Zwicker *et al.*, 2001; Gross et Pfister, 2007]. En revanche, d'un point de vue de la manipulation de ces surfaces, il n'est plus envisageable de manipuler les points de contrôle manuellement à cause de leur trop grand nombre ; l'édition passe alors par l'utilisation d'outils de plus haut niveau modifiant plusieurs primitives à la fois, et qui peuvent, au besoin, mettre à jour la densité et la connectivité. On peut citer plusieurs exemples, comme les logiciels *zBrush* ou *Sculptris*, ou encore les travaux de Zwicker *et al.* et ceux de Stanculescu *et al.*. Ces outils ressemblent généralement beaucoup aux outils de peinture utilisés dans les logiciels d'édition d'images matricielles : la surface est "sculptée" à l'aide de différentes "brosses". Clairement, d'un point de vue de l'édition, ces surfaces ne possèdent plus la majorité des propriétés des représentations vectorielles qui nous intéressent.

Une approche standard pour gérer la complexité de tels modèles est d'utiliser des représentations différentes pour la forme générale de la surface et les détails. Les détails eux-mêmes sont souvent avantageusement séparés en différentes couches correspondant à différentes fréquences : carte de déplacement permettant d'ajouter des détails géométriques, carte de normale pour donner l'illusion de micro-détails, plus différentes cartes contenant les paramètres des matériaux (couleur, brillance, etc.). C'est dans ce cadre que se situent nos travaux.

Représentation de la surface de base. Les surfaces de base étant *a priori* relativement simples, elles peuvent être représentées à l'aide de surfaces de subdivision tout en maintenant un nombre réduit de points de contrôle. Techniquement, les surfaces de subdivision sont souvent conçues pour représenter des surfaces *box-splines* lorsque le maillage est régulier, mais elles permettent en plus de gérer les sommets irréguliers. Cela constitue l'une des principales forces des surfaces de subdivision, mais également la principale limitation puisqu'ils sont sources de nombreux artefacts. Ces défauts peuvent être masqués manuellement par les graphistes expérimentés, mais cela complique l'utilisation des surfaces de subdivision.

Les surfaces *implicites* sont une alternative aux représentations paramétriques. Elles sont définies par l'ensemble des points $\mathbf{x}$ satisfaisant une équation de la forme $f(\mathbf{x}) = 0$, où f est une fonction de potentiel $f : \mathbb{R}^3 \to \mathbb{R}$ [Wyvill *et al.*, 1986; Max, 1983; Blinn, 1982; Bloomenthal et Wyvill, 1997]. Dans l'idéal,

f définie un champ de distance *signé*, ce qui permet de facilement différencier l'intérieur de la surfaces ($f < 0$), de l'extérieur ($f > 0$). Le principal avantage de ces représentations est la facilité avec laquelle des opérations booléennes ou de mélanges sophistiqués peuvent être réalisées[Wyvill *et al.*, 1999; Barthe *et al.*, 2003; Bernhardt *et al.*, 2010]. De plus, les surfaces produites sont souvent très lisses. Toute la difficulté ici est de proposer une méthode simple et intuitive permettant de définir et manipuler la fonction de potentiel f.

Une approche intéressante consiste à construire f de telle sorte que la surface implicite correspondante passe à proximité des sommets d'un nuage de points donné en entrée. Il s'agit de représentations semi-implicites puisque la surface approche un nuage de points qui lui est explicite. Cela inclue les méthodes à base de *Radial Basis Functions* (RBF) [Carr *et al.*, 2001; Macêdo *et al.*, 2009], ou encore les approches variationnelles [Hoppe *et al.*, 1992; Kazhdan *et al.*, 2006; Alliez *et al.*, 2007]. Dans les deux cas, une minimisation globale et coûteuse est requise, ce qui rend ces approches non-appropriées à l'édition interactive de surfaces. Au contraire, les *point set surfaces* (PSS) [Alexa *et al.*, 2003; Amenta et Kil, 2004] reposent sur des minimisations purement locales ce qui permet de les manipuler et visualiser en temps-réel [Guennebaud *et al.*, 2008]. Les PSS produisent généralement des surfaces de haute qualité, à la condition que l'échantillonnage soit suffisamment dense et localement uniforme. Les nuages de points en entrée doivent donc être assez denses et ne peuvent pas être manipulés directement. Du point de vue de l'éditabilité nous ne pouvons donc pas parler de représentation vectorielle. Par exemple, le logiciel expérimental Pointshop 3D [Pauly *et al.*, 2003] utilise des outils de type sculpture pour éditer la surface.

Représentation des détails Les cartes de détails sont plaquées sur les surfaces de base, il s'agit donc d'objets 2D. Elles sont généralement représentées par des images matricielles, avec tous les défauts qui leur sont propres : coût en mémoire élevé, impossibilité de représenter des discontinuités exactement et nécessité d'utiliser des outils de haut niveau pour l'édition en raison d'un trop grand nombre de paramètres. L'alternative consiste à utiliser des méthodes *vectorielles*. Cependant, le dessin vectoriel classique est surtout conçu pour représenter des discontinuités, et offre peu de possibilités pour ce qui est des variations lisses, telles que les dégradés de couleurs. Depuis quelques années, des représentations dédiées ont fait leur apparition comme les *gradient meshes* [Sun *et al.*, 2007]. Néanmoins, la majorité d'entre elles sont généralement conçues pour la vectorisation automatique [Lecot et Levy, 2006; Liao *et al.*, 2012] et sont peu pratiques à manipuler. Les images de diffusion [Orzan *et al.*, 2008; Finch *et al.*, 2011] font figure d'exception : elles permettent de créer des dégradés complexes avec un nombre réduit de courbes et de points de contrôle. De plus, elle offrent des contrôles riches et ne nécessitent pas de

manipuler une quelconque connectivité, ce qui les rend particulièrement faciles à utiliser. En contrepartie, le calcul de l'image finale passe par la résolution d'un problème global qui est assez délicat à résoudre efficacement.

Contributions et organisation du mémoire

Ce mémoire est composé de deux parties correspondant à nos deux contributions principales. Dans la première, présentée chapitres 1 et 2, nous étudions les représentations utilisables pour créer la surface de base et proposons une évolution de celles-ci que nous avons baptisée *least square subdivision surfaces*, atténuant certains de leurs défauts. Nous proposons d'utiliser cette méthode pour modeler des surfaces de base ; c'est à dire sans les détails les plus fin, de façon à garder une complexité raisonnable. Dans la deuxième partie, constituée des chapitres 3 et 4, nous explorons l'emploi de méthodes basées sur la diffusion pour la représentation des détails colorimétriques et géométriques, afin d'avoir un paradigme d'édition vectoriel à tous les niveaux.

Chapitre 1 - Représentations des surfaces. Nous présentons ici deux familles de représentations de surfaces. D'abord, les surfaces de subdivision qui définissent une surface de manière explicite à partir d'un maillage en s'appuyant fortement sur sa connectivité. Puis, nous abordons les *point set surfaces* qui, au contraire, définissent une surface implicite à partir d'un nuage de points non connectés. Nous détaillons les forces et les faiblesses de ces deux approches qui s'avèrent être complémentaires.

Chapitre 2 - *Least square subdivision surfaces.* Dans ce chapitre nous présentons notre nouvelle représentation de surfaces, baptisée LS^3, issue de la combinaison entre surfaces de subdivision et *point set surfaces*. Dans un premier temps, nous détaillons le fonctionnement de notre méthode et la comparons aux surfaces de subdivision classiques. Finalement, nous présentons une analyse numérique des propriétés de ces surfaces qui tend à montrer que du point de vue de la continuité elles se comportent au moins aussi bien que les schémas de subdivision linéaires traditionnels, tout en réduisant de manière significative les artefacts autour des sommets irréguliers.

Chapitre 3 - Dessin vectoriel par diffusion. Notre méthode LS^3 produit des surfaces de qualité mais présente fondamentalement les mêmes difficultés que les surfaces de subdivision pour ce qui est de représenter les surface extrêmement détaillées. Nous avons choisis de représenter les détails des surfaces via une autre représentation, et proposons d'utiliser des *images de diffusion*. Nous présentons dans un premier temps le principe des images de diffusion ainsi

que les nombreuses extensions proposées ces dernières années. Nous étudions ensuite les différents solveurs proposés jusqu'alors, et pointons leur limitations.

Chapitre 4 - Solveur vectoriel pour les images de diffusion. Dans ce chapitre, nous présentons d'abord une classification des contraintes de diffusion généralisant les différents travaux et proposons une nouvelle extension appelée *transmission* permettant un contrôle local de la diffusion. Après avoir constaté que les solveurs existants ne sont pas adaptés à une utilisation intensive des images de diffusion, nous présentons un nouveau solveur dont le principal avantage est de produire en sortie une autre représentation vectorielle légère et très rapide à évaluer. Ce solveur calcule la solution de la diffusion sur une triangulation plutôt qu'une grille de pixels ce qui permet en plus un calcul rapide et robuste de la diffusion. Nous illustrons la force de note solveur sur de nombreux exemples difficiles ou impossibles à réaliser avec les méthodes précédentes.

Chapitre 1

Représentations de surfaces

La représentation d'objets tri-dimensionnels est devenue un enjeu important dans de nombreux domaines comme la conception assistée par ordinateur, l'imagerie médicale ou l'infographie. Nous nous intéressons particulièrement à ce dernier domaine, où la *qualité visuelle* des surfaces, la *facilité d'utilisation* et la *flexibilité* des représentations sont des priorités.

La qualité visuelle est un critère subjectif difficile à évaluer indiquant que la surface est plaisante à l'œil. Une surface peut être considérée de bonne qualité en l'absence de défauts visibles, comme de légères déformations.

La facilité d'utilisation dépend des outils utilisés pour manipuler la surface, cependant, quelle que soit la méthode utilisée, elle doit permettre de modifier la surface de façon *prévisible* et *intuitive*. La possibilité de modifier les surfaces de façon interactive simplifie grandement l'édition, ce qui demande des représentations de surfaces pouvant être calculées et rendues très efficacement.

La flexibilité des représentations correspond à leur capacité à représenter une vaste gamme de surfaces. Principalement, les représentations doivent permettre de manipuler des surfaces de *topologie arbitraire* de façon simple, mais d'autres fonctionnalités comme la possibilité de représenter des arêtes vives sont fréquemment utilisées.

Dans ce chapitre, nous présentons deux familles de représentations de surfaces présentant les qualités sus-citées. Nous présentons d'abord les *surfaces de subdivision* section 1.1, qui sont actuellement les plus utilisées dans le domaine de l'infographie [DeRose *et al.*, 1998], principalement grâce à leur grande flexibilité. Nous discuterons ensuite des *point set surfaces* (PSS) section 1.2, plus récentes, qui produisent des surfaces de grande qualité.

1.1 Les surfaces de subdivision

Depuis plus de dix ans [DeRose *et al.*, 1998], les *surfaces de subdivision* sont les représentations de surfaces privilégiées pour la modélisation géométrique interactive lorsque la qualité visuelle et la facilité d'utilisation prévalent sur les propriétés de continuité théoriques [Zorin et Schröder, 2000]. Elles permettent de produire des surfaces lisses approchant ou interpolant un maillage polygonal, et sont particulièrement flexibles grâce à leur capacité à représenter des surfaces de topologie arbitraire, des arêtes vives, etc.

Le principe de la subdivision est de raffiner un maillage $\mathcal{M}_k$ de façon à obtenir un maillage plus dense $\mathcal{M}_{k+1}$ dont les sommets sont positionnés de façon à les rapprocher d'une surface *lisse*. Les positions des sommets de $\mathcal{M}_{k+1}$ sont calculées à partir d'un nombre limité de sommets de $\mathcal{M}_k$, le plus souvent à l'aide d'une simple combinaison linéaire. Cette opération peut être répétée en partant d'un maillage de base (aussi appelé *polygone de contrôle*) $\mathcal{M}_0$ pour obtenir une série de maillages $\mathcal{M}_0, \mathcal{M}_1, \ldots$ de plus en plus denses, convergeant au final vers une surface lisse qui peut, en fonction des pondérations utilisées

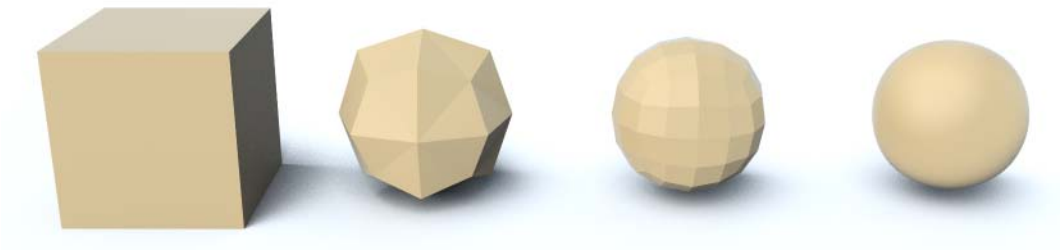


FIGURE 1.1 – **Le principe des surfaces de subdivision.** Le maillage de contrôle $\mathcal{M}_1$ (à gauche) est récursivement *découpé* et *lissé* jusqu'à tendre vers une surface lisse $\mathcal{M}_\infty$ (à droite).

pour le lissage, approcher ou interpoler les sommets de $\mathcal{M}_0$ (voir figure 1.1).

Les surfaces de subdivision présentent de nombreuses propriétés désirables :

Simplicité : le découpage du maillage et le calcul de la nouvelle position des sommets sont deux opérations relativement simples à mettre en œuvre.

Efficacité : la plupart des schémas de subdivision sont peu gourmands en calculs. En outre, il est aussi possible de faire du *raffinement adaptatif*, c'est à dire de ne subdiviser que les parties du maillage qui n'ont pas encore atteint un certain critère de qualité. Cela permet d'obtenir des maillages optimisés pour ce critère sans produire de sur-échantillonage.

Multi-résolution : les raffinements successifs produisent naturellement une structure multi-résolution particulièrement pratique pour le rendu adaptatif ou pour éditer des objets simultanément à différentes échelles.

Support compact : la zone d'influence d'un sommet est locale, restreinte à quelques voisins. Ainsi, lorsqu'un sommet est déplacé, la surface n'est déplacée que localement, ce qui facilite l'édition.

Invariance affine : appliquer une transformation affine $\mathcal{T}$ au maillage $\mathcal{M}_0$ puis subdiviser donne le même résultat que subdiviser puis appliquer $\mathcal{T}$ dans le cas des schémas linéaires.

***Fairness* :** les surfaces produites sont généralement lisses et agréables à l'œil, ce qui les rend utilisables dans le domaine de l'infographie.

Notons que certains schémas offrent la possibilité de calculer directement la position de n'importe quel point sur la surface limite [Stam, 1998]. Récemment, plusieurs travaux se sont intéressés à l'évaluation des surfaces de subdivision directement sur GPU afin de permettre la déformation et l'animation du maillage de contrôle en temps réel [Kovacs *et al.*, 2009; Nießner *et al.*, 2012].

1.1.1 Principe

Un schéma de subdivision est la combinaison d'une *règle de découpage* et d'une *règle de lissage* qui est elle-même généralement définie par un ensemble de *masques* de subdivision. En résumé, la règle de découpage modifie la topologie du maillage, alors que la règles de lissage s'occupe de la géométrie.

La règle de découpage

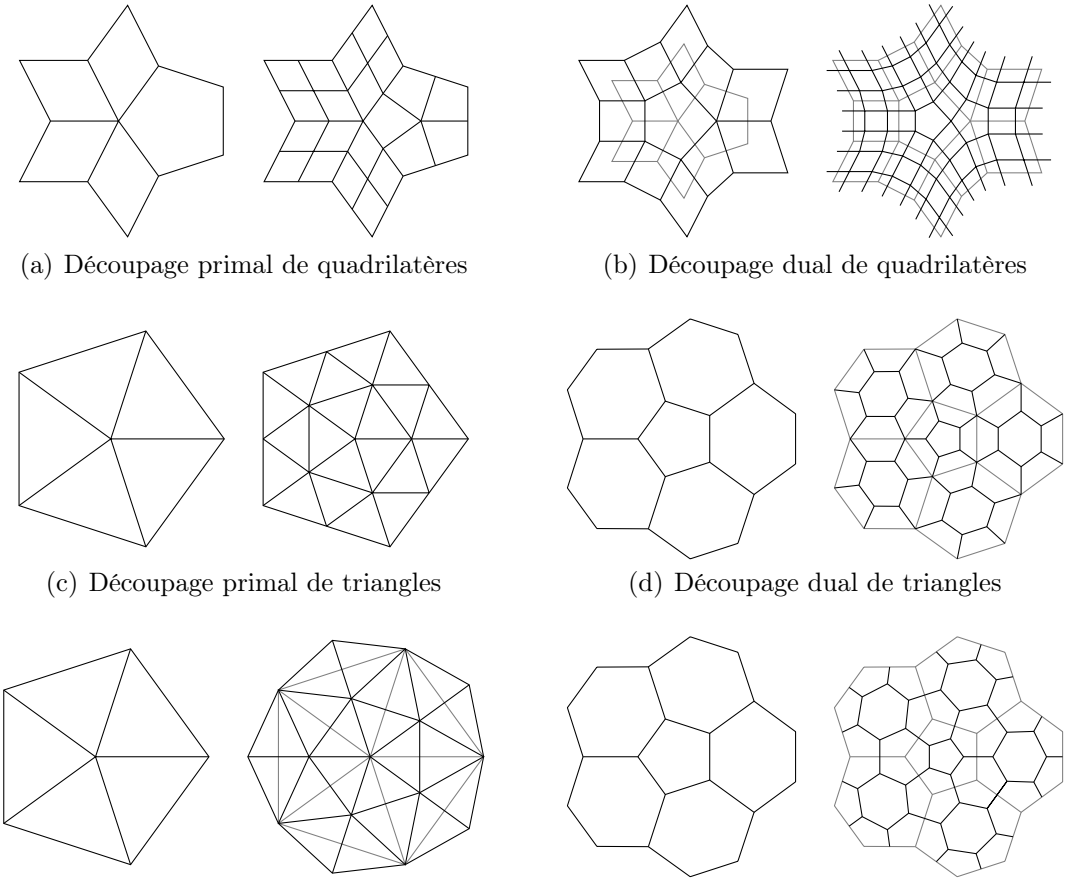
Une règle de découpage définit la façon dont le maillage $\mathcal{M}_k$ doit être raffiné pour obtenir un maillage plus dense $\mathcal{M}_{k+1}$. Il existe plusieurs critères permettant de classifier les règles de découpage [Zorin et Schröder, 2000; Oswald et Schröder, 2003] :

Primal/dual : les méthodes *primales* découpent les faces ; les méthodes *duales* découpent les sommets. Chaque méthode primale possède un dual, et inversement. La figure 1.2 illustre différents schémas primaux (à gauche) et leurs duaux (à droite). Dans le cas primal, les sommets présents au niveau k sont aussi présents au niveau $k+1$, ils sont alors appelés *sommets pairs*. En pratique, les méthodes les plus utilisées sont les méthodes primales.

Type de maillage : la règle de découpage produit le plus souvent des maillages d'un type bien particulier, constitués uniquement d'un certain type de polygones ou de sommets ayant tous la même valence. Ainsi, la règle de découpage définit le cas régulier, et aucun sommet (ou face) extraordinaire n'est inséré par la subdivision. La figure 1.2 présente quelques règles parmi les plus utilisées pour produire des maillages triangulaires (lignes 2 et 3) ou quadrangulaires (première ligne).

Certaines règles acceptent n'importe quel type de faces en entrée alors que d'autres sont plus restrictives. Par exemple, le raffinement primal diadique de maillages triangulaires (figure 1.2.c) n'accepte que des triangles en entrée alors que celui pour les maillages rectangulaires accepte des maillages arbitraires. Dans ce dernier cas, lors du premier pas de subdivision, les polygones sont remplacés par un nombre de quadrilatères égal au nombre de sommets de la face en question (figure 1.2.a). Cette opération insère un sommet extraordinaire pour les faces de valence différente de quatre, mais cela ne se produit qu'au premier pas de subdivision car par la suite toutes les faces sont des quadrilatères.

Vitesse de raffinement : les règles de découpage multiplient le nombre de faces entre deux niveaux de raffinements successifs. Les règles les plus utilisées sont des quadrisections qui multiplient le nombre de faces par quatre. De manière équivalente, les règles de découpage sont fréquemment caractérisées par le nombre de subdivisions des arêtes à chaque



(a) Découpage primal de quadrilatères (b) Découpage dual de quadrilatères

(c) Découpage primal de triangles (d) Découpage dual de triangles

(e) Découpage primal $\sqrt{3}$ (f) Découpage dual $\sqrt{3}$

FIGURE 1.2 – **Principales règles de découpages.** Découpages primaux/duaux pour la quadrisection de quadrilatères (en haut) et de triangles (au milieu) et pour la trisection de triangles (en bas). Les maillages de la figure (a) possèdent une face extraordinaire qui est transformée en un sommet extraordinaire après subdivision, et inversement pour la subdivision duale (b). Le découpage primal $\sqrt{3}$ provoque une rotation des faces autour des sommets pairs. Après deux subdivisions, les faces se réalignent et les arêtes initiales sont découpées en trois.

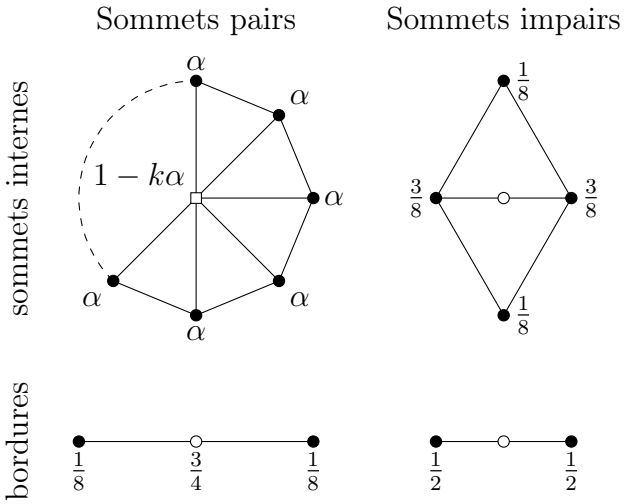


FIGURE 1.3 – **Masques du schéma de subdivision de Loop [1987].** Où k est la valence du sommet ▫ et $\alpha = \frac{1}{k}(\frac{5}{8} - (\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{k})^2)$. Pour les sommets de valence 3, on préfère utiliser $\alpha = 3/16$.

étape. Ainsi, les quadrisections des maillages quadrangulaires et triangulaires divisent les arêtes en deux et sont donc souvent qualifiées de règles *diadiques*.

Notons qu'il existe quelques schémas exotiques ne rentrant pas dans les critères définis précédemment. Par exemple, Stam et Loop [2003] ont développé un schéma de subdivision primal diadique qui produit des maillages mélangeant triangles et quadrangles en respectant le type des polygones en entrée.

La règle de lissage

La règle de découpage modifie la connectivité du maillage pour y ajouter de nouveaux sommets sans leur attribuer de position particulière. C'est le rôle de la règle de lissage qui calcule la position des sommets de $\mathcal{M}_{k+1}$, le plus souvent par une simple combinaison linéaire des sommets de $\mathcal{M}_k$ voisins. Les sommets qui entrent en jeux dans ce calcul ainsi que les poids associés sont définis par un ensemble de *masques* adaptés aux différentes configurations.

Les surfaces de subdivision sont souvent fondées sur la capacité des (box-)splines à être produites par subdivisions successives. Cependant, cela n'est possible que pour des maillages réguliers. Les maillages réguliers les plus communs incluent les maillages triangulaires où tous les sommets sont de valence six, et les maillages rectangulaires dont les sommets sont de valence quatre. En revanche les maillages réguliers ne permettent de créer que des surfaces isomorphes à un plan. Il est par exemple impossible de réaliser des

surfaces fermées comprenant uniquement des sommets réguliers. Afin de gérer des maillages de topologie arbitraire, les surfaces de subdivision utilisent des règles spéciales pour les sommets dits *extraordinaires* dont la valence est différente du cas régulier. La gestion des sommets irréguliers est la principale force des surfaces de subdivision, mais est également à l'origine de nombreux problèmes.

A titre d'exemple, nous présentons ici le schéma de Loop [1987] utilisant une règle de découpage diadique sur des maillages triangulaires et les masques présentés figure 1.3. Le schéma propose deux masques pour les sommets à l'intérieur de la surface et deux autres pour gérer les bordures. Dans les deux cas, il y a un masque pour les sommets pairs et un pour les sommets impairs qui sont ici insérés au centre des arêtes. Comme c'est le cas avec de nombreux schémas, les masques de Loop n'utilisent que le 1-voisinage du sommet, de l'arête ou de la face correspondante dans $\mathcal{M}_k$. Sans cela, de nombreux cas particuliers peuvent apparaître, principalement près des bordures, comme c'est le cas pour le schéma Butterfly [Dyn *et al.*, 1990; Zorin *et al.*, 1996].

Le schéma de Loop ne gère que les maillages triangulaires, il peut donc être nécessaire de trianguler un maillage polygonal avant d'y appliquer cette méthode. Cependant, il existe généralement plusieurs façons de procéder qui produisent des résultats différents, et il n'est pas rare qu'aucune triangulation ne donne des résultats totalement satisfaisant. Certains schémas, comme Catmull-Clark [1978], possèdent des masques pour les faces extraordinaires.

Il est généralement nécessaire que les pondérations de certains masques dépendent de la valence de certains sommets pour produire des surfaces de qualité aux alentours des sommets extraordinaires : c'est le cas du masque pour les sommets internes pairs de Loop. Le choix des pondérations est particulièrement important car celles-ci influencent directement, entre autres, la convergence du schéma, la qualité de la surface limite ainsi que sa continuité. Il n'existe généralement pas de solution idéale satisfaisant toutes les propriétés souhaitées, et des compromis doivent être effectués.

Il est possible de créer des surfaces de subdivision approchant le maillage initial ou bien l'interpolant. Afin de produire un résultat lisse, les surfaces générées par les méthodes interpolantes doivent sortir de l'enveloppe convexe du polygone de contrôle. Cela implique des poids négatifs ainsi que des voisinages plus grands. Les méthodes interpolantes sont donc généralement plus complexes à mettre en œuvre, tout en produisant souvent des surfaces visuellement moins plaisantes que les méthodes approchantes.

1.1.2 Quelques schémas

De nombreux schémas de subdivision ont été développés ces trente dernières années. Ces méthodes peuvent être classées selon les caractéristiques de leur

règle de découpage (type de maillage, primal/dual, vitesse de raffinement), mais également selon les caractéristiques de la règle de lissage :

Linéaire : les règles de lissage qui utilisent une combinaison linéaire des sommets pour calculer la position des nouveaux sommets sont dites *linéaires*.

Stationnaire : les schémas *stationnaires* utilisent les mêmes masques à chaque pas de subdivision.

Approchant/interpolant : les schémas *interpolants* produisent des surfaces qui passent par tous les sommets du polygone de contrôle, alors que les surfaces des schémas *approchants* ne passent qu'à proximité.

Continuité : les propriétés de continuité des surfaces limites varient en fonction des schémas. Les méthodes approchantes utilisées en pratique produisent généralement des surfaces C^2 sauf aux sommets extraordinaires où elles ne sont que C^1.

Par exemple, le schéma de Loop décrit précédemment est un schéma diadique, stationnaire et linéaire fonctionnant sur des maillages triangulaires uniquement et produisant des surfaces approchantes de continuité C^2 partout sauf aux sommets extraordinaires où elles ne sont que C^1.

Nous présentons ici les autres principaux schémas approchants, qui nous serviront de base pour notre méthode LS^3 présentée chapitre 2. Catmull-Clark [1978] est un schéma diadique produisant des maillages quadrangulaires utilisés dans la majorité des logiciels de modélisation. Finalement $\sqrt{3}$ [Kobbelt, 2000] est un schéma approchant pour maillages triangulaires produisant des résultats similaires à ceux de Loop, mais possédant des propriétés intéressantes pour le *raffinement adaptatif*.

Catmull-Clark

Le schéma de Catmull-Clark [1978] est historiquement l'un des tous premiers proposés, et néanmoins l'un des plus utilisés de nos jours. Il s'agit d'une généralisation des splines bicubiques conçue pour accepter des sommets extraordinaires (de valence différente de quatre) et des faces arbitraires. Les faces régulières sont des quadrilatères, qui sont bien souvent préférés aux triangles en modélisation, car il est plus aisé de manipuler des arêtes qui se croisent à angle droit puisque cela permet, par exemple, de les aligner sur les directions de courbures principales.

La règle de découpage est illustrée figure 1.2.a : un sommet est ajouté pour chaque arête et pour chaque face du maillage initial, puis chaque face est découpée en n quadrilatères, n étant le nombre de côtés de la face d'origine. Lors du premier pas de subdivision, les faces n'ayant pas quatre cotés sont donc découpées de façon à produire des quadrilatères et un sommet extraordinaire est inséré au centre de chaque face. Le reste du temps, seuls des sommets

de valence quatre et des quadrilatères sont ajoutés, ce qui correspond au cas régulier, et donc à des B-splines cubiques.

Les masques des cas réguliers sont présentés figure 1.4. Remarquez que les masques ne dépassent pas le 1-voisinage, ce qui rend le schéma de Catmull-Clark relativement simple et efficace. Les surfaces produites par la méthode de Catmull-Clark sont C^2 en tout points, sauf sur les sommets extraordinaires où elles ne sont que C^1. Comme pour le schéma de Loop, ce n'est pas toujours vrai sur les bordures et une règle particulière peut être instaurée pour corriger cela [Zorin et Schröder, 2000].

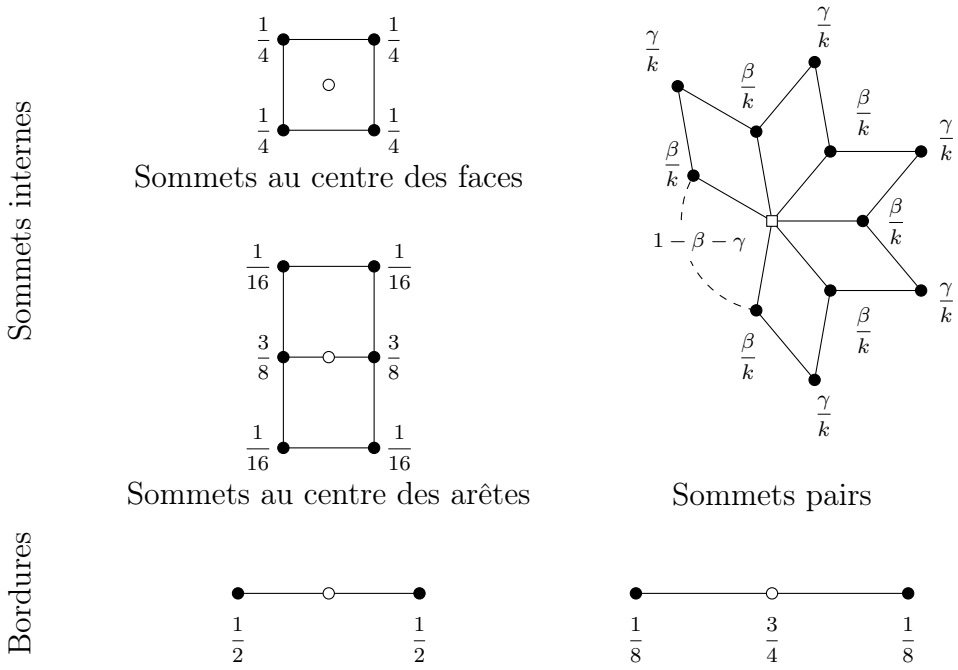


FIGURE 1.4 – **Masques du schéma de subdivision de Catmull-Clark 1978.** Où k est la valence du sommet $\square$, $\beta = \frac{3}{2k}$ et $\gamma = \frac{1}{4k}$.

Le schéma $\sqrt{3}$

La méthode $\sqrt{3}$ [Kobbelt, 2000] produit des surfaces très similaires à celles de Loop. Il s'agit d'une méthode approchante, réservée aux maillages triangulaires et offrant les mêmes propriétés de continuité. La différence majeure vient de la règle de découpage : là où les schémas de Catmull-Clark et Loop multiplient le nombre de faces par quatre à chaque pas de subdivision, $\sqrt{3}$ le multiplie par trois. Cette propriété est particulièrement utile pour le *raffinement adaptatif* car elle permet une meilleur granularité. En effet, le nombre

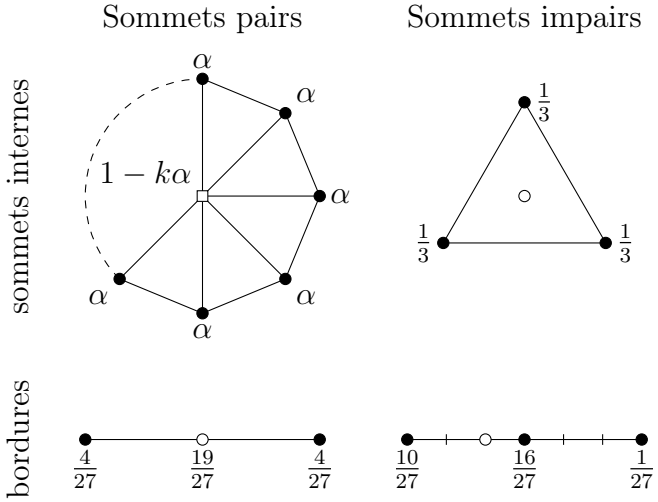


FIGURE 1.5 – **Masques du schéma $\sqrt{3}$ [Kobbelt, 2000].** Où k est la valence du sommet $\square$ et $\alpha = \frac{1}{9k}(4 - 2\cos\frac{2\pi}{k})$. Les bordures sont découpées en trois tous les deux pas de subdivision.

de faces augmentant moins vite, les surfaces $\sqrt{3}$ adaptatives produisent typiquement des maillages avec moins de faces pour un même critère d'arrêt. En contrepartie, il faut plus de pas de subdivision pour obtenir une certaine densité, ce qui signifie que le raffinement est plus coûteux.

La règle de découpage (figure 1.2.e) consiste à ajouter un sommet pour chaque face, les nouvelles faces reliant les sommets pairs aux nouveaux sommets des faces. Contrairement aux règles vues précédemment, les arêtes du maillage d'origine ne se retrouvent pas dans le maillage après raffinement : les triangles effectuent une rotation autour des sommets pairs. Après deux pas de subdivision, les arêtes sont à nouveau alignées avec celles du maillage d'origine et sont découpées en trois, là où les méthodes de subdivision diadiques les découpent en quatre. Le nom de la méthode vient du fait qu'on peut considérer qu'à chaque pas de subdivision les arêtes sont divisées par $\sqrt{3}$. Utilisée pour le raffinement adaptatif, cette règle de découpage permet d'éviter naturellement l'insertion de trous, tout en permettant de passer plus rapidement d'un maillage peu raffiné à un maillage dense que les schémas diadiques classiques.

Contrairement aux schémas présentés jusqu'ici, le $\sqrt{3}$ n'est pas conçu pour reproduire un type de spline particulier dans le cas régulier, mais plutôt de façon à utiliser les plus petits masques possibles pour la règle de découpage choisie. Ainsi, les sommets impairs n'utilisent que les trois sommets de la face correspondante, ce qui laisse pour seul choix les poids $\frac{1}{3}; \frac{1}{3}; \frac{1}{3}$. Les sommets pairs n'utilisent aussi que leur voisinage direct, ce qui ne laisse qu'un degré de liberté pour des raisons de symétrie, résolu à l'aide d'un procédé similaire à

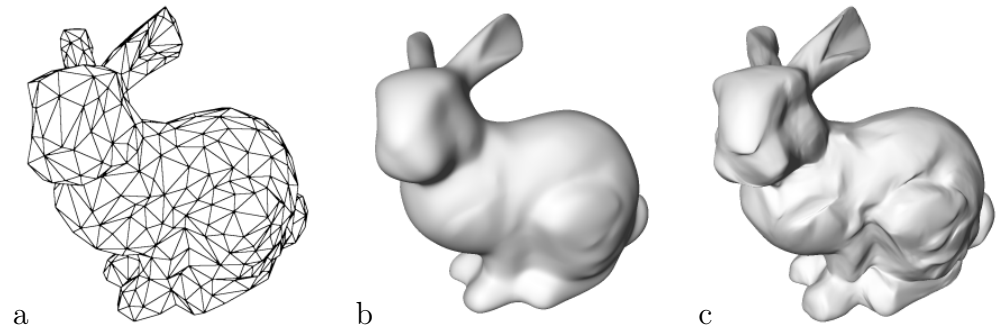


FIGURE 1.6 – **Surfaces de subdivision interpolantes et approchantes.** Le maillage (a), raffiné avec le schéma approchant de Loop [1987] et le schéma de butterfly modifié [Zorin *et al.*, 1996]

ceux utilisés pour analyser les surfaces de subdivision. Les masques du schéma $\sqrt{3}$ sont illustrés figure 1.5.

Autres schémas

Il existe bien d'autres schémas. Par exemple, il existe des équivalents interpolants aux schémas présentés ici : Butterfly [Dyn *et al.*, 1990; Zorin *et al.*, 1996] fonctionne avec un découpage diadique de triangles (comme Loop), le schéma de Kobbelt [1996] est quant à lui destiné au raffinement diadique de quadrilatères et pour finir, il existe un schéma interpolant pour la subdivision $\sqrt{3}$ de triangles [Labsik et Greiner, 2000]. Cependant, les schémas interpolants génèrent des surfaces de qualité médiocre (voir figure 1.6), tout en nécessitant des masques complexes et sont donc peu intéressants dans le cadre de cette étude.

Il est intéressant de noter l'existence de schémas utilisant les normales [Biermann *et al.*, 2000]. Plus récemment, Cashman *et al.* [2009] ont mis au point une méthode permettant de reproduire des surfaces NURBS de degré impair avec des sommets extraordinaires.

1.1.3 Les artefacts des surfaces de subdivision

Malgré tous les avantages sus-mentionnés, les surfaces de subdivision présentent de nombreux artefacts que Sabin et Barthe [2002] ont tenté de classifier. Ils définissent par "artefact" toute caractéristique de la surface qui ne peut pas être contrôlée en manipulant le réseau de points de contrôle. Cela les conduit à considérer comme artefact toute composante fréquentielle supérieure à un cycle tous les deux points de contrôle. Cette définition des artefacts est particulièrement large et inclut certains phénomènes qui ne sont pas toujours

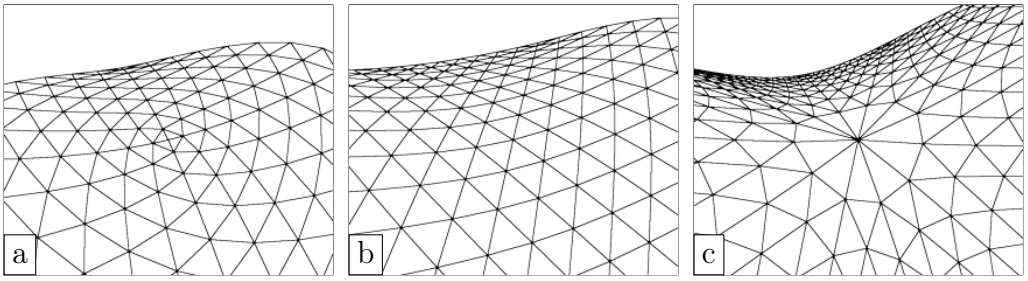


FIGURE 1.7 – **Artefacts polaires.** Comportement des surfaces de Loop [1987] aux alentours des sommets extraordinaires. Les effets de contraction (a) ou de dilatation (c) sont appelés *artefacts polaires*. En comparaison, le cas régulier (b) ne présente pas ce genre de problèmes.

problématiques en pratique. Par exemple, les *artefacts longitudinaux* se produisent lorsque les arêtes du polygone de contrôle ne sont pas alignées avec la courbure de la surface désirée, ce qui est toujours possible.

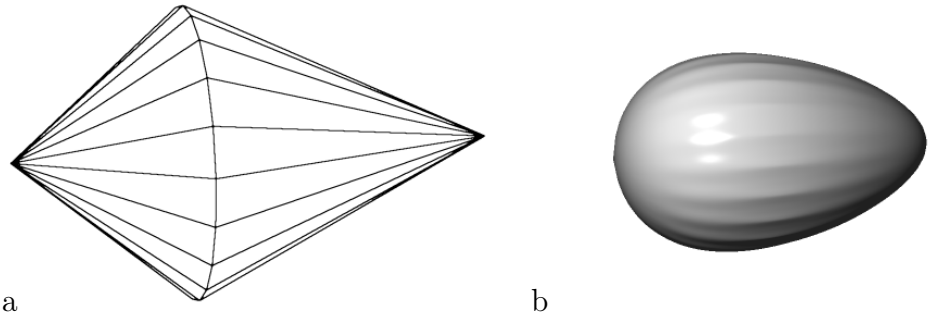
En revanche, les sommets extraordinaires sont sources de problèmes bien plus sérieux et difficiles à contourner. Nous pouvons notamment identifier les quatre types d'artefacts suivants :

Les artefacts polaires sont un problème d'échantillonnage autour des sommets extraordinaires, illustrés figure 1.7. Les sommets de faible valence ont tendance à attirer les sommets voisins alors que les sommets de forte valence les repoussent. Bien que ce phénomène n'empêche pas la convergence, ils forcent à effectuer des pas de subdivision supplémentaires pour obtenir une qualité équivalente aux zones régulières.

La continuité de la surface aux sommets extraordinaires est généralement inférieure comparée au reste de la surface. C'est une des raisons qui empêche l'adoption des surfaces de subdivision dans le domaine de la conception par ordinateur (CAO).

Des oscillations de basse fréquence mais de forte amplitude apparaissent près des sommets de forte valence comme illustré figures 1.8 et 1.9.f, particulièrement lorsqu'ils sont voisins à des sommets de faible valence. En pratique, ces oscillations apparaissent dès le premier pas de subdivision. Peu de travaux ont été réalisés sur ce type d'artefacts bien qu'il s'agisse d'un des plus visible en pratique. Ces oscillations peuvent être réduites en prenant énormément de précautions lors de la modélisation pour éviter l'introduction de telles situations en favorisant les sommets réguliers ou quasi-réguliers.

Des déformations apparaissent aux alentours des sommets extraordinaires, même si leur valence est proche du cas régulier (figure 1.9). Intuitivement, ces problèmes sont dus au fait que des sommets de valences différentes se



a b

FIGURE 1.8 – **Oscillations** dans les cas extrêmes. Ici, le modèle (a) est entièrement constitué de sommets extraordinaires de valence 4 et 24, ce qui engendre de fortes oscillations (b) dans la surface produite avec la méthode de Loop.

comportent de façon légèrement différente, produisant des surfaces plus ou moins courbées. Ces déformations sont notamment visibles au travers des reflets.

Pour contourner ces problèmes, les artistes tentent généralement d'isoler les sommets extraordinaires dans des zones peu visibles ou relativement plates et peu déformées par les animations, bien que cela rende l'édition très fastidieuse.

De nombreux travaux se sont intéressés à réduire certains de ces artefacts. En particulier, nous pouvons distinguer les méthodes tentant d'optimiser les règles de lissage elles-mêmes, et des méthodes plus radicales remplaçant localement la surface faisant défaut par un autre morceau de surface. Cependant, comme nous allons le voir, il n'existe pas de méthode miracle permettant d'éliminer tous les défauts et des compromis doivent être faits.

Optimisation des règles de lissage

Afin d'étudier le comportement des surfaces de subdivision à la limite autour d'un sommet extraordinaire $\mathbf{q}$, une méthode standard consiste à utiliser une matrice $\mathbf{S}$, appelée *matrice de subdivision* associant à un certain voisinage régulier autour d'un sommet $\mathbf{q}^k$ le même voisinage autour du sommet correspondant au niveau de subdivision suivant $\mathbf{q}^{k+1}$. Par exemple, dans le cas des schémas dont les masques n'utilisent que le 1-voisinage, utiliser un 2-anneau autour d'un sommet suffit à calculer les sommets du 2-anneau du pas de subdivision suivant :

$$\begin{pmatrix} \mathbf{q}^{k+1} \\ \mathbf{p}_1^{k+1} \\ \vdots \\ \mathbf{p}_m^{k+1} \end{pmatrix} = \mathbf{S} \begin{pmatrix} \mathbf{q}^{k} \\ \mathbf{p}_1^{k} \\ \vdots \\ \mathbf{p}_m^{k} \end{pmatrix}$$

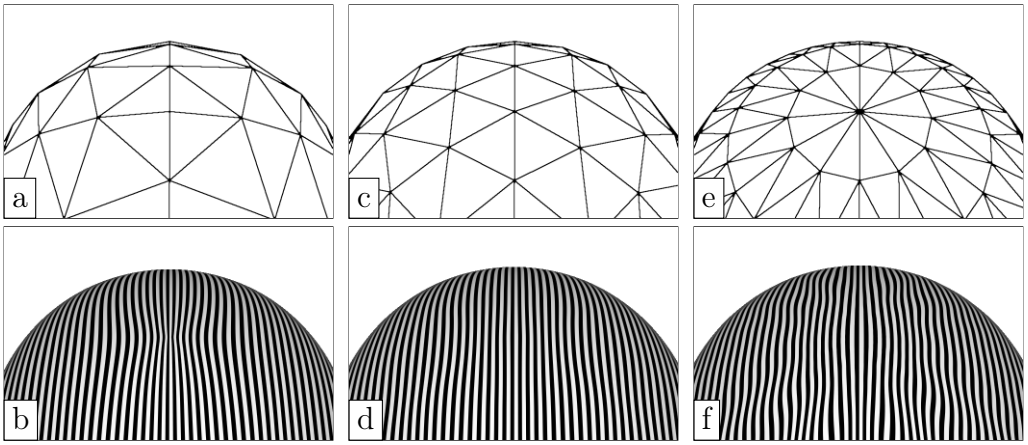


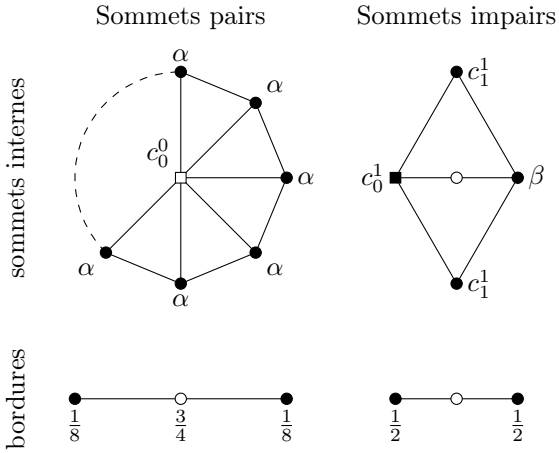
FIGURE 1.9 – **Problèmes de *fairness*** aux alentours des sommets extraordinaires. Tous les sommets appartiennent à une même sphère et sont réguliers, à l'exception des sommets centraux, de valence 4, 6 et 12 de gauche à droite. La ligne du haut représente le maillage d'origine et celle du bas la version lissée par la méthode de Loop, visualisée à l'aide de lignes de réflexions. Le cas régulier (au centre) possède de légères oscillations dues au fait que la méthode de Loop ne reproduit pas les sphères. Dans le cas de sommets de faible (resp. forte) valence, un effet de contraction (resp. dilatation) des lignes de réflexion est clairement visible. De plus, de fortes oscillations apparaissent aux alentours des sommets de forte valence (f.).

où les sommets $\mathbf{p}_1, \ldots, \mathbf{p}_m$ représentent les m voisins de $\mathbf{q}$. En appliquant cette opérateur un nombre infini de fois, nous obtenons une description de la surface autour de ce sommet. Cela revient à calculer $\mathbf{S}^\infty$ qui peut être obtenue via une décomposition en valeurs propres $\mathbf{\Lambda}$ et vecteurs propres $\mathbf{V}$ de la matrice $\mathbf{S}$, ce qui permet d'expliciter les points à la limite :

$$\begin{pmatrix} \mathbf{q}^\infty \\ \mathbf{p}_1^\infty \\ \vdots \\ \mathbf{p}_m^\infty \end{pmatrix} = \mathbf{V}\mathbf{\Lambda}^\infty\mathbf{V}^{-1} \begin{pmatrix} \mathbf{q}^0 \\ \mathbf{p}_1^0 \\ \vdots \\ \mathbf{p}_m^0 \end{pmatrix}$$

Puisque que $\mathbf{\Lambda}$ est une matrice diagonale, $\mathbf{\Lambda}^\infty$ peut effectivement être évaluée facilement. En pratique, l'étude des valeurs propres de $\mathbf{S}$ permet donc d'établir la majorité des informations sur la surface limite autour du point $\mathbf{q}^\infty$. Par exemple, il est nécessaire que les valeurs propres soient entre zéro et un pour que $\mathbf{\Lambda}^\infty$ soit calculable, et donc que le schéma converge. Nous nous référons à [Barthe *et al.*, 2005] pour plus de détails sur l'analyse des surfaces de subdivision.

Différents travaux cherchent à réduire les artefacts en calculant des masques optimisés de façon à obtenir des matrices de subdivision avec les valeurs



n	c_0^0	c_0^1	c_1^1
3	0,32517	0,15658	0,14427
4	0,49954	0,25029	0,12524
5	0,59549	0,34547	0,11182
6	0,625	0,375	0,125
7	0,63873	0,38877	0,14771
8	0,64643	0,39644	0,1768
9	0,65127	0,40132	0,21092
10	0,67358	0,42198	0,20354
11	0,68678	0,43423	0,20505
12	0,69908	0,44579	0,19828

FIGURE 1.10 – **Schéma de Loop modifié** avec la méthode de Barthe et Kobbelt [2004]. Les poids c_0^0, c_0^1 et c_1^1 dépendent de la valence k des sommets carrés ▫ et ■, $\alpha = (1 - c_0^0)/k$ et $\beta = 1 - c_1^0 - 2c_1^1$. Les poids indiqués dans la table sont optimisés pour réduire les *artefacts polaires*.

propres nécessaires pour garantir certaines propriétés [Reif, 1996; Loop, 2002; Karčiauskas *et al.*, 2004]. Le succès de ces méthodes est cependant mitigé car ces nouveaux masques font généralement apparaître d'autres comportements indésirables. Par exemple, il est possible de garantir la propriété de *courbure bornée* en ajustant les pondérations de certains schémas [Loop, 2002], mais ces méthodes ont tendance à aplatir la surface aux alentours des sommets extraordinaires comme noté par Karčiauskas *et al.* [2004], ce qui provoque des déformations lorsqu'ils sont utilisés dans des zones courbes.

En particulier, Barthe et Kobbelt [2004] ont proposé une méthode numérique unifiant et généralisant les méthodes précédentes d'optimisation des règles aux alentours des sommets extraordinaires. Elle se démarque par sa souplesse – il est possible de choisir précisément quelles propriétés privilégier – et par sa puissance : elle permet de régler plusieurs masques à la fois, ce qui laisse plus de degrés de liberté à l'optimisation et donc la possibilité de satisfaire plus de critères. Ils illustrent leur approche avec le schéma de Loop en proposant deux alternatives. La première élimine les artefacts polaires tout en diminuant légèrement les problèmes de *fairness* (figure 1.11.a). La deuxième est entièrement consacrée à l'optimisation de la courbure (en la bornant) mais augmente sensiblement les artefacts polaires tout en produisant plus d'oscillations (figure 1.11.c). Les masques et les pondérations correspondantes sont illustrés figure 1.10.

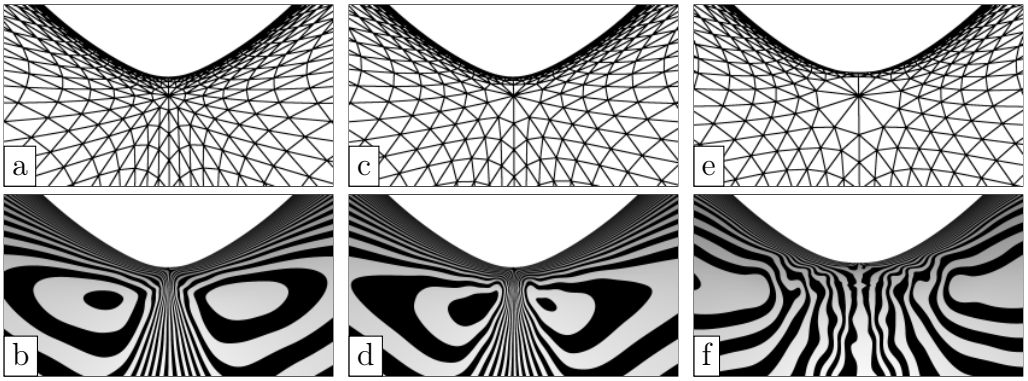


FIGURE 1.11 – **Comparaison du schéma de Loop (au centre) et ses versions modifiées [Barthe et Kobbelt, 2004].** Il est possible de corriger les *artefacts polaires* (à gauche) ou de garantir la propriété de *courbure bornée* (à droite).

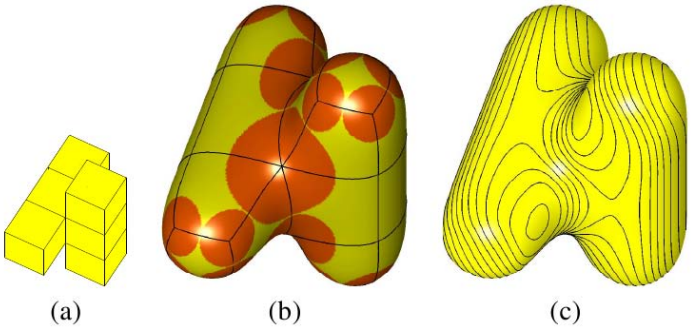


FIGURE 1.12 – **Utilisation de patchs** pour corriger les surfaces aux alentours des sommets extraordinaires (b). Images issues de l'article [Levin, 2006].

Utilisation de patchs

Il semble cependant difficile de résoudre certains problèmes sans ajouter de nouveaux comportements indésirables en utilisant de la subdivision pure. Cette observation amène à chercher des solutions alternatives. Par exemple, une approche pour corriger l'absence de continuité C^2 aux sommets extraordinaires, proposée par Levin [2006] consiste à mélanger la surface limite avec une surface polynomiale de faible degré aux alentours des sommets extraordinaires comme illustré figure 1.12. Si cette méthode produit des surfaces C^2 en tout point, elle ne corrige pas la plupart des autres artefacts. Par exemple, les oscillations typiques aux alentours des sommets de forte valence sont toujours présents, principalement dues au fait que quelques pas de subdivision sont nécessaires au calcul du patch polynomial.

1.1.4 Bilan

Les surfaces de subdivision sont des outils puissants, surtout grâce à leur grande flexibilité. Cependant, leurs défauts aux alentours des sommets extraordinaires les rend inutilisables dans certains domaines comme la CAD. Même pour les applications où la présence de quelques défauts de surface n'est pas critique, elles nécessitent un certain temps d'apprentissage et une bonne dose de patience pour être utilisées à bon escient. Ces défauts proviennent de la difficulté à gérer la connectivité des maillages arbitraires. Comme nous allons le voir dans la section suivante, les méthodes *meshless* permettent de contourner ces défauts.

1.2 *Point Set Surfaces*

Les représentations dites *meshless* définissent une surface à partir d'un nuage de points non structuré. L'absence d'information de connectivité apporte certains avantages. En effet, manipuler la connectivité d'un maillage (par exemple), que ce soit à la main ou via des algorithmes, peut rapidement devenir laborieux. De plus, ces informations ne sont pas toujours disponibles : c'est notamment le cas des données acquises par le biais de scanners 3D. Cependant, en l'absence de connectivité, il n'y a aucun *a priori* sur la surface, ce qui peut conduire à des situations ambiguës lorsque la densité du nuage de points est insuffisante. Comme nous le verrons plus tard, cela implique des contraintes fortes au niveau de l'échantillonnage des surfaces.

Parmi l'ensemble des modèles de surfaces *meshless*, la classe la plus importante et pertinente pour la modélisation géométrique interactive est de loin les *Point Set Surfaces* (PSS), illustrées figure 1.13, qui allient flexibilité, performances et facilité de mise en oeuvre. Une PSS définit une surface lisse à l'aide d'approximations locales réalisées au sens des *Moving Least Squares* (MLS). L'utilisation des PSS dans le cadre de la modélisation géométrique interactive a été explorée il y a quelques années, notamment avec le logiciel de recherche Pointshop3D [Zwicker *et al.*, 2002; Pauly *et al.*, 2003]. Comme souvent avec les représentations de bas niveau, il n'est pas envisageable de manipuler les points directement car cela serait trop fastidieux. Pour compenser, des outils de plus haut niveau sont proposés dans Pointshop3D permettant de manipuler les surfaces à l'aide, entre autres, d'outils de sculpture. En arrière plan, le logiciel s'appuie sur une PSS pour ajuster la densité du nuage de points pendant l'édition et ainsi maintenir un échantillonnage suffisant.

Dans cette section, nous présentons d'abord le principe des MLS dans le cas fonctionnel en 1D dans la section 1.2.1 avant d'étudier plus en détail la reconstruction de variétés, section 1.2.2. Puis nous discutons des contraintes d'échantillonnage inhérentes à ces méthodes ainsi que des techniques utilisées pour les amoindrir en section 1.2.3.

FIGURE 1.13 – **Reconstruction d'un nuage de points avec les PSS.** La méthode employée est celle de Guennebaud *et al.* [2008].

1.2.1 Principe des MLS

Nous allons commencer par présenter le principe des MLS dans le cas fonctionnel. Pour la simplicité des écritures, nous considérerons le cas 1D, mais le principe s'étend de manière triviale à la reconstruction de données multi-variées définies dans un espace de dimension arbitraire. Nous disposons d'un ensemble d'échantillons $\mathbf{p}_i = (x_i, f_i)$ et nous cherchons une fonction lisse $f : \mathbb{R} \to \mathbb{R}$ qui approche au mieux les échantillons $\mathbf{p}_i$.

Une première solution consiste à approcher l'ensemble des échantillons à l'aide d'un modèle simple g (par exemple, un polynôme) au sens des moindres carrés :

$$f = \arg\min_{g} \sum_{i} (g(x_i) - f_i)^2 \tag{1.1}$$

Comme illustré figure 1.14, la difficulté avec cette méthode consiste à choisir un modèle qui permette d'approcher convenablement les données : un modèle avec trop de degrés de liberté produit une erreur faible au niveau des échantillons mais produit de fortes oscillations, alors qu'un modèle doté de trop peu de degrés de liberté conduit à un lissage excessif.

Le principe des MLS est d'approcher un modèle g_x *localement* autour d'un point d'évaluation x à l'aide de la méthode des moindres carrés pondérés :

$$g_x = \arg\min_{g} \sum_{i} w_i (g(x_i) - f_i)^2 \tag{1.2}$$

où les poids $w_i = \phi(|x_i - x|)$ sont calculés grâce à une *fonction de poids* ϕ continue, positive et décroissante. La valeur de la fonction f au point x correspond alors à la valeur de l'approximation locale g_x au point x, soit :

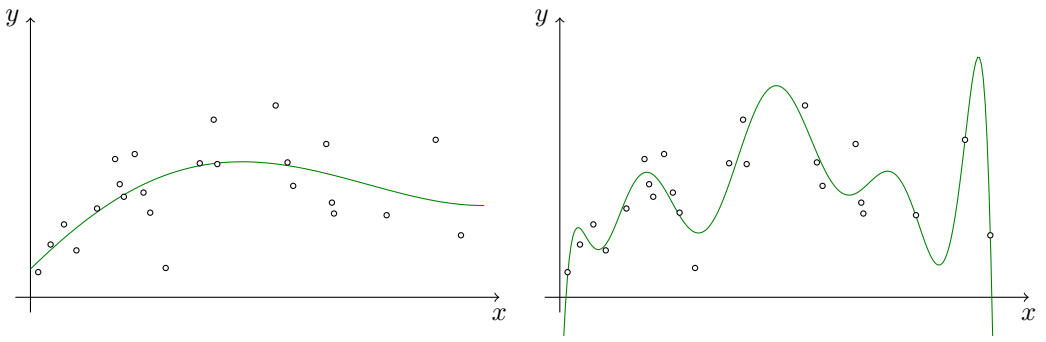


FIGURE 1.14 – **Approximation fonctionnelle à l'aide des moindres carrés.** Le choix du degré du polynôme à approcher est délicat : un degré trop faible (à gauche) approche les échantillons trop grossièrement, mais un degré trop élevé provoque des oscillations. Le degré idéal est fortement dépendant des données.

$f(x) = g_x(x)$. Ainsi, lorsque le point d'évaluation x est déplacé de façon continue, l'*approximation locale* g_x varie elle aussi de façon continue [Levin, 1998]. En pratique, la classe de continuité de f est directement liée à celle de ϕ.

Comme illustré figure 1.15, la fonction de poids a une importance majeure sur les résultats. Dans cet exemple nous avons utilisé la fonction de poids suivante

$$\phi(x) = \begin{cases} \left(1 - (x/r)^2\right)^4 & \text{si } x < r, \\ 0 & \textit{sinon}. \end{cases} \tag{1.3}$$

où le paramètre r est appelé *rayon d'influence* de la fonction de poids. Ce paramètre permet de contrôler quels échantillons sont pris en compte lors du de l'ajustement de l'approximation locale, ce qui permet en pratique de lisser plus ou moins les données. En d'autres termes, la fonction ϕ joue donc le rôle d'un filtre *passe-bas*. Notons qu'une fonction de poids, à support compact et de rayon trop faible, peut rendre le calcul de l'approximation locale impossible ou casser la continuité de la surface. Ce problème est abordé plus en détails section 1.2.3.

Le choix du modèle d'approximation locale joue aussi un rôle important. En ne considérant que les modèles polynomiaux, il reste le choix de leur degré. Les détails fins sont difficiles à capturer avec des polynômes de très faible degré. Au contraire, un trop haut degré rend le polynôme extrêmement sensible aux variations de poids, ce qui provoque des oscillations comme illustré figure 1.15. En pratique, nous recommandons des polynômes quadratiques car ce sont les polynômes de plus faible degré permettant de "sortir" de l'enveloppe convexe ce qui est indispensable pour s'approcher d'une interpolation lisse sans introduire d'oscillations.

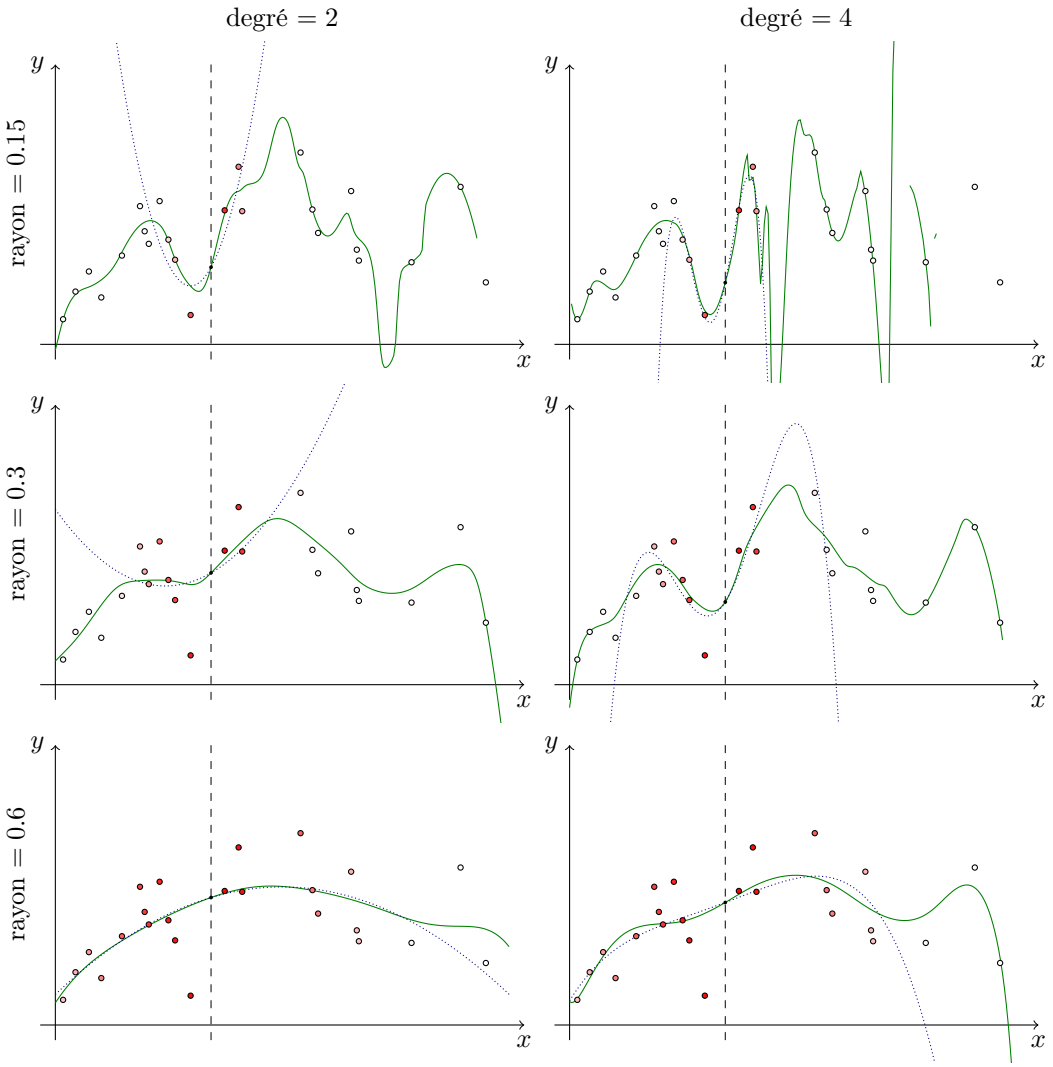


FIGURE 1.15 – **MLS fonctionnel 1D.** Exemples de reconstructions MLS (en vert) pour différents rayons d'influence de la fonction de poids en utilisant des polynômes de degré 2 et 4. La courbe en pointillés correspond à l'approximation locale (en bleu) calculée au niveau de la ligne verticale, et la couleur des échantillons à leur influence sur l'ajustement.

Notons que ces deux paramètres sont interdépendants. En effet, les oscillations qui apparaissent avec des polynômes de haut degré peuvent être réduites en augmentant le rayon de la fonction de poids, et un polynôme de faible degré va approcher les échantillons plus fidèlement en réduisant le rayon. Pour une reconstruction lisse, le modèle doit être fixé mais le rayon d'influence de la fonction de poids peut varier, ce qui permet de contrôler la qualité de l'approximation. Comme nous le verrons section 1.2.3, cela s'avère indispensable pour produire une reconstruction correcte des échantillons lorsque la densité n'est pas uniforme.

1.2.2 Reconstruction MLS des surfaces de formes libres

Le problème de la reconstruction de $(d-1)$-variétés, c'est à dire de courbes arbitraires dans le plan, ou de surfaces 2D de formes libres dans un espace 3D est légèrement différent du cas fonctionnel. En effet, nous ne cherchons plus à approcher des valeurs f_i, mais à trouver une surface approchant au mieux les points en entrée. Soit un ensemble d'échantillons $\mathbf{p}_i$, pour un point d'évaluation $\mathbf{q}$ donné dans l'espace, l'idée générale consiste à approcher les échantillons localement par une surface analytique $\mathcal{S}_\mathbf{q}$ au sens des moindres carrés pondérés.Cependant, la procédure de minimisation ainsi que la généralité et stabilité qui en résultent varient de façon significative en fonction de la paramétrisation dans laquelle $\mathcal{S}_\mathbf{q}$ est définie.

Calcul de l'approximation locale

Une paramétrisation 2D globale permet de définir $\mathcal{S}_\mathbf{q}$ de manière paramétrique à l'aide d'une fonction polynomiale bivariée de $\mathbb{R}^2 \rightarrow \mathbb{R}^3$. Nous nous retrouvons alors dans le cas fonctionnel précédent où les valeurs à reconstruire sont simplement les coordonnées des sommets positionnés dans l'espace paramétrique 2D. Cette approche est notamment utilisée en mécanique pour discrétiser de manière *meshless* des problèmes de déformation de membranes (*thin-shell*) [Guo *et al.*, 2006]. Elle est en revanche peu intéressante pour nos objectifs puisqu'elle nécessite une paramétrisation globale du nuage de points. En outre, avec cette approche, l'approximation locale $\mathcal{S}_\mathbf{q}$ n'est définie que pour un point $\mathbf{q}$ dans la paramétrisation. Une opération aussi basique que de projeter un point arbitraire sur la surface sous-jacente est donc un réel challenge.

Une paramétrisation planaire locale peut être obtenue en approchant un plan de référence sur le voisinage du point d'évaluation $\mathbf{q}$ [Alexa *et al.*, 2001; Levin, 2003]. Dans cette paramétrisation, le voisinage peut être considéré comme un champ de hauteur et donc directement approché par un polynôme bivarié comme dans le cas fonctionnel illustré figure 1.16. Cependant, une telle approche nécessite une surface relativement plane localement pour, d'une part,

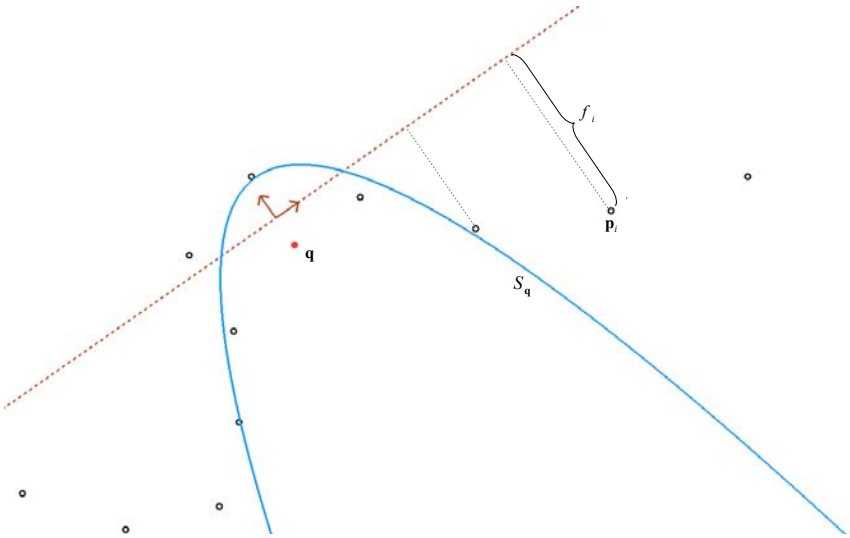


FIGURE 1.16 – **Calcul de l'approximation locale à l'aide d'une paramétrisation planaire.** Un plan de référence (en pointillés rouges) approchant les échantillons autour du point $\mathbf{q}$ est d'abord calculé. Dans ce nouveaux repère, un polynôme (ici de degré 2, en bleu) est ensuite calculé de façon à approcher les distances f_i séparant les échantillons $\mathbf{p}_i$ du plan.

pouvoir approcher le plan de référence de manière stable, et, d'autre part, garantir que le voisinage puisse être correctement représenté par un champ de hauteur sans repliement.

La paramétrisation cartésienne naturelle permet d'utiliser directement des surfaces algébriques (i.e. implicites) comme approximation locale [Amenta et Kil, 2004; Guennebaud et Gross, 2007]. Nous parlons alors de PSS algébriques, ou APSS pour *Algebraic Point Set Surfaces*. Plus précisément, ces méthodes approchent un champ scalaire trivarié $s_\mathbf{q} : \mathbb{R}^3 \to \mathbb{R}$ tel que la 0-isosurface $S_\mathbf{q} = \{\mathbf{x} \in \mathbb{R}^3; s_\mathbf{q}(\mathbf{x}) = 0\}$ soit la plus proche possible des échantillons :

$$s_\mathbf{q} = \arg\min_s \sum_i w_i s_\mathbf{q}(\mathbf{p}_i)^2 \ . \tag{1.4}$$

Afin d'éviter la solution triviale $s_\mathbf{q} = 0$, des contraintes de régularisation supplémentaires doivent être ajoutées. En l'absence d'information supplémentaire, l'idée est de contraindre la norme du gradient de $S_\mathbf{q}$ à 1. Cette approche a été utilisée avec succès avec des plans implicites [Amenta et Kil, 2004; Alexa et Adamson, 2004] et des sphères algébriques [Guennebaud et Gross, 2007].

L'utilisation de normales $\mathbf{n}_i$ associées à chaque échantillon $\mathbf{p}_i$ permet de simplifier la procédure d'approximation des APSS, tout en la rendant bien

plus robuste et stable dans les cas extrêmes. La nouvelle procédure consiste à commencer par calculer le gradient de $s_{\mathbf{q}}$ de telle sorte qu'il approche au mieux les normales données :

$$\nabla s_{\mathbf{q}} = \arg\min_{\nabla s} \sum_i w_i \left\| \nabla s(\mathbf{p}_i) - \mathbf{n}_i \right\|^2 . \tag{1.5}$$

Une simple intégration permet d'obtenir $s_{\mathbf{q}}$ à un paramètre constant C près, qui est lui même obtenu de telle sorte à minimiser la distance entre l'iso zéro et les échantillons :

$$C = -\frac{\sum_i w_i s_{\mathbf{q}}(\mathbf{p}_i)}{\sum_i w_i} . \tag{1.6}$$

Cette approche à été utilisée pour obtenir des procédures d'approximation pour des plans [Alexa et Adamson, 2004] et des sphères [Guennebaud *et al.*, 2008] très efficaces. En particulier, l'utilisation de sphères algébriques présente l'avantage d'être bien plus robuste au sous-échantillonage (régions de fortes courbures, morceaux de surfaces proches) tout en étant très rapide à calculer.

Définition implicite des surfaces MLS

Dans le cadre de la reconstruction de surfaces arbitraires, calculer des approximations locales définies dans une paramétrisation planaire locale, ou dans la paramétrisation cartésienne, ne suffit pas à définir une surface. Deux approches sont couramment utilisées. La première, illustrée figure 1.17, consiste à définir un opérateur de projection permettant de projeter un point $\mathbf{q}_0$ sur la surface. Projeter $\mathbf{q}_0$ sur son approximation locale $S_{\mathbf{q}_0}$ permet d'obtenir un nouveau point $\mathbf{q}_1$. Cependant, contrairement au cas fonctionnel vu précédemment, cela n'est pas suffisant car le point $\mathbf{q}_1$ n'a généralement pas la même approximation locale que $\mathbf{q}_0$; $\mathbf{p}_1$ n'appartient donc généralement pas à la surface MLS mais s'en est rapproché. En re-projetant itérativement le point $\mathbf{q}_0$ sur l'approximation locale calculée en $\mathbf{q}_i$, nous obtenons une suite de points qui converge rapidement vers la surface. Il s'agit de la définition initiale des PSS où un point est considéré sur la surface si il appartient lui même à son approximation locale : l'ensemble des points $\{\mathbf{q};\, \mathbf{q} \in \mathcal{S}_{\mathbf{q}}\}$ définit alors une surface lisse.

L'autre approche, qui est bien entendue équivalente, consiste à considérer le champ de potentiel $f(\mathbf{x}) : \mathbb{R}^3 \to \mathbb{R}$ retournant la distance entre le point d'évaluation $\mathbf{x}$ et son approximation locale associée : $dist(\mathbf{x}, S_{\mathbf{x}})$. Dans le cas où les approximations locales $S_{\mathbf{x}}$ sont elles-mêmes définies par une surface algébrique $s_{\mathbf{x}}$, le champ de potentiel global est simplement défini par : $f(\mathbf{x}) = s_{\mathbf{x}}(\mathbf{x})$. Cette fonction définit une surface implicite constituée de tous les points $\mathbf{x}$ où $f(\mathbf{x}) = 0$, ce qui correspond bien à l'ensemble des points appartenant à leur approximation locale. Cette définition implicite via un champ

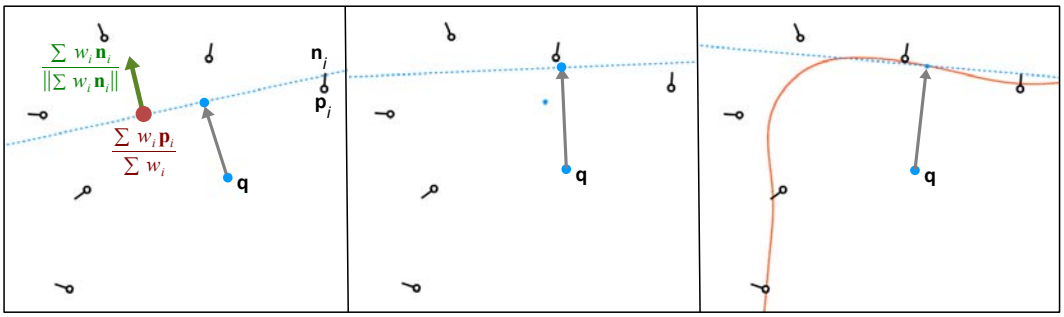


FIGURE 1.17 – **Fonctionnement de l'opérateur de projection MLS** à l'aide une approximation locale planaire. Projeter le point **q** sur son approximation locale (en pointillés bleus) ne produit pas un point appartenant à la surface MLS (à gauche). Le point **q** est ensuite reprojeté récursivement sur l'approximation locale correspondant au point calculé précédemment (au milieu). Cette procédure converge rapidement vers un point appartenant à la surface MLS (à droite).

de potentiel est particulièrement intéressante lors de l'utilisation d'approximations locales orientées, c'est à dire différenciant l'intérieur et l'extérieur de la surface, puisque cela permet d'obtenir un champ de distance signé. Cette information ne peut être obtenue que par les méthodes exploitant les normales des points. Il est alors possible de reconstruire la surface à l'aide de méthodes bien connues comme les *marching cubes* [Lorensen et Cline, 1987].

1.2.3 MLS et échantillonnage

Afin de garantir la stabilité de l'ajustement en tout point voisin de la surface, le rayon d'influence de la fonction de poids r doit être suffisant pour qu'il y ait suffisamment de points considérés. D'un autre coté, un rayon d'influence trop important conduit au mieux à un lissage excessif comme illustré figure 1.18, et au pire à des artefacts dus à la prise en compte de points incohérents. Dans le cas d'un échantillonnage non uniforme, le rayon d'influence doit alors être ajusté localement. C'est ce que proposent Pauly *et al.* [2003] en lançant un calcul d'estimation locale de la densité pour chaque point d'évaluation **q** à l'aide de la méthode des k-plus-proches voisins. En revanche, la reconstruction n'est alors plus que C^0, et les oscillations présentes dans l'estimation de densité se retrouvent dans la surface finale. Une meilleure approche consiste à assigner un rayon d'influence r_i à chaque échantillon $\mathbf{p}_i$ qui peut être pre-calculé en analysant son voisinage. Les pondérations deviennent alors : $w_i = \phi(\frac{|\mathbf{q}-\mathbf{p}|}{h \cdot r_i})$, où h est un paramètre permettant d'ajuster globalement le degré de lissage. Une variante consiste à associer des fonctions de poids ellipsoïdales [Adamson et Alexa, 2006] qui permettent de prendre en compte un échantillonnage dense

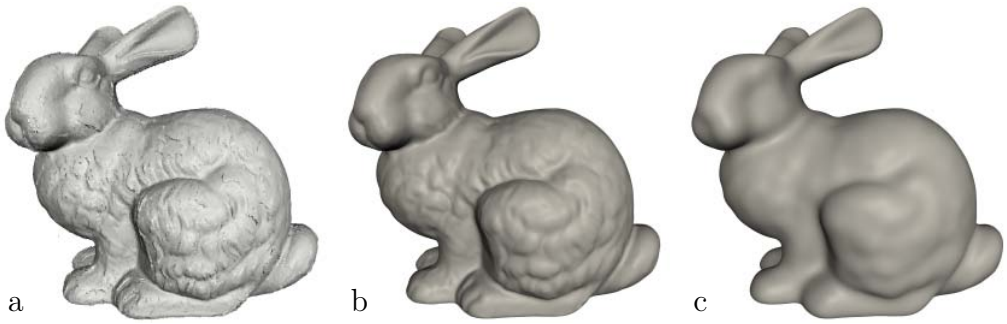


FIGURE 1.18 – **Influence du rayon de lissage.** Le modèle du *bunny* (a) reconstruit à l'aide de la méthode de Guennebaud *et al.* [2008] avec un faible rayon de lissage (b) et un rayon plus important (c).

dans une direction mais épars dans une direction orthogonale. En pratique, cette approche n'est envisageable que si le nuage de points a été généré spécifiquement pour une telle pondération. Dans le cas contraire, une pondération isotrope reste préférable.

D'autres problèmes peuvent survenir indépendamment du choix du rayon d'influence des échantillons. Dans le cas où deux surfaces sont très proches, même les meilleures méthodes de pondération ne permettent pas d'isoler les échantillons des deux surfaces, ce qui perturbe la procédure d'ajustement. C'est alors à la méthode d'approximation locale d'être capable de gérer ces cas. L'utilisation d'approximations planaires est très instable dans ces situations : soit les deux surfaces se retrouvent fusionnées, soit le plan calculé ne parvient pas à approcher convenablement la surface (voir figure 1.19). Dans ces situations, les approximations locales capables d'évaluer les deux morceaux de surface, même de façon très grossière, sont avantagées. C'est notamment le cas des sphères, qui s'avèrent produire des résultats bien plus robustes que les plans. Dans ce cas, les méthodes utilisant les normales permettent à la procédure d'ajustement de plus facilement différencier les deux surfaces, à condition que celles-ci aient pu être calculées correctement en premier lieu.

1.3 Bilan

Les surfaces de subdivision sont des outils puissants, surtout grâce à leur grande flexibilité. Notamment, les possibilités de représenter des objets possédant une topologie arbitraire, de pouvoir aligner les *flux* (c'est à dire la direction des arêtes) avec les courbures de la surface désirée et d'adapter la densité du réseau de points de contrôle en fonction de la quantité de détails désirée ; tout cela combiné aux possibilités d'édition multi-résolution et au fait que tous les sommets ont un comportement uniforme, rendent les surfaces de subdivi-

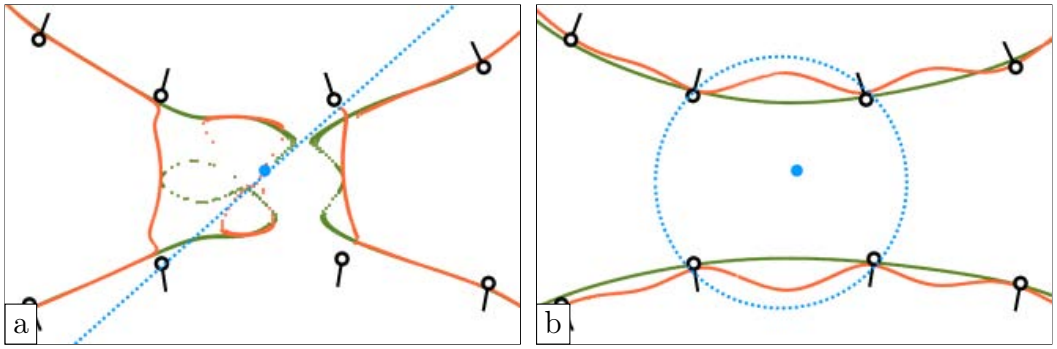


FIGURE 1.19 – **MLS dans le cas de surfaces proches.** L'utilisation de droites (plans) (a) comme approximation locale est moins robuste que l'utilisation de cercles (sphère) (b). Les courbes bleues correspondent à l'approximation locale au point bleu, les courbes rouges aux méthodes d'approximation sans normale et les vertes aux méthodes avec normales. Image provenant de [Guennebaud et Gross, 2007].

sion particulièrement puissantes et intuitives à manipuler. Cependant, leurs défauts autour des sommets extraordinaires les rendent inutilisables dans certains domaines comme la CAD et compliquent leur utilisation dans les autres domaines.

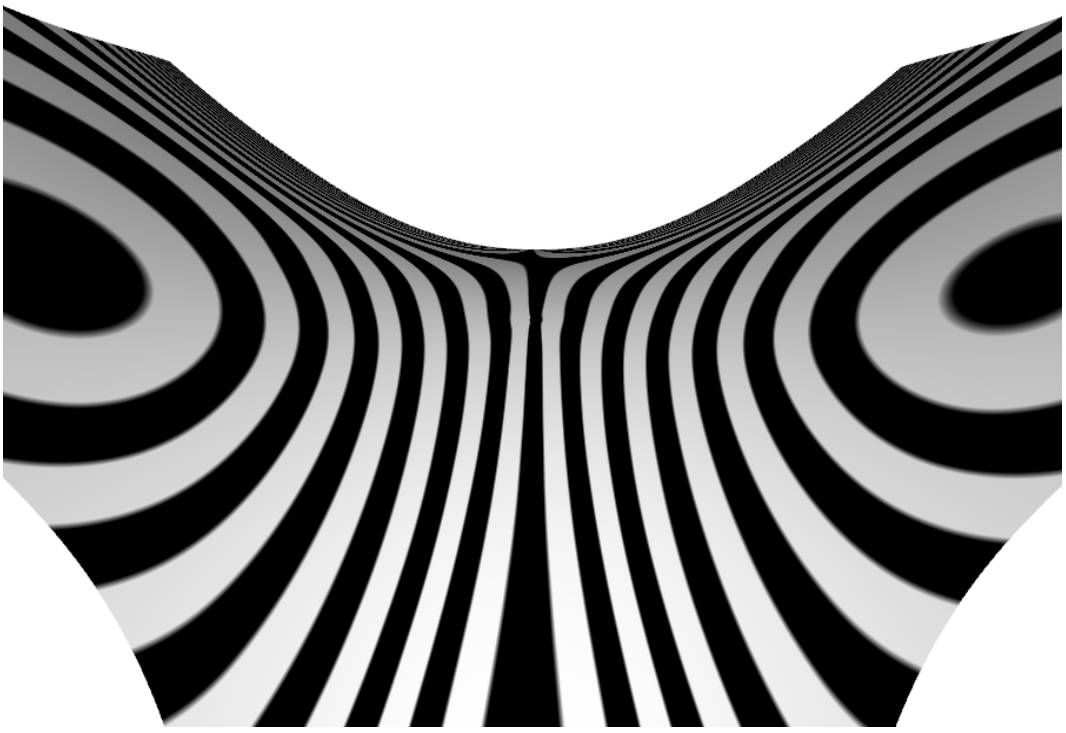
D'un autre côté, les surfaces MLS produisent des surfaces de très bonne qualité mais sont difficiles à évaluer de façon robuste à cause de l'absence d'*a priori* sur la surface. Plus précisément, la difficulté avec les surfaces MLS vient du fait que l'absence d'information sur la topologie de la surface oblige à fournir un échantillonnage suffisant afin de pouvoir retrouver cette information en analysant localement les échantillons. Dans le cadre de la modélisation géométrique interactive, les surfaces MLS sont peu utilisées : une manipulation directe serait trop fastidieuse (les points sont trop nombreux) et conduirait à des problèmes de robustesse étant donné que l'utilisateur peut déplacer les sommets de façon arbitraire. Cela oblige à les manipuler via des outils de plus haut niveau, comme des outils de sculpture. Cependant, même dans ce cas, leur intérêt est moindre : elles ne gèrent pas l'édition multi-résolution aussi naturellement que les surfaces de subdivision et les représentations denses sont assez peu adaptées pour créer des animations complexes. En pratique, les principaux outils de sculpture comme ZBrush ou Mudbox produisent des surfaces de subdivision multi-résolution en sortie. Donc, même si la représentation interne de tels logiciels se fonde sur des surfaces de très bonne qualité comme les MLS, le résultat final possédera les défauts des surfaces de subdivision.

En pratique, l'omniprésence des surfaces de subdivision dans les domaines de l'infographie peut s'expliquer facilement : malgré leurs défauts, leur flexibilité permet de réduire les coûts de production et améliorent la productivité

des artistes. En plus de cela, les artistes d'aujourd'hui sont formés à leurs utilisations. Il est donc clair que, pour qu'une nouvelle représentation de surface remplace les surfaces de subdivision dans ce domaine, il faut qu'elle soit au moins aussi flexible.

Chapitre 2

Least Square Subdivision Surfaces



Comme nous l'avons vu au chapitre précédent, les surfaces de subdivision sont devenues grâce à leur polyvalence une représentation standard dans le domaine de l'infographie, depuis plus de dix ans. Cependant, les difficultés liées à la gestion des sommets extraordinaires – gestion indispensable pour modéliser des surfaces de topologie arbitraire – conduisent à des artefacts dégradant la qualité des surfaces générées autour de ces sommets.

D'un autre côté, nous avons vu en section 1.2 que les représentations *meshless* basées sur les MLS permettent d'obtenir des surfaces d'excellente qualité à partir de nuages de points non structurés. En contre-partie, la surface doit être suffisamment échantillonnée afin d'éviter toute ambiguïté lors de la reconstruction implicite.

Une idée naturelle est donc d'essayer de combiner ces deux approches afin de tirer parti au mieux de la flexibilité d'utilisation des surfaces de subdivision et de la qualité de reconstruction des surfaces MLS. Pour atteindre cet objectif, deux approches sont envisageables. Notre première idée était de partir des surfaces MLS et de les rendre parfaitement robustes en exploitant un minimum d'informations topologiques issues du maillage. Rappelons que nous désirons obtenir une représentation facile à manipuler et ayant un comportement intuitif et surtout *prédictible*. Pour cela, il semble intéressant d'essayer de reproduire le comportement de la subdivision, à savoir que le rayon de lissage est directement lié à la connectivité, ce qui offre un contrôle purement *local*, ce qui est fortement désirable. En effet, une même forme du maillage de contrôle permet d'obtenir des surfaces différentes en changeant simplement la connectivité et la densité de celui-ci, comme illustré figure 2.1. Il s'agit d'un contrôle explicite qui permet d'obtenir une grande richesse de formes. Intuitivement, afin d'obtenir un résultat similaire avec une approche MLS, le problème revient donc à associer à chaque sommet d'un maillage une fonction de poids approchant au mieux son voisinage de sorte que les sommets d'un k-anneau de voisinage se trouvent sur une même iso de poids. Nous appelons une telle pondération, une *pondération topologique*.

Bien qu'il s'agisse d'une approche particulièrement attractive, qui potentiellement permettrait en plus de bénéficier d'une représentation implicite, elle s'est avérée difficile à concrétiser en pratique. La première difficulté consiste à formuler ces fonctions de poids topologiques, sachant que pour approcher fidèlement le maillage, celles-ci doivent pouvoir être arbitrairement compliquées, comme illustré figure 2.2, tout en étant au minimum C^1, ou mieux C^2 continues. Pour cela une première idée consiste à formuler le problème de manière variationnel, ce qui nous amène à un problème très similaire à celui des images de diffusion que nous aborderons au chapitre 3. Clairement, résoudre un tel problème pour chacun des voisinages n'est pas envisageable. Une seconde idée consisterait à simplement approcher les poids topologiques que l'on cherche afin d'obtenir une formulation analytique. Cela reste très difficile car certaines

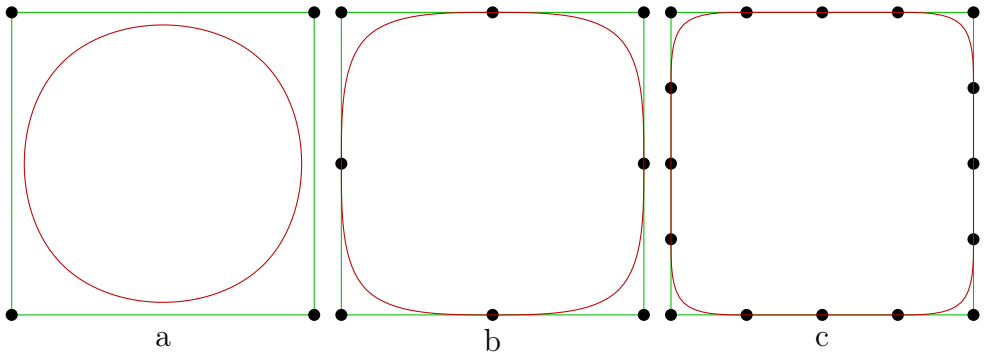


FIGURE 2.1 – **Comportement de la subdivision,** ici illustré avec des *splines cubiques* (voir les règles en bordure du schéma de Loop, figure 1.3). L'importance du lissage est directement liée à la connectivité : un carré approche visuellement d'un cercle presque parfait (a), mais l'ajout de sommets sur les arêtes du carré permet d'obtenir un carré avec les bords plus ou moins arrondis (b et c).

contraintes ne peuvent être relaxées : il est par exemple nécessaire de faire très attention à ne pas créer de "trous" où trop peu d'échantillons seraient disponibles pour le calcul de l'approximation locale, ce qui nécessite de sur-évaluer la taille de ces fonctions de poids, et donc implique un contrôle moins local. Quand bien même de telles fonctions peuvent être définies sur le maillage, se pose la question de leur "épaisseur". La figure 2.2 illustre le problème : des fonctions de poids trop fines produisent une fonction dont l'apparence n'est pas assez lisse et une fonction trop épaisse risque d'interagir avec les surfaces proches de façon contre-intuitive. Dans tous les cas, il semble impossible d'éviter que des surfaces proches interagissent, ce qui est contraire au principe de *contrôle local*. De plus, cette approche possède comme défaut supplémentaire l'impossibilité de représenter des auto-intersections.

Plutôt que d'utiliser des fonctions de poids dans $\mathbb{R}^3$ comme dans les méthodes MLS, nous nous sommes dirigés vers une méthode utilisant des pondérations non seulement topologiques, mais *discrètes*. En d'autres termes, nous n'effectuons la projection qu'en des points précis de la surface à l'aide de poids discrets déterminés par la connectivité. Cela nous a conduit à intégrer l'opérateur de projection qui est au coeur des surfaces MLS directement dans un processus de subdivision, d'où le nom que nous avons donné à cette méthode : *Least Square Subdivision Surfaces* (LS^3). Il s'agit en quelque sorte de l'approche symétrique où nous partons des surfaces de subdivision pour les enrichir avec ce qui fait la force des méthodes MLS.

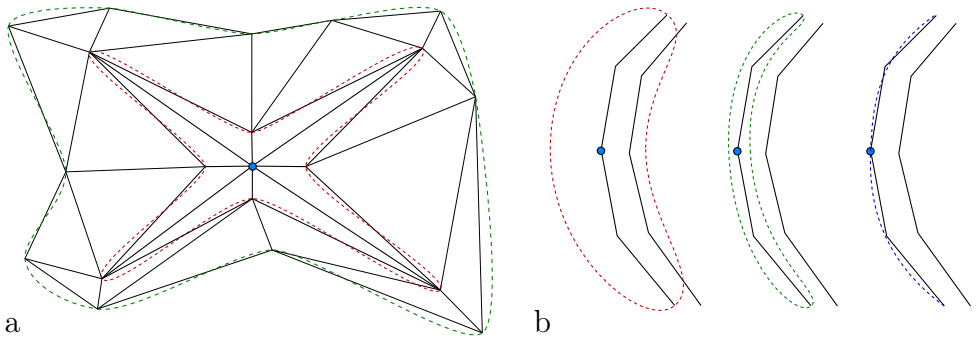


FIGURE 2.2 – **Difficultés pour trouver des pondérations dans $\mathbb{R}^3$.** (a) La fonction de poids associée au sommet bleu peut prendre une forme arbitrairement complexe. Ici, on souhaiterai avoir un poids nul au niveau de la courbe verte et un poids moyen sur la courbe rouge. (b) Quelle épaisseur donner au fonctions de poids ? Trop épais implique des repliements. Idéalement, la fonction de poids doit être d'épaisseur nulle pour éviter les interactions, mais cela est impossible (ou alors la fonction de poids définie à elle seule la surface).

2.1 L'operateur LS^3

La méthode LS^3 peut être vue comme une généralisation des surfaces de subdivision. Étant donné un maillage $\mathcal{M}_k$, l'opérateur LS^3 permet d'obtenir un maillage plus dense et plus lisse : $\mathcal{M}_{k+1}$. Si l'étape de découpage de notre méthode est identique à celle des surfaces de subdivision, notre étape de lissage se compose d'une *étape de relaxation* destinée à régulariser le maillage, et d'un *opérateur de projection* directement inspiré des techniques MLS. Ceci est illustré par la figure 2.3 où chaque sommet $\mathbf{d}_j^{k+1}$ (sans position) issu de l'étape de découpage est traité par ces deux étapes :

L'étape de relaxation qui a pour but de déplacer les sommets tangentiellement à la surface, de manière à obtenir des faces localement plus uniformes. Les coordonnées du sommet $\mathbf{r}_j^{k+1}$ issu de cette étape sont calculées à l'aide d'un barycentre pondéré d'un certain voisinage. Cela ressemble aux règles de lissage des surfaces de subdivision, mais les objectifs sont différents : les poids sont choisis dans l'unique but d'obtenir un maillage régulier plutôt qu'une surface très lisse. Les détails de cette étape sont présentés en section 2.1.1.

L'opérateur de projection qui est constitué de deux sous étapes : le calcul d'une approximation locale du maillage $\mathcal{M}_k$ par une surface simple $\mathcal{S}_j^{k+1}$, puis la projection du point $\mathbf{r}_j^{k+1}$ issu de l'étape de relaxation sur celle-ci pour obtenir le sommet final $\mathbf{v}_j^{k+1}$. Cette étape est détaillée en section 2.1.2.

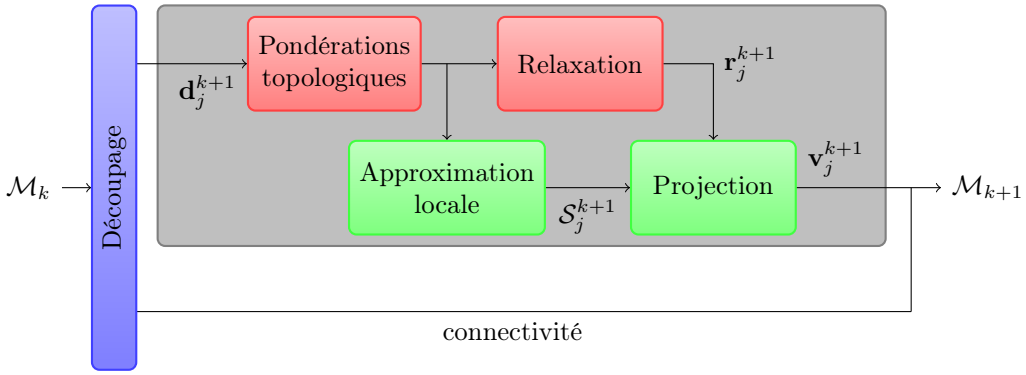


FIGURE 2.3 – **Aperçu de la méthode LS^3.** Le maillage $\mathcal{M}_k$ passe d'abord par l'étape de découpage chargée de modifier la topologie, ce qui produit un ensemble de sommets $\mathbf{d}_j^{k+1}$ sans position. Les étapes suivantes ont pour but de modifier la géométrie du maillage et s'appliquent à chaque sommet. L'étape de relaxation consiste à d'abord calculer les pondérations topologiques associées à $\mathbf{d}_j^{k+1}$, puis à effectuer la relaxation pour obtenir le sommet $\mathbf{r}_j^{k+1}$. L'étape de projection utilise les mêmes pondérations que l'étape de relaxation pour calculer l'approximation locale $\mathcal{S}_j^{k+1}$, sur laquelle est projeté $\mathbf{r}_j^{k+1}$ pour obtenir la position finale $\mathbf{v}_j^{k+1}$. L'ensemble des sommets obtenus forme le maillage $\mathcal{M}_{k+1}$.

En résumé, l'étape de relaxation peut être vue comme un lissage tangentiel et l'étape de projection comme un lissage de la forme. Comme nous le verrons, ce découplage de l'opérateur de lissage des surfaces de subdivision offre de nouvelles possibilités pour mettre au point des opérateurs dédiés et plus performants.

De prime abord, cette approche semble bien plus proche des concepts des surfaces de subdivision que des surfaces MLS. En fait, nous pouvons remarquer que comme pour la projection d'un point sur une surface MLS, avec notre méthode, itération après itération, les sommets s'approchent de leur propre approximation locale. Si cette procédure converge et qu'à la limite les approximations locales varient de façon continue, nous nous retrouvons alors avec une surface MLS et ses propriétés puisque la surface limite est constituée de l'ensemble des points qui appartiennent à leur approximation locale.

Il y a donc deux façons de voir notre méthode : la première consiste à la considérer comme une méthode de subdivision *non-linéaire*, la deuxième comme une méthode MLS particulière utilisant une fonction de poids *discrète*. Par la suite, nous expliquons le fonctionnement de notre approche comme celui d'une méthode de subdivision et utilisons les principes des MLS pour justifier nos choix.

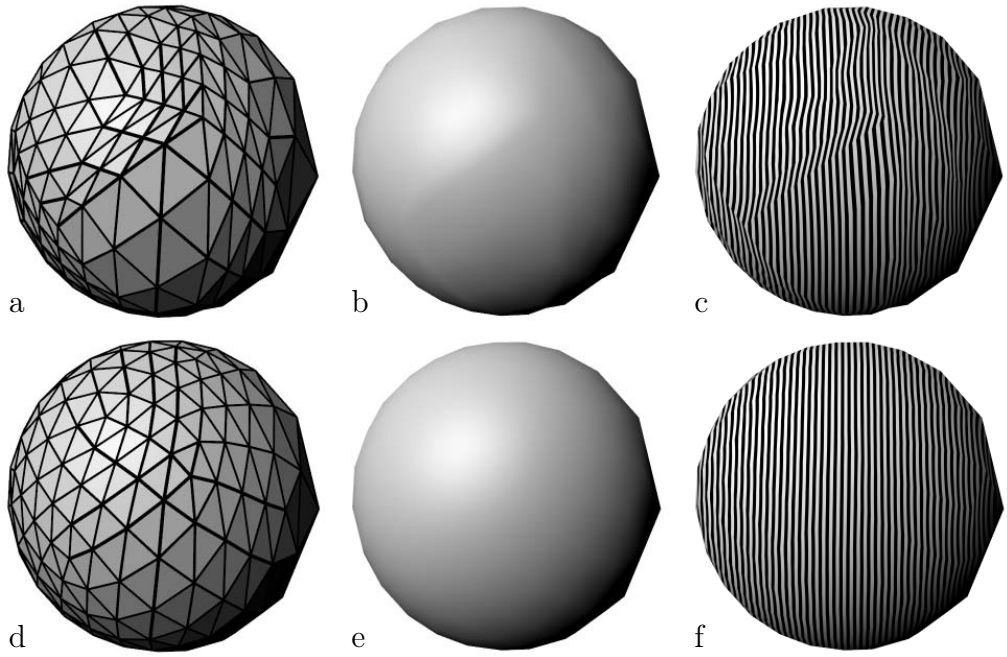


FIGURE 2.4 – **Importance de la relaxation pour le rendu.** Ces images sont issues d'un maillage représentant une sphère échantillonnée de façon irrégulière, raffinée sans relaxation en haut et avec relaxation en bas. Tous les sommets appartiennent à la sphère et les normales sont exactes.

En pratique, l'étape de relaxation (section 2.1.1) est similaire à de la subdivision classique. L'opérateur de projection vient donc s'ajouter en tant que lissage supplémentaire qui parvient à réduire considérablement certains artefacts comme nous le verrons en section 2.3.2. Nous utilisons les normales des sommets pour faciliter l'étape de projection (voir section 2.1.2), tout en ajoutant un contrôle supplémentaire obtenu en manipulant directement les normales (voir section 2.2). Finalement, nous proposons une méthode pour gérer également les bordures et les arêtes (semi-)vives en section 2.1.3.

2.1.1 L'étape de relaxation

L'étape de relaxation a pour but de produire des maillages localement plus uniformes en déplaçant les sommets sur le plan tangent. Cette étape est cruciale pour produire des surfaces de bonne qualité pour plusieurs raisons. Tout d'abord, de nombreuses applications fonctionnent mieux avec des maillages localement uniformes. Par exemple, comme illustré figure 2.4, les algorithmes de calcul de l'éclairage ont tendance à produire des comportements indésirables en cas de fortes variations de l'échantillonnage. Ensuite, cela permet de dépeindre les variations de courbure plus proprement. Finalement, comme nous le ver-

rons en section 2.1.2, cette étape s'avère également cruciale dans la génération des pondérations MLS.

Plus précisément, l'étape de relaxation doit idéalement :

- arranger les arêtes de sorte qu'elles produisent des courbes lisses et que leurs longueurs varient progressivement,
- répartir les voisins d'un sommet de façon à ce qu'ils tendent vers une ellipsoïde,
- préserver la forme originelle du maillage.

Une approche naturelle pour cela est de repositionner les sommets à l'aide d'une combinaison linéaire de leur voisinage. Cette façon de faire, combinée à une reprojection sur le plan tangent, est couramment utilisée pour faire du *lissage tangentiel*, c'est à dire du lissage avec préservation du volume [Botsch et Kobbelt, 2004]. Dans notre cas, l'étape de projection sur le plan tangent est difficile à réaliser puisque les nouveaux sommets n'ont ni position, ni plan tangent associé. Une telle étape est de toute façon peu utile en pratique puisque les sommets seront par la suite reprojetés sur leur approximation locale.

Concrètement, la position $\mathbf{r}_j^{k+1}$ est calculée à partir des sommets de $\mathcal{M}_k$:

$$\mathbf{r}_j^{k+1} = \sum_{i \in V_j^{k+1}}^{n} w_{i,j} \mathbf{v}_i^k \tag{2.1}$$

tel que $\sum_{i=1}^{n} w_{i,j} = 1$ où V_j^{k+1} est l'ensemble des sommets voisins du point d'insertion de $\mathbf{r}_j^{k+1}$ ainsi que lui même dans $\mathcal{M}_k$, et $w_{i,j}$ est le poids associé au sommet $\mathbf{v}_i^k$. Les poids doivent être choisis en accord avec les règles de découpage afin de considérer un voisinage le plus restreint possible. Comme nous l'avons vu en section 1.1.3, il est possible d'obtenir des masques de subdivision optimisés en fonction des propriétés désirées. Dans notre cas, nous souhaitons avoir un maillage le plus uniforme possible sans nous soucier de la qualité de la surface obtenue. Un choix logique est donc d'utiliser des optimisations éliminant les artefacts polaires. La majorité des exemples que nous présentons ici utilisent un découpage triangulaire diadique avec les pondérations du schéma de Loop modifiées par Barthe et Kobbelt [2004] pour éliminer les artefacts polaires (voir figure 1.10). Nous verrons en section 2.3 des résultats basés sur le schéma de Catmull-Clark et le schéma $\sqrt{3}$.

2.1.2 L'étape de projection

L'étape de projection a pour but de déplacer les sommets le long de la normale de la surface pour obtenir une surface lisse. Pour cela, nous utilisons un opérateur de projection inspiré des techniques utilisant les MLS : l'idée est de calculer une approximation locale $\mathcal{S}_j^{k+1}$ à l'aide de la méthode des moindres

carrés pondérés, puis de projeter le point $\mathbf{r}_j^{k+1}$ sur $\mathcal{S}_j^{k+1}$ pour obtenir sa position finale.

Pondération topologique discrète

Afin d'éviter les difficultés liées au calcul d'une fonction de poids lisse dans $\mathbb{R}^3$ approchant convenablement le maillage , nous proposons d'utiliser des *pondérations topologiques* discrètes définies à l'aide de masques en fonction de l'emplacement du sommet $\mathbf{r}_j^{k+1}$. De telles pondérations apportent de nombreux avantages. Leur simplicité permet de les calculer très efficacement et leur lien direct avec la connectivité du maillage permet de contrôler précisément l'influence des sommets. Comme pour les surfaces de subdivision, n'utiliser que le voisinage direct permet de minimiser la zone d'influence des sommets tout en évitant les cas particuliers qui peuvent survenir près des bordures avec de plus grand masques.

De façon similaire aux surfaces MLS, l'approximation doit être *locale* par rapport au point d'évaluation. Dans notre cas il s'agit du point $\mathbf{r}_j^{k+1}$ qui est ensuite projeté sur $\mathcal{S}_j^{k+1}$. Il semble donc naturel d'utiliser le même voisinage V_j^{k+1} pour calculer l'approximation locale que pour l'étape de relaxation. En pratique, en observant l'équation 2.1, on constate que, par construction, les pondérations $w_{i,j}$ utilisées pour l'étape de relaxation correspondent aux *coordonnées barycentriques* de $\mathbf{r}_j^{k+1}$ vis à vis des sommets de $\mathcal{M}_k$. Ces poids sont donc tout indiqués pour l'étape de projection.

L'approximation locale de la surface

Le calcul de l'approximation se fait à l'aide de la méthode des moindres carrés pondérés, comme pour les surfaces MLS. Cependant, contrairement aux surfaces MLS où il est toujours possible d'augmenter le rayon de la fonction de poids de façon à avoir suffisamment d'échantillons pour la procédure d'ajustement, nous utilisons des masques discrets prenant en compte un nombre restreint de sommets. Pour rester compatible avec la majorité des schémas de raffinement, tout en conservant la possibilité d'utiliser des voisinages minimaux, il nous faut donc une méthode capable de calculer l'approximation locale avec très peu d'échantillons : le raffinement $\sqrt{3}$ [Kobbelt, 2000] par exemple, insère des sommets au centre des faces triangulaires à l'aide d'un masque ne contenant que les trois sommets, comme nous l'avons vu en section 1.1.2. De plus le système à résoudre pour calculer l'approximation locale doit être sur-contraint pour éviter d'effectuer un ajustement exact, auquel cas les poids n'auraient aucune influence (voir en section 1.2.3). La position des sommets seule n'est donc pas suffisante pour permettre de calculer une approximation locale stable dans de nombreux cas. Par exemple, un plan requiert au minimum quatre sommets pour être sur-contraint, et pour une sphère cinq sommets

sont nécessaires. Comme nous l'avons vu dans la section 1.2.2, les méthodes MLS s'appuyant sur les informations de normales sont bien plus performantes, que ce soit en termes de coût de calculs, de stabilité, ou de qualité générale des surfaces obtenues. L'utilisation des normales des sommets voisins est donc tout indiquée à notre problématique. Intuitivement, un 1-voisinage équipé de l'information de normales permet d'obtenir une information similaire à un 2-voisinage, sans les difficultés associées.

Dans la littérature des MLS, deux procédures d'ajustement exploitant les normales ont été proposées. La première ajuste des plans et est extrêmement simple, mais n'a aucun effet dans notre cas : par construction, un plan approché au sens des moindres carrés passe nécessairement par le barycentre des sommets approchés qui s'avère être le point $\mathbf{r}_j^{k+1}$. Par conséquent, l'étape de projection n'a aucun effet et la surface obtenue correspond à une surface de subdivision linéaire classique dans le cas où les pondérations de l'étape de relaxation ont été choisies comme telles. Néanmoins, cette observation reste intéressante puisqu'elle montre que notre méthode est bien une généralisation des surfaces de subdivision. Cette remarque nous conforte encore un peu plus dans le fait de choisir les mêmes poids pour les étapes de relaxation et de projection.

La deuxième procédure, qui approche des sphères algébriques, et que nous appelons *Algebraic Sphere Fitting* (ASF), permet de calculer une sphère à partir de seulement deux échantillons (position plus normale) et s'avère donc robuste avec tous les schémas de raffinement que nous avons considérés jusque là, d'où notre choix de cette méthode.

Ajustement des sphères algébriques

La procédure ASF a été proposée par Guennebaud *et al.* [2008] pour approcher des sphères algébriques définies comme étant la *0-isosurface* d'un champ scalaire $s(\mathbf{x}) = [1, \mathbf{x}^T, \mathbf{x}^T\mathbf{x}]\mathbf{u}$, où $\mathbf{u} \in \mathbb{R}^{d+2}$ est un vecteur de paramètres définissant la forme de la sphère, et d est la dimension de l'espace ambiant (dans notre cas, $d = 3$). Cette représentation permet à la sphère de naturellement dégénérer en un plan, ce qui est nécessaire pour approcher des régions planaires et gérer les points d'inflexion de façon robuste. Comme expliqué en section 1.2.2, l'idée clé des procédures d'approche utilisant des normales est de d'abord calculer les paramètres $[u_1, \ldots, u_{d+1}]$ de façon à minimiser l'erreur pondérée entre ∇s et les normes des sommets. Le coefficient constant u_0 est obtenu en minimisant la contrainte de distance standard ($s(\mathbf{p}_i) = 0$). Cela

conduit à une forme close rendant la procédure d'approche très efficace :

$$\begin{aligned} u_{d+1} &= \beta\frac{1}{2}\frac{\sum w_i\mathbf{p}_i^T\mathbf{n}_i \;-\; \tilde{\mathbf{p}}^T\sum w_i\mathbf{n}_i}{\sum w_i\mathbf{p}_i^T\mathbf{p}_i \;-\; \tilde{\mathbf{p}}^T\sum w_i\mathbf{p}_i} \\ \begin{bmatrix} u_1 \\ \vdots \\ u_d \end{bmatrix} &= \sum w_i\mathbf{n}_i - 2u_{d+1}\tilde{\mathbf{p}} \\ u_0 &= -[u_1 \ldots u_d]\tilde{\mathbf{p}} \;-\; u_{d+1}\sum w_i\mathbf{p}_i^T\mathbf{p}_i \end{aligned} \tag{2.2}$$

où w_i sont les poids associés aux sommets $\mathbf{p}_i$ (tel que $\sum w_i = 1$), et $\tilde{\mathbf{p}}$ est le barycentre $\sum w_i\mathbf{p}_i$. Le paramètre β permet de passer d'un ajustement de sphères standard lorsque $\beta = 1$ à un fit de plan avec $\beta = 0$ de façon continue. Cela permet dans notre cas de passer progressivement d'une surface LS^3 à une de surface de subdivision linéaire.

La projection d'un point sur une sphère algébrique est effectuée en la convertissant préalablement en une sphère explicite ou en un plan, respectivement si $|u_{d+1}| > \epsilon$ ou $u_{d+1} = 0$. Si u_{d+1} est proche de zéro, alors quelques itérations de Newton sont nécessaires pour obtenir une bonne stabilité numérique [Guennebaud *et al.*, 2008].

2.1.3 Bordures et arêtes vives

Les surfaces ouvertes ou présentant des arêtes vives sont fréquemment utilisées en modélisation. Les bordures peuvent servir à représenter des objets fins, comme des vêtements, et les arêtes vives sont tout simplement extrêmement communes, que ce soit pour de la modélisation organique ou pour représenter des objets manufacturés. Il est donc indispensable que notre méthode puisse gérer ces caractéristiques proprement.

Dans le cas des surfaces de subdivision, les bordures sont traitées très simplement avec des masques adaptés, qui produisent généralement une courbe spline classique en n'utilisant que les sommets en bordure. Cela permet aux bordures d'être indépendantes des sommets internes. En juxtaposant deux maillages partageant la même bordure, les bordures des surfaces correspondantes sont donc assurées de coïncider sans aucune contrainte de lisseur entre les deux ce qui produit une arête vive.

En revanche, dans notre cas, un effet secondaire de l'opérateur de projection est de décaler la surface de façon à la rapprocher des points de contrôle (voir section 2.3.2). Si les schémas de bordure des surfaces de subdivision sont utilisés directement, cela conduit à des résultats incohérents comme illustré figure 2.5.b. En laissant de côté le problème des arêtes vives, et en se concentrant sur les bordures, il serait envisageable d'utiliser les sommets internes pour le calcul d'une approximation locale aux bords, mais les poids ne seraient du

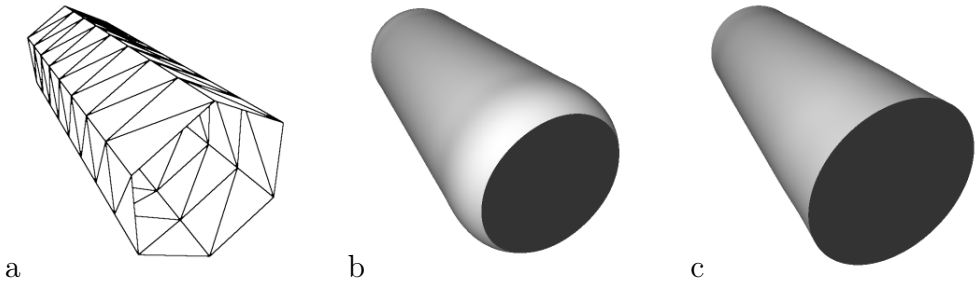


FIGURE 2.5 – **Gestion des bordures avec LS^3.** Le maillage (a) raffiné avec notre méthode en utilisant des splines pour les bordures produit un décalage entre les bordures et la surface (b). Utiliser l'étape de projection avec les masques en bordure corrige le problème tout en étant plus cohérent (c).

coup plus cohérents avec ceux de l'étape de relaxation. En fait, les normales des sommets en bordure apportent déjà suffisamment d'informations sur la surface voisine d'une bordure et comme la procédure ASF est sur-contrainte avec seulement deux échantillons, il est donc tout à fait possible de l'utiliser avec les masques de bordure classiques. La figure 2.5.c illustre les résultats produits par cette approche.

Ce traitement est suffisant pour les bordures, cependant cela complique le traitement des arêtes vives car le calcul des bordures dépend des normales des sommets en bordure, et celles-ci sont supposées être différentes de chaque coté d'une arête vive. Il n'est donc plus possible de simplement juxtaposer deux maillages ouverts. Nous proposons donc la solution suivante : lorsque nous traitons un sommet appartenant à une arête vive, nous calculons deux approximations locales $\mathcal{S}_\mathrm{l}$ et $\mathcal{S}_\mathrm{r}$ correspondant aux deux côtés, puis nous projetons le sommet sur leur *intersection* $\mathcal{S}_l \cap \mathcal{S}_r$, comme illustré figure 2.6. Cela permet de créer une arête vive cohérente avec les deux surfaces comme illustré figure 2.7.

Finalement, il est également possible de représenter des "arêtes douces" à l'aide de méthodes existantes telle que celle décrite par DeRose *et al.* [1998]. Cette méthode consiste simplement à considérer les arêtes comme vives durant quelques pas de subdivision, puis à les traiter normalement par la suite. Les résultats obtenus en utilisant cette méthode avec les surfaces LS^3 sont similaires à ceux des surfaces de subdivision (voir figure 2.7).

2.2 Normales

Une caractéristique de notre approche est l'utilisation des normales des sommets voisins pour le calcul des approximations locales. Par conséquent,

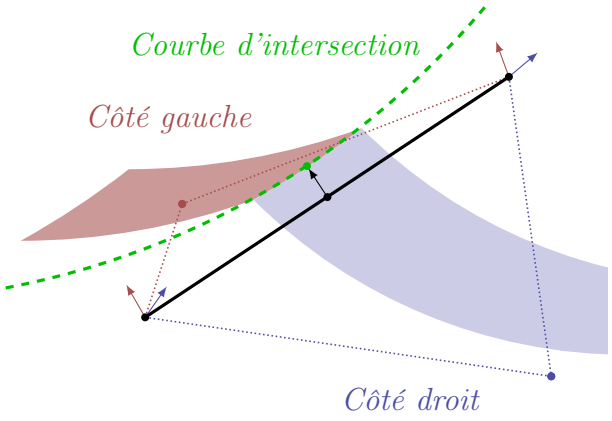


FIGURE 2.6 – **Approximations locales des arêtes vives.** Le sommet issu de la relaxation est projeté sur l'intersection des approximations locales à gauche et à droite.

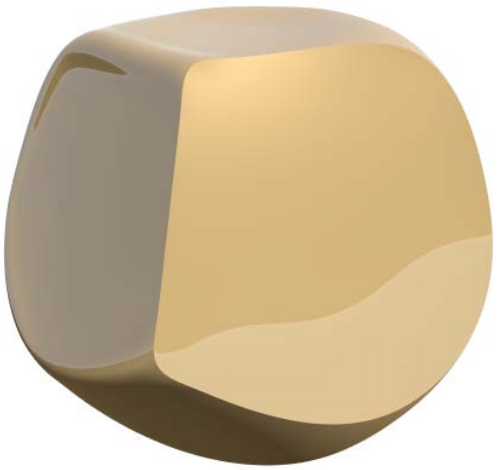


FIGURE 2.7 – **Arêtes vives et semi-vives.** Le modèle d'origine est un dodécaèdre dont les arêtes ont été marquées comme vives (à droite) ou semi-vives (à gauche).

le résultat final est influencé par celles-ci. Leur importance est d'autant plus grande du fait que la procédure d'approximation par des sphères est tout aussi sensible aux modifications des normales qu'au déplacement des sommets. Dans cette section, nous étudions comment les normales peuvent être utilisées pour manipuler la surface ainsi que les différentes méthodes permettant de les calculer automatiquement, avec leurs avantages et leurs inconvénients respectifs.

2.2.1 Méthodes d'estimation des normales

Il existe de nombreuses méthodes pour calculer les normales des sommets d'un maillage [Jin *et al.*, 2005]. Supposons que nous voulons calculer la normale $\mathbf{n}$ d'un sommet de coordonnée $\mathbf{q}$ et dont les coordonnées des sommets voisins sont $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{n-1}$ dans l'ordre anti-horaire. Afin de simplifier les équations suivantes, nous pouvons nous centrer sur le sommet $\mathbf{q}$ en considérant les coordonnées $\bar{\mathbf{p}}_j = \mathbf{p}_j - \mathbf{q}$. Nous rappelons ici les méthodes de calcul des normales les plus communes, qui, comme nous allons le voir, peuvent toutes être ramenées à une moyenne pondérée des normales des faces adjacentes.

Pondération par les aires. Cette méthode consiste à pondérer les normales par l'aire des triangles correspondants. Il s'agit de la méthode la plus simple à mettre en œuvre car les normales, générées par le produit vectoriel de deux arêtes d'un triangle, ont une norme proportionnelle à l'aire de celui-ci, ce qui nous amène à la formule suivante (en considérant les indices modulo n) :

$$\tilde{\mathbf{n}} = \sum_{j=0}^{n-1} \bar{\mathbf{p}}_j \times \bar{\mathbf{p}}_{j+1} \tag{2.3}$$

où $\times$ est le produit vectoriel et le vecteur $\tilde{\mathbf{n}}$ correspond à la normale non normalisée. Cette méthode est la plus rapide à calculer et la plus utilisée en pratique.

Pondération par les angles. Cette fois, les normales sont pondérées par les angles des faces adjacentes au sommet calculé :

$$\tilde{\mathbf{n}} = \sum_{j=0}^{n-1} \cos^{-1}(\bar{\mathbf{p}}_j \cdot \bar{\mathbf{p}}_{j+1}) \frac{\bar{\mathbf{p}}_j \times \bar{\mathbf{p}}_{j+1}}{\|\bar{\mathbf{p}}_j \times \bar{\mathbf{p}}_{j+1}\|} \tag{2.4}$$

où $\cdot$ est le produit scalaire. Cette méthode est peu utilisée en pratique car elle est plus coûteuse en calcul que la pondération par les aires et ne présente pas d'intérêt particulier en comparaison.

Méthode de Max [1999]. Cette méthode est conçue pour reproduire la normale de la sphère passant par le sommet et ses voisins (si celle-ci existe) :

$$\tilde{\mathbf{n}} = \sum_{j=0}^{n-1} \frac{\bar{\mathbf{p}}_j \times \bar{\mathbf{p}}_{j+1}}{\|\bar{\mathbf{p}}_j\|^2 \|\bar{\mathbf{p}}_{j+1}\|^2} \tag{2.5}$$

Cette méthode peut être intéressante dans le cas où l'on souhaite reconstruire la normale exacte d'une sphère. Cependant, dans le cadre du rendu de maillages peu denses, elle ne produit pas de résultats particulièrement supérieurs à une autre méthode et on lui préfère la pondération par les aires, plus rapide.

Ces méthodes sont trivialement invariantes aux translations et aux rotations, puisqu'elles reposent uniquement sur les normales, les aires et les angles du maillage qui sont eux-même invariants à ces transformations. Ce n'est par contre pas le cas des mises à l'échelle non-uniformes, qui sont pourtant des opérations communes lors de l'édition de surfaces. Il est facile de montrer que la méthode de pondération par les aires présente l'avantage d'être invariante aux transformations affines.

2.2.2 Calcul des normales pour la méthode LS^3

Nous avons trois types de normales à prendre en considération avec notre méthode : les *normales en entrée* qui sont soit calculées par un algorithme soit définies à la main, les *normales intermédiaires* qui sont produites en sortie d'un niveau de subdivision pour servir en entrée du niveau suivant, et les *normales en sortie* qui servent principalement au rendu.

Choix des normales en entrée

Deux options sont possibles pour les normales en entrée : les éditer manuellement ou les calculer automatiquement. En pratique, les éditer entièrement manuellement est rarement envisageable car les sommets sont souvent trop nombreux. Par contre, il peut être intéressant d'éditer quelques normales pour corriger localement la surface. Reste à choisir la méthode la plus adaptée pour calculer les normales en entrée.

Supposons que le maillage $\mathcal{M}_0$ est un cube. Les faces et les arêtes étant toutes identiques et régulières, les différentes pondérations proposées pour calculer les normales donnent le même résultat (en supposant que le cube ne soit pas triangulé) qui s'avère correspondre aux normales de la sphère circonscrite. Dans ce cas, la procédure ASF produit exactement la sphère circonscrite au cube. En choisissant judicieusement la méthode de calcul des normales intermédiaires comme expliqué ci-dessous, celles-ci sont aussi celles de la sphère circonscrite. Par conséquent, la surface limite s'avère être cette sphère.

La figure 2.8 illustre le résultat des trois méthodes de calcul des normales en entrée avec la méthode LS^3 sur un parallélépipède qui n'est autre que notre cube déformé par une mise à l'échelle non-uniforme. La méthode de pondération par les aires calcule dans ce cas précis les normales de l'ellipsoïde correspondant à la sphère circonscrite du cube après transformation. Le résultat

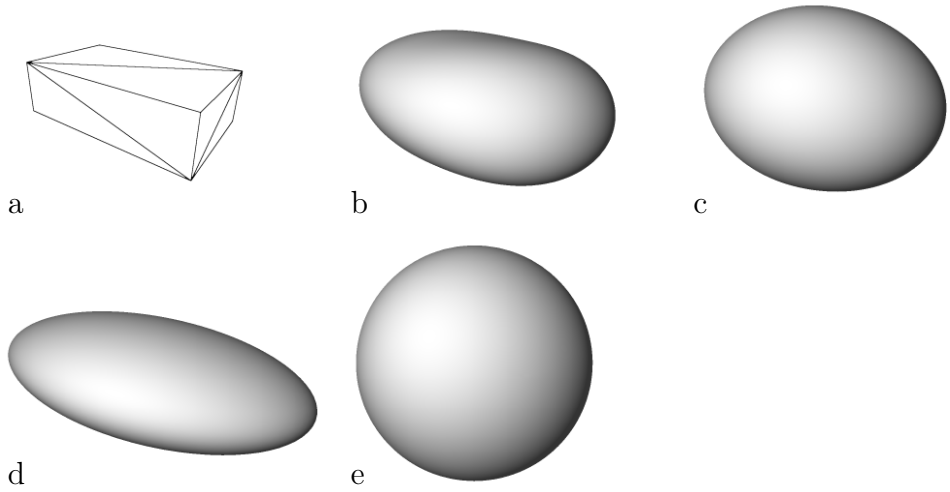


FIGURE 2.8 – **Différentes méthodes de calcul des normales.** Le maillage (a) peut être vu comme un cube ayant subi une mise à l'échelle non-uniforme, auquel cas, l'ellipsoïde (d) est la sphère circonscrite du cube ayant subi la même transformation. Les autres images correspondent au résultat de la méthode LS^3en calculant les normales initiales à l'aide d'une pondération par les aires (b), les angles (c) ou avec la méthode de Max [1999] (e). Toutes les images sont à la même échelle et prises du même point de vue.

est donc logiquement une surface assez proche de cet ellipsoïde comme illustré figure 2.8.b. La méthode de Max produit les normales de la sphère circonscrite du parallélépipède. La procédure ASF produit donc exactement la sphère circonscrite, par conséquent la surface limite est à nouveau une sphère (figure 2.8.e). Finalement, la méthode de pondération par les angles se situe à mi-chemin entre les deux pondérations précédentes (figure 2.8.c). Cela s'explique par le fait que la méthode de Max et la pondération par les aires produisent respectivement des normales proches de celles des petites faces et des grandes faces, alors que la pondération par les angles produit un équivalent de bissectrice en 3D et ne favorise aucune face.

La méthode de Max est utile dans les applications où il est nécessaire de reconstruire des sphères parfaites, ce qui n'est pas commun dans le domaine qui nous intéresse. Par contre, elle produit clairement un résultat contre-intuitif lors des mises à l'échelle non-uniformes. Son utilisation est donc déconseillée dans un cadre général.

La méthode de pondération par les angles ne possède pas de propriété particulière qui la rende intéressante et est plus compliquée à calculer que la pondération par les aires ; elle est donc, elle aussi, déconseillée.

La méthode de pondération par les aires semble la plus intéressante : son

invariance aux transformations affines garantit que les surfaces produites auront un comportement cohérent vis à vis de ces transformations. De plus, il s'agit de la méthode la plus simple et efficace à calculer.

Choix des normales intermédiaires

Après chaque étape de subdivision, de nouveaux sommets ont été ajoutés, il est donc indispensable de leur assigner une normale afin de pouvoir effectuer la projection lors de l'étape de subdivision suivante. Ces normales intermédiaires, comme les autres, ont une influence directe sur les surfaces obtenues avec notre méthode.

Une première solution consiste à recalculer les normales à l'aide d'une des méthodes proposées entre chaque pas de subdivision. Cependant cette méthode est lourde car il n'est pas possible de calculer les normales de cette façon pendant l'étape de lissage puisque celle-ci nécessite de connaître la position des sommets voisins. Il faut donc effectuer des passes de mise à jour des normales qui viennent s'intercaler entre chaque pas de subdivision. De plus, dans l'éventualité ou les normales en entrée auraient été éditées, celles-ci se retrouveraient ignorées juste après le premier pas de subdivision, ce qui réduirait considérablement leur impact.

Une bien meilleure stratégie consiste à calculer les normales directement durant l'étape de lissage. Deux variantes sont possibles pour cela : utiliser une moyenne pondérée des normales des sommets voisins, ou directement prendre la normale de l'approximation locale. Cette dernière méthode semble plus intéressante car, si les normales et les sommets en entrée appartiennent à une sphère, elle garantit de la reproduire. Elle n'est de toute façon pas plus coûteuse que l'autre, puisque le vecteur normal doit être calculé pour effectuer la projection.

Choix des normales en sortie

La stratégie que nous suggérons pour le calcul des normales intermédiaires a pour effet secondaire de les lisser fortement ; les normales obtenues en sortie reflètent donc assez mal la surface réelle et doivent être recalculées. L'étape de relaxation a pour effet d'uniformiser le voisinage, ce qui réduit fortement l'influence des différentes pondérations existantes. De plus, après quelques pas de subdivision, le lissage fait que les normales des faces adjacentes tendent à se rapprocher. La différence entre deux stratégies de pondération devient donc minime après quelques pas de raffinements ; par conséquent, nous suggérons d'utiliser la méthode la plus simple et efficace, à savoir la pondération par les aires.

FIGURE 2.9 – **Contrôle manuel des normales.** Le maillage (a) est raffiné avec notre méthode en utilisant les normales calculées automatiquement (b) et après édition manuelle des normales (c).

Résumé du choix des normales

En conclusion, nous avons jugé préférable d'utiliser la pondération par les aires en entrée et en sortie, et la normale des approximations locales au point de projection pour les normales intermédiaires. Sauf mention contraire, tous les résultats présentés utilisent cette combinaison.

Il est possible d'influencer le résultat en modifiant les normales comme illustré figure 2.9.

2.3 Résultats et discussions

2.3.1 Implémentation et performances

L'implémentation de la méthode LS^3 est assez directe. En pratique, il est très simple d'adapter un code de subdivision existant pour qu'il gère notre méthode, puisqu'il suffit d'ajuster le calcul des poids des masques pour optimiser uniquement la régularité du maillage et de rajouter le calcul de l'approximation locale et la projection à l'étape de lissage. Nous avons mis en oeuvre notre méthode dans le logiciel libre Meshlab ainsi que dans la bibliothèque CGAL. Le pseudocode correspondant à notre méthode est présenté figure 2.10.

Comparativement aux surfaces de subdivision, notre méthode est bien évidement plus lente puisqu'elle ajoute des étapes de calcul. Le calcul de l'approximation locale se fait en $O(n)$, avec n le nombre de sommets voisins, ce qui est la même complexité que l'étape de relaxation. La projection, quant à elle, s'effectue en $O(1)$. La complexité des surface LS^3 est donc du même ordre que celle des surfaces de subdivision.

Notre implémentation sous Meshlab de la méthode LS^3 avec les masques de Loop [1987] modifiés à l'aide de la méthode de Barthe et Kobbelt [2004] est environ 10% plus lente que la méthode de Loop originale. En pratique, cette implémentation n'est pas optimisée et la majorité du code de subdivision

pour chaque sommet $\mathbf{v}_i^0$ de $\mathcal{M}_0$ **faire**
  calculer sa normale initiale $\mathbf{n}_i^0$ à l'aide de la méthode de pondération par les aires
fin pour
pour chaque pas de subdivision k **faire**
  pour chaque sommet et arête o de $\mathcal{M}_{k-1}$ **faire**
    règle de découpage :
    • créer une nouveau sommet $\mathbf{d}_j^k$ associé à o
    règle de relaxation :
    • calculer sa position intermédiaire $\mathbf{r}_j^k$ à l'aide
      d'une moyenne pondérée en utilisant les masques de la figure 1.10
    opérateur de projection :
    • calculer une sphère algébrique $\mathcal{S}_j^k$ à l'aide de l'équation (2.2) et des masques de la figure 1.10
    • assigner au point $\mathbf{v}_j^k$ la projection de $\mathbf{r}_j^k$ sur $\mathcal{S}_j^k$
    • calculer la normale $\mathbf{n}_j^k$ à partir du gradient de $\mathcal{S}_j^k$ en $\mathbf{v}_j^k$
  fin pour
  mettre à jour la topologie pour produire $\mathcal{M}_k$
fin pour
pour chaque sommet $\mathbf{v}_i^n$ de $\mathcal{M}_n$ **faire**
  re-calculer sa normale finale $\mathbf{n}_i^n$ avec la méthode de pondération par les aires
fin pour

FIGURE 2.10 – **Pseudocode de la méthode LS^3.** Pour plus de clarté, le code présenté suppose que les masques sont ceux de Loop modifiés.

est commun à ces deux méthodes, ce qui peut expliquer un si faible écart de performances.

2.3.2 Comparaison avec les surfaces de subdivision

Dans le domaine de la création artistique, nous nous intéressons principalement à l'apparence des surfaces. Cette notion est assez subjective et dépend du contexte d'utilisation. La comparaison entre les surfaces de subdivision et la méthode LS^3 est donc délicate. Nous proposons une série d'exemples où les surfaces de subdivision ont un comportement indésirable et observons le comportement de notre méthode sur le même cas :

Les artefacts polaires sont inexistants car nous utilisons des masques optimisés pour les éliminer avec la méthode de Barthe et Kobbelt [2004]. En pratique, nous utilisons les poids qu'ils ont calculés dans leur article pour la méthode de Loop. Des pondérations similaires pourraient être dérivées pour les autres schémas de raffinement comme Catmull-Clark ou $\sqrt{3}$, cependant les exemples que nous présentons avec ces méthodes utilisent les pondérations d'origine.

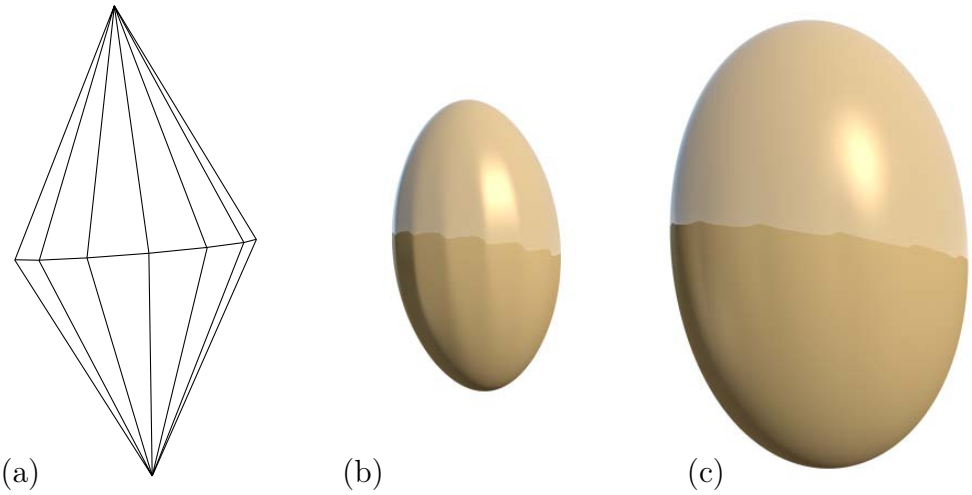


FIGURE 2.11 – **Comparaison des effets de réduction et des oscillations.** Le maillage d'origine (a) raffiné avec la méthode de Loop (b) et LS^3 (c). Les oscillations sont moins marquées avec notre méthode et l'étape de projection permet de compenser le phénomène de réduction des surfaces de subdivision approchantes.

La continuité des surfaces LS^3 ne peut pas être analysée avec les outils développés pour les surfaces de subdivision du fait que notre opérateur de projection n'est pas linéaire. La section 2.4 est dédiée à cette étude.

Les oscillations autour des sommets de forte valence sont fortement réduites sans être totalement éliminées comme illustré figure 2.11.

Les déformations produites par les sommets extraordinaires sont très fortement réduites. La figure 2.12 présente un exemple synthétique de point selle. Le résultat de notre méthode est quasiment indistinguable de la quadrique de référence utilisée. La figure 2.13 présente des modèles plus proches de cas d'utilisation réels. Le gain est moins important – les artefacts étant à la base moins visibles – mais les artefacts sont tout de même réduits de manière significative. En pratique, nous pensons que la capacité de notre méthode à réduire ces artefacts constitue un de ses principaux intérêts.

Les surfaces de subdivision ont la propriété de support compact, à savoir qu'un sommet n'influence la surface que sur un voisinage fini. Dans les cas des schémas de Loop et de Catmull-Clark, cette influence s'étend sur un voisinage de deux anneaux. Cependant, même en utilisant les mêmes masques, notre méthode possède des fonctions de base légèrement plus étendues. En effet, notre méthode ne considère pas seulement la position des sommets, mais aussi les normales, ce qui signifie que la modification d'un sommet entraîne également la modification des normales des sommets voisins. Cette modification est pro-

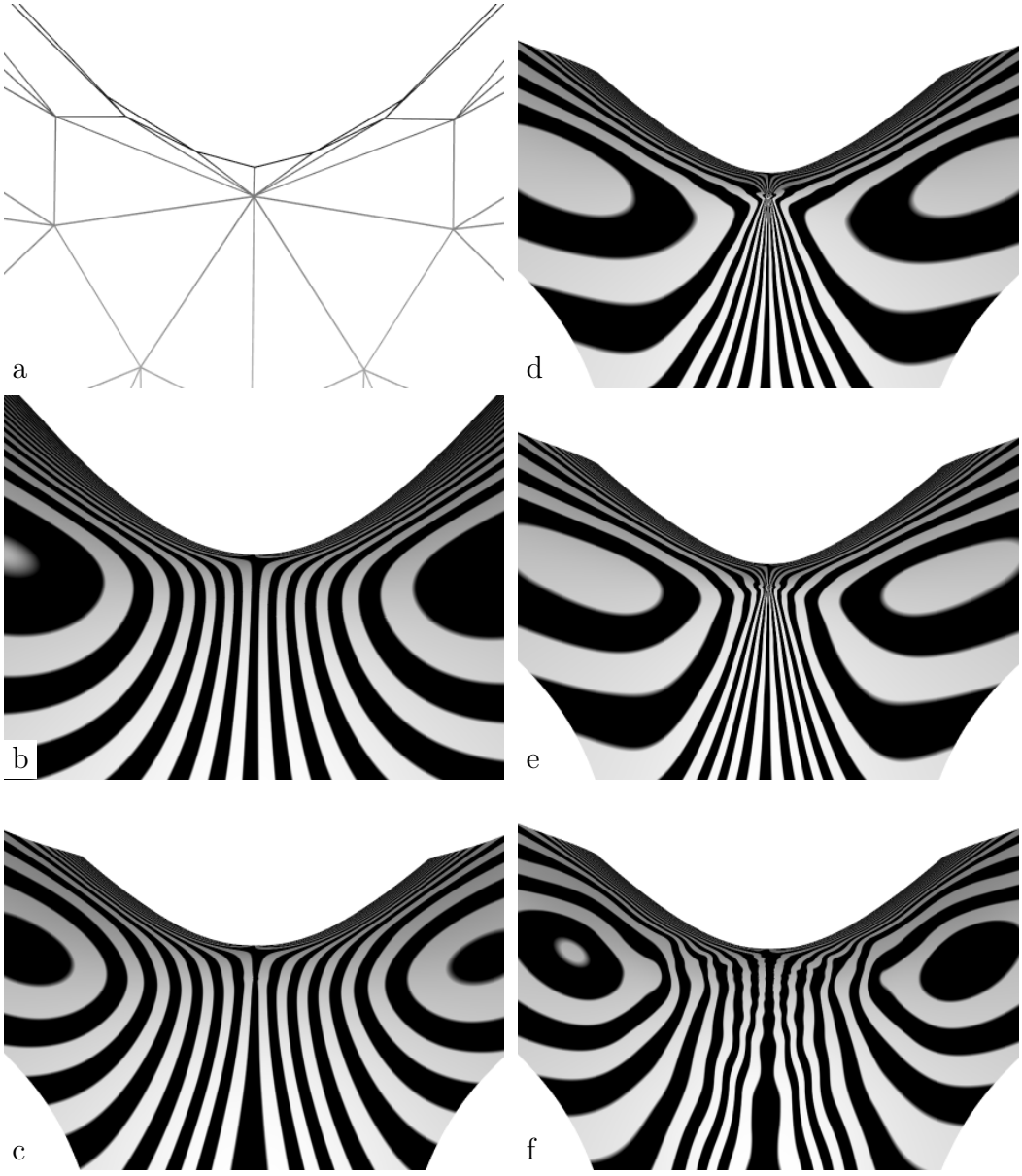


FIGURE 2.12 – **Reduction des déformations.** Le modèle de point selle (a) correspondant à une quadrique de référence (b) est raffiné en utilisant la méthode de Loop (d) et ses versions modifiées pour éliminer les artefacts polaires (e) et améliorer la courbure (f). La méthode LS^3 produit une surfaces très proche de la quadrique de référence (c) et seul un léger artefact est visible à l'emplacement du sommet extraordinaire.

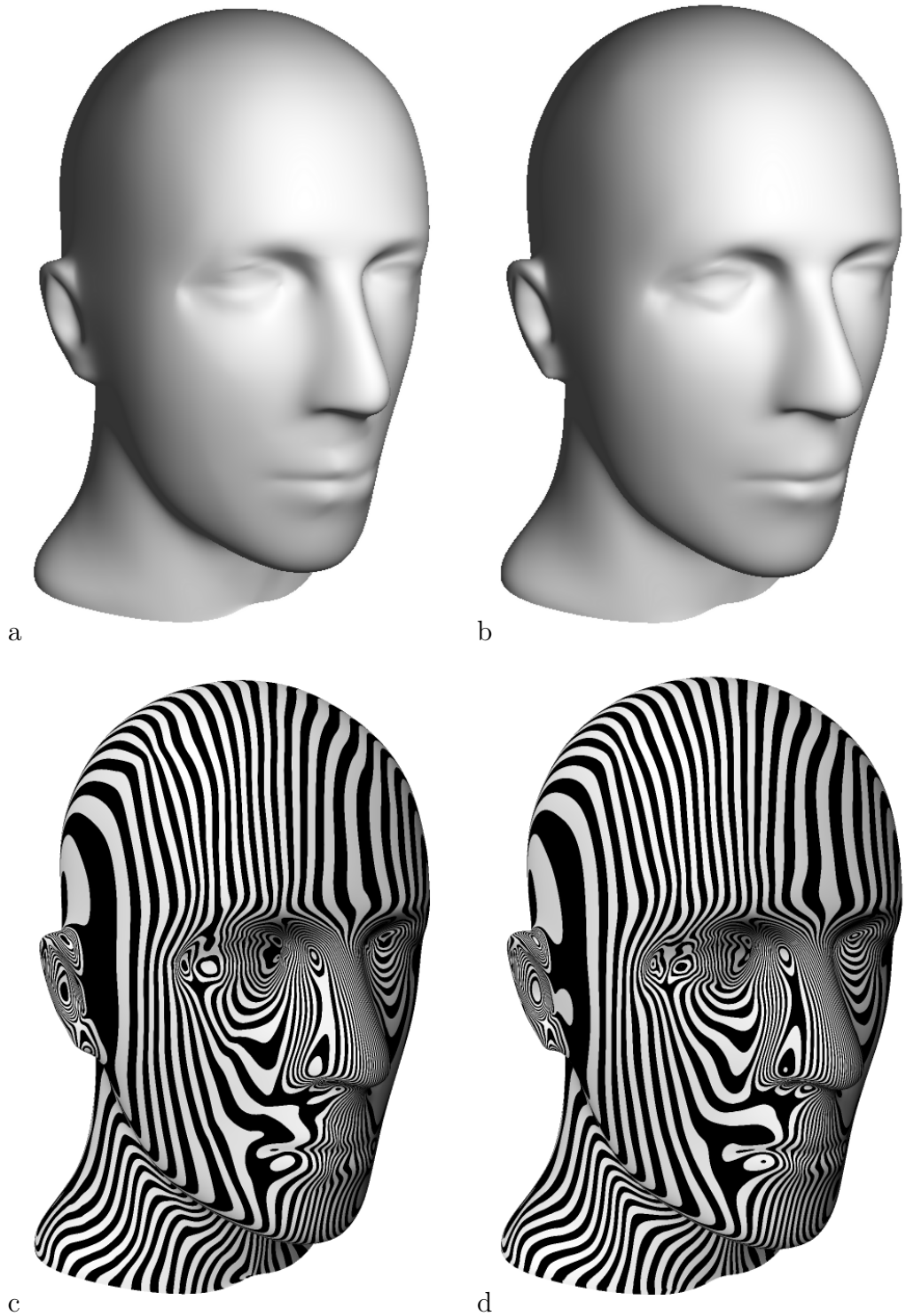


FIGURE 2.13 – **Reduction des artefacts sur un modèle complexe**, particulièrement au coin de l'œil et au niveau des lèvres.

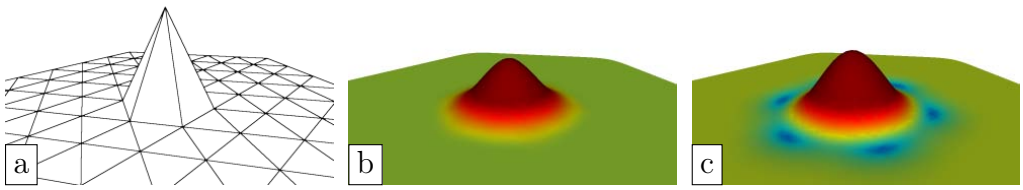


FIGURE 2.14 – **Zone d'influence d'un sommet.** Le maillage (a) est subdivisé avec la méthode de Loop (b) et LS^3 (c). La couleur correspond à l'altitude. Notez que LS^3 possède une zone d'influence légèrement supérieure à Loop.

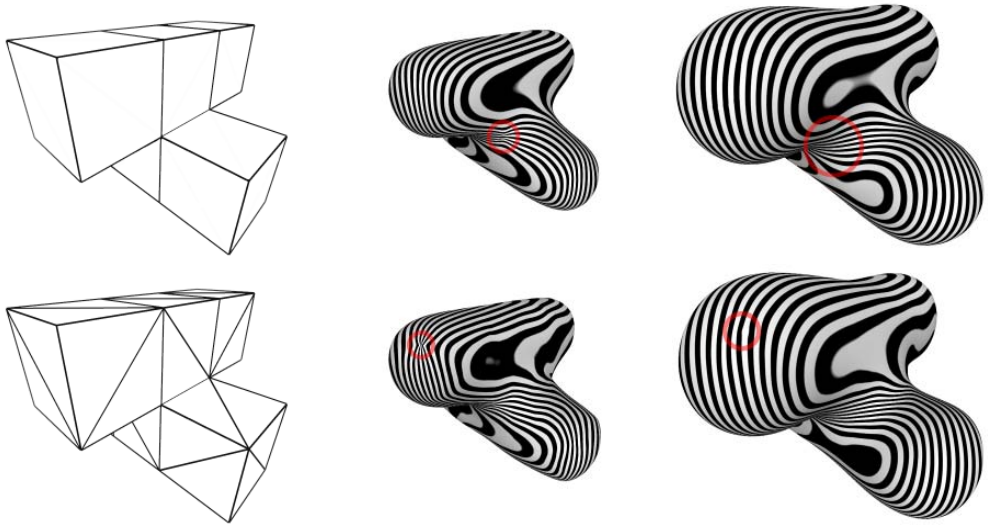


FIGURE 2.15 – **LS^3 avec Catmull-Clark et $\sqrt{3}$.** Les maillages (à gauche) sont raffinés avec des surfaces de subdivision classiques (au milieu) et avec notre méthode (à droite). La ligne du haut correspond au schéma de Catmull-Clark et celle du bas à $\sqrt{3}$.

pagée au fur et à mesure des étapes de subdivision jusqu'à atteindre à la limite un voisinage de trois anneaux. Ce phénomène est illustré figure 2.14.

La figure 2.15 illustre la généralité de notre méthode en utilisant un raffinement diadique quadrangulaire (et les pondérations de Catmull-Clark), et un raffinement $\sqrt{3}$.

Interpolation et approximation

Les surfaces de subdivision approchantes utilisent généralement des pondérations positives. Cela signifie que la nouvelle position des sommets se trouve systématiquement dans l'*enveloppe convexe* des sommets impliqués dans les masques de subdivision. Par conséquent, les surfaces de subdivision approchantes classiques sont inclues dans l'enveloppe convexe de leur polygone de

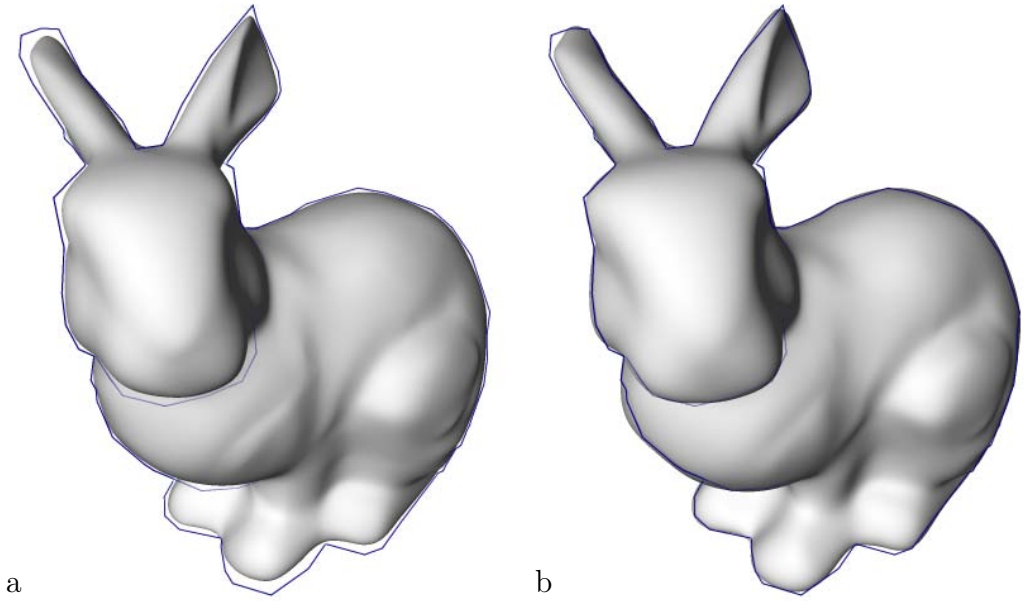


FIGURE 2.16 – **Comparaison des silhouettes** entre la méthode de Loop (a) et LS^3 (b). La ligne bleue correspond à la silhouette du maillage de contrôle.

contrôle. Cela produit un effet secondaire : les surfaces obtenues sont souvent beaucoup plus petites que le maillage d'origine, particulièrement dans les zones de fortes courbures, comme nous pouvons le constater figure 2.11.b.

Au contraire, notre opérateur de projection permet aux sommets de sortir de l'enveloppe convexe des masques utilisés. L'approximation locale passant au plus proche possible des sommets, cela a pour effet de rapprocher la surface des points de contrôle, et donc de mieux préserver le volume initial que les surfaces de subdivision approchantes (figure 2.11.c). En pratique, les surfaces LS^3 se situent entre les surfaces de subdivision approchantes et interpolantes.

Cet effet est intéressant lorsqu'on souhaite raffiner un maillage existant qui n'a pas été conçu pour cela. Un exemple typique est le raffinement de modèles dans les anciens jeux vidéo pour améliorer leur qualité. Dans une telle situation, et plus particulièrement en présence de maillages très grossiers, l'effet de réduction des surfaces de subdivision est trop important. En contrepartie, les surfaces interpolantes génèrent des oscillations disgracieuses, et peuvent même produire un léger grossissement des surfaces. Les surface LS^3 permettent de créer des surfaces de qualité tout en produisant des silhouettes très proches de celles des maillages de contrôle, comme illustré figure 2.16.

Un effet indésirable, issu du fait que les surfaces LS^3 puissent sortir de l'enveloppe convexe, est l'apparition de bourrelets survenant lors de l'utilisation d'arêtes semi-vives, comme illustré figure 2.17. Les arêtes semi-vives sont générées en considérant l'arête comme vive lors des premiers pas de subdivision

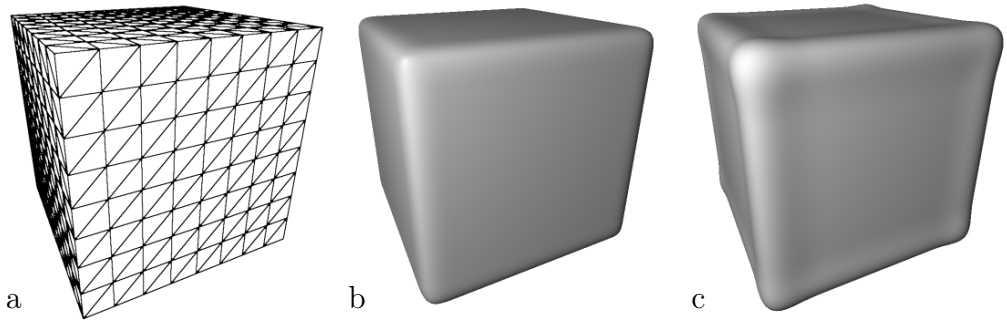


FIGURE 2.17 – **Bourrelets près des arêtes semi-vives.** Un maillage (a) raffiné avec la méthode de Loop (b) et LS^3 (c). En rapprochant la surface des sommets du maillage de contrôle, notre méthode provoque des bourrelets dans certains cas.

(2 dans l'illustration), puis comme lisse pour les pas suivants. Ce bourrelet est en fait typique des méthodes de subdivision interpolantes : pour produire une surface lisse qui passe par l'arête vive, la surface doit obligatoirement sortir de l'enveloppe convexe. Dans notre cas, nous n'obtenons pas une interpolation, mais la surface est tout de même décalée vers l'arête vive ce qui crée le bourrelet.

2.3.3 Invariance affine

Les surfaces de subdivision linéaires possèdent l'importante propriété d'invariance affine qui affirme que, lorsque le réseau de points de contrôle subit une transformation affine, la surface limite subit exactement la même transformation. Ce n'est pas le cas des surfaces LS^3. En effet, appliquer une transformation affine sur une sphère produit une ellipsoïde ; les approximations locales ne peuvent donc pas se déformer comme il faut. Notre méthode est néanmoins invariante aux transformations rigides et aux homothéties puisque le calcul de l'approximation locale et la projection le sont. Les transformations non-supportées sont donc les mises à l'échelle non uniformes et les transvections (*shears*).

Une solution envisageable à ce problème serait d'approcher des ellipsoïdes à la place des sphères. Cependant, celles-ci possèdent beaucoup plus de degrés de liberté et sont considérablement plus difficiles à approcher de manière stable et robuste. Nous discutons cette problématique plus en détail en annexe 4.6.5.

2.4 Analyse numérique

Les surfaces de subdivisions sont généralement analysées en étudiant la structure propre de leurs matrices de subdivision respectives [Zorin et Schröder, 2000]. Cependant, compte tenu de notre étape de projection, la position d'un sommet ne peut pas être exprimée sous forme d'une combinaison linéaire des sommets du niveau de subdivision précédent, par conséquent la matrice de subdivision ne peut pas être constituée.

Intuitivement, notre idée est que, si les voisins d'un sommets $\mathbf{q}$ tendent à former un ellipsoïde, les pondérations à la limite tendent vers des poids euclidiens, ce qui implique que la surface obtenue est une surface MLS. Toute la difficulté vient de l'évaluation de la continuité de cette "fonction de poids". Cela rend l'analyse théorique de notre schéma particulièrement difficile.

Dans le contexte des courbes de subdivision, les schémas non linéaires peuvent être analysés numériquement en calculant la régularité de Hölder de plusieurs courbes limites et en retenant le pire cas [Kuijt, 1998; Marinov *et al.*, 2005]. Bien que cela soit limité aux courbes, cela nous permet tout de même d'obtenir une intuition sur l'effet de notre opérateur. En utilisant la méthodologie décrite par Kuijt et en appliquant notre approche LS^3 à des courbes 2D planaires en utilisant les masques de pondération des splines cubiques, on remarque que notre étape de projection ne dégrade pas la régularité de Hölder qui est, dans les deux cas, supérieure à deux. Cela nous permet de conclure que les courbes ainsi définies sont très probablement C^2.

Le cas des surfaces est plus difficile, mais aussi plus intéressant puisqu'il contient des sommets extraordinaires. Dans ce but, nous avons mis au point une méthode numérique pour analyser les comportements C^0, G^1 et G^2 de n'importe quelle surface produite par un processus de subdivision. Afin d'être totalement générique, notre approche ne nécessite aucune paramétrisation. Le principe consiste à générer de manière aléatoire un grand nombres de maillages de contrôle représentant 2-anneaux autour d'un sommet central $\mathbf{p}$ d'une valence donnée. Chaque voisinage est subdivisé jusqu'à convergence numérique, et nous enregistrons certains ratios de convergence pour chacun des degrés de continuité. Afin d'effectuer un très grand nombre d'itérations, à chaque pas les deux anneaux les plus extérieurs sont supprimés, ce qui n'affecte pas le résultat final autour du sommet central. Il faut noter que dans tous nos tests, les itérations ont toujours convergé.

Le comportement C^0 est analysé en enregistrant à chaque itération k le facteur de contraction ρ_k correspondant au ratio de la longueur maximale des arêtes autour de $\mathbf{p}$ entre deux itérations successives :

$$\rho_k = \max_j \frac{\|\mathbf{p}_j^k - \mathbf{p}^k\|}{\|\mathbf{p}_j^{k-1} - \mathbf{p}^{k-1}\|} \tag{2.6}$$

où les sommets $\mathbf{p}_j^k$ sont les voisins du sommet central $\mathbf{p}^k$ au niveau k.

Pour que la surface soit C^0, ρ_k doit être strictement plus petit que un. Plus précisément, pour un schéma diadique, ρ_k devrait idéalement être égal à 0.5. S'il est supérieur à 0.5 nous avons une dilatation, et une contraction dans le cas contraire, ce qui permet d'identifier les artefacts polaires.

Le comportement G^1 équivaut à analyser le taux de convergence du voisinage de $\mathbf{p}$ vers une configuration planaire. Pour cela, nous proposons de nous appuyer sur une analyse spectrale de la matrice de covariance $\mathbf{D}^k = \sum_j (\mathbf{p}^k - \mathbf{p}_j^k)^T (\mathbf{p}^k - \mathbf{p}_j^k)$. Soit λ_i^k les trois valeurs propres de $\mathbf{D}^k$ ordonnées tel que $\lambda_0^k < \lambda_1^k < \lambda_2^k$. Puisque la magnitude des valeurs propres varie de façon quadratique par rapport à l'échelle des données en entrée, nous pouvons définir le facteur de contraction α_k vers une configuration planaire comme :

$$\alpha_k = \left(\frac{\lambda_0^k / \lambda_1^k}{\lambda_0^{k-1} / \lambda_1^{k-1}} \right)^{\frac{1}{2}} . \tag{2.7}$$

Une fois encore, α_k doit être strictement inférieur à un et, dans le cas d'un schéma diadique, α_k devrait idéalement être égal à 0.5.

En supposant une continuité G^1, c'est à dire $\alpha_k < 1$, nous pouvons assigner au sommet $\mathbf{p}^k$ (resp. ses voisins $\mathbf{p}_j^k$) une normale $\mathbf{n}^k$ (resp. $\mathbf{n}_j^k$) qui est le vecteur propre de la matrice de covariance $\mathbf{D}^k$ (resp. $\mathbf{D}_i^k$) correspondant à la plus petite valeur propre. Le comportement G^2 peut alors être observé en comparant les vitesses de convergence des normales des voisins $\mathbf{n}_j^k$ vers la normale $\mathbf{n}^k$ du sommet central, aux vitesses de convergence des positions des sommets :

$$\beta_k = \frac{\text{arclength}(\mathbf{n}^k, \mathbf{n}_j^k) \|\mathbf{p}_j^{k-1} - \mathbf{p}^{k-1}\|}{\text{arclength}(\mathbf{n}^{k-1}, \mathbf{n}_j^{k-1}) \|\mathbf{p}_j^k - \mathbf{p}^k\|} \tag{2.8}$$

Ici, $\beta_k > 1$ signifie que les normales autour du sommet $\mathbf{p}$ convergent plus lentement vers la normale visée que les sommets le font, c'est à dire que la surface n'est pas G^2. D'un autre coté, $\beta_k < 1$ signifie que les normales convergent trop vite, et nous sommes a priori en train de générer un point de courbure nulle aussi appelé *flat spot*. La valeur idéale pour β est donc 1.

Nous avons utilisé ces mesures numériques pour comparer notre schéma LS^3 au schéma standard de Loop (*Std-Loop*), et deux autres variantes proposées par Barthe et Kobbelt [2004] présentant respectivement la propriété de *courbure bornée* (*Curv-Loop*) et éliminant les artefacts polaires (*Polar-Loop*). Pour les sommets réguliers, de valence 6, et en négligeant les tous premiers pas de subdivision, dans tous les cas nous trouvons les valeurs idéales $\rho_k = 0.5$, $\alpha_k = 0.5$, and $\beta_k = 1$, ce qui est cohérent avec notre analyse 2D de la régularité de Hölder. En d'autres termes, notre étape de projection ne semble pas compromettre la continuité C^2 du cas régulier.

La figure 2.18 montre les graphiques correspondant aux pires scénarios possibles de ces trois mesures, ρ_k, α_k, et β_k, pour des milliers de configurations

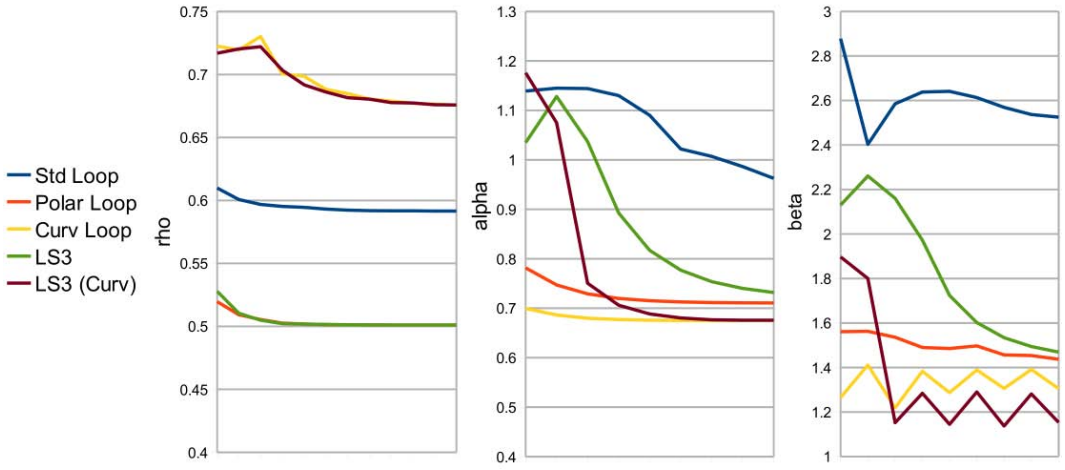


FIGURE 2.18 – **Analyse des surfaces LS^3.** Les graphiques correspondent, de gauche à droite, à nos trois métriques numériques pour étudier les classes de continuité C^0, G^1 et G^2 de différents schémas de subdivision. Ils correspondent au pire cas de 10k configurations aléatoires autour d'un sommet extraordinaire de valence 12.

aléatoires générées autour d'un sommet de valence 12. Par curiosité, nous avons aussi inclus une variante de notre schéma LS^3 utilisant les masques optimisés pour améliorer la courbure pour chacune des étapes de relaxation et de projection (notée LS^3-*Curv*). Comme on peut le voir, la variante LS^3 se comporte comme ses consœurs linéaires respectives, ce qui signifie que l'étape de projection a un impact minimal sur la continuité de la surface. Nous pouvons aussi remarquer que les artefacts polaires sont effectivement détectés par la mesure C^0 ρ tandis que le schéma de courbure bornée montre clairement un meilleur comportement G^2. Cette observation montre que notre méthode numérique correspond bien aux résultats théoriques.

2.5 Bilan

La méthode LS^3 que nous proposons est une généralisation des surfaces de subdivision qui permet de réduire considérablement certains des artefacts inhérents à ces dernières. Elle est à la fois simple, efficace et générique. Néanmoins, elle possède aussi quelques limitations qui lui sont propres, comme les bourrelets à proximité des arêtes semi-vives ou l'absence d'invariance affine. Afin de résoudre ces problèmes, nous avons commencé à explorer des procédures d'ajustement d'approximation locale de plus haut degré, ou encore des méthodes de régression locales plus sophistiquées, que nous décrivons plus en détail en annexe 4.6.5. Même si ces pistes se sont avérées infructueuses jusqu'à

présent, elles n'en restent pas moins intéressantes à poursuivre.

Nous n'avons jusqu'à présent testé notre méthode que sur des maillages statiques, c'est à dire sans animation et nous ne l'avons pas testé non plus dans le cadre d'une édition interactive. Le meilleur moyen de déterminer si les surfaces LS^3 sont supérieures aux surfaces de subdivision dans une utilisation courante serait de les mettre en oeuvre dans un logiciel de production, et de les faire tester par des graphistes expérimentés. Un prototype de plugin pour Maya a été réalisé avec succès, néanmoins nous n'avons pas pu aller plus loin par manque de temps et de moyens.

Finalement, l'analyse théorique de ces surfaces s'avère compliquée : les outils développés pour les surfaces de subdivision ne s'appliquent pas à notre méthode qui n'est pas linéaire. Nous avons donc développé des outils d'analyses numériques, afin d'étudier la classe de continuité des surfaces LS^3, qui nous laissent supposer qu'elle est directement héritée du schéma de subdivision linéaire utilisant les mêmes pondérations. Ceci est en accord avec les propriétés connues des surfaces MLS sur lesquelles notre approche LS^3 est basée : la continuité d'une reconstruction MLS est égale à la continuité des pondérations. Néanmoins, l'étude théorique des propriétés des surfaces LS^3 mériterait une analyse plus rigoureuse via la mise au point d'outils mathématiques adaptés.

Chapitre 3

Dessin vectoriel par diffusion

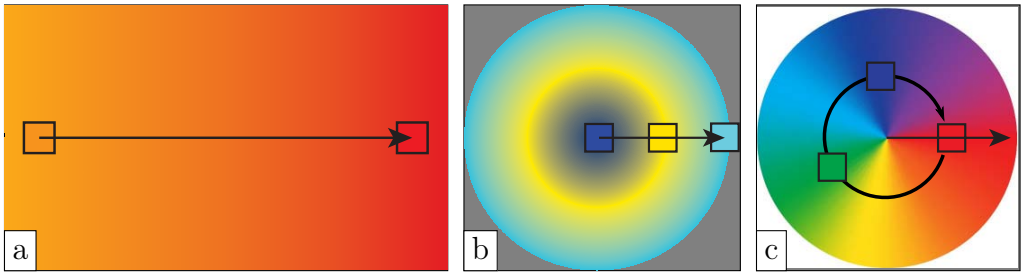


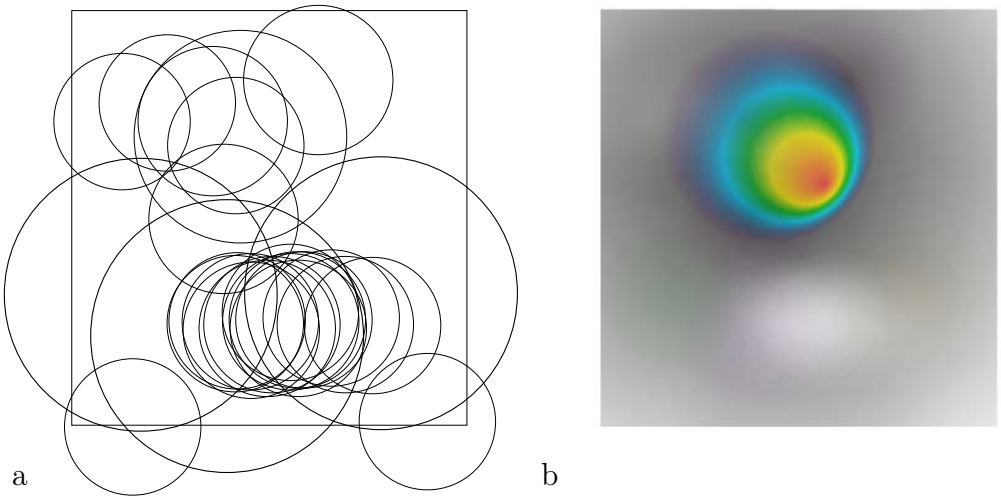
FIGURE 3.1 – **Primitives de dégradés vectoriels.** Les dégradés linéaires (a) et radiaux (b) sont les seuls dégradés définis dans la norme SVG [SVG]. Les dégradés coniques (c) sont aussi couramment utilisés.

Depuis les débuts de l'infographie, les méthodes de dessin vectoriel sont largement répandues et appréciées, principalement pour leur légèreté en mémoire et leur indépendance vis à vis de la résolution en sortie. Ces propriétés permettent de travailler sur de très grandes images avec peu de mémoire tout en garantissant un résultat net, c'est à dire sans effet d'aliasing quel que soit le niveau de zoom. Par ailleurs, de nombreux types d'images sont composés de formes géométriques simples (caractères d'impression, diagrammes, dessins stylisés, etc.) et s'avèrent beaucoup plus faciles à manipuler à l'aide de primitives vectorielles qu'avec des représentations de plus bas niveau comme de simples grilles de pixels.

Traditionnellement, les images vectorielles n'offrent que peu de possibilités pour représenter des dégradés de couleurs. Les primitives principales sont les dégradés linéaires, radiaux et coniques, illustrés figure 3.1. La figure 3.2 illustre le fait qu'il soit théoriquement possible d'obtenir des résultats très évolués en utilisant ces primitives avec du *blending*, plusieurs calques et des masques de visibilité, cependant, cela représente un travail fastidieux et peu intuitif.

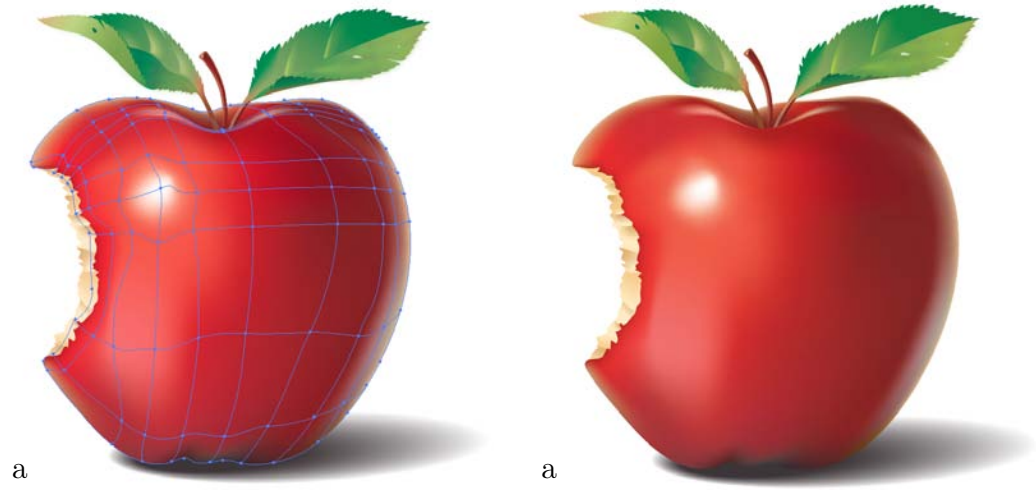
Récemment, des méthodes ont fait leur apparition pour permettre la représentation vectorielle de dégradés complexes. Par exemple, les *gradient meshes* [Sun *et al.*, 2007], illustrés figure 3.3, utilisent une grille quadrangulaire dont les arêtes sont des courbes de Bézier. Les couleurs, définies sur les sommets de la grille, sont interpolées dans chaque cellule, souvent à l'aide de patchs de Coons [1967] ou de Ferguson [1964]. Une autre approche proposée par Lecot et Levy [2006] pour la vectorisation d'image consiste à découper l'image en cellules contenant chacune une primitive de dégradé relativement simple (linéaire, radial ou quadratique). Cependant, ce genre de représentations est généralement difficile à manipuler car, d'une part, ces représentations sont souvent denses, et d'autre part, elles mettent en jeu des maillages dont la connectivité doit être manipulée manuellement (lorsque cela est possible).

Les représentations fondées sur un principe de *diffusion* [Orzan *et al.*, 2008; Finch *et al.*, 2011] évitent ces écueils en permettant de diffuser des couleurs



a b

FIGURE 3.2 – **Dégradé complexe à base de primitives simples.** En combinant plusieurs dégradés simple (a), en l'occurence, des dégradés radiaux, il est possible d'obtenir des résultats complexes (b). Cependant, cette façon de faire est fastidieuse et peu adaptée pour représenter des discontinuités.



a a

FIGURE 3.3 – ***Gradient meshes.*** Les couleurs sont définies sur les sommets du maillage de courbe (a) puis interpolées pour former des dégradés lisses.

de part et d'autre des courbes de contrôle. Le résultat est une représentation capable de produire des dégradés très évolués avec peu de primitives, sans nécessiter de gérer manuellement des informations de connectivité.

Dans ce chapitre, nous présentons un tour d'horizon des différents travaux qui ont été réalisés sur ces représentations. Nous verrons d'abord section 3.1 le principe des images de diffusion ainsi que de nombreuses extensions permettant de créer des images plus riches avec moins de courbes. En section 3.2, nous présentons les différentes stratégies proposées pour résoudre le problème de la diffusion, c'est à dire de l'interpolation des couleurs. Pour finir, nous discuterons en section 3.3 d'une extension récente consistant à utiliser une interpolation bi-harmonique afin de créer des dégradés de couleur plus naturels tout en offrant des contrôles supplémentaires.

3.1 Images de diffusion

Une *image de diffusion* est une représentation d'image permettant de générer des dégradés de couleur complexes à partir de contraintes de couleur *vectorielles* définies à la main sous forme de *courbes de diffusion* [Orzan *et al.*, 2008] (figure 3.4.a). L'image finale u est obtenue en interpolant des couleurs définies sur les courbes à l'aide d'une *diffusion Laplacienne* (figure 3.4.b), ce qui donne le système d'équation suivant

$$\begin{aligned} \Delta u &= 0 \\ u(\mathbf{p}) &= v(\mathbf{p}) \qquad \text{si } \mathbf{p} \in S \end{aligned} \tag{3.1}$$

où $v(\mathbf{p})$ est la couleur contrainte au point $\mathbf{p}$ et S est l'ensemble des points contraints. La solution de la diffusion Laplacienne est lisse en tout point, sauf au niveau des contraintes où elle n'est que C^0. Notons qu'une diffusion Laplacienne produit une interpolation dite *harmonique*.

Avec une telle interpolation, les *contraintes ponctuelles* produisent des "pics" peu élégants, comme illustré figure 3.4.c. En pratique, les couleurs sont donc spécifiées uniquement sur les courbes. Il semble donc naturel d'utiliser des courbes pour contraindre le processus de diffusion. Afin de pouvoir représenter des discontinuités, ces *courbes de diffusion* peuvent émettre une couleur différente de chaque côté ; ce qui est équivalent à deux courbes de diffusion simples infiniment proches.

Plus formellement, une image de diffusion est constituée d'un ensemble de courbes de diffusion paramétriques $\mathcal{C}_i(t)$ avec $i \in [1, n]$ et $t \in [0, 1]$, auxquelles sont associées deux fonctions $v_i^g(t)$ et $v_i^d(t)$ correspondant à la couleur émise respectivement à gauche et à droite. Le plus souvent, les courbes utilisées sont des courbes de Bézier et les fonctions de couleurs sont de simples dégradés linéaires.

FIGURE 3.4 – **Courbes de diffusion.** Une courbe de diffusion émet de la couleur de chaque côté (a) qui est ensuite diffusée pour produire des dégradés lisses (b). Les contraintes ponctuelles produisent des résultats médiocres (c) et ne sont donc pas utilisées.

Singularités. De par leur définition, les images de diffusion autorisent ce que l'on appelle des points singuliers, c'est à dire des points où plusieurs couleurs sont imposées. Les plus communs apparaissent aux extrémités des courbes lorsqu'une couleur différente est définie de chaque côté. Les autres points singuliers sont les points d'intersection ou les éventuelles discontinuités dans les fonctions de couleurs définies sur les courbes. Il peut aussi y avoir des cas clairement dégénérés où deux courbes aux couleurs différentes se superposent, mais nous partons du principe que de tels cas sont issus d'une mauvaise manipulation et ne considérons donc que les couleurs émises par l'une de ces courbes. Nous verrons plus tard que les points de singularité sont particulièrement délicats à évaluer de façon stable et sont la source de nombreuses difficultés dans la conception des solveurs.

3.1.1 Contraintes avancées

En pratique, la définition précédente des images de diffusion offre un contrôle relativement limité ne permettant pas de représenter aisément des images complexes. De nombreuses extensions ont été proposées dans un même but : permettre de produire des résultats plus évolués avec un minimum de courbes. Cependant, toutes ces extensions ne sont pas forcement compatibles entre elles et tous les solveurs ne sont pas en mesure de les gérer.

Les courbes bloquantes ou "barrières"

Les *courbes bloquantes*, aussi appelées barrières [Bezerra *et al.*, 2010b], bloquent la diffusion sans émettre de couleur comme illustré figure 3.5. Ces courbes sont particulièrement utiles pour représenter les occlusions qui apparaissent fréquemment dans les images : les courbes définissant la bordure d'un objet peuvent se représenter en émettant de la couleur du côté intérieur et en bloquant la diffusion de l'autre, ce qui évite d'avoir à éditer la fonction de couleur (potentiellement complexe) nécessaire pour que l'objet se fonde dans

FIGURE 3.5 – **Courbes bloquantes et courbes de contour.** Les courbes bloquantes stoppent la diffusion sans spécifier de contraintes d'un côté (a) ou des deux (b). Les courbes de contour (c) émettent la même couleur tout du long, mais celle-ci n'est pas définie manuellement mais calculée automatiquement (ici, la courbe centrale).

l'arrière plan. Cependant, l'utilisation de calques, lorsque c'est possible, est généralement plus flexible. La mise en œuvre de ce type de courbe dépend fortement du type de solveur.

Les courbes de contour

Les *courbes de contour* proposées par Finch *et al.* [2011] consistent à contraindre l'intégralité de la courbe à n'être que d'une seule et même couleur. Évidement, ce type de courbes n'a de sens que lorsque sa couleur unique est calculée automatiquement par le système.

L'intérêt pratique de ces courbes réside dans leur capacité à modeler la forme d'un dégradé sans avoir à placer de contraintes de valeur explicite comme illustré figure 3.5.c.

Force de diffusion

Cette extension proposée par Bezerra *et al.* [2010b] permet de contrôler l'influence relative des courbes de diffusion à l'aide d'un mécanisme similaire aux *coordonnées homogènes*. Une *couleur homogène* (r_h, g_h, b_h, w_h) peut être ramenée à une couleur normale par la projection : $(r, g, b) = (\frac{r_h}{w_h}, \frac{g_h}{w_h}, \frac{b_h}{w_h})$. La projection de la somme de plusieurs couleurs homogènes correspond à la somme des couleurs, pondérée par w_h ; il est ainsi possible de donner une plus grande importance à certaines contraintes de couleurs qu'à d'autres. Notons que l'influence peut même varier le long d'une courbe.

Ce procédé est facile à mettre en œuvre quel que soit le solveur puisqu'il ne nécessite que de diffuser un canal supplémentaire. En revanche, il s'agit d'un contrôle très indirect et contrôler individuellement les influences relatives de chacun des points de contraintes indépendamment peut rapidement s'avérer difficile.

Diffusion anisotrope

Bezerra *et al.* [2010b] proposent de modifier l'équation de Laplace 3.1 de façon à favoriser la diffusion dans une direction plutôt qu'une autre. La direction de la diffusion est contrôlée à l'aide de *courbes de direction* favorisant la diffusion dans leurs directions tangentes. Ces directions sont d'abord diffusées avec l'équation de Laplace pour obtenir un champ de vecteurs qui est ensuite utilisé pour effectuer la diffusion anisotrope.

Portails

Dans le cadre des images de diffusion, le principe des *portails* [Bezerra *et al.*, 2010b] est de permettre à la couleur d'un endroit de se diffuser à d'autres endroits éloignés. Cette extension a été originellement proposée pour gérer les occlusions : la couleur d'un coté de l'objet au premier plan peut-être directement transmise de l'autre coté. Cependant, il est considérablement plus simple et plus robuste d'utiliser des calques pour réaliser cet effet.

Néanmoins, cette extension se trouve être particulièrement utile pour créer des textures répétitives. Cela est réalisé en transmettant les couleurs qui se diffusent sur un bord de la texture au bord opposé.

Flou

Des courbes de diffusion permettent de représenter des discontinuités nettes mais n'offrent pas de moyen simple de créer des transitions douces, pourtant utiles, par exemple, pour créer des effets de profondeur de champ. On peut considérer qu'une courbe de diffusion émettant une couleur différente de chaque côté est en fait constituée de deux courbes infiniment proches émettant chacune leur propre couleur de part et d'autre. Il est alors possible de simuler un effet de flou en écartant ces deux courbes [Takayama *et al.*, 2010], cependant la discontinuités du gradient le long des courbes est clairement visible ce qui ne produit pas des dégradés élégants (figure 3.6.a).De plus, écarter les courbes de manière robuste, c'est à dire sans introduire d'intersections ou d'autres effets indésirables, est un énorme défi.

Comme illustré figure 3.6.b, une autre solution consiste à appliquer un filtre de flou Gaussien *a posteriori*. Afin de contrôler le rayon de lissage, Orzan *et al.* [2008] proposent d'assigner à chaque point d'une courbe un rayon de lissage qui est interpolé en même temps que les couleurs. Cette méthode possède l'inconvénient de nécessiter une passe de lissage supplémentaire qui peut devenir coûteuse lors de l'utilisation de grands rayons, ou d'un fort zoom.

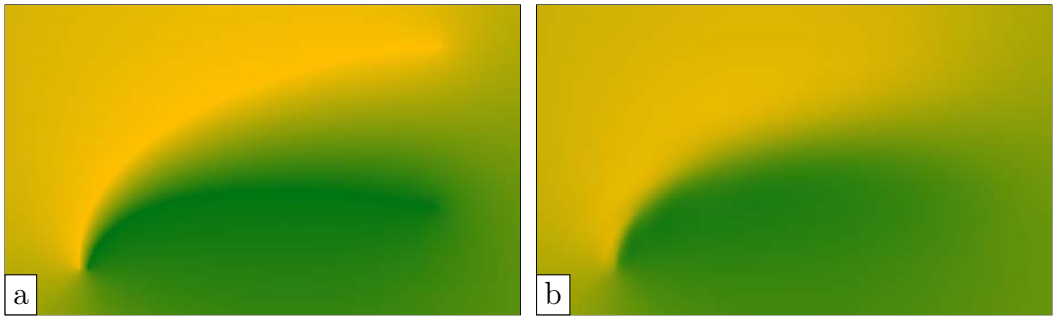


FIGURE 3.6 – **Flou.** Créer un effet flou en séparant la courbe en deux puis en décalant les deux parties obtenues est peu convainquant car les courbes décalées sont clairement visibles (a). Effectuer un flou gaussien de rayon variable donne de meilleurs résultats mais est plus coûteux.

Diffusion de textures

Les textures sont des éléments importants des images qui sont particulièrement difficiles à réaliser de façon purement vectorielle. C'est pourquoi, le plus souvent, les textures utilisées sont de simples images *bitmaps*. Winnemöller *et al.* [2009] proposent de texturer des images de diffusion à l'aide de textures elles-même définies avec des courbes de diffusion. Leur système se concentre sur l'édition des coordonnées de texture afin d'émuler des déformations visibles lorsqu'une texture est plaquée sur un objet 3D. Cependant, cette méthode ne permet pas de mélanger plusieurs textures ou de mélanger les textures avec des courbes de diffusion, donc seules des zones fermées peuvent être texturées de façon uniforme.

Une autre approche consiste à utiliser des textures procédurales dont les paramètres sont diffusés en même temps que les couleurs. Cela a été mis en œuvre avec succès par Jeschke *et al.* [2011] à l'aide de bruits de Gabor [Lagae *et al.*, 2009]. Cependant cette approche est limité en terme d'expressivité.

Contraintes de gradient

Dans le cadre de la génération de terrain, Hnaidi *et al.* [2010] proposent l'utilisation de contraintes de gradient. Pour cela, ils ajoutent à l'équation de Laplace et aux contraintes de valeurs des contraintes de gradient de la forme $\nabla u(\mathbf{p}) = g(\mathbf{p}), \mathbf{p} \in G$ où $g(\mathbf{p})$ est une contrainte de gradient et G est l'ensemble des points dont le gradient est contraint. Dans le cadre de l'édition de dégradés de couleur, les contraintes de gradient sont difficiles à manipuler et à concevoir car il faut fixer un gradient par canal. La seule contrainte pratique à utiliser est de fixer le gradient à 0. Nous verrons en section 3.3 une autre approche de la diffusion plus adaptée à l'utilisation de contraintes de gradient.

3.2 Solveurs

Dans la section précédente, nous avons vu qu'une image de diffusion était simplement le résultat d'une ou plusieurs interpolations harmoniques d'un ensemble de courbes de couleur soumises à des contraintes diverses. Une interpolation harmonique est réalisée en résolvant l'équation de Laplace 3.1 qui intervient dans de nombreux domaines, notamment en physique. Il existe donc une vaste littérature sur la résolution de cette équation, et les solutions employées dépendent énormément du contexte applicatif précis, des types de contraintes, des *a priori* sur la solution, etc. Ici, notre objectif est de créer des images visuellement plaisantes et non de calculer précisément le comportement d'un phénomène physique. Par exemple, la précision du solveur n'est pas fondamentale car les couleurs sont généralement quantifiées sur 8 bits par canal. Un léger *biais* (décalage entre le résultat et la solution théorique exacte) est même acceptable à condition qu'il n'introduise pas d'artefacts.

Voici une liste des principales propriétés souhaitables pour un solveur de diffusion de couleur :

Rapidité : le solveur doit être capable de donner un retour quasi-instantané à l'utilisateur pendant l'édition. En pratique, cela implique un budget maximal d'environ 100ms pour le calcul de la diffusion voir même de 30ms pour permettre une animation fluide en temps-réel.

Qualité : le solveur doit produire des images de qualité. Cela signifie notamment éviter les comportements indésirables couramment appelés *artefacts*.

Généralité : de nombreuses extensions ont été proposées pour les images de diffusion, mais tous les solveurs ne sont pas forcément capables de les gérer. Dans l'idéal, un solveur doit être suffisamment générique pour s'adapter facilement aux différents cas d'utilisations en autorisant en entrée une large palette de contraintes et de contrôles.

Cohérence temporelle : dans le cas d'animations où les données en entrée varient de façon continue avec le temps, le résultat doit varier de façon continue. Cela signifie principalement éviter les effets de scintillements.

Nous classons les solveurs existants en deux catégories. La première que nous présentons section 3.2.1 regroupe les solveurs résolvant directement l'équation Laplacienne dans une grille de pixels. Comme nous le verrons, cette approche pose des problèmes difficiles tout en étant assez coûteuse en calcul. La deuxième catégorie, présentée section 3.2.2, regroupe des méthodes cherchant à approcher le résultat de la diffusion Laplacienne en essayant de se ramener à de simples intégrales sur les contraintes.

3.2.1 Diffusion Laplacienne en espace image

La première étape pour résoudre une *équation aux dérivées partielles* (EDP) telle que l'équation Laplacienne est de discrétiser le domaine. Bien que plusieurs options soient envisageables, l'objectif ici étant d'obtenir une image *bitmap*, la méthode la plus simple est de discrétiser le problème directement sur une grille régulière. C'est la stratégie employée par les solveurs présentés ici.

L'équation de Laplace peut facilement se discrétiser sur une grille de pixels à l'aide de la méthode des différences finies, ce qui conduit au système d'équation linéaire suivant :

$$u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j} = 0 \ , \tag{3.2}$$

où $u_{i,j}$ correspond à la valeur du pixel (i, j). Tous les $u_{i,j}$ pour lesquels la valeur n'est pas fixée par les contraintes sont des inconnues du système.

Ce système d'équations peut être résolu de différentes manières : soit sous une forme matricielle à l'aide d'un solveur linéaire creux, soit directement sur la grille de pixels à l'aide d'une méthode itérative comme des itérations de Jacobi ou la méthode de Gauss-Seidel. Cependant, une image contient généralement plus d'un millions de pixels, et de telles méthodes ne permettent donc pas d'obtenir les performances voulues, même en exploitant les capacités de calculs des GPUs. Afin d'accélérer les calculs, deux approches que nous détaillerons dans la suite ont été proposées : la première se base sur les méthodes multi-grilles, tandis que la seconde, plus originale, exploite au mieux les capacités de rastérisation des GPUs.

Solveurs multi-grilles

Les solveurs *multi-grilles* [Briggs *et al.*, 2000] sont relativement rapides tant que les images produites sont de taille raisonnable, et sont suffisamment flexibles pour permettre d'implémenter la plupart des extensions présentées section 3.1.1. Leur fonctionnement est illustré figure 3.7. Les méthodes multi-grilles résolvent le problème sur une pyramide de grilles régulières partant d'une résolution relativement faible jusqu'à la résolution désirée. À la résolution initiale, la valeur des pixels est calculée à l'aide d'un solveur direct où itératif, ce qui s'avère assez rapide puisque la grille est de dimension réduite. Les grilles suivantes sont d'abord initialisées en sur-échantillonnant la grille de niveau inférieur, ce qui permet à un solveur itératif de converger rapidement.

Dans le contexte des images de diffusion, Orzan *et al.* [2008] utilisent des itérations de Jacobi aussi bien pour l'étape d'initialisation que pour le calcul des grilles intermédiaires. Les itérations de Jacobi et l'étape de sur-échantillonnage sont particulièrement faciles à paralléliser, ce qui permet d'implémenter cette méthode intégralement sur GPU. Comme les itérations de Jacobi convergent assez lentement sur des images de grandes tailles, la résolution de la grille

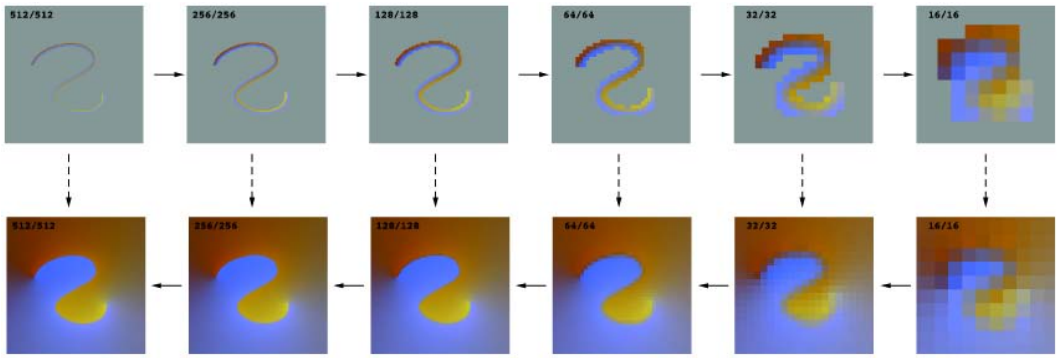


FIGURE 3.7 – **Fonctionnement des solveurs *multigrilles*.** Les contraintes sont d'abord initialisées sur des grilles de résolution de plus en plus faibles (en haut). Ensuite, la diffusion est d'abord effectuée sur la grille de plus faible résolution qui sert d'initialisation à la grille suivante. Le processus est répété jusqu'à obtenir la grille de résolution désirée (en bas).

initiale doit être assez faible, ce qui implique un grand nombre d'étapes de ré-échantillonage.

Plus récemment, Finch *et al.* [2011] ont également utilisé un solveur multigrilles pour créer des images de diffusion. Dans leur cas, un solveur linéaire direct est utilisé pour calculer la grille initiale à une résolution déjà conséquente, à savoir 64×64 pour l'interaction, et 128×128 pour obtenir des images de plus haute qualité. Les grilles intermédiaires sont ensuite amenées à convergence à l'aide d'itérations de Gauss-Seidel réalisées sur le CPU.

Initialisation des contraintes. Pour prendre en compte les contraintes, celles-ci sont discrétisées *sur chaque grille*. Cette opération doit être effectuée avec précaution pour éviter les fuites de couleur sur les courbes diffusant une couleur différente de chaque côté.

Orzan *et al.* utilisent la carte graphique pour le rendu des contraintes en traçant une poly-ligne différente pour chaque côté des courbes. Cependant, pour limiter les risques que ces poly-lignes se superposent, ce qui engendrerait des fuites, ils les séparent de plusieurs pixels (trois dans leur implémentation) et ajoutent des contraintes de gradient afin d'obtenir des discontinuités plus nettes.

Pour plus de robustesse, Finch *et al.* calculent explicitement les intersections entre chaque courbe et la grille de pixels. Lorsque une courbe entre en intersection avec une arête de la grille, les pixels à ses extrémités se voient assigner une couleur en fonction du côté où ils se trouvent. Comparée à la méthode d'Orzan *et al.*, cette méthode est plus précise, ce qui permet d'éviter les incohérences lorsque deux courbes sont proches l'une de l'autre, mais aussi plus coûteuse et difficile à mettre en œuvre sur GPU.

Solveur de Jeschke *et al.*

Afin d'offrir de meilleures performances que les solveurs multi-grilles, Jeschke *et al.* [2009a] ont développé un solveur exploitant les capacités de rasterisation des GPU. Ce solveur repose sur deux idées :

1. utiliser une *initialisation de Voronoï* consistant à assigner à chaque pixel de l'image la couleur de la contrainte la plus proche, et
2. appliquer des itérations de Jacobi avec un masque de taille variable pour accélérer la convergence.

L'initialisation est directement effectuée sur l'image à la résolution désirée en utilisant les capacités de rasterisation des cartes graphiques modernes : des polygones infinis sont tracés de chaque côté des courbes et le *tampon de profondeur* est utilisé pour stocker la distance à la courbe la plus proche afin que le *test de profondeur* ne garde que la couleur de la courbe la plus proche.

L'image initiale u^0 est ensuite filtrée à l'aide d'itérations de Jacobi :

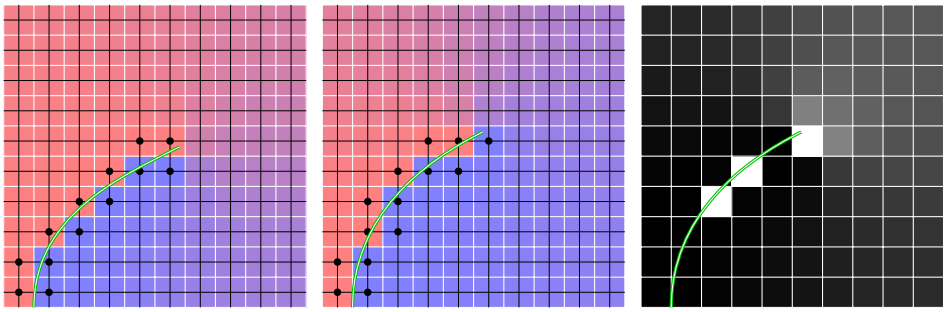
$$u_{i,j}^{k+1} = \frac{1}{4}\left(u_{i-r,j}^{k} + u_{i+r,j}^{k} + u_{i,j-r}^{k} + u_{i,j+r}^{k}\right) \tag{3.3}$$

où r est le rayon du filtre, qui dépend à la fois de la distance d de la courbe la plus proche et de l'itération k. Un choix simple pour le rayon du filtre est d'utiliser $r = d$ à la première étape, puis de diminuer r linéairement jusqu'à obtenir $r = 1$ à la dernière itération. Cette stratégie permet d'obtenir des dégradés lisses en seulement 8 passes.

Il est important de noter que quelques itérations de Jacobi, même en utilisant de larges masques, ne suffisent pas à atteindre la convergence, mais seulement à créer des dégradés lisses. Autrement dit, bien que la solution obtenue puisse être *biaisée*, les dégradés de couleurs obtenus sont lisses et le résultat agréable à l'œil. De plus, il s'agit du solveur le plus rapide existant actuellement. En revanche, il nécessite une carte graphique haut de gamme, et il s'avère peu flexible car il ne permet pas de gérer la plupart des extensions présentées section 3.1.1.

Difficultés avec les solveurs en espace image

Contraintes hors champ. Appliquées naïvement, les méthodes en espace image ignorent les contraintes se trouvant en dehors de la grille de pixels, comme cela arrive fréquemment lors de zooms. Pour les prendre en compte, la solution communément utilisée consiste à calculer la diffusion sur l'intégralité du domaine dans une grille grossière puis à utiliser le résultat pour initialiser les bordures de l'image finale [Orzan *et al.*, 2008]. Cette solution peut être appliquée récursivement pour obtenir une meilleure approximation même dans le cas de zooms importants [Finch *et al.*, 2011]. Cependant, cette méthode



a b c

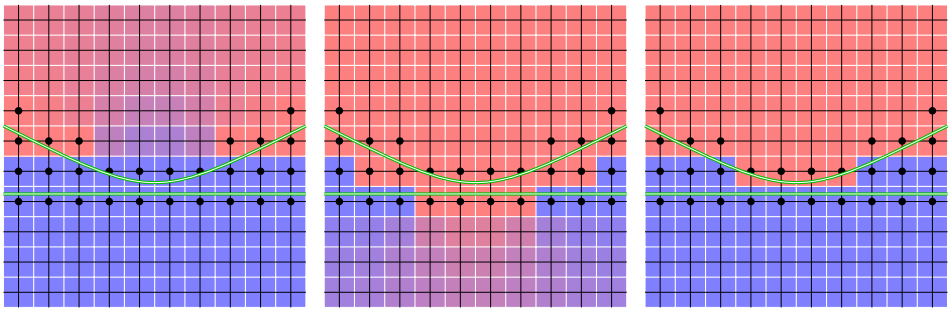
FIGURE 3.8 – **Extrémités des courbes avec les solveurs en espace image.** Une courbe (en vert) émet une couleur différente de chaque côté. Les contraintes sont discrétisées sur la grille de pixels de façon similaire à [Finch *et al.*, 2011] : lorsque la courbe entre en intersection avec une arête de la grille reliant le centre de chaque pixel (en noir), les pixels correspondant à ses extrémités (•) sont contraints. A l'extrémité de la courbe, les contraintes rouges et bleues sont soit l'une sur l'autre (a), soit côte à côte (b). Cela engendre des résultats très différents malgré le fait que l'extrémité de la courbe soit déplacée de moins d'un pixel. L'image (c) correspond à la différence entre (a) et (b).

est coûteuse puisqu'elle nécessite le rendu de plusieurs grilles et/ou de grilles plus grandes. Ces méthodes ont donc des performances moindres dès que des contraintes se trouvent hors champ, ce qui est extrêmement fréquent lors de l'utilisation d'images vectorielles.

Scintillements. La figure 3.8 illustre un problème typique des solveurs utilisant la grille de pixel comme discrétisation. Lorsqu'une courbe est déplacée, les pixels à son extrémité passent brusquement de l'état contraint à non-contraint, ou inversement. Or, le changement d'état d'un seul pixel peut avoir un impact très important. Cela conduit à des scintillements lorsqu'une courbe est déplacée.

Le solveur de Jeschke *et al.* [2009a] serait victime de ce problème si les itérations de Jacobi étaient effectuées jusqu'à convergence : le résultat serait alors entièrement défini par les pixels invariants, c'est à dire ceux qui se trouvent à une distance de moins d'un pixel d'une contrainte. Cependant, comme seulement quelques passes de filtrage sont effectuées, le résultat est fortement dépendant de l'initialisation de Voronoï, qui s'avère beaucoup plus stable que celle utilisée avec les solveurs multi-grilles.

Superposition des contraintes. Quelle que soit la méthode utilisée pour discrétiser les contraintes, il peut arriver que plusieurs contraintes entrent en intersection avec le même pixel. Il s'agit même d'une situation fréquente dans



a b c

FIGURE 3.9 – **Fuites de couleurs.** Si les contraintes sont initialisées de manière naïve, c'est à dire en discrétisant les courbes l'une après l'autre indépendamment, cela peut conduire à des fuites de couleur (a et b). En initialisant les contraintes à la couleur de la courbe la plus proche comme Jeschke *et al.* [2009a], le problème peut-être évité (c).

deux cas : lors d'un zoom arrière et dans les grilles de basse résolution des solveurs multi-grilles. Comme illustré figure 3.9, cela peut conduire à des fuites de couleur. Pour éviter cela, il est possible d'initialiser les pixels à la couleur de la contraintes la plus proche, de façon similaire à la méthode de Jeschke *et al.* [2009a], mais sans forcement faire une initialisation globale (voir figure 3.9.c).

3.2.2 Solveurs approchants

Résoudre le problème globalement sur une grille de pixels comme présenté précédemment est à la fois lourd en calcul et contraignant. Les solveurs que nous présentons ici utilisent des méthodes approchant le résultat de la diffusion Laplacienne par une intégrale sur les contraintes afin de réduire le coût en calcul ou d'apporter plus de flexibilité.

Mean value coordinnates. La plupart des méthodes que nous allons présenter ici s'appuient sur le concept des *mean value coordinnates* (MVC) [Floater, 2003]. Les MVC sont des coordonnées barycentriques généralisées [1] qui permettent de calculer, à partir d'un point $\mathbf{q}$ se trouvant à l'intérieur d'un polygone convexe de sommet $\mathbf{p}_1, \ldots, \mathbf{p}_i$, des pondérations w_i telles que $\mathbf{q} = \sum_i w_i \mathbf{p}_i$ et $\sum_i w_i = 1$. Ces pondérations sont positives et varient de manière lisse. Farbman *et al.* [2009] ont remarqué que ces coordonnées peuvent être utilisées pour produire des interpolations quasiment identiques à une diffusion Laplacienne.

1. La généralisation des coordonnées barycentriques à des polygones de plus de trois sommets possède de nombreuses applications en infographie [Meyer *et al.*, 2002a; Ju *et al.*, 2005].

En pratique, les MVC ne sont positives que pour des polygones convexes et dans notre cas, des valeurs négatives peuvent provoquer une extrapolation des couleurs qui nous éloigne grandement du comportement de la diffusion Laplacienne. Les *positive mean value coordinates* (PMVC) [Lipman *et al.*, 2007] étendent les MVC aux polygones concaves. Supposons que nous disposons d'un ensemble de contraintes de valeur $v(\mathbf{p})$, la fonction

$$u(\mathbf{q}) = \frac{\int_0^{2\pi} w(\mathbf{x}(\mathbf{q},\theta))v(\mathbf{x}(\mathbf{q},\theta))\,d\theta}{\int_0^{2\pi} w(\mathbf{x}(\mathbf{q},\theta))\,d\theta} \tag{3.4}$$

définit une interpolation des contraintes, avec $\mathbf{x}(\mathbf{q},\theta)$ le point d'intersection le plus proche entre les contraintes et la droite partant de $\mathbf{q}$ dans la direction θet $w(\mathbf{p}) = 1/\|\mathbf{q} - \mathbf{p}\ \|$.

Cette approche permet d'imaginer quelques contrôles supplémentaires. Notamment, Bowers *et al.* [2011] proposent d'utiliser $w(\mathbf{p}) = w_c(\mathbf{p})\|\mathbf{q} - \mathbf{p}\|^{-e}$ où l'ajustement de la fonction w_c et de l'exposant e permet de réaliser différents comportements. Par exemple, choisir $w_c(\mathbf{p}) = 0$ implique que la contrainte n'a aucune influence, ce qui en fait l'équivalent des courbes bloquantes. Lorsque $w_c(\mathbf{p}) > 0$, ce paramètre permet de contrôler l'importance d'une courbe par rapport aux autres, ce qui a un effet similaire à l'extension consistant à diffuser des couleurs homogènes, présentée section 3.1.1. Le paramètre e contrôle le comportement de l'interpolation à distance des contraintes. Imaginons que nos contraintes soient deux droites parallèles de valeur différentes ; lorsque $e = 1$ la valeur est interpolée linéairement entre les deux.

Au lieu d'utiliser des contraintes de valeur fixes sur les courbes, Bowers *et al.* [2011] proposent de les rendre dépendantes du point d'évaluation $\mathbf{q}$, c'est à dire de remplacer le terme $v(\mathbf{x}(\mathbf{q},\theta))$ par une fonction $v'(\mathbf{x}(\mathbf{q},\theta),\mathbf{q})$. Il est ainsi possible de diffuser, en plus d'une couleur unie, une texture, un dégradé analytique ou toute autre fonction calculable à partir de paramètres définis sur les contraintes et de la position du point d'évaluation $\mathbf{q}$ tel qu'illustré figure 3.10. Ces différents effets, définis par la fonction v', sont appelés *shaders*. Ce genre d'effets n'est pas réalisable facilement avec une diffusion Laplacienne car cela nécessiterait de diffuser un canal supplémentaire par *shader* (dans le sens où deux textures différentes sont deux *shaders* différents) pour déterminer leur influence relative et ensuite de les fusionner.

Dans les faits, l'attrait principal pour les approches à base de PMVC est le fait que la valeur de n'importe quel pixel puisse être évaluée directement sans avoir à résoudre le problème globalement. D'un autre coté, cela transpose la difficulté d'une résolution globale à un problème de calcul de **visibilité** qui est tout aussi difficile à résoudre efficacement. Nous allons voir dans la suite les différentes stratégies qui ont été expérimentées.

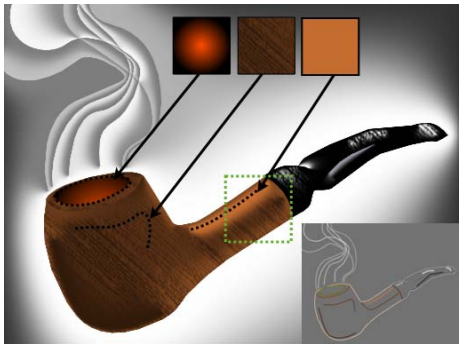


FIGURE 3.10 – ***Shaders* de diffusion** tels que proposés par Bowers *et al.* [2011]. Les encarts présentent les trois types de *shaders* utilisés dans cette image. De gauche à droite, un dégradé radial, une texture et une couleur unie. Image provenant de [Bowers *et al.*, 2011].

Évaluation par rastérisation

Takayama *et al.* [2010] ont utilisé les PMVC pour calculer des *volumes de diffusion*, qui sont l'équivalent des images de diffusion en trois dimensions. Les contraintes sont ici définies sur des surfaces et non sur des courbes. Ils utilisent ces volumes de diffusion pour décrire l'intérieur d'objets qui sont ensuite visualisés en coupe. Clairement, l'utilisation d'une grille régulière en 3D serait beaucoup trop coûteuse et impliquerait de calculer la couleur de nombreux voxels inutiles pour le rendu final car n'appartenant pas à la surface de coupe. Les PMVC permettent de calculer uniquement les couleurs des points appartenant à la surface de coupe sans se soucier du reste du volume de diffusion.

La première étape du rendu consiste à calculer l'intersection entre la surface de coupe et les surfaces de diffusion générant un ensemble de *segments de coupe*. La surface de coupe est ensuite triangulée à l'aide d'un raffinement de Delaunay contraint de façon à ce que les segments de coupe soient représentés dans la triangulation. La couleur des sommets appartenant aux segments de coupe est directement récupérée de la surface de diffusion correspondante. La couleur des autres sommets est calculée à l'aide des PMVC.

Bien que l'équation 3.4 puisse être évaluée analytiquement, cela nécessiterait de construire un polygone correspondant aux faces visibles en chaque point d'évaluation, ce qui est prohibitif en 3D. Pour obtenir des performances satisfaisantes, l'intégration est effectuée numériquement sur GPU. Pour calculer la couleur d'un sommet de coordonnées $\mathbf{q}$, les surfaces de diffusion sont rendues dans une *cube-map* centrée sur $\mathbf{q}$. Le test de profondeur effectue automatiquement le calcul de visibilité, et le tampon de profondeur contient la distance à chaque échantillon permettant de calculer les pondérations. La couleur du sommet est alors simplement obtenue par la moyenne des pixels de la *cube-map*, pondérée par une des formules vue précédemment.

D'après les tests effectués par Takayama *et al.*, cette méthode permet un meilleur rapport qualité/temps que la résolution de la diffusion sur un maillage tétraédrique. Le temps de rendu reste cependant important, s'étalant de 1 à 30 secondes par image suivant la complexité du modèle.

Lancer de rayons

Bowers *et al.* [2011] ont proposé une méthode évaluant l'équation 3.4 basée sur un *lancer de rayons stochastique* : pour évaluer la valeur $u(\mathbf{q})$, n rayons sont lancés de $\mathbf{q}$ vers des directions $\theta_i, i \in [1 : n]$ aléatoires. L'intersection entre le i-ème rayon et la contrainte la plus proche est notée $\mathbf{p}_i$. La valeur est calculée à l'aide d'une simple moyenne pondérée des couleurs des contraintes aux points $\mathbf{p}_i$: $u(\mathbf{q}) = \sum_i w(\mathbf{p}_i) \cdot v(\mathbf{p}_i, \mathbf{q}) \,/\, \sum_i w(\mathbf{p}_i)$, où $v(\mathbf{p}_i, \mathbf{q})$ est la valeur contrainte au point $\mathbf{p}_i$ dans la direction de $\mathbf{q}$.

Ce lancer de rayons est effectué sur GPU et la structure accélératrice utilisée est une simple grille régulière présentant l'avantage d'être facile à construire. Dans leur implémentation, Bowers *et al.* utilisent 128 rayons par pixel en appliquant un léger décalage (*jittering*) sur la source des pixels ce qui produit un effet d'antialiasing. Cette approche reste coûteuse puisque qu'ils arrivent à calculer des images de 512×512 à 2-3 fps seulement. Pour de meilleures performances durant l'interaction, l'image est découpée en blocs de 8×8 pixels, la couleur au coins des blocs est calculée en ne lançant que 64 rayons et les blocs sont remplis à l'aide d'une interpolation bilinéaire. Finalement, les blocs proches des contraintes sont évalués pixel par pixel. Cela permet d'obtenir entre 14 et 25 fps.

Intégration analytique

Pang *et al.* [2012] ont opté pour une intégration analytique de l'équation 3.4. Pour cela, les courbes sont tout d'abord discrétisées en un ensemble de segments à l'aide d'un algorithme de subdivision classique. Pour un point d'évaluation $\mathbf{q}$ donné, les sommets des segments *visibles* $\mathbf{p}_i$ sont arrangés dans l'ordre anti-horaire autour de $\mathbf{q}$. Les coordonnées barycentriques λ_i de $\mathbf{q}$ sont alors obtenues par $\lambda_i = \frac{w_i}{\sum_j w_i}$ où :

$$w_i = \frac{\tan(\alpha_{i-1}/2) + \tan(\alpha_i/2)}{\|\mathbf{p}_i - \mathbf{q}\|} \,, \tag{3.5}$$

et $\alpha_i = \widehat{\mathbf{p}_i\mathbf{q}\mathbf{p}_{i+1}}$. Pour déterminer quels sont les segments visibles, Pang *et al.* [2012] utilisent un algorithme parcourant tous les sommets issus de la discrétisation des courbes dans le sens anti-horaire en partant du plus proche et en suivant les segments un à un.

Cette méthode du calcul de la visibilité n'est pas assez rapide pour évaluer chacun des pixels d'une image. Pour accélérer les calculs, l'image est triangulée à l'aide d'un raffinement de Delaunay contraint ce qui permet de n'évaluer que les sommets de la triangulation, à la façon de Takayama *et al.* [2010]. Le raffinement de Delaunay ne suffit pas à produire une triangulation assez dense aux extrémités des courbes. Les triangles sont ensuite tracés en utilisant une interpolation linéaire entre les couleurs des sommets. Cela permet d'obtenir des performances similaires aux autres solveurs en utilisant uniquement le CPU. Cependant, cette méthode manque de flexibilité et produit des images de qualité médiocre (voir section 4.3).

Limitations des PMVC

Dans la plupart des cas, ce type d'interpolation produit des résultats assez lisses, avec malgré tout des transitions beaucoup plus franches qu'avec une diffusion Laplacienne puisque contrairement à un vrai processus de diffusion, ici seuls les points directement visibles des contraintes influencent un pixel donné. Par exemple, l'extrémité des courbes diffuse beaucoup moins dans le cas des PMVC. Cela est dû au fait que les points dans l'alignement de la courbe n'ont typiquement que peu de visibilité sur celle-ci, voir pas du tout lorsqu'il s'agit d'une droite. Un autre problème peut apparaître quand l'utilisateur utilise des barrières : on peut alors imaginer des points pour lesquels aucune source de couleur n'est visible, il est alors impossible de calculer leur couleur. Il semble cependant raisonnable de supposer que ce cas est peu fréquent en pratique.

Bien que l'objectif initial était de s'affranchir d'une étape de résolution globale et coûteuse, les solveurs à base de PMVC s'avèrent en pratique tout aussi coûteux en calcul, voir même plus selon les modèles et la qualité souhaitée. Pour obtenir des performances similaires aux solveurs multi-grilles, ces méthodes doivent utiliser des structures telles que des triangulations pour éviter d'avoir à évaluer chaque pixel, ce qui complique d'autant plus leur mise en œuvre.

Une approche différente utilisant les *fonctions de Green* a été proposée récemment par Sun *et al.* [2012] pour obtenir une solution explicite au problème de diffusion. Cette méthode permet de s'affranchir des problèmes de visibilité ce qui simplifie les calculs, mais nécessite en contrepartie l'évaluation d'une intégrale sur *toutes les contraintes* ce qui peut rapidement s'avérer assez lourd. En outre, il semble difficile d'utiliser la plupart des extensions vues précédemment avec cette approche. Pour finir, cette méthode ne fonctionne théoriquement que pour des courbes fermées, même si elle semble donner des résultats acceptables en pratique.

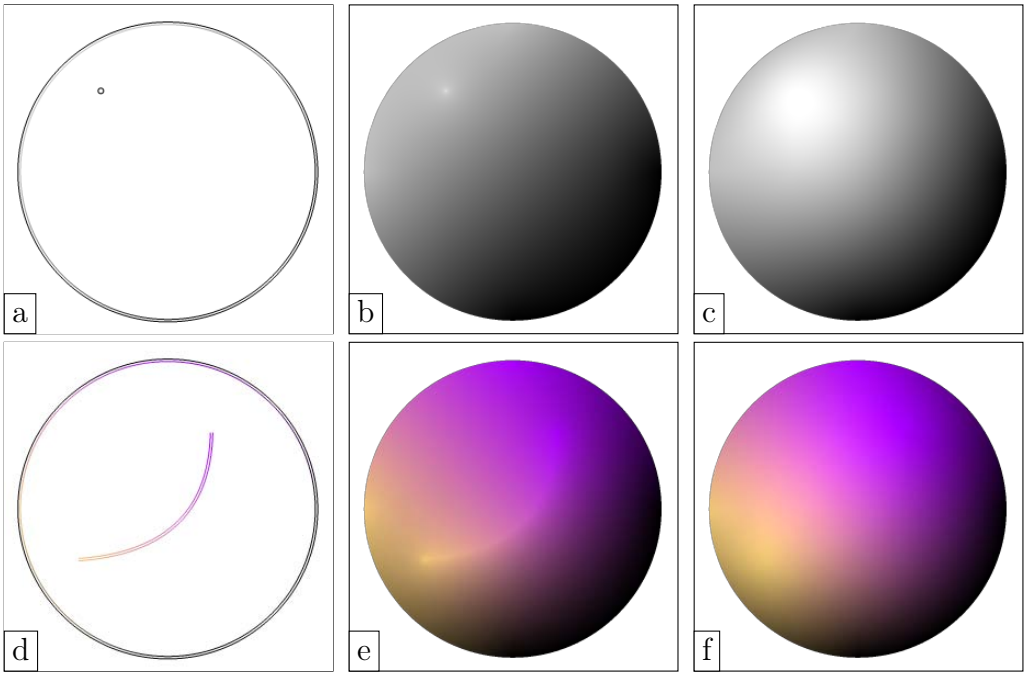


FIGURE 3.11 – **Comparaison entre la diffusion Laplacienne et bi-Laplacienne.** Les contraintes ponctuelles (a) produisent des "pics" de couleur et n'ont quasiment pas d'impact à distance avec la diffusion Laplacienne et produisent un résultat très lisse avec la diffusion bi-Laplacienne. Un ensemble de courbes simples représentant une sphère (d) diffusée à l'aide de l'équation Laplacienne (e) est nettement moins lisse que la version bi-Laplacienne (f).

3.3 Diffusion bi-harmonique

La diffusion Laplacienne produit des résultats lisses et plaisants, mais n'est pas pour autant exempte de défauts :

Continuité. La solution de la diffusion Laplacienne est C^1 en tout point, sauf sur les contraintes où elle n'est que C^0 sur les courbes émettant la même couleur des deux côtés, et discontinue sinon. Cela rend les courbes émettant la même couleur des deux côtés difficilement utilisables pour contrôler précisément les dégradés, comme illustré figure 3.11.e. La discontinuité du gradient est alors clairement visible, révélant ainsi des courbes qui ne sont pas destinées à être vues.

Contraintes ponctuelles. Les contraintes ponctuelles (voir figure 3.11.b) ne produisent généralement pas le résultat voulu. En pratique, l'influence d'une contrainte ponctuelle a tendance à s'estomper très rapidement ce qui, combiné au fait que les contraintes ne sont que C^0, crée un"pic" de couleur.

Plutôt que de résoudre l'équation Laplacienne 3.1 et produire une interpolation harmonique, Finch *et al.* [2011] ont proposé de réaliser une interpolation bi-harmonique obtenue en résolvant l'équation bi-Laplacienne :

$$\Delta^2 u = 0 \tag{3.6}$$

Cela produit une interpolation bi-harmonique qui est connue pour produire des fonctions plus lisses (i.e., C^2), qui sont de plus naturellement C^1 sur les contraintes de valeur. Dans notre contexte, cela produit des dégradés qui sont particulièrement adaptés pour représenter des effets d'ombrage, comme illustré figures 3.11.c et 3.11.f. Cela permet de réaliser des images à l'apparence beaucoup plus réaliste que la diffusion Laplacienne, qui tend à ne produire que des dégradés linéaires.

Contraintes de gradient Le comportement de la diffusion bi-harmonique permet de contrôler le gradient au niveau des courbes. Pour les courbes de diffusion, Finch *et al.* proposent trois types de contraintes. Le premier consiste à simplement forcer une continuité C^1, ce qui est le comportement par défaut. Le second consiste à contraindre le gradient à être nul perpendiculairement aux courbes. Ce type de courbes est appelé *slope*. Le troisième, appelé *crease*, consiste à casser la continuité C^1 pour obtenir des variations de couleur marquées.

La diffusion bi-harmonique offre d'avantage de contrôles de gradient que la diffusion harmonique [Hnaidi *et al.*, 2010]. Par exemple, dans le cas de la diffusion Laplacienne, il n'est pas possible d'avoir une continuité de gradient de part et d'autre d'une courbe de valeur sans contraindre explicitement la valeur du gradient, alors que cela se fait naturellement avec la diffusion bi-harmonique.

Contraintes ponctuelles Comme nous l'avons déjà mentionné, l'interpolation bi-harmonique produit des fonctions C^1 sur les contraintes. Cela signifie que les contraintes ponctuelles produisent des dégradés lisses, et non pas des "pics" comme le fait l'interpolation harmonique. Comme pour les courbes, dans leur système Finch *et al.* proposent de soit laisser le gradient des points libre soit de le contraindre à être nul.

Interpolation bi-harmonique en synthèse d'images Dans le contexte de la modélisation géométrique, des EDP de quatrième ordre similaires sont fréquemment employées pour contrôler la déformation de maillages triangulaires. Ces EDP sont soumises à des contraintes d'ordre zéro et de premier ordre [Welch et Witkin, 1992; Botsch et Kobbelt, 2004]. Voir par exemple [Botsch et Sorkine, 2008] pour un tour d'horizon plus complet de ces techniques. Ici, l'équation bi-harmonique est le plus souvent discrétisée directement

sur le maillage triangulaire (à la manière des différences finies) à l'aide d'approximations discrètes des opérateurs différentiels [Meyer *et al.*, 2002b]. Il s'agit d'une approche simple et efficace, mais ces approximations peuvent conduire à des artefacts notables [Grinspun *et al.*, 2006]. De plus, les contraintes de premier ordre (contraintes de tangente) sont spécifiés en fixant les valeurs d'une bande de triangles. Pour contourner ces limitations, Grinspun *et al.* [2006] décrivent une approche employant des éléments finis non conformes quadratiques. Plus récemment, Jacobson *et al.* [2010] ont présenté une discrétisation se basant sur une méthode par éléments finis mixtes qui montre le même ordre de complexité que les approximations précédentes, mais plus solide d'un point de vue mathématique tout en permettant de gérer de manière transparente des contraintes de premier ordre. Néanmoins, toutes ces techniques ont été développées spécifiquement pour les maillages linéaires, tandis que, comme nous le montrons section 4.3, les gradients de couleur requièrent des éléments d'un ordre supérieur.

3.4 Bilan

Les images de diffusion sont des outils puissants permettant de créer une large variété d'images. De nombreuses extensions ont été proposées afin d'élargir encore leurs possibilités tout en facilitant leurs utilisations. Néanmoins, leur grand nombre ainsi que la complexité de certaines les rendent difficiles à utiliser en pratique, car elles demandent un temps d'apprentissage non négligeable. De plus, certaines extensions proposées semblent surtout exister pour contourner les limitations des solveurs actuels, comme les portails, qui peuvent être avantageusement remplacés par des calques bien plus faciles à manipuler. En effet, la gestion *efficace* des calques fait cruellement défaut aux images de diffusion, d'autant plus qu'ils sont aujourd'hui standards dans tous les logiciels d'édition d'images et que les artistes sont habitués à leur utilisation. Cette limitation vient principalement du fait que les solveurs actuels doivent recalculer l'intégralité de l'image à chaque changement de point de vue : l'utilisation de n calques divise simplement les performance par n, ce qui est prohibitif.

Le problème de diffusion reste un problème difficile à résoudre efficacement. Le solveur de Jeschke *et al.* [2009a] est sans nul doute le plus rapide, mais également le moins flexible puisqu'il ne permet que de réaliser une diffusion Laplacienne de base, tout en nécessitant un GPU haut de gamme. Les autres solveurs peinent à offrir les performances suffisantes pour permettre une édition interactive. Cela inclut les solveurs à évaluation directe : bien que leur approche semble séduisante, ceux-ci ne s'avèrent pas plus rapides que les solveurs multigrilles tout en étant restreint à une approximation de la diffusion Laplacienne.

Finalement, nous avons vu que la diffusion bi-Laplacienne produit des dégradés bien plus lisses et plaisants à l'œil, comme illustré figure 3.12. En com-

FIGURE 3.12 – **Images de diffusion complexes.** Le zephir (a) est produit à l'aide d'une diffusion Laplacienne plus une passe de flou, principalement utilisée sur le visage. Le neko (b) est, lui, produit à l'aide d'une diffusion bi-harmonique en une seule passe. La diffusion bi-harmonique permet de créer des dégradés proches de ce que produit naturellement l'éclairage d'une scène ce qui permet de faire des images plus plausibles. L'image (a) est extraite de [Orzan *et al.*, 2008].

paraison, les images obtenues à l'aide de la diffusion Laplacienne semblent manquer de relief. Pour l'instant, seul un solveur multi-grille a été mis au point pour ce type d'images de diffusion d'ordre supérieur.

Chapitre 4

Solveur vectoriel pour les images de diffusion

Comme nous l'avons vu dans le chapitre précédent, le principe des images de diffusion est un concept très séduisant pour la création d'images vectorielles "riches". En revanche, sa non adoption par les logiciels de dessin vectoriel s'explique principalement par les lacunes des solveurs qui ont été proposés jusqu'à présent. En effet, d'une part les solveurs à évaluation directe sont limités à une approximation d'une interpolation harmonique, plus limité que l'interpolation bi-harmonique, et ceux-ci impliquent des calculs de visibilité coûteux. D'autre part, les solveurs s'appuyant sur une discrétisation par différences finies en espace image gèrent difficilement les contraintes proches (zooms arrières) et les contraintes hors de l'écran (zoom avant), ce qui les rend difficilement utilisables en pratique. De plus, la nécessité de recalculer la solution à chaque changement de point de vue implique des performances inacceptables lors de l'utilisation de calques, qui sont pourtant omniprésent dans les logiciels de dessin.

Afin de contourner ces difficultés, nous proposons le concept de *solveur vectoriel*. Ici, l'idée est de résoudre le problème de la diffusion de manière à produire en sortie une *représentation intermédiaire* vectorielle possédant toutes les bonnes caractéristiques des primitives vectorielles traditionnelles : elle est indépendante de la résolution en sortie, légère en mémoire, adaptative, permet de représenter les discontinuités fidèlement et peut être affichée très efficacement. L'aspect adaptatif de la représentation intermédiaire signifie qu'elle doit être capable de représenter fidèlement et efficacement des images contenant à la fois des motifs grossiers et des détails extrêmement fins. L'indépendance vis à vis de la résolution (en pixels) de sortie est fondamentale puisque cela permet de résoudre le problème une fois pour toutes sur l'intégralité du domaine, tout en autorisant un grand nombre d'opérations sans re-calcul. Par exemple, les zooms, rotations, translations et autres transformations de l'espace sont ainsi effectués avantageusement sur la représentation intermédiaire sans problème de contraintes hors-champs ou de superposition des contraintes. Cela permet également une gestion efficace de multiples calques (en pratique un seul calque est modifié à la fois), ainsi que la création d'images complexes peuplées de différentes instances d'un même motif.

Une autre difficulté majeure des images de diffusion est que la diffusion est par définition *globale*, ce qui implique que la modification d'une contrainte puisse avoir un impact à très grande distance. Afin d'offrir la possibilité d'une diffusion *locale*, nous proposons le concept de *courbes de transmission* qui permettent, entre autre, de contrôler précisément l'influence d'une primitive.

Dans ce chapitre, nous présentons un nouveau solveur vectoriel utilisant une triangulation comme représentation locale et une méthode d'éléments finis pour le calcul de la diffusion. Tout d'abord nous présentons une classification des contraintes qui adressent les limitations de la notation de Finch *et al.* [2011], ainsi qu'un nouveau type de courbes appelées *courbes de transmission* en section 4.1. Ensuite nous présentons un bref aperçu de notre nouvelle approche

pour le calcul des images de diffusion en section 4.2. Notre représentation intermédiaire est détaillée section 4.3, sa construction section 4.4, et le solveur FEM section 4.5. Pour finir, nous présentons les résultats de notre solveur dans plusieurs cas d'application et discutons des avantages et inconvénients de notre approche en section 4.6.

4.1 Classification et transmission

4.1.1 Classification des types de courbes

Au vu du nombre d'extensions et des possibilités offertes par la diffusion bi-harmonique, un grand nombre de combinaisons sont possibles et il est difficile d'avoir une vue d'ensemble des possibilités des méthodes de diffusion récentes. Finch *et al.* [2011] ont proposé un système de notation consistant à combiner différentes courbes élémentaires : les courbes bloquantes indiquant des discontinuités (T), les contraintes de valeur (V), les discontinuités C^0 (C), les courbes forçant la magnitude du gradient dans le sens normal à être nul (S) et pour finir les courbes de contour (N). Cependant, ce système de notation souffre d'un certain nombre de lacunes. En effet, certaines combinaisons n'ont pas de sens, comme par exemple les courbes VN (valeur et contour à la fois). L'ordre des courbes est parfois important, comme pour les courbes VT et TV correspondant à une courbe bloquante émettant de la couleur respectivement à gauche et à droite, mais pas systématiquement comme pour les courbes VC et CV (dans les deux cas, une courbe de valeur avec une continuité C^0). Finalement, il est possible d'imaginer des types de courbes non-représentables avec cette notation, par exemple, une courbe dont le gradient ne serait contraint que d'un côté.

Afin de faciliter la navigation dans l'espace des possibilités offertes par les différentes combinaisons de contraintes, nous avons mis en place une notation plus systématique. D'une manière générale, nous disposons de deux types de contraintes qui peuvent être fixées indépendamment : les contraintes de valeur et les contraintes de gradient. En notant $u_{\mathcal{C}}$ la contrainte de u le long d'une courbe $\mathcal{C}$, nous pouvons résumer l'espace des contraintes possibles dans le diagramme suivant :

où l et r représentent les côtés "gauche" et "droit" respectivement. Par exemple, le côté droit de $\mathcal{C}$ peut se voir assigner une fonction scalaire spécifique ν_r ou bien être laissé non-spécifié (nous utilisons le symbole $\otimes$ dans ce cas). De plus, le

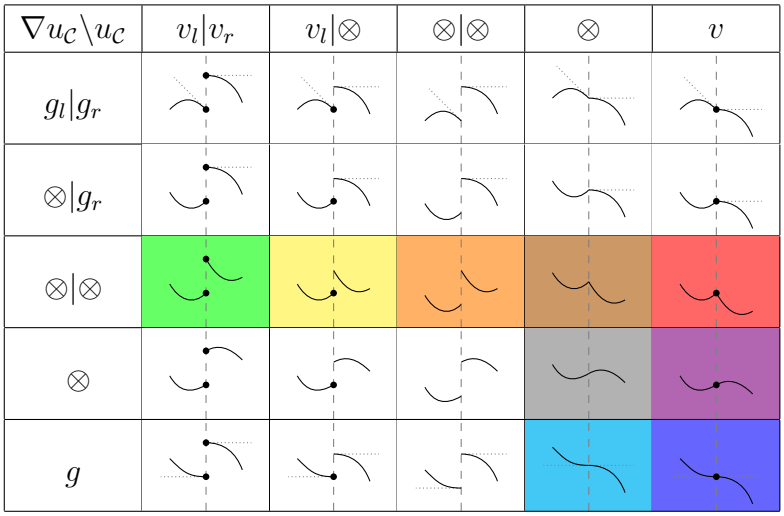


FIGURE 4.1 – **Types de courbes.** En combinant les contraintes de valeur (colonnes) et de gradient (lignes), nous obtenons un ensemble de courbes riches. Les points et les lignes pointillées correspondent respectivement aux contraintes de valeur et de gradient. Cette façon de classer les courbes généralise les travaux précédents, comme indiqué par les couleurs des cellules qui correspondent aux types de courbes de Finch *et al.* [2011] : V_1TV_2, VT, T, C, CV, V, VS, S. La courbe nulle n'a aucun effet puisqu'elle ne définit aucune contrainte ni aucune discontinuité ; il s'agit du comportement de la diffusion en tout point non-contraint.

côté gauche et le côté droit peuvent être contraints à être égaux (nous écrivons alors $u_C = \nu$). Des choix similaires sont disponibles pour les contraintes de gradient. L'égalité des contraintes est nécessaire lorsque les valeurs, ou les normes des gradients, sont laissées non-spécifiées et que l'on souhaite forcer la continuité C^0 ou C^1. Il est bien sûr possible de faire varier les contraintes de valeur et de gradient le long des courbes.

Le cas des contraintes ponctuelles est légèrement différent : elles sont traitées comme des contraintes isotropes et il est possible de contraindre leur valeur, la norme de leur gradient, mais aussi la direction du gradient. En d'autres termes, si nous considérons un champ de hauteur, une contrainte ponctuelle permet de spécifier (ou non) l'altitude du point, et/ou la normale de la surface générée.

Ce petit ensemble de paramètres permet d'obtenir un grand nombre de types de courbes. Ceux-ci sont résumés dans la figure 4.1 qui illustre toutes les combinaisons de contraintes de valeur et de gradient possibles, chaque cellule représentant un type de courbe. Cette façon de classifier les contraintes généralise les courbes de diffusion et certaines extensions présentées dans les travaux précédents, tels que les courbes bloquantes et des contraintes de gra-

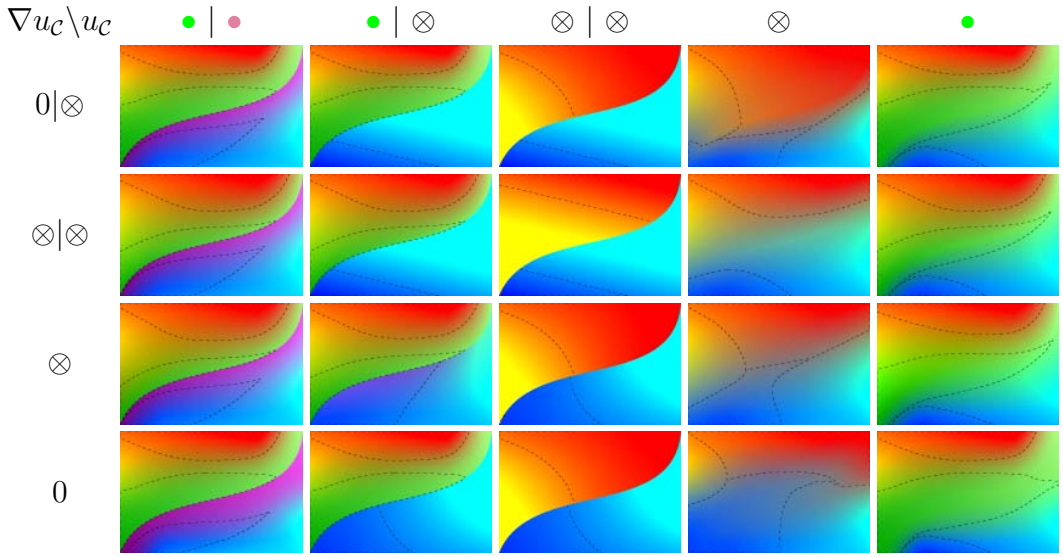


FIGURE 4.2 – **Types de courbes en image.** Les quatre dernières lignes de la figure 4.1 illustrées avec des dégradés de couleur. Deux courbes, en haut et en bas, émettent de la couleur et restent inchangées, mais la courbe centrale est modifiée. Les lignes pointillées soulignent les iso-contours des couleurs pour faciliter la visualisation.

dient. Les types de courbes représentables avec la méthode de Finch *et al.* sont indiqués par des couleurs, les cases à fond blanc sont donc les nouveaux types de courbes que fait apparaître notre classification. Il est important de noter que tous les types de courbes présents dans le tableau peuvent être mis en œuvre avec un solveur adapté, tel que celui que nous présentons dans ce chapitre. Un sous-ensemble des différents types de courbes est illustré figure 4.2 avec des dégradés de couleur.

Comparé à la notation proposée par Finch *et al.*, cette classification offre plus de possibilités tout en évitant les cas insensés, mais ne prend pas en compte les courbes de contour qui sont équivalentes à des contraintes de valeur calculées automatiquement. Les autres extensions, telles que l'application de flou en post-traitement ou la diffusion de couleurs homogènes, ne sont pas représentées ici car elles ne créent pas de nouveaux types de contraintes sur le processus de diffusion.

De prime abord, certains types de courbes, que fait apparaître notre classification, ne semblent pas très intéressants en pratique. Par exemple, spécifier des valeurs de gradient n'a que peu de sens dans le cas de l'édition d'images, sauf dans le cas $\Delta u_{\mathcal{C}} = 0$ que nous utilisons régulièrement. En revanche, ces contraintes s'avèrent particulièrement utiles pour la création de champs de hauteurs (terrains), puisqu'elles permettent alors de contrôler la normale de

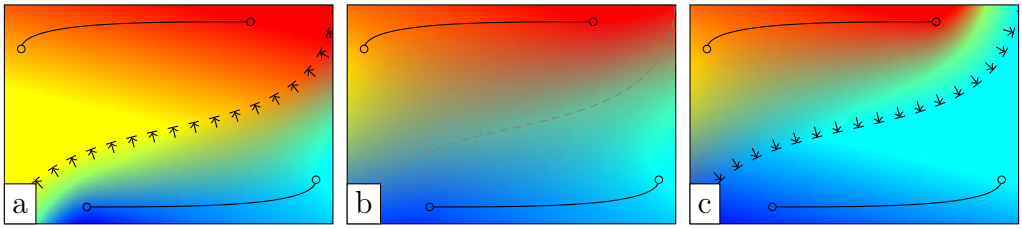


FIGURE 4.3 – **Transmission.** La courbe centrale est une courbe de transmission, avec $\alpha = -1$ (a), $\alpha = 0$ (b) et $\alpha = 1$ (c).

la surface sous-jacente. Par ailleurs, il est possible de forcer une continuité des gradients sans avoir une continuité des couleurs (ligne $\otimes$, trois premières colonnes), ce qui semble assez peu utile en pratique, particulièrement dans le cadre d'édition de dégradés de couleur où cela donne des résultats imprévisibles.

4.1.2 Transmission

Il est souvent utile, en pratique, de limiter précisément la diffusion d'une primitive sans pour autant introduire une discontinuité comme le font les courbes bloquantes. Pour cela nous proposons un nouveau type de courbes que nous appelons *courbes de transmission*. Ces courbes n'émettent pas de couleur et sont contraintes à être au moins C^0 (colonne $\otimes$ dans notre classification). Le but de ces courbes est de bloquer la diffusion seulement dans un sens, mais pas dans l'autre, ce qui permet de contrôler très précisément sur quelle zone une courbe peut avoir de l'influence, comme illustré figure 4.3. La transmission peut être contrôlée finement grâce à un paramètre $\alpha \in [-1, 1]$ qui permet de passer de façon lisse d'un blocage total des couleurs provenant du côté droit d'une courbe ($\alpha = -1$), à une courbe qui laisse passer la diffusion dans les deux sens ($\alpha = 0$), puis à un blocage des couleurs provenant du côté gauche ($\alpha = 1$).

Cette extension permet un contrôle bien plus précis que la méthode des couleurs homogènes de Bezerra *et al.* en offrant un vrai contrôle *local*. Il est important de rappeler que bien qu'une courbe de transmission soit particulièrement utile pour contrôler l'influence d'une primitive, ces courbes sont complètement indépendantes et ne sont absolument pas liées à d'autres primitives. Cela offre énormément de possibilités, mais demande également un usage raisonné. Par exemple des courbes de transmission ouvertes n'ont que peu de sens, car les couleurs ont alors tendance à contourner la courbe. D'un point de vue interface utilisateur, un outil de génération de courbes décalées [Ostromoukhov, 1993] nous semble particulièrement pertinent pour créer une version initiale d'une courbe de transmission qui peut ensuite être déformée à souhait pour obtenir

FIGURE 4.4 – **Exemple de transmission.** Les courbes de transmission permettent de contrôler précisément la zone d'influence d'une courbe (ici en forme d'étoile) tout en mélangeant le résultat naturellement avec le reste de l'image.

l'effet désiré.

4.2 Aperçu de notre solveur *vectoriel*

Le solveur que nous proposons ici utilise une triangulation comme représentation intermédiaire et se base sur une *méthode par éléments finis* (FEM pour *finite element methods*) [Zienkiewicz *et al.*, 2005]. Les FEM définissent une classe d'approches généralistes pour la résolution des *équations aux dérivées partielles* (EDP). Notre solveur fonctionne aussi bien avec la diffusion Laplacienne que bi-Laplacienne.

Notre méthode se divise en deux grandes étapes : premièrement la construction de la représentation intermédiaire, deuxièmement la résolution du problème de diffusion. Un diagramme représentant les différentes étapes de notre méthode est présenté figure 4.5.

Construction de la triangulation. La qualité du résultat ainsi que les performances de notre solveur sont directement dépendantes de la qualité de la représentation intermédiaire. Celle-ci est décrite en section 4.3. Pour obtenir le meilleur rapport qualité/temps d'exécution, il faut s'assurer que la triangulation soit suffisamment dense là où il y a des détails tout en prenant soin d'éviter de sur-échantillonner le maillage inutilement. En résumé, nous utilisons un raffinement de Delaunay, qui prend en entrée un ensemble de segments appelé *planar straight-line graph* (PSLG) et qui produit une triangulation très régulière. Afin d'obtenir une triangulation conforme à nos attentes, nous avons mis au point plusieurs heuristiques pour générer un PSLG optimisé évitant différents artefacts et les sur-échantillonnages inutiles. Les détails de cette étape sont donnés en section 4.4.

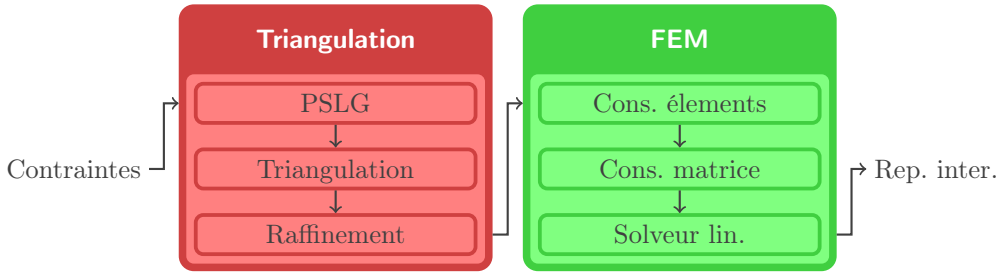


FIGURE 4.5 – **Aperçu de notre solveur *vectoriel*.** Il prend en entrée l'ensemble des contraintes (courbes et ponctuelles) sous la forme d'un *graphe planaire* qui est d'abord *discrétisé* pour obtenir un PSLG. Afin de garantir un échantillonnage suffisant près des points singuliers, *des sommets sont insérés* à leurs proximités après triangulation, puis le *raffinement de Delaunay* prend place pour uniformiser les triangles. Chaque triangle est alors *converti en élément*, opération durant laquelle un certain nombre de *nœuds* sont créés, représentant soit des contraintes soit des inconnues. Un *système d'équation linéaire* est alors construit en utilisant la FEM, puis résolu à l'aide d'un *solveur linéaire creux*. Le résultat est notre représentation intermédiaire.

Discrétisation du problème de diffusion. Une fois que la triangulation est construite, nous la transformons en un ensemble d'*éléments*. Cela consiste principalement à choisir le type d'élément associé à chaque triangle, et à insérer des *nœuds* correspondant aux degrés de liberté des éléments. La valeur d'un nœud peut être soit fixée par les contraintes en entrée, soit correspondre à une inconnue de notre problème. Nous discrétisons ensuite le problème de diffusion sur cet ensemble d'éléments à l'aide de la méthode des éléments finis, ce qui nous donne un système d'équations linéaires creux résolu à l'aide d'un solveur direct. Les détails du choix des éléments, de la discrétisation des contraintes ainsi que les méthodes mises en œuvre pour gérer les courbes de contour et la transmission sont détaillés section 4.5.

4.3 Représentation intermédiaire

Notre représentation intermédiaire est conçue pour représenter le résultat de la diffusion $\Delta^n u = 0$ $(n = 1, 2)$ à l'aide de primitives *vectorielles* légères et indépendantes de la résolution. Elle n'est cependant pas destinée à être manipulée directement car le nombre de paramètres nécessaires pour représenter convenablement la solution de la diffusion est généralement trop important.

Plusieurs choix sont possibles pour une représentation intermédiaire. Par exemple, les *quadtrees* permettent de discrétiser un domaine très rapidement [Frey et Marechal, 1998] mais souffrent essentiellement des mêmes limitations qu'une grille régulière pour s'adapter fidèlement aux courbes de contraintes.

De même, les représentations *meshless* [Belytschko *et al.*, 2004; Liu, 2009] sont faciles à générer et manipuler puisqu'il n'y a pas de connectivité à gérer. De plus, ces approches sont connues pour produire des résultats très lisses. En revanche, celles-ci ne sont pas adaptées pour représenter les discontinuités et sont assez coûteuses à évaluer. Le choix de se baser sur des triangulations est finalement assez naturel. Il existe en effet de nombreux outils pour les générer, et elles permettent de représenter des formes arbitrairement complexes ce qui nous permet de représenter les contraintes précisément. De plus les cartes graphiques modernes sont optimisées pour tracer des triangles.

Arêtes courbes. Bien qu'il soit possible d'approcher de manière suffisamment fidèle une courbe par une ligne polygonale, cela conduirait à une triangulation excessivement dense à proximité des courbes, ce qui serait extrêmement coûteux à la fois en terme de consommation mémoire et de temps de calcul. En pratique, nous avons constaté que la triangulation n'a généralement pas besoin d'être excessivement dense près des courbes pour produire des dégradés visuellement plaisants. Nous nous autorisons donc à avoir des arêtes courbes (représentées par des courbes de Bézier cubiques dans notre implémentation) nous permettant de représenter les contraintes et les discontinuités *très précisément* tout en conservant un nombre de triangles relativement faible.

Patch de couleur. À chaque triangle est associé un *patch*[1] u_i interpolant les couleurs définies en des points précis appelés *nœuds*. Formellement, l'image finale u peut donc être définie par :

$$u(\mathbf{x}) = \sum_i u_i(\mathbf{x}) = u_{\mathrm{idx}(\mathbf{x})}(\mathbf{x}) \ , \tag{4.1}$$

où $\mathrm{idx}(\mathbf{x})$ correspond à l'index de l'unique triangle contenant le point $\mathbf{x}$. Les patchs les plus simples possèdent trois nœuds de valeur v_j associés aux sommets $\mathbf{p}_j$ (voir figure 4.6.a) qui sont interpolés linéairement par $u_i(\mathbf{x}) = \sum_{j=1}^{3} v_j L_j(\mathbf{x})$, où L_1, L_2, et L_3 correspondent aux coordonnées barycentriques du points $\mathbf{x}$ dans le triangle i. De tels patchs ont été utilisés par Pang *et al.* [2012]. Cependant, les dégradés linéaires produisent d'importantes discontinuités du gradient au niveau des arêtes qui sont particulièrement visibles, à moins de créer une triangulation excessivement dense.

Une meilleure solution consiste à utiliser des patchs d'ordre supérieur. Nous avons opté pour une interpolation Lagrangienne d'ordre deux où les nœuds sont placés sur les sommets et au milieu des arêtes de la triangulation tel qu'illustré

1. Les fonctions d'interpolation utilisées pour l'affichage sont appelées *patchs* par opposition aux *éléments* utilisés pour la résolution du problème. Bien que ces notions soient similaires, nous verrons en section 4.5 que nous ne pouvons pas toujours utiliser le mêmes fonctions dans les deux cas.

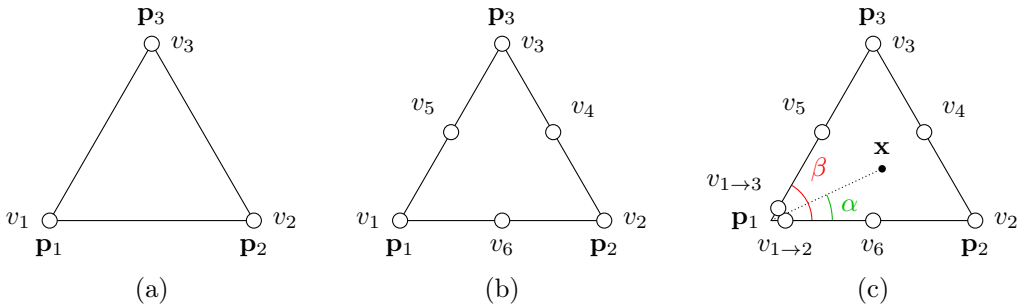


FIGURE 4.6 – **Patchs de couleur.** Les patchs linéaires possèdent trois nœuds placés au sommets (a), et les patchs Lagrangiens quadratiques (b) en possèdent six : trois sur les sommets et trois autres au centre des arêtes. Les patchs incidents à des singularités sont traités spécialement : le sommet singulier $\mathbf{p_1}$ est représenté par deux nœuds $v_{1\to2}$ et $v_{1\to2}$ (c). Le point $\mathbf{x}$ est évalué comme les patchs normaux en utilisant $v_1(\mathbf{x}) = v_{1\to2} + \frac{\alpha}{\beta}(v_{1\to3} - v_{1\to2})$.

figure 4.6.b. Nous avons donc $u_i(\mathbf{x}) = \sum_{j=1}^{6} v_j Q_j(\mathbf{x})$, où les fonctions de base Q_j associées aux nœuds v_j sont données par :

$$Q_j = (2L_j - 1)L_j, \qquad j = 1, 2, 3,$$

$$Q_4 = 4L_2L_3, \qquad Q_5 = 4L_3L_1, \qquad Q_6 = 4L_1L_2\ .$$

Formellement, le gradient de l'image n'est toujours pas continu le long des arêtes. Cependant, comme illustré figure 4.7, lorsqu'il s'agit de représenter des dégradés de couleur, ces patchs produisent des résultats bien supérieurs à ceux des patchs linéaires, même pour un nombre de nœuds et un temps de calcul similaire. Nous n'avons pas considéré les patchs d'ordre supérieur car cela compliquerait fortement le solveur, particulièrement avec la diffusion bi-Laplacienne. De plus, il n'est pas évident que la différence soit visible étant donné la qualité déjà obtenue avec des patchs quadratiques.

Représentation des singularités. Les extrémités des courbes et les intersections sont des points *singuliers* où deux valeurs différentes peuvent être contraintes en même temps. Les solveurs existants ne permettent pas de gérer ces singularités explicitement. Ceux-ci sont donc sources d'artefacts : scintillements dûs à la discrétisation des contraintes dans le cas des solveurs en espace image, et discontinuités dans les solveurs reposant sur une triangulation, car ces singularités ne peuvent pas être représentées correctement par des patchs triangulaires standards. Par exemple, Pang *et al.* [2012] se contentent de créer une triangulation très dense autour des singularités pour masquer la discontinuité engendrée par ces sommets. Nous avons préféré mettre au point une solution plus élégante utilisant des patchs spécifiques, ce qui nous permet,

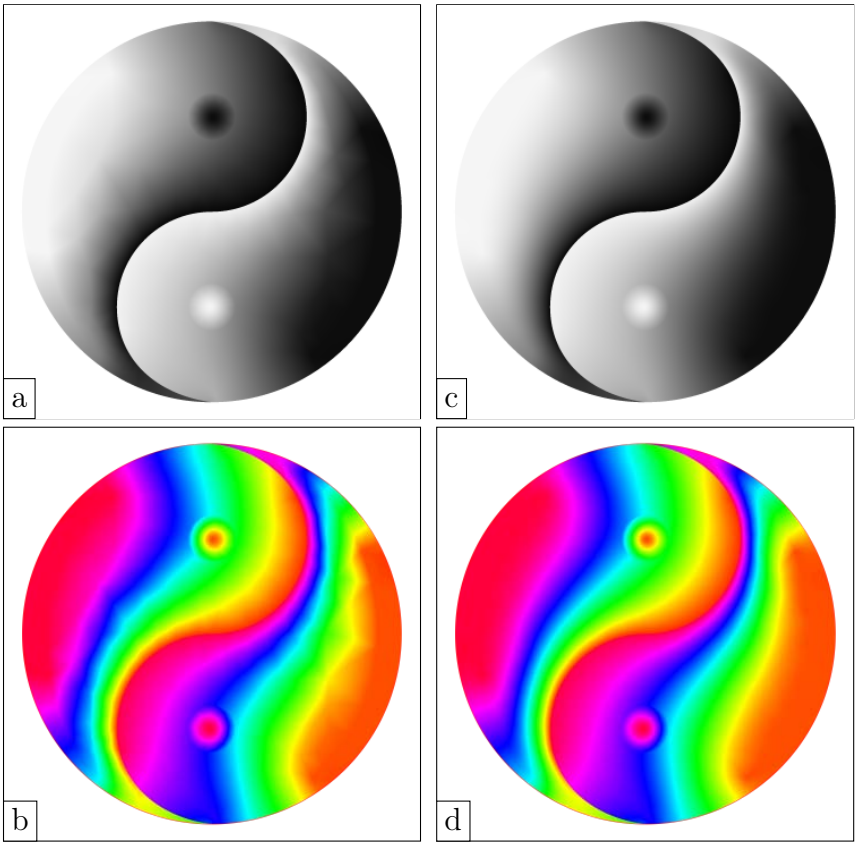


FIGURE 4.7 – **Comparaison des patchs linéaires (a et b) et quadratiques (c et d).** Les images (b) et (d) correspondent respectivement aux images (a) et (c) avec le niveau de gris codé en couleur pour mieux visualiser les différences. L'image (a) est constituée de 2400 patchs linéaires et l'image (c) de 1200 patchs quadratiques, ce qui correspond dans les deux cas à environ 5000 nœuds. Les deux images ont été calculées en environ 60ms chacune.

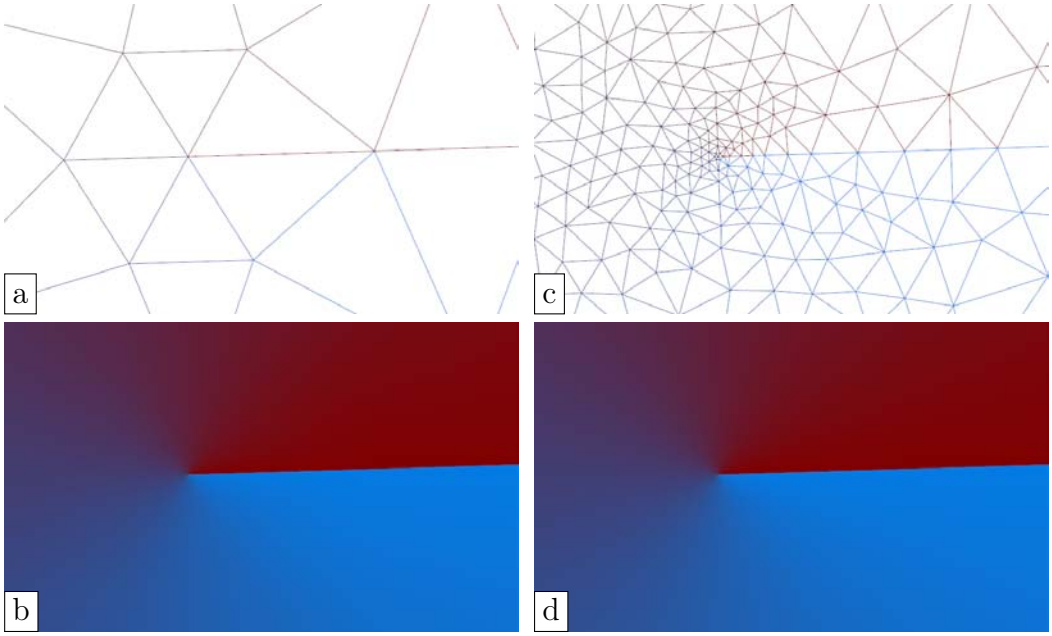


FIGURE 4.8 – **Patchs singuliers.** Les patchs singuliers (a et b) produisent un résultat visuellement indistinguable d'un maillage très dense (c et d).

avec très peu de patchs, d'obtenir une qualité équivalant celle d'un maillage raffiné infiniment comme illustré figure 4.8. Le nœud de valeur v_1 correspondant au point singulier d'un *patch singulier* est dédoublé en deux nœuds de valeurs $v_{1\to2}$ et $v_{1\to3}$ correspondant respectivement à la couleur émise dans la direction de $\mathbf{p}_2$ et $\mathbf{p}_3$ (voir la figure 4.6.c). Ils sont évalués comme les patchs réguliers, en faisant de v_1 une fonction dépendante du point d'évaluation $\mathbf{x}$ effectuant une interpolation linéaire entre $v_{1\to2}$ et $v_{1\to3}$ basée sur la position angulaire relative :

$$v_1(\mathbf{x}) = v_{1\to2} + \frac{\alpha}{\beta}\left(v_{1\to3} - v_{1\to2}\right) \ , \tag{4.2}$$

où $\alpha = \widehat{\mathbf{p}_2\mathbf{p}_1\mathbf{x}}$ et $\beta = \widehat{\mathbf{p}_2\mathbf{p}_1\mathbf{p}_3}$. Bien entendu, la triangulation doit être construite de manière à ce qu'il n'y ait pas plus d'un nœud singulier par patch.

4.4 Triangulation du domaine

Comme nous l'avons déjà mentionné, la triangulation du domaine est une étape cruciale de notre approche car elle a un impact direct sur les performances et la qualité des résultats. En plus de chercher à construire une triangulation optimisée, vient s'ajouter la nécessité de générer des triangles suffisamment réguliers afin d'éviter les problèmes d'instabilité numérique. Comme

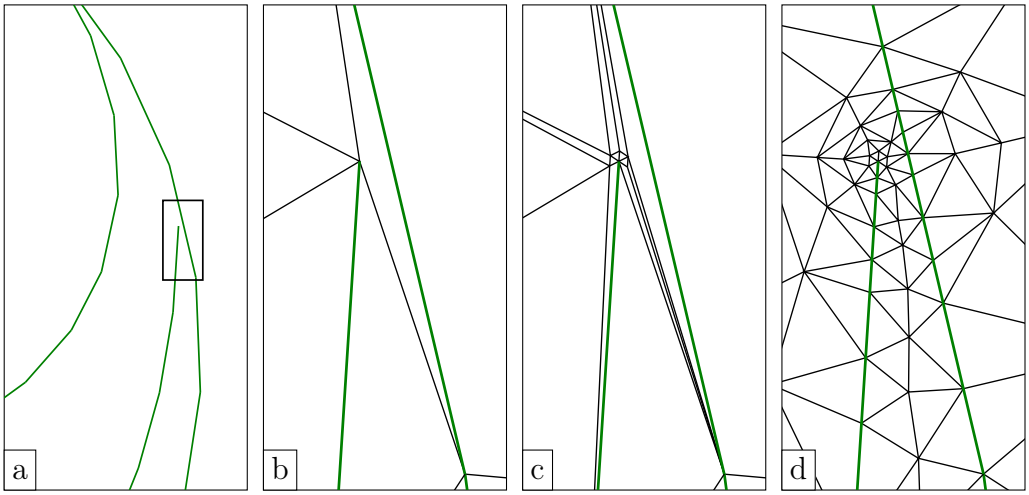


FIGURE 4.9 – **Construction de la triangulation.** La discrétisation des courbes produit le PSLG (a, en vert). Les autres figures présentent une vue de près de l'étape de triangulation : la triangulation de Delaunay (b) est utilisée pour insérer des sommets à proximité des points d'intérêts (c), puis le raffinement de Delaunay produit un maillage dense et régulier (d).

pour les surfaces de subdivision (chapitre 2, figure 2.4), des triangles réguliers améliorent également la qualité visuelle. Ces différents objectifs sont en conflit, et un compromis doit être fait entre qualité et rapidité des calculs. En pratique, la méthode de construction de la triangulation que nous présentons ici expose un certain nombre de paramètres permettant de facilement contrôler la qualité en sortie.

La triangulation du domaine se déroule en trois étapes illustrées figure 4.9. La première, détaillée section 4.4.1, consiste à discrétiser les courbes de façon à représenter fidèlement les contraintes et contrôler la densité de la triangulation à proximité des courbes, ce qui produit un PSLG. Puis, une triangulation de Delaunay contrainte est construite à partir du PSLG. Nous expliquons en section 4.4.2 comment cette triangulation nous permet de détecter facilement des situations où des raffinements spécifiques sont nécessaires : extrémités des courbes, intersections entre les arêtes du PSLG, etc. Pour finir, la triangulation est raffinée à l'aide d'un raffinement de Delaunay personnalisé décrit section 4.4.3. Avant d'étudier en détails chacune de ces étapes, nous allons motiver et présenter brièvement l'approche par raffinement de Delaunay car ce choix conditionne les autres étapes.

Raffinement de la triangulation. Le problème consistant à mailler un domaine de façon à obtenir les polygones les plus réguliers possible n'est pas nouveau et trouve de nombreuses applications en ingénierie, dont notamment

la génération de maillages pour les FEM. L'objectif est double : insérer des nœuds dans le domaine de façon à pouvoir approcher fidèlement la solution de la diffusion et éliminer les petits angles qui ont tendance à rendre le solveur instable.

Plusieurs stratégies existent dans ce but. Une première classe de techniques repose sur un partitionnement par quadtree suivit d'une triangulation s'adaptant aux contraintes [Frey et Marechal, 1998]. Il s'agit certainement de l'approche la plus rapide, mais les triangulations obtenues ainsi sont de très mauvaise qualité et doivent subir un certain nombre de traitements coûteux avant d'être exploitables. Des techniques à base de génération d'échantillonages pseudo-aléatoires permettent de générer très rapidement des triangulations [Donohue et Ostromoukhov, 2007]. Cependant, ces méthodes sont adaptées à la génération d'échantillonages relativement denses, alors que nous cherchons une triangulation parcimonieuse. Le raffinement de Delaunay [Shewchuk, 2000; Chernikov et Chrisochoides, 2006] est une méthode incrémentale produisant des maillages de qualité, insérant des points dans une triangulation de Delaunay contrainte jusqu'à ce que le plus petit angle soit supérieur à un angle paramétrable $\alpha_{\min}$. Cet algorithme est garanti de terminer si $\alpha_{\min} \leq 20,7°$. Comme les sommets de sont jamais supprimés ou déplacés lors du raffinement, cela en fait un algorithme plutôt performant qui peut même être parallélisé [Nave *et al.*, 2002; Spielman *et al.*, 2002]. D'un autre coté, la triangulation obtenue n'est généralement pas optimale puisque qu'un grand nombre d'insertions peut être nécessaire pour atteindre le critère de qualité, alors que le simple déplacement de certains sommets pourrait être suffisant. Il s'agit précisément de l'objectif des méthodes à base d'optimisations de maillages qui entrelacent des étapes de subdivisions/contractions avec des étapes de relaxations [Eppstein, 2001]. La subdivision peut être implémentée par du simple raffinement diadique ou $\sqrt{3}$ comme nous l'avons vu au chapitre 1, ou de manière plus sophistiquée par du raffinement de Delaunay [Chen, 2004; Tournois *et al.*, 2007]. L'étape d'optimisation repose généralement sur la notion de *centroidal Voronoi tessellation* [Du *et al.*, 1999; Liu *et al.*, 2009; Rong *et al.*, 2011] et de l'algorithme de Lloyd [1982]. Dans tous les cas, il s'agit de minimiser une énergie non linéaire, ce qui en fait donc une procédure coûteuse : Tournois *et al.* reportent des temps de calculs 100 fois plus important que pour un simple raffinement de Delaunay. En conclusion, afin de satisfaire nos contraintes d'interactivité, le raffinement de Delaunay semble de loin le meilleur compromis.

L'algorithme de raffinement de Delaunay fonctionne globalement comme suit : les triangles sont triés en fonction du ratio entre le rayon de leur cercle circonscrit et la longueur de leur plus petite arête. Les triangles avec les plus mauvais ratio sont alors récursivement raffinés en insérant un sommet au centre de leur cercle circonscrit, jusqu'à ce que tous les triangles aient atteint un

certain ratio, directement lié à l'angle le plus petit pouvant se trouver dans la triangulation. La principale difficulté vient du fait que de petits angles peuvent exister dans le PSLG, ils sont alors impossibles à éliminer et doivent être traités spécifiquement, comme expliqué dans l'article de Shewchuk [2000].

À proximité des contraintes, le raffinement de Delaunay produit généralement des triangles dont la longueur des côtés est similaire à la distance séparant les éléments du PSLG proches. En s'éloignant des contraintes, la triangulation devient de moins en moins dense. Ce comportement nous sied pour représenter des images de diffusion puisque les variations de couleur à proximité des contraintes sont généralement plus fortes qu'à distance.Cependant, cela signifie aussi que si deux sommets du PSLG se retrouvent inutilement proches, la triangulation risque d'être inutilement dense autour de ces points. La qualité de la triangulation en sortie est donc fortement dépendante de la qualité du PSLG, sa construction demande donc de prendre certaines précautions.

4.4.1 Discrétisation des courbes

La discrétisation des courbes se fait à l'aide d'un raffinement adaptatif, c'est à dire que les courbes sont récursivement subdivisées en deux tant qu'elles ne satisfont pas un certain critère (voir plus loin). Cette approche est efficace et échantillonne les courbes en évitant de placer des sommets trop proches les uns des autres, ce qui évite au raffinement de Delaunay de produire des maillages inutilement denses. Cependant, pour des raisons de performances, le critère de raffinement des courbes que nous utilisons est *intrinsèque*, autrement dit il détermine si une courbe doit être raffinée sans prendre en compte les autres courbes. Comme illustré figure 4.10, lorsque deux contraintes sont proches ou en intersection, cela peut engendrer des sommets inutilement proches qui contraignent la triangulation à être excessivement dense. Pour corriger cela, nous effectuons d'abord une étape de prétraitement destinée à détecter ces cas et à subdiviser les courbes de façon à ce que les points d'intersection soient situés aux extrémités des sous-courbes.

Prétraitements. Les extrémités des courbes et les points d'intersection se retrouvent obligatoirement dans le PSLG, il est donc nécessaire de les prendre en considération durant l'étape de discrétisation pour obtenir une triangulation optimisée. Les intersections ne sont pas prises en compte par l'algorithme de raffinement, qui peut alors insérer des sommets très proches de celle-ci, comme on peut le constater figure 4.10.a. Ce problème est résolu facilement en découpant les courbes aux points d'intersection dans une étape de prétraitement, ce qui produit le PSLG illustré figure 4.10.c.

Lorsque l'extrémité d'une courbe se trouve à proximité d'une autre courbe, cela engendre une triangulation très dense. Si *en théorie* il s'agit d'un com-

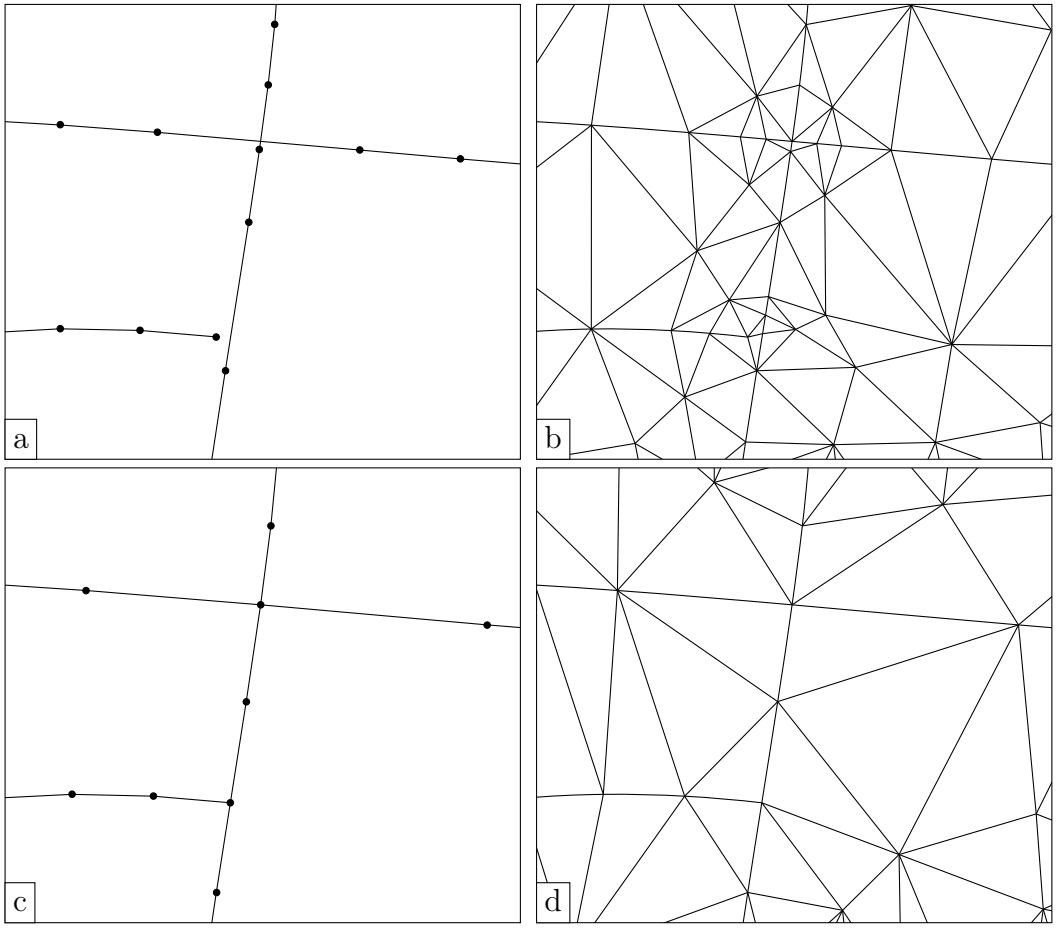


FIGURE 4.10 – **Prétraitement.** Le PSLG (a) est issu de courbes non traitées : on peut voir une intersection avec un sommet du PSLG très proche et une courbe qui s'arrête à proximité d'une autre, ce qui force la triangulation (b) à sur-raffiner ces zones. Le prétraitement divise les courbes aux points d'intersection avant la discrétisation et aimante les contraintes proches des courbes pour produire des intersections propres (c). Le résultat est une triangulation moins dense (d). Notez que certaines étapes de la génération de la triangulation n'ont pas été utilisées pour produire ces figures afin de les garder simples et claires. Notamment, le raffinement des extrémités n'a pas été effectué car il rendrait les triangles de l'extrémité de la courbe de la figure (b) impossibles à distinguer.

portement désiré car il faut une triangulation dense pour correctement représenter ce détail, en pratique ces situations apparaissent généralement lorsque l'utilisateur souhaite faire s'arrêter une courbe *exactement* sur une autre. Ces situations peuvent être détectées automatiquement durant le prétraitement de façon à fusionner les extrémités avec les courbes à proximité, comme illustré figure 4.10, ce qui simplifie grandement la triangulation tout en évitant toute fuite de couleur. Cependant, un algorithme ne peut pas déterminer avec certitude si l'utilisateur désire effectuer cette aimantation ; nous considérons donc que le choix d'effectuer ou non l'aimantation doit être reporté au niveau de l'interface utilisateur.

Critère de raffinement. Le raffinement des courbes a plusieurs objectifs :

1. faire en sorte que les segments du PSLG approchent suffisamment bien les courbes pour limiter l'erreur dûe au fait que le solveur FEM ne prend pas en compte les arêtes courbes,
2. s'assurer que les dégradés de couleur définis sur les courbes soient représentés fidèlement, et
3. raffiner suffisamment les courbes pour que l'étape de raffinement de Delaunay ne puisse pas engendrer de repliements.

Le premier objectif est purement géométrique et revient à raffiner les courbes jusqu'à ce que les sous-courbes soient quasiment linéaires.

Le deuxième objectif doit prendre en compte la représentation des dégradés : nous utilisons des dégradés linéaires définis dans la paramétrisation naturelle de courbes de Bézier cubiques que nous voulons approcher à l'aide de patchs cubiques.

Finalement, le dernier objectif vient du fait que le raffinement de Delaunay ne prend pas en compte les arêtes courbes pour des raisons de performances. Il peut donc arriver qu'il insère des sommets à proximité d'un segment du PSLG qui, après raffinement ou lors du rendu avec des arêtes courbes, se retrouve du mauvais côté de la courbe. L'algorithme de triangulation de Delaunay de Shewchuk [2000] dit que lorsqu'un sommet candidat pour être inséré dans la triangulation est inclus dans le plus petit cercle englobant une contrainte, celui-ci est rejeté, et la contrainte est raffinée à la place. Nous verrons plus tard que notre version du raffinement prend soin de subdiviser les contraintes correctement lorsque cela est nécessaire. Tout risque de repliement peut donc être évité en s'assurant que les arrêtes courbes soient intégralement inclues dans le cercle de diamètre $\mathbf{p}_1\mathbf{p}_2$.

Nous combinons ces différents objectifs en un seul critère. Soit une courbe de Bézier $\mathcal{B}$ cubique constituée des points de contrôle $\mathbf{p}_1, \mathbf{h}_1, \mathbf{h}_2, \mathbf{p}_2$. Sauf singularité, en raffinant suffisamment la courbe $\mathcal{B}$, nous convergeons vers une configuration linéaire de paramétrisation linéaire telle que $\mathbf{h}_i = \mathbf{m}_i$ où $\mathbf{m}_i =$

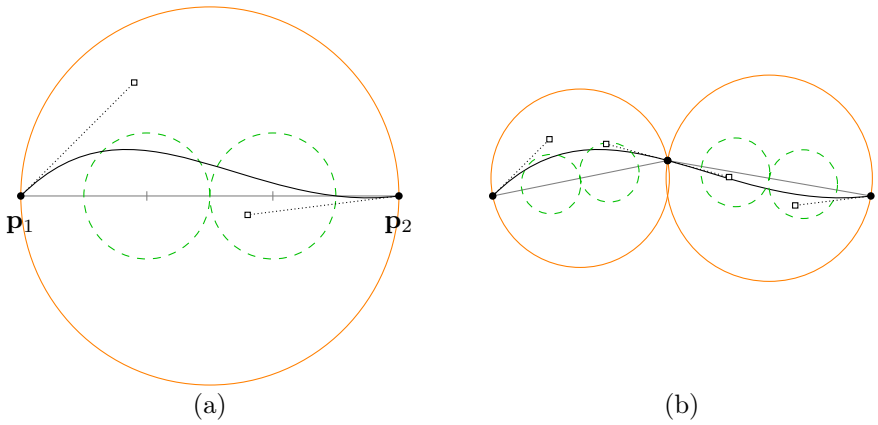


FIGURE 4.11 – **Critère de raffinement des courbes.** Pour éviter les repliements, les points de contrôle des arêtes courbes doivent être inclus dans le cercle diamétral du segment correspondant. Nous utilisons un critère plus restrictif qui assure en plus que les segments sont assez proches de leur approximation linéaire en raffinant les courbes tant que les poignées ne sont pas incluses dans le cercle vert correspondant. La courbe de la figure (a) ne satisfait pas ce critère et est donc divisée en deux (b).

$\frac{1}{3}(2\mathbf{p}_i+\mathbf{p}_j)$, $i \neq j$ et $i, j \in 1, 2$. Notre critère consiste donc à raffiner les courbes tant que $\|\mathbf{h}_i - \mathbf{m}_i\| > \varepsilon\|\mathbf{p}_i - \mathbf{p}_j\|$. Autrement dit la courbe est subdivisée tant que les sous-courbes ne sont pas suffisamment proches d'une configuration linéaire, comme illustré figure 4.11. L'objectif 3 est rempli en s'assurant que $\varepsilon \leq \frac{1}{3}$, car cela implique que les poignées $\mathbf{h}_1$ et $\mathbf{h}_2$ se trouvent dans le cercle diamétral. Cependant, pour éviter que les zones où sont autorisées les poignées se superposent, on préfère choisir $\varepsilon \leq \frac{1}{6}$. Le paramètre ε peut alors être choisi librement sous cette borne de façon à contrôler la densité de la triangulation à proximité des courbes.

Intersection des arêtes du PSLG. Des intersections peuvent être introduites lors de la discrétisation des courbes, même si celles-ci ne sont pas en intersection à l'origine. En pratique, ces cas sont extrêmement rares car cela nécessite à la fois des courbes très proches et de fortes courbures. Le problème peut toujours se corriger en détectant ces intersections et en raffinant les courbes impliquées. La construction de la triangulation de Delaunay contrainte détecte ces intersections, ce qui nous permet de nous passer d'une structure de partitionnement de l'espace supplémentaire pour réaliser cette opération.

4.4.2 Triangulation de Delaunay

Avant d'effectuer le raffinement de Delaunay, le PSLG doit être triangulé à l'aide d'une triangulation de Delaunay contrainte. Cette étape génère des triangles reliant les sommets du PSLG sans en insérer de nouveaux. Il est alors possible de parcourir les triangles autour d'un sommet du PSLG pour déterminer quelle est la contrainte la plus proche.

Nous utilisons cette information de voisinage pour insérer des *sommets de raffinement* autour des extrémités des courbes et des autres sommets singuliers, qui sont souvent sous-échantillonnés sans cela (voir figure 4.12.a). Les sommets de raffinement sont insérés uniformément autour des points singuliers de façon à ce que l'angle entre deux sommets successifs soit inférieur ou égal à $\pi/3$, comme illustré figure 4.12.b. La distance d'insertion de ces sommets doit évidemment dépendre du voisinage du point considéré. Nous avons choisi d'insérer les sommets de raffinement à une distance $l = d_{\min}/3$, où $d_{\min}$ est la distance à la contrainte la plus proche. Ainsi, lorsque deux singularités sont proches et reliées par une arête après triangulation, cette arête est approximativement divisée en trois de façon uniforme.

Le même traitement est effectué autour des contraintes ponctuelles pour les mêmes raisons. Cependant, pour obtenir de bons résultats, la distance l doit être plus petite. Cela est dû au fait qu'une contrainte ponctuelle peut ne pas avoir de contraintes à proximité, ce qui peut conduire à une triangulation très éparse, incapable de correctement représenter les dégradés complexes apparaissant autour de ces sommets. Choisir $l = d_{\min}/6$ produit généralement des résultats satisfaisants.

4.4.3 Raffinement de Delaunay

Le raffinement de Delaunay est l'étape finale de la construction de la triangulation. Nous utilisons un critère de raffinement personnalisé pour éviter que deux contraintes ne soient reliées par une arête et proposons de modifier légèrement l'algorithme pour gérer les arêtes courbes.

Critère de raffinement personnalisé Lorsque deux courbes sont à proximité l'une de l'autre, le raffinement de Delaunay produit souvent des triangles les reliant directement. Or, les éléments quadratiques que nous utilisons ne sont pas capables de représenter les inflexions qui apparaissent lorsque des contraintes de valeur et de gradient sont utilisées simultanément, comme illustré figure 4.13.b. Pour pouvoir représenter ces situations correctement, nous avons modifié le critère de raffinement de façon à assigner un très mauvais score aux triangles possédant une telle arête. Ainsi, après raffinement, il est impossible que deux contraintes soient reliées par une arête. Le résultat de cette procédure est illustré figure 4.13.d.

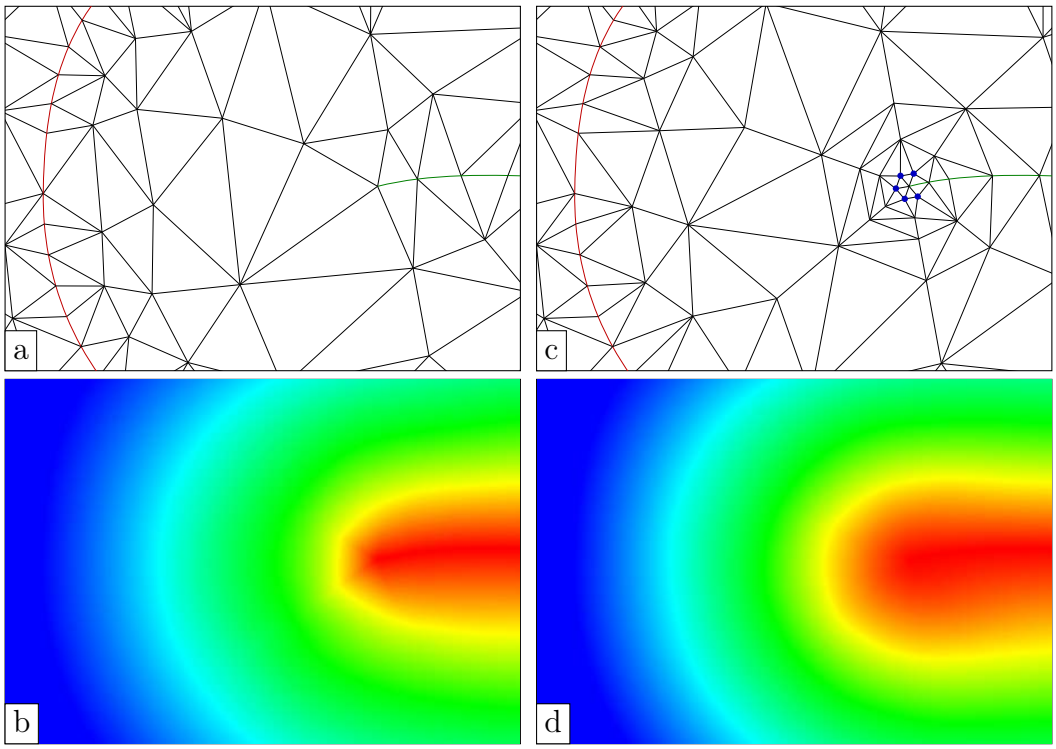


FIGURE 4.12 – **Raffinement des extrémités.** Les figures de gauche ont été réalisées sans raffiner les extrémités. En ajoutant quelques sommets autour des extrémités, ici les sommets bleus, on force la triangulation à produire un maillage plus dense qui permet de représenter ces zones plus fidèlement (à droite). Les images du bas ont été crées en noir et blanc, puis colorisées pour mettre en valeur les différences.

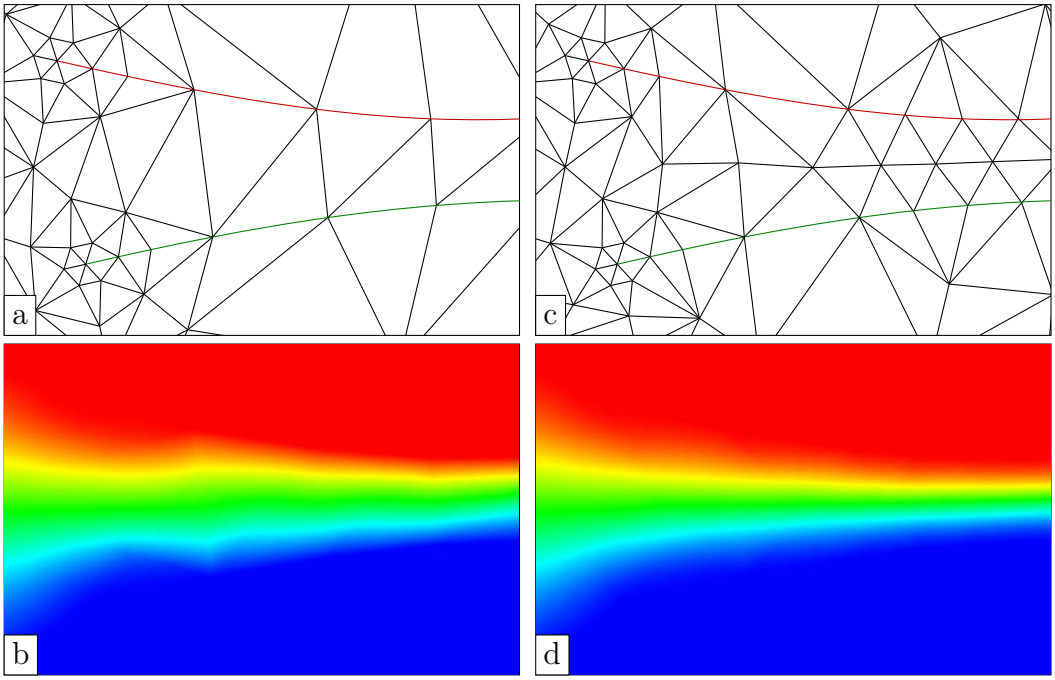


FIGURE 4.13 – **Critère de raffinement de la triangulation.** Avec le critère de raffinement par défaut, la triangulation peut contenir des arêtes reliant directement deux contraintes, ce qui ne permet pas de représenter correctement les inflexions (à gauche). Notre critère raffine les triangles reliant des contraintes en priorité, ce qui produit un bien meilleur résultat (à droite). Les images du bas ont été crées en noir et blanc, puis colorisées pour mettre en valeur les différences.

Raffinement des arêtes du PSLG L'algorithme de raffinement de Delaunay de Shewchuk [2000] ne prend pas en compte les arêtes courbes, donc lorsqu'un sommet est inséré sur une arête du PSLG, celui-ci ne se trouve pas sur la courbe. Il est possible de déplacer de tels sommets *à posteriori*, mais cela peut engendrer des repliements. Nous avons donc modifié l'algorithme de raffinement pour placer les sommets issus du raffinement d'une arête courbe sur celle-ci.

4.5 Solveur FEM

La *méthode des éléments finis* (FEM pour *Finite Element Method*) définit une classe de techniques permettant de résoudre numériquement un système d'équations aux dérivées partielles. Contrairement à la méthode des différences finies où la solution est discrétisée en un ensemble de valeurs ponctuelles, ici la solution est discrétisée en ensemble fini et continu de fonctions de base que nous appelons éléments [Zienkiewicz *et al.*, 2005]. Cette méthode permet de gérer des domaines arbitraires tout en offrant la possibilité d'adapter la précision en fonction du besoin. Elle repose sur l'idée qu'un problème complexe peut être appréhendé en le découpant en plusieurs *éléments* plus simples. Dans notre cas, les éléments sont définis sur les triangles de notre représentation intermédiaire, et la FEM nous permet de résoudre directement et efficacement le problème de diffusion sur cette représentation.

Comme nous l'avons remarqué section 3.3, des problèmes similaires ont déjà été étudiés par la communauté graphique [Jacobson *et al.*, 2010]. Cependant, ces travaux sont généralement développés pour générer des maillages linéaires, alors que notre application bénéficie grandement de l'utilisation de patchs quadratiques. Par ailleurs, l'utilisation d'éléments de plus haut degré permet souvent à la FEM de converger plus rapidement [Zienkiewicz *et al.*, 2005]. Cela est vérifié dans nos expérimentations : pour les dégradés de couleur, les patchs quadratiques produisent de meilleurs résultats que deux fois plus d'éléments triangulaires pour un temps de calcul similaire (figure 4.7), et la différence de qualité est amplifiée lorsque les courbes sont déplacées ou animées.

4.5.1 Formulation variationelle et forme faible

Les FEM ne résolvent jamais directement la formulation *forte* de l'équation différentielle. Une approche relativement standard dans la littérature consiste à projeter l'équation sur un ensemble approprié de *fonctions de test* t_j de façon à réduire les besoins en continuité et donc le degré des éléments, tout en améliorant la stabilité numérique du schéma [Zienkiewicz *et al.*, 2005]. Dans

notre cas de la diffusion Laplacienne, l'équation 3.1 devient :

$$\forall j,\ \int_\Omega \nabla u \cdot \nabla t_j \,\mathrm{d}x\,\mathrm{d}y = 0\,. \tag{4.3}$$

De la même manière, l'équation 3.6 de la diffusion bi-Laplacienne mène à la *forme faible* suivante [Chen, 2005] :

$$\forall j,\ \int_\Omega \Delta u \cdot \Delta t_j \,\mathrm{d}x\,\mathrm{d}y = 0\,. \tag{4.4}$$

4.5.2 Choix des éléments

La diffusion Laplacienne peut s'effectuer avec des éléments polynomiaux d'ordre un (linéaire) ou plus, et ne nécessite qu'une continuité C^0 entre les éléments. Il est donc possible de la mettre en œuvre directement en utilisant des éléments similaires aux patchs Lagrangiens utilisés dans notre représentation intermédiaire. Utiliser des éléments linéaires, quadratiques ou de plus haut degré se fait alors de façon relativement directe sans difficulté particulière.

Le cas de la diffusion bi-Laplacienne est plus complexe car elle nécessite une continuité C^1 entre les éléments, ce qui nécessite des éléments polynomiaux d'ordre cinq tel que le triangle d'Argyris [Chen, 2005]. Un alternative consiste à utiliser des éléments *non-conformes* de plus faible degré. Afin d'obtenir un problème bien posé pour de tels éléments, la forme bilinéaire précédente (équation 4.4) doit être légèrement modifiée. Comme expliqué dans un article de Lascaux et Lesaint [1975], la forme faible de l'équation bi-Laplacienne pour des éléments non-conformes devient :

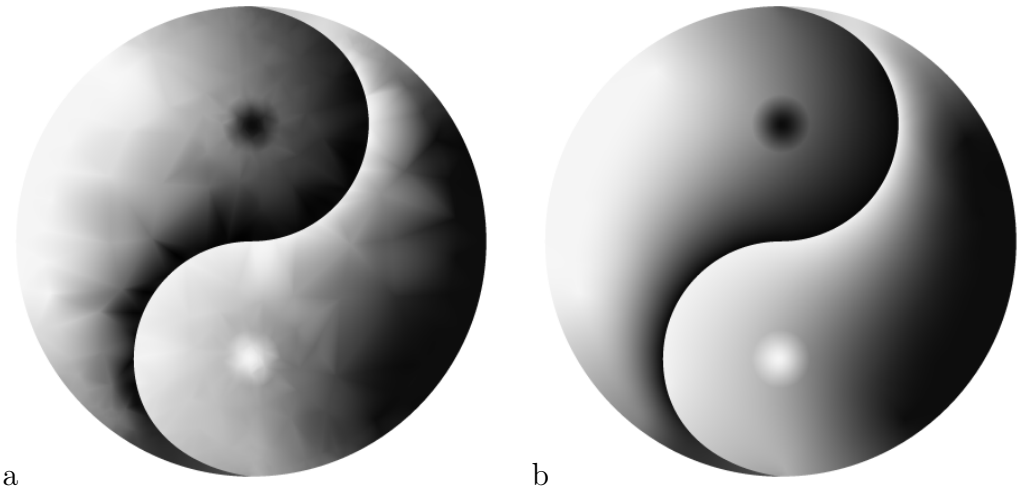
$$\forall j,\ \int_\Omega \Delta u \cdot \Delta t_j - \sigma \left(\frac{\partial^2 u}{\partial x^2}\frac{\partial^2 t_j}{\partial y^2} - 2\frac{\partial^2 u}{\partial x \partial y}\frac{\partial^2 t_j}{\partial x \partial y} + \frac{\partial^2 u}{\partial y^2}\frac{\partial^2 t_j}{\partial x^2} \right) \mathrm{d}x\,\mathrm{d}y = 0\,, \tag{4.5}$$

où σ est le coefficient de Poisson contrôlant l'influence du terme de régulation [Ciarlet, 1978]. En pratique, σ doit être choisi dans l'intervalle $[\frac{1}{2}, 1]$, auquel cas il n'a aucune influence sur le résultat. L'élément de Morley [1971], illustré figure 4.15.a, est l'élément non-conforme le plus simple, et le plus connu qui permet de résoudre l'équation 4.5. Il utilise des fonctions de base quadratique à six degrés de liberté : les valeurs à chaque sommet ainsi que les dérivées dans le sens normal au milieu des arêtes, ce qui donne les fonctions de base suivantes :

$$\begin{aligned} M_4 &= \frac{-2\Delta}{\|\mathbf{v}_i\|} L_1(1 - L_1) \\ M_1 &= L_1^2 + \alpha_{i,2} M_5 + \alpha_{i,3} M_6 \end{aligned} \tag{4.6}$$

avec

$$\alpha_{i,j} = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{2\Delta \|\mathbf{v}_j\|}$$



a b

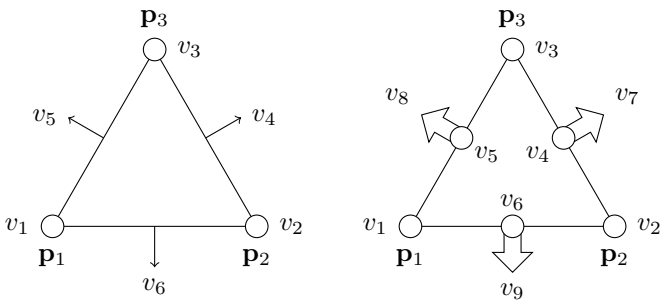
FIGURE 4.14 – **Éléments pour les patchs quadratiques.** Convertir les éléments de Morley [1971] en patchs quadratiques ne produit pas un résultat satisfaisant (a). En comparaison, les éléments FV permettent de donner directement les valeurs correspondant aux patchs quadratiques (b).

où $\mathbf{v}_1 = \mathbf{p}_3 - \mathbf{p}_2$, Δ est l'aire de l'élément, et M_2, M_3, M_5 et M_6 sont obtenus par permutations cycliques.

En pratique, les éléments de Morley ne sont même pas C^0 continus le long des arêtes et ne sont donc pas utilisables pour le rendu. Il est possible de ne considérer que les nœuds de valeur aux sommets et d'effectuer le rendu avec des patchs linéaires, mais comme nous l'avons déjà expliqué, cela ne nous permet pas d'atteindre la qualité désirée.

Nous avons envisagé la possibilité d'évaluer la valeur des éléments de Morley au centre des arêtes de chaque côté et de prendre leur moyenne pour initialiser les nœuds $v_i, i = 4, 5, 6$ des patchs Lagrangiens quadratiques. Cependant, comme illustré figure 4.14.a, cette stratégie naïve ne produit pas un résultat statisfaisant.

Afin de permettre l'utilisation de patches quadratiques pour le rendu, nous proposons d'utiliser les éléments *non-conformes* cubiques de Fraeijs de Veubeke [1974] (FV) illustrés figure 4.15.b. Ces éléments peuvent être vus comme une variante des éléments de Morley d'ordre supérieur : ils sont contrôlés par six valeurs nodales dont trois sur les sommets et trois au milieu des arêtes, plus les *valeurs moyennes* des dérivées normales le long de chaque arête, que nous appellerons *nœuds de gradient moyen*. Les fonctions de base F_i associées sont



(a) Morley (b) FV

FIGURE 4.15 – **Éléments *non-conformes* pour l'équation bi-Laplacienne.** L'élément quadratique de Morley [1971] (a) possède trois nœuds de valeur associés aux sommets et trois autres indiquant la magnitude du gradient dans la direction normale au centre des arêtes. L'élément de Fraeijs de Veubeke [1974] (FV) (b) possède en plus trois nœuds de valeur associés au centre des arêtes, et utilise la magnitude du gradient *moyenne* sur les arêtes.

données par les formules suivantes :

$$\begin{aligned} F_1 &= L_1(L_1 - 1/2)(L_1 + 1) + 3L_1L_2L_3 \\ &\quad - d_2L_2(2L_2 - 1)(L_2 - 1) + d_3L_3(2L_3 - 1)(L_3 - 1) \\ F_4 &= 4L_1(1 - L_1)(1 - 2L_1) + 4L_2L_3 - 12L_1L_2L_3 \\ F_7 &= -\frac{2\Delta}{\|\mathbf{v}_1\|}L_1(2L_1 - 1)(L_1 - 1) \end{aligned} \tag{4.7}$$

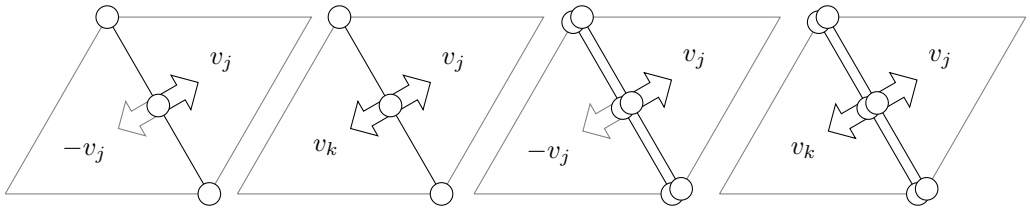
avec

$$d_1 = \frac{\mathbf{v}_1^T\mathbf{v}_3}{2\|\mathbf{v}_1\|^2}$$

où les valeurs d_2, d_3 et les autres fonctions de base sont obtenues par permutations cycliques. Comme avec les éléments de Morley, la jonction des éléments FV n'est pas C^0. En pratique, il suffit d'ignorer les nœuds de gradient moyen pour obtenir des patchs Lagrangiens quadratiques C^0, ce qui correspond exactement à ce nous cherchions à obtenir.

Un avantage supplémentaire de ces éléments est qu'ils fournissent un contrôle direct sur la longueur du gradient dans le sens normal à une arête. Comme nous allons le voir dans la suite, cela permet d'assigner les contraintes de gradient relativement facilement.

Dans ces conditions, nous recherchons une approximation de u de la forme $\sum_j w_j F_j$ où F_j sont les fonctions de base des éléments FV, et w_j sont les nœuds de valeur recherchés. Si m est le nombre total de nœuds dans notre représentation intermédiaire, nous avons $w_j = v_j$ pour $j \leq m$ (rappelons que les v_j sont les valeurs nodales des patchs quadratiques de la représentation intermédiaire.). Comme cela est souvent effectué avec les FEM, nous prenons pour les



(a) $v = \cdot/g = \cdot$ (b) $v = \cdot/g = \cdot|\cdot$ (c) $v = \cdot|\cdot/g = \cdot$ (d) $v = \cdot|\cdot/g = \cdot|\cdot$

FIGURE 4.16 – **Gestion des contraintes.** Les différents types de contraintes sont obtenus en partageant ou en dupliquant les nœuds de valeur ou de gradient.

fonctions de test t_j le même ensemble de fonctions de base F_j. L'équation 4.5 devient alors un problème linéaire creux de la forme $A\mathbf{w} = 0$, où la matrice A est appelée la matrice de *rigidité* dont les coefficients $A_{i,j}$ sont donnés par :

$$A_{i,j} = \int \Delta F_i \cdot \Delta F_j - \sigma \left(\frac{\partial^2 F_i}{\partial x^2} \frac{\partial^2 F_j}{\partial y^2} - 2 \frac{\partial^2 F_i}{\partial x \partial y} \frac{\partial^2 F_j}{\partial x \partial y} + \frac{\partial^2 F_i}{\partial y^2} \frac{\partial^2 F_j}{\partial x^2} \right) . \quad (4.8)$$

4.5.3 Gestion des contraintes

Les éléments FV nous permettent de contraindre les valeurs et les gradients directement. Comme illustré figure 4.16, lorsque les valeurs ou les gradients ne sont pas contraints à être égaux de chaque côté d'une courbe, les nœuds correspondants sont dédoublés. Un nœud attaché à une contrainte spécifiant une valeur donnée prend directement sa valeur de la contrainte et est éliminé de la liste des inconnues. De même, une contrainte de gradient associée à une courbe est intégrée sur les arêtes correspondantes pour fixer les nœuds de gradient moyen. Les courbes de contours sont gérées de façon triviale en utilisant une seule valeur nodale, c'est à dire une seule inconnue, pour tous les nœuds de valeur appartenant à la même courbe de contour.

Prendre en compte les gradients spécifiés à un sommet isolé est légèrement plus compliqué puisque les élément FV permettent uniquement de contrôler les gradients situés le long des arêtes. Considérons l'élément FV $\mu(\mathbf{x}) = \sum_{k=1}^{9} F_k(\mathbf{x}) w_k$ illustré figure 4.15.b, et supposons que le sommet $\mathbf{p}_1$ soit une contrainte ponctuelle isolée de gradient $\mathbf{g}$. Soient $\mathbf{e}_2 = \mathbf{p}_2 - \mathbf{p}_1$ et $\mathbf{e}_3 = \mathbf{p}_3 - \mathbf{p}_1$ les deux arêtes incidentes, et $\mu_2(t)$, $\mu_3(t)$ la valeur de l'élément sur ces deux arêtes ; contraindre $\nabla\mu(\mathbf{p}_1) = \mathbf{g}$ est équivalent à contraindre $[\mu_2'(0)\ \mu_3'(0)]^T = \bar{\mathbf{g}}$, où $\bar{\mathbf{g}} = [\mathbf{e}_2\ \mathbf{e}_3]^{-1}\mathbf{g}$ est la projection de $\mathbf{g}$ sur la base engendrée par $\mathbf{e}_2$,$\mathbf{e}_3$. En observant les fonctions de base des éléments FV (équation 4.7), nous pouvons remarquer que le nœud de gradient d'une arête donnée n'influence pas la valeur de l'élément le long de cette arête. Par conséquent, les contraintes de dérivé

dans chaque direction peuvent être forcées indépendamment en contraignant les nœuds de gradient moyen v_8 et v_9. Par exemple, cela conduit pour v_8 à

$$v_8 = \frac{\bar{g}_0 - \sum_{k \neq 8,9} (F_k^2)'(0) v_k}{(F_8^2)'(0)} ,$$

où $F_k^2 = F_k(\mathbf{p}_1 + t\mathbf{e}_2)$ sont les fonctions de base de l'élément FV restreintes à l'arête $\mathbf{e}_2$. Ces contraintes sont introduites dans l'équation 4.8 lors de l'assemblage de la matrice de rigidité, ce qui a pour effet de supprimer deux degrés de liberté, et de créer un membre à droite non nul.

4.5.4 Gestion de la transmission

La transmission est une contrainte asymétrique permettant à la couleur d'un côté de la courbe de se diffuser de l'autre côté tout en bloquant complètement ou partiellement la diffusion dans la direction opposée. De telles courbes n'ont pas de contrainte de valeur et la couleur des deux côtés doit être identique. Nous pouvons observer que l'inconnue w_j d'un nœud donné appartenant à une courbe de transmission possède une influence sur les valeurs nodales des éléments incidents. Étant donné deux poids β_l et β_r, l'idée est de faire en sorte que le nœud j émette $\beta_\mathrm{l} w_j$ aux nœuds à gauche de la courbe et $\beta_\mathrm{r} w_j$ aux nœuds à droite de cette même courbe. En construisant la matrice de rigidité, cela revient à insérer $\beta_\mathrm{l}\mathbf{A}_{i,j}$ au lieu de $\mathbf{A}_{i,j}$ si i est à gauche de la courbe, et $\beta_\mathrm{r}\mathbf{A}_{i,j}$ au lieu de $\mathbf{A}_{i,j}$ si i est à droite, et directement $\mathbf{A}_{i,j}$ autrement (i est sur la courbe). Il faut noter que cela conduit à une matrice de rigidité numériquement asymétrique.

Les poids β_l et β_r sont calculés à partir du facteur de transmission α : lorsque $\alpha = 0$, nous utilisons $\beta_\mathrm{l} = \beta_\mathrm{r} = 1$ ce qui produit le comportement par défaut. Quand $\alpha > 0$, $\beta_\mathrm{r} = 1$ et $\beta_\mathrm{l} = 1 - \alpha$ de sorte que les éléments à droite de la courbe se comportent normalement, mais que ceux à sa gauche soient moins influencés par w_j et donc moins influencés par le côté droit. Au final, avec $\alpha = 1$, $\beta_\mathrm{l} = 0$, les éléments à gauche de la courbe ne dépendent plus des éléments à sa droite, ce qui revient à bloquer la diffusion de droite à gauche. Inversement, lorsque $\alpha < 0$, nous avons $\beta_\mathrm{l} = 1$ et $\beta_\mathrm{r} = 1 + \alpha$.

4.5.5 Gestion des singularités

Comme expliqué en section 4.3, afin de représenter correctement les singularités nous avons introduit des patchs spécifiques. Ces patchs jouent pleinement leur rôle pour l'affichage. En revanche, ils ne peuvent pas être utilisés pour la résolution de l'équation car ils ne sont pas différentiables au point de singularité. Nous contournons ce problème en remplaçant un élément singulier par deux éléments standards superposés connectés aux deux différents nœuds de

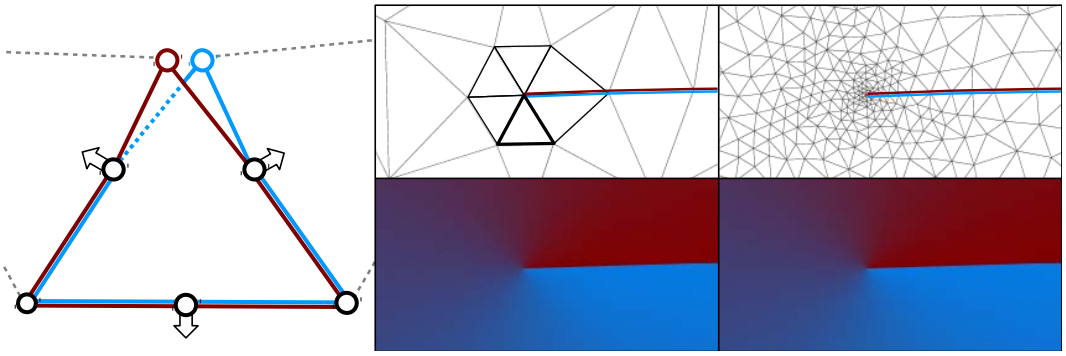


FIGURE 4.17 – **Gestion des éléments singuliers.** Les éléments singuliers sont remplacés par deux éléments standards superposés, connectés à un nœud différent au point singulier (à gauche). Cela donne un résultat indistinguable de celui obtenu en tesselant fortement la triangulation (à droite).

la singularité comme illustré figure 4.17. Bien que nous n'ayons pas mené une étude théorique de l'impact de cette solution sur la convergence de la méthode, cette approche simple fonctionne remarquablement bien en pratique.

4.6 Résultats et discussions

4.6.1 Mise en œuvre et performances

Notre prototype fonctionne entièrement sur le CPU, et n'exploite que très peu le *multithreading*. Il utilise CGAL [CGAL] pour la triangulation et le raffinement de Delaunay, et Eigen [Eigen] pour la résolution directe du système linéaire creux. Comparé aux méthodes itératives, un solveur direct s'appuie sur une factorisation de la matrice de rigidité, ce qui permet de résoudre le problème de diffusion très efficacement pour plusieurs jeux de contraintes (en l'occurrence les différents canaux de l'image, soit rouge, vert, bleu et alpha) ; Cela permet de diffuser des textures à la façon de Bowers *et al.* [2011] pour un coût raisonnable par *shader* (voir section 3.2.2). De plus la matrice de rigidité dépend uniquement de la topologie de la triangulation. Ainsi, lorsque seules les valeurs des contraintes sont modifiées, il n'est pas nécessaire de re-factoriser la matrice. En d'autres termes, notre implémentation permet de régler directement un ensemble de contraintes riche (tel que le contrôle de la magnitude des gradients, les courbes de contour ou la transmission asymétrique) directement dans le système d'équations linéaires.

Bien que notre implémentation actuelle exploite très peu le parallélisme, il est intéressant de noter que celui-ci peut intervenir à différents niveaux. Par exemple, plusieurs calques peuvent être calculés en parallèle. Au niveau d'un calque, il est fréquent que l'image puisse être divisée en de nombreuses cellules

	Cr	N	T	Ce	PSLG	Raf.	Mat.	Sol.	Total
Yin-yang (4.8.c)	24	5033	1259	3	3.9	9.6	9.6	32.5	55.6
Arbre (4.20.b)	71	11751	2935	4	5.5	7.9	19.8	22.5	55.7
Parapluie (4.18)	74	24724	6032	9	5.6	24.8	37.3	25.3	93.1
Volcan (4.21)	481	19734	5351	3	34.3	16.1	35.0	49.0	134.5
Fée*	423	60016	14330	58	84.0	55.3	97.9	65.0	302.1
	192	36334	8805	27	24.1	32.5	59.7	41.9	158.1
Neko* (3.12.b)	288	49731	12439	32	64.9	43.6	85.3	42.5	236.3
Elian (4.19)	635	87971	20931	115	177.8	92.7	145.0	78.9	494.4
Rose*	576	114592	28514	29	153.1	108.6	198.3	101.7	561.6
Cendrillon (4.19)	1218	147815	34944	231	547.8	161.9	247.3	133.7	1090.7
	92	17935	4492	10	9.9	14.6	29.3	28.0	81.8
Perroquet*	1724	219773	53028	159	702.6	247.9	382.7	225.1	1558.3

TABLE 4.1 – **Performances du solveur.** Les colonnes du premier bloc correspondent à des statistiques sur l'image : dans l'ordre, le nombre de courbes (Cr) dans le graphe planaire en entrée, le nombre de nœuds (N) et de triangles (T) dans la représentation intermédiaire et le nombre de cellules (Ce) indépendantes. Les colonnes du deuxième bloc sont les durées des différentes étapes en millisecondes : construction du PSLG (toutes les étapes jusqu'au raffinement), Raffinement de Delaunay, construction de la matrice et solveur linéaire. Les illustrations avec plusieurs calques sont présentées sur plusieurs lignes. * Indique les illustrations de chapitre.

indépendantes (voire table 4.1). Par exemple, 32 zones peuvent être extraites pour le modèle de la figure 3.12.b. Actuellement, ces cellules ne sont détectées qu'une fois la matrice de rigidité assemblée, ce qui nous permet de diviser le système d'équation en plusieurs systèmes indépendants plus petits qui sont résolus simultanément. Idéalement, cette détection devrait être réalisée le plus en amont possible, c'est à dire à partir du simple graphe planaire en entrée. Cela permettrait de paralléliser efficacement la construction de la représentation intermédiaire ainsi que l'assemblage du système linéaire. Finalement, chacune des tâches (triangulation, raffinement, assemblage, solveur direct) pourrait-être elle-même parallélisée en utilisant des algorithmes adaptés.

Des optimisations spécifiques pour accélérer le solveur durant l'édition sont envisageables. Lors de l'édition de la géométrie d'une courbe, il est possible de ne mettre à jour que les cellules modifiées, ce qui peut réduire drastiquement la zone à trianguler et la taille du système à résoudre. Il est possible de dégrader la qualité de la triangulation durant l'édition pour un retour plus rapide. En l'occurrence, nous pouvons exploiter le fait que le point de vue est généralement fixe pendant l'édition pour fortement dégrader la qualité de la triangulation hors de l'écran tout en conservant la qualité par défaut dans les parties visibles. Une fois l'édition terminée, le solveur peut être relancé en haute qualité sur tout le domaine.

Nous avons testé notre prototype sur une machine équipée d'un processeur Intel Core i7-3610QM et obtenons les performances indiqués table 4.1. Dans l'ensemble, le coût des différentes étapes est relativement uniforme. Notons que le solveur linéaire et le raffinement de Delaunay sont effectués par des bibliothèques tiers qui sont relativement bien optimisées. En revanche, ce n'est pas le cas de l'étape de construction du PSLG et de la matrice de rigidité. Les mauvaises performances de la construction du PSLG pour les images complexes s'expliquent par une implémentation naïve de la détection des intersections entre les courbes de Bézier. Une fois optimisée, cette étape ne devrait représenter qu'un faible pourcentage des coûts de calculs globaux. L'assemblage de la matrices de rigidité est dominée par les évaluations analytiques de l'intégrale 4.8 qui actuellement engendre des calculs redondants qui pourraient être pré-calculés. Il est en effet contre nature que la résolution du système soit plus rapide que son assemblage.

Il est difficile de comparer nos performances avec celles des méthodes basées pixel : la complexité de ces solveurs croît avec la résolution en sortie, alors que notre solveur dépend de la complexité en entrée. Nous pensons que des solveurs insensibles à la résolution de sortie sont préférables dans le sens où cela nous permet de conserver la majorité des avantages du dessin vectoriel, comme un faible nombre de primitives.

On pourrait s'inquiéter du fait que pour des images plus complexes, notre solveur finirait par devenir moins performant que les solveurs utilisant des grilles de pixels pour des images de résolution moyenne. Cependant, nous pensons qu'en pratique cela ne constituera pas un problème. Effectivement, la majorité des images que nous présentons sont constituées d'un seul calque très chargé, mais il ne s'agit pas de la méthode la plus intelligente d'utiliser le principe des images de diffusions. Il est en effet souvent préférable d'utiliser de nombreux calques pour structurer la scène et faciliter son édition. Il s'agit d'une pratique communément employée par les artistes, que ce soit pour la création d'images vectorielles ou bitmaps. Dans une telle situation, nous devons calculer la diffusion sur plusieurs images simples, ce que notre solveur peut gérer facilement grâce à son indépendance vis à vis de la résolution en sortie et qui est facilement parallélisable. De plus, nous pouvons arguer que le nombre de cellules isolées croît logiquement avec le nombre de primitives en entrée.

4.6.2 Rendu

Rappelons que notre solveur doit être entièrement mis à jour à chaque fois qu'un point de contrôle est modifié, et seulement partiellement lorsque les contraintes sont changées. Cela fait partie intégrante du processus de dessin et d'édition. C'est à différencier du rendu, qui consiste principalement à éva-

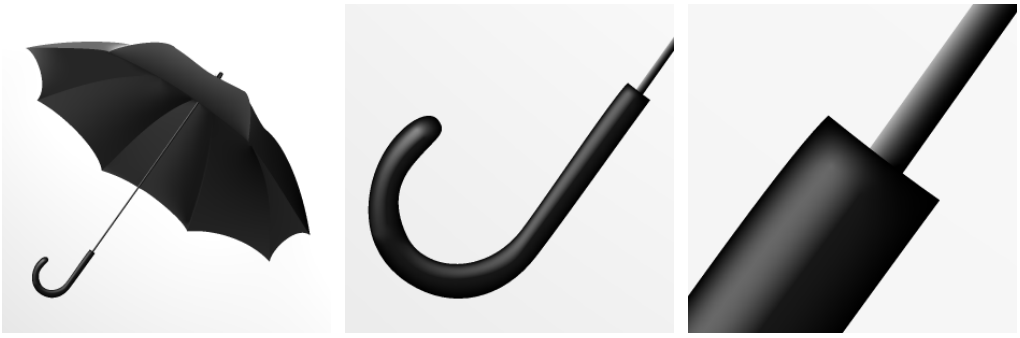


FIGURE 4.18 – **Zooms.** Un parapluie à différents niveaux de zoom. La représentation intermédiaire n'a été calculée qu'une fois pour les trois images.

luer notre représentation intermédiaire en des points spécifiques. Comme notre solution est donnée en forme close, cela simplifie et améliore grandement les applications faisant usage de dégradés vectoriels comme nous allons le détailler dans cette section.

Images en couleur

Le rendu de notre représentation intermédiaire revient à tracer des triangles en évaluant une fonction quadratique pour chaque pixel. Il peut donc facilement être effectué, que ce soit sur le CPU ou sur le GPU. Bien que nous ayons supposé que les arêtes étaient droites pour la résolution de la diffusion, au moment du rendu les arêtes liées à une courbe doivent être lissées. Pour cela, la méthode standard consiste à raffiner les courbes de manière adaptative en fonction du zoom et des éventuelles autres déformations. Dans notre implémentation, les triangles impliqués sont raffinés sur le CPU puis rasterisés à l'aide d'OpenGL et de *shaders* pour évaluer les patchs quadratiques. La tesselation matérielle disponible sur les GPU les plus récents peut bien sur être utilisée à la place du raffinement logiciel pour de meilleures performances. Par ailleurs, notre représentation intermédiaire étant un ensemble de patchs quadratiques, il est possible de la stocker directement dans un fichier au format PostScript ou PDF, ce qui permet de l'imprimer directement.

Plusieurs images générées avec notre méthode sont disséminées dans ce document. Les illustrations au début de l'introduction, des chapitres 3 et 4 ainsi que de la conclusion ont été réalisées avec notre prototype. La figure 4.18 illustre le fait qu'une même représentation intermédiaire permet de visualiser une image de diffusion à différents niveau de zoom. Par ailleurs, la figure 4.19 présente quelques résultats supplémentaires [2].

2. Ces images ont été dessinées à partir de modèles disponibles sur `http://loucie.artblog.fr/` et `http://loucie.deviantart.com/` avec autorisation.

FIGURE 4.19 – **Résultats supplémentaires.**

Notre approche autorise une grande variété de déformations au moment du rendu : translations, rotations, mises à l'échelle, mais aussi des déformations en perspectives comme illustré figure 4.20. Il suffit pour cela de déformer la représentation intermédiaire sans relancer le solveur. Le faible coût du rendu de la représentation intermédiaire permet de facilement mélanger plusieurs calques, comme dans la figure 4.20.a (voir aussi la figure introductive du chapitre 4). De même, l'instanciation d'une image est particulièrement facile comme illustré 4.20.b : le solveur n'est exécuté qu'une seule fois, mais la représentation intermédiaire peut-être utilisée plusieurs fois, avec des transformations différentes.

Il est également possible d'effectuer des transformations qui ne conservent pas les lignes droites, auquel cas la représentation intermédiaire peut nécessiter un raffinement pour éviter les artefacts. Il est à noter que dans le cas de transformations non rigides, déformer la représentation intermédiaire ne donne pas le même résultat que déformer les courbes puis effectuer la diffusion. Or, dans bien des cas, notamment pour l'animation, les déformations sont vues comme un post-traitement. Il s'agit encore une fois d'une situation où une représentation intermédiaire vectorielle se révèle être indispensable. Par exemple, on peut imaginer animer un personnage uniquement en déformant la représentation intermédiaire, auquel cas la diffusion n'a besoin d'être effectuée qu'une

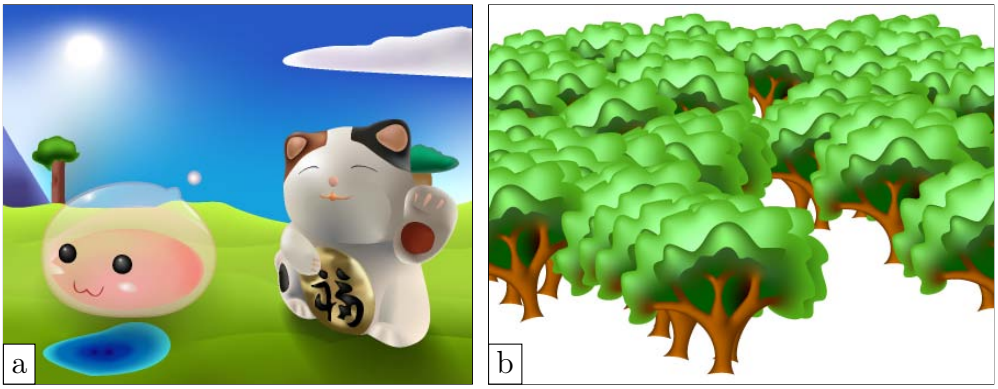


FIGURE 4.20 – **Calques, déformations et instanciation.** L'image (a) est constituée de plusieurs calques et fait usage de transparence et de déformation en perspective. L'image (b) est constituée de 64 fois le même arbre avec différentes transformations afin de produire une forêt. La représentation intermédiaire étant particulièrement efficace à rasteriser, le point de vue peut être modifié en temps réel.

seule fois.

La représentation intermédiaire s'avère particulièrement utile pour l'impression puisque des images de résolution arbitraire peuvent être utilisées en sortie, pour du copié-collé où chaque copie est traitée comme une instance (éventuellement déformée) sans avoir à relancer le solveur, et pour le stockage et la distribution.

Images 2.5D

Si spécifier des contraintes de gradient arbitraires pour des images couleur n'a pas réellement de sens, cela s'avère particulièrement utile pour travailler avec des données 2.5D telles que des champs de hauteur. Cela est illustré figure 4.21. Cependant, nos patchs ne sont que C^0 à leurs jonctions. Bien que cela s'avère suffisant pour le rendu d'images couleur, l'éclairage sur la surface d'un champ de hauteur peut révéler l'absence de continuité C^1. Il est possible d'améliorer le rendu en appliquant, par exemple, une interpolation quadratique des normales [Vlachos *et al.*, 2001], cependant, une méthode plus simple et logique (mais plus coûteuse) consiste à utiliser des patchs linéaires tout en augmentant la résolution du maillage.

Notre méthode apporte plusieurs avantages comparée aux méthodes précédentes pour la création de champs de hauteur [Hnaidi *et al.*, 2010]. D'abord, elle fournit un ensemble de contrôles artistiques plus riche grâce à la diffusion bi-Laplacienne et aux extensions que nous gérons. Ensuite, la sortie de notre solveur est une structure directement utilisable pour le rendu de champs de hauteur alors que les autres méthodes nécessitent soit un algorithme de rendu

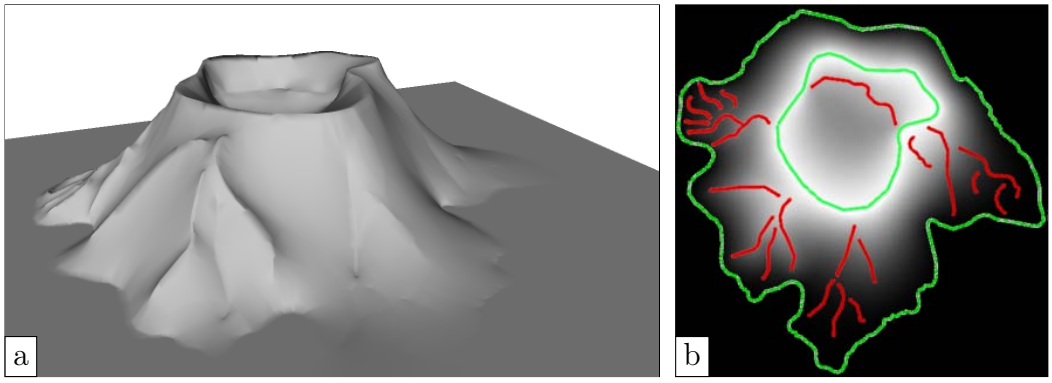


FIGURE 4.21 – **Édition de *height map*.** Le terrain (a) est généré à partir des courbes (b). Les courbes vertes sont des contraintes de valeur alors que les courbes rouges sont des contraintes de gradient pures.

spécifique, soit une conversion en maillage. En effet, notre maillage intermédiaire n'a qu'à être "soulevé" en fonction des hauteurs calculées pour pouvoir être chargé dans un moteur de rendu. Pour la création de champs de hauteur, notre approche semble comparable à la méthode de Jacobson *et al.* [2010] utilisant des éléments mixtes.

Il est aussi possible de générer des cartes de normales. Deux approches sont envisageables : la première consiste à diffuser un champ de hauteur puis à calculer ces normales, mais là encore l'utilisation de patchs C^0 à leurs jonctions pose problème. Une approche plus simple consiste à diffuser directement des normales à la place des couleurs, comme Winnemöller *et al.* [2009]. Plus précisément, nous ne diffusons que les coordonnés x et y des normales et calculons la coordonnée z à l'aide de la formule $z = \sqrt{1 - x^2 - y^2}$, ce qui nous permet d'éviter les problèmes lorsque $x = y = z = 0$. Cependant, pour que cela fonctionne, il faut utiliser la diffusion Laplacienne afin d'éviter les problèmes de saturation. Un exemple de cette méthode est présenté figure 4.23.a.

4.6.3 Application à la 3D

La représentation intermédiaire peut-être utilisée pour ajouter des détails sur des surfaces. Il est possible de l'utiliser pour faire du plaquage de texture. Bien que des méthodes aient déjà été proposées pour cela, celles-ci souffrent soit de problèmes lorsque l'image subit de fortes déformations [Jeschke *et al.*, 2009b], soit elles sont relativement lourdes en calcul tout en étant approximatives [Sun *et al.*, 2012]. Ensuite, elle peut servir de carte de déplacements. Dans les deux cas, elle peut être utilisée de différentes façons offrant différents compromis.

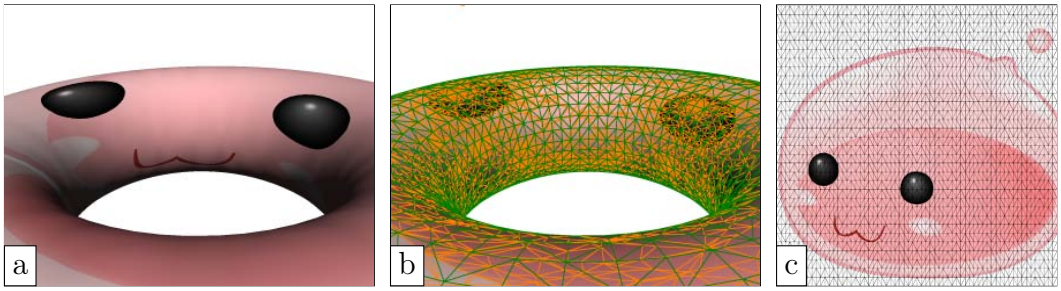


FIGURE 4.22 – **Plaquage de texture vectorielles.** La texture sur le tore (a) est obtenue en subdivisant le maillage pour y inclure les triangles de la représentation intermédiaire (b). Les arêtes vertes sont celles du maillage d'origine et les oranges sont celles ajoutées par ce procédé. La figure (c) correspond à l'espace paramétrique.

Triangulations compatibles. La méthode que nous avons implémentée consiste à rendre les deux triangulations – la représentation intermédiaire et le maillage dans son espace paramétrique – *compatibles*. Cela se fait en calculant les intersections entre elles de façon à découper les triangles du modèle 3D, ce que nous avons implémenté avec CGAL [CGAL]. La figure 4.22 illustre cette méthode dans le cadre du plaquage de texture. Elle produit un ensemble de patchs quadratiques en 3D, utilisables directement avec les GPU modernes à l'aide un *shader* adapté. Cette approche permet de facilement représenter les discontinuités. Cependant, le calcul de l'intersection entre les triangulations est une opération coûteuse et délicate à réaliser, et il est nécessaire de l'effectuer pour chaque modèle et à chaque changement dans la paramétrisation. De plus, cette solution n'est pas adaptée pour faire de l'accès aléatoire, par exemple, pour une carte de réflexion.

Pour une carte de déplacement, les sommets peuvent simplement être déplacés le long des normales et les trous engendrés par les discontinuités bouchés. Il faut tout de même prendre quelques précautions : d'abord, lorsque les faces du maillage sont découpées pour rendre les triangulations compatibles, il faut idéalement projeter les sommets sur une surface lisse. Ensuite, il faut raffiner les arêtes courbes et prendre soin de boucher les trous engendrés par les discontinuités. La principale limitation de cette approche est la même que pour les champs de hauteur : raffiner les patchs quadratique ne produit pas un résultat visuellement plaisant à cause de l'absence de continuité C^1 entre les patchs. Il est possible d'utiliser des patchs linéaires, mais cela rend l'intersection des triangulations plus coûteux.

Dans l'ensemble, cette approche fonctionne bien pour faire du rendu temps-réel sur carte graphique, mais manque de *scalabilité*. En particulier, il est difficile d'employer plusieurs couches car il faudrait rendre toutes les triangulations compatibles ce qui produit typiquement de nombreux petits triangles inutiles

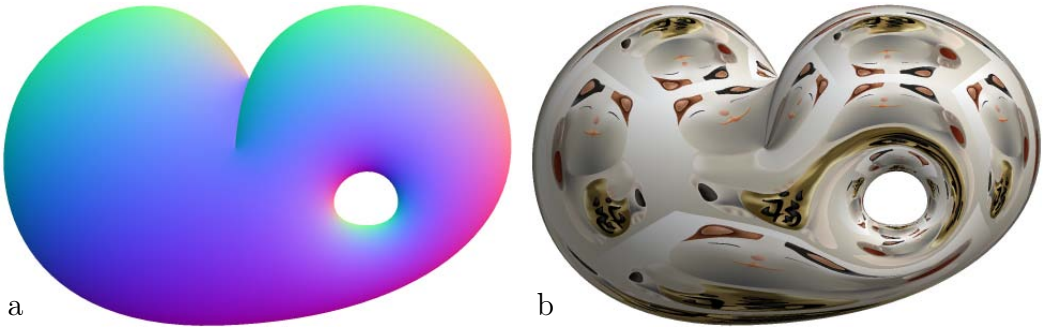


FIGURE 4.23 – **Carte de normales et de réflexions.** La figure (a) est une carte de normales crée avec notre méthode. Nous l'utilisons pour réfléchir une autre image de diffusion. Il est alors possible de zoomer à volonté sur l'image en conservant des discontinuités nettes.

pour la qualité finale. Une solution pourrait être de calculer une seule triangulation pour tous les calques (contenant donc toutes les arêtes nécessaires pour représenter les contraintes), mais cela produirait une triangulation excessivement dense qui ralentirait le solveur.

Accès aléatoire Dans le cas où une même texture doit être plaquée sur différents objets ou lorsque l'image de diffusion subit de fortes déformations comme dans le cas de réflexions, l'approche proposée précédemment ne fonctionne pas. On peut alors faire de l'accès aléatoire dans la représentation intermédiaire. Le problème se résume alors à trouver l'unique triangle contenant le point d'évaluation. Nous avons implémenté cela sur CPU pour faire du rendu de cartes d'environnement vectorielles, comme illustré figure 4.23. Une implémentation GPU est envisageable : le problème est similaire au rendu d'image vectorielles sur GPU pour lequel plusieurs approches ont été proposées [Bruno et Cavin, 2005; Parilov et Zorin, 2008; Nehab et Hoppe, 2008].

4.6.4 Limitations

Scintillements. La triangulation de la représentation intermédiaire est recalculée à partir de zéro à chaque fois qu'un point est déplacé. Puisque la triangulation est effectuée indépendamment de la configuration précédente (ce qui est souhaitable), celle-ci peut varier de manière significative et non continue lors de l'édition d'une courbe. Dans ces conditions, il est nécessaire que le résultat obtenu soit suffisamment proche de la solution exacte du problème de diffusion pour éviter que des scintillements apparaissent. Bien qu'il s'agisse d'un problème majeur avec les éléments linéaires, l'utilisation d'éléments quadratiques rend de tels artefacts difficilement remarquables.

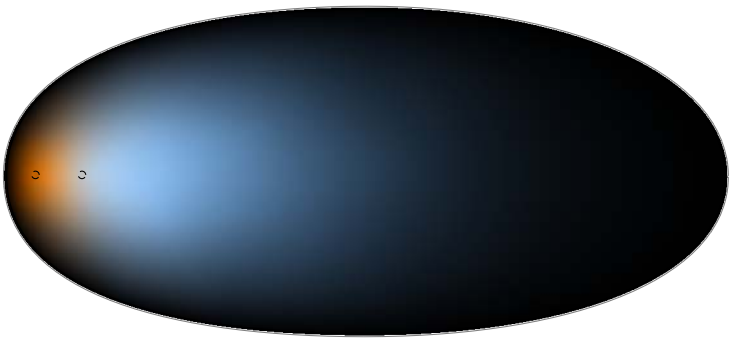


FIGURE 4.24 – **Déviation des couleurs.** Ici, l'ellipse émet du noir, et il y a deux points de contraintes : le plus à gauche est orange, l'autre est gris. A droite, du bleu apparaît (la couleur complémentaire du orange). Ce comportement est propre au bi-Laplacien et non à notre solveur.

En pratique, les problèmes de scintillement sont inexistants avec de la diffusion Laplacienne. Avec la diffusion bi-Laplacienne des variations de couleur rapides peuvent se produire à distance des courbes, auquel cas nos heuristiques ne suffisent pas pour créer une triangulation adaptée. Détecter *a priori* l'emplacement de tels détails est très difficile, et seule une approche itérative entrelaçant des passes de résolution avec des passes de re-triangulation/raffinement permettrait de garantir la qualité du résultat. Bien entendu, une telle approche n'est envisageable que pour générer une version haute qualité car elle violerait nos contraintes de performance lors de l'édition. En pratique, ces cas se produisent très rarement et sont souvent associés à des situations indésirables, comme la présence de fortes saturations.

Limitations inhérentes à l'interpolation bi-harmonique La diffusion bi-harmonique possède la capacité d'extrapoler les valeurs. Ainsi, trois contraintes ponctuelles (non colinéaires) produisent un dégradé linéaire s'étendant à l'infini. Ce comportement est désirable pour la création de cartes de détails mais pose problème lorsqu'il s'agit d'interpoler des couleurs. Les couleurs peuvent ainsi saturer, ce qui produit des artefacts visibles. En outre, cette extrapolation se produit séparément sur les différents canaux de l'image, ce qui peut faire apparaître des couleurs non présentes dans les contraintes, comme on peut le voir figure 4.24.

Dans le cadre d'un outils d'édition interactif, la saturation des couleurs peut facilement être corrigée par l'utilisateur en ajoutant quelques contraintes ponctuelles avec un gradient nul.

Dans un contexte différent, Jacobson *et al.* [2012] ont récemment mit au

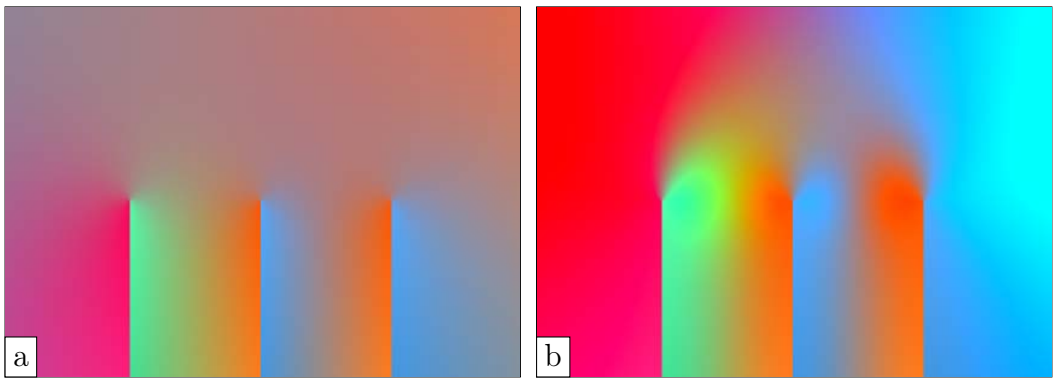


FIGURE 4.25 – **Mélange des contraintes.** Les courbes de la figure (a) contraignent une couleur différente de chaque côté en laissant le gradient libre, ce qui ne pose pas de problème. Les mêmes courbes avec des contraintes de gradient (dérivée normale nulle) (b) provoquent un comportement indésirable autour des extrémités (forte saturation), dont l'intensité dépend de la taille des éléments incidents.

point une méthode d'interpolation bi-harmonique où des contraintes supplémentaires permettent d'éviter l'insertion de nouveaux extrema. Le principe est relativement simple : une première étape d'interpolation harmonique permet d'initialiser en tout point du domaine une direction de gradient. Ensuite, un processus itératif d'optimisation non linéaire permet de résoudre l'équation bi-harmonique en contraignant le gradient de la solution finale à former un angle inférieur à $\pi/2$ avec le gradient de la solution harmonique. Bien entendu, cette approche s'avère trop gourmande pour notre contexte d'édition interactive.

L'interpolation bi-harmonique présente aussi quelques cas dégénérés. D'une part, il faut au moins trois contraintes de couleurs non-alignées pour que le problème ait une solution unique. Finch *et al.* utilisent une méthode de régularisation pour résoudre ce problème qui amène leur solveur à s'écarter de la solution de l'équation 3.6 à distance des contraintes. Une solution plus satisfaisante consiste à ajouter une contrainte de gradient nul dans une direction donnée, par exemple dans la direction orthogonale de l'alignement des points de contraintes. D'autre part, mélanger des contraintes de valeur et de gradient rend la solution instable aux extrémités des courbes ouvertes (figure 4.25.b). Il n'y a pas de solution à ce problème actuellement hormis d'éviter ces cas. En pratique, cela influence la façon dont l'interpolation bi-harmonique est utilisée : plutôt que d'utiliser des courbes avec des couleurs de chaque côté, il semble plus intéressant de définir des courbes sans contrainte de valeur qui décrivent les discontinuités et d'y ajouter des courbes émettant la même couleur des deux côtés : en évitant les intersections, il est ainsi possible d'éviter totalement les points de singularité.

4.6.5 Bilan

Comme nous l'avons vu section 3.2, les solveurs proposés jusqu'à présent peuvent se classer en deux catégories : la première regroupe les solveurs cherchant à résoudre une équation aux dérivées partielles, à savoir l'équation Laplacienne ou bi-Laplacienne. Tous les solveurs de cette catégorie résolvent le problème de manière globale directement sur la grille de pixels, ce qui engendre un certain nombre de difficultés : lorsque deux contraintes sont proches, leur discrétisation peut mener à des fuites de couleur, il est difficile de prendre en compte les contraintes hors champ et la discrétisation des courbes engendre des scintillements. De plus, dès que le point de vue est modifié (zoom ou translation), le solveur doit être relancé, ce qui implique que les performances se dégradent très rapidement dans le cas d'utilisation de calques.

La seconde catégorie regroupe les solveurs approchants. Ils utilisent une autre formulation du problème qui permet d'évaluer la solution en un point arbitraire. Cette propriété est utilisée pour faire du rendu adaptatif, en utilisant une triangulation intermédiaire, ou plus simplement une grille de résolution variable. Cependant, ces solveurs ne peuvent qu'approcher la diffusion Laplacienne et il n'est pas clair qu'une formulation équivalente existe pour approcher la diffusion bi-Laplacienne.

Nous avons présenté un solveur capable de résoudre l'équation voulue sur une triangulation adaptative. Il en résulte une représentation intermédiaire légère qui surpasse les solutions proposées précédemment dans de nombreux cas de figure. De plus, notre solveur possède un très bon rapport qualité/coût de calcul, ce qui permet de le faire tourner intégralement sur un CPU grand public. Notre implémentation actuelle montre déjà des performances compétitives par rapport aux autres solveurs, alors que, comme nous l'avons déjà mentionné, nous disposons de nombreuses opportunités pour l'accélérer d'avantage. Il s'agit principalement d'un travail d'ingénierie.

Plusieurs pistes sont ouvertes pour améliorer notre solveurs et poursuivre ces travaux. Par exemple, de nombreuse variantes de la FEM existent. En particulier, il serait intéressant d'étudier la possibilité de dériver une méthode à base d'éléments finis mixtes quadratique qui pourrait potentiellement simplifier le problème et ainsi améliorer encore les performances.

Le choix de l'espace de couleur dans lequel est effectué l'interpolation a une influence directe sur le résultat, et il serait intéressant d'étudier quel espace est le plus adapté. De plus, la diffusion de couleur se fait actuellement sur les différents canaux indépendamment, mais pourrait bénéficier d'une formulation les prenant en compte simultanément. De la même manière, la diffusion de carte de normales bénéficierait d'une véritable interpolation sphérique.

Une application plus directe que nous envisageons concerne la vectorisation pour la compression de cartes d'éclairement et de cartes d'environnement, qui sont principalement constituées de dégradés très lisses plus quelques dis-

continuités. Ces cartes sont généralement utilisées pour simuler des réflexions, pour lesquelles une image bitmap doit être de très haute résolution si l'on veut conserver des variations nettes. Notre représentation intermédiaire permet de représenter ces images en étant à la fois peu coûteuse en mémoire et relativement efficace en calculs. Notons que ces cartes sont généralement des images à grande dynamique, qui sont tout à fait représentables avec des images de diffusion et compatibles avec notre solveur. En ce qui concerne les cartes d'environnement, nous pourrions imaginer un outils de création dédié.

Les équations Laplaciennes et bi-Laplaciennes présentent chacune des avantages et des défauts. Une autre formulation du problème de diffusion pourrait peut-être permettre d'obtenir de meilleurs compromis. Actuellement, de nombreux travaux sont menés pour trouver des méthodes produisant des résultats similaires à de la diffusion poly-harmonique sans certains de ses défauts, comme par exemple dans l'article de Jacobson *et al.* [2012]. Il est possible de s'inspirer de ces travaux pour améliorer le comportement des outils de diffusion de couleur afin de les rendre plus prévisibles et contrôlables.

Conclusion

Représentation de surface vectorielle hybride

Avant de faire un bilan sur nos contributions et de présenter quelques perspectives de recherche, il nous semble intéressant de montrer en quoi nos deux techniques sont complémentaires. En effet, grâce à notre représentation intermédiaire, il est possible de combiner la méthode LS^3 et le principe des images de diffusion pour créer des modèles 3D complexes composés d'une surface de base définissant la forme générale de l'objet qui est enrichie par des cartes vectorielles représentant les détails, les micro-variations de normales, et les textures. Ce procédé fait appel à plusieurs étapes. La première consiste à calculer les représentations intermédiaires correspondant aux différents calques et cartes à l'aide de notre solveur présenté au chapitre 4. Le modèle de base est ensuite raffiné avec la méthode LS^3 afin de produire une surface lisse comme nous l'avons vu au chapitre 2. En supposant que des détails complexes sont définis par plusieurs calques de déplacements et de couleurs, il est difficilement envisageable de rentre ce maillage compatible avec toutes les triangulations des différents calques. À la place, nous envisageons une solution plus légère consistant à découper le maillage pour le rendre compatible uniquement avec les *discontinuités* (de tangentes et de valeurs) de l'image de diffusion. Les sommets peuvent ensuite être déplacés, et les fragments colorés, en réalisant des accès aléatoires directs aux différentes cartes vectorielles. Le résultat est une surface détaillée et texturée entièrement à l'aide de primitives vectorielles.

Contributions principales

Nous avons d'abord présenté une nouvelle **représentation de surface** fondée sur le principe de subdivision, appelée *least square subdivision surfaces* (LS^3). La principale originalité provient du découplage de l'opérateur de lissage traditionnel en une étape de régularisation du maillage (lissage tangentiel), suivi d'un lissage de la forme. Cette approche permet d'imaginer de nouvelles stratégies pour le lissage. Ici, nous avons opté pour une procédure de *projection sur des sphères* approchant localement la surface, directement inspirée des surfaces MLS. Nous avons aussi présenté des outils d'**analyse numérique des surfaces obtenues par un processus de subdivision** que nous avons utilisé pour étudier le comportement de nos surfaces LS^3 à la limite. La méthode LS^3 possède la majorité des bonnes propriétés des surfaces de subdivision tout en produisant des surfaces de meilleure qualité à proximité des sommets extraordinaires. Elle possède aussi quelques propriétés supplémentaires, comme un contrôle limité à l'aide des normales et une surface plus proche d'une interpolation.

Dans un second temps, nous nous sommes intéressés au problème de la représentation et de calculs de dégradés 2D, afin de représenter des détails

géométriques et des variations de couleurs. Pour cela nous avons généralisé certains travaux précédents sur les images de diffusion via une **classification** claire et précise des courbes et contraintes ne contenant pas de cas dégénéré. Nous avons ensuite introduit la notion de **courbes de transmission** apportant aux images de diffusion un moyen simple et efficace de limiter l'influence d'un ensemble de primitives. Il s'agit d'un *contrôle local* qui faisait cruellement défaut à ces méthodes. Finalement, notre principale contribution, est un nouveau **solveur vectoriel** pour la diffusion utilisant une triangulation associée à des patchs quadratiques comme représentation intermédiaire, et les FEM avec des éléments non-conformes pour la résolution des EDP. Nous avons démontré les avantages d'une telle représentation dans de nombreux cas d'utilisation, dont la représentation de détails géométriques et colorimétriques sur une surface.

Perspectives

Il est intéressant de remarquer que nos deux principales contributions s'attaquent toutes deux à des problèmes d'interpolation : dans le premier nous cherchons une surface approchant un réseau de points de contrôle, et dans le second nous interpolons des données très éparses et non structurées. Nous avons opté pour une approche par subdivision et approximation locale pour le premier, et une approche globale et variationnelle pour le second.

Curieusement, la subdivision a été récemment employée pour représenter des images vectorielles [Liao *et al.*, 2012], et inversement, les approches variationnelles peuvent également servir à définir des surfaces lisses [Andrews *et al.*, 2011]. Comme nous allons le voir, cette observation suggère de nombreuses perspectives intéressantes pour la suite de ces travaux.

Surface MLS appliquées aux maillages. Lors de la conception de la méthode LS^3, nous avons envisagé plusieurs pistes pour appliquer des surfaces MLS sur les maillages polygonaux. Ces pistes ont été mises de côté car il semble *a priori* difficile d'éviter certains défauts comme les interactions non désirées entre deux surfaces proches. Elles restent néanmoins intéressantes car elles offriraient la possibilité de produire une surface implicite lisse à partir d'un maillage, ce qui permet par exemple d'effectuer des opérations booléennes efficacement et facilement, opérations très délicates à effectuer avec des surfaces de subdivision par exemple. De telles surfaces seraient aussi facilement évaluable à des positions arbitraires, puisqu'il suffit d'appliquer l'opérateur de projection des surfaces MLS pour cela. Le principal challenge est donc de calculer des fonctions de poids lisses, à support compact se limitant de préférence au 2-anneau des sommets, et garantissant que l'intégralité de la surface soit couvert par suffisamment d'échantillons pour garantir un ajustement robuste.

Vers des approches unifiées. L'application d'images de diffusion vectorielles sur des modèles 3D n'a été que survolée et pose de nombreux défis. Dans l'état actuel, nous disposons de deux triangulations ; l'une est la surface de base, l'autre est une représentation vectorielle dense des détails. Nous avons montré qu'il est possible de rendre les deux triangulations compatibles en les superposant, mais cela s'avère assez coûteux et délicat à réaliser de manière robuste. L'approche par accès aléatoires n'est pas non plus sans poser des difficultés, surtout lorsqu'il s'agit d'appliquer des cartes de déplacement présentant des discontinuités. De plus, dans les deux cas, une paramétrisation du domaine est requise.

Une approche plus ambitieuse consisterait à définir les courbes directement sur la surface [Li *et al.*, 2005], puis d'effectuer la diffusion directement sur la surface 3D. Ainsi, il ne serait plus nécessaire de paramétrer la surface globalement, ce qui ouvrirait la possibilité d'habiller des surfaces par des textures vectorielles pour lesquelles calculer une telle paramétrisation est délicat, comme les surfaces implicites. De plus, cette approche permettrait de prendre en compte la forme de la surface durant la diffusion via l'opérateur de Laplace-Beltrami, qui est l'analogue de l'opérateur de Laplace pour les fonctions définies sur une surface. Plusieurs défis sont soulevés par cette approche : comment générer efficacement la triangulation de façon à représenter fidèlement à la fois la surface et la diffusion ? Comment représenter des courbes lisses et vectorielles sur une surface, de préférence en laissant à l'utilisateur la possibilité d'éditer la surface et les courbes à volonté ?

Une extension de cette approche consisterait à la combinée avec une technique de création et de manipulation de surface à partir de courbes [Andrews *et al.*, 2011]. Cela permettrai d'aboutir à une approche complètement unifiée pour la représentation et manipulation de surfaces et de sont habillages.

En parallèle, il nous semble intéressant d'étudier la possibilité d'utiliser des surfaces de subdivisions à la fois comme représentation intermédiaire et comme base de discrétisation pour le solveur FEM [Cirak *et al.*, 2000]. Un avantage immédiat est de produire un résultat réellement C^1 (et même plus), et donc une qualité accrue notamment lorsqu'il s'agit de créer des champs de hauteurs ou d'autres surfaces dont les défauts sont révélés par l'éclairement et les réflexions. Dans cette approche, la représentation est effectivement une surface de subdivision, et le concept des courbes de diffusion est alors vu comme une technique de plus haut niveau permettant de manipuler aisément la représentation sous-jacente.

Vectorisation d'images. Dans cette thèse nous nous sommes concentré sur les aspects création et manipulation des représentations vectorielles. Un autre aspect tout aussi important concerne la vectorisation automatique d'images matricielles. Convertir directement une image matricielle sous la forme de

courbes de diffusion s'avère peu efficace en pratique [Orzan *et al.*, 2008; Jeschke *et al.*, 2011]. Une approche inverse de notre solveur inspire une nouvelle façon de procéder : d'abord trianguler l'image, par exemple via un processus de subdivision inverse [Liao *et al.*, 2012], puis de convertir cette représentation intermédiaire en un ensemble de courbes et contraintes permettant de reproduire fidèlement l'image d'origine tout en étant facilement éditable.

Synthèse de texture Les détails et textures des objets sont souvent composés de motifs répétitifs. Afin d'éviter les répétitions trop marquées, des techniques de synthèse de textures procédurales à partir d'une image bitmap ont vu le jour [Lefebvre et Hoppe, 2006]. Une piste intéressante serait donc d'étudier la possibilité d'adapter ces techniques à la synthèse de textures vectorielles à partir de motifs vectoriels. Ici, partir directement de la représentation par courbes ne semble pas adapté puisque cela impliquerait un calcul de la diffusion dans tout l'espace. De nouveau, notre approche par une représentation intermédiaire vectorielle semble pertinente, ce qui n'empêche pas de conserver les information en entrée pour guider le processus de synthèse.

Approche multi-résolution Finalement, jusqu'à présent nous avons supposé que nos objets détaillés pouvaient être décomposés en une surface de base, des détails macroscopiques, des variations de normale et des variations de matériaux, et que chacune de ces couches pouvait être modélisée plus ou moins indépendamment. Dans des travaux futures, il serait intéressant d'étudier la possibilité de combiner ces différentes couches plus étroitement pour obtenir une vraie représentation multi-résolution où les informations pourraient être transférées d'une couche à un autre de manière continue en fonction de la résolution en sortie.

Bibliographie

Adamson, Anders et Alexa, Marc, 2006. Anisotropic point set surfaces. Dans *Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa - Afrigaph '06*, tome 25, page 7. ACM Press, New York, New York, USA. ISBN 1595932887. doi : 10.1145/1108590.1108592.

Alexa, Marc et Adamson, Anders, 2004. On normals and projection operators for surfaces defined by point sets. Dans Markus Gross, Hanspeter Pfister, Marc Alexa et Szymon Rusinkiewicz, rédacteurs, *Symposium on point-based graphics*, pages 149–155. Eurographics Association. doi : 10.2312/SPBG/SPBG04/149-155.

Alexa, Marc, Behr, Johannes, Cohen-Or, Daniel, Fleishman, Shachar, Levin, David et Silva, Claudio T., 2001. Point set surfaces. Dans *Proceedings Visualization 2001*, pages 21–537. IEEE. ISBN 0-7803-7200-X. doi : 10.1109/VISUAL.2001.964489.

Alexa, Marc, Behr, Johannes, Cohen-Or, Daniel, Fleishman, Shachar, Levin, David et Silva, Claudio T., 2003. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1) :3–15. doi :10.1109/TVCG.2003.1175093.

Alliez, P, Cohen-Steiner, D, Tong, Y et Desbrun, M, 2007. Voronoi-based variational reconstruction of unoriented point sets. Dans *Proceedings of the fifth Eurographics symposium on Geometry processing*, SGP '07, pages 39–48. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland. ISBN 978-3-905673-46-3.

Amenta, Nina et Kil, Yong Joo, 2004. Defining point-set surfaces. *ACM Transactions on Graphics*, 23(3) :264. doi :10.1145/1015706.1015713.

Andrews, James, Joshi, Pushkar et Carr, Nathan, 2011. A Linear Variational System for Modeling From Curves. *Computer Graphics Forum*, 30(6) :1850–1861.

Barthe, Loïc, Dodgson, Neil, Sabin, Malcolm, Wyvill, Brian et Gaildrat, Véronique, 2003. Two-dimensional potential fields for advanced implicit modeling operators. *Computer Graphics Forum*, 22(1) :1–11.

Barthe, Loïc, Gérot, Cédric, Sabin, Malcolm A. et Kobbelt, Leif, 2005. Simple computation of the eigencomponents of a subdivision matrix in the fourier domain. Dans Neil Dodgson, Michael S. Floater et Malcolm Sabin, rédacteurs, *Advances in Multiresolution for Geometric Modelling*, chapitre Simple com, pages 245–257. ISBN 3-540-21462-3. doi : 10.1007/3-540-26808-1_13.

Barthe, Loïc et Kobbelt, Leif, 2004. Subdivision scheme tuning around extraordinary vertices. *Computer Aided Geometric Design*, 21(6) :561–583. doi :10.1016/j.cagd.2004.04.003.

Belytschko, T, Huerta, A, Fernández-Méndez, S et Rabczuk, T, 2004. *Meshless methods*, chapitre 10. John Wiley & Sons.

Bernhardt, Adrien, Barthe, Loic, Cani, Marie-Paule et Wyvill, Brian, 2010. Implicit Blending Revisited. *Computer Graphics Forum*, 29(2) :367–375. doi :10.1111/j.1467-8659.2009.01606.x.

Bezerra, Hedlena, Eisemann, Elmar, DeCarlo, Doug et Thollot, Joëlle, 2010a. Controllable Diffusion Curves. Rapport technique 6, INRIA.

Bezerra, Hedlena, Eisemann, Elmar, DeCarlo, Doug et Thollot, Joëlle, 2010b. Diffusion constraints for vector graphics. Dans *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering - NPAR '10*, page 35. ACM Press, New York, New York, USA. ISBN 9781450301251. doi :10.1145/1809939.1809944.

Biermann, Henning, Levin, Adi et Zorin, Denis, 2000. Piecewise smooth subdivision surfaces with normal control. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00*, pages 113–120. doi :10.1145/344779.344841.

Blinn, James F., 1982. A generalization of algebraic surface drawing. *ACM SIGGRAPH Computer Graphics*, 16(3) :273. doi :10.1145/965145.801290.

Bloomenthal, Jules et Wyvill, Brian, rédacteurs, 1997. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. ISBN 155860233X.

Botsch, Mario et Kobbelt, Leif, 2004. A remeshing approach to multiresolution modeling. *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing - SGP '04*, page 185. doi : 10.1145/1057432.1057457.

Botsch, Mario et Sorkine, Olga, 2008. On linear variational surface deformation methods. *IEEE transactions on visualization and computer graphics*, 14(1) :213–30. doi :10.1109/TVCG.2007.1054.

Bowers, Jhon C., Leahey, Jonathan et Wang, Rui, 2011. A ray tracing approach to diffusion curves. *Computer Graphics Forum*, 30(4).

Briggs, William L., Henson, Van Emden et McCormick, Steve F., 2000. A multigrid tutorial : second edition.

Bruno, L et Cavin, Xavier, 2005. Vector Texture Maps. Rapport technique, INRIA - ALICE.

Carr, J C, Beatson, R K, Cherrie, J B, Mitchell, T J, Fright, W R, McCallum, B C et Evans, T R, 2001. Reconstruction and representation of 3D objects with radial basis functions. Dans *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 67–76. ACM, New York, NY, USA. ISBN 1-58113-374-X. doi :10.1145/383259.383266.

Cashman, Thomas J., Augsdörfer, Ursula H., Dodgson, Neil A. et Sabin, Malcolm A., 2009. NURBS with extraordinary points. *ACM Transactions on Graphics*, 28(3) :1. doi :10.1145/1531326.1531352.

Catmull, Edwin et Clark, Jim, 1978. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6) :350–355. doi :10.1016/0010-4485(78)90110-0.

CGAL, 2012. *Computational Geometry Algorithms Library*.
URL `http://www.cgal.org/`

Chen, Long, 2004. Mesh Smoothing Schemes Based On Optimal Delaunay Triangulations. Dans *In Proceedings of 13th International Meshing Roundtable*, pages 109–120.

Chen, Zhangxin, 2005. *Finite element methods and their applications*. ISBN 978-3-540-24078-5.

Chernikov, Andrey et Chrisochoides, Nikos, 2006. Generalized Delaunay Mesh Refinement : From Scalar to Parallel. Dans *International Meshing Roundtable*, pages 563—-580.

Ciarlet, Philippe, 1978. *The finite element method for elliptic problems*. Studies in mathematics and its applications ; v. 4.

Cirak, Fehmi, Ortiz, Michael et Schröder, Peter, 2000. Subdivision Surfaces : A New Paradigm For Thin-Shell Finite-Element Analysis. *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING*, 47 :2039–2072.

Coons, Steven Anson, 1967. Surfaces for computer-aided design of space forms. Rapport technique.

DeRose, Tony, Kass, Michael et Truong, Tien, 1998. Subdivision surfaces in character animation. Dans *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*, pages 85–94. ACM Press, New York, New York, USA. ISBN 0897919998. doi : 10.1145/280814.280826.

Donohue, Charles et Ostromoukhov, Victor, 2007. Fast Generation of Importance-sampled Point Sets with Associated Delaunay Triangulation. Dans *Proceedings of GRAPHICON'2007*, pages 125–130.

Du, Qiang, Faber, Vance et Gunzburger, Max, 1999. Centroidal Voronoi Tessellations : Applications and Algorithms. *SIAM Rev.*, 41(4) :637–676. doi :10.1137/S0036144599352836.

Dyn, Nira, Levin, David et Gregory, John A., 1990. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2) :160–169. doi :10.1145/78956.78958.

Eigen, 2012. *Eigen 3.1.1*.
URL `http://eigen.tuxfamily.org/`

Eppstein, David, 2001. Global optimization of mesh quality. Tutorial at the 10th International Meshing Roundtable.

Farbman, Zeev, Hoffer, Gil, Lipman, Yaron, Cohen-or, Daniel et Lischinski, Dani, 2009. Coordinates for instant image cloning. *ACM SIGGRAPH 2009 papers on - SIGGRAPH '09*, 28(3) :1. doi :10.1145/1576246.1531373.

Farin, Gerald, 2002. *Curves and Surfaces for CAGD : A Practical Guide*. Morgan-Kaufmann, 5th editio édition. ISBN 1-55860-737-4.

Ferguson, James, 1964. Multivariable Curve Interpolation. *Journal of the ACM*, 11(2) :221–228. doi :10.1145/321217.321225.

Finch, Mark, Snyder, John et Hoppe, Hugues, 2011. Freeform vector graphics with controlled thin-plate splines. *ACM Transactions on Graphics*, 30(6) :1. doi :10.1145/2070781.2024200.

Floater, Michael S., 2003. Mean value coordinates. *Computer Aided Geometric Design*, 20(1) :19–27. doi :10.1016/S0167-8396(03)00002-5.

Fraeijs de Veubeke, Baudouin M., 1974. Variational principles and the patch test. *International Journal for Numerical Methods in Engineering*, 8(4) :783–801. doi :10.1002/nme.1620080408.

Frey, Pascal J. et Marechal, Loïc, 1998. Fast adaptive quadtree mesh generation. Dans *Proceedings of the Seventh International Meshing Roundtable*, pages 211–224.

Goldman, Ron, 2005. Curvature formulas for implicit curves and surfaces. *Comput. Aided Geom. Des.*, 22(7) :632–658. doi :10.1016/j.cagd.2005.06.005.

Grinspun, Eitan, Gingold, Yotam, Reisman, Jason et Zorin, Denis, 2006. Computing discrete shape operators on general meshes. *Computer Graphics Forum*, 25(3) :547–556. doi :10.1111/j.1467-8659.2006.00974.x.

Gross, Markus et Pfister, Hanspeter, 2007. *Point-Based Graphics*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. ISBN 0123706041, 9780080548821.

Grossman, J P et Dally, William J, 1998. Point Sample Rendering. Dans *In Rendering Techniques '98*, pages 181–192. Springer.

Guennebaud, Gaël, Germann, Marcel et Gross, Markus, 2008. Dynamic Sampling and Rendering of Algebraic Point Set Surfaces. *Computer Graphics Forum*, 27(2) :653–662. doi :10.1111/j.1467-8659.2008.01163.x.

Guennebaud, Gaël et Gross, Markus, 2007. Algebraic point set surfaces. *ACM Transactions on Graphics*, 26(3) :23. doi :10.1145/1276377.1276406.

Guo, Xiaohu, Li, Xin, Bao, Yunfan, Gu, Xianfeng et Qin, Hong, 2006. Meshless thin-shell simulation based on global conformal parameterization. *IEEE transactions on visualization and computer graphics*, 12(3) :375–85. doi : 10.1109/TVCG.2006.52.

Hnaidi, Houssam, Guérin, Eric, Akkouche, Samir, Peytavie, Adrien et Galin, Eric, 2010. Feature based terrain generation using diffusion equation. *Computer Graphics Forum*, 29(7) :2179–2186. doi :10.1111/j.1467-8659.2010.01806.x.

Hoppe, Hugues, DeRose, Tony, Duchamp, Tom, McDonald, John et Stuetzle, Werner, 1992. Surface reconstruction from unorganized points. *SIGGRAPH Comput. Graph.*, 26(2) :71–78. doi :10.1145/142920.134011.

Jacobson, Alec, Tosun, Elif, Sorkine, Olga et Zorin, Denis, 2010. Mixed Finite Elements for Variational Surface Modeling. *Computer Graphics Forum*, 29(5) :1565–1574. doi :10.1111/j.1467-8659.2010.01765.x.

Jacobson, Alec, Weinkauf, Tino et Sorkine, Olga, 2012. Smooth Shape-Aware Functions with Controlled Extrema. *Computer Graphics Forum*, 31(5) :1577–1586. doi :10.1111/j.1467-8659.2012.03163.x.

Jeschke, Stefan, Cline, David et Wonka, Peter, 2009a. A GPU Laplacian solver for diffusion curves and Poisson image editing. Dans *ACM SIGGRAPH Asia 2009 papers on - SIGGRAPH Asia '09*, page 1. ACM Press, New York, New York, USA. ISBN 9781605588582. doi :10.1145/1661412.1618462.

Jeschke, Stefan, Cline, David et Wonka, Peter, 2009b. Rendering surface details with diffusion curves. Dans *ACM SIGGRAPH Asia 2009 papers on - SIGGRAPH Asia '09*, tome 28, page 1. ACM Press, New York, New York, USA. ISBN 9781605588582. doi :10.1145/1661412.1618463.

Jeschke, Stefan, Cline, David et Wonka, Peter, 2011. Estimating Color and Texture Parameters for Vector Graphics. *Computer Graphics Forum*, 30(2) :523–532. doi :10.1111/j.1467-8659.2011.01877.x.

Jin, Shuangshuang, Lewis, Robert R et West, David, 2005. A comparison of algorithms for vertex normal computation. *The Visual Computer*, 21(1-2) :71–82. doi :10.1007/s00371-004-0271-1.

Ju, Tao, Schaefer, Scott et Warren, Joe, 2005. Mean value coordinates for closed triangular meshes. *ACM Transactions on Graphics*, 24(3) :561. doi :10.1145/1073204.1073229.

Karčiauskas, Kestutis, Peters, Jörg et Reif, Ulrich, 2004. Shape characterization of subdivision surfaces—case studies. *Computer Aided Geometric Design*, 21(6) :601–614. doi :10.1016/j.cagd.2004.04.005.

Kazhdan, Michael, Bolitho, Matthew et Hoppe, Hugues, 2006. Poisson surface reconstruction. Dans *Proceedings of the fourth Eurographics symposium on Geometry processing*, SGP '06, pages 61–70. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland. ISBN 3-905673-36-3.

Kobbelt, Leif, 1996. Interpolatory Subdivision on Open Quadrilateral Nets with Arbitrary Topology. *Computer Graphics Forum*, 15(3) :409–420. doi : 10.1111/1467-8659.1530409.

KOBBELT, Leif, 2000. $\sqrt{3}$-subdivision. Dans *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00*, pages 103–112. ACM Press, New York, New York, USA. ISBN 1581132085. doi :10.1145/344779.344835.

KOVACS, Denis, MITCHELL, Jason, DRONE, Shanon et ZORIN, Denis, 2009. Real-time creased approximate subdivision surfaces. Dans *Proceedings of the 2009 symposium on Interactive 3D graphics and games - I3D '09*, tome 16, page 155. ACM Press, New York, New York, USA. ISBN 9781605584294. doi :10.1145/1507149.1507174.

KUIJT, Frans, 1998. *Convexity preserving interpolation - stationary nonlinear subdivision and splines*. Phd, University of Twenty.

LABSIK, Ulf et GREINER, Günther, 2000. Interpolatory $\sqrt{3}$-Subdivision. *Computer Graphics Forum*, 19(3) :131–138. doi :10.1111/1467-8659.00405.

LAGAE, Ares, LEFEBVRE, Sylvain, DRETTAKIS, George et DUTRÉ, Philip, 2009. Procedural noise using sparse Gabor convolution. *ACM Transactions on Graphics*, 28(3) :1. doi :10.1145/1531326.1531360.

LASCAUX, P. et LESAINT, P., 1975. Some nonconforming finite elements for the plate bending problem. *ESAIM : Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 9 :9—-53.

LECOT, Gregory et LEVY, Bruno, 2006. ARDECO : Automatic region detection and conversion. Dans *17th Eurographics Symposium on Rendering - EGSR'06*, pages 349–360.

LEFEBVRE, Sylvain et HOPPE, Hugues, 2006. Appearance-space texture synthesis. *ACM Transactions on Graphics*, 25(3) :541. doi :10.1145/1141911.1141921.

LEVIN, Adi, 2006. Modified subdivision surfaces with continuous curvature. *ACM Transactions on Graphics*, 25(3) :1035. doi :10.1145/1141911.1141990.

LEVIN, David, 1998. The approximation power of moving least-squares. *Mathematics of computation*, 67 :1517–1531.

LEVIN, David, 2003. Mesh Independent Surface Interpolation. Dans G. Brunnett, B. Hamann, H. Müller et L. Linsen, rédacteurs, *Geometric Modeling for Scientific Visualization*, pages 37–49. Springer-Verlag.

LI, WC, LEVY, B. et PAUL, JC, 2005. Mesh editing with an embedded network of curves. Dans *International Conference on Shape Modeling and Applications 2005 (SMI' 05)*, pages 62–71. IEEE Comput. Soc. ISBN 0-7695-2379-X. doi :10.1109/SMI.2005.29.

Liao, Zicheng, Hoppe, Hugues, Forsyth, David et Yu, Yizhou, 2012. A Subdivision-Based Representation for Vector Image Editing. *IEEE transactions on visualization and computer graphics*, pages 1–11. doi :10.1109/TVCG.2012.76.

Lipman, Yaron, Kopf, Johannes, Cohen-Or, D et Levin, David, 2007. GPU-assisted positive mean value coordinates for mesh deformations. Dans *SGP '07 Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 117–123.

Liu, G R, 2009. *Mesh Free Methods, 2nd ed.* CRC Press.

Liu, Yang, Wang, Wenping, Lévy, Bruno, Sun, Feng, Yan, Dong-Ming, Lu, Lin et Yang, Chenglei, 2009. On centroidal voronoi tessellation—energy smoothness and fast computation. *ACM Transactions on Graphics*, 28(4) :1–17. doi :10.1145/1559755.1559758.

Lloyd, Stuart P, 1982. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28 :129–137.

Loop, Charles, 1987. *Smooth subdivision surfaces based on triangles.* Masters thesis, The University of Utah.

Loop, Charles, 2002. Bounded curvature triangle mesh subdivision with the convex hull property. *The Visual Computer*, 18(5-6) :316–325. doi :10.1007/s003710100148.

Lorensen, William E. et Cline, Harvey E., 1987. Marching cubes : A high resolution 3D surface construction algorithm. Dans *Proceedings of the 14th annual conference on Computer graphics and interactive techniques - SIGGRAPH '87*, pages 163–169. ACM Press, New York, New York, USA. ISBN 0897912276. doi :10.1145/37401.37422.

Macêdo, Ives, Gois, João Paulo et Velho, Luiz, 2009. Hermite Interpolation of Implicit Surfaces with Radial Basis Functions. Dans *Proceedings of the 2009 XXII Brazilian Symposium on Computer Graphics and Image Processing*, SIBGRAPI '09, pages 1–8. IEEE Computer Society, Washington, DC, USA. ISBN 978-0-7695-3813-6. doi :10.1109/SIBGRAPI.2009.11.

Marinov, Martin, Dyn, Nira et Levin, David, 2005. Geometrically Controlled 4-Point Interpolatory Schemes. Dans Neil Dodgson, Michael S. Floater et Malcolm Sabin, rédacteurs, *Advances in Multiresolution for Geometric Modelling*, pages 301–319. Springer-Verlag.

Max, Nelson, 1983. Computer Representation of Molecular Surfaces. *IEEE Computer Graphics and Applications*, 3(5) :21–29. doi :10.1109/MCG.1983.263183.

Max, Nelson, 1999. Weights for Computing Vertex Normals from Facet Normals. *Journal of Graphics Tools*, 4(2) :1–6. doi :10.1080/10867651.1999.10487501.

Meshlab, 2012. *Meshlab*.
URL `http://meshlab.sourceforge.net/`

Meyer, Mark, Barr, Alan, Lee, Haeyoung et Desbrun, Mathieu, 2002a. Generalized Barycentric Coordinates on Irregular Polygons. *Journal of Graphics Tools*, 7(1) :13–22. doi :10.1080/10867651.2002.10487551.

Meyer, Mark, Desbrun, Mathieu, Schröder, Peter et Barr, Alan H., 2002b. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. Dans *Visualization and Mathematics*, pages 35–57.

Morley, L. S. D., 1971. The constant-moment plate-bending element. *The Journal of Strain Analysis for Engineering Design*, 6(1) :20–24. doi :10.1243/03093247V061020.

Nave, Démian, Chrisochoides, Nikos et Chew, L Paul, 2002. Guaranteed : quality parallel delaunay refinement for restricted polyhedral domains. Dans *Proceedings of the eighteenth annual symposium on Computational geometry*, SCG '02, pages 135–144.

Nehab, Diego et Hoppe, Hugues, 2008. Random-access rendering of general vector graphics. *ACM Transactions on Graphics*, 27(5) :1. doi :10.1145/1409060.1409088.

Niessner, Matthias, Loop, Charles, Meyer, Mark et Derose, Tony, 2012. Feature-adaptive GPU rendering of Catmull-Clark subdivision surfaces. *ACM Transactions on Graphics*, 31(1) :1–11. doi :10.1145/2077341.2077347.

Orzan, Alexandrina, Bousseau, Adrien, Winnemöller, Holger, Barla, Pascal, Thollot, Joëlle et Salesin, David, 2008. Diffusion curves : A Vector Representation for Smooth-Shaded Images. *ACM Transactions on Graphics*, 27(3) :1. doi :10.1145/1360612.1360691.

Ostromoukhov, Victor, 1993. Hermite approximation for o set curve computation. Dans S.P. Mudur et S.N. Pattanaik, rédacteurs, *Proceedings of the International Conference on Computer Graphics ICCG93*, pages 189–196.

Oswald, Peter et Schröder, Peter, 2003. Composite primal/dual sqrt{3}-subdivision schemes. *Computer Aided Geometric Design*, 20(3) :135–164. doi :10.1016/S0167-8396(03)00026-8.

Pang, Wai-Man, Qin, Jing, Cohen, Michael, Heng, Pheng-Ann et Choi, Kup-Sze, 2012. Fast Rendering of Diffusion Curves with Triangles. *IEEE Computer Graphics and Applications*, 32(4) :68–78. doi :10.1109/MCG.2011.86.

Parilov, Evgueni et Zorin, Denis, 2008. Real-time rendering of textures with feature curves. *ACM Transactions on Graphics*, 27(1) :1–15. doi : 10.1145/1330511.1330514.

Pauly, Mark, Keiser, Richard, Kobbelt, Leif P. et Gross, Markus, 2003. Shape modeling with point-sampled geometry. Dans *ACM SIGGRAPH 2003 Papers on - SIGGRAPH '03*, pages 641–650. ACM Press, New York, New York, USA. ISBN 1581137095. doi :10.1145/1201775.882319.

Petitjean, Sylvain, 2002. A survey of methods for recovering quadrics in triangle meshes. *ACM Computing Surveys*, 34(2) :211–262. doi :10.1145/508352.508354.

Reif, Ulrich, 1996. A degree estimate for subdivision surfaces of higher regularity. *Procedings of the American Mathematical Society*, 124 :2167–2174.

Rong, Guodong, Liu, Yang, Wang, Wenping, Yin, Xiaotian, Gu, Xianfeng David et Guo, Xiaohu, 2011. GPU-Assisted Computation of Centroidal Voronoi Tessellation. *IEEE Trans. Vis. Comput. Graph.*, 17(3) :345–356.

Sabin, Malcolm A. et Barthe, Loïc, 2002. Artifacts in recursive subdivision surfaces. Dans *Curve and Surface Fitting*, pages 353–362. Nashboro Press.

Shewchuk, Jonathan Richard, 2000. Mesh generation for domains with small angles. Dans *Proceedings of the sixteenth annual symposium on Computational geometry - SCG '00*, tome 94720, pages 1–10. ACM Press, New York, New York, USA. ISBN 1581132247. doi :10.1145/336154.336163.

Spielman, Daniel A, Teng, Shang-Hua, Üngör, Alper, Shang-hua, I et Alper, Teng, 2002. Parallel Delaunay Refinement : Algorithms and Analyses. Dans *In Proceedings, 11th International Meshing Roundtable*, pages 205–217.

Stam, Jos, 1998. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. Dans *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*, pages

395–404. ACM Press, New York, New York, USA. ISBN 0897919998. doi : 10.1145/280814.280945.

Stam, Jos et Loop, Charles, 2003. Quad/Triangle Subdivision. *Computer Graphics Forum*, 22(1) :79–85. doi :10.1111/1467-8659.t01-2-00647.

Stanculescu, Lucian, Chaine, Raphaëlle et Cani, Marie-Paule, 2011. Freestyle : Sculpting Meshes with Self-Adaptive Topology. *Computers & Graphics*, 35(3) :614–622. doi :10.1016/j.cag.2011.03.033.

Sun, Jian, Liang, Lin, Wen, Fang et Shum, Heung-Yeung, 2007. Image vectorization using optimized gradient meshes. *ACM Transactions on Graphics*, 26(3) :11. doi :10.1145/1276377.1276391.

Sun, Xin, Xie, Guofu, Dong, Yue, Lin, Stephen, Xu, Weiwei, Wang, Wencheng, Tong, Xin et Guo, Baining, 2012. Diffusion curve textures for resolution independent texture mapping. *ACM Transactions on Graphics*, 31(4) :1–9. doi :10.1145/2185520.2185570.

SVG, 2011. *Scalable Vector Graphics (SVG) 1.1 (Second Edition)*. URL `http://www.w3.org/TR/SVG/`

Takayama, Kenshi, Sorkine, Olga, Nealen, Andrew et Igarashi, Takeo, 2010. Volumetric modeling with diffusion surfaces. Dans *ACM SIGGRAPH Asia 2010 papers on - SIGGRAPH ASIA '10*, page 1. ACM Press, New York, New York, USA. ISBN 9781450304399. doi :10.1145/1882262.1866202.

Tournois, Jane, Alliez, Pierre et Devillers, Olivier, 2007. Interleaving Delaunay Refinement and Optimization for 2D Triangle Mesh Generation. Dans Michael L Brewer et David L Marcum, rédacteurs, *IMR*, pages 83–101. Springer. ISBN 978-3-540-75102-1. doi :10.1007/978-3-540-75103-8_5.

Vlachos, Alex, Peters, Jörg, Boyd, Chas et Mitchell, Jason L., 2001. Curved PN triangles. Dans *Proceedings of the 2001 symposium on Interactive 3D graphics - SI3D '01*, pages 159–166. ACM Press, New York, New York, USA. ISBN 1581132921. doi :10.1145/364338.364387.

Welch, William et Witkin, Andrew, 1992. Variational surface modeling. *Proceedings of the 19th annual conference on Computer graphics and interactive techniques - SIGGRAPH '92*, pages 157–166. doi :10.1145/133994.134033.

Winnemöller, Holger, Orzan, Alexandrina, Boissieux, Laurence et Thollot, Joëlle, 2009. Texture Design and Draping in 2D Images. *Computer Graphics Forum*, 28(4) :1091–1099. doi :10.1111/j.1467-8659.2009.01486.x.

Wyvill, Brian, Guy, Andrew et Galin, Eric, 1999. Extending the CSG Tree - Warping, Blending and Boolean Operations in an Implicit Surface Modeling System. *Comput. Graph. Forum*, 18(2) :149–158.

Wyvill, Geoff, McPheeters, Craig et Wyvill, Brian, 1986. Data structure for soft objects. *The Visual Computer*, 2(4) :227–234. doi :10.1007/BF01900346.

Zienkiewicz, O. C., Taylor, R. L. et Zhu, J. Z., 2005. *The Finite Element Method : Its Basis and Fundamentals*. Butterworth-Heinemann. ISBN 978-0-7506-6320-5.

Zorin, Denis et Schröder, Peter, 2000. Subdivision for Modeling and Animation.

Zorin, Denis, Schröder, Peter et Sweldens, Wim, 1996. Interpolating Subdivision for meshes with arbitrary topology. Dans *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*, pages 189–192. ACM Press, New York, New York, USA. ISBN 0897917464. doi :10.1145/237170.237254.

Zwicker, Matthias, Pauly, Mark, Knoll, Oliver et Gross, Markus, 2002. Pointshop 3D : an interactive system for point-based surface editing. *ACM Transactions on Graphics*, 21(3) :322–329. doi :10.1145/566654.566584.

Zwicker, Matthias, Pfister, Hanspeter, van Baar, Jeroen et Gross, Markus H, 2001. Surface splatting. Dans *SIGGRAPH*, pages 371–378.

Expériences d'ajustement

Nous avons présenté la méthode LS^3 en utilisant des sphères comme approximations locales. Bien qu'une sphère seule ne permette de capturer fidèlement que des portions de surfaces isotropes, nos résultats montrent qu'une fois combinées entre elles, cela n'empêche pas la génération de surfaces de qualités dans des configurations bien plus complexes comme des selles ou cylindres. Néanmoins, une extension évidente serait d'utiliser des ellipsoïdes voire même des quadriques généralisées comme approximation locale, ce qui, à priori, offrirait de nombreux avantages. En effet, ajuster des quadriques généralisées permettrait de reproduire exactement non seulement des sphères ou plans comme nous pouvons le faire actuellement, mais également des ellipsoïdes, cylindres, paraboloïdes, et hyperboloïdes. De plus, comme la transformation affine d'une quadrique généralisée est également une quadrique généralisée, cela permettrait potentiellement de définir des surfaces LS^3 invariantes aux transformations affines. Ce dernier point semble en effet atteignable notamment grâce à l'utilisation de poids purement topologiques, et du calcul des normales initiales via la méthode de pondération par les aires qui, comme nous l'avons déjà vu, possède la propriété d'invariance affine.

Approximation locale de plus haut degré. Pour les deux objectifs, une attentions particulière doit être portée sur la procédure de minimisation qui doit être non biaisée et continue par rapport aux données d'entrée.

Une quadrique algébrique est définie par la 0-isosurface d'un champ de potentiel de la forme $f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \mathbf{x}^{\mathrm{T}}\mathbf{l} + c$ où $\mathbf{Q}$ est une matrice symétrique. Si $\mathbf{Q} = q\mathbf{I}$, nous retombons alors sur l'équation algébrique d'une sphère. Les ellipsoïdes correspondent aux cas où les valeurs propres de $\mathbf{Q}$ sont de même signes et non nulles. Alors que le passage de plans aux sphères n'augmente le nombre de degrés de liberté que d'une unité, le passage de sphères aux quadriques généralisées (ou ellipsoïdes) le double. Cela rend l'utilisation des normales des échantillons encore plus important pour que l'ajustement soit robuste avec peu d'échantillons. Par ailleurs, les normales sont nécessaires pour lever certaines ambiguïtés, car sans elles, par exemple, une ellipse très plate peut être obtenue en approchant un plan bruité, comme illustré figure 27.a. La reconstruction de quadriques à partir d'un ensemble de points est un problème

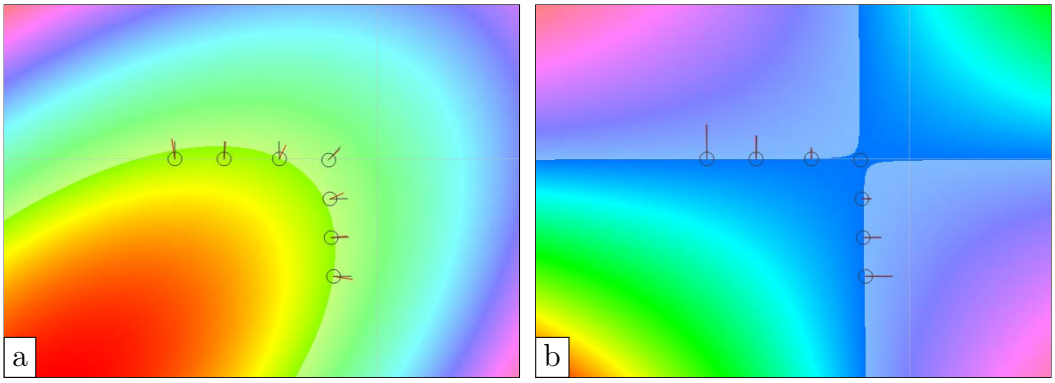


FIGURE 26 – **Ajustement de quadriques.** Plusieurs difficultés apparaissent avec l'ajustement de quadriques en utilisant une approche similaire à ASF. D'abord, le gradient n'est pas constant, ce qui produit une mauvaise approximation (a). Il est possible de redimensionner les normales itérativement pour que leurs magnitudes correspondent à celles du gradient. Cette procédure converge vers une bien meilleure approximation (b) ; cependant nous pouvons constater que les quadriques à deux nappes engendrent des "trous" dans la surface.

étudié régulièrement [Petitjean, 2002]. Cependant, les méthodes existantes sont fortement biaisées, et n'exploitent que très rarement l'information de normale.

Nous avons donc étudié la possibilité d'adapter la procédure ASF qui revient ici à minimiser $\sum w_j(\nabla f(\mathbf{p}_i) - \mathbf{n}_j)^2$, avec $\nabla f(\mathbf{p}_i) = 2\mathbf{Q}\mathbf{p}_j + \mathbf{l} - \mathbf{n}_j$, ce qui permet d'obtenir $\mathbf{Q}$ et $\mathbf{l}$. La constante c est obtenue dans un second temps en minimisant $f(\mathbf{p}_i)$. Cependant, cette procédure, que nous appellerons AAQF pour *approximative algebraique quadrique fitting*, présente de nombreux raccourcis. En effet, contrairement au cas des sphères où la magnitude du gradient est constante le long de chaque isosurface, dans le cas des quadriques la norme du gradient est bien plus importante dans les zones de faibles courbures. En d'autres termes, cela introduit un fort biais que ce soit dans la première ou même la seconde minimisation (puisque la distance algébrique n'est pas uniforme). Une telle procédure n'est donc pas capable de reproduire une quadrique même si les sommets et normales en entrée correspondent exactement à une quadrique.

Minimiser les différences de normales $\nabla f(\mathbf{p}_j)/|\nabla f(\mathbf{p}_j)| \approx \mathbf{n}_j$ n'est pas une option envisageable car cela conduit à une minimisation non linéaire. Une piste plus prometteuse consiste à mettre à l'échelle les normales $\mathbf{n}_j$ en fonction de la courbure locale de telle sorte qu'elles correspondent à des vecteurs gradient d'une quadrique. Plus précisément, la minimisation devient $\nabla f(\mathbf{p}_j) \approx \beta_j \mathbf{n}_j$. Pour un voisinage donné, les β_j peuvent être déterminés relativement aux autres en instançiant la formule de la courbure moyenne pour les surfaces

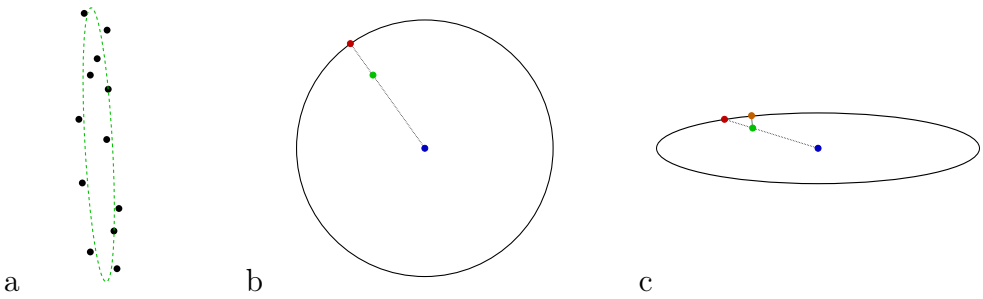


FIGURE 27 – **Ajustement d'ellipses et projections.** Ajuster une ellipse sur un nuage de points bruité représentant un plan produit généralement une ellipse très aplatie (a). La projection (en rouge) d'un point (en vert) se trouve dans l'alignement du centre de la sphère (en bleu) et du point projeté (b). Dans le cas d'une ellipse, la projection au point le plus proche (en orange) n'est pas invariante aux transformations affines. Il est possible de prendre le point dans l'alignement du centre de l'ellipse et du point à projeter, mais cela implique un déplacement tangentiel indésirable.

implicites [Goldman, 2005] aux quadriques. Fixer $\beta_0 = 1$ par exemple, permet donc de fixer les autres β_j en fonction du ratio entre κ_0 et κ_j. En supposant que nous soyons capable d'estimer les courbures moyennes de manière satisfaisante, cette approche souffre encore d'un problème fondamental lorsqu'un voisinage est à cheval sur une zone plane et une zone courbe puisque cela implique des gradients de magnitudes infinies pour la zone plane. De plus, même si nos tests ont montré que cette méthode produit des quadriques approchant raisonnablement bien la surface, elle ne sont pas pour autant optimales.

Une autre solution consiste à calculer une estimation initiale avec la procédure AAQF en utilisant des normales unitaires $\mathbf{n}_j^0$, puis à itérativement redimensionner les normales des échantillons tel que $\mathbf{n}_j^{k+1} = \frac{\mathbf{n}_j^k \|\nabla f(\mathbf{p}_j)\|}{\|\mathbf{n}_j^k\|}$. Le résultat de cette approche est illustré figure 26. Cependant, deux problèmes se posent : d'abord cette méthode converge assez lentement, ensuite il n'est pas garanti que la solution obtenue ne soit pas un minimum local.

Projection En supposant que nous ayons calculé une quadrique ou un éllipsoïde, il faut ensuite être capable de projeter le point d'évaluation dessus. Si l'on souhaite obtenir la propriété d'invariance affine, la projection doit aussi posséder cette propriété. Ce n'est clairement pas le cas de la projection sur le point de la surface le plus proche, qui peut être approché à l'aide d'une descente de gradient. Dans le cas des sphères, nous projetons le point d'évaluation sur la sphère dans l'axe de son centre. Une approche similaire transposée aux ellipsoïdes est invariante aux transformations affines, qui conservent les lignes

droites, si l'approximation locale l'est aussi. Cependant, cette méthode n'est pas transposable aux quadriques généralisées, qui peuvent représenter des surfaces ouvertes : il se peut donc qu'il n'y ait pas de surface dans l'alignement du "centre" de la quadrique et du point d'évaluation. De plus, cette approche induit un décalage tangentiel non désiré comme illustré figure 27. Il est difficile d'évaluer l'impact d'un tel décalage, car nous ne disposons pas d'une méthode d'ajustement d'ellipsoïde adaptée.

Pour conclure, nous avons vu que si l'approche d'ellipsoïdes pouvait apporter certains avantages, l'approche de quadrique généralisée ne permet pas d'obtenir l'invariance affine et peut facilement produire des résultats contre-intuitifs, comme illustré figure 27.b à cause de la présence de plusieurs nappes. Malheureusement, nous ne disposons pas de méthode satisfaisante pour ajuster des ellipsoïdes. Ainsi, de notre point de vu, l'utilisation de quadriques généralisées ou ellipsoïdes au sein de notre approche LS^3, ou de la reconstruction par MLS en générale, reste un problème ouvert.