Université de Montréal

Génération et édition de textures géométriques représentées par des ensembles de points

par

François Duranleau

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures et postdoctorales
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique

décembre 2008

Université de Montréal

Faculté des études supérieures

Cette thèse intitulée :

Génération et édition de textures géométriques représentées par des ensembles de points

présentée par :

François Duranleau

a été évaluée par un jury composé des personnes suivantes :

Neil F. Stewart
président-rapporteur

Pierre Poulin
directeur de recherche

Victor Ostromoukhov
membre du jury

Mathias Paulin
examinateur externe

Abraham Broer
représentant du doyen de la FAS

Résumé

Avec l'augmentation continuelle de la puissance des ordinateurs, nous sommes en mesure de produire des images par ordinateur comportant des détails de plus en plus riches. Ces détails des objets virtuels 3D sont souvent représentés sous forme de textures colorées et de géométrie. Or, il s'avère qu'avec les méthodes répandues de modélisation géométrique (modeleurs, numérisation 3D, modèles procéduraux, etc.), la tâche de modélisation de ces détails, autant pour la création que l'édition, peut représenter un investissement gigantesque en temps et en talent.

Depuis la dernière décennie, les représentations géométriques par points ont gagné en popularité pour leur simplicité et leur efficacité de représentation pour des détails géométriques fins. D'un autre côté, la génération de textures à partir d'exemples a aussi connu beaucoup de progrès au sein de la communauté graphique. Il s'agit de produire une nouvelle texture de taille arbitraire possédant les mêmes caractéristiques qu'un exemple donné. En alliant ces deux concepts, nous présentons dans un premier volet de cette thèse un nouvel algorithme de génération de textures géométriques représentées par des ensembles de points. Cet algorithme se base sur les méthodes de génération par *patch* de textures colorées avec l'ajout de déformations locales pour mieux aligner les caractéristiques des *patches* jointes ensemble lors de la génération. Nous présentons des résultats expérimentaux qui montrent la généralité et l'efficacité de cette approche.

Il est important aussi de pouvoir modifier à différents niveaux les objets détaillés, tout en préservant les structures des détails. Une approche classique utilise une représentation multirésolution. Ainsi, le second volet de cette thèse traite d'une représentation multirésolution pour des surfaces constituées d'ensembles de points. À chaque niveau de résolution correspond une surface plus lisse et échantillonnée par moins de points que la surface du niveau de résolution précédent, *i.e.* plus détaillé. De plus, un ensemble de détails relatifs à cette surface est conservé. Comparativement aux travaux antérieurs, notre représentation est généralement plus compacte et efficace, et de robustesse comparable et parfois meilleure.

Mots clefs :

Texture géométrique, surface représentée par ensembles de points, génération de textures, surface multirésolution, édition multirésolution.

Abstract

Computers are continuously getting more powerful, and thus we are able to produce computer generated images that are very rich in details. The details present in 3D virtual objects are often encoded as color or geometry textures. Considering the wide spectrum of geometric modeling methods (modeling software, 3D scanning, procedural modeling, etc.), the task of modeling these details, including both initial creation and later edition, can require a huge investment in time and talent.

In the past decade, point-based geometry representations have gained popularity due to their simplicity and their representation efficiency for geometric details. Texture synthesis from examples, involving the generation of a new texture of arbitrary size that shares the same features as a given example, has also much progressed in the graphics community. By combining these two concepts, we present in the first part of this thesis a new texture synthesis algorithm for geometry textures represented by point-set surfaces. The algorithm is based on color texture synthesis methods proceeding by patch, with local warping to improve feature alignment of patches to be joined during synthesis. Our experimental results show the generality and efficiency of our algorithm.

It is important to be able to edit detailed objects at different resolution levels while preserving detail structures. A classic approach uses a multiresolution representation. The second part of this thesis exploits multiresolution point-set surfaces. Each resolution level corresponds to a smoother surface represented by fewer points than the surface at the previous more detailed resolution level. In addition, a set of details relative to the surface at this resolution level is stored. Compared to previous work, our representation is generally more compact and efficient, and its robustness is comparable and sometimes better.

Keywords :

Geometry texture, point-set surface, texture synthesis, multiresolution surface, multiresolution editing.

Table des matières

Table des figures

Liste des tableaux

Liste des remarques

Remerciements

Il n'y a pas que le travail personnel qui nous aide à passer au travers de toutes ces années d'études doctorales. Il y a tant de gens que nous rencontrons, avec qui nous échangeons, et de qui nous soutirons des nouvelles idées, du support moral, etc.

D'abord, j'ai été choyé d'avoir eu la compagnie de tous les gens sympathiques qui ont aussi fait leurs études graduées au Laboratoire d'informatique graphique de l'Université de Montréal. Dans aucun ordre particulier, sinon peut-être approximativement chronologique, je remercie Emric Epstein, Martin Granger-Piché, Jean-François St-Amour, Martin Côté, Pierre-Marc Jodoin, Guillaume Poirier, Charles Donohue, Jean-François Dufort, Yannick Simard, Simon Clavet, Nicolas Bergeron, Malika Zidani Boumedien, Philippe Beaudoin, Fabrice Rousselle, Luc Leblanc, Romain Pacanowski, Simon Bouvier Zappa, Marie-Élise Cordeau, Michelle Laprade, Mathieu "Drummondville" Gauthier, Yann Rousseau, Frédérik Rozon, Chang Jiang Hao et Wu Zhe. De tous ces gens, je dois un remerciement particulier d'abord à Simon Bouvier Zappa et Yann Rousseau, pour leur aide précieuse dans la production d'exemples pour mes résultats, puis à Charles Donohue, pour m'avoir trouvé une place au sein de l'industrie du jeu vidéo chez Gameloft. Mais surtout, je dois remercier grandement Philippe Beaudoin, car sans lui pour me relever le moral lorsque les travaux ne se portaient pas bien, et sans nos discussions enrichissantes, cette thèse ne serait pas ce qu'elle est.

Je dois aussi remercier les visiteurs qui sont venus au laboratoire pendant la période de mon long séjour : ils viennent ouvrir nos fenêtres pour que nous puissions voir le monde extérieur. Je remercie nos visiteurs suisses Jonathan Maïm et Barbara Yersin (grâce à qui nous avons pu déguster une authentique fondue au fromage suisse) et nos visiteurs français Mathieu Nesme, David Vanderhaeghe et Vincent Nivoliers. Je remercie aussi le professeur Christopher Healey, qui a été des nôtres quelques mois d'été et qui a accepté de réviser un de mes articles en soumission. Merci aussi à Jocelyn Houle et Mathieu Ouimet, anciens du laboratoire qui venaient

donner un coup de main à certains parmi nous. Je remercie Mathieu Ouimet en particulier pour m'avoir fait découvrir *Boost* et les plaisirs de la programmation moderne en C++.

Je ne peux pas non plus négliger de remercier nos chers professeurs du laboratoire. Je remercie Victor Ostromoukhov pour m'avoir initié au rendu non photoréaliste, au rendu demi-ton et à l'étude de la perception humaine. Si je n'avais pas connu Victor, j'aurais sans doute continué de regarder ces domaines comme des curiosités, mais de très loin. Merci à Neil Stewart, pour qui il a été très agréable de travailler en tant qu'auxiliaire d'enseignement pour son cours de structures de données et grâce à qui j'ai pu appronfondir ma compréhension des fondements mathématiques des surfaces de subdivision. Mais surtout, je dois remercier mon directeur de recherche Pierre Poulin. C'est en bonne partie grâce à lui que nous bénéficions d'une superbe ambiance de travail au laboratoire. Mais surtout, avec Pierre, j'ai toujours senti que j'ai été supporté jusqu'au bout, tout en ayant carte blanche sur mes choix et décisions.

Je réserve mes plus grands remerciements à mon épouse bien aimée Jiang Di. Di a aussi été une compagne de laboratoire, sous la supervision de Neil Stewart. Puis nous sommes sortis ensemble et un peu plus tard, nous nous sommes mariés. Di a été ma principale source de motivation pour aller de l'avant, et je ne saurais où j'en serais rendu maintenant si elle n'avait pas été là pour donner un sens à ma vie. Je pouvais toujours compter sur elle lorsque survenaient des moments plus difficiles, y compris pendant les sprints intenses avant les échéances de soumissions d'articles. Je me comptais chanceux aussi d'avoir quelqu'un avec qui je pouvais discuter et échanger lorsque je travaillais à la maison.

Enfin, je remercie mes parents pour leur grand support et leur compréhension de notre situation de pauvres étudiants gradués. Je dois aussi remercier mes supérieurs chez Gameloft, en particulier Patrick Allaire, pour leur compréhension du travail supplémentaire que je devais faire en dehors des heures de bureau pour terminer cette thèse. Et finalement, je remercie tous les membres de mon jury, dont Mathias Paulin pour avoir accepté de venir depuis Toulouse pour ma soutenance.

Les travaux de cette thèse ont été supportés par des subventions du CRNSG et du FQRNT.

Terminologie

Certains termes ne sont pas toujours employés avec le même sens parmi tous les domaines d'études, et parfois même à l'intérieur d'un même domaine. Afin de tenter de réduire l'ambiguïté sur leur usage dans cette thèse, nous donnons ci-après une liste de certains termes et nos choix sur le sens que nous employons.

Analyse *vs.* décomposition Le terme *analyse* est utilisé en multirésolution, souvent pour parler de décomposition et parfois pour l'ensemble du principe. Ce terme est aussi utilisé dans d'autres contextes. Le terme *décomposition* est également utilisé pour prendre un niveau de résolution donné et produire un niveau plus *grossier* ou *lisse* avec son vecteur de détails. Nous préférons l'usage de ce dernier terme plutôt qu'analyse, car nous le jugeons plus précis dans son contexte.

Ensemble de points (*point set*) *vs.* nuage de points Nous préférons utiliser *ensemble de points* en général, car le terme *nuage* a intuitivement une connotation plutôt volumétrique, alors qu'*ensemble* est neutre.

Distance signée Lorsque nous parlons de distance, nous sous-entendons la distance euclidienne. Lorsque la distance entre un point et une surface est calculée, il s'agit de prendre la distance entre ce point et le point le plus proche de la surface. Si la surface est orientable, cette distance peut être *signée* en fonction du côté où se situe le point. Elle est positive si le point et la normale au point le plus près de la surface pointe vers lui, négative autrement.

Génération (*synthesis*) Le terme *génération* (*synthesis*) est employé, entre autres, en génération de textures et en multirésolution. Comme cette thèse traite des deux sujets, nous le réservons pour le premier cas, et nous préférons l'usage du terme *reconstruction* dans le second cas, qui est aussi employé dans la littérature dans ce sens. De plus, nous avons préféré l'usage du terme *génération* plutôt que *synthèse* car, bien que ce dernier soit

employé en chimie pour la composition de corps avec ses éléments, il n'est pas clair pour nous à quel point il est approprié en français dans le sens désiré pour cette thèse, alors que le terme *génération* est certainement acceptable.

Géométrie Le terme *géométrie* désigne principalement une branche des mathématiques concernant la mesure, les propriétés et les relations entre points, lignes, angles, surfaces et solides. Cependant, ce terme peut aussi être employé pour désigner la forme d'un objet[1]. Comme cette thèse s'intéresse principalement à la forme des objets, nous employons ce terme dans le second sens.

Modèle *vs.* objet Dans le "langage populaire" en infographie, le terme *modèle* est souvent employé pour désigner un objet virtuel. Bien que ce sens soit officiellement reconnu[2], nous préférons plutôt employer le terme *objet* (ou parfois *objet virtuel*) et réserver l'usage de *modèle* dans le sens mathématique.

Multirésolution, multiéchelle et multiniveau Le terme *multirésolution* est utilisé à bien des sauces en infographie, mais il sous-entend toujours la notion de niveaux de détails. Il peut s'agir d'une pyramide d'images, d'un groupe de surfaces représentant le même objet à différents niveaux de détails, d'une décomposition en ondelettes, etc. Dans cette thèse, le terme *multirésolution* est employé pour décrire toute représentation où les niveaux de résolution sont définis relativement entre eux. Un autre terme généralement synonyme est *multiéchelle*. Suivant le titre de certaines publications (mentionnées au chapitre 5), nous faisons usage de ce terme pour un cas particulier des représentations multirésolutions, soit celui où la quantité d'information requise pour chaque niveau de résolution est invariant. Dans tout autre cas, nous employons le terme *multiniveau*, et ce, seulement pour départager les autres approches.

Patch Il s'agit d'un terme utilisé pour décrire un morceau, typiquement rectangulaire, d'une texture ou d'une surface. À défaut d'avoir trouvé un terme français satisfaisant, nous avons opté pour conserver ce terme dont le sens est largement compris et sous-entendu dans la communauté graphique.

Réduire *vs.* sous-échantillonner *vs.* simplifier Soit un ensemble d'éléments E, si nous voulons produire un ensemble d'éléments E' à partir de E tel que $|E'| < |E|$, mais sans

[1] *geometry — Definition from the Merriam-Webster Online Dictionary.* `http://www.m-w.com/dictionary/geometry`

[2] Selon le *Grand dictionnaire terminologique* de l'Office de la langue française du Québec. `http://www.granddictionnaire.com/`

nécessairement que $E' \subset E$, nous préférons parler de *simplifier* E plus que de *réduire* ou *sous-échantillonner* E. Le terme *réduire* peut aussi avoir le sens géométrique de mise à l'échelle avec un facteur inférieur à un, en particulier si E décrit la géométrie d'un objet. Le terme *sous-échantillonner* a souvent la connotation que $E' \subset E$. Le terme *simplifier* est plus général et même souvent employé dans le sens décrit ici.

Représentation Nous employons ce terme pour décrire le modèle (mathématique) représentant une entité géométrique, très souvent la surface d'un objet.

Topologie En modélisation géométrique, le terme *topologie* est employé à la fois pour désigner la variété (*manifold*) et aussi pour désigner la connectivité dans un maillage. Dans cette thèse, nous utilisons ce terme lorsque nous parlons de variété et *connectivité* lorsque nous parlons de l'information de connectivité d'un maillage.

Finalement, le vocabulaire d'origine du domaine de cette thèse provient généralement de l'anglais et, souvent, il n'y a pas de consensus sur la traduction exacte en français. Dans ces cas, lorsqu'une telle expression survient la première fois dans le texte, nous la faisons suivre de son expression originale anglaise entre parenthèse.

Notation

Comme la terminologie, la notation n'est pas toujours consistante au sein d'un domaine. Nous effectuons ci-après un sommaire des règles de notation que nous avons employées dans cette thèse.

minuscule ou grecque minuscule	Un scalaire ou une fonction scalaire.
gras	Un vecteur, un nuplet ou une fonction vectorielle.

Particularité pour les règles ci-avant : soit une fonction $f : D \mapsto \mathbb{R}$, où D est un domaine quelconque, pour $S \subset D$, alors $f(S) \equiv \{f(s) \mid s \in S\}$ [3]. Il en va de même pour les fonctions vectorielles.

MAJUSCULE ou grecque majuscule	Ensemble ou fonction produisant un ensemble ou une entité géométrique. Pour les ensembles, la lettre minuscule correspondante (en gras si vectorielle) est habituellement employée pour désigner un de ses éléments, mais pas toujours.

Exceptions à la règle précédente : Δ, qui représente l'aire d'un triangle, et L, un nombre naturel utilisé pour noter le nombre maximum de niveaux de résolution dans une décomposition multirésolution.

GRAS	Une matrice ou fonction matricielle.
CALLIGRAPHIQUE	Entité géométrique (par exemple, un plan, une boîte, un graphe, etc.).

[3] Selon notre notation, ceci suppose que S est un ensemble de scalaires. Bien entendu, ceci se généralise aussi pour les fonctions à plusieurs variables, *i.e.* $f(S) \equiv \{f(s) \mid s \in S\}$.

minuscule ou chiffre en indice	Un nombre naturel étiquetant un élément d'un ensemble. À noter l'usage des nombres naturels : tout indice commence donc à un, et non zéro.

Exceptions à la règle précédente : x en indice marque une variable dépendante de x dans un problème de minimisation.

lettre en caractère roman en indice	Étiquette supplémentaire au symbole joint.
majuscule, grecque majuscule ou calligraphie indice	Indique une dépendance sur l'ensemble identifié par la lettre.
minuscule ou chiffre entre parenthèses en exposant	Compteur d'itération.
expression de l entre crochets en exposant	Niveau de résolution.
barre ($\bar{}$) au-dessus d'une expression	Indique une moyenne quelconque (centroïde, bissectrice, etc.), où un ensemble de moyennes quelconques si en lien avec un ensemble d'ensembles. Exception : une barre au-dessus de deux points représente un segment (un exemple est donné dans le prochain tableau).
tilde ($\tilde{}$) au-dessus d'un autre symbole	Indique habituellement une approximation ou un résultat intermédiaire.
brève ($\breve{}$) au-dessus d'un autre symbole	Indique une valeur associée à un agrégat ou un calcul produisant un ensemble d'agrégats.
caron ($\check{}$) au-dessus d'un autre symbole	Indique la projection sur une entité géométrique, habituellement un plan.
prime ($'$) adossé à un autre symbole	Indique le résultat d'un transformation quelconque.
astérisque ($*$) adossé à un autre symbole	Indique le résultat d'une sélection quelconque.

Ci-après, nous énumérons plus précisément la notation et le choix de symbole courant pour dénoter une entité, en commençant par des généralités et en allant au plus spécifique.

$\mathbb{N}$	Ensemble des nombres naturels, *i.e.* $\{1, 2, 3, \ldots\}$.
$\mathbb{Z}$	Ensemble des nombres entiers.
$\mathbb{R}$	Ensemble des nombres réels.
$\mathbb{R}^+$	Ensemble des nombres réels positifs, *i.e.* $\{x \in \mathbb{R} \mid x \geq 0\}$.
$\mathbb{S}^2$	Sphère unitaire de $\mathbb{R}^3$, *i.e.* $\left\{ \boldsymbol{x} \in \mathbb{R}^3 \mid \|\boldsymbol{x}\| = 1 \right\}$.
$\mathbb{Z}[a, b]$	Intervalle d'entiers de a à b inclusivement, *i.e.* $\{a, a+1, \ldots, b\}$, où $a \leq b \in \mathbb{Z}$.
π	Le nombre π, *i.e.* $3.14159\ldots$
e	Le nombre e, *i.e.* $2.71828\ldots$
$\boldsymbol{x}$	Un élément quelconque de $\mathbb{R}^3$, également un vecteur colonne.
$[x_1, \ldots, x_d]$	Vecteur ligne de dimension d, $x_i \in \mathbb{R}\ \forall i \in \mathbb{Z}[1, d]$. Parfois la notation présente un mélange de scalaires et de vecteurs lignes ; il faut alors interpréter les vecteurs comme un regroupement. Par exemple, soit $\boldsymbol{x} \in \mathbb{R}^3$, $[0, \boldsymbol{x}^\mathsf{T}, 1]$ dénote un vecteur ligne de dimension 5.
$\boldsymbol{e}_j$	j^e vecteur de la base canonique de $\mathbb{R}^d$, $d \in \mathbb{N}$ et $j \in \mathbb{Z}[1, d]$, *i.e.* un vecteur tel que sa j^e composante est 1 et les autres valent zéro.
$\boldsymbol{n}$	Une direction, *i.e.* un élément de $\mathbb{S}^2$.
i, j, k	Indice, élément de $\mathbb{N}$.
∇f	Gradient d'une fonction scalaire f à plusieurs variables.
$\lvert x \rvert$	Valeur absolue du scalaire x.
$\lvert X \rvert$	Cardinalité d'un ensemble X fini.
$\wp(X)$	Ensemble puissance de X, *i.e.* $\wp(X) = \{Y \mid Y \subseteq X\}$.
$\|\boldsymbol{x}\|$	Norme euclidienne de $\boldsymbol{x}$.
$\boldsymbol{x}^\mathsf{T}$	Transposée de $\boldsymbol{x}$, où $\boldsymbol{x}$ est un vecteur colonne.
$\boldsymbol{x}^\mathsf{T}\boldsymbol{y}$	Produit scalaire entre deux vecteurs $\boldsymbol{x}$ et $\boldsymbol{y}$ exprimé sous forme de produit matriciel entre la transposée de $\boldsymbol{x}$ (un vecteur ligne) et $\boldsymbol{y}$ (un vecteur colonne).
$\boldsymbol{x} \times \boldsymbol{y}$	Produit vectoriel entre deux vecteurs $\boldsymbol{x}$ et $\boldsymbol{y}$.
$X \times Y$	Produit cartésien de deux ensembles X et Y.
$\mathbf{1}_n$	Matrice identité d'ordre n.
$\mathbf{0}_{n \times m}$	Matrice nulle de dimension $n \times m$.

$\bar{\boldsymbol{n}}$	Direction bissectrice.
λ_i	i^{e} plus petite valeur propre d'une matrice carrée quelconque.
P, Q	Ensemble de points.
$\boldsymbol{p}, \boldsymbol{q}$	Point d'un ensemble de points P ou Q.
$\boldsymbol{p}_i$	i^{e} point d'un ensemble de points P.
$\boldsymbol{n}_i$	Normale associée au i^{e} point d'un ensemble de points donné.
$\bar{\boldsymbol{p}}$	Centroïde d'un ensemble de points P.
$\check{P}$	Ensemble des points de P projetés sur un plan quelconque.
$\check{\boldsymbol{p}}_i$	Projection de $\boldsymbol{p}_i \in P$ sur un plan quelconque.
C	Ensemble d'agrégats de points.
C_i	i^{e} agrégat d'un ensemble d'agrégats de points C.
$\bar{\boldsymbol{c}}_i$	Centroïde de l'agrégat de points C_i.
$\bar{C}$	Ensemble des centroïdes des agrégats de points de C.
$\check{\boldsymbol{n}}_i$	Normale de l'agrégat C_i.
$\overline{\boldsymbol{xy}}$	Segment allant du point $\boldsymbol{x}$ au point $\boldsymbol{y}$.
$\angle \boldsymbol{xyz}$	L'angle entre les vecteurs $\boldsymbol{x} - \boldsymbol{y}$ et $\boldsymbol{z} - \boldsymbol{y}$.
$\triangle \boldsymbol{xyz}$	Triangle formé par les sommets $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$.
$\mathcal{N}_P(\boldsymbol{x})$	Voisinage dans P autour d'un point $\boldsymbol{x}$. Le type exact de voisinage est indiqué par une marque ou un symbole supplémentaire. Les types sont énumérés ci-après.
$\mathcal{N}^h$	Dénote un voisinage euclidien de rayon h.
$\mathcal{N}^k$	Dénote un voisinage des k plus près voisins (voisinage-k).
$\mathcal{N}^{kh}$	Dénote un voisinage-kh.
$\ddot{\mathcal{N}}$	Dénote un voisinage symétrique.
$\mathcal{N}^\alpha$	Dénote un voisinage-α.
$\mathring{\mathcal{N}}$	Dénote un voisinage-*BSP*.
$\theta(d)$	Fonction de pondération réelle positive monotone décroissante selon une distance $d \in \mathbb{R}^+$.
h	Support de $\theta(d)$ ou taille de caractéristiques (*feature size*).
$h_P(\boldsymbol{x})$	Support de $\theta(d)$ ou taille de caractéristiques (*feature size*) locale de l'ensemble de points P autour du point $\boldsymbol{x}$.
$\boldsymbol{n}_P(\boldsymbol{x})$	Estimation de la normale en un point $\boldsymbol{x}$ sur l'ensemble de points P.

$\boldsymbol{a}_P(\boldsymbol{x})$	Centroïde des points de P pondéré en fonction de la distance de chaque point et $\boldsymbol{x}$.
$\sigma_P(\boldsymbol{x})$	Estimation de la variation des points P autour d'un point $\boldsymbol{x}$.
$\mathcal{S}_P$	Surface implicitement définie par l'ensemble de points P.
$\boldsymbol{\psi}_P(\boldsymbol{x})$	Projection d'un point $\boldsymbol{x}$ sur $\mathcal{S}_P$.
ε	Fonction d'énergie quelconque.
δ	Une distance signée entre un point et une surface.
$\mathcal{H}$	Un plan.
$\mathcal{T}$	Un triangle ou une texture géométrique, selon le contexte (triangle au chapitre 6 et texture géométrique au chapitre 4).
$\mathcal{P}$	Une *patch* d'une texture géométrique.
$\mathcal{E}$	Un ellipsoïde ou une région périphérique d'une texture géométrique, selon le contexte (ellipsoïde au chapitre 2 et région au chapitre 4).
$\mathcal{B}$	Une boîte englobante alignée sur les axes.
$\mathcal{B}_\mathcal{X}$	Boîte englobante alignée sur les axes de l'entité géométrique $\mathcal{X}$.
$\Delta_\mathcal{T}$	Aire du triangle $\mathcal{T}$.
$\mathcal{G}$	Un graphe.
V	Ensemble de sommets d'un graphe ou d'un maillage.
E	Ensemble d'arêtes d'un graphe.
$\mathcal{M}$	Un maillage.
Ψ	Un ensemble de candidats.
$\delta_\mathcal{S}(\boldsymbol{x})$	Champ de distance de la surface $\mathcal{S}$.
$\delta_P(\boldsymbol{x})$	Champ de distance de la surface $\mathcal{S}_P$.
Θ	Ensembles de paires de points.
$W_\Theta(\boldsymbol{x})$	Une fonction de déformation de l'espace par spline en plaques minces suivant les contraintes Θ.
$P_\mathcal{X}$	Ensemble de points de la texture géométrique $\mathcal{X}$ ou d'une région ou d'une *patch* $\mathcal{X}$ d'une texture géométrique.
$\mathcal{M}^{[l]}$	Un maillage pour un niveau de résolution l.
$V^{[l]}$	Ensemble des sommets d'un maillage pour un niveau de résolution l.
$\boldsymbol{V}^{[l]}$	Représentation matricielle de l'ensemble des sommets d'un maillage pour un niveau de résolution l.

$\Lambda(P)$	Opérateur de simplification d'un ensemble de points P.
$\breve{\Lambda}(P)$	Opérateur de simplification d'un ensemble de points P produisant un ensemble d'agrégats plutôt qu'un autre ensemble de points.
$\Phi(P)$	Opérateur de lissage d'un ensemble de points P.
$P^{[l]}$	Ensemble de points pour un niveau de résolution l.
$\boldsymbol{p}_i^{[l]}$	i^{e} point de $P^{[l]}$.
$D^{[l]}$	Ensemble d'information de détails géométriques pour un niveau de résolution l.
$\boldsymbol{d}_i^{[l]}$	i^{e} élément de $D^{[l]}$.

Chapitre 1

Introduction

Depuis longtemps l'homme cherche à reproduire la réalité ou son imaginaire dans son art. En dehors des qualités figuratives, les éléments parmi les plus importants dans la richesse d'une œuvre d'art sont tous les petits détails. Un tel exemple se trouve à la figure 1.1. Deux artistes pourraient produire chacun une statuette d'un dragon en jade, mais celle qui attirera plus probablement notre attention est celle qui comportera les plus fins détails sculptés. Ce souci du détail est d'autant plus important lorsqu'il est question de reproduire la réalité.

Depuis l'avènement de l'ordinateur et de l'infographie, un nouveau médium pour créer et présenter les créations artistiques s'offre aux artistes. Non seulement bénéficient-ils d'une vaste gamme d'outils *interactifs*, mais l'aspect dynamique de ces outils combiné à la puissance de calcul des ordinateurs leur permet de produire des scènes virtuelles très complexes. Il suffit de penser aux derniers films d'animation 3D pour le constater. Mais encore une fois, l'importance des détails est toujours au rendez-vous.

1.1 Détails géométriques

Un type de détail particulier mais important est celui correspondant aux détails se retrouvant à la surface des objets, ce qui est souvent appelé texture. Il peut s'agir de couleurs, de propriétés de réflectance, mais aussi de détails géométriques. Dans ce dernier cas, nous parlons de *texture géométrique*. Ce type de texture correspond à ce qui définit localement, *i.e.* à proximité de la surface, la forme d'un objet. Si nous regardons à nouveau la figure 1.1, ce dragon de jade est riche en détails géométriques : les formes des pattes, les encavures des écailles, la crinière

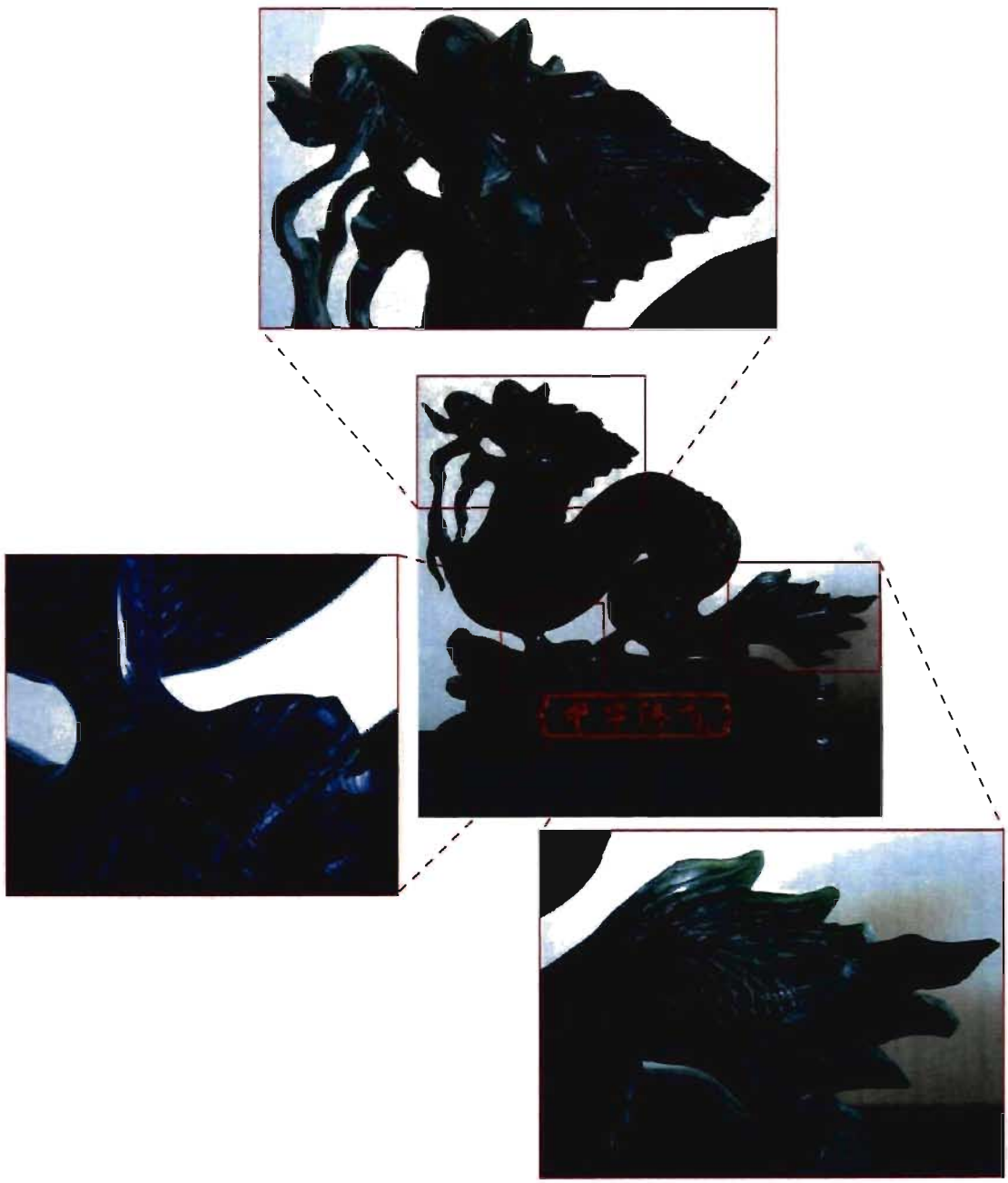


FIG. 1.1 – Détails de la sculpture d'un dragon de jade.

derrière la tête, etc. Pour créer un objet 3D virtuel de qualité, un artiste doit se soucier de ce genre de détails.

1.2 Création et édition

En terme de modélisation géométrique, la création et l'édition d'objets détaillés n'est pas une tâche simple. Bien que les logiciels classiques de modélisation 3D, tels que *Blender* [Blender Foundation], *Maya* [Autodesk, Inc.b], *Softimage|XSI* [Avid Technology, Inc.] et *3ds Max* [Autodesk, Inc.a], soient puissants, ce qui entraîne une certaine complexité, le travail demeure fastidieux et demande beaucoup de temps. Il existe des méthodes dites procédurales pouvant produire beaucoup de détails de qualité, par exemple les systèmes-L (*L-systems*) [Lindenmayer 1968], qui peuvent être utilisés pour produire des plantes [Prusinkiewicz et Lindenmayer 1990] ou même des écosystèmes entiers [Deussen *et al.* 1998]. Mais ces méthodes sont soit spécialisées pour un type d'objets, soit peu intuitives pour produire les résultats désirés (c'est le cas des systèmes-L).

Lorsque la tâche est de reproduire un objet réel, la numérisation 3D permet d'extraire une version virtuelle à partir de divers instruments de mesures. Cependant, le niveau de bruit dans le résultat peut nuire à la qualité de reproduction des détails géométriques, sans compter qu'il n'est parfois pas possible de bien les mesurer, en particulier s'il y a de l'occultation entre eux. De plus, la numérisation 3D ne peut que mesurer la réalité. Si l'objet que l'artiste veut produire n'existe pas, il doit se rabattre sur les autres approches.

Une approche plus récente est la génération à partir d'exemples. Elle est inspirée de la même approche pour la génération de textures colorées (*color textures*) représentées pour la plupart par des images à trame (*raster*) 2D. Le but est de produire une nouvelle texture qui ressemble à l'exemple fourni, sans en être une réplique ou un pavage. Cette approche a reçu beaucoup d'attention depuis la dernière décennie et la qualité des résultats actuels la rend une solution viable pour modéliser un grand nombre de textures. Nous disons un grand nombre car ces méthodes fonctionnent par échantillonnage stochastique généralement sans considération sémantique du contenu de la texture[1].

En terme de texture géométrique, le même principe s'applique. Cependant, les extensions directes de l'approche 2D, qui incluent la génération de cartes de déplacement (*displacement*

[1]Par sémantique, il est question de formes ou de structures dans la texture que l'on peut reconnaître à un niveau plus abstrait. Un exemple riche en sémantique est un portrait. Cependant, il est généralement entendu que ce type d'images n'est pas considéré comme une texture.

maps) [Wei et Levoy 2001, Ying *et al.* 2001, Lai *et al.* 2005, Nguyen *et al.* 2005], sont limitatives quant à la complexité des textures géométriques qu'elles peuvent générer. Par exemple, elles ne peuvent pas générer des textures avec des composantes non connexes fermées. Cette même limitation est présente pour plusieurs autres méthodes de génération à partir d'exemples [Sharf *et al.* 2004, Park *et al.* 2005, Zelinka et Garland 2006]. Seules les méthodes volumétriques [Bhat *et al.* 2004b, Lagae *et al.* 2005] et la méthode de génération de maillages de Zhou *et al.* [2006] ont la flexibilité requise pour générer des textures géométriques de complexité quasi arbitraire. Par contre, dans le premier cas, la quantité de mémoire requise et les temps de calcul peuvent être considérablement élevés, et dans l'autre cas, la gestion des structures de connectivité est passablement complexe.

Un autre aspect important dans le processus de création d'un objet est de pouvoir modifier ou éditer l'objet à toute étape. Ainsi, si un artiste désire changer la forme d'un objet, il devrait pouvoir le faire avant autant qu'après avoir généré tous les détails géométriques de l'objet. Le problème principal avec les opérations d'édition sur des objets détaillés est d'arriver à les appliquer de sorte que les détails géométriques soient préservés, ou qu'ils soient changés de façon naturelle suivant l'opération. Une gamme de méthodes existent pour parvenir à cette fin [Botsch et Sorkine 2008]. Alors que la plupart de ces méthodes sont axées seulement sur la déformation préservant les détails, les représentations multirésolutions offrent aussi la possibilité d'éditer les détails eux-mêmes, ce qui les rend plus intéressantes dans un contexte d'édition plus général. Une représentation multirésolution correspond essentiellement à décomposer la surface d'un objet en une série de surfaces exhibant de moins en moins de détails, où chaque surface plus détaillée est représentée de façon relative à la surface moins détaillée suivante dans la série. Les surfaces détaillées peuvent alors être reconstruites par "addition" successive de la surface grossière avec les surfaces relatives. Les opérations d'édition peuvent alors être effectuées en modifiant la surface grossière pour obtenir des déformations de sorte à ce que les détails géométriques soient changés en conséquence, ou en jouant au niveau des surfaces relatives pour obtenir des effets sur les détails mêmes, par exemple, pour accentuer ou lisser certains détails à un niveau de résolution donné.

1.3 Représentations géométriques

Peu importe la méthode de création ou d'édition, la géométrie des objets 3D doit être représentée d'une certaine façon. La géométrie d'un objet correspond essentiellement à ce qui

définit sa forme. Parmi les représentations les plus fréquemment rencontrées, il y a les maillages, incluant les surfaces de subdivision et les maillages de surfaces paramétriques autant que les maillages polygonaux, les surfaces implicites et les représentations volumétriques.

Les maillages sont composés de faces, planaires ou courbes, liées entre voisines. Il s'agit de la représentation à l'usage le plus répandu, en particulier les maillages polygonaux. Ces derniers peuvent représenter des surfaces de complexité arbitraire et la distribution des faces peut aisément s'adapter à la finesse des détails géométriques. Ce n'est cependant pas le cas des représentations volumétriques. Ces dernières sont habituellement implémentées par une grille régulière 3D pour laquelle chaque cellule contient une information telle que l'appartenance à l'objet et la plus petite distance à sa surface. À moins de passer par une méthode adaptative, la résolution de la grille doit être grande pour bien échantillonner les petits détails géométriques.

Les surfaces implicites, quant à elles, sont définies comme l'ensemble des zéros d'un champ scalaire. Pour représenter des formes arbitraires, ce champ scalaire doit être exprimé en fonction d'un jeu de données et non de façon purement analytique. Il est possible de définir une surface implicite par l'intermédiaire de données volumétriques, mais il existe d'autres façons qui, comme pour les maillages, permettent de concentrer l'information à proximité de la surface seulement, notamment par l'usage de RBFs (*radial basis functions* ou fonctions à base radiale) [Savchenko *et al.* 1995]. Cependant, la quantité de données requise pour représenter des détails fins dépasse quand même celle requise pour les maillages, sans compter qu'il est difficile de représenter des détails très abrupts.

Depuis la dernière décennie, une autre approche pour représenter la géométrie d'un objet a connu un gain de popularité : la représentation par points. L'idée d'utiliser les points comme primitive, sans doute la plus simple qui soit, a été à l'origine apportée pour le rendu d'objets complexes, car lorsque le nombre de sommets dans un maillage devient grand, le poids relatif des structures requises pour maintenir l'information de connectivité est aussi grand. En modélisation géométrique, la source principale de points est la numérisation 3D. Mais traditionnellement, une étape de reconstruction est nécessaire pour les transformer en maillages, par exemple, ce qui amène déjà un intérêt pour vouloir traiter directement les points non seulement comme primitive de rendu mais aussi comme primitive de modélisation géométrique. De plus, l'absence d'information de connectivité explicite peut rendre certaines opérations simples et efficaces, en particulier tout ce qui concerne le découpage, la jonction et le rééchantillonnage. Ces opérations sont d'ailleurs souvent requises lorsqu'il est question de générer ou éditer les

détails géométriques, ce qui est particulièrement le cas pour les approches de génération à partir d'exemples.

1.4 Contributions

Cette thèse apporte deux contributions principales. D'abord, nous avons souligné les avantages des représentations par points pour des opérations de découpage et de jonction. De plus, nous constatons le succès des méthodes de génération de textures colorées à partir d'exemples ainsi que le côté prometteur des méthodes similaires de génération de géométrie. Nous proposons alors un nouvel algorithme de génération de textures géométriques représentées par des ensembles de points. Nous espérons ainsi obtenir une méthode qui n'a pas la surcharge de mémoire des approches volumétriques et qui est plus efficace que les méthodes de génération de maillages.

Puis, comme nous l'avons mentionné précédemment, une autre problématique reliée aux détails géométriques est l'édition d'objets possédant de tels détails. Ces détails pourraient provenir, par exemple, d'une génération comme celle de notre algorithme précédent. Ainsi, il est possible que la représentation soit toujours par points. Il y a alors un intérêt pour pouvoir effectuer les opérations d'édition en restant dans cette représentation pour éviter les conversions possiblement coûteuses mais surtout enclines aux erreurs. Nous avons mentionné comment les représentations multirésolutions permettent l'édition d'objets détaillés, mais peu de travaux ont touché ces représentations pour des ensembles de points ; les méthodes existantes limitent les opérations d'édition possibles ou pourraient être améliorées. Ainsi, la seconde contribution de cette thèse concerne une nouvelle représentation multirésolution pour des ensembles de points. La représentation que nous proposons offre toute la puissance requise de la multirésolution et tend à consommer moins d'espace mémoire que les représentations antécédentes.

1.5 Plan de la thèse

Puisque la représentation par points est fondamentale pour tout ce qui est présenté dans cette thèse, nous débutons par un survol de la modélisation géométrique par points au chapitre 2. Ensuite, chaque contribution de la thèse est présentée en paire de chapitres : le premier décrit les travaux antérieurs plus spécifiquement reliés au sujet et le second décrit notre contribution. Ainsi, nous présentons au chapitre 3 les travaux antérieurs en lien avec la génération de textures

et de géométrie et au chapitre 4 se trouve la présentation de notre algorithme de génération de textures géométriques représentées par ensemble de points. Puis, les travaux antérieurs sur les représentations multirésolutions sont couverts au chapitre 5 et nous présentons notre représentation multirésolution pour des surfaces représentées par points au chapitre 6. Enfin, nous concluons et discutons de travaux futurs généraux au chapitre 7.

Chapitre 2

Modélisation géométrique par points

La *modélisation géométrique* est une partie de la modélisation qui rend compte des propriétés géométriques d'un objet[1]. Autrement dit, il s'agit d'établir un modèle pour représenter la géométrie d'un objet, ou sa forme, et les opérations fondamentales que l'on peut effectuer. En infographie, la modélisation géométrique sert à la génération d'images synthétiques d'objets, possiblement animés, à partir de points de vue quelconques. Ainsi la première étape avant de pouvoir effectuer un rendu et produire une image est de construire une représentation géométrique d'un objet.

Un des plus grands défis en modélisation géométrique est la génération et la manipulation (ou édition) d'objets complexes, c'est-à-dire des objets qui comportent des détails géométriques fins, des structures fines entremêlées, des textures de haute résolution avec motifs élaborés, etc. Quelques exemples d'objets avec des détails complexes incluent des arbres, des sculptures avec traits de ciseau, les détails de surface d'un chandail de tricot, etc. Ainsi, cette complexité géométrique est souvent substituée par des textures colorées (§3.1) plus riches plaquées sur des surfaces ayant une géométrie plus simple.

Pour parvenir à bien générer et manipuler des objets complexes, la géométrie de ces derniers doit être bien représentée. Une grande variété de représentations géométriques et d'outils ont été développés depuis les débuts de l'infographie. Les plus répandus sont les maillages, ce qui inclut aussi les surfaces de subdivision et les maillages de surfaces paramétriques autant que les maillages polygonaux. Les surfaces implicites et les représentations volumétriques sont aussi

[1] Définition du *Grand dictionnaire terminologique* de l'Office de la langue française du Québec. `http://www.granddictionnaire.com/`

fréquemment rencontrées. Cependant, depuis la dernière décennie, une autre approche pour représenter la géométrie d'un objet a connu un essor : les représentations par points.

Avec une telle représentation, la surface d'un objet est implicitement définie par un ensemble de points. En terme de structure, un ensemble de points est simple à traiter. Ceci confère plusieurs avantages aux représentations par points, notamment lorsqu'il est question d'effectuer des opérations de découpage, de jonction ou de rééchantillonnage, ce qui est fréquemment requis pour générer ou éditer les détails géométriques.

Puisque les travaux de cette thèse sont basés sur les représentations par points, nous présentons dans ce chapitre les bases de la modélisation géométrique par points. Cependant, pour aider à mettre les représentations par points en contexte, il convient d'abord de donner un aperçu général de la modélisation géométrique, des représentations aux méthodes de création et édition d'objets, ce que nous faisons à la §2.1. Nous présentons ensuite les origines des représentations par points à la §2.2 avant d'aborder les représentations de géométrie elles-mêmes à la §2.3. Nous finissons notre étude d'ensemble par un bref survol de l'utilisation des représentations par points pour le traitement de géométrie à la §2.4.

Il est à noter que l'étude d'ensemble présentée dans ce chapitre n'est pas exhaustive, particulièrement pour les aspects concernant moins les travaux de cette thèse. Pour une lecture plus approfondie en lien avec les points, nous suggérons l'étude d'ensemble de Kobbelt et Botsch [2004] ou le livre édité par Gross et Pfister [2007]. Une collection d'articles quasi-exhaustive est aussi maintenue par Huang [2007].

Remarque 2.1 (Espace) *Pour toute description en lien avec la géométrie, dans ce chapitre autant que les autres de cette thèse, nous parlerons toujours (ou presque) en terme de l'espace euclidien canonique à trois dimensions (3D), i.e. l'espace vectoriel* $\mathbb{R}^3$ *muni du produit scalaire canonique, et parfois à deux dimensions (2D), en particulier pour les illustrations. Bien que la plupart des concepts se généralisent à des dimensions arbitraires, voire à d'autres espaces, tenir compte de ces généralisations dépasse la portée de cette thèse et alourdirait le contenu inutilement.*

2.1 Modélisation géométrique

Dans cette section, nous effectuons un bref survol de la modélisation géométrique, des représentations (§2.1.1) aux méthodes de création et d'édition (§2.1.2).

2.1.1 Représentations

La représentation géométrique d'un objet est la base fondamentale de la modélisation géométrique. Il existe une grande variété de représentations, chacune ayant ses avantages et inconvénients. Les principales sont décrites dans les sections suivantes (voir [Foley *et al.* 1996, Kobbelt 2003] pour une couverture plus complète), outre les représentations par points, auxquelles nous donnons une attention particulière plus tard à la §2.3.

2.1.1.1 Maillages polygonaux

La surface d'un objet est approximée linéairement par morceaux avec un ensemble de polygones (la plupart du temps des triangles) partageant des sommets[2]. Il s'agit essentiellement d'un graphe, où chaque sommet correspond à un point de la surface, et les polygones correspondent aux faces du graphe. Ainsi, les maillages polygonaux conservent de façon explicite l'information de connectivité, *i.e.* l'information de voisinage entre sommets[3], et requièrent alors des structures de données plus sophistiquées [Weiler 1985].

Ce type de représentation est très puissant car il peut représenter aisément une surface de topologie arbitraire. Cependant, il faut beaucoup de petits polygones pour bien approximer des formes courbes avec des détails de hautes fréquences, l'édition manuelle de ce type de surface peut demander un travail fastidieux, et le maintien des informations de connectivité est enclin aux erreurs lorsqu'il s'agit de "coudre" des maillages ensemble. À ce jour, c'est la représentation la plus utilisée dans le domaine de l'infographie et le matériel graphique est optimisé pour cette représentation.

2.1.1.2 Surfaces paramétriques

La position 3D d'un point d'une surface est exprimée par une fonction à deux paramètres. Le type de fonction le plus fréquemment employé est la NURBS (*non uniform rational B-spline* ou B-spline rationnelle non uniforme) [Piegl et Tiller 1995]. Les NURBS sont très populaires auprès des infographistes 3D car elles sont plus intuitives à manipuler que les maillages polygonaux[4], et elles peuvent bien représenter les surfaces courbes, dont les coniques, les quadriques et les surfaces de révolution. Cependant, représenter des objets ayant une topologie autre que

[2] À noter que pour des polygones non triangulaires, il est possible que les faces ne soient pas planaires.

[3] Pour certaines applications, notamment pour le rendu, des informations partielles de connectivité peuvent être suffisantes.

[4] Nous devrions dire *naturellement* plus intuitives en l'absence d'outils de manipulation de plus haut niveau pour les maillages.

celle d'un plan, d'un cylindre ou d'un tore requiert un effort plus méticuleux pour assurer la continuité entre les diverses *patches* de surface.

2.1.1.3 Surfaces de subdivision

Cette représentation est une sorte d'hybride entre les deux précédentes, mais il plutôt s'agit d'une généralisation des splines [Andersson et Stewart 2009]. Une surface de subdivision [Zorin *et al.* 2000, Warren et Weimer 2001] correspond à la limite d'un processus de subdivision sur un maillage de contrôle avec une opération de lissage, ce qui produit une suite de maillages de contrôle de plus en plus lisses et denses, *i.e.* plus de faces et de sommets. En pratique, la subdivision est appliquée un nombre fini de fois, et le maillage de contrôle ou la surface produite sont manipulés comme un maillage régulier lors de l'édition.

Ce type de représentation a presque toute la puissance des surfaces paramétriques sans la contrainte topologique. Cependant, ces dernières ont une paramétrisation naturelle plus directe et facile d'usage, contrairement aux surfaces de subdivision. De plus, elles n'offrent aucun avantage particulier lorsqu'il est question de représenter des détails géométriques fins, sauf peut-être pour leur aspect naturel pour les représentations multirésolutions (§5.2).

2.1.1.4 Représentations implicites

La surface est exprimée par tous les points $\boldsymbol{x}$ tels que $F(\boldsymbol{x}) = 0$, F étant une fonction scalaire quelconque. Elles peuvent aisément représenter des formes de quadriques naturelles (sphère, cylindre, etc.). Mais pour des formes plus complexes, il faut plutôt passer par des compositions de fonctions implicites, par exemple les RBFs [Savchenko *et al.* 1995, Carr *et al.* 2001, Dinh *et al.* 2002, Turk et O'Brien 2002], les compositions de fonctions à support compact [Muraki 1991, Morse *et al.* 2001, Ohtake *et al.* 2003a], les méthodes de moindres carrés mobiles (*moving least squares* ou *MLS*) et les méthodes d'ensembles de niveaux (*level sets*) [Osher et Paragios 2003]. D'autres types sont décrits dans un ouvrage de référence édité par Bloomenthal [1997].

Ces types de représentation sont intéressants pour leur capacité de représenter des objets de topologie arbitraire et qui incorporent aussi la notion de volume. Il est aussi plus facile de détecter des collisions, ce qui les rend aussi intéressants comme représentations de base pour les méthodes de composition (§2.1.1.6). La consommation de mémoire peut poser un problème pour les méthodes d'ensembles de niveaux si l'objet représenté a beaucoup de détails fins.

2.1.1.5 Représentations volumétriques

L'objet est représenté non pas par sa surface mais par son volume, *i.e.* des informations sur l'intérieur de l'objet sont conservées, et sa surface est définie par la frontière entre l'intérieur et l'extérieur du volume. Une telle représentation typique est la grille de voxels. Un voxel est un élément de volume et il est l'équivalent 3D d'un pixel dans une image à trame[5]. Autrement dit, le volume d'un objet est échantillonné aux sommets d'une grille 3D. L'information conservée dans un voxel est typiquement la couleur, la densité et la distance au point le plus proche de la surface [Chen *et al.* 2000]. Cette représentation peut également servir pour échantillonner des fonctions implicites, comme c'est le cas pour les méthodes d'ensembles de niveaux ou pour les champs de distance (*distance fields*), *i.e.* chaque voxel a un attribut de distance. Cette représentation est fréquemment employée en imagerie médicale. La quantité de mémoire peut rapidement être un problème, de même que des problèmes associés au crénelage (*aliasing*). Ces problèmes peuvent être réduits en utilisant des représentations adaptatives [Frisken *et al.* 2000, Frisken et Perry 2002, Houston *et al.* 2006, Nielsen et Museth 2006].

2.1.1.6 Compositions

Il s'agit davantage d'une opération de modélisation que d'une représentation en soi. Les compositions prennent un ensemble d'objets, eux-mêmes ayant leur représentation, et les organisent ensemble selon diverses opérations. Les deux types de composition les plus communs sont les modèles hiérarchiques et la CSG (*constructive solid geometry* ou géométrie de construction de solides) [Requicha et Voelcker 1977, Foley *et al.* 1996].

2.1.1.7 Représentations spécialisées

Il existe aussi d'autres représentations plus spécialisées, dont les systèmes de particules (§2.2.1), les modèles définis avec une grammaire et les fractales [Foley *et al.* 1996]. Ces représentations sont souvent employées pour modéliser des objets naturels, tels des plantes [Prusinkiewicz et Lindenmayer 1990] ou des terrains [Musgrave *et al.* 1989].

2.1.2 Création et édition d'objets

Le choix de la représentation géométrique a bien entendu un impact sur les méthodes de création et d'édition de l'objet à modéliser. Les représentations frontalières (maillages polygo-

[5] *i.e.* une grille 2D régulière.

naux, surfaces paramétriques, surfaces de subdivision, ensemble de points) sont généralement plus populaires car elles donnent plus d'informations explicites sur la forme de l'objet, permettant de les afficher plus rapidement, et offrant un plus grand contrôle direct sur la forme, mais au prix souvent d'algorithmes plus complexes, particulièrement au niveau du travail d'implémentation. Ainsi, la majorité des méthodes de création et d'édition ont été développées en fonction de ces représentations.

Nous regroupons les méthodes en trois catégories principales, que nous décrivons dans les sections suivantes : l'approche traditionnelle (§2.1.2.1), la modélisation procédurale (§2.1.2.2) et la numérisation 3D (§2.1.2.3).

2.1.2.1 Approche traditionnelle

Dans le domaine de l'infographie, ce que nous appelons l'approche traditionnelle consiste à employer un logiciel interactif (un *modeleur*) pour manipuler les informations de la représentation géométrique d'un objet. Il existe plusieurs logiciels commerciaux puissants couramment utilisés dans l'industrie du cinéma et du jeu vidéo, dont les principaux sont *Maya* [Autodesk, Inc.b], *XSI* [Avid Technology, Inc.] et *3ds max* [Autodesk, Inc.a]. Avec ces logiciels, un usager a à sa disposition une vaste gamme d'outils pour créer et transformer des maillages polygonaux, des surfaces NURBS, des surfaces de subdivision, etc. Ces logiciels, bien que très puissants, sont plutôt complexes et, même avec beaucoup de talent et d'expérience, la quantité de travail pour produire des objets complets et bien détaillés est considérable. Des systèmes interactifs plus simples ont été développés pour créer certaines formes [Zeleznik *et al.* 1996, Igarashi *et al.* 1999, Markosian *et al.* 1999, Karpenko *et al.* 2002, Schmidt *et al.* 2005, Karpenko et Hughes 2006, Kara et Shimada 2007, Nealen *et al.* 2007], mais au prix d'une réduction sur la complexité et le type des formes que l'utilisateur peut créer. Nous qualifions aussi de *directe* l'approche de ces logiciels et systèmes, *i.e.* c'est l'utilisateur qui *dessine* les formes.

2.1.2.2 Modélisation procédurale

Une autre approche, souvent partiellement incorporée dans les logiciels de la section précédente, est celle qualifiée de procédurale [Ebert *et al.* 1998], *i.e.* la géométrie d'un objet est définie par une procédure, similairement à une procédure dans un langage de programmation. Voici quelques exemples de modèles procéduraux :

- la génération de terrains par simulation d'érosion [Kelley *et al.* 1988] ou par fractales [Musgrave *et al.* 1989] ;
- les systèmes-L [Lindenmayer 1968], qui décomposent un objet par un ensemble de règles de réécriture ; ils ont grandement été utilisés pour la modélisation de plantes [Prusinkiewicz et Lindenmayer 1990, Měch et Prusinkiewicz 1996, Deussen *et al.* 1998] et pour la génération de réseaux routiers et la forme d'édifices dans les villes [Parish et Müller 2001] ;
- la génération de murs de briques ou de pierres [Miyata 1990, Lefebvre et Poulin 2000] ;
- les méthodes cellulaires, généralement employées pour produire des textures géométriques (enrichissement des détails géométriques de surface d'un objet) [Fleischer *et al.* 1995, Legakis *et al.* 2001] ;
- les langages de *shading*, tels *RenderMan* [Upstill 1989, Apodaca et Gritz 1999], qui est très fréquemment employé dans l'industrie pour la modélisation et le *shading* procéduraux ; il ne s'agit pas d'un modèle en soi mais il est utilisé pour en définir.

C'est pour ce type d'approche que les représentations spécialisées (§2.1.1.7) sont le plus utilisées. La plupart ont d'ailleurs été conçues en fonction d'un modèle procédural.

Bien que très puissants dans leur contexte d'applications, les modèles procéduraux sont généralement limités dans leur champ d'applications, et sont souvent moins intuitifs d'usage. Savoir comment jauger les paramètres pour obtenir le résultat désiré n'est pas toujours facile. La plupart des modèles procéduraux ont été conçus dans un but de création et non d'édition.

2.1.2.3 Numérisation 3D

L'ensemble des outils de modélisation des sections précédentes assument que nous restons dans le monde virtuel pour créer des objets. Par contre, le monde réel contient certainement une énorme variété d'objets à partir desquels on peut s'inspirer pour en créer des versions virtuelles. Plutôt que de répliquer un objet par un travail de modélisation traditionnelle comme un peintre le ferait pour une peinture, ce qui peut prendre beaucoup de temps et de ressources, il est possible d'en *prendre une photographie*[6], *i.e.* de faire usage d'instruments de mesure pour *numériser* un objet.

[6]Dans le cas précis du peintre, la photographie sera préférable en considérant que nous recherchons le réalisme (ou photoréalisme). D'un point de vue artistique, c'est un tout autre monde.

Il existe une très vaste littérature sur le domaine de la numérisation 3D et nous n'allons faire qu'un bref survol, ce travail ne portant pas sur la numérisation en soi. Pour une couverture plus large de la littérature du domaine, des livres généraux devraient être consultés [Ayache 1991, Faugeras 1993, Trucco et Verri 1998, Klette *et al.* 1998], ou alors des revues de domaines plus spécifiques [Zhang *et al.* 1999]. Une importante collection d'articles est également maintenue par Price [2008] de *USC* (l'université de la Californie du Sud).

Brièvement, la numérisation 3D consiste à mesurer la position 3D de points sur la surface d'un objet. La mesure peut être faite par palpage ou télémétrie, par des méthodes optiques passives ou par triangulation optique active. Le palpage fait usage d'un équipement spécialisé et calibré (*i.e.* de connaître la position du capteur relativement à un repère de référence) qui mesure les positions par mise en contact avec un objet. La télémétrie se base sur l'émission d'ondes et la mesure du temps pour le retour au capteur. Les méthodes optiques passives procèdent par mise en correspondance de pixels d'un ensemble d'images calibrées (possiblement une séquence vidéo) de points de vue différents d'un objet. Enfin, la triangulation optique active utilise une source lumineuse (laser ou motifs de lumière) et une caméra, calibrées entre elles, pour calculer la position 3D d'un point observé depuis la source et la caméra par l'intersection des deux vecteurs formés par leur direction de projection respective. Les méthodes basées sur la triangulation optique active produisent une grande quantité de points de manière précise, d'où leur grande popularité.

Il existe enfin d'autres méthodes qui exploitent des informations supplémentaires pour soit aider les méthodes précédentes, soit s'en servir pour développer de nouvelles méthodes. Parmi les types d'information exploités, il y a la silhouette des objets dans les images [Cipolla et Blake 1992, Szeliski 1993, Zheng 1994, Sullivan et Ponce 1998, Snow *et al.* 2000], la cohérence du mouvement [Tomasi et Kanade 1992, Toscani et Faugeras 1987], l'illumination [Horn et Brooks 1989], la mise au point et le flou de mise au point (*focus/defocus*) [Ens et Lawrence 1993], les textures sur la surface des objets [Kanatani et Chou 1989], etc.

Plusieurs méthodes interactives ont également été développées dans le domaine académique [Debevec *et al.* 1996, Sato *et al.* 1997, Poulin *et al.* 1998, Guillou *et al.* 2000, Shlyakhter *et al.* 2001, Dedieu 2001, Poulin *et al.* 2003, Granger-Piché *et al.* 2004, Popescu *et al.* 2004] et plusieurs systèmes commerciaux font usage de l'apport de l'usager [Canoma, Eos Systems Inc., REALVIZ S.A.]. L'intégration de l'usager et ses connaissances au processus de recons-

truction peuvent souvent aider à accroître la qualité de la reconstruction par la sémantique des éléments absente des systèmes.

Le résultat de la numérisation est un ensemble de positions 3D échantillonnant la géométrie de l'objet. Des attributs supplémentaires peuvent être présents, comme la couleur ou une normale. Les numérisations sont parfois effectuées par morceaux (notamment pour les méthodes par triangulation optique active). Dans ces cas, il faut d'abord les fusionner. L'étape suivante est de transformer les informations obtenues de la numérisation vers une représentation plus adéquate, tel un maillage polygonal[7]. Plusieurs méthodes de reconstruction à partir d'un nuage de points ont ainsi été développées [Bolle et Vemuri 1991, Hoppe *et al.* 1992, Hoppe *et al.* 1994, Curless et Levoy 1996, Wheeler *et al.* 1998, Amenta et Bern 1999, Kobbelt et Botsch 2000, Amenta *et al.* 2001, Dey *et al.* 2001, Zhao *et al.* 2001, Dinh *et al.* 2002, Jeong et Kim 2002, Dey et Goswami 2004, Kolluri *et al.* 2004, Zwicker et Gotsman 2004, Kazhdan 2005, Wicke *et al.* 2005a, Hornung et Kobbelt 2006, Jenke *et al.* 2006, Allègre *et al.* 2007, Sharf *et al.* 2007][8]. Il est à noter que certaines méthodes de numérisation produisent des représentations volumétriques. Par exemple, il y a la coloration de voxels [Seitz et Dyer 1999] et la sculpture d'espace [Kutulakos et Seitz 2000]. Les méthodes de stéréovision produisent quant à elles des cartes de profondeur (*depth maps*). Ces représentations peuvent être utilisées directement pour certaines applications, notamment en robotique, mais pour des fins de génération d'images, les maillages polygonaux, les maillages de NURBS, les surfaces implicites ou les surfaces de subdivision sont préférés.

Peu importe la méthode de numérisation, les résultats comportent toujours un certain niveau de bruit ou alors l'objet n'est numérisé que partiellement dû à l'occultation, à des régions inatteignables de l'objet, ou encore à des propriétés de réflectance de la surface de l'objet (en particulier pour les méthodes optiques). En effet, l'hypothèse d'un modèle de réflectance diffuse est presque toujours présente, et certaines méthodes peuvent échouer localement si la surface est trop spéculaire. En conséquence, un travail additionnel de filtrage [Taubin 1995, Kobbelt 1997, Desbrun *et al.* 1999, Guskov et Wood 2001, Peng *et al.* 2001, Liu *et al.* 2002, Tasdizen *et al.* 2002, Fleishman *et al.* 2003b, Jones *et al.* 2003, Hu *et al.* 2004, Li *et al.* 2004, Fournier *et al.* 2003, Schall *et al.* 2007, Sun *et al.* 2007] et de réparation [Pfeifle et Seidel 1996, Davis *et al.* 2002, Liepa 2003, Nooruddin et Turk 2003, Ju 2004, Sharf *et al.* 2004, Wood

[7]Nous verrons à la §2.3 que cette étape peut en fait être facultative.

[8]Cette liste de références, à défaut d'avoir trouvé des études d'ensembles, est très partielle et ne donne qu'un aperçu de l'ensemble du domaine de la reconstruction de surface.

et al. 2004, Bischoff *et al.* 2005, Nguyen *et al.* 2005, Park *et al.* 2005, Pauly *et al.* 2005b, Gingold et Zorin 2007][9] de la géométrie reconstruite est généralement nécessaire. La plupart de ces méthodes ont un effet de lissage (filtrage passe-bas) sur la géométrie, à l'exception des travaux de Sharf *et al.* [2004], Nguyen *et al.* [2005], Park *et al.* [2005], Pauly *et al.* [2005b] et Xiao *et al.* [2007], que nous allons couvrir plus en détail à la §3.2. Ainsi, pour être en mesure de conserver les détails, il faut une numérisation d'une très grande précision [Levoy *et al.* 2000].

Certains types de travaux additionnels de réparation que nous pouvons effectuer après numérisation indiquent une voie de recherche au niveau de la génération de géométrie, en particulier avec des méthodes de génération comme celles de Sharf *et al.* [2004], Nguyen *et al.* [2005], Park *et al.* [2005] et Xiao *et al.* [2007], qui, pour remplir un trou, recherchent dans la géométrie de l'objet des morceaux similaires à sa géométrie avoisinante. Comme nous le verrons au prochain chapitre (3), générer un objet (texture ou géométrie) à partir d'un jeu d'exemples peut s'avérer être un outil puissant de modélisation.

2.2 Il était une fois, des points...

Il est possible d'obtenir des numérisations 3D très détaillées d'objets contenant des millions de points, et ce de manière de plus en plus précise. Pour pouvoir traiter ces données, il faut généralement les convertir en une représentation telle un maillage (§2.1.2.3). Bien entendu, cette conversion a un coût en temps et en ressources, sans compter le travail supplémentaire pour "nettoyer" les résultats. Il y a donc un intérêt à traiter directement les ensembles de points initiaux. Cependant, l'idée d'utiliser les points comme primitives géométriques n'origine pas du traitement de numérisations 3D, ce qui renforce davantage son intérêt. Elle a d'abord été exploitée pour représenter des classes d'objets particuliers (§2.2.1) et pour le rendu d'objets complexes (§2.2.2).

2.2.1 Représentations pour des objets particuliers

Historiquement, les points ont d'abord été utilisés pour la modélisation d'objets difficiles pour des surfaces traditionnelles (polygonales et paramétriques). Par exemple, il y a la fumée de Csuri *et al.* [1979], les nuages de Blinn [1982], les forêts et explosions de Reeves [1983], et les arbres de Smith [1984].

[9] À nouveau, à défaut de référence pour des études d'ensemble, ces listes de références ne sont que des échantillons jugés représentatifs.

La représentation commune employée dans chaque cas est le système de particules, *i.e.* un ensemble de points 3D (particules) qui évoluent dans le temps et l'espace selon diverses règles. Des systèmes de particules plus génériques ont été développés par Reeves et Blau [1985] et Sims [1990]. Cette idée d'un échantillonnage dans l'espace d'un phénomène complexe a été reprise par la suite pour le rendu et la modélisation d'objets complexes, *i.e.* dont la surface possède des détails géométriques de hautes fréquences ou des structures fines.

2.2.2 Le rendu par points

Une grande vogue est apparue aux environs de l'an 2000 pour l'usage de points comme primitives de rendu. Cependant, les pionniers en la matière ont été Levoy et Whitted [1985], qui ont proposé un système de rendu dont la primitive fondamentale est le point. Un de leurs arguments principaux était qu'un affichage de qualité de surfaces avec beaucoup de caractéristiques géométriques courbes requiert une grande subdivision de polygones, souvent jusqu'à ce qu'un polygone soit affiché dans moins d'un pixel. Alors on perd l'intérêt principal de l'exploitation de la cohérence du tramage (*rasterization*) des polygones. Leur système génère des points durant le rendu à partir d'objets qui doivent être représentés par des surfaces différentiables pour combler les trous possibles dans l'image finale.

Max et Ohsaki [1995] ont pris également l'idée d'utiliser des points pour faire le rendu d'arbres, dont la complexité géométrique des branches et des feuilles est très grande. Les points étaient utilisés pour faire un rendu approximatif mais suffisant pour la visualisation.

Grossman et Dally [1998] ont plus tard remis à l'ordre du jour les idées de Levoy et Whitted [1985] pour le rendu interactif d'objets complexes. Jusque là, la méthode populaire pour le rendu de tels objets était le rendu à base d'images (*image-based*) [McMillan et Bishop 1995, Levoy et Hanrahan 1996, Gortler *et al.* 1996, Debevec *et al.* 1998, Shade *et al.* 1998], mais ces méthodes sont très coûteuses en mémoire (dans l'ordre de centaines de mégaoctets), d'où leur recherche d'une autre solution sans la nécessité de matériel graphique particulier. Les objets sont échantillonnés par projections orthographiques sur différents points de vue autour de l'objet, ces derniers étant séparés par un angle maximum pour garantir l'affichage sans trous pour une résolution d'image cible. Une variante d'un algorithme *pull-push* [Gortler *et al.* 1996] est employée pour combler les trous après la projection des points.

Avec l'augmentation continuelle de la complexité des modèles géométriques, en particulier due à l'amélioration des techniques de numérisation 3D (§2.1.2.3) qui fournissent beaucoup de

données, Grossman et Dally avaient entamé le pas pour le boom qui allait suivre, dont les deux travaux fondamentaux sont ceux de Pfister *et al.* [2000] et de Rusinkiewicz et Levoy [2000].

Pfister *et al.* [2000] ont repris directement les idées de Grossman et Dally [1998] dans un système plus avancé où ils décrivent un pipeline graphique (*graphic pipeline*) complet adaptable pour une implémentation matérielle et qui est en mesure de produire des images avec peu de crénelage et de façon interactive sans accélération matérielle. Ils ont également popularisé le nom de *surfels* (*surface elements* ou éléments de surface) pour la primitive, c'est-à-dire un point avec une normale, le rayon d'un disque perpendiculaire à la normale et d'autres attributs facultatifs (couleur, texture, etc.). Des travaux subséquents par Zwicker *et al.* [2001b] sur le *splatting* de surfaces ont eu également un grand impact sur le rendu de haute qualité avec points.

Rusinkiewicz et Levoy [2000, 2001] ont développé une autre approche avec leur système hiérarchique *QSplat* pour la visualisation de maillages de très grande taille (les sculptures de Michel-Ange numérisées par Levoy *et al.* [2000], pouvant aller jusqu'à plus de 127 millions de points pour la numérisation du Saint Mathieu). La hiérarchie est construite par agrégation (*clustering*) de points voisins dans des sphères (incluant leurs attributs de couleur et normale), qui à leur tour sont regroupées dans d'autres sphères, et ainsi de suite jusqu'à ce qu'il n'en reste qu'une. L'affichage est ensuite fait en traversant à partir de la racine (une sphère englobante) la hiérarchie et dessinant un point par sphère à l'aide d'un petit carré ou d'un disque dont la taille est déterminée par le rayon de la sphère et en s'arrêtant lorsqu'un point est projeté dans moins d'un pixel.

Par la suite, une panoplie de techniques de rendu par points ont été développées. Une bonne partie de ces techniques suivent la ligne d'idée du *splatting* [Botsch *et al.* 2002, Coconu et Hege 2002, Ren et Zwicker 2002, Botsch et Kobbelt 2003, Guennebaud et Paulin 2003, Křivánek *et al.* 2003, Botsch *et al.* 2004, Guennebaud *et al.* 2004a, Pajarola *et al.* 2004, Sainz *et al.* 2004b, Zwicker *et al.* 2004, Botsch *et al.* 2005, Talton *et al.* 2005, Wu *et al.* 2005, Zhang et Pajarola 2006, Zhang et Pajarola 2007, Weyrich *et al.* 2007, Heinzle *et al.* 2008b], qui a aussi été appliquée pour le rendu à vues multiples [Hübner *et al.* 2006]. D'autres ont proposé l'usage d'une primitive enrichie d'informations de courbure pour permettre un rendu de qualité avec moins de primitives [Kalaiah et Varshney 2001, Kalaiah et Varshney 2003a]. Récemment, il y a eu un petit regain pour les méthodes de rendu travaillant en espace image [Marroquim *et al.* 2007, Rosenthal et Linsen 2008], de façon similaire à Grossman et Dally [1998]. Un certain nombre de travaux concernent des structures hiérarchiques,

comme *QSplat*, pour accélérer le rendu en présence d'un grand nombre de points [Klein *et al.* 2002, Klein *et al.* 2004, Dachsbacher *et al.* 2003, Duguet et Drettakis 2004, Gobbetti et Marton 2004, Namane *et al.* 2004, Pajarola *et al.* 2005, Wimmer et Scheiblauer 2006, He et Liang 2007, Holst et Schumann 2007, Zhang et Kaufman 2007], quand il ne s'agit pas de déterminer des PVS (*potential visible sets* ou ensembles potentiellement visibles) [Mantler et Fuhrmann 2003]. Le nombre de points à afficher peut aussi être contrôlé par un échantillonnage dynamique ou semi-dynamique de la surface des objets [Stamminger et Drettakis 2001, Chhugani et Kumar 2003, Qu *et al.* 2006, Ziegler *et al.* 2006, Guennebaud *et al.* 2008].

Le rendu par points n'étant pas la méthode la plus indiquée pour afficher des surfaces relativement planes, des méthodes hybrides entre points, segments et polygones ont été développées [Chen et Nguyen 2001, Cohen *et al.* 2001, Dey et Hudson 2002, Deussen *et al.* 2002, Zheng *et al.* 2002, Wilson *et al.* 2002, Bala *et al.* 2003, Dachsbacher *et al.* 2003, Guennebaud *et al.* 2006, Thangudu *et al.* 2007]. Il existe aussi des méthodes de lancer de rayons (*ray-tracing*) de surfaces représentées avec des points [Schaufler et Jensen 2000, Adamson et Alexa 2003b, Wand et Straßer 2003, Wald et Seidel 2005, Hubo *et al.* 2006] pour le rendu d'effets plus sophistiqués d'illumination. D'autres effets ont été étudiés, dont le flou de mouvement [Guan et Mueller 2004], des méthodes de rendu non photoréaliste [Kawata *et al.* 2004, Xu *et al.* 2004, Xu et Chen 2004, Runions *et al.* 2007], et diverses méthodes reliées à l'illumination [Subr *et al.* 2003, Dobashi *et al.* 2004, Liu *et al.* 2006]. Le rendu par points a également trouvé son utilité pour le rendu d'objets représentés par des surfaces implicites [Bærentzen et Christensen 2003, Co *et al.* 2003, Reuter *et al.* 2003, Sigg *et al.* 2006, Waters *et al.* 2006] ou par des opérations CSG [Wicke *et al.* 2004], et aussi pour le rendu de données volumétriques [Zwicker *et al.* 2001a, Wilson *et al.* 2002, Welsh et Mueller 2003, Museth et Lombeyda 2004].

Nous invitons le lecteur à consulter les études d'ensemble traitant, au moins en partie, du rendu par points [Kobbelt et Botsch 2004, Sainz *et al.* 2004a, Gross et Pfister 2007] pour en savoir plus sur le sujet. Les travaux de cette thèse, bien qu'ils font usage de la représentation par points, portent plus spécifiquement sur la modélisation géométrique, et non le rendu. Les images des résultats qui apparaissent dans cette thèse ont été produites par des techniques courantes, et même certaines en utilisant un logiciel déjà disponible : *PointShop 3D* [Zwicker *et al.* 2002].

2.3 Représenter la géométrie avec des points

À la base, les ensembles de points sont sans doute l'une des représentations géométriques les plus simples : la surface d'un objet est représentée par un ensemble de points distribués sur sa surface, ou à proximité. Cette simplicité est l'avantage principal de cette représentation. L'absence de structure sophistiquée de connectivité en fait une représentation très flexible pour la création et l'édition d'objets, d'où notre choix de représentation pour les travaux présentés dans cette thèse.

Szeliski et Tonnesen [1992] ont été les premiers à représenter des surfaces sans maillage avec des points. Ils utilisent un système de particules orientées pour représenter la surface d'objets déformables. Autrement, l'idée de représentation géométrique par un ensemble de points fini $P = \{\boldsymbol{p}_i\}_{i=1}^{n} \subset \mathbb{R}^3$ est venue des besoins du rendu par points [Grossman et Dally 1998, Pfister *et al.* 2000, Rusinkiewicz et Levoy 2000]. Mais peu de temps après, on s'est vite rendu compte du grand potentiel de telles représentations pour la modélisation géométrique.

Dans cette section, nous faisons un survol de l'ensemble des travaux qui ont été effectués sur la représentation géométrique par points. Nous débutons par les éléments fondamentaux, soit la notion de voisinage et d'estimation de normales (§2.3.1), puis nous décrivons les modèles principaux de représentations, soient les représentations purement par points (§2.3.2) et les surfaces définies par moindres carrés mobiles (§2.3.3).

2.3.1 Voisinage et normale

À la base de toute surface définie par un ensemble de points est la notion de voisinage autour d'un point. C'est ce qui détermine en quelque sorte la continuité de la surface, et donc l'information de normale provient aussi habituellement du voisinage. Pour les surfaces à base de maillages, le voisinage est explicitement donné par l'information de connectivité. Pour les surfaces paramétriques ou les représentations volumétriques, le voisinage est donné par la paramétrisation. Dans le cas des ensembles de points, le voisinage doit être reconstruit. À la §2.3.1.1, nous présentons divers types de voisinage utilisés pour les représentations par points, et à la §2.3.1.2, nous abordons l'estimation de normales.

2.3.1.1 Types de voisinage

Il existe plusieurs types de voisinage sur un ensemble de points $P \subset \mathbb{R}^3$. Le plus simple, illustré à la figure 2.1 (a), est le voisinage euclidien autour d'un point $\boldsymbol{x} \in \mathbb{R}^3$, que nous notons

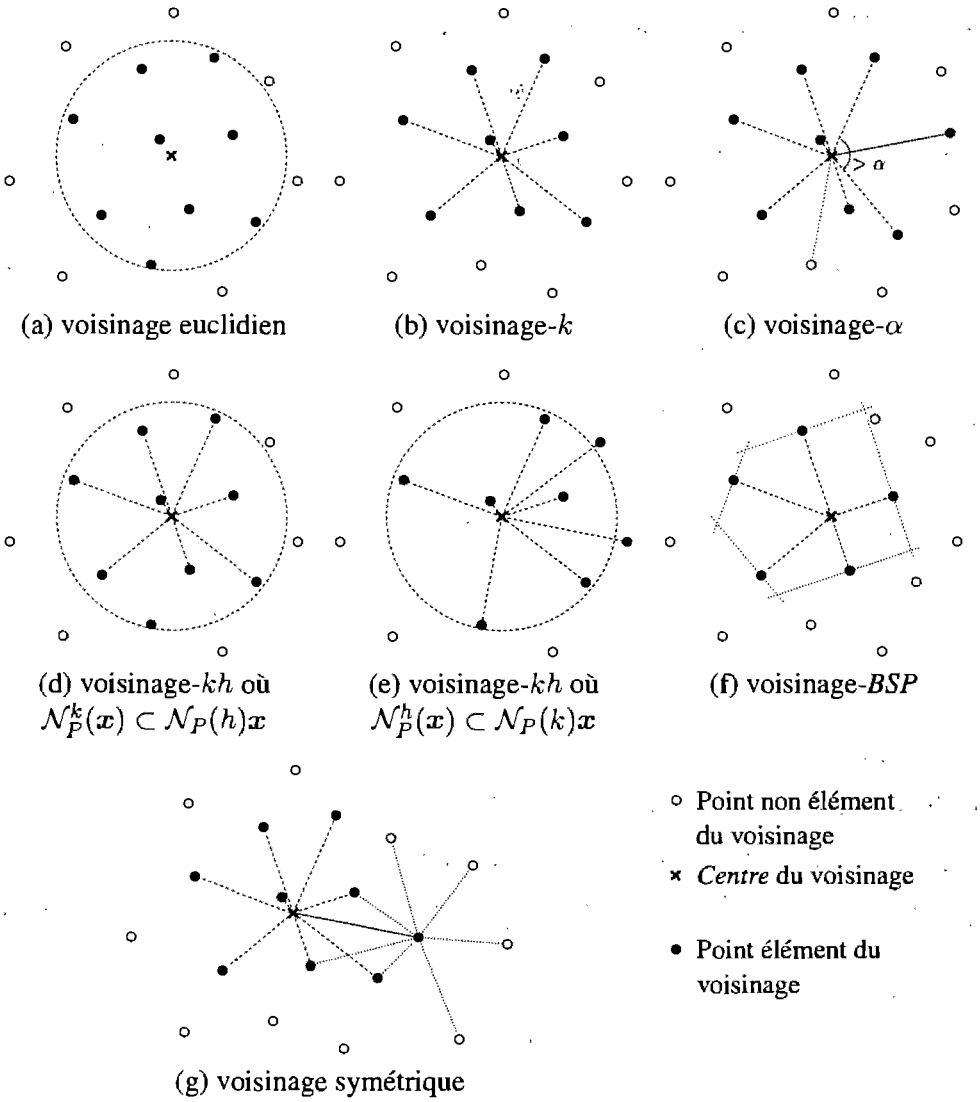


FIG. 2.1 – Types de voisinage.

$\mathcal{N}_P^h(\boldsymbol{x})$, qui correspond à l'ensemble des points de P inclus dans une sphère de rayon h centrée sur $\boldsymbol{x}$, *i.e.*

$$\mathcal{N}_P^h(\boldsymbol{x}) = \{\boldsymbol{p} \in P \mid \|\boldsymbol{p} - \boldsymbol{x}\| \leq h\}. \tag{2.1}$$

Un autre type commun est le voisinage-k autour d'un point $\boldsymbol{x}$, que nous notons $\mathcal{N}_P^k(\boldsymbol{x})$. Il s'agit d'un sous-ensemble de P contenant les k voisins plus proches de $\boldsymbol{x}$ (en supposant bien sûr que $k \leq |P|$), *i.e.*

$$\mathcal{N}_P^k(\boldsymbol{x}) \in \{\mathcal{N} \subseteq P \mid |\mathcal{N}| = k \wedge (\forall \boldsymbol{p} \in \mathcal{N}\, \forall \boldsymbol{q} \in P \backslash \mathcal{N}\ \|\boldsymbol{p} - \boldsymbol{x}\| \leq \|\boldsymbol{q} - \boldsymbol{x}\|)\}. \tag{2.2}$$

Il est en effet à noter que $\mathcal{N}_P^k(\boldsymbol{x})$ n'est pas nécessairement unique. Par exemple, si tous les points de P sont distribués sur une sphère et que $\boldsymbol{x}$ est situé exactement en son centre, il y a alors $\binom{|P|}{k}$ voisinages-k différents. Le choix exact de $\mathcal{N}_P^k(\boldsymbol{x})$ est habituellement arbitraire. La figure 2.1 (b) montre un exemple de voisinage-k. Ces deux types de voisinages peuvent être calculés efficacement par l'intermédiaire de partitionnements de l'espace [Samet 2006] tels un arbre-*kd* (*kd-tree*) [Bentley 1975]. Sankaranarayanan *et al.* [2006] ont proposé aussi une méthode de recherche efficace pour des ensembles de points trop grands pour être entièrement chargés en mémoire. Récemment, une implémentation matériel pour accélérer la recherche d'un voisinage-k a été proposée par Heinzle *et al.* [2008a].

Un problème associé au voisinage euclidien survient pour les ensembles de points avec une densité de points variable. En effet, le nombre de points accumulés dans les zones de haute densité sera beaucoup plus grand que dans les zones de faible densité, où il est même possible de ne récolter aucun point si h n'est pas assez grand. Le voisinage-k n'a pas ce problème, mais dans les zones de changement abrupt de densité, la distribution des voisins aura un fort biais vers la zone de plus haute densité. Il est possible de réduire l'impact de ces problèmes à l'aide d'un voisinage-$k\epsilon$ [Bendels *et al.* 2006] (que nous renommons voisinage-kh pour rester cohérent avec le reste de notre notation) simplement défini ainsi :

$$\mathcal{N}_P^{kh}(\boldsymbol{x}) = \mathcal{N}_P^k(\boldsymbol{x}) \cup \mathcal{N}_P^h(\boldsymbol{x}). \tag{2.3}$$

Les figures 2.1 (d) et (e) montrent les deux cas typiques d'instance de voisinage-kh. Cependant, le voisinage euclidien n'est pas nécessairement mieux que le voisinage-k pour les zones de changement de densité. Bendels *et al.* [2006] ont proposé alors un voisinage symétrique, illustré à la figure 2.1 (g), ainsi :

$$\ddot{\mathcal{N}}_P(\boldsymbol{x}) = \{\boldsymbol{p} \in P \mid \boldsymbol{p} \in \mathcal{N}_P(\boldsymbol{x}) \vee \boldsymbol{x} \in \mathcal{N}_P(\boldsymbol{p})\} \tag{2.4}$$

où $\mathcal{N}_P$ est soit $\mathcal{N}_P^k$ ou $\mathcal{N}_P^{kh}$, mais pas $\mathcal{N}_P^h$ car celui-ci est déjà symétrique. Pour calculer efficacement $\ddot{\mathcal{N}}_P(\boldsymbol{x})$, un graphe de proximité est construit, *i.e.* un graphe $\mathcal{G} = (V, E)$ où $V = P$ et

$$E = \{(i, j) \mid \boldsymbol{p}_i \in \ddot{\mathcal{N}}_P(\boldsymbol{p}_j)\},$$

ce qui prend plus de ressources mémoire comparativement à la plupart des structures de partitionnement de l'espace.

Linsen et Prautzsch [Linsen 2001, Linsen et Prautzsch 2003] utilisent une autre approche pour compenser le biais des voisinages euclidiens et voisinages-k. Leur voisinage autour d'un point $\boldsymbol{x}$, que nous appelons un voisinage-α $\mathcal{N}_P^\alpha(\boldsymbol{x})$, est calculé en débutant avec $\mathcal{N}_P^k(\boldsymbol{x})$. Soit $\boldsymbol{q}_1, ..., \boldsymbol{q}_k$ les points de $\mathcal{N}_P^k(\boldsymbol{x})$. Un plan est ajusté par moindres carrés aux points $\boldsymbol{x}, \boldsymbol{q}_1, ..., \boldsymbol{q}_k$, puis leur projection $\check{\boldsymbol{x}}, \check{\boldsymbol{q}}_1, ..., \check{\boldsymbol{q}}_k$ sur ce plan est calculée. Les points projetés sont ensuite triés par ordre croissant d'angle $\alpha_i = \angle \check{\boldsymbol{q}}_1 \check{\boldsymbol{x}} \check{\boldsymbol{q}}_i$, pour $i \in \mathbb{Z}[2, k]$. Si $\alpha_i - \alpha_{i-1} > \alpha$, où $\alpha_1 = 0$ et α est un seuil de tolérance (typiquement 90°), le point $\boldsymbol{q}_k$ est substitué par le $(k+1)^{\text{e}}$ voisin le plus proche dans P [10]. Ce processus de remplacement se poursuit jusqu'à ce que le critère d'angle soit satisfait ou jusqu'à ce qu'un nombre maximal donné de remplacements soit atteint. Un échec peut indiquer une frontière. La figure 2.1 (c) montre un exemple d'un voisinage-α. Bendels *et al.* [2006] ont aussi proposé un critère d'angle pour définir une frontière.

Il existe d'autres types de voisinage moins communs [Pauly 2003]. Par exemple, il y a le voisinage-*BSP* (*BSP-neighborhood*), que nous notons $\mathring{\mathcal{N}}_P(\boldsymbol{x})$, qui est défini par l'ensemble des points de P qui ne s'*occultent* pas les uns les autres par rapport à $\boldsymbol{x}$. Un point est occulté par un autre s'il se trouve derrière le plan passant par l'occultant et perpendiculaire à la direction de l'occultant à $\boldsymbol{x}$, à condition que l'occultant ne soit pas lui-même occulté. Plus formellement, le voisinage-*BSP* est défini ainsi :

$$\mathring{\mathcal{N}}_P(\boldsymbol{x}) = \{\boldsymbol{p} \in P \mid v_P(\boldsymbol{p})\} \tag{2.5}$$

où $v_P(\boldsymbol{p})$ est un prédicat de validité défini récursivement ainsi :

$$v_P(\boldsymbol{p}) = \begin{cases} \text{vrai} & \text{si } |P| = 1 \\ \forall \boldsymbol{q} \in P \backslash \{\boldsymbol{p}\} \left((\boldsymbol{x} - \boldsymbol{q})^\mathsf{T} (\boldsymbol{p} - \boldsymbol{q}) \geq 0 \vee \neg v_{P \backslash \{\boldsymbol{p}\}}(\boldsymbol{q}) \right) & \text{sinon.} \end{cases} \tag{2.6}$$

Une illustration de ce type de voisinage se trouve à la figure 2.1 (f). Il existe une variante *floue* de ce dernier type, présentée par Guennebaud *et al.* [2005], pour laquelle il y a une forme de tolérance sur le test d'occultation. Pour rendre le calcul de ces types de voisinage plus rapide, il est généralement effectué à partir d'un voisinage euclidien ou d'un voisinage-k. Le résultat est alors une approximation, qui est cependant très bonne pour les distributions uniformes locales de points.

[10] Ceci correspond à la description de la méthode par Linsen et Prautzsch [2003]. Il nous paraît cependant plus approprié de substituer $\boldsymbol{q}_i$.

Remarque 2.2 (Définition de $\mathcal{N}_P(\boldsymbol{x})$) *À moins d'avis contraire, pour le reste de cette thèse, nous ferons usage du voisinage-k, que nous noterons simplement $\mathcal{N}_P(\boldsymbol{x})$. Nous avons préféré ce type plutôt que le voisinage euclidien car il peut plus aisément accomoder des densités non uniformes de points, et fixer k est considérablement plus simple que déterminer h. Puis, nous avons préféré le voisinage-k plutôt que les autres types car il est plus rapide à calculer. La rapidité est importante compte tenu que ce calcul doit être effectué très fréquemment. Nous considérons les autres types plutôt pour répondre à des besoins particuliers. Enfin, nous faisons usage de arbres-*kd *pour accélérer l'évaluation du voisinage-k.*

2.3.1.2 Estimation de normales

L'un des principaux usages du voisinage est l'estimation d'une normale, notée $\boldsymbol{n}_P(\boldsymbol{x})$, à un point $\boldsymbol{x}$ à proximité de ou sur la surface induite par un ensemble de points P. L'estimation la plus commune est la normale d'un plan ajusté par moindres carrés aux points $\{\boldsymbol{x}\} \cup \mathcal{N}_P(\boldsymbol{x})$, *i.e.*

$$\boldsymbol{n}_P(\boldsymbol{x}) = \arg\min_{\boldsymbol{n}_{\boldsymbol{x}} \in \mathbb{S}^2} \sum_{\boldsymbol{q} \in \{\boldsymbol{x}\} \cup \mathcal{N}_P(\boldsymbol{x})} (\boldsymbol{n}_{\boldsymbol{x}}{}^{\mathsf{T}}(\boldsymbol{q} - \bar{\boldsymbol{q}}))^2 \tag{2.7}$$

où

$$\bar{\boldsymbol{q}} = \frac{1}{|\mathcal{N}_P(\boldsymbol{x})| + 1} \sum_{\boldsymbol{q} \in \{\boldsymbol{x}\} \cup \mathcal{N}_P(\boldsymbol{x})} \boldsymbol{q} \tag{2.8}$$

et $\mathbb{S}^2$ est la sphère unitaire dans $\mathbb{R}^3$. Cette minimisation peut être calculée par analyse de covariance en $\boldsymbol{x}$, *i.e.*

$$\boldsymbol{n}_P(\boldsymbol{x}) = \arg\min_{\boldsymbol{n}_{\boldsymbol{x}} \in \mathbb{S}^2} \boldsymbol{n}_{\boldsymbol{x}}{}^{\mathsf{T}} \left(\sum_{\boldsymbol{q} \in \{\boldsymbol{x}\} \cup \mathcal{N}_P(\boldsymbol{x})} (\boldsymbol{q} - \bar{\boldsymbol{q}})(\boldsymbol{q} - \bar{\boldsymbol{q}})^{\mathsf{T}} \right) \boldsymbol{n}_{\boldsymbol{x}}. \tag{2.9}$$

La solution est alors le vecteur propre associé à la valeur propre la plus petite λ_1 de la matrice de covariance (l'expression entre grandes parenthèses). Mitra *et al.* [2003] calculent $\boldsymbol{n}_P(\boldsymbol{x})$ plutôt

ainsi[11] :

$$\boldsymbol{n}_P(\boldsymbol{x}) = \arg\min_{\boldsymbol{n}_{\boldsymbol{x}} \in \mathbb{S}^2} \boldsymbol{n}_{\boldsymbol{x}}^{\mathsf{T}} \left(\bar{\boldsymbol{q}}\bar{\boldsymbol{q}}^{\mathsf{T}} + \frac{1}{|\mathcal{N}_P(\boldsymbol{x})| + 1} \sum_{\boldsymbol{q} \in \{\boldsymbol{x}\} \cup \mathcal{N}_P(\boldsymbol{x})} \boldsymbol{q}\boldsymbol{q}^{\mathsf{T}} \right) \boldsymbol{n}_{\boldsymbol{x}}. \tag{2.10}$$

D'autres méthodes se basent sur le diagramme de Voronoi [Amenta et Bern 1999, Dey *et al.* 2005].

Pour que l'estimation de la normale soit bonne, non seulement le voisinage doit bien représenter la géométrie locale de la surface [Andersson *et al.* 2004], mais la surface doit être suffisamment bien échantillonnée. Un critère simple d'échantillonnage est que l'estimation de la normale par analyse de covariance (équation (2.9)) soit unique [Adamson et Alexa 2003a], *i.e.* que λ_1 soit strictement inférieure aux autres valeurs propres. Une analyse plus rigoureuse a été présentée par Bremer et Hart [2005]. Celle-ci s'appuie sur un échantillonnage-(ϵ, δ) (l'ensemble de points P) de la surface $\mathcal{S} \subset \mathbb{R}^3$, *i.e.* un échantillonnage qui respecte

$$(\forall \boldsymbol{x} \in \mathcal{S}\ \exists \boldsymbol{p} \in P\ \|\boldsymbol{x} - \boldsymbol{p}\| \leq \epsilon h(\boldsymbol{x})) \wedge (\forall \boldsymbol{p}, \boldsymbol{q} \in P\ \|\boldsymbol{p} - \boldsymbol{q}\| \geq \delta h(\boldsymbol{p})) \tag{2.11}$$

où $h : \mathcal{S} \mapsto \mathbb{R}$ est une fonction donnant la taille des caractéristiques, *i.e.* la distance la plus courte entre $\boldsymbol{x}$ et l'axe médian de $\mathcal{S}$. Il est à noter que les travaux cités [Adamson et Alexa 2003a, Bremer et Hart 2005] concernent d'abord des calculs par moindres carrés mobiles, que nous décrivons à la §2.3.3.1, mais les concepts s'appliquent aussi aux moindres carrés classiques.

L'unicité de λ_1 n'est pas suffisante pour garantir que la normale estimée soit la "bonne". En effet, dans les zones très courbes d'une surface, un sous-échantillonnage peut donner une estimation de la normale tangente à la surface tout en ayant λ_1 unique. Pauly *et al.* [2004a] ont ainsi présenté une méthode pour estimer la confiance de cette estimation basée sur la notion de variation de surface [Pauly *et al.* 2002a]. La variation de surface, notée $\sigma_P(\boldsymbol{x})$, est calculée par le ratio entre λ_1 et la somme de toutes les valeurs propres λ_1, λ_2 et λ_3 de la matrice de covariance en $\boldsymbol{x}$, *i.e.* [12]

$$\sigma_P(\boldsymbol{x}) = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3}. \tag{2.12}$$

Les valeurs propres correspondent aux longueurs des axes principaux d'un ellipsoïde centré sur $\bar{\boldsymbol{q}}$ (équation (2.8)), ou sur $\boldsymbol{x}$ dans le cas MCM (*MLS*) (§2.3.3.1). Cet ellipsoïde donne une

[11] Il y a une petite différence entre cette formule et celle de Mitra *et al.* [2003]. Ils ne considèrent pas $\boldsymbol{x}$ dans le calcul, ce qui est plus robuste si $\boldsymbol{x}$ est plus à l'écart de la surface, mais ce qui induit une estimation linéaire par morceaux (*piecewise linear*).

[12] Il est à noter que λ_1, λ_2 et λ_3 dépendent de $\boldsymbol{x}$, d'où le paramètre à σ_P.

indication sur la distribution des points. Plus il est sphérique, moins claire est l'orientation, et donc possiblement moins bonne est l'estimation de la normale. Cette variation de surface donne aussi une indication sur la courbure de la surface, quoiqu'il existe de meilleures estimations pour calculer cette dernière [Miao *et al.* 2005b, Guennebaud et Gross 2007, Yang et Qian 2007].

Un dernier point à propos de l'estimation de normales concerne l'orientation. En effet, toute méthode basée sur l'ajustement d'un plan, voire d'une forme d'ordre supérieur, produit une normale à un *signe* près, *i.e.* si $\boldsymbol{n}_P(\boldsymbol{x})$ est normale à la fonction ajustée en $\boldsymbol{x}$, $-\boldsymbol{n}_P(\boldsymbol{x})$ l'est aussi. Pour assurer la continuité de la direction de la normale estimée, l'orientation doit être déterminée de façon cohérente entre points voisins. Hoppe *et al.* [1992] ont proposé une méthode de propagation d'orientation une fois que toutes les normales sont estimées. Ils construisent un graphe de proximité (ou graphe riemannien) en attribuant le poids $1 - |\boldsymbol{n}_P(\boldsymbol{p}_i)^\mathsf{T}\boldsymbol{n}_P(\boldsymbol{p}_j)|$ à chaque arête (i, j), $i, j \in \mathbb{Z}[1, |P|]$. Un arbre couvrant minimum est ensuite calculé. Un des sommets est choisi comme racine et l'orientation de sa normale est déterminée, puis propagée en parcourant l'arbre. À chaque nouveau sommet visité, si l'orientation de sa normale est opposée à celle de la normale du sommet parent, *i.e.* leur produit scalaire est négatif, elle est inversée. Si l'ensemble de points P représente un objet fermé, le sommet initial peut être choisi parmi ceux qui touchent à la frontière de la boîte englobante de P, et l'orientation de la normale est choisie de telle sorte qu'elle pointe vers l'extérieur de la boîte. Des variantes de cette méthode [Mitra et Nguyen 2003, Guennebaud et Gross 2007] utilisent des fonctions de poids différentes pour tenir compte, par exemple, d'informations sur la courbure.

Si une normale est déjà connue pour chaque point de P, nous pouvons alors déduire l'orientation de $\boldsymbol{n}_P(\boldsymbol{x})$ en testant le signe du produit scalaire entre $\boldsymbol{n}_P(\boldsymbol{x})$ et la normale donnée du point le plus proche dans le voisinage. Ceci suppose que les normales fournies sont fiables et cohérentes avec l'estimation.

2.3.2 Représentations purement par points

Dans sa forme la plus simple, un ensemble de points P forme une approximation constante par morceaux (*piecewise constant*) d'une surface. L'erreur de cette approximation est donc directement proportionnelle à l'espace entre les points. Une meilleure approximation, cette fois linéaire par morceaux, peut être obtenue en considérant l'information de la normale à chaque point. Accessoirement, un rayon peut aussi être associé à chaque point pour explicitement don-

ner l'étendue de son approximation. Ce rayon est utile principalement en rendu (§2.2.2) pour couvrir les trous entre les points.

Puisque l'une des principales méthodes de création d'ensembles de points est la numérisation 3D, il est possible que le résultat contienne du bruit. Au lieu de tenter d'extraire un bon échantillonnage avec des positions absolues, Kalaiah et Varshney [2003b, 2005] ont proposé plutôt une représentation non déterministe, où une PCA (*principal component analysis* ou analyse de composantes principales) hiérarchique partitionne la géométrie et ses attributs en un ensemble de distributions de probabilités gaussiennes locales. L'échantillonnage aléatoire de ces distributions de probabilités permet de reconstruire et de rendre une surface.

Avec les représentations purement par points, certains soins doivent être pris lorsque des opérations d'édition sont effectuées sur P. Une déformation peut possiblement produire des fentes si l'espacement entre les points devient trop grand. L'échantillonnage doit souvent être changé pour s'adapter à la forme de détails fins. Ainsi, une paramétrisation locale doit souvent être reconstruite pour effectuer une opération [Pauly et Gross 2001, Zwicker *et al.* 2002]. Cette paramétrisation est généralement donnée sous la forme d'une carte d'élévation, ce qui limite considérablement la capabilité des opérations. Pour des déformations de plus grande amplitude ou plus générales, il faut recourir à des méthodes d'interpolation [Pauly *et al.* 2003b, Guennebaud *et al.* 2004c, Guennebaud *et al.* 2004b, Guennebaud *et al.* 2005], ou à des représentations hybrides comme les surfaces MCM, que nous décrivons ci-après.

2.3.3 Surfaces MCM

Les surfaces MCM (moindres carrés mobiles, de *moving least squares* ou *MLS*) ont été présentées à la communauté graphique par Alexa *et al.* [2001, 2003], qui se sont eux-mêmes basés sur les ensembles stationnaires de Levin [1998, 2003]. Pour ce type de surface, un ensemble de points P correspond plutôt à des points de contrôle pour une surface implicitement définie ou, autrement dit, reconstruite. Cette dualité entre une représentation explicite (les points P) et implicite (la surface reconstruite) offre un outil puissant de modélisation. Les points P peuvent directement servir pour le rendu, et la surface permet des opérations bien définies sur P. Le principe fondamental des surfaces MCM est la projection sur une fonction polynomiale localement ajustée par moindres carrés mobiles.

Dans un cadre général, considérons P comme un ensemble de points dans un domaine ($\mathbb{R}^d$, où d est le nombre de dimensions). À chaque point $\boldsymbol{p}_i \in P, i \in \mathbb{Z}[1, |P|]$, une donnée observée

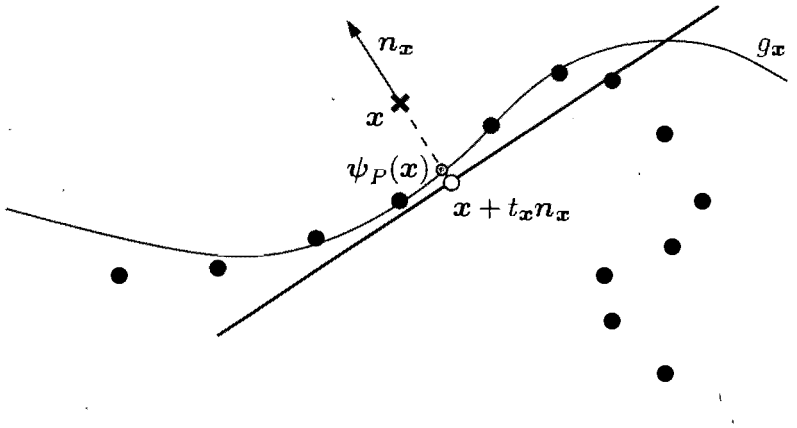


FIG. 2.2 – Projection de Alexa *et al.* [2001, 2003] : un plan perpendiculaire à une direction n_x et à une distance t_x du point x est d'abord estimé ; puis une fonction polynomiale g_x est ajustée aux points au-dessus du plan, centrée en $x + t_x n_x$; la projection $\psi_P(x)$ est alors l'intersection entre $g_x(0,0)$ et la droite de projection de x sur le plan.

$f_i \in \mathbb{R}$ est connue. Pour calculer une approximation $f(x)$, la méthode par moindres carrés classique procède en résolvant le problème de minimisation suivant :

$$\min_{f \in \Pi_m^d} \sum_{i=1}^{|P|} (f(p_i) - f_i)^2 \tag{2.13}$$

où Π_m^d est l'espace des polynômes de degré m définis sur $\mathbb{R}^d$. Il s'agit donc d'une approximation globale, indépendante du point d'évaluation x dans le domaine. En contrepartie, la méthode par moindres carrés mobiles calcule plutôt une fonction $f(x) = f_x(x)$, où f_x est calculée en résolvant

$$\min_{f_x \in \Pi_m^d} \sum_{i=1}^{|P|} (f_x(p_i) - f_i)^2 \theta(\|x - p_i\|) \tag{2.14}$$

où $\theta : \mathbb{R}^+ \mapsto \mathbb{R}^+$ est une fonction monotone décroissante. La localité de l'ajustement de $f_x(x)$ dépend du support de θ.

Nous présentons à la §2.3.3.1 comment le principe des moindres carrés mobiles est appliqué pour définir des surfaces. Nous abordons à la §2.3.3.2 les choix de θ, et nous présentons à la §2.3.3.3 certaines particularités en lien avec la présence de caractéristiques abruptes (*sharp features*) sur les surfaces.

2.3.3.1 Définitions

L'application exacte des MCM pour la définition d'une surface varie selon la méthode. Celle proposée par Alexa *et al.* [2001, 2003] procède en deux étapes. Soit $\boldsymbol{x}$ un point à proximité de la surface (ou de P). D'abord, un plan perpendiculaire à une direction $\boldsymbol{n_x}$ et à une distance $t_{\boldsymbol{x}}$ de $\boldsymbol{x}$ est calculé par MCM en résolvant le problème suivant :

$$\min_{\substack{\boldsymbol{n_x}\in\mathbb{S}^2 \\ t_{\boldsymbol{x}}\in\mathbb{R}}} \sum_{i=1}^{|P|} \big(\boldsymbol{n_x}^{\mathsf{T}}(\boldsymbol{p}_i - \boldsymbol{x} - t_{\boldsymbol{x}}\boldsymbol{n_x})\big)^2 \theta(\|\boldsymbol{p}_i - \boldsymbol{x} - t_{\boldsymbol{x}}\boldsymbol{n_x}\|). \tag{2.15}$$

Il est à noter qu'il s'agit d'un problème d'optimisation non linéaire car les variables de décision $\boldsymbol{n_x}$ et $t_{\boldsymbol{x}}$ apparaissent en argument à θ. L'étape suivante est d'ajuster une fonction polynomiale $g_{\boldsymbol{x}} \in \Pi^2_m$ définie au-dessus du plan également par MCM en résolvant

$$\min_{g_{\boldsymbol{x}}\in\Pi^2_m} \sum_{i=1}^{|P|} \big(g_{\boldsymbol{x}}(u_i, v_i) - \boldsymbol{n_x}^{\mathsf{T}}(\boldsymbol{p}_i - \boldsymbol{x} - t_{\boldsymbol{x}}\boldsymbol{n_x})\big)^2 \theta(\|\boldsymbol{p}_i - \boldsymbol{x} - t_{\boldsymbol{x}}\boldsymbol{n_x}\|) \tag{2.16}$$

où (u_i, v_i) est la valeur paramétrique de $\boldsymbol{p}_i$ dans le plan de la projection, avec $(0, 0)$ étant réservé pour $\boldsymbol{x} + t_{\boldsymbol{x}}\boldsymbol{n_x}$. Un opérateur de projection est alors donné par

$$\psi_P(\boldsymbol{x}) = \boldsymbol{x} + \big(t_{\boldsymbol{x}} + g_{\boldsymbol{x}}(0, 0)\big)\boldsymbol{n_x}. \tag{2.17}$$

La figure 2.2 illustre cette projection. La surface, notée $\mathcal{S}_P$, est définie comme l'ensemble des points stationnaires de ψ_P, *i.e.*

$$\mathcal{S}_P = \{\boldsymbol{x} \in \mathbb{R}^3 \mid \boldsymbol{x} = \psi_P(\boldsymbol{x})\}. \tag{2.18}$$

Une définition plus simple avec une formulation implicite et qui évite l'optimisation non linéaire a été proposée par Adamson et Alexa [2003a]. Un centroïde pondéré est d'abord défini ainsi :

$$\boldsymbol{a}_P(\boldsymbol{x}) = \frac{\sum_{i=1}^{|P|} \boldsymbol{p}_i \theta(\|\boldsymbol{x} - \boldsymbol{p}_i\|)}{\sum_{i=1}^{|P|} \theta(\|\boldsymbol{x} - \boldsymbol{p}_i\|)}. \tag{2.19}$$

Puis une estimation de la direction normale au point $\boldsymbol{x}$ est calculée comme ceci :

$$\boldsymbol{n}_P(\boldsymbol{x}) = \arg\min_{\boldsymbol{n}\in\mathbb{S}^2} \sum_{i=1}^{|P|} (\boldsymbol{n}^{\mathsf{T}}(\boldsymbol{x} - \boldsymbol{p}_i))^2 \theta(\|\boldsymbol{x} - \boldsymbol{p}_i\|). \tag{2.20}$$

Pour plus de robustesse en s'éloignant de la surface, il est préférable de plutôt la calculer ainsi :

$$\boldsymbol{n}_P(\boldsymbol{x}) = \arg\min_{\boldsymbol{n}\in\mathbb{S}^2} \sum_{i=1}^{|P|} (\boldsymbol{n}^\mathsf{T}(\boldsymbol{a}_P(\boldsymbol{x}) - \boldsymbol{p}_i))^2 \theta(\|\boldsymbol{x} - \boldsymbol{p}_i\|). \tag{2.21}$$

Cette minimisation peut être résolue par analyse de covariance pondérée en $\boldsymbol{x}$, de façon similaire à l'estimation de normales de l'équation (2.9), à partir de la matrice suivante :

$$\sum_{i=1}^{|P|} (\boldsymbol{p}_i - \boldsymbol{a}_P(\boldsymbol{x}))(\boldsymbol{p}_i - \boldsymbol{a}_P(\boldsymbol{x}))^\mathsf{T} \theta(\|\boldsymbol{x} - \boldsymbol{p}_i\|). \tag{2.22}$$

Si une normale $\boldsymbol{n}_i$ est fournie pour chaque point $\boldsymbol{p}_i \in P$, l'estimation peut être calculée par une moyenne pondérée :

$$\boldsymbol{n}_P(\boldsymbol{x}) = \frac{\sum_{i=1}^{|P|} \boldsymbol{n}_i \theta(\|\boldsymbol{x} - \boldsymbol{p}_i\|)}{\left\| \sum_{i=1}^{|P|} \boldsymbol{n}_i \theta(\|\boldsymbol{x} - \boldsymbol{p}_i\|) \right\|}. \tag{2.23}$$

Cette dernière est souvent plus robuste que l'équation (2.21), dans la mesure où les normales fournies soient fiables.

Remarque 2.3 (Définition de $\boldsymbol{n_P(x)}$) *À moins d'avis contraire, dans l'ensemble de cette thèse, nous ferons toujours usage de l'estimation de la normale $\boldsymbol{n}_P(\boldsymbol{x})$ définie à l'équation* (2.23).

Avec $\boldsymbol{a}_P(\boldsymbol{x})$ et $\boldsymbol{n}_P(\boldsymbol{x})$, la surface $\mathcal{S}_P$ est définie comme l'ensemble des zéros de la fonction

$$f_P(\boldsymbol{x}) = \boldsymbol{n}_P(\boldsymbol{x})^\mathsf{T}(\boldsymbol{x} - \boldsymbol{a}_P(\boldsymbol{x})) \tag{2.24}$$

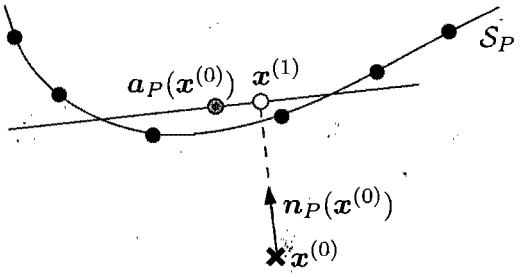
i.e.

$$\mathcal{S}_P = \{\boldsymbol{x} \mid f_P(\boldsymbol{x}) = 0\}. \tag{2.25}$$

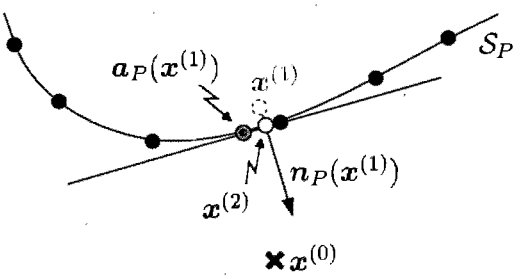
À partir de cette définition, un opérateur de projection peut être simplement calculé par le processus itératif suivant [Alexa et Adamson 2004] :

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - f_P(\boldsymbol{x}^{(t)})\boldsymbol{n}_P(\boldsymbol{x}^{(t)}) \tag{2.26}$$

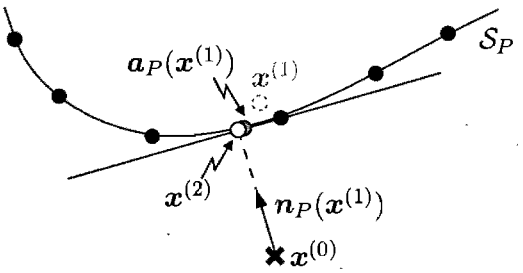
ce qui correspond à itérativement projeter $\boldsymbol{x}^{(t)}$ sur le plan passant par $\boldsymbol{a}_P(\boldsymbol{x}^{(t)})$ perpendiculaire à $\boldsymbol{n}_P(\boldsymbol{x}^{(t)})$, tel qu'illustré à la figure 2.3 (a) et (b). La projection est le point de convergence de



(a) Le point $x = x^{(0)}$ est projeté sur le plan passant par $a_P(x^{(0)})$ et perpendiculaire à $n_P(x^{(0)})$.



(b) La première variante de la procédure de projection poursuit en projetant de la même façon $x^{(1)}$.



(c) La variante quasi-orthogonale de la procédure de projection poursuit en projetant toujours $x^{(0)}$, mais en pondérant à partir de $x^{(t)}$.

FIG. 2.3 – Projection de Alexa et Adamson [2004].

ce processus, *i.e.*

$$\psi_P(x) = \psi_P(x^{(0)}),\ \psi_P(x^{(t)}) \begin{cases} x^{(t)} & \text{si } f_P(x^{(t)}) = 0\ ^{13} \\ \psi_P(x^{(t+1)}) & \text{sinon.} \end{cases} \tag{2.27}$$

En pratique, il suffit d'arrêter le processus lorsque $|f_P(x^{(t)})| < \epsilon$, pour une tolérance d'erreur $\epsilon > 0$, ce qui survient habituellement après peu d'itérations. Il peut être facilement vérifié que $\mathcal{S}_P$ (équation (2.25)) est définie de façon équivalente avec l'équation (2.18) en faisant usage de $\psi_P(x)$ de l'équation (2.27)[14] [Alexa et Adamson 2004], pourvu que la projection converge.

[13] *A priori*, il faudrait plutôt écrire la condition ainsi : si $x^{(t+1)} = x^{(t)}$. Or, si $x^{(t+1)} = x^{(t)}$, alors $f_P(x^{(t)})n_P(x^{(t)}) = 0$, mais lorsque $n_P(x^{(t)}) = \mathbf{0}$, en vertu de l'équation (2.24), $f_P(x^{(t)}) = 0$ aussi. Cette condition d'arrêt est d'ailleurs celle utilisée par Alexa et Adamson [2004].

[14] En anecdote, il se trouve que la procédure de projection utilisée dans *PointShop* [Zwicker *et al.* 2002] décrite par Pauly [2003], selon les descriptions de Amenta et Kil [2004a] et Dey et Sun [2005], définit une surface équivalente à celle de Adamson et Alexa [2003a]. La surface implicite de *PointShop* est en fait $f_P(x) \sum_{i=1}^{|P|} \theta(\|x - p_i\|)$, ce qui produit le même ensemble de zéros.

Une variante quasi-orthogonale de cette projection est, à chaque itération, de projeter le point initial $\boldsymbol{x} = \boldsymbol{x}^{(0)}$ plutôt que $\boldsymbol{x}^{(t)}$, *i.e.*

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(0)} - \boldsymbol{n}_P(\boldsymbol{x}^{(t)})^{\mathsf{T}}(\boldsymbol{x}^{(0)} - \boldsymbol{a}_P(\boldsymbol{x}^{(t)}))\boldsymbol{n}_P(\boldsymbol{x}^{(t)}). \tag{2.28}$$

Cette variante est illustrée à la figure 2.3 (a) et (c). Adamson et Alexa ont aussi donné une projection orthogonale, similaire à la précédente, mais au lieu de projeter orthographiquement $\boldsymbol{x}^{(0)}$ sur le plan, $\boldsymbol{x}^{(t+1)}$ est calculé par l'intersection de la droite passant par $\boldsymbol{x}^{(0)}$ de direction $\nabla f_P(\boldsymbol{x}^{(t)})$ avec le plan, *i.e.*

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(0)} - \frac{\boldsymbol{n}_P(\boldsymbol{x}^{(t)})^{\mathsf{T}}(\boldsymbol{x}^{(0)} - \boldsymbol{a}_P(\boldsymbol{x}^{(t)}))}{\boldsymbol{n}_P(\boldsymbol{x}^{(t)})^{\mathsf{T}}\nabla f_P(\boldsymbol{x}^{(t)})}\nabla f_P(\boldsymbol{x}^{(t)}). \tag{2.29}$$

Par contre, le calcul du gradient $\nabla f_P(\boldsymbol{x}^{(t)})$ est plus exigeant, et la différence entre le résultat du processus de l'équation (2.28) et celui de l'équation (2.29) n'est souvent pas suffisamment significative pour justifier le coût du second, particulièrement avec $\boldsymbol{n}_P(\boldsymbol{x})$ de l'équation (2.23). Il est à noter cependant que $\forall \boldsymbol{x} \in S_P$, par la définition de $f_P(\boldsymbol{x})$, $\nabla f_P(\boldsymbol{x})$ est la véritable normale [Alexa et Adamson 2004].

Un critère pour déterminer une surface avec frontière peut être simplement défini par un seuil sur la distance entre $\boldsymbol{a}_P(\boldsymbol{x})$ et $\boldsymbol{x}$ [Adamson et Alexa 2004], *i.e.*

$$S_P = \{\boldsymbol{x} \in \mathbb{R}^3 \mid f_P(\boldsymbol{x}) = 0 \wedge \|\boldsymbol{a}_P(\boldsymbol{x}) - \boldsymbol{x}\| < \epsilon_B\} \tag{2.30}$$

où ϵ_B est un seuil de tolérance défini par l'usager. Des indications sur le choix du seuil sont données par Adamson et Alexa [2004].

Kolluri [2005] a proposé une autre définition implicite d'une surface MCM, très similaire à celle de l'équation (2.24). Il s'est inspiré de la définition MCM implicite à partir d'un ensemble de polygones de Shen *et al.* [2004], mais considérant seulement un ensemble de points comme ceci :

$$f_P(\boldsymbol{x}) = \frac{\sum_{i=1}^{|P|} \boldsymbol{n}_i{}^{\mathsf{T}}(\boldsymbol{x} - \boldsymbol{p}_i)\theta(\|\boldsymbol{x} - \boldsymbol{p}_i\|)}{\sum_{i=1}^{|P|} \theta(\|\boldsymbol{x} - \boldsymbol{p}_i\|)}. \tag{2.31}$$

Cette définition suppose qu'une normale $\boldsymbol{n}_i$ est donnée pour chaque $\boldsymbol{p}_i \in P$. Si nous substituons $\boldsymbol{n}_i$ par $\boldsymbol{n}_P(\boldsymbol{x})$, alors nous retrouvons l'équation (2.24). Une projection d'un point $\boldsymbol{x}$ sur cette

surface peut être calculée par le processus itératif suivant [Dey et Sun 2005] :

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \frac{f_P(\boldsymbol{x}^{(t)})}{\|\nabla f_P(\boldsymbol{x}^{(t)})\|^2} \nabla f_P(\boldsymbol{x}^{(t)}) \tag{2.32}$$

ce qui correspond essentiellement au processus de l'équation (2.29) mais non orthogonale[15] et où la normale estimée serait égale au gradient.

Amenta et Kil [2004a] ont généralisé la définition de surfaces MCM par l'ensemble des minimums locaux d'une fonction d'énergie le long des directions données par un champ vectoriel. Soit $\varepsilon_P(\boldsymbol{x}, \boldsymbol{n})$ une fonction d'énergie en fonction de la position $\boldsymbol{x}$ et d'une direction unitaire $\boldsymbol{n}$ et soit[16]

$$\boldsymbol{n}_P(\boldsymbol{x}) = \arg \min_{\boldsymbol{n}_x \in \mathbb{S}^2} \varepsilon_P(\boldsymbol{x}, \boldsymbol{n}_x). \tag{2.33}$$

Amenta et Kil [2004a] définissent alors la surface par l'ensemble des zéros de la fonction

$$f_P(\boldsymbol{x}) = \boldsymbol{n}_P(\boldsymbol{x})^{\mathsf{T}} \left(\nabla_{\boldsymbol{y}} \varepsilon_P(\boldsymbol{y}, \boldsymbol{n}_P(\boldsymbol{x})) \big|_{\boldsymbol{x}} \right). \tag{2.34}$$

Leur procédure générale requiert cependant aussi une optimisation non linéaire.

Les méthodes présentées jusqu'ici peuvent être qualifiées d'approximations de premier ordre, *i.e.* elles se basent sur l'ajustement de plans[17]. Pour que ces approximations soient bonnes dans les zones très courbes d'une surface, la densité de points doit être suffisamment grande (§2.3.1.2). Ce n'est que récemment que des travaux ont été effectués sur des approximations d'un ordre supérieur. Yang et Kim [2007] ont repris les idées originales de Alexa *et al.* [2001, 2003] et Levin [1998, 2003] en combinant essentiellement les calculs reliés aux équations (2.15) et (2.16) en une seule minimisation pour directement ajuster un paraboloïde à proximité de $\boldsymbol{x}$ de coefficients $a_{\boldsymbol{x}}, b_{\boldsymbol{x}}, c_{\boldsymbol{x}}$ défini au-dessus d'un plan perpendiculaire à $\boldsymbol{n}_{\boldsymbol{x}}$ et à

[15] $\boldsymbol{x}^{(t)}$ est projeté, pas $\boldsymbol{x}^{(0)}$, et en substituant $\boldsymbol{x}^{(t)}$ par $\boldsymbol{x}^{(0)}$ dans le numérateur de la fraction dans l'équation (2.29), ce dernier devient en effet $f_P(\boldsymbol{x}^{(t)})$.

[16] Amenta et Kil [2004a] effectuent cette minimisation pour $\boldsymbol{n}_{\boldsymbol{x}} \in \mathbb{P}^2$, *i.e.* l'espace des directions unitaires non orientées. Mais à toute fin pratique, minimiser sur $\mathbb{P}^2$ ou $\mathbb{S}^2$ revient essentiellement au même car une solution dans $\mathbb{P}^2$ correspond à deux solutions possibles équivalentes dans $\mathbb{S}^2$.

[17] Dans le cas de la méthode générale de Amenta et Kil [2004a], il est possible d'obtenir des ordres supérieurs si la fonction d'énergie n'est pas définie en termes de plans.

une distance $t_{\boldsymbol{x}}$ de $\boldsymbol{x}$. Ces paramètres sont calculés en résolvant la minimisation suivante :

$$\min_{\substack{\boldsymbol{n_x}\in\mathbb{S}^2 \\ t_{\boldsymbol{x}},a_{\boldsymbol{x}},b_{\boldsymbol{x}},c_{\boldsymbol{x}}\in\mathbb{R}}} \sum_{i=1}^{|P|} \Big(\boldsymbol{n_x}^{\mathsf{T}}\boldsymbol{q}_i - t_{\boldsymbol{x}} - \frac{1}{2}\Big(a_{\boldsymbol{x}}(\boldsymbol{u_{n_x}}^{\mathsf{T}}\boldsymbol{q}_i)^2 + 2b_{\boldsymbol{x}}\boldsymbol{u_{n_x}}^{\mathsf{T}}\boldsymbol{q}_i\boldsymbol{v_{n_x}}^{\mathsf{T}}\boldsymbol{q}_i + c_{\boldsymbol{x}}(\boldsymbol{u_{n_x}}^{\mathsf{T}}\boldsymbol{q}_i)^2 \Big) \Big)^2 \theta(\|\boldsymbol{q}_i - t_{\boldsymbol{x}}\boldsymbol{n_x}\|) \quad (2.35)$$

où $\boldsymbol{q}_i = \boldsymbol{p}_i - \boldsymbol{x}$, et $\boldsymbol{u_{n_x}}$ et $\boldsymbol{v_{n_x}}$ sont deux vecteurs unitaires tels que $\{\boldsymbol{u_{n_x}}, \boldsymbol{v_{n_x}}, \boldsymbol{n_x}\}$ forme une base orthonormale de $\mathbb{R}^3$. L'opérateur de projection est alors

$$\psi_P(\boldsymbol{x}) = \boldsymbol{x} + t_{\boldsymbol{x}}\boldsymbol{n_x}. \quad (2.36)$$

Les paramètres $a_{\boldsymbol{x}}, b_{\boldsymbol{x}}, c_{\boldsymbol{x}}$ peuvent servir pour directement évaluer la courbure. Les calculs requis pour résoudre l'équation (2.35) sont cependant beaucoup plus complexes que ceux requis pour les équations (2.15) et (2.16).

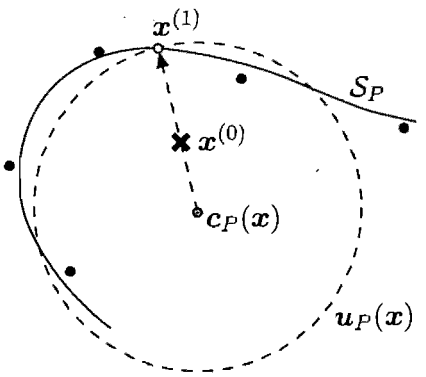


FIG. 2.4 – Projection de Guennebaud *et al.* [2007, 2008] : une sphère algébrique $\boldsymbol{u}_P(\boldsymbol{x})$ est ajustée aux points avoisinant $\boldsymbol{x} = \boldsymbol{x}^{(0)}$; $\boldsymbol{x}^{(0)}$ est projeté sur la sphère, donnant $\boldsymbol{x}^1$.

Un autre type d'approximation d'ordre supérieur est l'ajustement d'une sphère algébrique de Guennebaud *et al.* [2007, 2008]. Leur approche est plus en ligne avec la définition de Adamson et Alexa [2003a]. Leur surface est l'ensemble des zéros de la fonction

$$f_P(\boldsymbol{x}) = [1, \boldsymbol{x}^{\mathsf{T}}, \boldsymbol{x}^{\mathsf{T}}\boldsymbol{x}]\boldsymbol{u}_P(\boldsymbol{x}) \quad (2.37)$$

où $\boldsymbol{u}_P : \mathbb{R}^3 \mapsto \mathbb{R}^5$ est le vecteur de paramètres d'une sphère algébrique ajustée aux points P avec pondération MCM en fonction de la distance des points avec $\boldsymbol{x}$. Le centre $\boldsymbol{c}_P(\boldsymbol{x})$ et le

rayon $r_P(\boldsymbol{x})$ de la sphère peuvent être calculés ainsi :

$$\boldsymbol{c}_P(\boldsymbol{x}) = -\frac{1}{2\boldsymbol{e}_5{}^{\mathsf{T}}\boldsymbol{u}_P(\boldsymbol{x})}[\boldsymbol{e}_2, \boldsymbol{e}_3, \boldsymbol{e}_4]^{\mathsf{T}}\boldsymbol{u}_P(\boldsymbol{x}) \tag{2.38}$$

$$r_P(\boldsymbol{x}) = \sqrt{\boldsymbol{c}_P(\boldsymbol{x})^{\mathsf{T}}\boldsymbol{c}_P(\boldsymbol{x}) - \frac{\boldsymbol{e}_1{}^{\mathsf{T}}\boldsymbol{u}_P(\boldsymbol{x})}{\boldsymbol{e}_5{}^{\mathsf{T}}\boldsymbol{u}_P(\boldsymbol{x})}} \tag{2.39}$$

où $\boldsymbol{e}_j$, $j \in \mathbb{Z}[1,5]$, est le j^{e} vecteur unitaire de la base canonique de $\mathbb{R}^5$. Si $\boldsymbol{e}_5{}^{\mathsf{T}}\boldsymbol{u}_P(\boldsymbol{x}) = 0$, alors $\boldsymbol{u}_P(\boldsymbol{x})$ est le vecteur de paramètres d'un plan. En particulier, la normale du plan est

$$\frac{[\boldsymbol{e}_2, \boldsymbol{e}_3, \boldsymbol{e}_4]^{\mathsf{T}}\boldsymbol{u}_P(\boldsymbol{x}^{(t)})}{\|[\boldsymbol{e}_2, \boldsymbol{e}_3, \boldsymbol{e}_4]^{\mathsf{T}}\boldsymbol{u}_P(\boldsymbol{x}^{(t)})\|}.$$

Les auteurs suggèrent le processus suivant, illustré à la figure 2.4, pour calculer la projection d'un point $\boldsymbol{x} = \boldsymbol{x}^{(0)}$ sur la surface :

$$\boldsymbol{x}^{(t+1)} = \begin{cases} \boldsymbol{c}_P(\boldsymbol{x}^{(t)}) + \frac{r_P(\boldsymbol{x}^{(t)})}{\|\boldsymbol{x}^{(0)} - \boldsymbol{c}_P(\boldsymbol{x}^{(t)})\|}(\boldsymbol{x}^{(0)} - \boldsymbol{c}_P(\boldsymbol{x}^{(t)})) & \text{si} \quad \boldsymbol{e}_5{}^{\mathsf{T}}\boldsymbol{u}_P(\boldsymbol{x}^{(t)}) \neq 0 \\ \boldsymbol{x}^{(0)} - \frac{f_P(\boldsymbol{x}^{(t)})}{\|[\boldsymbol{e}_2, \boldsymbol{e}_3, \boldsymbol{e}_4]^{\mathsf{T}}\boldsymbol{u}_P(\boldsymbol{x}^{(t)})\|^2}[\boldsymbol{e}_2, \boldsymbol{e}_3, \boldsymbol{e}_4]^{\mathsf{T}}\boldsymbol{u}_P(\boldsymbol{x}^{(t)}) & \text{si} \quad \boldsymbol{e}_5{}^{\mathsf{T}}\boldsymbol{u}_P(\boldsymbol{x}^{(t)}) = 0. \end{cases} \tag{2.40}$$

Ce type de surface MCM offre plusieurs avantages intéressants, dont une estimation de courbure (le rayon de la sphère), mais surtout l'échantillonnage plus faible requis pour bien représenter la surface. Certains soins doivent être pris cependant pour assurer la robustesse numérique des algorithmes [Guennebaud 2007].

Un dernier type de surface à base de projection est celle de Lipman *et al.* [2007b]. Leur projection se démarque des autres car elle projette un ensemble de points X sur P plutôt qu'un seul point à la fois. La projection de X est l'ensemble fixe d'une fonction $F_P(X)$ calculant un ensemble de points minimisant une fonction de coût qui estime l'écart de chaque point par rapport à la médiane locale autour du point et la proximité entre points voisins. Les auteurs donnent une méthode itérative relativement simple à implémenter pour calculer cette projection, mais nous n'entrerons pas davantage dans les détails. Bien que leur projection bénéficie des avantages de ne nécessiter aucun ajustement de fonctions, de produire des points projetés bien distribués et d'être robuste, projeter un ensemble de points plutôt qu'un seul à la fois peut aussi être limitatif dans le choix d'applications, en particulier si une projection orthogonale est requise. De plus, pour évaluer une normale à un point projeté, il faut recourir à d'autres méthodes.

Remarque 2.4 (Définitions de $\mathcal{S}_P$ et $\psi_P(x)$) *À moins d'avis contraire, dans les travaux de cette thèse, nous faisons usage de la définition de surfaces MCM de Adamson et Alexa (équations* (2.19) *à* (2.28)*). La projection utilisée est généralement la version quasi-orthogonale (équation* (2.28)*). Nous avons préféré cette définition de surface car elle offre une bonne balance entre la robustesse, la souplesse, le temps d'exécution et la simplicité d'implémentation. La définition de Guennebaud et al. [2007, 2008] serait sans doute un meilleur choix, mais notre connaissance de celle-ci est venue trop tardivement dans la progression des travaux de cette thèse.*

2.3.3.2 Fonctions de pondération

Levin [1998, 2003] a démontré que les surfaces MCM sont C^∞, pourvu que θ soit aussi C^∞. Un choix naturel pour θ est une fonction gaussienne

$$\theta(d) = \mathrm{e}^{-9d^2/h^2} \tag{2.41}$$

où h correspond à la taille des caractéristiques (*feature size*) de la surface[18]. Puisque la valeur de $\theta(d)$ est négligeable pour $d > h$, nous pouvons forcer un support fini en la définissant plutôt ainsi :

$$\theta(d) = \begin{cases} \mathrm{e}^{-9d^2/h^2} & \text{si} \quad 0 \leq d \leq h \\ 0 & \text{si} \quad d > h. \end{cases} \tag{2.42}$$

D'autres choix populaires pour θ sont les fonctions radiales de Wendland [1995], en particulier

$$\theta(d) = \begin{cases} \left(1 - \frac{d}{h}\right)^4 \left(\frac{4d}{h} + 1\right) & \text{si} \quad 0 \leq d \leq h \\ 0 & \text{si} \quad d > h. \end{cases} \tag{2.43}$$

Guennebaud et Gross [2007] ont proposé la fonction suivante :

$$\theta(d) = \begin{cases} \left(1 - \frac{d^2}{h^2}\right)^4 & \text{si} \quad 0 \leq d \leq h \\ 0 & \text{si} \quad d > h \end{cases} \tag{2.44}$$

ce qui permet d'éviter la racine carrée dans le calcul de $\|\boldsymbol{x} - \boldsymbol{p}_i\|$.

[18] En considérant $\theta(d) = \mathrm{e}^{-d^2/h^2}$, s'inspirant de la règle 65-95-99.7 des distributions gaussiennes (ou normales) [Moore et McCabe 1999], nous pouvons substituer h par $h/3$ pour obtenir une meilleure représentation de la taille des caractéristiques, d'où le 9 devant d^2 dans l'équation (2.41).

Si l'interpolation des points P est désirée, il faut utiliser une fonction de poids avec une singularité à l'origine. Un exemple est la fonction suivante [Adamson et Alexa 2006b] :

$$\theta(d) = \begin{cases} \left(\ln \frac{d}{h}\right)^2 - \left(\frac{d}{h}\right)^2 + 2\frac{d}{h} - 1 & \text{si} \quad 0 \leq d \leq h \\ 0 & \text{si} \quad d > h. \end{cases} \tag{2.45}$$

Par contre, ceci peut créer des oscillations dans la surface reconstruite, à moins de modifier légèrement la façon de calculer $\boldsymbol{a}_P(\boldsymbol{x})$ [Alexa et Adamson 2008] comme ceci :

$$\boldsymbol{a}_P(\boldsymbol{x}) = \frac{\sum_{i=1}^{|P|}(\boldsymbol{x} - \boldsymbol{n}_i{}^{\mathsf{T}}(\boldsymbol{x} - \boldsymbol{p}_i)\boldsymbol{n}_i)\theta(\|\boldsymbol{x} - \boldsymbol{p}_i\|)}{\sum_{i=1}^{|P|}\theta(\|\boldsymbol{x} - \boldsymbol{p}_i\|)}. \tag{2.46}$$

Ceci requiert l'information de normale à chaque point. Les approximations d'ordre supérieur n'exhibent pas le problème d'oscillations [Guennebaud et Gross 2007].

Remarque 2.5 (Définition de $\theta(d)$) *Dans nos travaux, nous avons fait exclusivement usage de $\theta(d)$ défini à l'équation* (2.42). *Nos expérimentations avec d'autres fonctions n'apportaient pas de changements significatifs, même en temps d'exécution.*

Klein et Zachmann [2004c, 2004b] ont noté un problème associé à la distance euclidienne utilisée en paramètre à $\theta(d)$, *i.e.* $\|\boldsymbol{x} - \boldsymbol{p}_i\|$. Si $\boldsymbol{x}$ se trouve à proximité de l'axe médian d'une zone très courbe de la surface, la différence de pondération des points voisins de $\boldsymbol{x}$ dans P ne sera pas très grande. Or dans un tel cas, ceux-ci sont distribués de façon plus uniforme autour de $\boldsymbol{x}$. La surface reconstruite, $\mathcal{S}_P$, peut alors se comporter de façon inattendue, ou l'erreur d'approximation peut être plus grande qu'espérée. Leur idée est d'utiliser une distance géodésique plutôt qu'euclidienne. La distance géodésique est la longueur du chemin le plus court *sur la surface* entre deux points de $\mathcal{S}_P$. Comme $\boldsymbol{x}$ n'est généralement pas sur $\mathcal{S}_P$, la distance euclidienne entre $\boldsymbol{x}$ et son point le plus proche sur $\mathcal{S}_P$ est ajoutée. Cependant, puisque $\mathcal{S}_P$ n'est pas connue, cette distance est approximée par

$$\|\boldsymbol{x} - \boldsymbol{p}\|_{\text{geo}} = \min_{\boldsymbol{q} \in \mathcal{N}_P(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{q}\| + d_{\mathcal{G}}(\boldsymbol{q}, \boldsymbol{p}) \tag{2.47}$$

où $d_{\mathcal{G}}$ est la longueur du chemin le plus court entre deux sommets du graphe de proximité $\mathcal{G}$. Ce graphe est une variante des graphes de sphères d'influence [Boyer *et al.* 2000, Michael et Quint 2003].

Le problème souligné par Klein et Zachmann est en fait relié au domaine de ψ_P [Amenta et Kil 2004b]. Malgré que leur méthode peut s'en tirer assez bien avec un faible échantillonnage, elle souffre des mêmes problèmes associés au voisinage symétrique (§2.3.1.1), sans compter le coût d'évaluation un peu plus grand[19]. Pour des points sortant du domaine et pour un échantillonnage suffisant, nous pouvons nous en tirer assez bien avec des heuristiques similaires à celle utilisée par Pauly *et al.* [2003b], *i.e.* la projection de $\boldsymbol{x}$ est approximée simplement par son point le plus proche de P. Nous pouvons améliorer cette approximation en considérant plutôt ce point comme $\boldsymbol{x}^{(1)}$ dans l'itération décrite à l'équation (2.28).

Si le support de θ est fini, plusieurs termes des sommations pondérées de cette section s'annuleront. Si h est le paramètre de support de θ, alors tout point de P à une distance plus grande que h de $\boldsymbol{x}$ n'aura aucune influence. Il est alors possible de substituer les bornes des sommations ($i \in \mathbb{Z}[1, |P|]$) par

$$i \in \{j \in \mathbb{N} \mid \boldsymbol{p}_j \in \mathcal{N}_P^h(\boldsymbol{x})\}. \tag{2.48}$$

Puisque que $\mathcal{S}_P$ est une approximation (sauf si θ est choisie pour interpoler), le support h correspond en quelque sorte à la taille du noyau d'un filtre passe-bas. Par contre, si la densité des points P n'est pas uniforme sur toute la surface, l'effet de lissage ne sera pas uniforme, mais possiblement beaucoup plus grand dans les zones de haute densité. Le paramètre h devrait donc plutôt être une fonction $h_P(\boldsymbol{x})$ dépendant de la densité à proximité de $\boldsymbol{x}$ [Pauly *et al.* 2002a, Pauly *et al.* 2003b]. Un choix simple est le suivant [Pauly *et al.* 2002a] :

$$h_P(\boldsymbol{x}) = \max_{\boldsymbol{q} \in \mathcal{N}_P(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{q}\|. \tag{2.49}$$

Avec cette définition, le voisinage $\mathcal{N}_P(\boldsymbol{x})$ peut être réutilisé dans les bornes de l'équation (2.48) à la place du voisinage euclidien. Une autre approche pour calculer $h_P(\boldsymbol{x})$ est en fonction de l'erreur de l'approximation MCM [Lipman *et al.* 2006, Wang *et al.* 2008].

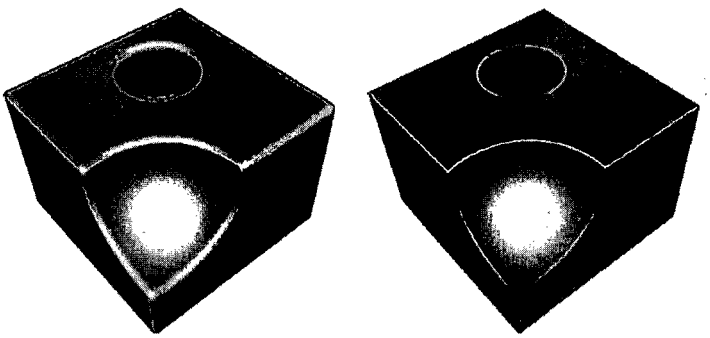
Un contrôle plus précis par l'usager peut être obtenu en donnant explicitement une valeur h_i pour chaque $\boldsymbol{p}_i \in P$ [Amenta et Kil 2004a]. Ainsi, en évaluant $\theta(\|\boldsymbol{x} - \boldsymbol{p}_i\|)$, le paramètre choisi est h_i. Suivant cette ligne d'idées, Adamson et Alexa [2006a] ont observé que pour des distributions irrégulières de points, une fonction de poids anisotropique peut donner une meilleure reconstruction. Pour ce faire, ils calculent un ellipsoïde $\mathcal{E}_i$ pour chaque $\boldsymbol{p}_i \in P$, qui

[19] Les auteurs ont montré une méthode pour précalculer une carte des chemins les plus courts par paires avec un stockage en $O(|P|)$. Leur évaluation de d_G peut donc être très rapide.

représente la distribution des points autour de $\boldsymbol{p}_i$. La pondération est alors effectuée en fixant $h = 1$ puis en transformant $\boldsymbol{x} - \boldsymbol{p}_i$ par $\mathcal{E}_i$, *i.e.* le poids est $\theta(\|\mathcal{E}_i^{-1}(\boldsymbol{x} - \boldsymbol{p}_i)\|)$. Un problème associé aux paramètres dépendant des $\boldsymbol{p}_i$ est que suivant toute déformation de P, tout h_i ou $\mathcal{E}_i$ doit être recalculé, selon la localité de la déformation. De plus, pour être exacte, la substitution des bornes suggérées à l'équation (2.48) n'est plus valide, ni même en utilisant un voisinage-k. Mais en pratique, si la densité ne change pas trop abruptement, un voisinage-k s'avère être bonne approximation.

Remarque 2.6 (Définition de $h_P(\boldsymbol{x})$) *Pour cette thèse, nous faisons usage de $h_P(\boldsymbol{x})$ de l'équation* (2.49) *et de l'heuristique d'estimation de $\boldsymbol{x}^{(1)}$ pour $\boldsymbol{x}$ éloigné de P. De plus, lors du processus de projection, si $\boldsymbol{x}^{(0)}$ (ou $\boldsymbol{x}^{(1)}$) est près de la surface, alors $\mathcal{N}_P(\boldsymbol{x}^{(t)})$ changera très peu, et donc les bornes de l'équation* (2.48) *aussi changeront peu. Nous pouvons même négliger ces changements et fixer le voisinage à $\mathcal{N}_P(\boldsymbol{x}^{(0)})$ (ou $\mathcal{N}_P(\boldsymbol{x}^{(1)})$), ce qui permet de nous en tirer avec une ou deux évaluations du voisinage seulement pour le calcul d'une projection.*

2.3.3.3 Caractéristiques abruptes



(a) (b)

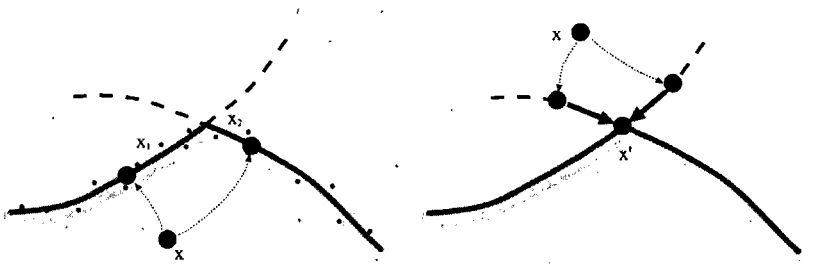
FIG. 2.5 – Caractéristiques abruptes et surfaces MCM : (a) les surfaces MCM produisent naturellement des surfaces lisses, perdant ainsi la forme des caractéristiques abruptes ; (b) des traitements particuliers doivent être apportés pour les préserver. Les images proviennent de [Fleishman *et al.* 2005].

Toute définition de surfaces MCM présentée jusqu'ici concerne seulement des surfaces lisses. Ainsi, toute caractéristique abrupte apparente dans P aura une allure arrondie dans $\mathcal{S}_P$, tel qu'illustré à la figure 2.5. Pour les représentations purement par points, ces caractéristiques sont exprimées par des paires de demi-*surfels* coupés à leur intersection [Pauly *et al.* 2003b], mais l'approximation $\mathcal{S}_P$ n'en produira pas pour autant des caractéristiques abruptes. Une

simple modification à la pondération a été suggérée par Miao *et al.* [2005a], qui permet de mieux les préserver, pourvu qu'une normale $\boldsymbol{n}_i$ soit fournie pour chaque $\boldsymbol{p}_i \in P$. Chaque facteur de pondération $\theta(\|\boldsymbol{x} - \boldsymbol{p}_i\|)$ est remplacé par

$$\theta(\|\boldsymbol{x} - \boldsymbol{p}_i\|)\vartheta(|\boldsymbol{n}_i{}^{\mathsf{T}}(\boldsymbol{x} - \boldsymbol{p}_i)|) \tag{2.50}$$

où ϑ est une autre fonction de même nature que θ, mais pour laquelle le support est fixé par l'usager. Bien que cette méthode préserve beaucoup mieux les caractéristiques abruptes, ces dernières sont tout de même encore approximées. Autrement, il y a deux approches principales pour gérer les caractéristiques abruptes : par des opérations d'intersection entre surfaces lisses [Fleishman *et al.* 2005, Adamson et Alexa 2006b, Guennebaud et Gross 2007] et par l'ajustement local de fonctions avec discontinuités de dérivées [Reuter *et al.* 2005, Lipman *et al.* 2007a].



(a) (b)

FIG. 2.6 – Gestion des caractéristiques abruptes avec opérations d'intersection entre surfaces lisses : (a) un point $\boldsymbol{x}$ est projeté sur chaque surface et la projection la plus près est conservée ; (b) si les projections sont invalides, alors elles sont projetées sur l'arête. Les images proviennent de [Fleishman *et al.* 2005].

Pour la première, l'ensemble de points P est partitionné selon la sous-surface à laquelle appartiennent ses points. S'il y a n sous-surfaces, alors $P = \bigcup_{i=1}^{n} P_i$ tel que $\forall i \neq j, P_i \cap P_j = \emptyset$. Si les points tombant à l'intérieur du support de θ autour de $\boldsymbol{x}$ n'appartiennent pas au même sous-ensemble, alors $\psi_{P_i}(\boldsymbol{x})$ est calculé pour tout P_i ayant au moins un point dans le support, et la projection la plus près de $\boldsymbol{x}$ est le résultat. Si une projection tombe dans une zone invalide, elle peut être soit rejetée ou alors un processus de projection sur la crête (ou le coin) peut être utilisé [Pauly *et al.* 2003b]. La figure 2.6 montre la procédure de projection. Le partitionnement de P peut être fait manuellement, par diverses analyses statistiques [Fleishman *et al.* 2005,

Guennebaud et Gross 2007], ou par un complexe hiérarchique [Adamson et Alexa 2006b], *a priori* construit aussi manuellement.

La seconde approche se base sur les surfaces de Alexa *et al.* [2001, 2003] et Levin [1998, 2003]. La projection commence avec l'évaluation d'un plan de référence (équation (2.15)). Ensuite, au lieu de simplement ajuster une fonction polynomiale (équation (2.16)), une fonction tenant compte des discontinuités de dérivées est ajustée. Reuter *et al.* [2005] font usage d'approximations avec un noyau reproducteur enrichi [Joyot *et al.* 2005]. Les discontinuités à reproduire doivent cependant être explicitement spécifiées par l'usager. Dans le cas de Lipman *et al.* [2007a], un ajustement similaire à l'équation (2.16) est effectué, mais l'ajustement est effectué par morceaux, où chacun est déterminé en fonction d'une courbe traçant les singularités, lesquelles sont approximées à partir d'un champ indicateur de singularités, lui-même calculé à base d'estimations des dérivées.

2.4 Traitement de géométrie avec points

Depuis l'avènement des représentations par points, en plus des techniques de rendu, une panoplie de travaux ont été effectués sur le traitement de géométrie basé sur ces représentations. Plusieurs ont été consacrés à illustrer l'avantage de la représentation et d'autres par nécessité d'adaptation depuis d'autres représentations, typiquement les maillages polygonaux, généralement par souci de complétude sur le type d'opération possible sur les représentations par points. Dans cette section, nous ne donnons pas une description de chacun des travaux comme nous l'avons fait à la section précédente, d'abord parce qu'ils sont trop nombreux, mais surtout parce qu'une telle étude d'ensemble dépasserait la portée de cette thèse. Nous détaillerons les travaux concernés au besoin dans les sections appropriées des prochains chapitres. Ici, nous nous contentons d'énumérer un sous-ensemble représentatif des travaux du domaine pour illustrer l'utilisation et les avantages des représentations par points.

Puisqu'une des principales (et premières) sources de création d'ensembles de points est la numérisation 3D, la plupart des travaux initiaux ont été faits pour leur traitement. Ainsi, nous retrouvons des méthodes de traitement de signaux ou de réduction de bruit ou de lissage [Pauly et Gross 2001, Clarenz *et al.* 2004a, Clarenz *et al.* 2004b, Dey *et al.* 2004b, Lange et Polthier 2005, Miao *et al.* 2005a, Schall *et al.* 2005, Hu *et al.* 2006, Xiao *et al.* 2006, Moenning *et al.* 2007]. En lien avec celles-ci sont des méthodes d'extraction de caractéristiques [Gumhold *et al.* 2001, Pauly *et al.* 2003a, Li et Guskov 2005, Daniels *et al.* 2007, Demarsin *et al.* 2007],

pouvant servir à l'analyse de points saillants pour, par exemple, l'alignement de deux surfaces ou pour un rendu non photoréaliste. Aussi reliée est la segmentation d'ensembles de points [Clarenz *et al.* 2004b, Dey *et al.* 2004a, Yamazaki *et al.* 2006], qui est souvent utilisée pour la correspondance ou comparaison de formes, parmi d'autres méthodes [Collins *et al.* 2004, Mémoli et Sapiro 2004, de Silva et Carlsson 2004, Mémoli 2007, Lien 2007, Singh *et al.* 2007]. Une force des représentations par points dans ce contexte est la résistance au bruit dans les données. Un maillage reconstruit contient souvent des inconsistances (fentes, croisements, erreurs topologiques), ce qui pose des difficultés dans l'élaboration d'algorithmes robustes pour traiter sa géométrie. Pour les représentations par points, la plupart de ces inconsistances ne peuvent pas se produire, par absence d'information de connectivité. En substituant cette dernière par le voisinage (§2.3.1), des algorithmes de traitement efficaces peuvent être élaborés.

La numérisation 3D produit souvent une très grande quantité de points, parfois même trop grande pour que l'objet entier puisse être chargé entièrement en mémoire centrale [Levoy *et al.* 2000]. La simplification d'ensembles de points [Cohen *et al.* 2001, Pauly *et al.* 2002a, Fleishman *et al.* 2003a, Linsen 2001, Linsen et Prautzsch 2003, Moenning et Dodgson 2003, Moenning et Dodgson 2004, Wu et Kobbelt 2004, Boubekeur *et al.* 2006, Kim *et al.* 2006, Reniers et Telea 2006, Singh et Narayanan 2006, Yu et san Wong 2006, Wang *et al.* 2007] est donc une nécessité pour réduire les ressources requises, autant pour traiter un ensemble de points que pour le rendu efficace. La simplification procède souvent par une forme de rééchantillonnage de la surface d'un objet. Avec des représentations à base de maillages, ceci implique la reconstruction (ou transformation, selon la méthode) de l'information de connectivité, mais pour les représentations par points, il n'y a rien de particulier à faire en plus. Pour certaines configurations géométriques, notamment des structures étroites ou peu courbes, afin d'assurer des voisinages représentatifs, il faut plus de points que de sommets dans un maillage. Ainsi, en complément, ou parfois en alternative à la simplification, des techniques de compression d'ensembles de points [Ochotta et Saupe 2004, Waschbüsch *et al.* 2004, Krüger *et al.* 2005, Kitago et Gopi 2006, Huang *et al.* 2006, Schnabel et Klein 2006, Hubo *et al.* 2007, Schnabel *et al.* 2007] peuvent être aussi employées pour réduire leur stockage ou l'usage de bande passante pour la transmission sur un réseau ou sur le bus d'un ordinateur. Celles-ci peuvent bénéficier de la simplicité des structures requises[20] pour efficacement encoder un ensemble de points.

[20] *A priori*, il ne s'agit que d'une simple liste de points. Les structures d'accélération pour le calcul de voisinages ne font pas partie intégrante de la représentation. Elle sont dynamiquement construites au besoin. Autrement, il s'agit de structures hiérarchiques généralement plus simples que les structures de connectivité des maillages.

Dans le flot de production d'objets virtuels, la numérisation 3D sert habituellement à fournir la géométrie de base. En dehors des considérations de "nettoyage" des résultats, leur édition est souvent requise pour rencontrer les besoins finaux. Les représentations par points ont aussi leurs avantages pour développer des outils de modélisation et des systèmes interactifs [Zwicker *et al.* 2002, Adams et Dutré 2003, Pauly *et al.* 2003b, Adams et Dutré 2004, Adams *et al.* 2004, Guo *et al.* 2004a, Guo *et al.* 2004b, Liu *et al.* 2004c, Reuter *et al.* 2004, Weyrich *et al.* 2004, Boubekeur et Schlick 2006, Miao *et al.* 2006, Boubekeur *et al.* 2007, Wand *et al.* 2007, Miao *et al.* 2008]. Elles peuvent être aisément restructurées sans avoir à tenir compte des conditions de variété (*manifold*), ce qui est bénéfique pour toute opération qui requiert un rééchantillonnage fréquent de la géométrie. Un exemple de ce type d'opération est la sculpture de détails fins réalisée dans *PointShop* [Zwicker *et al.* 2002], un système interactif de traitement de points fonctionnant dans les mêmes principes que les logiciels de traitement d'images tels *The GIMP* [GIM]. La peinture sur des objets est un autre exemple d'opération pouvant profiter de la capacité de restructuration [Zwicker *et al.* 2002, Adams *et al.* 2004, Reuter *et al.* 2004, Boubekeur et Schlick 2006]. En effet, en plus de la géométrie, les points peuvent aussi servir pour échantillonner la texture colorée à la surface d'un objet. Ceci permet ainsi de peindre les objets sans devoir recourir à une paramétrisation globale. Les opérations CSG (§2.1.1.6) peuvent aussi grandement bénéficier des représentations par points [Adams et Dutré 2003, Pauly *et al.* 2003b, Adams et Dutré 2004] car il n'y a aucun besoin de "coudre" les surfaces à leurs intersections pour maintenir la consistance de la représentation. Il est aussi bien connu que les approches d'édition multirésolutions peuvent, entre autres, produire des déformations préservant bien les détails plus fins à la surface. La multirésolution n'a cependant pas été aussi étudiée pour les ensembles de points que pour les maillages ou surfaces de subdivision. Nous y reviendrons aux chapitres 5 et 6.

Les représentations par points ont aussi trouvé une niche en animation [Keiser *et al.* 2004, Klein et Zachmann 2004a, Müller *et al.* 2004, Pauly *et al.* 2004b, Keiser *et al.* 2005, Pauly *et al.* 2005a, Wicke *et al.* 2005b, Steinemann *et al.* 2006, Wicke *et al.* 2006, Zhang *et al.* 2006, Solenthaler *et al.* 2007, Steinemann *et al.* 2007], en particulier parce qu'elles peuvent aisément accommoder les changements de topologie des corps déformables. De plus, la dualité des surfaces MCM (§2.3.3) procure le même avantage que les surfaces implicites (§2.1.1.4) pour la détection de collision.

Pour faire un bilan de tout ceci, l'ensemble des travaux qui ont été réalisés jusqu'à maintenant montre amplement non seulement l'utilité des représentations par points pour le traitement de géométrie, de la création à l'édition, mais aussi comment elles peuvent être avantageuses par rapport aux représentations classiques pour plusieurs types d'opérations. Les avantages des représentations par points découlent toutes de l'absence du maintien explicite de l'information de connectivité. Pour la création d'objets, l'avantage principal est au niveau de la composition de diverses surfaces en évitant la "couture". Pour l'édition, il s'agit de la facilité de restructuration et la réduction d'inconsistances suite à une opération. Nous mettrons à profit ces avantages dans une méthode de génération de textures géométriques et de représentation multirésolution que nous présenterons prochainement (chapitres 4 et 6 respectivement).

Nous finissons par noter que les travaux sur les représentations par points ont d'abord été effectués au sein de la communauté graphique. Ce n'est que tout récemment qu'il y a eu des premières investigations pour des applications en manufacture directe à partir d'ensembles de points [Yang et Qian 2008]. Un nouveau champ d'applications vient ainsi de s'ouvrir.

Chapitre 3

Génération de textures et de géométrie

Les différentes méthodes de création d'objets décrites au chapitre précédent, à la §2.1.2, ont toutes leurs lots de problèmes lorsqu'il est question de produire des détails géométriques à la surface des objets. L'approche traditionnelle exige beaucoup de travail, la modélisation procédurale est souvent trop spécialisée ou peu intuitive (notamment pour les systèmes-L), et la numérisation 3D n'arrive souvent pas à bien capturer les détails, entre autres soit parce que le niveau de bruit est trop élevé, ou parce qu'il y a trop d'occultations.

Toutes ces méthodes demeurent cependant raisonnables d'usage à petite échelle, donnant ainsi un moyen de produire un petit échantillon de détails que nous voudrions par la suite appliquer sur la surface d'un objet, en partie ou partout. Malheureusement, couvrir la surface par un simple pavage de cet échantillon fait ressortir des motifs pour la plupart visuellement inacceptables. Ainsi, nous souhaiterions plutôt appliquer les détails géométriques sur une surface de telle sorte à ce que le résultat soit similaire à l'échantillon mais sans en être une simple répétition.

Cette idée de génération à partir d'un exemple a été largement abordée dans le domaine de la génération de textures, qui forme la base de notre source d'inspiration pour la méthode que nous présenterons au chapitre suivant. Il convient donc d'abord de faire un survol de la littérature dans ce domaine, ce que nous présentons à la §3.1. Un certain nombre de travaux, que nous décrivons à la §3.2, ont aussi abordé ce problème pour traiter de la géométrie, ce qui correspond au même problème nous concernant en partie dans cette thèse.

3.1 Génération de textures

Un moyen pour enrichir l'apparence des modèles géométriques est d'associer une *couleur* à tout élément de surface. Ce changement simule le matériel de la surface et est ce que nous appelons, dans la communauté graphique, une texture colorée. Cette association est appelée le placage de textures (*texture mapping*) [Catmull 1974, Blinn et Newell 1976]. Pour être en mesure d'effectuer une telle association, il faut une paramétrisation de la surface pour définir le domaine de la texture. Il faut ensuite évaluer la couleur pour chaque point du domaine. Une des méthodes les plus populaires pour représenter une texture est une image de type trame et chaque pixel, appelé dans ce cas-ci *texel*, est associé à un point du domaine paramétrique (ou vice versa). Une autre méthode pour produire une texture est d'évaluer une fonction définie sur ce domaine. Ce type de méthode est dit procédural [Ebert *et al.* 1998].

La couleur n'est pas le seul attribut pouvant être appliqué ainsi. Un texel peut aussi représenter une perturbation de la normale pour simuler de la micro-géométrie [Blinn 1978], ou alors un déplacement des points de la surface [Cook 1984], ce qui est appelé une carte de déplacement. Dans ce cas-ci, la texture est dite géométrique, *i.e.* elle change l'apparence géométrique de l'objet. D'autres possibilités d'attributs incluent la transparence et les propriétés de réflectance de la surface. Ceci permet de représenter des surfaces avec des BRDF (*bidirectional reflectance distribution function* ou distribution du coefficient de réflexion bidirectionnel)[1] non uniformes sur la surface. La BTF (*bidirectional texture function* ou fonction bidirectionnelle de textures) [Dana *et al.* 1999][2]est le terme employé pour ce type de texture. En bref, le placage de textures peut être employé pour appliquer tout attribut sur une surface.

Il est à noter que la paramétrisation naturelle d'une surface a deux dimensions, et en conséquence lorsque nous parlons de texture, nous sous-entendons un modèle défini en 2D, tel une image. Cependant, rien n'empêche d'avoir une paramétrisation avec un nombre arbitraire de paramètres. Par exemple, certains types de matériaux sont mieux représentés par des textures 3D, telles le bois ou le marbre. La texture est alors définie par une grille de *voxels* au lieu d'une image[3].

[1] Une BRDF est une fonction donnant la distribution de la réflexion de la lumière sur une surface en fonction des angles d'incidence de la lumière et du point de vue. Il s'agit d'un modèle fréquemment employé pour représenter les propriétés de réflectance du matériau d'une surface.

[2] Il s'agit en fait d'une BRDF avec un paramètre additionnel : la position sur la surface, ce qui peut être vu comme une texture de BRDFs.

[3] Nous rappelons qu'une grille de voxels n'est en fait rien d'autre qu'une image 3D, où le voxel est l'analogue du pixel ou du texel.

Le problème maintenant réside en la modélisation des textures. Tout comme la modélisation géométrique d'objets (§2.1.2), nous voulons produire une version numérique de l'apparence en surface d'objets réels ou fictifs. Les mêmes classes de méthodes de création de modèles géométriques se retrouvent pour la création de textures, telles :

- les méthodes directes par un artiste avec des logiciels d'édition d'images tels *Photoshop* [Adobe Systems Inc.] ou tout autre logiciel de peinture numérique ;
- les méthodes procédurales [Peachey 1985, Perlin 1985, Perlin 2002, Miyata 1990, Turk 1991, Ebert *et al.* 1998] ;
- les méthodes par numérisation, *i.e.* photographie numérique ou numérisation d'images.

Les avantages et inconvénients de ces méthodes sont essentiellement les mêmes que pour les méthodes de création de modèles géométriques.

Il reste le problème du placage de textures en soi, pour lequel il faut trouver une paramétrisation de la surface. Ce problème ne se pose pas pour les textures 3D car nous avons directement une paramétrisation naturelle, soit la position 3D d'un point. Par contre pour les textures 2D, cette paramétrisation est loin d'être toujours évidente, en particulier pour les surfaces de topologie de *genus* élevé, et il s'agit en soi un domaine de recherche.

Même si nous pouvons trouver aisément une paramétrisation, celle-ci va généralement engendrer des distorsions de la texture. Un exemple est d'appliquer une texture d'échiquier sur une surface bicubique. La paramétrisation de cette dernière est naturelle, mais l'échiquier sera distordu. Dans le cas de ce type de surface, nous pouvons corriger ce problème en essayant de reparamétrer la surface par contrainte d'aire ou de distance en espace 3D. Mais il n'existe pas toujours des solutions faciles pour éviter ce problème sur des surfaces de complexité et topologie arbitraires.

Un autre problème survient lorsque l'aire de la surface d'application est plus grande que le domaine de la texture. Afin de préserver un certain niveau de détails, il faut la paver. Or, en faisant ceci, il y a possibilité d'engendrer des faux contours aux frontières du domaine, à moins de garantir que la texture soit cyclique. Même si c'est le cas, nous avons la même texture qui se répète partout sur la surface, et l'œil humain perçoit souvent bien les alignements et les répétitions [Palmer 1999], ce qui décroît la qualité visuelle du résultat.

Une alternative au placage de textures est de peindre la texture directement sur la surface d'un objet [Hanrahan et Haeberli 1990, Carr et Hart 2004], mais le procédé a les mêmes in-

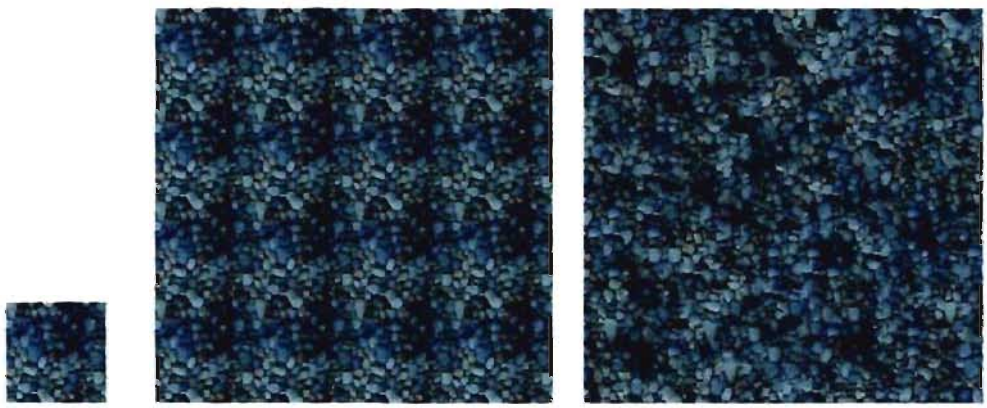


FIG. 3.1 – Pavage d'un échantillon *vs.* génération de textures. Les images sont tirées de [De Bonet 1997].

convénients que les méthodes directes de création de textures. Turk [1991] a développé une méthode procédurale de génération sur la surface basée sur un modèle de réaction-diffusion. Une autre possibilité est d'utiliser les textures 3D avec leur paramétrisation naturelle, *i.e.* la position 3D d'un point donne directement la valeur du paramètre de la texture à cette position. Pour toute texture ayant une représentation solide naturelle (par exemple le marbre ou le bois), cette méthode fonctionne bien, mais pour les textures définies plus naturellement au-dessus d'un plan (par exemple de l'herbe ou un motif d'un tissu), il faut des structures plus élaborées, telles l'encodage en *octree* [DeBry *et al.* 2002, Benson et Davis 2002], adaptées à la géométrie de l'objet sur lequel nous voulons appliquer la texture. En conséquence, avec une telle représentation, nous ne pouvons pas réutiliser la même texture sur deux objets aux géométries différentes.

Une autre solution, apportée pour la première fois au sein de la communauté graphique par Heeger et Bergen [1995], consiste à générer une texture qui ressemble visuellement à un échantillon donné mais sans en être une copie ou un pavage. L'échantillon, ou texture source, est une image de taille réduite mais suffisamment grande pour capturer l'essentiel de la texture, et dont la provenance est le produit de l'une des méthodes de création mentionnées auparavant. La taille de la texture générée peut être arbitraire. Cette classe de méthodes est maintenant simplement nommée génération de textures[4]. La figure 3.1 illustre un exemple de génération de textures.

[4]Le terme englobe en fait toute méthode qui produit une texture synthétique, mais une convention s'est installée dans la communauté graphique depuis quelques années pour réserver ce terme seulement pour ce type de méthode.

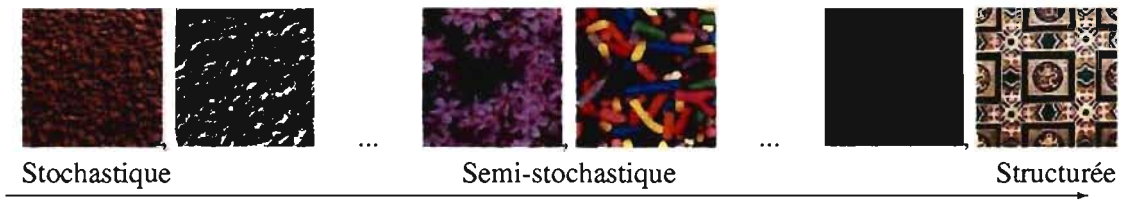


FIG. 3.2 – Types principaux de textures.

La génération de textures se veut être une méthode générique, indépendante du type de texture, contrairement aux méthodes procédurales. Cependant, il est important de connaître les catégories générales de textures pour être en mesure d'évaluer la qualité des résultats et les failles possibles d'une méthode particulière de génération. Les textures sont souvent classées selon deux types principaux [Heeger et Bergen 1995] :

- les textures *déterministes* : elles sont caractérisées par un ensemble de primitives et de règles de placement, par exemple un plancher de tuiles, un mur de briques régulières ou les fenêtres d'un immeuble ;
- les textures *stochastiques* ou *structurées* : textures sans structures évidentes, par exemple du granite ou du marbre.

Une grande partie des textures du monde réel tombent quelque part entre ces deux catégories, par exemple le grain du bois ou un champ labouré. La figure 3.2 montre quelques exemples de textures et leur type.

Plusieurs types de méthodes de génération de textures ont été développées, chacun cherchant à couvrir un plus large éventail de types de textures qu'un autre. Nous les regroupons dans les catégories suivantes :

Méthodes paramétriques (§3.1.1) Ces méthodes se basent sur un modèle statistique donné et génèrent de nouvelles textures selon les distributions des texels observées en réponse à divers filtres simulant la perception humaine.

Génération par texel (§3.1.2) Ces méthodes procèdent par échantillonnage direct de la texture et, essentiellement, procèdent en réorganisant les texels. Nous les classifions dans les sous-types suivants :

Méthodes multirésolutions (§3.1.2.1) Ces méthodes construisent une pyramide de la texture source, généralement de type laplacien, et remplissent la pyramide de la texture à générer niveau par niveau, du plus grossier au plus fin. Le texel du niveau courant de la pyramide de la texture source dont les parents (jusqu'au niveau le plus grossier) sont les plus similaires à ceux du texel à générer est choisi pour remplacer ce dernier.

Méthodes par recherche de voisinage (§3.1.2.2) Le principe de base de ces méthodes est de simplement chercher un texel dans la texture source dont le voisinage est similaire au voisinage courant du texel à générer.

Optimisation globale (§3.1.2.3) L'échantillonnage de la texture se fait, avec ces méthodes, par l'entremise d'une optimisation itérative sur l'ensemble de la texture générée.

Génération par *patch* (§3.1.3) Ces méthodes fonctionnent essentiellement comme celles de la catégorie par texel, mais au lieu de générer texel par texel, elles procèdent *patch* par *patch*, *i.e.* par groupe de texels.

Méthodes par analyse structurale (§3.1.4) Il s'agit en fait d'un enrichissement aux méthodes précédentes. Les textures sources sont analysées pour extraire plus d'information que seulement la couleur, et la génération est guidée ou modifiée en fonction de ces informations.

Génération adaptée aux surfaces (§3.1.6) Toutes les méthodes de génération de textures s'intéressent essentiellement à la génération 2D, ce qui ne règle pas les problèmes de distorsions engendrées par le placage de textures. Ces méthodes adaptent les méthodes précédentes pour effectuer la génération en tenant compte de la géométrie de la surface de l'objet sur laquelle elle sera appliquée.

Dans les prochaines sections, nous abordons plus en détail ce qui a été développé pour chaque type de méthode, et à la §3.1.7, nous discutons de l'application de la génération de textures dans d'autres contextes. Nous insérons aussi une discussion sur une comparaison qualitative de certaines méthodes à la §3.1.5.

3.1.1 Méthodes paramétriques

Les méthodes paramétriques partent d'un modèle de définition de textures et jouent sur les paramètres statistiques du modèle pour générer de nouvelles textures. Une description plus

enrichie de ces méthodes de génération a été apportée par Portilla et Simoncelli [2000]. Ils définissent les ingrédients de ces méthodes ainsi :

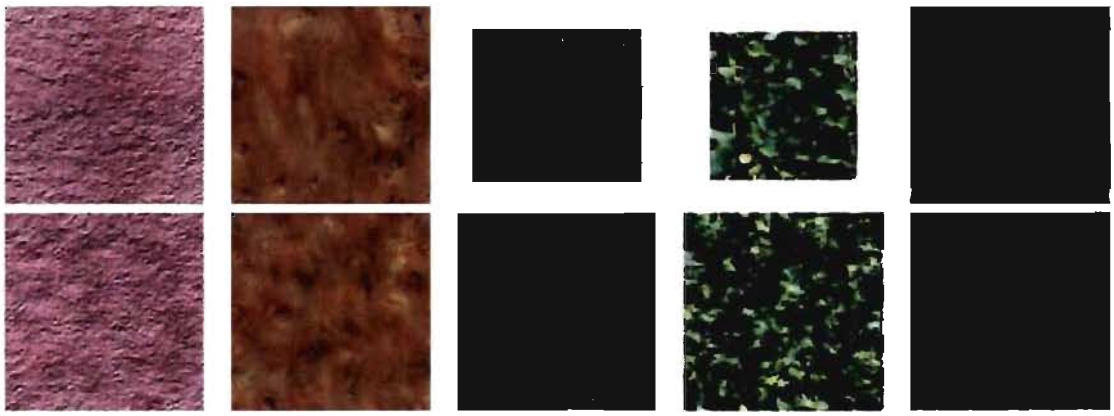
- un ensemble de contraintes statistiques ;
- une bibliothèque d'exemples de textures ;
- une méthode d'estimation de paramètres statistiques ;
- un algorithme pour générer des échantillons d'un champ aléatoire ;
- une méthode pour mesurer la différence perceptuelle entre deux textures.

Les contraintes statistiques peuvent inclure, par exemple, l'histogramme des couleurs [Heeger et Bergen 1995, Zhu *et al.* 1998], les fonctions de potentiel des champs aléatoires de Markov (*Markov random fields* ou MRF) [Coughlan et Yuille 2002] ou les fonctions de transfert et les paramètres de transformations géométriques d'un modèle de moyenne mobile (*moving average*) [Eom 2000]. Portilla et Simoncelli [2000] eux-mêmes font usage d'une banque de statistiques plus grande qui, en plus d'inclure certaines des statistiques énumérées ci-avant, inclut des mesures sur les coefficients de la représentation en ondelettes [Gonzalez et Woods 2002] des textures.

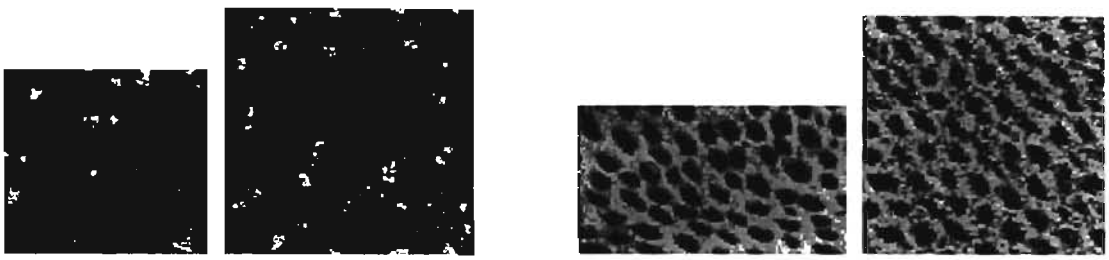
Les méthodes paramétriques de génération procèdent typiquement en transformant progressivement une image de bruit blanc de la taille finale désirée jusqu'à ce qu'elle soit suffisamment similaire à un exemple donné. La transformation peut être effectuée, par exemple, par l'ajustement hiérarchique d'histogrammes [Heeger et Bergen 1995], par échantillonnage de Gibbs [Geman et Geman 1984, Zhu *et al.* 1998], par des variantes de mises à l'échelle itératives généralisées [Coughlan et Yuille 2002] ou par des manipulations dans le domaine fréquentiel de processus stochastiques [Eom 2000]. Une bonne partie des méthodes de génération se basent sur la comparaison des textures en réponse à un ensemble de filtres basés sur un modèle de perception humaine [Heeger et Bergen 1995, Zhu *et al.* 1998, Portilla et Simoncelli 2000] pour déterminer la qualité de la transformation, alors que d'autres se basent purement sur l'estimation des paramètres [Eom 2000, Coughlan et Yuille 2002]. Des exemples de résultats pour des méthodes paramétriques se trouvent à la figure 3.3.

3.1.2 Génération par texel

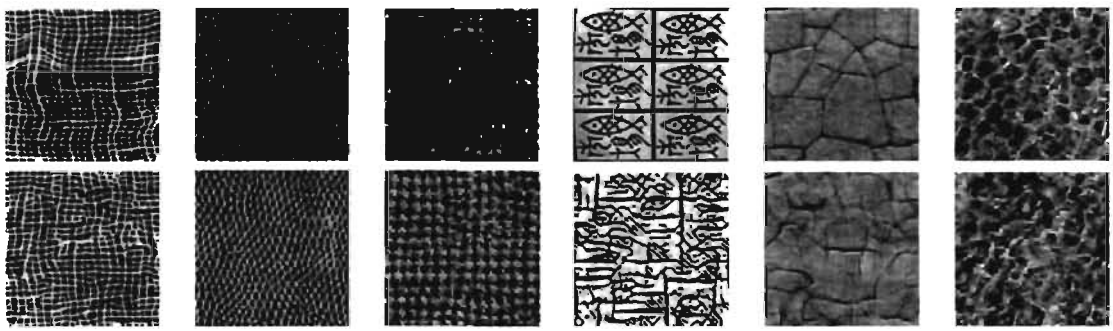
Cette classe de méthodes et ses dérivées (notamment la génération par *patch*) ont gagné beaucoup de popularité pour leur facilité d'implémentation et la qualité de leurs résultats. Il s'agit de méthodes dites non paramétriques car elles ne se basent pas sur un modèle de texture



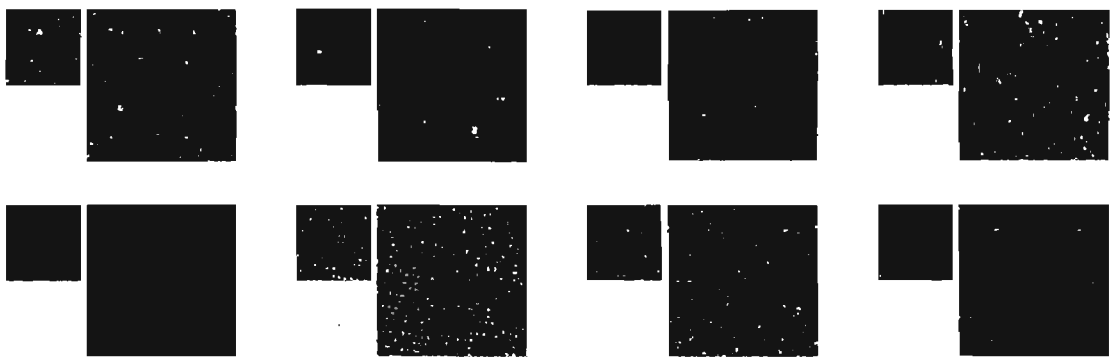
Heeger et Bergen [1995] ; haut : sources ; bas : générées.



Zhu *et al.* [1998] ; petites : sources ; grandes : générées.



Portilla et Simoncelli [2000] ; haut : sources ; bas : générées.



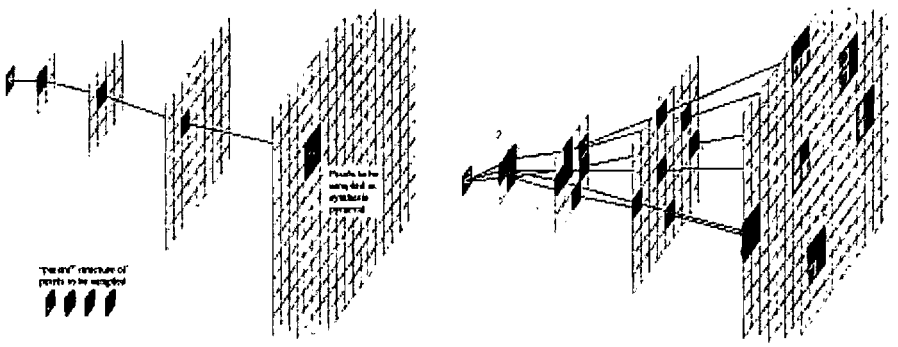
Eom [2000] ; petites : sources ; grandes : générées.

FIG. 3.3 – Résultats de génération pour des méthodes paramétriques. Les images proviennent des articles respectifs.

mais procèdent plutôt par un échantillonnage d'une texture source. Les méthodes de génération par texel composent la texture produite texel par texel. Nous classifions les méthodes de ce type en trois catégories principales : multirésolutions, recherche de voisinage et optimisation globale.

3.1.2.1 Méthodes multirésolutions

Ces méthodes passent par une décomposition multirésolution[5] des textures, typiquement de type laplacien [Burt et Adelson 1983], et effectuent la génération niveau par niveau dans la hiérarchie (en commençant par la racine) et texel par texel pour chaque niveau. Un nouveau texel est choisi en cherchant un texel qui a une suite similaire d'ancêtres dans la hiérarchie source, ce qui est illustré à la figure 3.4. Afin d'obtenir des comparaisons d'ancêtres plus significatives, leur similarité peut être évaluée suivant la réponse à un ensemble de filtres basés sur des modèles de perception humain [De Bonet 1997]. La texture finale est produite par l'opération inverse de la décomposition multirésolution.



(a) (b)

FIG. 3.4 – Génération multirésolution : (a) voisinage pyramidal et (b) recherche de l'ensemble de texels candidats. Les images proviennent de [De Bonet 1997].

Plusieurs textures sources peuvent être utilisées au lieu d'une seule pour produire des mélanges de textures, tel qu'illustré à la figure 3.5. Il s'agit essentiellement de décomposer chaque source et d'effectuer la recherche dans l'ensemble des sources [Bar-Joseph *et al.* 2001]. Pour éviter l'explosion exponentielle de l'espace de recherche, il peut s'avérer nécessaire de recycler les candidats de sélection du parent immédiat pour un texel donné, ce qui peut aussi accroître la cohérence de la génération.

[5] Au chapitre 5, nous abordons les représentations multirésolutions dans le contexte de la modélisation géométrique. Le principe est essentiellement le même pour les images.

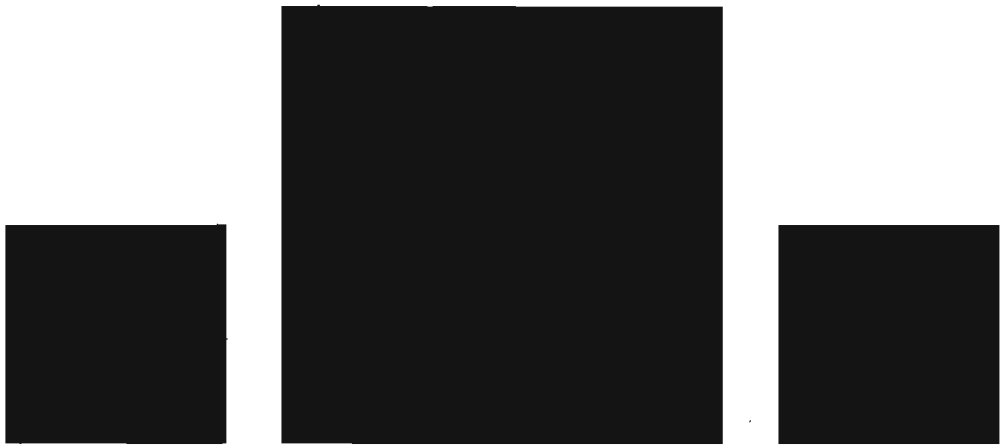


FIG. 3.5 – Mélange de textures de Bar-Joseph *et al.* [2001] : à gauche et à droite se trouvent les deux textures sources et au milieu, la texture générée par un mélange des deux. Les images proviennent de [Bar-Joseph *et al.* 2001].

3.1.2.2 Méthodes par recherche de voisinage

Le fonctionnement général de ces méthodes va comme suit. Pour chaque nouveau texel, un voisinage dans la texture générée jusqu'à cette étape est accumulé. Une recherche est ensuite lancée dans la texture source[6] pour trouver un ensemble de texels candidats dont le voisinage correspondant est similaire au voisinage accumulé. La valeur d'un des texels candidats, typiquement choisi au hasard, est ensuite copiée dans la texture générée et la procédure passe au texel suivant, selon un ordre donné. Le type de voisinage, la méthode de recherche et l'ordre de génération varie selon la méthode.

Le voisinage le plus simple est un carré de texels autour du texel courant [Efros et Leung 1999], *i.e.* le plein voisinage. Ce type de voisinage s'adapte bien à différents ordres de génération, mais il faut gérer les cas spéciaux des voisins non encore générés. L'ordre de génération le plus commun est l'ordre par balayage de ligne (*scanline*) [Wei et Levoy 2000, Ashikhmin 2001, Hertzmann *et al.* 2001, Zelinka et Garland 2002, Sabha et Dutré 2007b, Dong *et al.* 2007], dont le processus est illustré à la figure 3.6. Cet ordre est populaire car le nombre de texels voisins considérés est moindre, et, en dehors des premières rangées, il n'y a pas de cas particuliers parmi les voisins. Ces voisinages en forme de "L" sont appelés des voisinages causals. Pour les premières rangées, l'image peut être initialisée avec du bruit dont l'histogramme correspond à celui de la texture source [Wei et Levoy 2000]. De plus, pour être en mesure de gé-

[6] Il est à noter que tout comme le font Bar-Joseph *et al.* [2001] pour les méthodes multirésolutions, il est possible d'effectuer la rechercher dans plus d'une texture source pour obtenir des mélanges [Wei 2002].

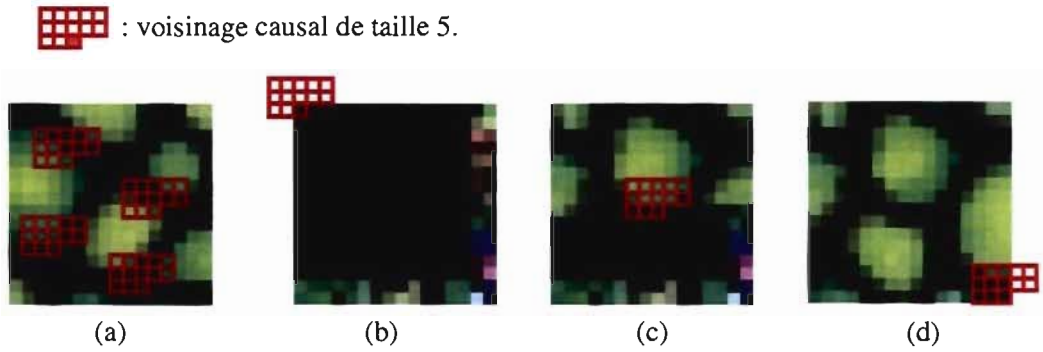


FIG. 3.6 – Génération en ordre de balayage de ligne : (a) la texture source est utilisée pour produire une liste de candidats ; (b) initialisation de la génération (les voisinages sont calculés sur la texture toroïdale) ; (c) génération d'un texel dans un cas typique ; (d) texture générée finale. Les images proviennent de [Wei et Levoy 2000].

nérer des textures toroïdales, les positions des voisins sont habituellement calculées en modulo avec la taille de la texture générée.

Pour que ces méthodes arrivent à préserver les structures d'une texture source, la taille des voisinages doit être suffisamment grande, ce qui peut rendre le temps d'exécution très grand. Une solution populaire à ce problème, initialement apportée par Wei et Levoy [2000] pour ces méthodes, est d'effectuer la génération à différents niveaux de résolution[7], du plus grossier au plus fin, et en enrichissant le voisinage du niveau courant avec un voisinage complet réduit au niveau parent. Ceci permet de mieux préserver les structures et s'effectue à moindre coût.

La distinction principale des différentes méthodes de génération par texel se situe au niveau de la façon de chercher des candidates dans la texture source pour générer le texel courant. Il y a la recherche directe et exhaustive [Efros et Leung 1999], qui est simple mais très lente. Cette dernière peut être grandement accélérée en considérant le voisinage accumulé comme un vecteur, où toutes les composantes de couleurs sont aboutées les unes après les autres, et en construisant un espace de recherche de tous les voisinages présents dans la texture source. L'accélération vient du fait que cet espace peut être structuré, par exemple par quantification de vecteurs structurée par arborescence (*tree-structured vector quantization* ou *TSVQ*) [Wei et

[7]Bien que la littérature à propos de cette solution parle de génération multirésolution, la pyramide d'images utilisée est habituellement une pyramide gaussienne, qui correspond essentiellement à une séquence d'images de plus en plus floues et de résolutions de plus en plus faibles. L'image originale fait partie de cette pyramide, soit le niveau de plus grande résolution. Selon ce que nous considérons comme une représentation multirésolution au chapitre 5, la pyramide gaussienne n'en est pas une. Par contre, Wei et Levoy [2000] mentionnent que cette approche peut très bien s'appliquer sur d'autres types de pyramide, et alors la démarcation entre les méthodes multirésolutions (§3.1.2.1) et celles-ci n'est plus aussi nette. Nous préférons plutôt utiliser le terme *génération multiniveau*.

Levoy 2000] ou en arbre-*kd* [Hertzmann *et al.* 2001][8]. L'espace de recherche peut être aussi inversé [Sabha et Dutré 2005, Sabha et Dutré 2007b], *i.e.* pour chaque valeur de couleur différente, une liste de toutes ses positions dans la texture source est conservée. Les couleurs sont stockées dans un arbre-*kd* pour retrouver leur liste rapidement. La recherche de candidats est ensuite faite par des requêtes dans cet arbre-*kd* pour chaque couleur dans le voisinage, et les listes de positions trouvées sont ensuite parcourues de façon exhaustive, en comparant les voisinages autour du texel de la texture source ayant la même position relative que dans le voisinage dans la texture générée. Cette idée de relation de position peut être poussée à l'extrême en utilisant les cartes de sauts (*jump maps*) [Zelinka et Garland 2002, Zelinka et Garland 2004a], qui consistent essentiellement à précalculer la similarité entre les voisinages d'une texture source, conserver les liens, et suivre ces liens pendant la génération pour choisir la couleur du texel suivant, ce qui donne une méthode de génération très rapide, mais qui peut briser les structures présentes dans la texture.

Une autre approche de sélection de candidats est celle d'Ashikhmin [2001], dite recherche par cohérence, qui est souvent utilisée en conjonction avec les méthodes précédentes [Hertzmann *et al.* 2001, Lefebvre et Hoppe 2005]. Cette méthode garde en mémoire l'origine des texels déjà générés, et la recherche s'effectue parmi les texels voisins des texels sources asso-

[8]Hertzmann *et al.* [2001] font en réalité usage d'une recherche hybride avec la méthode *cohérente* d'Ashikhmin [2001].

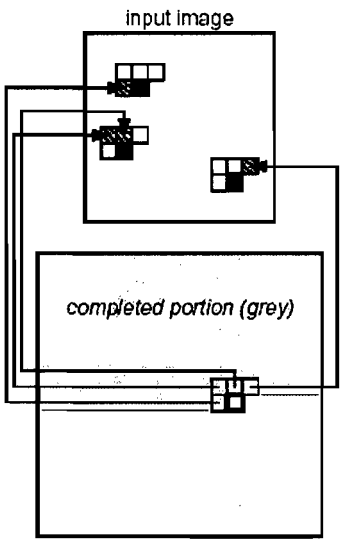


FIG. 3.7 – Recherche par cohérence de la méthode d'Ashikhmin [2001]. Image tirée de [Ashikhmin 2001].

ciés aux texels du voisinage causal courant, tel qu'illustré à la figure 3.7. Dong *et al.* [2007] ont étendu cette méthode en considérant des voisinages de tailles variables pour tenir compte de textures pour lesquelles les *textons*[9] ne sont pas de taille uniforme.

Pour la plupart des méthodes, le résultat de la génération dépend de l'ordre de génération, mais il est possible d'obtenir une génération indépendante de l'ordre [Wei et Levoy 2002, Lefebvre et Hoppe 2005], dite aussi génération parallèle. En considérant une approche multiniveau avec passes de correction, les texels du niveau courant sont générés seulement à partir du voisinage complet dans le niveau parent, ou aussi avec le voisinage complet du niveau courant mais à une itération précédente. Autrement dit, aucun texel à générer ne dépend de texels venant d'être générés au niveau courant et à la passe courante. Profitant également de ces passes de correction, Lefebvre et Hoppe [2005] ajoutent une légère perturbation sur les positions après la première passe pour augmenter la variation dans les résultats.

Il est important de noter que les méthodes de génération par recherche de voisinage reposent sur les champs aléatoires de Markov comme modèle de textures, *i.e.* la probabilité de la réalisation d'un texel donné dépend seulement d'un voisinage autour de lui. En effet, parce que les recherches sont faites à partir de voisinage fini, il n'y a aucun moyen de reproduire fidèlement des caractéristiques qui s'étendent sur tout l'échantillon. Pour parvenir à générer des distributions de caractéristiques suivant certains patrons, les méthodes actuelles doivent recourir à l'intervention d'un usager [Ashikhmin 2001, Hertzmann *et al.* 2001, Lefebvre et Hoppe 2005].

3.1.2.3 Optimisation globale

À l'exception des méthodes de génération parallèle [Wei et Levoy 2002, Lefebvre et Hoppe 2005], les méthodes de la section précédente procèdent par agrandissement d'une région générée, ce qui peut faire en sorte qu'une petite erreur puisse s'accumuler. En contrepartie, les méthodes par optimisation globale posent le problème de génération par la minimisation de l'énergie d'une texture [Kwatra *et al.* 2005, Huang *et al.* 2007, Dong *et al.* 2008a]. En fait, plusieurs des méthodes paramétriques (§3.1.1) sont également des méthodes de génération globale [Heeger et Bergen 1995, Portilla et Simoncelli 2000, Eom 2000, Coughlan et Yuille 2002], mais qui passent par l'estimation de paramètres d'un modèle de texture. Bien que, comme la plupart des méthodes non paramétriques, le modèle sous-jacent de texture des méthodes par optimisation globale soit les champs aléatoires de Markov, la fonction d'énergie n'est pas définie

[9] Les textons réfèrent aux micro-structures fondamentales dans les images naturelles [Zhu *et al.* 2005].

en terme de paramètres, mais plutôt en terme de similarité de voisinages autour de texels dans la texture générée et de voisinages correspondants dans la texture source. Ainsi, les variables inconnues à estimer sont directement les couleurs des texels de la texture générée. L'accumulation d'erreurs pour des texels près l'un de l'autre dans la texture générée, mais dont les voisinages correspondants dans la texture source ne sont pas cohérents, renforce la cohérence de la génération avec la texture source. La fonction d'énergie est typiquement minimisée par moindres carrés repondérés itérativement (*iteratively reweighted least squares*) [Coleman *et al.* 1980].

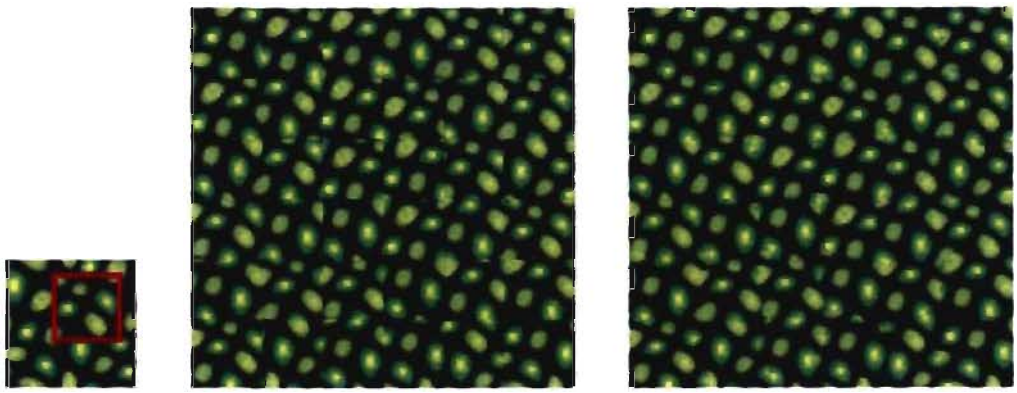
A priori, puisque la fonction d'énergie se base sur des voisinages, comme les méthodes par recherche de voisinages, la taille du voisinage doit être choisie conséquemment avec la taille des textons [Zhu *et al.* 2005] de la texture source. Il est possible aussi d'effectuer l'optimisation à plusieurs niveaux, et alors, par la globalité de la génération, ces méthodes obtiennent des résultats similaires aux meilleures méthodes de génération par *patch*, décrites dans la prochaine section.

3.1.3 Génération par *patch*

La plupart des méthodes de génération par texel éprouvent quelques difficultés avec les éléments structuraux : les résultats présentent soit d'importantes déformations ou du bruit irrégulier, ils ont même la possibilité de diverger[10]. Pour être en mesure de réduire ces effets, le voisinage doit avoir une très grande taille. Les méthodes de génération par *patch* cherchent à réduire ces problèmes. Elles procèdent pour la plupart de la même façon que les méthodes par recherche de voisinage (§3.1.2.2) [Liang *et al.* 2001, Efros et Freeman 2001, Long et Mould 2007, Kwatra *et al.* 2003, Nealen et Alexa 2003, Nealen et Alexa 2004, Sabha et Dutré 2006] ou même que les méthodes multirésolution (§3.1.2.1) [Singh et Dana 2004], mais au lieu de générer la texture un texel à la fois, ces méthodes échantillonnent la texture source par groupes de texels, *i.e.* par *patches*. Ces dernières sont typiquement de forme rectangulaire, mais des formes arbitraires sont également possibles [Xu *et al.* 2000, Xu *et al.* 2001]. Ceci comporte deux avantages principaux :

- une génération souvent plus rapide par le fait que le nombre de recherches est réduit (moins de *patches* par texture) et qu'il y a généralement moins de texels dans le voisinage d'une *patch* que dans la *patch* elle-même ;
- une meilleure préservation des éléments structuraux de la texture.

[10] Le problème de divergence ne se pose pas pour les méthodes de génération parallèle et par optimisation globale.



(a) (b) (c)

FIG. 3.8 – Problématique de la génération par *patch* : en échantillonnant simplement la texture source par *patch*, indiqué par le carré rouge en (a) en minimisant la différence des voisinages, un résultat tel qu'en (b) sera obtenu, où des discontinuités sont encore visibles entre *patches* adjacentes. Un travail supplémentaire doit être fait pour joindre les *patches* et obtenir un résultat tel qu'en (c). Les images sont tirées de [Efros et Freeman 2001].

Le problème *a priori* de ces méthodes est de gérer les discontinuités entre les *patches*, illustré à la figure 3.8, et la recherche efficace d'une *patch*. Principalement, c'est à ce niveau que chaque méthode se distingue.

Pour être en mesure de convenablement joindre une nouvelle *patch* avec la texture générée à l'étape courante, il doit y avoir une région de superposition. Cette dernière sert également d'information de voisinage pour rechercher une *patch* dans la texture source, de façon similaire à la recherche de voisinage pour les méthodes de génération par texel. La région de superposition peut ensuite servir pour faire un mélange en dégradé (*feathering*) [Xu *et al.* 2000, Xu *et al.* 2001, Liang *et al.* 2001, Singh et Dana 2004]. Autrement, chaque texel de la région de superposition doit être choisi parmi le texel courant dans la texture générée ou le texel correspondant dans la *patch* de sorte à limiter les discontinuités. Cette sélection peut être faite par un algorithme de programmation dynamique [Efros et Freeman 2001, Sabha et Dutré 2006, Long et Mould 2007], ou par un algorithme de coupe minimal de graphe [Kwatra *et al.* 2003], également connu sous le nom d'algorithme de calcul de flot maximum [Ford et Fulkerson 1962]. Pour cette dernière méthode, le graphe correspondant au problème est composé d'un sommet par texels qui se superposent, d'un sommet représentant les texels de l'ancienne *patch* voisins à la région de superposition, et un autre sommet similairement pour la nouvelle *patch*. Il existe une arête entre deux sommets si les texels correspondants sont voisins. La figure 3.9 illustre la

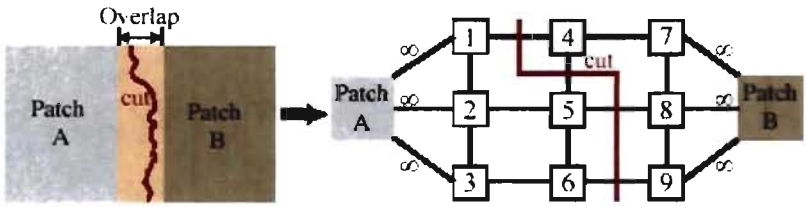


FIG. 3.9 – Coupe de graphe de la méthode de Kwatra *et al.* [2003]. Image tirée de [Kwatra *et al.* 2003].

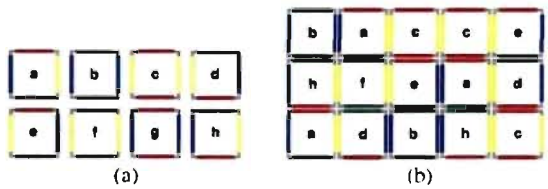
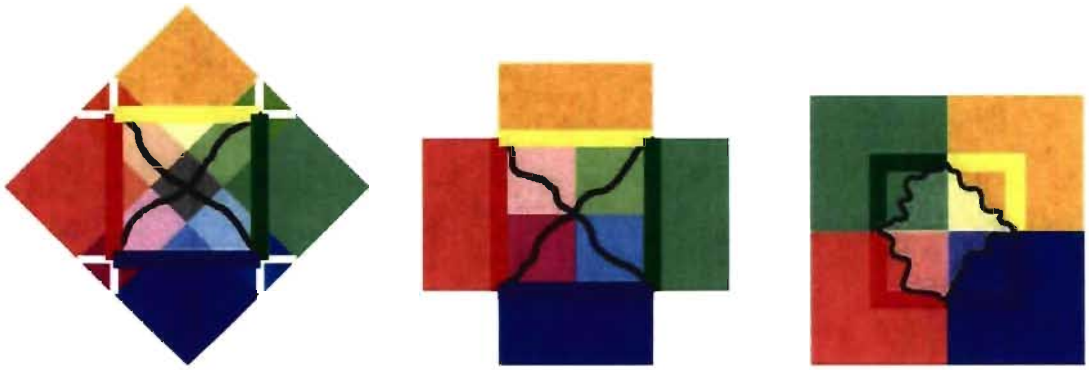


FIG. 3.10 – Pavage de tuiles de la méthode de Cohen *et al.* [2003]. (a) Jeu de tuiles et (b) un pavage simple généré. Image tirée de [Cohen *et al.* 2003].

construction du graphe. Le poids de chaque arête est calculé en fonction de la différence entre les couleurs des texels et des gradients. Il est également possible de combiner certaines méthodes, notamment le mélange en dégradé avec une méthode de découpage [Kwatra *et al.* 2003]. Nealen et Alexa [2003, 2004] ont plutôt proposé de faire usage de la génération par texel pour corriger les erreurs dans la région de superposition.

Une méthode particulière de génération par *patch* est celle par tuile de Wang [1961]. Initialement proposée par Cohen *et al.* [2003], cette méthode construit un jeu de tuiles de Wang à partir de la texture source, puis combine les tuiles aléatoirement ensemble avec la restriction que deux tuiles adjacentes doivent avoir le même code de couleur sur le côté partagé. La figure 3.10 montre un exemple de jeu de tuiles et un pavage simple. Il est possible aussi d'établir les contraintes de couleurs sur les coins plutôt que sur les côtés, et alors les tuiles se font appeler tuiles-ω [Ng *et al.* 2005] ou simplement tuiles de coin [Lagae et Dutré 2006]. La difficulté principale de cette méthode est de générer le jeu de tuiles. Pour assurer une jonction continue sur leur frontière, elles sont formées en superposant partiellement quatre *patches* de la texture source puis en utilisant un algorithme de programmation dynamique pour découper les régions de superposition [Cohen *et al.* 2003, Burke 2003], ou alors en procédant par coupe de graphe [Ng *et al.* 2005, Lagae et Dutré 2006]. La figure 3.11 illustre diverses stratégies de composition. En (c), une cinquième *patch* est utilisée. Le choix des *patches* à combiner peut



(a) (b) (c)

FIG. 3.11 – Stratégies de combinaison pour générer des tuiles : (a) Cohen *et al.* [2003], (b) Burke [2003], (c) Ng *et al.* [2005] et Lagae et Dutré [2006]. Les images proviennent de [Ng *et al.* 2005].

être fait interactivement, ou par des procédures automatiques évoluées [Dong *et al.* 2005, Dong *et al.* 2008b]. L'avantage principal de cette méthode de génération est la rapidité de la génération, et sa facilité d'adaptation au matériel graphique [Wei 2004, Lagae et Dutré 2006].

3.1.4 Méthode par analyse structurale

Une texture contient plus d'information qu'une simple couleur par texel. À plus haut niveau, elle contient des textons [Zhu *et al.* 2005], des formes, etc. Les méthodes par analyse structurale cherchent à profiter de ces informations de plus haut niveau pour améliorer la qualité de la génération, au prix parfois de se spécialiser pour certaines classes de texture.

Sans doute l'information la plus fondamentale, mais de plus haut niveau que la couleur des texels, est celle des textons. Pour en profiter, Zhang *et al.* [2003] font l'extraction d'un masque de textons à partir de la texture source (un exemple se trouve à la figure 3.12) qu'ils utilisent avec une méthode de génération par texel avec contraintes [Hertzmann *et al.* 2001]. Une alternative au masque est l'usage d'un diagramme de Voronoï associé à la distribution d'un texton identifié [Sabha et Dutré 2007a]. Un nouveau diagramme de Voronoï est alors généré puis la texture est produite en cherchant les cellules du diagramme source similaires à celles du diagramme généré. Le procédé est illustré à la figure 3.13.

Les textures contiennent souvent des caractéristiques de fréquences plus élevées telles que des contours. Les méthodes de génération par *patch* n'arrivent pas à toujours bien aligner ces contours. Ainsi, Wu et Yu [2004] ont présenté une méthode de génération par *patch* qui dé-

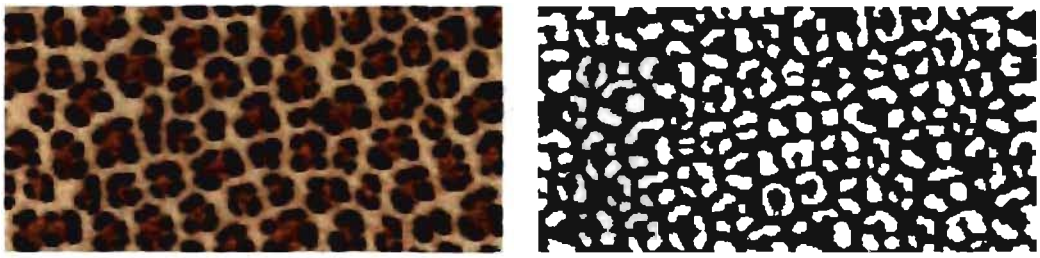


FIG. 3.12 – Masque de textons de la méthode de Zhang *et al.* [2003] : à gauche se trouve la texture source et à droite le masque de textons correspondant. Les images proviennent de [Zhang *et al.* 2003].

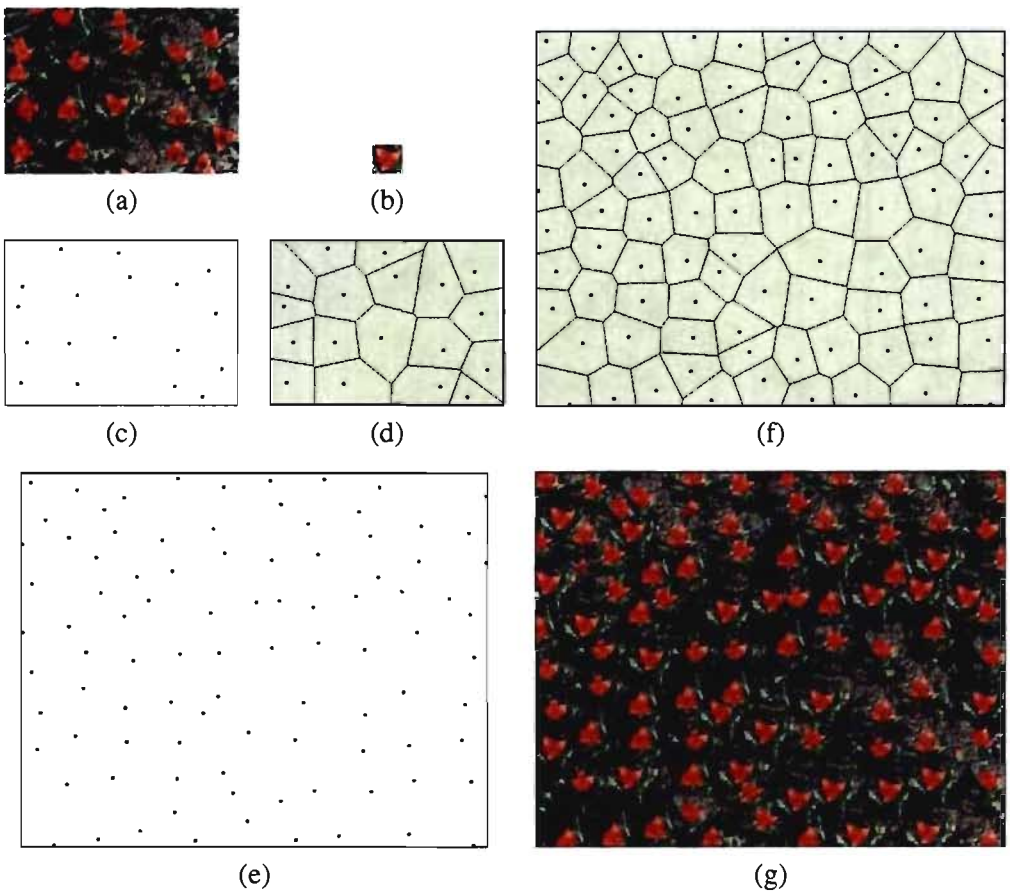


FIG. 3.13 – Génération par diagramme de Voronoï : (a) texture source ; (b) identification d'un texton ; (c) distribution du texton dans la texture source ; (d) diagramme de Voronoï de (c) ; (e) génération d'une distribution similaire à (c) ; (f) diagramme de Voronoï correspondant à (e) ; (g) la texture est générée en cherchant les cellules du diagramme en (d) similaires à celles du diagramme en (f). Les images proviennent de [Sabha et Dutré 2007a].

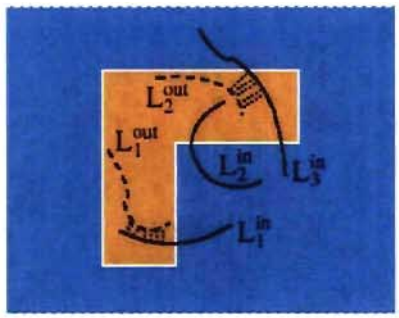


FIG. 3.14 – Détection et déformation de caractéristiques de la méthode de Wu et Yu [2004] : à l'intérieur de la région de superposition entre la portion déjà générée de la texture et la nouvelle *patch*, les contours L_1^{out} et L_2^{out} sont détectés dans la texture générée de même que les contours L_1^{in}, L_2^{in} et L_3^{in} dans la nouvelle *patch*. L'interpolation entre les contours sera faite entre les paires de contours qui correspondent le mieux selon la proximité, la forme et l'orientation. L'image provient de [Wu et Yu 2004].

tecte et, suivant une mise en correspondance, effectue une déformation locale dans la région de superposition. La figure 3.14 illustre la méthode.

Les caractéristiques d'une texture ont souvent une distribution particulière. Certaines méthodes font alors usage de champs de déformation pour étudier cette distribution et générer des textures. Le champ de déformation peut correspondre à l'écart de la distribution par rapport à une texture régulière [Liu *et al.* 2004b, Liu *et al.* 2005], ou à l'écart entre les caractéristiques de paires de textures [Matusik *et al.* 2005]. Le champ de déformation peut servir directement pour la génération de textures, comme c'est le cas de la méthode de Liu *et al.* [2004b, 2005], illustrée à la figure 3.15, ou pour interpoler et composer diverses textures ensemble [Matusik *et al.* 2005]. Pour générer des textures quasi régulières, l'usage d'un système évolutif a aussi été proposé par Wu *et al.* [2007].

Une façon plus générique d'enrichir l'information et de l'exploiter a été proposée par Lefebvre et Hoppe [2006]. Auparavant, chaque texel pouvait être considéré comme un vecteur 3D (rouge, vert, bleu). La plupart des méthodes de génération font abstraction de l'espace auquel appartiennent les texels. *A priori*, rien n'empêche de considérer des espaces de plus haute dimension. Par exemple, chaque texel peut être accompagné d'information de voisinage, d'une distance au contour le plus près, etc. Pour profiter d'un maximum d'information sans trop alourdir les calculs, une technique de réduction de dimension adaptée à la texture source est utilisée. En combinant cette réduction avec un méthode de génération parallèle [Lefebvre et

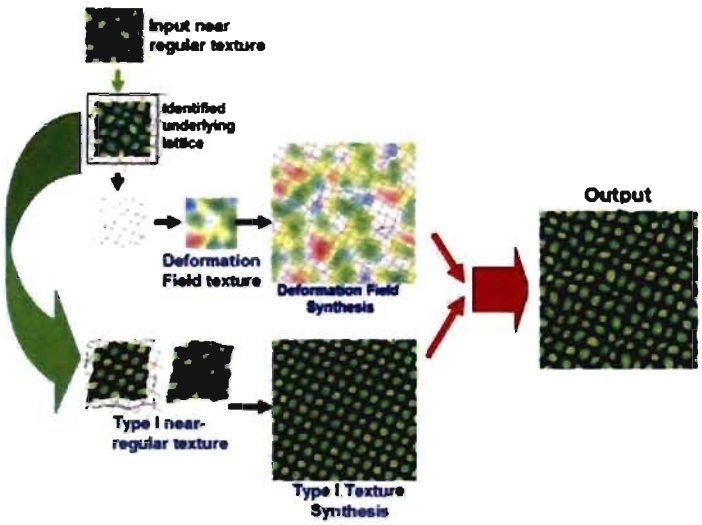


FIG. 3.15 – Méthode de Liu *et al.* [2004b] : un treillis de déformation est identifié et la texture source est transformée en redressant le treillis. Le résultat est ce que les auteurs appellent une texture quasi régulière de type I. Une méthode de génération par texel est utilisée pour générer un champ de déformation plus grand et une méthode spécialisée pour les textures quasi régulières de type I est employée pour agrandir l'échantillon transformé. La déformation est ensuite appliquée sur ce dernier résultat pour donner la texture finale. La figure provient de [Liu *et al.* 2004b], mais elle a été simplifiée par souci de clarté.

Hoppe 2005] adaptée au matériel graphique, la génération peut être calculée en temps interactif, et ce, avec de très bons résultats.

3.1.5 Comparaison qualitative des méthodes de génération

Parmi la panoplie de méthodes de génération de textures, certaines sont plus appropriées que d'autres pour un type de texture donné. Ainsi, comme nous pouvons le voir aux figures 3.3 et 3.16 (Heeger et Bergen [1995]), les méthodes paramétriques (§3.1.1) performent généralement bien pour les textures stochastiques et même semi-stochastiques, mais elles éprouvent des difficultés avec certains types de structures. La difficulté principale avec ces méthodes est de bien établir les paramètres du modèle de texture employé. Il est en effet difficile de le faire pour générer un vaste éventail de types de structures.

Les méthodes de génération multirésolution par texel (§3.1.2.1) s'en tirent généralement mieux (figures 3.1 et 3.5), mais pas autant pour des textures structurées plus globalement. Un exemple de mauvais résultat est à la figure 3.16 (De Bonet [1997]). Le problème vient du fait que

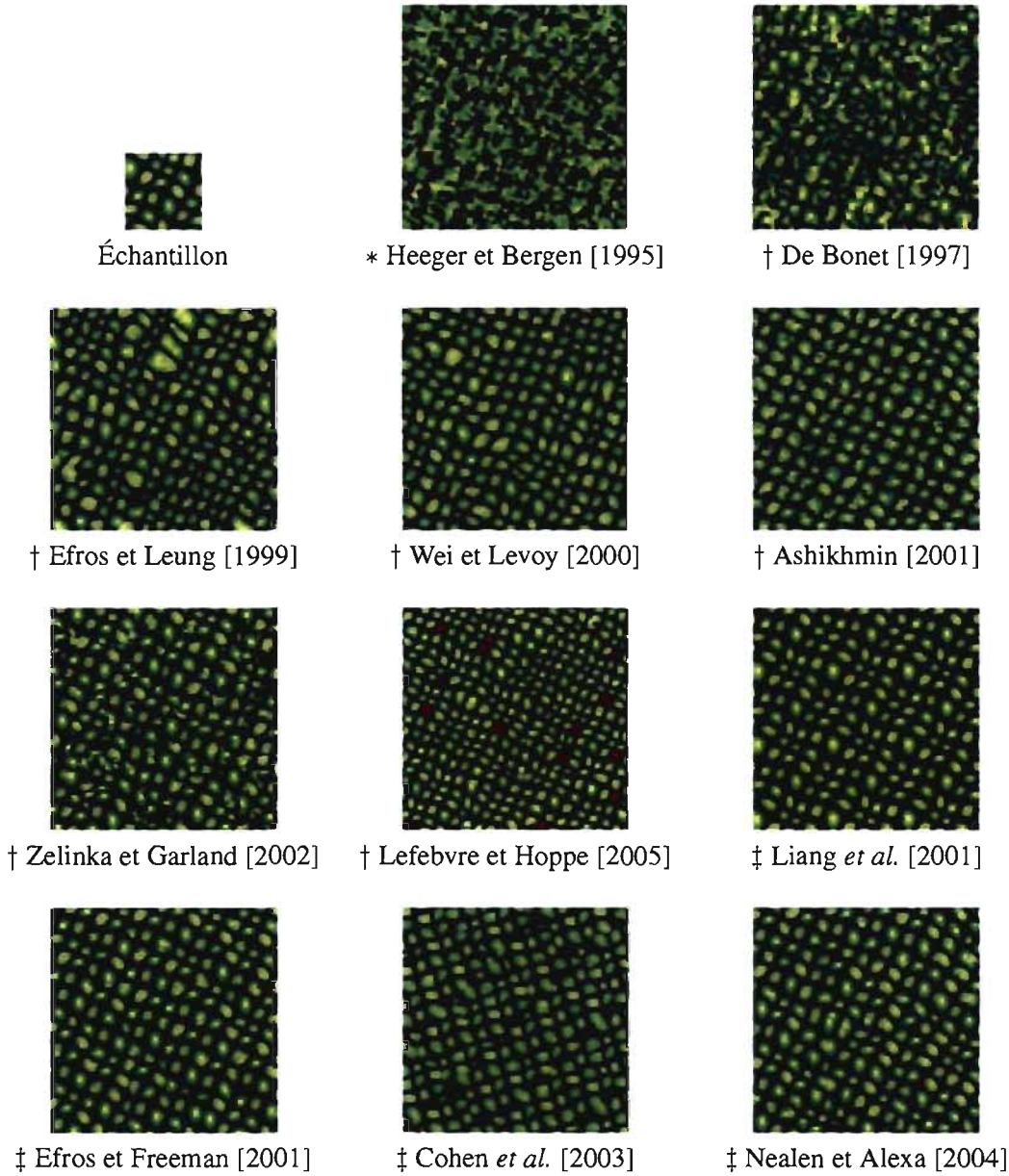


FIG. 3.16 – Comparaison de méthodes de génération paramétrique (∗), par texel (†) et par *patch* (‡). Toutes les images proviennent de la référence correspondante, à l'exception des trois premières qui proviennent de [Wei et Levoy 2000].

la recherche de texels se fait uniquement en fonction de la suite des parents. Ainsi, les méthodes de génération par texel avec recherche de voisinage (§3.1.2.2), par la contrainte explicite de similarité du voisinage, arrivent à obtenir de meilleurs résultats, tels que montrés à la figure 3.16 (de Efros et Leung [1999] à Lefebvre et Hoppe [2005]). La différence entre le résultat de Efros

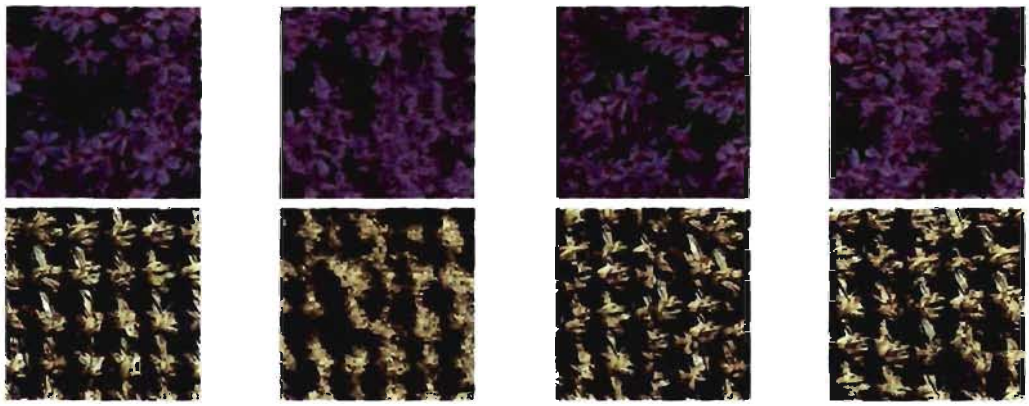


FIG. 3.17 – **Résultats de génération pour des textures structurées avec des méthodes par texel avec recherche de voisinage. Les images proviennent de [Hertzmann *et al.* 2001].**

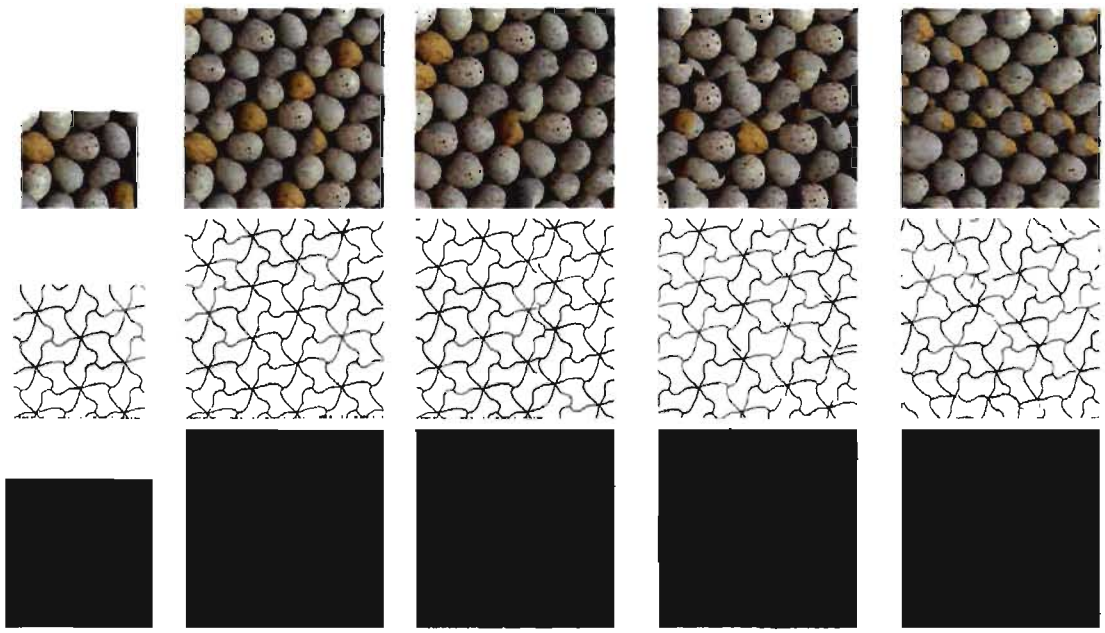


FIG. 3.18 – **Comparaison de différentes méthodes de génération par *patch* (∗) et par analyse structurale (†). Images tirées de [Wu et Yu 2004].**

et Leung [1999] et Wei et Levoy [2000] montre l'avantage de la génération multiniveau pour mieux préserver la distribution des structures.

Malgré l'emploi d'une génération multiniveau [Wei et Levoy 2000], par cohérence [Ashikhmin 2001] ou un hybride entre les deux [Hertzmann *et al.* 2001], ces méthodes ne performent

pas toujours bien pour certaines textures structurées. Quelques exemples de trouvent à la figure 3.17. Il faudrait augmenter la taille du voisinage à un point tel que la recherche deviendrait soit trop lente ou trop contrainte. Les seules méthodes de génération par texel qui arrivent à produire des résultats généralement plus satisfaisants sont celles qui incluent un aspect global dans la génération, soit par la génération parallèle avec des passes de correction [Wei et Levoy 2002, Lefebvre et Hoppe 2005] ou par une optimisation globale [Kwatra *et al.* 2005] (§3.1.2.3). Autrement, il faut plutôt regarder du côté des méthodes de génération par *patch*.

En effet, avant l'avènement des méthodes de génération par analyse structurale, les méthodes de génération par *patch* (§3.1.3) étaient considérées comme l'état de l'art en matière de génération de textures, en particulier avec la méthode par coupe de graphe [Kwatra *et al.* 2003]. Les méthodes de génération par *patch* précédentes à cette dernière éprouvaient souvent plus de difficultés à bien joindre une nouvelle *patch* avec la texture générée, soit parce que les caractéristiques n'étaient pas parfaitement alignées pour obtenir un bon mélange en dégradé [Liang *et al.* 2001], ou parce que le chemin de coupe était trop contraint [Efros et Freeman 2001]. Les méthodes de génération par tuile de Wang [Cohen *et al.* 2003, Ng *et al.* 2005, Lagae et Dutré 2006] produisent généralement de belles jonctions entre les *patches*, mais la répétition des mêmes tuiles peut être discernable.

Enfin, il y a les méthodes de génération par analyse structurale. La figure 3.18 compare quelques résultats entre certaines méthodes de génération par *patch* et par analyse structurale. Ces dernières sont sans doute les méthodes donnant parmi les meilleurs résultats, mais souvent au prix d'une implémentation plus complexe ou même d'une spécialisation sur le type de textures pouvant être bien générées. Par exemple, la méthode de Liu *et al.* [2004b] produit généralement les meilleurs résultats pour les textures quasi régulières, une classe de textures fréquemment rencontrées, mais il n'est cependant pas indiqué de l'utiliser pour des textures stochastiques. D'autres méthodes demeurent plus génériques, comme celle de Wu et Yu [2004], mais la correspondance des contours peut être problématique s'ils sont présents en trop grand nombre. La seule méthode de cette catégorie qui peut être considérée complètement générique est celle de Lefebvre et Hoppe [2006].

3.1.6 Génération adaptée aux surfaces

Du point de vue de la modélisation 3D, la destination finale d'une texture est son placage sur une surface 3D. Nous avons déjà mentionné qu'une des motivations de la génération de textures

est d'éviter les problèmes associés au placage de textures. Cependant, toutes les méthodes vues jusqu'ici n'abordent pas ce problème, en dehors de pouvoir générer une texture plus grande sans répétition. C'est ainsi que les méthodes de génération adaptées aux surfaces sont apparues. Il y a principalement trois approches : le coloriage de sommets [Wei et Levoy 2001, Turk 2001, Zhang *et al.* 2003, Bangay et Morkel 2006, Han *et al.* 2006], la génération d'une texture adaptée à la paramétrisation d'une surface [Ying *et al.* 2001, Lefebvre et Hoppe 2006] et l'approche inverse de la dernière, soit la génération d'une paramétrisation adaptée à la texture source [Soler *et al.* 2002, Magda et Kriegman 2003, Zelinka et Garland 2003, Zelinka et Garland 2004a, Sabha et Dutré 2006].

Les méthodes utilisant l'approche par coloriage de sommets procède par une adaptation directe des méthodes de génération de textures. Cependant, le voisinage autour d'un sommet d'un maillage n'a généralement pas la régularité de l'arrangement des texels dans une texture. Ainsi, pour faire retrouver un voisinage dans la texture source, Wei et Levoy [2001] utilisent un aplanissement local [Praun *et al.* 2000] suivi d'une interpolation aux points d'une grille régulière. Une autre approche est d'utiliser un parcours de surface (*surface marching*) [Turk 2001, Zhang *et al.* 2003]. L'orientation de la texture sur la surface est dirigée par un champ de directions (*direction field*), souvent fourni par l'usager. Pour les méthodes qui adaptent des techniques de génération par recherche de voisinage [Wei et Levoy 2001, Turk 2001, Zhang *et al.* 2003] ou par *patch* [Bangay et Morkel 2006], ce même champ de directions sert aussi pour déterminer l'ordre de génération, à moins qu'une technique parallèle soit utilisée [Wei et Levoy 2001]. Cet ordre n'est pas requis pour les méthodes qui font usage d'une optimisation globale [Han *et al.* 2006], comme celle de Kwatra *et al.* [2005] (§3.1.2.3). La génération est habituellement à plusieurs niveaux de résolution du maillage, quand ce n'est pas fait par l'intermédiaire de surfaces de subdivision [Bangay et Morkel 2006].

La génération d'une texture adaptée à la paramétrisation d'une surface, apportée par Ying *et al.* [2001] et reprise par Lefebvre et Hoppe [2006], procède essentiellement en remplissant une texture déjà plaquée sans répétition sur une surface. Une méthode quelconque de génération peut être utilisée, mais tout voisinage dans la texture générée doit être déformé en fonction de sa mise en correspondance sur la surface. Comme l'approche précédente, un champ de directions est aussi utilisé pour déterminer l'orientation de la texture sur la surface. La texture résultante paraîtra distordue dans son domaine, mais pas une fois appliquée sur la surface.

Une autre méthode de génération adaptée aux surfaces est l'approche inverse de la précédente, *i.e.* une paramétrisation de la surface est construite en fonction de la texture source. Le principe de base est de commencer par assigner les coordonnées de texture d'un triangle du maillage, puis de parcourir les autres triangles de voisin en voisin en attribuant des coordonnées de texture de sorte à limiter la discontinuité entre deux triangles voisins. Un mélange (*blending*) le long des arêtes peut être requis [Magda et Kriegman 2003], voire même un mélange de plusieurs couches [Zelinka et Garland 2003, Zelinka et Garland 2004a]. La génération peut être aussi faite de façon hiérarchique par groupe de triangles [Soler *et al.* 2002]. Un cas particulier de l'approche par construction de paramétrisation est la méthode de Sabha et Dutré [2006], où la continuité des jonctions est effectuée en modifiant la texture source, et donc il ne peut pas y avoir d'intersection entre les triangles correspondants dans le domaine de la texture. Par conséquent, il peut être requis que la texture soit agrandie par une méthode de génération 2D.

3.1.7 Applications

La génération de textures, hormis son but premier, a été employée dans plusieurs autres applications. Ismert *et al.* [2003] et Wang et Mueller [2004] ont fait usage de la génération de textures pour générer des détails de sous-résolution, *i.e.* de pouvoir augmenter la résolution en comblant l'espace introduit entre les pixels par une génération de détails plutôt que par une interpolation. D'autres l'utilisent comme outil d'édition d'images. La génération de textures peut être utilisée pour effectuer du remplacement dans une image [Igehy et Pereira 1997, Bertalmio *et al.* 2003, Criminisi *et al.* 2003, Drori *et al.* 2003, Sun *et al.* 2005, Pavić *et al.* 2006, Eisenacher *et al.* 2008]. Brooks et Dodgson [2002] font de l'édition d'images par auto-similarité pour répliquer une opération à d'autres endroits similaires dans l'image. Elle peut aussi être utilisée pour changer l'apparence (la texture ou le matériau) des objets dans une image [Fang et Hart 2004, Dong et Chantler 2005, Zelinka *et al.* 2005].

La notion de génération de textures a également été portée sur la dimension temporelle, *i.e.* de générer une séquence vidéo apériodique arbitrairement longue à partir d'une séquence échantillon. Wei et Levoy [2000] (§3.1.2.2) ont déjà montré une application de leur méthode pour la génération d'une séquence vidéo. Bar-Joseph *et al.* [2001] (§3.1.2.1) et Kwatra *et al.* [2003] (§3.1.3) avaient aussi adapté leurs méthodes pour les séquences vidéo. Autrement, Schödl *et al.* [2000] ont été les premiers à développer une méthode dédiée aux séquences vidéo, avec une méthode similaire à celle des cartes de sauts [Zelinka et Garland 2002] (en fait, Zelinka et Gar-

land s'étaient inspirés de la leur). Doretto *et al.* [2003], quant à eux, utilisent une approche plus rigoureuse en définissant un modèle stochastique similaire à celui de Zhu *et al.* [1998]. Enfin, Bhat *et al.* [2004a] ont développé une méthode simple de génération et d'édition de vidéo basée sur l'écoulement continu de phénomènes naturels (par exemple, une chute d'eau).

Les applications de la génération de textures ne se limitent pas aux images ou aux séquences vidéo. Elle peut servir pour générer des distributions d'éléments [Cohen *et al.* 2003, Lagae et Dutré 2005, Lagae et Dutré 2006, Ijiri *et al.* 2008], par exemple dans une scène 3D. Taponecco et Alexa [2003] l'utilisent pour la visualisation de champs de vecteurs. Nous retrouvons aussi la génération de son [Parker et Chan 2003] et de mouvements de personnages articulés [Arikan et Forsyth 2002, Li *et al.* 2002, Pullen et Bregler 2002, Deng *et al.* 2003]. Tong *et al.* [2002] et Liu *et al.* [2004a] ont également repris les idées de la génération de textures sur les surfaces pour générer des BTFs. D'un intérêt plus particulier pour nous, il y a aussi la génération de géométrie, que nous couvrons à la prochaine section.

3.2 Génération de géométrie

Comme nous l'avons mentionné à la §3.1, une texture peut décrire autre chose que la couleur d'un point sur une surface. Nous avons donné l'exemple de carte de déplacement. Lorsqu'une texture affecte la géométrie d'un objet, nous parlons alors de textures géométriques. Un exemple est illustré à la figure 3.19. Cependant, une carte de déplacement peut exprimer seulement certaines catégories de textures géométriques.

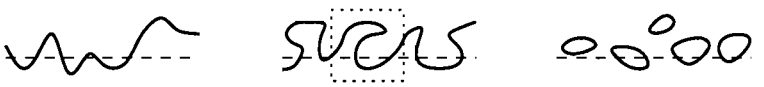
Nous classifions les textures géométriques selon deux aspects. D'abord, il y a le type, dans le sens utilisé pour les textures, *i.e.* stochastique *vs.* structuré (figure 3.2), que nous appelons dans ce contexte le type fondamental. Puis il y a le type d'altération géométrique que fait subir la texture à la surface, que nous appelons le type géométrique. Nous considérons trois principaux types géométriques :

1. les déplacements dans l'axe normal, *i.e.* les cartes d'élévation (*height maps*) ;
2. les déformations continues généralisées ou cartes de déplacements[11], *i.e.* toute déformation produisant une surface continue ne pouvant pas être exprimée par des cartes d'élévation ;

[11] Par rapport à une carte d'élévation, une carte de déplacement est une généralisation où le déplacement n'est pas contraint à l'axe normal mais dans une direction arbitraire.

Surface sans détail Surface avec détails

FIG. 3.19 – Les textures géométriques contribuent aux détails géométriques à la surface d'un objet.



(a) (b) (c)

FIG. 3.20 – Types géométriques : (a) déplacement dans l'axe normal, (b) déformation continue généralisée, (c) changement de topologie. Le trait en pointillé indique la surface de base.

3. les changements de topologie, *i.e.* ce qui peut introduire des trous, des poignées, ou des composantes non connexes.

La figure 3.20 illustre en 2D les trois types géométriques. Bien entendu, ces types ne sont pas exclusifs les uns des autres. Par exemple, une texture géométrique peut être représentée par une carte d'élévation et aussi produire des changements de topologie (nous reparlerons de cet exemple à la §3.2.1.1). De plus, le type 1 est en fait un cas particulier, mais commun, du type 2.

De l'ensemble des travaux qui ont été réalisés jusqu'à maintenant sur la génération de textures géométriques, nous distinguons deux approches principales. Il y a l'application directe des méthodes de génération de textures (§3.2.1) pour laquelle il s'agit de traduire le problème de génération de textures géométriques en un problème d'exprimer une texture géométrique par une *image* pour ensuite appliquer les méthodes de génération de textures. L'autre approche passe par la manipulation directe de la géométrie et adapte plutôt les méthodes de génération de textures (§3.2.3).

Nous avons illustré les types géométriques par des courbes à la figure 3.20. Or, il se trouve que les premiers travaux sur la génération de géométrie[12], en dehors de la génération de cartes d'élévation (§3.2.1.1), traitaient de génération de courbes ou d'ensembles de courbes. Nous parlons de ces travaux à la §3.2.2.

Nous pouvons aussi penser à étendre le concept de génération de textures à plus haut niveau en terme de géométrie pour construire non seulement les détails géométriques, mais les objets eux-mêmes. Dans le sens large, nous parlons alors de génération de géométrie[13]. Bien que cette thèse porte plutôt sur la notion de texture géométrique, par souci de complétude, nous effectuons aussi un survol des techniques de génération de géométrie axées sur l'édition ou la construction d'objets à la §3.2.4.

3.2.1 Application directe des méthodes de génération de textures

Pour être en mesure de *directement* appliquer les méthodes de génération de textures, il faut arriver à exprimer la géométrie sous forme d'*image*, ou de grille régulière. Il y a trois approches : la génération de cartes d'élévation (§3.2.1.1), la génération d'images de géométrie (§3.2.1.2) et la génération volumétrique (§3.2.1.3).

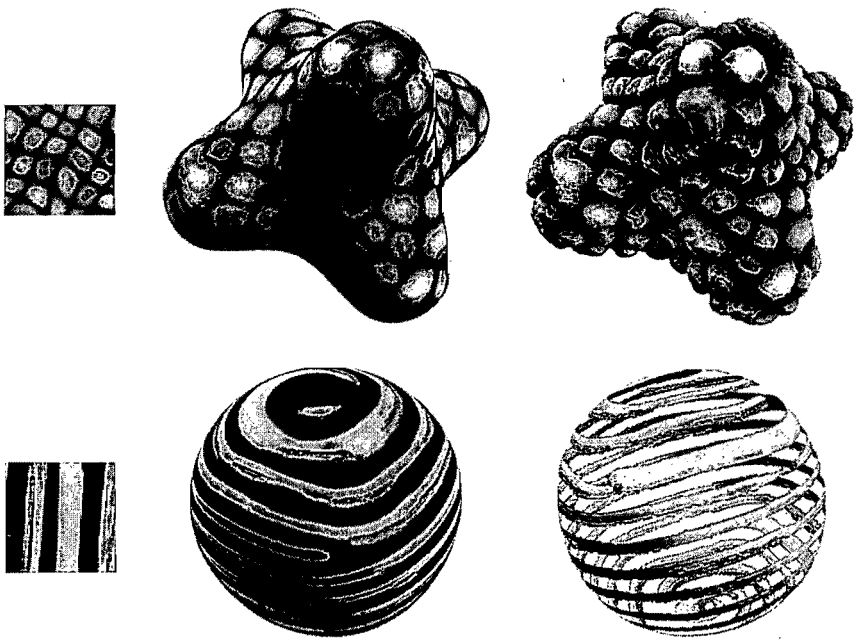
3.2.1.1 Génération de cartes d'élévation

Il s'agit sans doute de la méthode la plus simple, voire même triviale, pour produire des textures géométriques. Elle a d'ailleurs été mentionnée par Wei et Levoy [2001] et Ying *et al.* [2001]. Il suffit de générer des textures en tons de gris, où noir correspond à un déplacement de 0 et blanc au déplacement maximal. La figure 3.21 montre deux exemples.

A priori, les cartes d'élévation peuvent seulement exprimer des textures géométriques de type 1 (déplacement dans l'axe normal). Il est cependant possible d'obtenir une forme de type 3 (changement de topologie) si de la transparence est utilisée avec la texture (figure 3.21 en bas à droite), soit en ajoutant un canal de couleur indiquant la transparence ou en étiquetant une couleur comme transparente. Par contre, ceci non seulement change la topologie, mais aussi la surface n'est plus localement fermée.

La contrainte de déplacement dans l'axe normal pourrait être contourné avec l'usage de champs d'écoulement (*flow fields*) [Pedersen 1994], ce qui ferait en sorte de déplacer le problème vers comment déformer le champ d'écoulement régulier (*i.e.* celui qui émane en ligne

[12]Nous distinguons la génération de géométrie et la génération de textures géométriques au prochain paragraphe.
[13]D'où le titre plus général de cette section.



Échantillon Texture générée Carte d'élévation

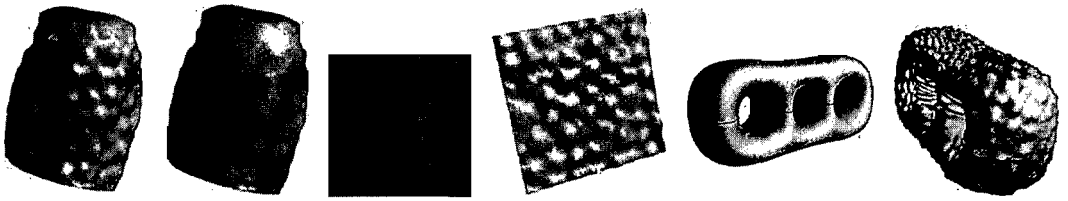
FIG. 3.21 – Génération de cartes d'élévation : une procédure de génération de textures est d'abord utilisée (colonne du milieu), puis la texture générée est appliquée comme carte d'élévation. Les changements de topologie en bas à droite sont produits par la transparence. Les images proviennent de [Ying *et al.* 2001].

droite en direction des normales à la surface) pour permettre d'avoir les caractéristiques de surface désirées. Ce problème est toutefois beaucoup moins intuitif à résoudre et beaucoup plus complexe.

3.2.1.2 Génération d'images de géométrie

Une autre façon de pouvoir appliquer directement les méthodes de génération de textures est de passer par une image de géométrie comme représentation [Lai *et al.* 2005, Nguyen *et al.* 2005]. Une image de géométrie est une image pour laquelle la couleur de chaque pixel est interprétée comme une position 3D. À la base, une telle image est produite en construisant une paramétrisation bijective entre le domaine $[0,1]^2$ et tous les points de la surface d'un objet [Gu *et al.* 2002].

Une texture géométrique est normalement définie de façon relative à une surface, et donc une image de géométrie avec position absolue ne peut être directement utilisée. Deux approches ont été proposées pour représenter une texture géométrique avec une image de géométrie. La



(a) (b) (c) (d) (e) (f)

FIG. 3.22 – Représentation et extraction d'une texture géométrique par une image de géométrie relative : (a) surface détaillée ; (b) surface lissée ; (c) image de géométrie relative extraite ; (d) reconstruction de la texture géométrique sur un plan ; (e) autre surface sur laquelle la texture sera appliquée ; (f) résultat de l'application. Les images proviennent de [Lai *et al.* 2005].

plus simple, proposée par Lai *et al.* [2005], est essentiellement de la convertir en carte de déplacement en encodant la différence de position entre un point de la texture et le point correspondant sur le plan du domaine paramétrique. Un exemple est donné à la figure 3.22 (d) et (e). La texture peut aussi être extraite d'un objet en reconstruisant une surface lisse, moins détaillée, et en prenant la différence de position entre un point de la surface détaillée et le point correspondant sur la surface lisse, le tout exprimé en coordonnées locales pour reconstruire le plan du domaine. Le résultat est une image de géométrie relative. Un exemple est illustré à la figure 3.22.

Une autre approche a été suggérée par Nguyen *et al.* [2005], qui s'intéressaient d'abord au problème de complétion de surface, ce qui est illustré à la figure 3.23. Ils définissent une texture géométrique par une paire d'images de gradient local. Pour construire une telle paire, ils calculent d'abord l'image de géométrie, puis le gradient est calculé selon les deux axes principaux de l'image (X et Y). Un système de coordonnées locales est calculé pour chaque point et chaque gradient est transformé dans ce système, produisant les gradients locaux.

Peu importe l'approche, au final, les images produites peuvent être maintenant directement utilisées avec des méthodes de génération de textures. Il faut ensuite une procédure pour appliquer les textures géométriques produites sur une surface. Dans le cas de la méthode de Lai *et al.* [2005], il s'agit d'une simple application d'une carte de déplacement [Cook 1984]. La situation n'est pas aussi simple pour la méthode de Nguyen *et al.* [2005]. En effet, une partie de l'information est perdue en considérant seulement les gradients locaux. Il faut d'abord cal-

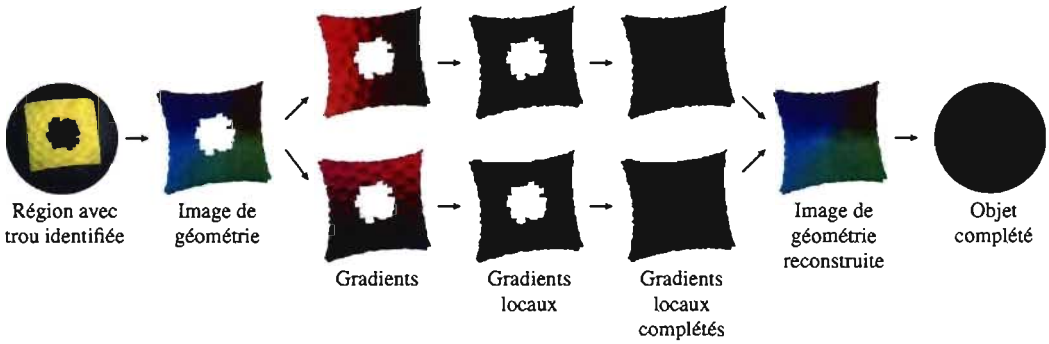


FIG. 3.23 – Complétion de géométrie par la méthode de Nguyen *et al.* [2005] : une image de géométrie est calculée à partir d'une région identifiée avec un trou ; les gradients sont ensuite calculés puis convertis en gradients locaux ; une technique de génération de textures est utilisée pour combler le trou et l'image de géométrie est reconstruite par une résolution itérative d'équations différentielles partielles. Les images proviennent de [Nguyen *et al.* 2005].

culer les gradients (non locaux) de la surface cible[14], puis une résolution itérative d'équations différentielles partielles doit être effectuée.

Relative ou non, une image de géométrie n'a aucune contrainte sur la valeur d'un pixel, et donc leur expressivité permet de représenter des textures de type géométrique 2. Cependant, il pourrait y avoir des problèmes pour bien paramétrer la surface pour être en mesure de représenter des textures avec de grandes distorsions tangentielles, *i.e.* avec des caractéristiques similaires à celle dans le carré en pointillé rouge de la figure 3.20 (b), mais encore plus proéminentes. Ce problème est encore plus présent avec la méthode de Nguyen *et al.* [2005].

Les méthodes présentées ne peuvent pas non plus exprimer des textures de type géométrique 3. Nous pourrions imaginer l'usage de transparence comme pour les cartes d'élévation (§3.2.1.1), mais avec les mêmes limitations. Des travaux ont cependant été faits pour gérer des topologies plus complexes avec des images de géométrie par l'intermédiaire de cartes multiples (*multi-charts*) [Sander *et al.* 2003, Carr *et al.* 2006], mais il n'est pas clair comment ceci peut arriver à efficacement gérer de petits détails géométriques avec une topologie complexe.

3.2.1.3 Génération volumétrique

Dans des travaux concurrents, Lagae *et al.* [2004, 2005] et Bhat *et al.* [2004b] ont développé des techniques provenant d'une extension directe de la génération de textures. Pour exprimer la géométrie des surfaces, ils utilisent des champs de distance réguliers, *i.e.* une grille de voxels

[14] Dans le cas d'une complétion, les gradients du *trou* sont initialisés par une simple interpolation linéaire.

(a) (b)

FIG. 3.24 – Résultats de la méthode de Lagae *et al.* [2004, 2005] : (a) génération d'un terrain et (b) d'un treillis. Les images sont tirées de [Lagae *et al.* 2004].

contenant chacun une information sur la distance entre le centre du voxel et le point le plus proche sur la surface de l'objet. Cette représentation capture mieux les informations géométriques qu'une représentation binaire indiquant si le voxel fait partie ou non de l'objet. Ils ont cependant utilisé des méthodes de génération différentes, mais que nous regroupons dans une même catégorie sous le nom de génération volumétrique. Il est à noter que, comme il s'agit d'extensions des méthodes de génération de textures, l'hypothèse de champs aléatoires de Markov comme modèle pour la géométrie est sous-entendue.

La méthode de Lagae *et al.* [2004, 2005] est inspirée des méthodes par *patch* de Efros et Freeman [2001] et Liang *et al.* [2001] (§3.1.3), seulement au lieu de *patches* 2D (imagettes), il s'agit de cubes contenant des voxels, et la méthode fonctionne en multiniveau, similairement à la méthode de Wei et Levoy [2000]. Contrairement à ces méthodes, elle ne fait pas usage de voisinage causal, par souci de simplicité[15], et alors le volume est initialisé avec du bruit blanc. Le temps de recherche brute pour des cubes de voxels de voisinages similaires est très grand (l'évaluation seule de la métrique est dans $\theta(k^3)$, k étant la taille d'un côté de la *patch*, typiquement avec une valeur de 16 ou 32). Ils ont développé une méthode hiérarchique similaire à un algorithme A* [Russell et Norvig 2002] pour accélérer la recherche. La hiérarchie (l'espace de recherche) est la pyramide multiniveau de l'échantillon de départ, et la recherche s'effectue de haut en bas selon un algorithme A*, où l'heuristique de recherche est la similarité entre les *patches*. La figure 3.24 montre deux exemples de résultats de la méthode. Contrairement aux

[15]En fait, les auteurs disent que de trouver un parcours dans le volume qui permet d'obtenir un voisinage causal en forme de "L" (ou un coin en 3D) comme pour les méthodes 2D de génération de textures par *patch* n'est pas facile.

méthodes 2D, elle ne fait aucun effort particulier pour régler les problèmes de discontinuités entre les *patches* voisines, et nous pouvons en observer les conséquences dans leurs résultats. Le champ de distance produit peut être converti en une texture volumétrique [Neyret 1995, Neyret 1996, Neyret 1998] ou en carte de déplacement généralisée [Wang *et al.* 2004] pour être appliqué comme texture géométrique.

La méthode de Bhat *et al.* [2004b] se base plutôt sur la méthode des analogies d'images de Hertzmann *et al.* [2001], seulement il s'agit d'une analogie de volumes[16]. En plus du champ de distance, les voxels ont les attributs suivants :

1. *densité volumétrique* : les voxels à l'intérieur de l'objet ont une valeur de 1, 0 à l'extérieur, et ceux qui sont près de la surface peuvent avoir une valeur intermédiaire, ce qui permet de réduire les effets de crénelage des représentations purement binaires. La densité sert de valeur pour construire le vecteur de caractéristiques pour la métrique de comparaison.

2. *système de coordonnées locales* : ceci est généré à partir d'un champ de directions pour permettre d'orienter la *texture géométrique* sur la surface de façon similaire à plusieurs méthodes de génération de textures adaptées aux surfaces (§3.1.6). Les valeurs du système de coordonnées locales calculées pour les surfaces sont propagées par interpolation pour les voxels intérieurs et extérieurs.

3. *distance de balayage* : ceci provient également d'une méthode de génération de textures adaptée aux surfaces (celle de Turk [2001]) et indique l'ordre de parcours des voxels de façon à toujours pouvoir créer un voisinage causal. La distance de balayage est propagée dans le volume le long des normales des voxels de la surface de l'objet.

La génération procède voxel par voxel en fonction de la distance de balayage. Pour construire le voisinage, les valeurs sont échantillonnées selon l'orientation du système de coordonnées locales. La méthode implante aussi une génération multiniveau comme Wei et Levoy [2000]. Étant une méthode qui ne fonctionne pas par *patches*, pour améliorer les résultats, une deuxième passe est effectuée. La figure 3.25 montre quelques exemples de résultats de la méthode. Nous pouvons voir que la méthode peut engendrer des changements topologiques indésirables : trous dans l'oreille du lapin et nageoire dorsale de l'hippocampe fragmentée. Le problème survient

[16]L'analogie d'images de Hertzmann *et al.* [2001] fonctionne en fournissant une segmentation avec la texture source et en initialisant la texture à générer avec des couleurs de la segmentation. La génération ajoute alors les texels correspondants dans ces images de segmentations comme la comparaison de voisinage dans la recherche. Dans le cas de la méthode de Bhat *et al.* [2004b], la "segmentation" initiale correspond à la surface de l'objet.

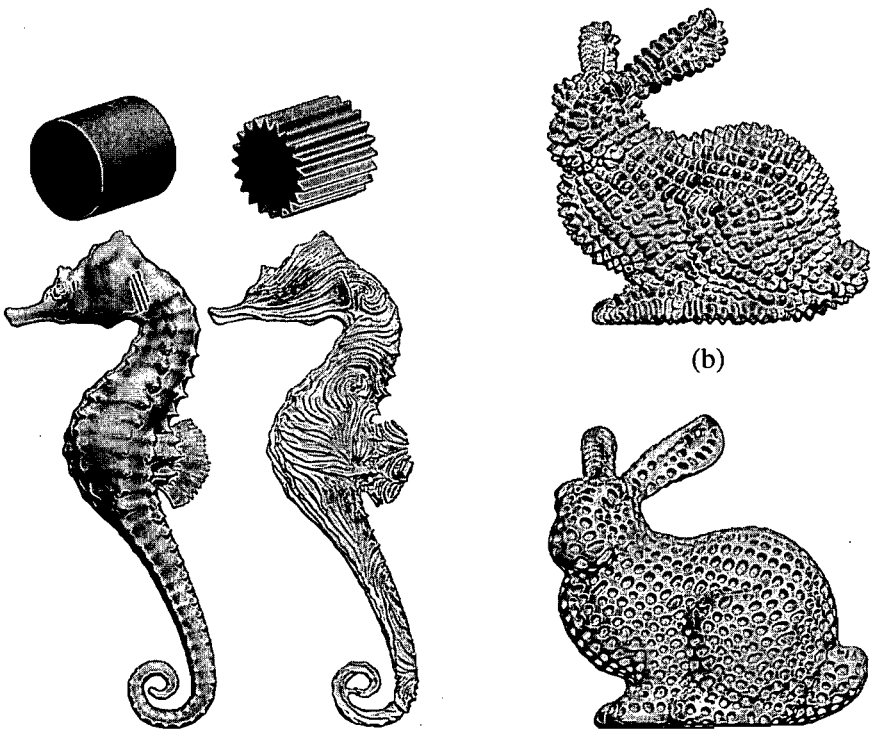


FIG. 3.25 – Résultats de la méthode de Bhat *et al.* [2004b] : illustration de l'analogie par un ajout (a) de cannelures à un hippocampe, (b) de pointes et (c) de concavités sur le lapin. Les images sont tirées de [Bhat *et al.* 2004b].

lorsque l'échelle de la texture géométrique est plus grande que celle des détails déjà présents sur la surface de l'objet.

Avec l'usage d'une représentation volumétrique, les méthodes décrites dans cette section peuvent exprimer des textures de type géométrique. Cependant, les temps de calcul de ces méthodes sont longs (les auteurs rapportent des temps allant de une à douze heures sur des ordinateurs standard en 2004). De plus, les données volumétriques prennent rapidement beaucoup de mémoire en augmentant la résolution.

3.2.2 Des courbes et leur distribution

Nous insérons ici un petit intermède pour présenter des travaux réalisés pour un cas particulier de génération de géométrie, soit la génération de courbes 2D, avant de poursuivre avec les autres méthodes pour la génération de géométrie 3D. Il s'agit en effet de travaux précurseurs

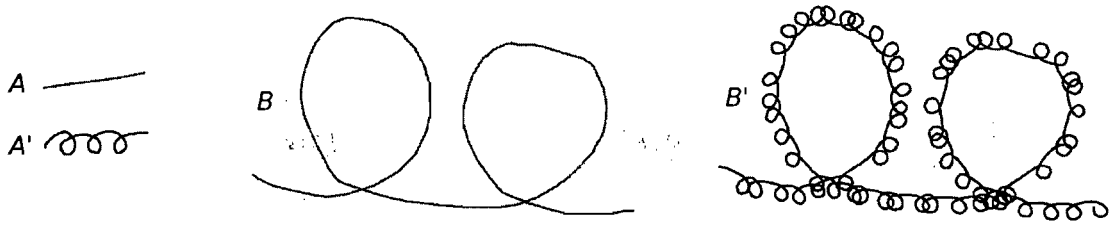


FIG. 3.26 – Résultat de la méthode de Hertzmann *et al.* [2002] : A et A' forment la paire d'analogies, et B est la courbe sur laquelle une transformation similaire est appliquée. Le résultat est B'. L'image est tirée de [Hertzmann *et al.* 2002], mais elle a été modifiée par souci de clarté.

aux autres qui montrent comment les idées de génération de textures peuvent être utilisées sans recourir à des images.

Hertzmann *et al.* [2002] reprennent leurs travaux sur les analogies d'images [Hertzmann *et al.* 2001][17] pour générer des courbes. Ces dernières sont représentées par un ensemble continu de segments de droite. La méthode cherche à produire une courbe de longueur arbitraire qui ressemble à un échantillon de courbe sans être une réplique ou une suite de répétitions. La génération procède en ajoutant de nouveaux segments à l'extrémité de la courbe. Ces derniers sont choisis parmi les segments de l'échantillon dont la section de courbes voisines est la plus similaire. La distance entre un nombre fixé de paires de points, évaluée à des pas unitaires de longueur d'arc, sert de métrique de comparaison de courbe. Une rotation et une translation doivent être estimées pour rendre cette comparaison invariante à ces transformations. Avec cette métrique, la méthode par analogies d'images [Hertzmann *et al.* 2001] peut être simplement adaptée en substituant les images par des courbes. La figure 3.26 montre un résultat de la méthode. Zelinka et Garland [2004b] ont utilisé cette méthode pour induire des déformations le long de courbes sur une surface.

Jodoin *et al.* [2002] ont une méthode en quelque sorte complémentaire à la précédente. Ils partent avec l'hypothèse qu'une méthode de comparaison de traits, *i.e.* de courbes, soit disponible. Leur méthode génère ensuite une séquence de traits de longueur arbitraire qui ressemble à une séquence fournie (l'échantillon), mais sans être une copie ou une suite de répétitions. Pour ce faire, ils adaptent la méthode de Efros et Leung [1999] (§3.1.2.2) pour générer une distribution de traits. Un nouveau trait est généré en le choisissant parmi les traits de l'échantillon dont les traits voisins sont les plus similaires aux traits voisins dans la séquence générée. La figure 3.27 montre quelques résultats de leur méthode. À noter le niveau de structure qu'elle peut at-

[17] Voir la note de bas de page 16 à la page 79.

Échantillon Séquence générée

FIG. 3.27 – Résultats de la méthode de Jodoin *et al.* [2002]. Images tirées de [Jodoin *et al.* 2002].

teindre. Un des grands problèmes de la génération de textures est la divergence visuelle qui se produit lors de génération de grande taille, *i.e.* à partir d'un certain moment, une méthode peut se mettre à générer ce qui s'apparente plus à du bruit[18]. Ceci se produit lorsqu'une méthode se met à réutiliser constamment un sous-ensemble des texels de l'échantillon. Jodoin *et al.* [2002] abordent ce problème pour leur génération de séquences de traits en évaluant la distance entre la distribution globale de la séquence générée et celle de l'échantillon[19]. Si cette dernière est trop grande, ils corrigent quelques traits et reprennent la génération.

3.2.3 Génération par manipulation de surfaces

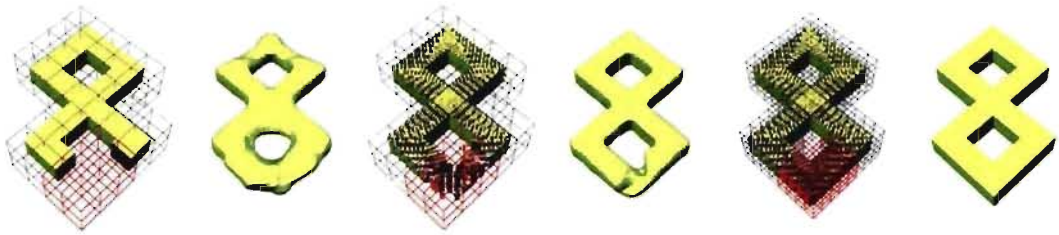
De la même façon que les travaux en génération de courbes adaptent les idées de génération de textures à la manipulation de courbes plutôt que de transformer le problème dans un espace sur grille régulière, des travaux ont aussi été faits pour la génération de surfaces 3D. La plupart d'entre eux ont été développés dans un contexte de complétion de surfaces, *i.e.* de remplir des trous dans la surface d'un objet, ou de transfert de textures.

Les premiers travaux ont été ceux de Sharf *et al.* [2004], qui, à la base, s'intéressaient au remplissage contextuel, *i.e.* à la réparation d'objets contenant des trous, de telle sorte à ce que chacun soit rempli par une surface visuellement similaire à son voisinage, mais leur méthode pourrait très bien s'appliquer aussi pour le transfert de textures[20]. Elle procède par décomposition multiniveau de l'objet avec un *octree*. Ceci permet de remplir les trous d'abord grossièrement, puis en raffinant la solution, similairement à la méthode multiniveau de génération de textures de Wei et Levoy [2000] (§3.1.2.2). Dans un contexte de transfert, il suffirait d'initialiser

[18]Ceci n'est pas vrai pour toutes les méthodes, notamment pour les méthodes de génération parallèle [Wei et Levoy 2002, Lefebvre et Hoppe 2005], d'optimisation globale [Kwatra *et al.* 2005], et les méthodes de génération par tuile de Wang [Cohen *et al.* 2003, Ng *et al.* 2005, Lagae et Dutré 2006], et toutes méthodes dérivées de ces dernières. De plus, les méthodes de génération par *patch*, si elles divergent, auront tendance à le faire beaucoup plus lentement.

[19]Ils le font à l'aide d'une formule dérivée de la distance de Kullback-Leibler [1951].

[20]Les auteurs ne font aucune mention de cette application ni des ajustements nécessaires à leur méthode dans ce contexte. Les modifications que nous suggérons sont donc de nature spéculative car nous n'avons pas implémenté cette méthode pour la valider.



(a) (b) (c) (d) (e) (f)

FIG. 3.28 – Remplissage d'un trou suivant la méthode de Sharf *et al.* [2004] : (a) à partir du niveau le plus grossier d'une décomposition multiniveau avec *octree*, (b) une approximation de la surface manquante est calculée, (c) ce qui sert de guide pour générer à ce niveau ; (d) une nouvelle approximation plus fine de la surface est calculée et (e) la complétion reprend à un niveau de résolution plus fin, pour finalement obtenir le résultat en (f). Les images proviennent de [Sharf *et al.* 2004].

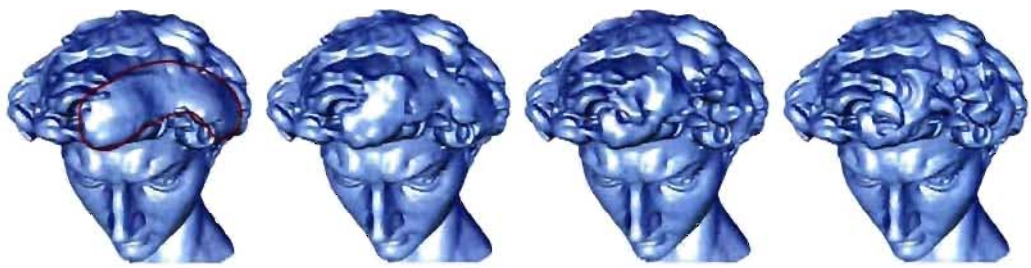


FIG. 3.29 – Progression multiniveau de la méthode de Sharf *et al.* [2004] : une région grossière indiquée en rouge à gauche est d'abord remplie par une surface très approximative, puis au fur et à mesure que les cellules d'*octree* sont subdivisées, la région est raffinée pour inclure plus de détails transférés d'autres régions de l'objet. Les images proviennent de [Sharf *et al.* 2004].

la surface cible par une approximation lisse. La représentation géométrique utilisée est le point (§2.3) pour éviter les problèmes de topologie et du maintien de la connectivité d'un maillage.

Pour chaque cellule à générer, une approximation de la forme est calculée à l'aide d'une fonction implicite polynomiale locale. Une cellule est à générer si elle est identifiée comme un trou ou, dans le contexte d'un transfert, comme une cible. Le remplissage consiste alors à trouver une cellule valide, *i.e.* sans trou et non cible, qui pourrait provenir d'un autre objet, dont la signature de la forme de son parent est la plus similaire à celle du parent de la cellule à remplir. La signature consiste en un échantillonnage de divers attributs (par exemple la distance à la surface) de la fonction implicite d'une cellule et de ses six cellules voisines. La métrique de comparaison est alors une somme pondérée de la distance euclidienne entre les vecteurs

d'échantillons des divers attributs. Le contenu de la cellule trouvée est copié dans la cellule à remplir en appliquant une transformation non rigide pour aligner les points et joindre les frontières. Pour ce faire, une procédure ICP (*iterative closest point* ou procédure itérative de points les plus près) [Besl et McKay 1992, Chen et Medioni 1992, Zhang 1994] avec l'ajout d'un terme non rigide est employée. La figure 3.28 illustre la méthode dans le contexte de complétion, et la figure 3.29 en montre la progression multiniveau. Il est à noter que, puisqu'un *octree* divise l'information géométrique par des plans alignés avec les axes principaux, pour enrichir l'ensemble de candidats, l'objet est pivoté quelques fois (par exemple à tous les $\frac{\pi}{4}$ autour des axes principaux) et un *octree* additionnel est construit pour chaque pivotement.

Il n'y a *a priori* pas de restriction sur le type géométrique du contenu des cellules pour la méthode de Sharf *et al.* [2004], mais la signature employée est telle que nous ne pouvons pas vraiment espérer bien traiter le type géométrique 3. De plus, la procédure ICP avec le terme non rigide ne peut pas induire des déformations très complexes[21].

Une autre approche pour résoudre le même problème de remplissage contextuel, profitant aussi de la représentation par points, est celle de Park *et al.* [2005]. Un *octree* est aussi utilisé pour découper l'objet en *patches*. Les *patches* avec des trous sont comblées en cherchant d'autres *patches* au contexte similaire. Dans leur cas, pour être en mesure de comparer des *patches*, une paramétrisation locale est d'abord calculée pour chaque *patch*, et une signature est composée à partir de la courbure de la surface de chaque point de la *patch*. Une autre signature est calculée à partir des couleurs pour pouvoir aussi traiter la couleur. La *patch* avec la signature la plus similaire est choisie. Pour joindre la *patch* choisie avec le voisinage, un alignement par

[21] Nous avons implémenté cette méthode de déformation et dans le contexte de la méthode de génération de textures géométriques que nous présenterons au chapitre 4, ce n'était clairement pas assez puissant §(4.4.2).

FIG. 3.30 – Résultats de la méthode de complétion de Park *et al.* [2005] : pour chaque paire, l'objet à gauche contient un trou qui est rempli à droite. Les images proviennent de [Park *et al.* 2005].

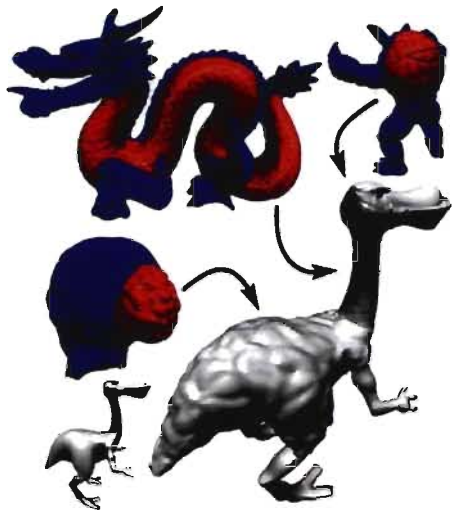
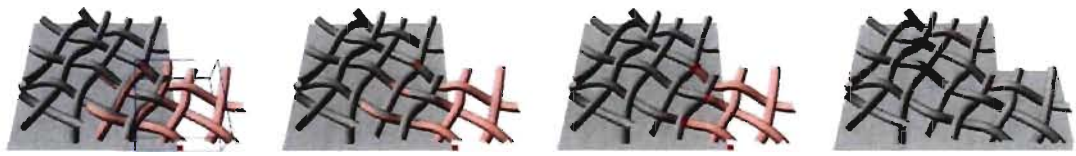


FIG. 3.31 – Transfert de textures géométriques par la méthode de Zelinka et Garland [2006] : des régions sources sont sélectionnées sur des objets et les textures géométriques s'y trouvant sont transférées dans des régions cibles sur un autre objet. Image tirée de [Zelinka et Garland 2006].

ICP est d'abord effectué, comme pour la méthode précédente, mais la déformation est faite par la résolution d'équations différentielles partielles de Poisson avec des conditions frontalières de Dirichlet [Pérez *et al.* 2003, Yu *et al.* 2004]. La figure 3.30 montre deux résultats de la méthode de Park *et al.* [2005]. L'usage de paramétrisation locale dans leur méthode écarte la possibilité d'occurrences du type géométrique 3 et, de la même façon qu'avec les images de géométrie (§3.2.1.2), limite aussi l'ampleur du type géométrique 2.

Zelinka et Garland [2006] ont aussi une approche faisant usage de paramétrisation locale. Leurs travaux portent sur le transfert de texture et s'inspirent de la méthode de coupe de graphe de Kwatra *et al.* [2003] (§3.1.3). Le principe de fonctionnement est très similaire à la méthode de Lai *et al.* [2005] (§3.2.1.2) mais aucune image de géométrie relative n'est construite. La méthode reste au niveau du maillage et se sert d'une paramétrisation locale mutuelle entre deux *patches* pour faire correspondre les sommets. De plus, contrairement à la plupart des méthodes de génération par *patch*, la génération ne transfère pas des *patches* de taille fixe. Plutôt, un voisinage de taille fixe autour d'une position cible est utilisé pour chercher une position dans la source au voisinage similaire. La recherche se fait par l'intermédiaire d'éventails géodésiques [Zelinka et Garland 2004c]. Une fois une position source trouvée, une *patch* est sélectionnée autour de cette position en grandissant une région tant que la paramétrisation locale a peu de distorsion. Une *patch* destination est sélectionnée en cohérence avec la source, établis-



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

FIG. 3.32 – Procédure de génération de textures géométriques de Zhou *et al.* [2006] : (a) une *patch* avec le meilleur alignement avec la texture générée est choisie, (b) puis une déformation est effectuée pour améliorer l'alignement des composantes ; (c) une coupe optimale est déterminée pour joindre les composantes pour finalement produire de résultat en (d). Les images proviennent de [Zhou *et al.* 2006].

sant ainsi la paramétrisation mutuelle. La *patch* source est ensuite transférée et elle est jointe par une adaptation de la coupe de graphe de Kwatra *et al.* [2003], où les faces remplacent les texels, et le poids des arêtes est déterminé en fonction de la différence de signal (déplacement ou couleur) le long de l'arête entre les deux faces correspondantes. La figure 3.31 donne un exemple de résultat de la méthode de Zelinka et Garland [2006]. En terme de type géométrique de texture, cette méthode partage les mêmes restrictions que les méthodes de génération d'images de géométrie (§3.2.1.2).

Dans des travaux concurrents aux nôtres, présentés au chapitre 4, Zhou *et al.* [2006] ont présenté une méthode de génération pouvant bien gérer tous les types géométriques de textures. Ils représentent leurs textures géométriques par un maillage de géométrie arbitraire inclus dans une boîte alignée sur les axes, où l'axe le plus petit est celui de l'épaisseur. Comme Zelinka et Garland [2006], ils ont aussi une approche inspirée de génération de textures par *patch* (§3.1.3) par coupe de graphe [Kwatra *et al.* 2003], mais il s'agit d'une méthode axée vraiment sur la génération et non sur le transfert ou la réparation.

La procédure de génération est illustrée à la figure 3.32. Une boîte de nouvelles dimensions mais de même hauteur que la texture source est à remplir. Les boîtes (source et à remplir) sont subdivisées en grilles de pas réguliers dans les axes du plan de génération. Cette subdivision sert pour se ramener à un cas similaire aux méthodes de génération de textures colorées. Une *patch* est définie par la géométrie tombant dans une boîte formée par un groupe de cellules d'une grille. L'algorithme de génération procède ensuite de façon similaire aux méthodes de génération par *patch*, où chaque nouvelle *patch* est placée de sorte à créer une région de superposition. La *patch* dans la source pour laquelle sa géométrie est la mieux alignée avec

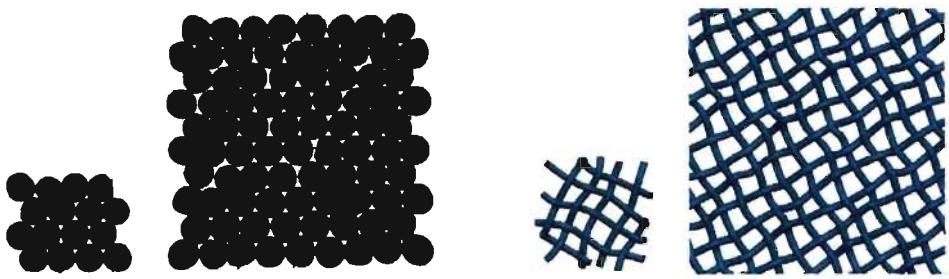


FIG. 3.33 – Résultats de génération planaire de la méthode de Zhou *et al.* [2006] : pour chaque paire, la source est à gauche et le résultat à droite. Les images proviennent de [Zhou *et al.* 2006].

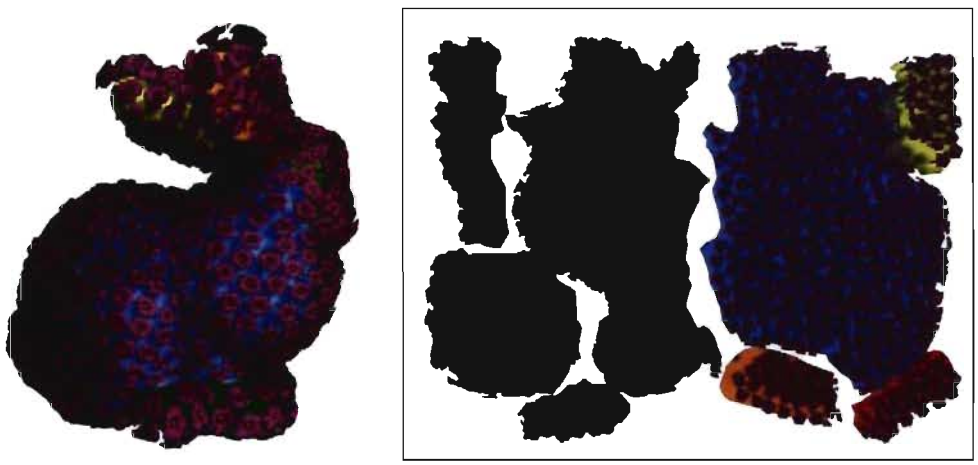
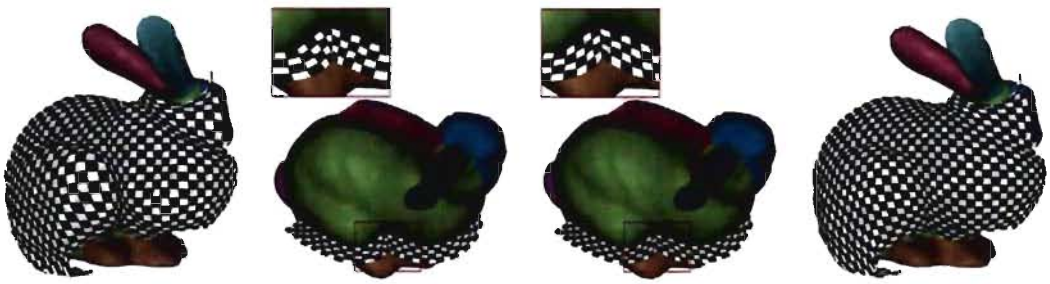


FIG. 3.34 – Résultat de génération adaptée aux surfaces de Zhou *et al.* [2006] : (gauche) la texture géométrique affichée sur le lapin est finalement représentée (droite) dans un atlas de cartes multiples. Les images proviennent de [Zhou *et al.* 2006].

la géométrie de la région de superposition est choisie (figure 3.32 (a)). Ceci est évalué par la somme des distances minimales entre sommets et faces en plus de la différence de normales. Les composantes connexes sont ensuite mises en correspondance et une déformation de type laplacien [Yu *et al.* 2004, Sorkine *et al.* 2004] est effectuée pour améliorer leur alignement (figure 3.32 (b)). Elles sont ensuite jointes par une coupe optimale de façon très similaire à la méthode de Zelinka et Garland [2006] décrite précédemment (figure 3.32 (c)). La figure 3.33 montre deux résultats de génération.

La méthode décrite précédemment est une méthode de génération planaire. Une méthode de génération adaptée aux surfaces est aussi présentée. Dans ce cas, les *patches* sont définies plutôt par groupes de triangles du maillage de la surface, où chaque groupe est sélectionné de façon quelque peu similaire à la méthode de Zelinka et Garland [2006], mais au lieu de



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

FIG. 3.35 – Optimisation de la paramétrisation de la couche extérieure d'une carte en couche (*shell map*) : (a) sans ajustement, une carte en couche peut produire des distorsions importantes sur la couche extérieure, (b) en particulier dans les concavités ; (c)-(d) une optimisation de la paramétrisation de la couche extérieure vient corriger la situation. Les images proviennent de [Zhou *et al.* 2006].

construire une paramétrisation mutuelle, la *patch* sélectionnée sur la surface est aplanie suivant une mise en correspondance conforme (*conformal mapping*) discrète [Desbrun *et al.* 2002] ou de moindres carrés [Lévy *et al.* 2002] pour se ramener localement au cas planaire. Les *patches* sont alors plutôt formées par des groupes de prismes à base triangulaire. Au final, la texture générée sur la surface est représentée dans un atlas de cartes multiples, comme illustré à la figure 3.34. Cette dernière étape est nécessaire pour être en mesure d'appliquer la texture sans intersection sur elle-même. Cette application peut être faite par l'intermédiaire de cartes en couche (*shell maps*) [Porumbescu *et al.* 2005], mais il peut y avoir des distorsions importantes sur la couche extérieure, comme illustré à la figure 3.35 (a) et (b). Une optimisation sur la paramétrisation de la couche extérieure est alors effectuée pour corriger la situation, comme montré à la figure 3.35 (c) et (d).

Plus de discussion sur la méthode de Zhou *et al.* [2006] se trouve au chapitre 4.

3.2.4 Génération par composition d'objets

La génération de textures géométriques s'intéresse aux détails géométriques se trouvant à la surface des objets. Il est certainement concevable de vouloir utiliser les principes de génération de textures pour générer des objets entiers plutôt que seulement leurs détails. Bien qu'il ne s'agisse pas de l'intérêt de cette thèse, nous mentionnons les quelques travaux qui ont été faits en la matière par souci de complétude avec le thème de génération de géométrie.

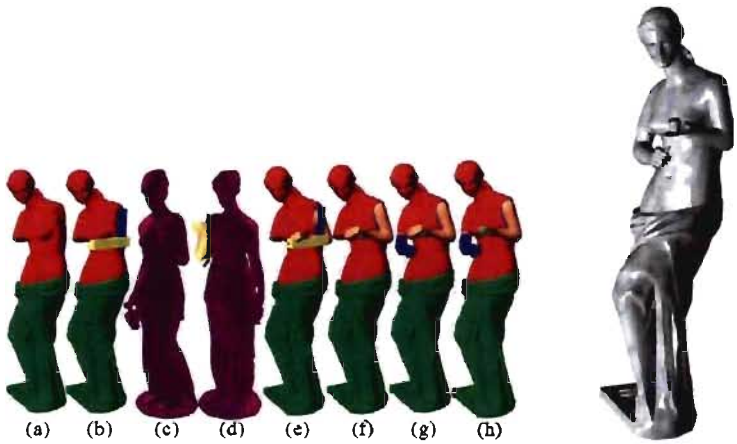


FIG. 3.36 – Séance de modélisation avec le système de Funkhouser *et al.* [2004] : (a) objet initial ; (b) sélection de la forme de recherche pour un bras ; (c) un des objets trouvés par la recherche est choisi par l'usager ; (d) découpage de la pièce d'intérêt ; (e-f) raccordement de la nouvelle pièce ; (g-h) autre ajout à l'objet. Le résultat final se trouve à droite. Images tirées de [Funkhouser *et al.* 2004].

FIG. 3.37 – Génération d'une chaise avec la méthode de Funkhouser *et al.* [2004]. Image tirée de [Funkhouser *et al.* 2004].

Funkhouser *et al.* [2004] ont une approche de composition par pièces d'objets représentés par des maillages polygonaux. Dans une séance typique de modélisation avec leur système, un usager sélectionne un premier modèle qu'il transformera pièce par pièce. L'usager sélectionne une pièce à remplacer ou une région où y sera *cousue* une nouvelle pièce. Il effectue ensuite une recherche d'une nouvelle pièce dans une base de données de modèles géométriques. Le résultat de la requête est un ensemble de modèles dont au moins une de leurs parties est similaire à la

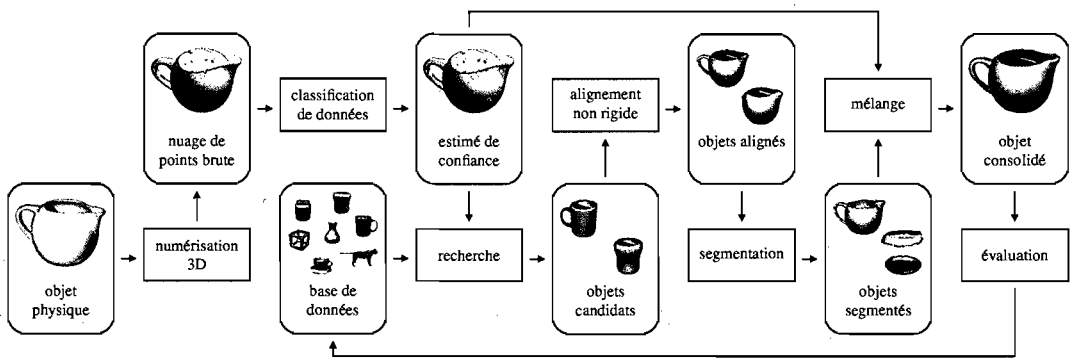


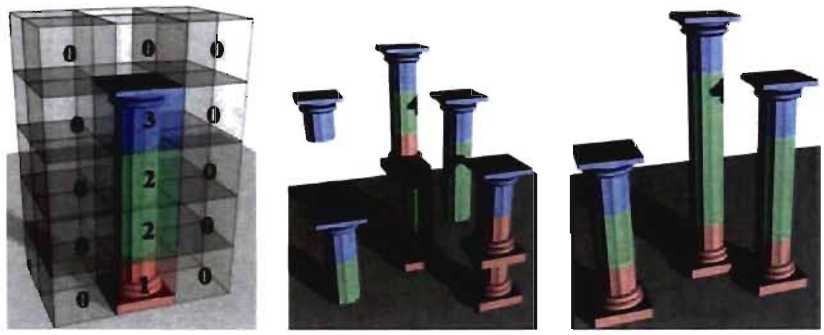
FIG. 3.38 – Survol de la procédure de complétion d'objet de Pauly *et al.* [2005b]. Les images sont adaptées de [Pauly *et al.* 2005b].

pièce désirée, et l'usager doit choisir un des modèles et découper la pièce. Cette dernière est cousue au modèle par un raccord (*fillet*) lissé. La figure 3.36 illustre un exemple d'une telle séance et la figure 3.37 montre un autre résultat. La recherche de pièces dans cette méthode est effectuée par requête sous forme d'objets. Les recherches d'objets se font par l'intermédiaire de champs de distance, de façon similaire aux méthodes de génération volumétrique (§3.2.1.3).

Pauly *et al.* [2005b] ont une approche similaire, mais axée sur la réparation d'objets numérisés incomplets, ce qui leur permet d'automatiser la procédure. La procédure générale de leur méthode est illustrée à la figure 3.38, que nous avons reproduite depuis la référence pour donner une intuition au lecteur. Nous n'en donnerons cependant pas les détails, puisque la partie d'intérêt pour nous dans cette méthode est essentiellement la même que la méthode de Funkhouser *et al.* [2004] décrite précédemment, *i.e.* la méthode de recherche d'objets. Nous invitons le lecteur à consulter la référence pour en savoir plus.

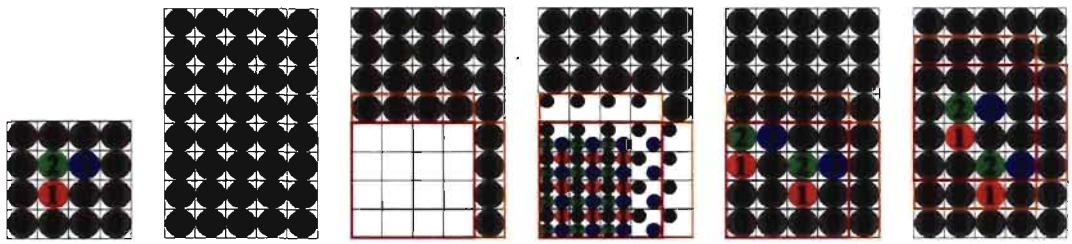
Une approche en quelque sorte intermédiaire entre celle de Funkhouser *et al.* [2004] et les méthodes de génération de textures est celle de Merrell [2007]. L'algorithme prend en entrée un objet (la source) décomposé en pièces selon une grille régulière, tel qu'illustré aux figures 3.39 (a) et 3.41 à gauche. L'idée est de remplir une autre grille avec un agencement des pièces de l'objet source de telle sorte à ce qu'il soit consistant avec l'agencement des pièces dans la source. Un agencement est consistant si l'adjacence de chaque pièce respecte celle trouvée dans l'objet source. Par exemple, la figure 3.39 (b) montre un agencement qui n'est pas consistant, alors qu'il l'est en (c).

Pour ce faire, l'algorithme, illustré en 2D à la figure 3.40, commence, en (b), par effectuer une initialisation simple de toute la grille. En (c), une région est ensuite sélectionnée et son



(a) (b) (c)

FIG. 3.39 – Consistance de la génération : (a) un objet composé de quatre pièces ; (b) génération inconsistante *vs.* (c) consistante. Les images proviennent de [Merrell 2007].



(a) (b) (c) (d) (e) (f)

FIG. 3.40 – Méthode de génération de Merrell [2007] : (a) décomposition en pièces de l'objet source ; (b) initialisation ; (c) sélection d'une région à générer ; (d) initialisation du jeu de pièces disponibles en fonction du voisinage de la région ; (e) génération complétée dans la région ; (f) le procédé est recommencé pour d'autres régions qui superposent celles qui ont déjà été générées. Les images proviennent de [Merrell 2007].

contenu est supprimé. Puis, en (d), l'ensemble de pièces possibles pour chaque élément de la grille dans la région est calculé en éliminant les choix non consistants avec les pièces voisines de la région. Le reste de la génération dans la région, en (e), procède en fixant au hasard un choix pour un des éléments de la grille et en mettant à jour les ensembles de choix des autres éléments pour maintenir la consistance ; il s'arrête lorsque tout élément est fixé. En (f), le tout est répété pour d'autres régions qui se superposent avec les régions déjà générées. Il est possible durant la génération d'une région qu'un ensemble de choix devienne vide. Dans un tel cas, la génération pour la région est soit recommencée ou une autre région est choisie.

La figure 3.41 montre un résultat de la méthode. En fait, les résultats généralement obtenus sont très intéressants compte tenu de la simplicité de la méthode. Cependant, la décomposition

FIG. 3.41 – Exemple de résultat de la méthode de Merrell [2007] : (gauche) objet source décomposé en pièces ; (droite) résultat de génération. Les images proviennent de [Merrell 2007].

en pièces par une grille régulière impose une restriction non négligeable sur le type d'objet pouvant être généré à cause de l'alignement forcé de toute pièce sur une grille.

Chapitre 4

Génération de textures géométriques définies par des ensembles de points

Tous les travaux qui ont été faits en matière de génération de textures et de géométrie indiquent qu'il s'agit d'une approche de modélisation puissante qui permet d'alléger le travail pour générer des détails géométriques sur de grandes surfaces en évitant les répétitions, qui sont facilement discernables par l'œil humain. Cependant, parmi tous ceux qui traitent de génération de textures géométriques, peu d'entre eux sont capables de convenablement générer des textures de tout type géométrique (les types sont décrits à la §3.2). Seules les approches volumétriques (§3.2.1.3) et la méthode de Zhou *et al.* [2006] (§3.2.3) ont pleinement cette capacité.

Le problème principal des approches volumétriques [Lagae *et al.* 2004, Lagae *et al.* 2005, Bhat *et al.* 2004b] est la quantité de mémoire qu'elles peuvent requérir pour atteindre un niveau suffisant de détails géométriques, et par conséquent, les temps de calcul sont généralement grands à cause du grand volume de données à traiter. Il semblerait alors qu'une approche traitant directement la géométrie des surfaces, comme celle de Zhou *et al.* [2006] (§3.2.3), soit plus indiquée pour être plus efficace. Cependant, quiconque a déjà eu à implémenter des opérations affectant la connectivité de maillages en connaît, complexité, les problèmes de robustesse la et la lourdeur de traitement.

D'un autre côté, les méthodes de Sharf [2004] et Park *et al.* [2005] (§3.2.3), bien que plus limitatives sur le type géométrique des textures générées, offrent une certaine efficacité sur le traitement de la géométrie par l'usage d'ensembles de points pour représenter les surfaces. Dans ce chapitre, nous proposons ainsi une méthode de génération de textures géométriques dans l'es-

prit de celle de Zhou *et al.* [2006] mais cherchant à profiter de l'efficacité d'une représentation par points. Notre méthode a été publiée à la conférence internationale *Vision, Modeling and Visualization 2006* [Duranleau et Poulin 2006].

Nous présentons d'abord les particularités de notation employées dans ce chapitre à la §4.1, puis nous donnons la définition de notre représentation d'une texture géométrique à la §4.2. Nous présentons l'algorithme de génération à la §4.3 et les étapes principales sont détaillées à la §4.4. Enfin, nous présentons des résultats obtenus avec notre méthode de génération à la §4.5 et nous discutons de la méthode et d'extensions possibles à la §4.6.

4.1 Notation

Nous notons par $\mathcal{T}$ une texture géométrique, par $\mathcal{B}_\mathcal{T}$ sa boîte englobante alignée sur les axes, et par $P_\mathcal{T}$ son ensemble de points. La définition exacte d'une texture géométrique se trouve à la prochaine section. Nous notons par $\mathcal{T}_\mathrm{e}$ la texture servant d'entrée à notre algorithme de génération, et $\mathcal{T}_\mathrm{s}$ la texture produite en sortie.

Nous définissons une région $\mathcal{R}$ d'une texture géométrique $\mathcal{T}$ comme étant une sous-texture de $\mathcal{T}$. Alors $P_\mathcal{R} \subset P_\mathcal{T}$ est l'ensemble de points inclus dans la région. Si la région est de forme *rectangulaire*, alors $\mathcal{B}_\mathcal{R}$ dénote sa boîte englobante alignée sur les axes de même que sa forme. $\mathcal{P}$ dénote une *patch*, qui est une région de $\mathcal{T}_\mathrm{e}$, et $\mathcal{E}$ dénote une région autour d'une *patch* dans $\mathcal{T}_\mathrm{e}$. $\mathcal{V}$ dénote une région de $\mathcal{T}_\mathrm{s}$ située à l'endroit de génération. $\mathcal{O}_\mathcal{R}$ est une sous-région de la région $\mathcal{R}$ marquant une zone de chevauchement. Plusieurs de ces concepts sont illustrés à la figure 4.1.

Nous utilisons aussi les concepts et la notation de la §2.3.

4.2 Représentation d'une texture géométrique

Pour représenter des textures géométriques de topologie arbitraire, nous avons besoin de plus que des cartes d'élévation (§3.2.1.1) ou des cartes de déplacement (§3.2.1.2). L'idée de notre représentation est apparentée aux cartes de déplacement généralisées [Wang *et al.* 2004], et est très similaire en concept aux textures de Zhou *et al.* [2006] (§3.2.3).

Nous définissons une texture géométrique $\mathcal{T}$ par les éléments suivants :

- une boîte englobante alignée sur les axes $\mathcal{B}_\mathcal{T}$ définissant les dimensions de la texture géométrique ;

- un ensemble de points $P_{\mathcal{T}}$ tel que la surface $\mathcal{S}_{P_{\mathcal{T}}}$ est une variété de topologie arbitraire résidant non seulement à l'intérieur de $\mathcal{B}_{\mathcal{T}}$, mais aussi au-delà des limites de $\mathcal{B}_{\mathcal{T}}$ pour une meilleure définition de la surface aux bordures ;
- un plan aligné sur les axes définissant l'orientation de la texture, ce qui permet aussi de définir la *hauteur* de la texture comme la dimension de $\mathcal{B}_{\mathcal{T}}$ dans la direction perpendiculaire au plan.

Les figures 4.5, 4.6, 4.8 et 4.9 montrent des exemples de textures géométriques (côté gauche des figures).

Remarque 4.1 (Plan de $\mathcal{T}_e$) *Sans perte de généralité, nous pouvons supposer que le plan est toujours le même. En effet, nous pourrions transformer la texture de sorte que son plan soit toujours confondu avec un plan de référence commun. Dans notre implémentation, le plan commun choisi est le plan XY, faisant de Z l'axe de hauteur. Ainsi, pour le reste de ce chapitre, nous pouvons ignorer ce paramètre de définition.*

Notre méthode de génération procède par *patch*. Soit $\mathcal{T}_e$ la texture géométrique originale (ou d'entrée) à partir de laquelle nous voulons en générer une autre, notée $\mathcal{T}_s$. Une *patch* $\mathcal{P} \subset \mathcal{T}_e$ est une texture géométrique telle que son ensemble des points $P_{\mathcal{P}} \subset P_{\mathcal{T}_e}$ se trouve à l'intérieur d'une boîte alignée sur les axes $\mathcal{B}_{\mathcal{P}}$, incluse dans $\mathcal{B}_{\mathcal{T}_e}$ et de même *hauteur*. De la même façon que les méthodes de génération de textures par *patch* (§3.1.3), la taille d'une *patch* devrait être choisie selon la taille et la distribution des caractéristiques de $\mathcal{T}_e$. Elle devrait être suffisamment grande pour cerner un élément de texture, par exemple une bosse dans la figure 4.5, ainsi qu'une part de voisinage. Ceci est apparenté avec la notion de *texton* en génération de textures[1]. Si la *patch* est trop petite, la forme et la distribution des éléments pourraient ne pas être bien préservées. Si elle est trop grande, il y aura moins de variation dans la texture générée.

4.3 Algorithme de génération

La procédure générale de notre algorithme de génération est similaire à celle de Efros et Freeman [2001]. Elle débute avec une *patch* germe et procède par accroissement en ordre par balayage de ligne. Puisque la génération est effectuée sur le plan de référence de la texture, il s'agit donc d'un procédé essentiellement 2D. Ainsi, quand nous parlons de région, nous faisons

[1] Voir la note en bas de page 9 à la page 59 pour une définition de *texton*.

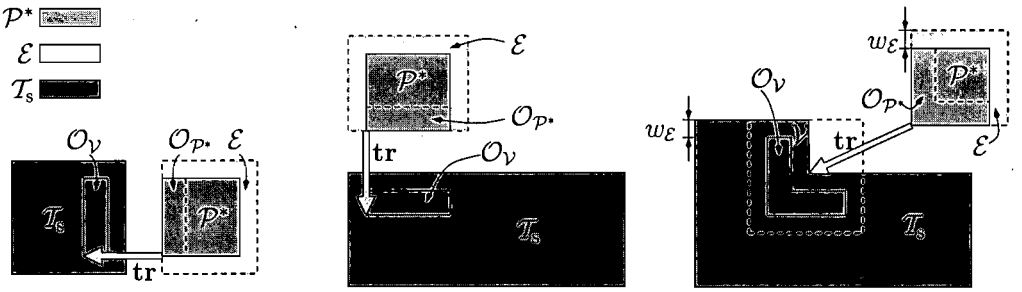


FIG. 4.1 – Procédé de génération : (a) nous débutons avec une *patch* initiale et procédons en ordre de balayage de ligne ; (b) une nouvelle *patch* commence une nouvelle ligne ; (c) le cas général. Le carré en pointillé au-dessus de $\mathcal{T}_s$ correspond à $\mathcal{B}_\mathcal{V}$.

en fait allusion au volume défini par son aire sur le plan de référence extrudée jusqu'à la pleine hauteur de la texture.

À une itération donnée, nous amassons un voisinage $\mathcal{V} \subset \mathcal{T}_s$ autour d'une région où une nouvelle *patch* sera placée pour accroître la texture de sortie $\mathcal{T}_s$. Nous sélectionnons ensuite un ensemble de *patches* telles que leur voisinage respectif dans $\mathcal{T}_e$ est similaire à $\mathcal{V}$ et nous en choisissons une au hasard. La *patch* choisie $\mathcal{P}^*$ est ensuite translatée à sa position de destination, puis transformée pour mieux correspondre avec $\mathcal{V}$, et finalement fusionnée avec $\mathcal{T}_s$. Pour des raisons expliquées plus tard (§4.4.1), la *patch* doit être augmentée d'une région périphérique $\mathcal{E}$ (*padding*) qui s'étend hors de $\mathcal{P}^*$ dans la direction générale de la génération. La figure 4.1 illustre les différentes régions utilisées.

Notre algorithme de génération procède donc comme suit :

1. Choisir au hasard ou manuellement une *patch* de $\mathcal{T}_e$ et la copier et coller dans le coin initial de $\mathcal{T}_s$, de même que sa région périphérique. La région périphérique de la première *patch* s'étend tout autour de la *patch*[2].

2. Déterminer l'emplacement de la prochaine *patch*. Notre algorithme fonctionne selon un ordre de balayage de ligne, comme les méthodes d'Efros et Freeman [2001] et Liang *et al.* [2001].

[2]En réalité, une partie de cette région périphérique n'est pas utile pour la génération, mais procéder ainsi simplifie l'implémentation.

3. Soit $\mathcal{V}$ le voisinage à l'endroit courant de génération, trouver un ensemble de *patches* candidates

$$\Psi = \{\mathcal{P} \subset \mathcal{T}_e \mid d(\mathcal{O}_\mathcal{V}, \mathcal{O}_\mathcal{P}) < d^+\} \tag{4.1}$$

où

$$d^+ = (1+\epsilon) \min_{\mathcal{P} \subset \mathcal{T}_e} d(\mathcal{O}_\mathcal{V}, \mathcal{O}_\mathcal{P}). \tag{4.2}$$

$\mathcal{O}_X$ désigne la région de chevauchement d'intérêt à l'itération courante de l'algorithme à l'intérieur de X (voir la figure 4.1), d est une métrique de comparaison des régions de chevauchement, et ϵ est l'erreur relative tolérée sur d (détails à la §4.4.1).

4. Choisir aléatoirement une *patch* $\mathcal{P}^* \in \Psi$. Soient $F = \mathbf{tr}(P_{\mathcal{P}^*} \cup P_\mathcal{E})$ et $\mathcal{B} = \mathbf{tr}(\mathcal{B}_{\mathcal{P}^*} \cup \mathcal{B}_\mathcal{E})$, où $\mathbf{tr}$ est une fonction qui effectue une translation jusqu'à la position destination.

5. Déformer l'ensemble de points F pour le rendre concordant avec $\mathcal{V}$; soit F' le résultat (détails à la §4.4.2).

6. Interpoler la géométrie de $\mathcal{V}$ et F' ; soit F'' le résultat (détails à la §4.4.3).

7. Supprimer tous les points de $P_{\mathcal{T}_s}$ se trouvant à l'intérieur de $\mathcal{B}$ et copier tous les points de F'' à l'intérieur de $\mathcal{B}$ dans $P_{\mathcal{T}_s}$.

8. Répéter les étapes 2 à 7 jusqu'à ce que $\mathcal{T}_s$ soit remplie.

9. Supprimer une région de largeur $w_\mathcal{E}$ (définie à la §4.4.1 et illustrée à la figure 4.1) le long de la frontière de $\mathcal{T}_s$.

La prochaine section donne des détails sur certaines étapes importantes de l'algorithme.

4.4 Traitement d'une *patch*

Plusieurs étapes dans l'algorithme précédent impliquent la recherche et l'ajustement d'une nouvelle *patch* pendant le procédé de génération. Dans cette section, nous fournissons les détails de trois étapes fondamentales : la recherche (§4.4.1), la déformation (§4.4.2) et l'interpolation (§4.4.3).

4.4.1 Recherche d'une *patch* similaire

L'étape 3 de notre algorithme crée un ensemble Ψ des *patches* formant les meilleures candidates, selon une métrique de similarité d. Puisqu'un grand nombre de *patches* doivent être évaluées, d doit être rapide à évaluer.

Zhou *et al.* [2006] calculent une fonction de coût basée sur la distance entre les sommets des maillages et leur projection sur le maillage de la surface comparée, et sur la différence de normales. Cependant, cette fonction est coûteuse à évaluer, particulièrement avec des ensembles de points, car ceci impliquerait le calcul de plusieurs projections pendant la comparaison.

Une métrique de similarité fréquemment rencontrée est la distance euclidienne ; elle est rapide à évaluer, mais elle requiert une correspondance entre nos ensembles de points et un espace euclidien. Park *et al.* [2005] et Zelinka et Garland [2006] requièrent une paramétrisation locale pour évaluer des vecteurs de caractéristiques, ce qui limite la topologie locale (seulement des composante connexes). Sharf *et al.* [2004] ajustent localement des fonctions implicites et prennent quelques mesures de la distance signée et du gradient de ces fonctions. Dans notre cas, nous avons besoin d'une densité de mesures plus grande pour permettre des géométries plus complexes dans nos *patches*. Nous avons alors opté pour utiliser des champs de distance [Lagae *et al.* 2005, Funkhouser *et al.* 2004] avec leur champ de gradient (*gradient field*). Ces champs sont calculés seulement à l'intérieur des régions de chevauchement.

Soit $\delta_{\mathcal{S}}(\boldsymbol{x}) : \mathbb{R}^3 \mapsto \mathbb{R}$ le champ de distance de la surface orientée $\mathcal{S}$, *i.e.*

$$\delta_{\mathcal{S}}(\boldsymbol{x}) = \boldsymbol{n}_{\mathcal{S}}(\arg\min_{\boldsymbol{y}\in\mathcal{S}} \|\boldsymbol{y}-\boldsymbol{x}\|)^{\mathsf{T}}(\boldsymbol{x} - \arg\min_{\boldsymbol{y}\in\mathcal{S}} \|\boldsymbol{y}-\boldsymbol{x}\|) \tag{4.3}$$

où $\boldsymbol{n}_{\mathcal{S}}(\boldsymbol{x})$ est la normale de la surface au point $\boldsymbol{x} \in \mathcal{S}$. Dans notre cas, la surface est de type MCM, définie par un ensemble de points P. Ainsi,

$$\delta_{\mathcal{S}_P}(\boldsymbol{x}) = \boldsymbol{n}_P(\psi_P(\boldsymbol{x}))^{\mathsf{T}}(\boldsymbol{x} - \psi_P(\boldsymbol{x})). \tag{4.4}$$

Cependant, cette définition est exacte seulement si $\boldsymbol{n}_P(\boldsymbol{x})$ estime la véritable normale de la surface $\mathcal{S}_P$ en $\boldsymbol{x}$ et si $\psi_P(\boldsymbol{x})$ est une projection orthogonale (voir §2.3.3.1). Dans notre cas, ce n'est pas tout à fait le cas (remarque 2.4), mais pour les fins d'évaluation de similarité entre *patches*, l'erreur induite est peu importante. Par contre, pour garantir la continuité du signe de la fonction de distance, l'orientation de $\boldsymbol{n}_P(\boldsymbol{x})$ doit être cohérente entre points voisins. Ainsi, nous requérons qu'une normale soit donnée explicitement pour chaque point de nos ensembles

de points. Ensuite, suivant les propriétés du gradient d'un champ de distance [Jones *et al.* 2006], nous pouvons calculer le gradient de $\delta_{S_P}(\boldsymbol{x})$ ainsi[3] :

$$\nabla\delta_{S_P}(\boldsymbol{x}) = \frac{\boldsymbol{x} - \boldsymbol{\psi}_P(\boldsymbol{x})}{\|\boldsymbol{x} - \boldsymbol{\psi}_P(\boldsymbol{x})\|}. \tag{4.5}$$

Pour établir notre métrique de comparaison entre les *patches* dans un espace euclidien, nous devons passer à l'état discret, ce que nous accomplissons en échantillonnant les champs de distance aux sommets d'une grille régulière. Cependant, puisque nous voulons seulement comparer les voisinages, il suffit d'effectuer cet échantillonnage seulement dans les régions de chevauchement $\mathcal{O}_\mathcal{V}$ et $\mathcal{O}_{\mathcal{P}^*}$. Les champs de distance à l'état discret sont donc calculés ainsi :

$$D_{\mathcal{O}_X} = \{\delta_{P_X}(\boldsymbol{o}_{\mathcal{O}_X} + h\boldsymbol{k}) \mid \boldsymbol{k} \in \mathbb{Z}^3 \wedge \boldsymbol{o}_{\mathcal{O}_X} + h\boldsymbol{k} \in \mathcal{O}_X\} \tag{4.6}$$

où X est $\mathcal{V}$ ou $\mathcal{P}^*$, $\boldsymbol{o}_{\mathcal{O}_X}$ est l'origine de la région $\mathcal{O}_X$ et h est la distance d'échantillonnage. Puisque $\mathcal{O}_\mathcal{V}$ et $\mathcal{O}_{\mathcal{P}^*}$ ont la même forme et les mêmes dimensions, nous sommes toujours garantis que $|D_{\mathcal{O}_\mathcal{V}}| = |D_{\mathcal{O}_{\mathcal{P}^*}}|$. En échantillonnant toujours dans le même ordre, nous pouvons convertir $D_{\mathcal{O}_\mathcal{V}}$ et $D_{\mathcal{O}_{\mathcal{P}^*}}$ en deux vecteurs respectifs $\boldsymbol{\delta}_{\mathcal{O}_\mathcal{V}}, \boldsymbol{\delta}_{\mathcal{O}_{\mathcal{P}^*}} \in \mathbb{R}^k$, où $k = |D_{\mathcal{O}_\mathcal{V}}| = |D_{\mathcal{O}_{\mathcal{P}^*}}|$. Le même raisonnement s'applique pour les gradients pour obtenir deux vecteurs $\nabla\boldsymbol{\delta}_{\mathcal{O}_\mathcal{V}}, \nabla\boldsymbol{\delta}_{\mathcal{O}_{\mathcal{P}^*}} \in \mathbb{R}^{3k}$. La dimension de ces derniers est trois fois plus grande car chaque élément du gradient discret a trois dimensions. Finalement, la métrique de comparaison n'est que la distance euclidienne (au carré) entre ces vecteurs, *i.e.*

$$d(\mathcal{O}_\mathcal{V}, \mathcal{O}_{\mathcal{P}^*}) = \|\boldsymbol{\delta}_{\mathcal{O}_\mathcal{V}} - \boldsymbol{\delta}_{\mathcal{O}_{\mathcal{P}^*}}\|^2 + \omega\|\nabla\boldsymbol{\delta}_{\mathcal{O}_\mathcal{V}} - \nabla\boldsymbol{\delta}_{\mathcal{O}_{\mathcal{P}^*}}\|^2 \tag{4.7}$$

où ω est un facteur de poids sur la différence de gradient dans la métrique.

Remarque 4.2 (Poids du gradient) *Pour toutes nos expérimentations, $\omega = 1$, car nous n'avons pas observé de changement significatif en faisant fluctuer la valeur de ω, sauf si $\omega \ll 1$, dans quel cas les résultats sont la plupart du temps moins bons.*

Le paramètre h, défini par l'usager, devrait être suffisamment petit pour saisir les détails géométriques, mais parce le champ de distance ne sert qu'à des fins de comparaison, et non de modélisation, la précision n'a pas besoin d'être très grande. De plus, nous avons seulement besoin de calculer $\boldsymbol{\delta}_\mathcal{V}$ au moment de la génération. Nous pouvons précalculer un champ de

[3] En réalité, il s'agit plutôt d'une estimation si $\boldsymbol{\psi}_P(\boldsymbol{x})$ n'est pas exactement orthogonale.

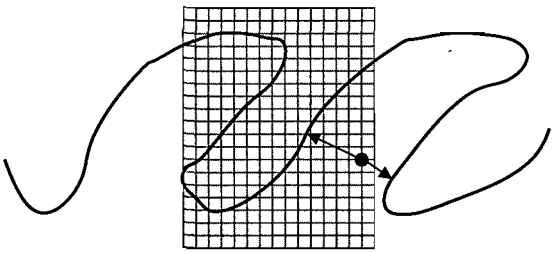


FIG. 4.2 – Erreur sur le calcul de distance : la longue flèche indique la distance calculée à partir de la géométrie uniquement à l'intérieur de la boîte, alors que la flèche plus courte indique la vraie distance à la géométrie.

distance discret pour tout $\mathcal{T}_e$ avec la même distance d'échantillonnage h, et alors $\delta_{\mathcal{P}^*}$ peut être extrait du champ de distance discret de $\mathcal{T}_e$. Ainsi, h devient aussi un paramètre pour découper $\mathcal{T}_e$ en un ensemble discret de *patches*, définissant l'espace de recherche de candidats.

Il faut faire attention pour bien calculer la distance et le gradient près de la bordure de la région d'échantillonnage. Seulement considérer la géométrie tombant à l'intérieur de la région peut donner des fausses distances, tel qu'illustré à la figure 4.2. C'est donc pourquoi, tel que décrit à la §4.2 et tel qu'illustré à la figure 4.1, la géométrie de $\mathcal{T}_e$ doit s'étendre au-delà de $\mathcal{B}_{\mathcal{T}_e}$. C'est pourquoi aussi $\mathcal{B}_\mathcal{V}$ englobe $\mathcal{O}_\mathcal{V}$ sans y toucher, et qu'une région périphérique $\mathcal{E}$ doit accompagner $\mathcal{P}^*$. Cette région périphérique fournit une bande de géométrie autour de $\mathcal{O}_\mathcal{V}$. La largeur de cette bande est identifiée par $w_\mathcal{E}$ dans la figure 4.1 (c). La valeur de $w_\mathcal{E}$ est choisie par l'usager en fonction du type et de la densité des caractéristiques de $\mathcal{T}_e$. Dans nos exemples de la §4.5, elle varie entre $\frac{1}{4}$ et $\frac{1}{3}$ de la taille d'une *patch*. Les proportions plus grandes ont tendance à seulement ralentir l'exécution sans améliorer les résultats.

Une métrique sur un espace euclidien offre aussi la possibilité d'utiliser des structures d'accélération de recherche. Deux approches populaires et généralement efficaces sont les recherches de voisins approximatives (RVA) [Arya *et al.* 1998] et les quantifications de vecteurs structurées par arborescence (*TSVQ*) [Gersho et Gray 1991], toutes les deux ayant été utilisées avec succès en génération de textures [Liang *et al.* 2001, Wei et Levoy 2000, Hertzmann *et al.* 2001, Bhat *et al.* 2004b]. Cependant, la haute dimensionalité de nos espaces de recherche fait en sorte que les RVA deviennent à peu près équivalentes à des recherches exhaustives. Nous pourrions faire usage de *TSVQ* en sacrifiant qualité pour rapidité, mais nous avons trouvé que même en utilisant une recherche exhaustive, l'étape de la recherche n'est pas un goulot d'étranglement de notre algorithme de génération (elle prend environ 20% à 30% du temps total selon

nos résultats du tableau 4.1 à la page 111). Nous pouvons rendre la recherche un peu plus rapide avec l'extraction de tous les $\delta_{\mathcal{P}}$, $\mathcal{P} \subset \mathcal{T}_e$, dans une étape de précalcul. Ceci consomme beaucoup plus de mémoire, mais dans tous nos exemples, la consommation excède rarement 100 à 200 mégaoctets, en utilisant des nombres en virgule flottante de simple précision.

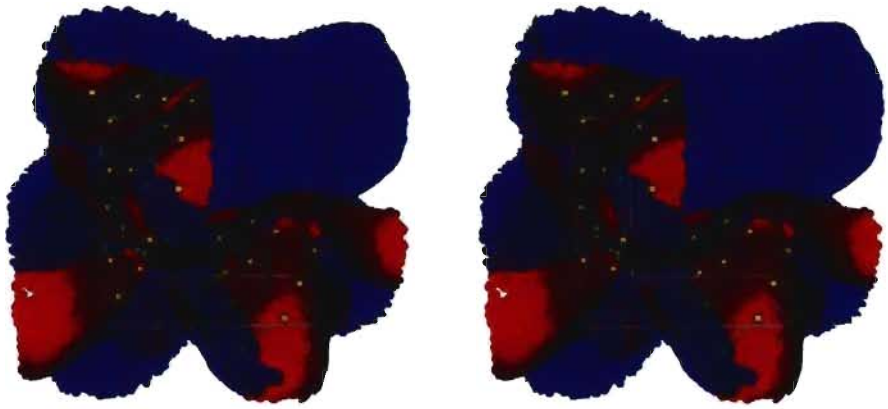
4.4.2 Déformation

Après l'étape 4 de notre algorithme, la *patch* $\mathcal{P}^*$ choisie ne s'aligne généralement pas parfaitement avec $\mathcal{V}$. Dans le cas de la génération de textures colorées, ceci est corrigé soit par l'application d'un mélange [Liang *et al.* 2001], soit en cherchant une coupe optimale [Efros et Freeman 2001, Kwatra *et al.* 2003], ou soit en utilisant les deux méthodes. Par contre, mélanger de la géométrie n'est pas aussi simple que mélanger des couleurs.

Sharf *et al.* [2004] (§3.2.3) utilisent une méthode similaire à l'ICP [Besl et McKay 1992, Chen et Medioni 1992] avec l'ajout d'une déformation (*warping*) non rigide quadratique. Nous avons implémenté et essayé ce type de déformation dans notre algorithme, mais ce n'était pas assez puissant pour nos besoins. Wu et Yu [2004] (§3.1.4) utilisent une correspondance de caractéristiques pour déformer une *patch* avec une interpolation par spline en plaques minces (*thin-plate spline*) [Wahba 1990]. Nous avons opté pour une approche similaire. L'usage de splines en plaques minces implique de trouver un ensemble de paires de points correspondants à partir de surfaces source et cible. Wu et Yu [2004] trouvent ces paires automatiquement en se basant sur la proximité et la tangente des points caractéristiques. Cependant, c'est plus compliqué lorsqu'il est question de géométrie. En particulier, nous devons faire attention pour éviter que les paires voisines ne se croisent pas pour éviter que la surface se retourne sur elle-même localement. La figure 4.3 (a) montre un exemple de pairage problématique.

Soient S l'ensemble de points de la surface source et Q l'ensemble de points de la surface cible, où $|Q| < |S|$. Soient Θ un ensemble de paires de points correspondants, ou pairage, et $W_\Theta : \mathbb{R}^3 \mapsto \mathbb{R}^3$ une fonction de déformation par spline en plaques minces suivant les contraintes données par le pairage Θ. Nous cherchons un pairage $\Theta^* \subset S \times Q$ qui minimise une fonction de différence entre les surfaces $\mathcal{S}_{W_{\Theta^*}(S)}$ et $\mathcal{S}_Q$.

Il se trouve que le problème tel que posé est difficile à résoudre efficacement. Déjà, au lieu d'évaluer une fonction de différence entre deux surfaces, nous pouvons évaluer plutôt une



(a) (b)

FIG. 4.3 – Pairages avec et sans croisements, où la surface rouge opaque est $P_{\mathcal{V}}$ (la destination Q) et celle bleue semi-transparente est F (la source S). (a) La région encerclée en vert montre des paires se croisant ou dans des directions non cohérentes peuvent mener à des repliements ou de grandes distorsions locales indésirables. (b) Un pairage plus cohérent résulte de l'algorithme de sélection vorace.

fonction de coût d'un pairage $c : \wp(S \times Q) \mapsto \mathbb{R}$ comme ceci :

$$c(\Theta) = \sum_{(\boldsymbol{s},\boldsymbol{q})\in\Theta} (p(\boldsymbol{s},\boldsymbol{q}) + g(\boldsymbol{s},\boldsymbol{q},\Theta)) \tag{4.8}$$

où $p : S \times Q \mapsto \mathbb{R}$ est une fonction de coût évaluant la qualité d'une paire donnée $(\boldsymbol{s},\boldsymbol{q}) \in S \times Q$, et $g : S \times Q \times \wp(S \times Q) \mapsto \mathbb{R}$ est une autre fonction[4] de coût qui évalue la cohérence de $(\boldsymbol{s},\boldsymbol{q})$ avec les paires voisines dans Θ. Autrement dit, il s'agit d'un terme de lissage entre paires voisines. Le choix de cette fonction de coût c s'appuie sur l'hypothèse que dans le contexte d'application de W_Θ, les surfaces source $\mathcal{S}_S$ et destination $\mathcal{S}_Q$ devraient déjà être semblables, et donc la distance entre les points de chaque paire ne devrait pas être arbitraire, la surface autour de chaque point devrait être similaire, et un certain niveau de cohérence devrait exister entre les paires voisines.

Nous définissons p comme une fonction de la distance entre deux points, la différence entre leur normale et la différence de leur variation de surface locale [Pauly *et al.* 2002a] (§2.3.1.2). L'importance de chaque aspect de p peut varier d'un type de texture géométrique à l'autre, donc

[4]Pour être entièrement cohérent avec le reste de notre notation, nous aurions dû plutôt écrire $g_\Theta(\boldsymbol{s},\boldsymbol{q})$ au lieu de $g(\boldsymbol{s},\boldsymbol{q},\Theta)$. Cependant, ceci poserait un problème de lourdeur d'indice pour les fonctions f_i et f_d des équations (4.11) à (4.14). Nous avons alors opté de placer Θ en paramètre supplémentaire à ces fonctions.

ils sont pondérés par des valeurs déterminées par l'usager comme ceci :

$$p(\boldsymbol{s},\boldsymbol{q}) = \omega_{\mathrm{p}}\|\boldsymbol{s}-\boldsymbol{q}\| + \frac{\omega_{\mathrm{n}}}{2}\big(1-\boldsymbol{n}_S(\boldsymbol{s})^{\mathsf{T}}\boldsymbol{n}_Q(\boldsymbol{q})\big) + \omega_\sigma\big(\sigma_S(\boldsymbol{s})-\sigma_Q(\boldsymbol{q})\big)^2 \tag{4.9}$$

où ω_{p}, ω_{n} et ω_σ sont les poids respectifs associés aux différences de position, normale et variation. Puisque nous avons l'information explicite de la normale associée à chaque point, $\boldsymbol{n}_S(\boldsymbol{s})$ et $\boldsymbol{n}_Q(\boldsymbol{q})$ correspondent alors à cette information plutôt qu'au calcul d'estimation de normale.

Remarque 4.3 (Précalcul de $\sigma_S(\boldsymbol{s})$) *Dans notre cas, comme le sera expliqué plus loin, la source correspond à une* patch *de $\mathcal{T}_{\mathrm{e}}$, qui ne change pas durant la génération. Ainsi, nous pouvons précalculer $\sigma_{P_{\mathcal{T}_{\mathrm{e}}}}(\boldsymbol{x}), \forall \boldsymbol{x} \in P_{\mathcal{T}_{\mathrm{e}}}$ et éviter le calcul de $\sigma_S(\boldsymbol{s})$ dans l'équation* (4.9)*, de la même façon que les normales.*

Afin d'écarter les mauvais pairages, tels un pairage contenant une paire de points avec des normales opposées, un seuil est imposé pour chaque terme de l'équation (4.9), et s'il n'est pas respecté, alors nous fixons $p(\boldsymbol{s},\boldsymbol{q}) = \infty$. La définition complète de p est donc

$$p(\boldsymbol{s},\boldsymbol{q}) = \begin{cases} \omega_{\mathrm{p}}\|\boldsymbol{s}-\boldsymbol{q}\| & \text{si } \|\boldsymbol{s}-\boldsymbol{q}\| \le \tau_{\mathrm{p}} \\ \quad +\frac{\omega_{\mathrm{n}}}{2}\big(1-\boldsymbol{n}_S(\boldsymbol{s})^{\mathsf{T}}\boldsymbol{n}_Q(\boldsymbol{q})\big) & \quad \wedge\, \boldsymbol{n}_S(\boldsymbol{s})^{\mathsf{T}}\boldsymbol{n}_Q(\boldsymbol{q}) \ge \tau_{\mathrm{n}} \\ \quad +\omega_\sigma\big(\sigma_S(\boldsymbol{s})-\sigma_Q(\boldsymbol{q})\big)^2 & \quad \wedge\, \big(\sigma_S(\boldsymbol{s})-\sigma_Q(\boldsymbol{q})\big)^2 \le \tau_\sigma \\ \infty & \text{sinon} \end{cases} \tag{4.10}$$

où τ_p, τ_n et τ_σ sont les seuils respectifs pour la position, normale et variation.

La fonction g de l'équation (4.8) évalue la différence en longueur et direction entre une paire $(\boldsymbol{s},\boldsymbol{q})$ et ses voisines dans Θ. La longueur d'une paire $(\boldsymbol{s},\boldsymbol{q})$ est la distance entre $\boldsymbol{s}$ et $\boldsymbol{q}$, et sa direction est le vecteur unitaire de $\boldsymbol{q}$ vers $\boldsymbol{s}$. La différence d'un aspect d'une paire et de ses voisines est en fait calculée avec celui de la paire et une moyenne pondérée de l'aspect des voisines en fonction de la distance à la paire. Chaque aspect est aussi pondéré de la même façon que ceux de la fonction p. Voici la définition exacte de la fonction g :

$$g(\boldsymbol{s},\boldsymbol{q},\Theta) = \omega_{\mathrm{l}}\big|\|\boldsymbol{s}-\boldsymbol{q}\| - f_{\mathrm{l}}(\boldsymbol{q},\Theta)\big| + \frac{\omega_{\mathrm{d}}}{2}\left(1 - f_{\mathrm{d}}(\boldsymbol{q},\Theta)^{\mathsf{T}}\frac{\boldsymbol{s}-\boldsymbol{q}}{\|\boldsymbol{s}-\boldsymbol{q}\|}\right) \tag{4.11}$$

où

$$f_{\mathrm{l}}(\boldsymbol{q},\Theta) = \frac{\sum_{(\boldsymbol{t},\boldsymbol{r})\in\Theta} \|\boldsymbol{t}-\boldsymbol{r}\|\theta(\|\boldsymbol{r}-\boldsymbol{q}\|)}{\sum_{(\boldsymbol{t},\boldsymbol{r})\in\Theta}\theta(\|\boldsymbol{r}-\boldsymbol{q}\|)} \tag{4.12}$$

$$f_{\mathrm{d}}(\boldsymbol{q},\Theta) = \frac{\sum_{(\boldsymbol{t},\boldsymbol{r})\in\Theta}(\boldsymbol{t}-\boldsymbol{r})\theta(\|\boldsymbol{r}-\boldsymbol{q}\|)}{\left\|\sum_{(\boldsymbol{t},\boldsymbol{r})\in\Theta}(\boldsymbol{t}-\boldsymbol{r})\theta(\|\boldsymbol{r}-\boldsymbol{q}\|)\right\|}. \tag{4.13}$$

Les fonctions f_{l} et f_{d} correspondent respectivement à la moyenne pondérée des longueurs et directions des paires voisines de $(\boldsymbol{s},\boldsymbol{q})$. Une paire $(\boldsymbol{t},\boldsymbol{r}) \in S \times Q$ est considérée comme voisine de $(\boldsymbol{s},\boldsymbol{q})$ si la distance entre $\boldsymbol{r}$ et $\boldsymbol{q}$ est inférieure à un seuil donné. Nous fixons ce seuil à $\sqrt{\tau_{\mathrm{p}}}$, puisque τ_{p} est le seuil de la distance au carré entre les deux points d'une paire (équation (4.10)). La pondération pour f_{l} et f_{d} se fait selon une fonction monotone décroissante de la distance entre $\boldsymbol{r}$ et $\boldsymbol{q}$. Nous utilisons une gaussienne telle que définie à l'équation (2.41) avec $h = \sqrt{\tau_{\mathrm{p}}}$. De plus, de la même façon que nous seuillons les différents aspects de la fonction p dans l'équation (4.10), nous seuillons les aspects de la fonction g pour aussi éviter les pairages avec des croisements évidents ainsi :

$$g(\boldsymbol{s},\boldsymbol{q},\Theta) = \begin{cases} \omega_{\mathrm{l}}\big|\|\boldsymbol{s}-\boldsymbol{q}\| - f_{\mathrm{l}}(\boldsymbol{q},\Theta)\big| & \text{si } \|\boldsymbol{s}-\boldsymbol{q}\| < (1+\tau_{\mathrm{l}}) f_{\mathrm{l}}(\boldsymbol{q},\Theta) \\ \quad + \frac{\omega_{\mathrm{d}}}{2}\left(1 - f_{\mathrm{d}}(\boldsymbol{q},\Theta)^{\mathsf{T}}\frac{\boldsymbol{s}-\boldsymbol{q}}{\|\boldsymbol{s}-\boldsymbol{q}\|}\right) & \quad \wedge f_{\mathrm{l}}(\boldsymbol{q},\Theta) < (1+\tau_{\mathrm{l}})\|\boldsymbol{s}-\boldsymbol{q}\| \\ & \quad \wedge f_{\mathrm{d}}(\boldsymbol{q},\Theta)^{\mathsf{T}}\frac{\boldsymbol{s}-\boldsymbol{q}}{\|\boldsymbol{s}-\boldsymbol{q}\|} \geq \tau_{\mathrm{d}} \\ \infty & \text{sinon} \end{cases} \tag{4.14}$$

où τ_{l} est le seuil de différence relative entre les longueurs et τ_{d} est le seuil sur la différence des directions. Toute pondération et tout seuillage se font par rapport à des distances des points de Q plutôt que de S simplement parce que $|Q| \leq |S|$, ce qui fait en sorte qu'il est plus efficace et pratique de construire les pairages en faisant correspondre des éléments de S à des éléments de Q.

Ensuite, nous définissons l'ensemble de tous les pairages possibles ainsi :

$$\mho = \{\Theta \subset \wp(S \times Q) \mid |\Theta| = |Q| \wedge c(\Theta) < \infty\}. \tag{4.15}$$

Alors le pairage Θ^* que nous cherchons est

$$\Theta^* = \arg\min_{\Theta\in\mho} c(\Theta). \tag{4.16}$$

Cependant, l'équation (4.16) est un problème difficile à résoudre sans lancer une recherche exhaustive. Nous approximons alors la solution par l'algorithme vorace suivante :

1. $M = \emptyset, \Theta = \emptyset$.
2. $\forall \boldsymbol{q} \in Q$:
   (a) $\boldsymbol{s} = \arg\min_{\boldsymbol{t} \in \mathcal{N}_S^{\sqrt{\tau_p}}(\boldsymbol{q}) \setminus M} p(\boldsymbol{t}, \boldsymbol{q})$.
   (b) $M = M \cup \{\boldsymbol{s}\}, \Theta = \Theta \cup \{(\boldsymbol{s}, \boldsymbol{q})\}$.
3. $M = \emptyset, \Theta^* = \emptyset$.
4. $\forall (\boldsymbol{s}, \boldsymbol{q}) \in \Theta$:
   (a) $\boldsymbol{s} = \arg\min_{\boldsymbol{t} \in \mathcal{N}_S^{\sqrt{\tau_p}}(\boldsymbol{q}) \setminus M} \big(p(\boldsymbol{t}, \boldsymbol{q}) + g(\boldsymbol{t}, \boldsymbol{q}, \Theta)\big)$.
   (b) $M = M \cup \{\boldsymbol{s}\}, \Theta^* = \Theta^* \cup \{\boldsymbol{s}, \boldsymbol{q}\}$.
5. $\Theta = \Theta^*$.
6. Répéter les étapes 3 à 5 un certain nombre de fois.

Le domaine de sélection de $\boldsymbol{s}$ aux étapes 2a et 4a est limité au voisinage euclidien de rayon $\sqrt{\tau_p}$ autour de $\boldsymbol{q}$, sans les points déjà sélectionnés, à cause du seuil τ_p sur la distance au carré entre $\boldsymbol{s}$ et $\boldsymbol{q}$. Quand l'algorithme se termine, le dernier état de Θ^* est la solution. Le pairage Θ^* est ensuite utilisé pour résoudre le système d'équations linéaires de la spline en plaques minces [Wahba 1990] pour obtenir notre fonction de déformation W_{Θ^*}. La figure 4.3 (b) montre un exemple de résultat de notre algorithme à partir de l'initialisation en (a).

La fonction W_{Θ^*} transforme essentiellement seulement les positions, pas les normales. Nous pourrions simplement réestimer les normales suite à la déformation (§2.3.1.2 et équation (2.20) à la §2.3.3.1), mais nous avons trouvé la technique suivante plus robuste, quoique plus coûteuse à évaluer. Soient $\boldsymbol{s} \in S$, $\boldsymbol{s}' = W_{\Theta^*}(\boldsymbol{s})$, $\boldsymbol{n_s}$ la normale associée à $\boldsymbol{s}$, et $\boldsymbol{u_s}$ et $\boldsymbol{v_s}$ deux vecteurs tels que $\{\boldsymbol{u_s}, \boldsymbol{v_s}, \boldsymbol{n_s}\}$ est une base orthonormale de $\mathbb{R}^3$. Alors la normale de $\boldsymbol{s}'$ peut être calculée ainsi :

$$\boldsymbol{n_{s'}} = \frac{\boldsymbol{J u_s} \times \boldsymbol{J v_s}}{\|\boldsymbol{J u_s} \times \boldsymbol{J v_s}\|} \tag{4.17}$$

où $\boldsymbol{J}$ est la matrice jacobienne de la fonction W_{Θ^*}. Il est à noter que $\boldsymbol{J}$ peut être analytiquement calculée.

Pour faire correspondre toutes ces équations avec la terminologie des sections précédentes, nous avons $Q \subset P_V$ et $P_F \subset S$. D'abord, Q ne peut pas être le plein voisinage P_V pour deux raisons :

1. les points près de la frontière de $\mathcal{V}$ sont moins fiables, en particulier pour l'estimation de la variation de surface, et

2. considérer tous les points mènerait à des systèmes d'équations linéaires de grande taille et coûteux à résoudre, et rendrait aussi plus difficile l'élimination de paires qui se croisent.

Nous avons trouvé que de simplement utiliser un algorithme adaptatif d'agrégation [Pauly *et al.* 2002a] (des détails sur cet algorithme se trouvent à la §6.3.1) dans une sous-région de $\mathcal{V}$ donne un ensemble Q raisonnable. De plus, parce que les déformations par splines en plaques minces ont un effet global, W_{Θ^*} pourrait induire des déformations trop fortes en dehors de la région de chevauchement dans F. Pour contrer ceci, nous ajoutons à Θ^* quelques paires supplémentaires de points correspondant à eux-mêmes dans les coins de la région de F. Pour la même raison, nous devons étendre F avant la déformation pour assurer que la région à l'intérieur de $\mathcal{B}$ soit pleinement couverte par $W_{\Theta^*}(F)$. Finalement, nous avons $F \subset S$ car certains points près de la frontière de Q pourraient avoir une meilleure correspondance si nous étendons F.

4.4.3 Interpolation

Parce que la déformation de la section précédente n'est pas parfaite pour aligner la *patch* $\mathcal{P}^*$ avec $\mathcal{V}$, des écarts d'alignement peuvent persister (voir la figure 4.4 (a) et (b)). Cependant, comme ces écarts devraient être petits, nous pouvons les enlever par une simple interpolation linéaire progressive de F' à $\mathcal{V}$. L'interpolation est en fait effectuée entre les points (avec leur normale) de F' et leur projection MCM (§2.3.3.1) sur $\mathcal{V}$. Le paramètre d'interpolation varie graduellement tel qu'illustré à la figure 4.4 (c).

Un petit problème survient avec la projection MCM si l'écart se produit au-dessus d'une vallée car il est plus probable que la projection tombe sur l'un des côtés de la vallée, créant ainsi un trou dans l'échantillonnage du résultat final. Nous pouvons améliorer la situation en utilisant une forme simplifiée d'intersection de rayons avec une surface MCM [Adamson et Alexa 2003a][5]. Nous testons l'intersection d'un rayon émanant dans la direction de la normale au point et un autre rayon dans la direction opposée. L'intersection la plus près est utilisée pour l'interpolation. Si aucune intersection n'est trouvée, nous utilisons alors la projection MCM

[5] Au moment de l'écriture de notre article [Duranleau et Poulin 2006], nous avons fait usage de la méthode d'intersection d'Adamson et Alexa [2003a] sans la traversée hiérarchique. Ce n'est que durant les travaux suivants (chapitre 6) que nous avons bien pris connaissance de la méthode d'intersection d'Adamson et Alexa [2004], quelque peu cachée dans leur article. Cette dernière méthode est plus simple et semble plus robuste. Quelques détails sur cette méthode se trouvent à la §6.4.

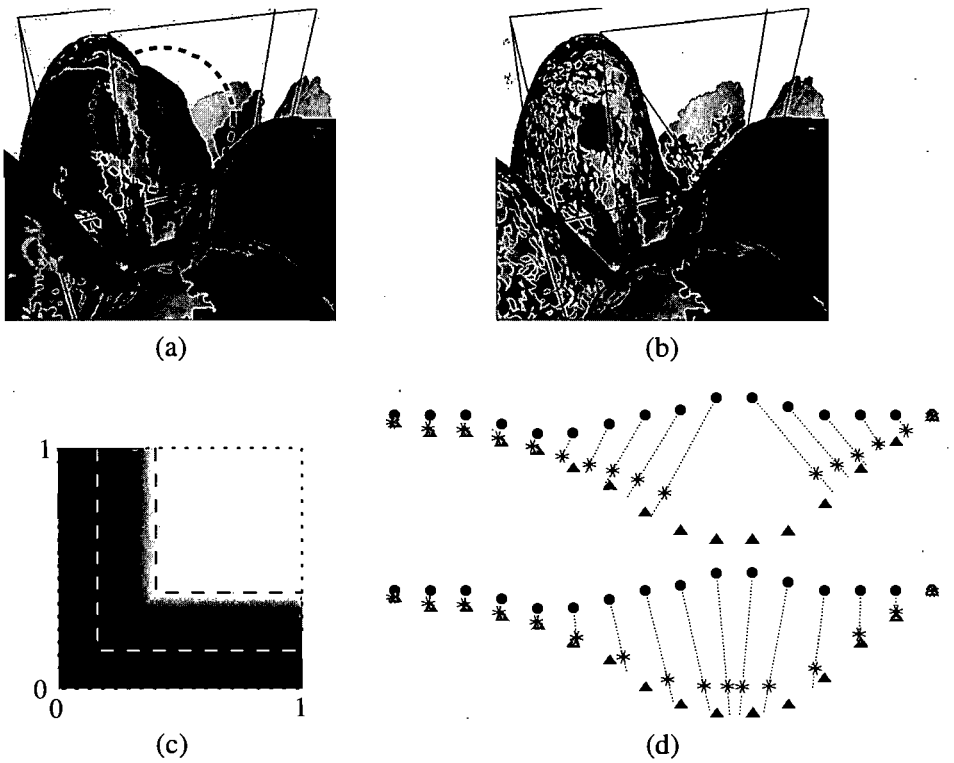


FIG. 4.4 – Interpolation. (a) $\mathcal{V}$ (gris clair) superposé avec F' (gris foncé). La bosse au centre montre une imperfection de l'alignement. (b) Le résultat de l'interpolation de F' et $\mathcal{V}$. (c) Valeur du paramètre d'interpolation comme fonction de la position à l'intérieur de la région de $\mathcal{V}$. (d) Différence entre l'interpolation avec projection (haut) et par intersection de rayons (bas) au-dessus d'une vallée.

régulière. La figure 4.4 (d) illustre un exemple de l'effet d'utiliser l'intersection de rayons plutôt que la projection MCM.

4.5 Résultats

Notre algorithme de génération de textures géométriques hérite des avantages mais aussi des inconvénients à la fois de la génération de textures par *patch* et des surfaces représentées par points. Dans cette section, nous présentons des résultats obtenus et nous discutons de leurs particularités.

Le premier exemple provient d'une texture colorée fréquemment utilisée en génération de textures. La texture a été convertie en carte d'élévation. La figure 4.5 montre, en haut à gauche,

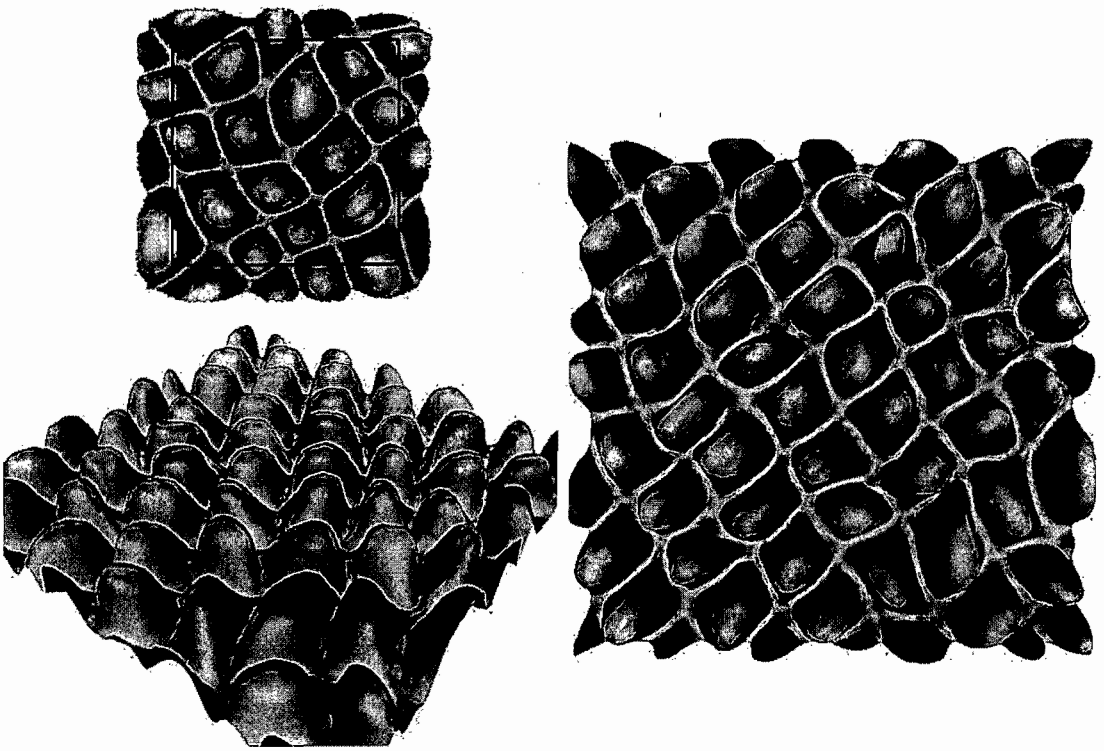


FIG. 4.5 – Carte d'élévation originale (en haut à gauche) et générée (deux vues).

la carte d'élévation originale avec la boîte englobante en fil de fer, et deux vues de la carte d'élévation générée à quatre fois la taille de l'originale.

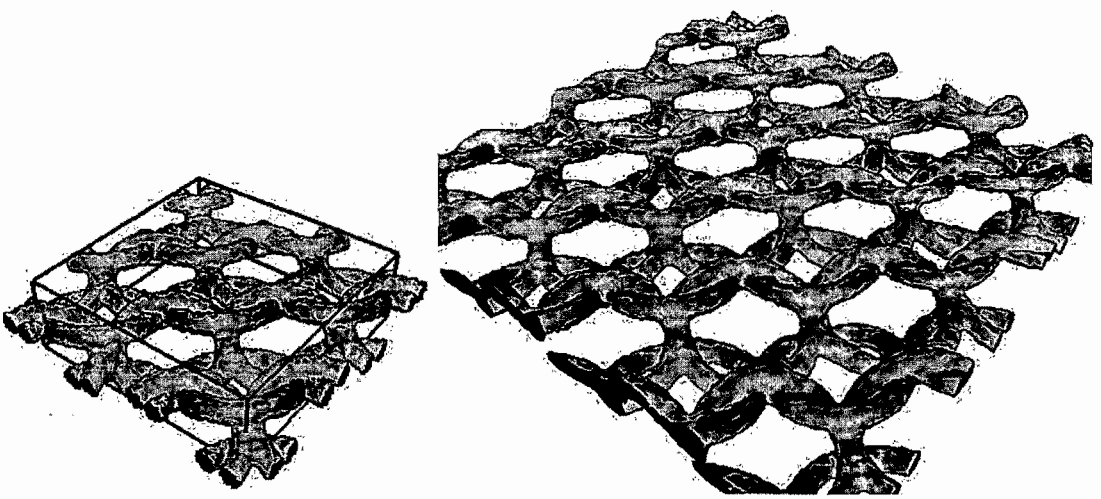


FIG. 4.6 – Cotte de mailles originale et générée.

La figure 4.6 illustre la flexibilité de l'algorithme de génération pour des topologies plus complexes de même qu'en présence de détails de surface plus fins. La présence de détails plus

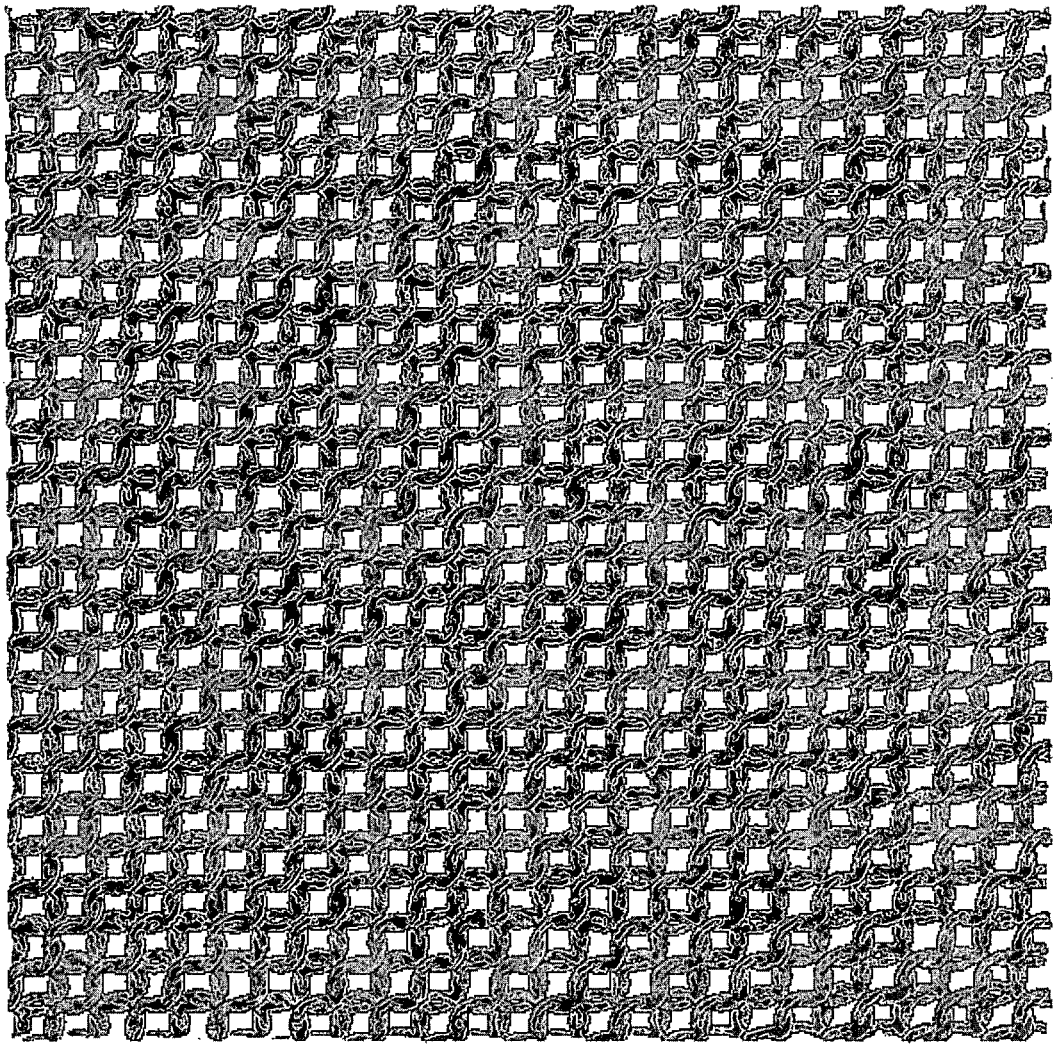


FIG. 4.7 – Génération d'une cotte de mailles plus grande.

fins pouvait possiblement causer des difficultés à l'aspect de similitude des normales pour la fonction de coût p de l'équation (4.10). L'approche par *patch* donne aussi de bons résultats pour des générations plus grandes, comme à la figure 4.7.

La figure 4.8 montre un exemple d'un treillis avec des composantes non connexes, où chaque fil demeure bien connecté dans le résultat. Il s'agit de l'exemple qui nous a fait considérer l'ajout du gradient dans la métrique de comparaison (§4.4.1).

La figure 4.9 montre un exemple avec des composantes, des fleurs, encore plus non connexes. La distribution générale est bien préservée dans la génération, et la plupart de fleurs individuelles sont adéquatement reconstruites.

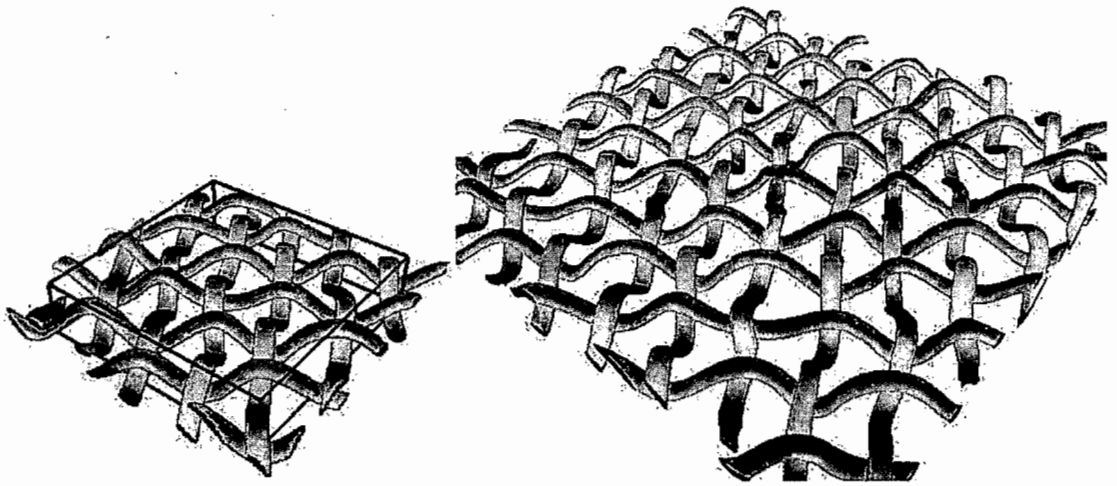


FIG. 4.8 – Treillis original et généré.

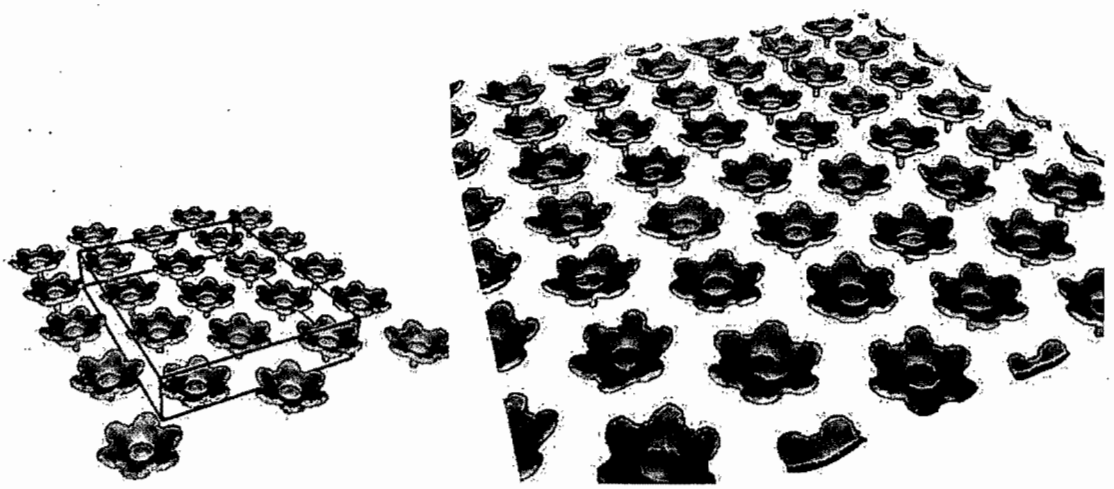


FIG. 4.9 – Fleurs originales et générées.

Remarque 4.4 (Post-traitement) *Dans tous nos exemples, nous avons effectué une relaxation sur les ensembles de points en traitement après la génération pour améliorer l'uniformité de la densité locale.*

Nous donnons au tableau 4.1 la valeur de chaque poids utilisé dans l'évaluation des fonctions de coût de la §4.4.2. Pour tous nos exemples, nous avons fixé $\omega_p = 1$ et ajusté les autres poids en conséquence. Les dimensions de $\mathcal{B}$ sont à peu près les mêmes pour tous, *i.e.* environ de taille unitaire dans l'axe du plan de génération. Nous observons que plus une caractéristique de la texture est grande par rapport à la densité d'échantillonnage (carte d'élévation), plus les termes de variation de surface et de normale sont importants dans l'équation (4.10), alors que

	Nb. points		Poids				Temps de génération (secondes)				
	$\mathcal{T}_e$	$\mathcal{T}_s$	ω_n	ω_σ	ω_l	ω_d	Rech.	Déf.	Inter.	Autre	Total
Carte d'élévation	33194	87101	3	30	9	7	6.7	11.2	10.8	2.0	30.7
Cotte de mailles	71454	197282	1	10	8	6	8.5	7.3	18.4	3.8	38.0
Grande cotte de mailles	71454	2947085	1	10	8	6	147.5	201.3	316.2	45.8	710.8
Treillis	43592	126139	2	15	6	5	5.7	7.7	12.0	2.6	28.0
Fleurs	37473	68290	2	2	6	6	11.0	14.9	7.4	2.7	36.0

TAB. 4.1 – Paramètres de l'usager et temps de calcul (Rech. : recherche ; Déf. : déformation ; Inter. : interpolation). Le nombre de points pour $\mathcal{T}_e$ correspond au nombre total de points, incluant la géométrie périphérique (voir la §4.4.1). En moyenne, environ seulement 40% des points de $P_{\mathcal{T}_e}$ tombent à l'intérieur de $\mathcal{B}_{\mathcal{T}_e}$.

des caractéristique plus petites requièrent des valeurs de poids plus faibles, mais un lissage des paires voisines plus fort (cotte de mailles *vs.* treillis). La variation de surface est moins importante en présence de symétrie à petite échelle (fleurs).

Les temps de calcul du tableau 4.1 ont été obtenus sur un processeur *AMD Athlon 64* 2.2 GHz, toutes implémentations étant faites en C++ [Stroustrup 2000]. Les textures générées ont environ quatre fois le nombre de points de la texture originale, à l'exception de la génération de la grande cotte de mailles, qui en a un peu plus de quarante fois. Pendant la génération, nous devons dynamiquement ajouter et supprimer des points dans $P_{\mathcal{T}_s}$, de même que trouver des voisins. Pour atteindre de bonnes performances pour ces opérations, les points $P_{\mathcal{T}_s}$ sont organisés dans une grille régulière hachée [Bentley et Friedman 1979] de compartiments de points. L'ensemble de tous les précalculs (les vecteurs extraits du champ de distance de $\mathcal{T}_e$ à la §4.4.1 et les variations de surface de chaque point de $P_{\mathcal{T}_e}$ à la §4.4.2) ont pris moins de 10 secondes.

Nous pouvons observer que nos temps de génération sont beaucoup plus rapides que ceux des approches volumétriques [Lagae *et al.* 2005, Bhat *et al.* 2004b] (§3.2.1.3), pour lesquelles des temps de calcul en termes d'heures sont rapportés. Sous des conditions similaires (exemples de génération planaire), Zhou *et al.* [2006] (§3.2.3) rapportent des temps d'environ trois minutes sur une plate-forme similaire pour aussi générer une texture d'environ quatre fois la taille de l'échantillon. Dans notre cas, nous excédons rarement une minute.

Plusieurs facteurs peuvent affecter le temps de génération. Le plus influent est le nombre de points dans $\mathcal{T}_e$. Ensuite vient le nombre de paires pour l'étape de déformation. La taille du système d'équations linéaires des splines en plaques minces est directement proportionnelle à ce nombre, de même de l'évaluation de la fonction de déformation W_{Θ^*}. La densité d'échantillon-

nage des champs de distance est un autre facteur important, qui affecte la recherche, quoique dans tous nos exemples, nous avons fait usage de densités relativement faibles, donnant des champs de distance discrets de résolution allant de $51 \times 51 \times 12$ (treillis) à $84 \times 68 \times 15$ (fleurs).

4.6 Discussion et extensions

Nous avons présenté un algorithme de génération de textures géométriques qui, à la base, procède de façon similaire aux méthodes de génération par *patch* de Efros et Freeman [2001] et Liang *et al.* [2001], mais la jonction des *patches* est effectuée par une combinaison de déformation et d'interpolation. Contrairement à la méthode mise en correspondance de contours et de déformation de Wu et Yu [2004], notre pairage de points est fait plus soigneusement car avec de la géométrie, les discontinuités pouvant résulter d'un mauvais pairage sont beaucoup plus discernables qu'avec des textures colorées.

De la même façon que Zhou *et al.* [2006] définissent leur texture géométrique, parce qu'une géométrie arbitraire peut se trouver dans la boîte englobant la texture, nous pouvons aussi générer des textures de tout type géométrique. Cependant, en substituant les maillages pour des surfaces définies par des ensembles de points, nous obtenons un gain en efficacité pour plusieurs opérations. En effet, comme nous n'avons pas à maintenir des structures de connectivité entre sommets, toute opération de découpage ou de jonction devient triviale à implémenter. Nous devons toutefois maintenir des structures de partitionnement de l'espace pour efficacement évaluer les voisinages de points, mais la méthode de Zhou *et al.* requiert aussi de telles structures pour accélérer les calculs de projection de sommets sur des faces de triangles lors de l'évaluation de leur fonction de comparaison. De plus, sans passer par des maillages, nous pouvons directement traiter des textures géométriques provenant de numérisations 3D sans nécessiter de reconstruction de maillages. Étant donné aussi qu'il existe plusieurs méthodes efficaces de conversion entre les surfaces définies par des ensembles de points et d'autres types de surfaces, plus spécifiquement les surfaces implicites [Carr *et al.* 2001, Ohtake *et al.* 2003a, Kolluri 2005, Ohtake *et al.* 2005a, Tobor *et al.* 2006][6], le point semble alors un bon choix de primitive de travail, même si la représentation finale désirée est autre.

Bien entendu, notre méthode de génération n'est pas sans limitations. Dans les sections qui suivent, nous discutons de ces limitations, puis nous parlons de quelques améliorations

[6]Construire un maillage directement à partir d'un ensemble de points est une tâche complexe. Il est cependant plus aisé de le faire à partir d'une surface implicite, par exemple avec l'algorithme des *marching cubes* [Lorensen et Cline 1987, Kobbelt *et al.* 2001].

possibles sur certains éléments de notre algorithme, et enfin, nous donnons quelques directions de recherches futures.

4.6.1 Limitations

La première limitation fondamentale de notre méthode de génération concerne la nature même des textures géométriques pouvant être générées. En effet, puisque notre algorithme suit les mêmes principes que les méthodes de génération de textures par *patch*, nous partageons alors la même restriction de ces méthodes, *i.e.* que les textures pouvant être adéquatement générées sont celles qui sont une réalisation d'un champ aléatoire de Markov (§3.1.2.2, dernier paragraphe). Cependant, à moins de considérer des méthodes de génération de plus haut niveau (nous pensons en particulier à la méthode de Funkhouser *et al.* [2004], §3.2.4), cette limitation est partagée par toute méthode de génération de textures colorées ou géométriques. Malgré tout, un très grand nombre de textures tombent dans la catégorie des champs aléatoires de Markov.

Aucun de nos exemples n'affiche des caractéristiques abruptes. Cette seconde limitation est due à la représentation en surfaces MCM. En effet, contrairement aux approches basées sur les maillages [Zelinka et Garland 2006, Zhou *et al.* 2006], les surfaces MCM ne sont *a priori* pas bien adaptées pour représenter des caractéristiques abruptes. Un certain nombre de méthodes ont été développées pour les représenter (§2.3.3.3), mais les utiliser augmenterait considérablement la complexité de l'algorithme de génération. Les surfaces MCM requièrent aussi une grande densité de points pour éviter l'ambiguïté causée par deux surfaces opposées mais proches. Malgré tout, en utilisant une représentation en surfaces MCM pures, nous pouvons tout de même couvrir un large éventail de textures géométriques.

4.6.2 Améliorations possibles

Certains problèmes peuvent survenir pendant la génération. D'abord, la déformation W_{Θ^*} (§4.4.2) peut engendrer des étirements locaux pouvant réduire la qualité de l'échantillonnage de la surface. Il faudrait alors ajouter une évaluation d'étirement local et adaptativement ajuster l'échantillonnage [Pauly *et al.* 2003b]. De plus, le fait que la déformation est calculée par un interpolant (spline en plaques minces) peut rendre la sélection du pairage plus sensible aux choix des poids. Il serait sans doute mieux d'ajouter un terme d'approximation [Wahba 1990]. Ensuite, la méthode d'intersection de rayons utilisée pour l'interpolation (§4.4.3) tente de corriger les problèmes d'échantillonnage en présence de vallées. Cependant, le problème inverse peut se

produire, *i.e.* si les points d'où émanent les rayons sont eux-mêmes dans une vallée. L'idéal serait donc d'utiliser une projection tenant compte de la distribution spatiale des voisins, comme celle de Lipman *et al.* [2007b] (§2.3.3.1).

Dans notre algorithme, la jonction entre une *patch* et $\mathcal{T}_s$ est faite par une combinaison de méthodes de déformation et d'interpolation, ce qui est respectivement similaire aux méthodes de Wu et Yu [2004] et Liang *et al.* [2001]. L'avantage d'une telle approche est la simplicité d'implémentation au niveau du découpage. Cependant, puisque la déformation n'ajuste pas parfaitement la *patch* avec son voisinage $\mathcal{V}$, il se pourrait que l'interpolation n'arrive pas à entièrement régler une situation difficile. Au lieu de découper le long de la boîte $\mathcal{B}$, il serait plus judicieux de calculer une coupe optimale similairement à la coupe de graphe de Kwatra *et al.* [2003], et comme le font Zelinka et Garland [2006] et Zhou *et al.* [2006] pour la jonction de leurs maillages. Par contre, un tel découpage implique plus de calcul, mais il pourrait faire en sorte que l'étendue de la région d'interpolation soit plus petite (une bande le long de la coupe), réduisant alors le nombre de projections à calculer.

La projection MCM est d'ailleurs le goulot d'étranglement de notre méthode de génération en terme de temps de calcul. Une plus grande proportion du temps à l'étape de recherche est passée dans le calcul du champ de distance $\delta_{\mathcal{O}_\mathcal{V}}$ (§4.4.1) et l'interpolation, qui utilise une variante de la projection, est souvent l'étape qui prend le plus de temps. Une amélioration possible serait d'utiliser la définition de surface MCM de Guennebaud et Gross [2007] (équation (2.37), §2.3.3.1), qui permettrait de réduire le nombre de points. Un autre gain pourrait être fait en cherchant à calculer le champ de distance en utilisant une méthode similaire à celle de Mauch [2000]. Le problème deviendrait alors d'identifier efficacement les cellules de la grille du champ de distance qui croisent la surface.

Un autre point d'amélioration est au niveau de l'ordre de génération et des dimensions ou de la forme de la texture générée. En réalité, notre choix actuel est presqu'arbitraire. La seule contrainte importante est le procédé par accroissement. Pour le reste, nous procédons tel que décrit principalement pour simplifier l'implémentation, *i.e.* nous avons ainsi moins de cas particuliers à gérer. Pour des formes non rectangulaires ou même rectangulaires mais sans que la taille soit un multiple de la taille d'une *patch* (ou presque, il faut tenir compte du chevauchement), le plus simple est de faire une sorte de tramage de la forme à remplir telle qu'elle soit entièrement couverte, et au final, exclure les points excédants.

4.6.3 Travaux futurs

L'intérêt premier d'une texture géométrique est de pouvoir l'appliquer sur la surface d'un objet. Or, notre méthode n'effectue que des générations planaires. Nous pourrions appliquer la texture par l'intermédiaire d'une carte en couche (*shell map*) [Porumbescu *et al.* 2005, Zhou *et al.* 2006], mais, tel que mentionné à la §3.1.6, il y a des avantages pour la qualité du résultat de pouvoir générer la texture de façon adaptée à la surface cible. Si la surface cible est représentée par un maillage, nous pourrions alors utiliser la méthode de Zhou *et al.* [2006], suivant les mêmes modifications requises qu'ils mentionnent pour passer de la génération planaire à la génération adaptée aux surfaces, puisque ces modifications sont indépendantes de la représentation elle-même de texture géométrique. Ainsi, du point de vue contribution, nous avons peu à apporter à ce niveau. Cependant, là où il y a plus d'intérêt de recherche est au niveau de la généralisation de la méthode de génération pour des surfaces cibles de représentation arbitraire.

Les méthodes de génération de textures ont toutes leurs jeux de paramètres, et celui de la nôtre est grand : taille de *patch*, taille de chevauchement, résolution de la grille d'échantillonnage des champs de distance, les divers poids dans les fonctions d'estimation de pairage, les seuils, et nous en passons. Wang *et al.* [2005] ont fait une étude afin d'optimiser les paramètres de méthodes de génération par *patch*. Afin d'obtenir une meilleure compréhension de l'effet des paramètres et ainsi pouvoir obtenir de meilleurs résultats, il serait souhaitable d'effectuer ce genre d'étude plus rigoureuse.

Il y a d'autres méthodes de génération de textures qui produisent de très bons résultats, notamment la méthode d'optimisation globale de Kwatra *et al.* [2005] (§3.1.2.3) et la méthode enrichie de Lefebvre et Hoppe [2006] (§3.1.4). Un avantage de ces méthodes (par texel) sur celles de génération par *patch* est une plus grande richesse de variation dans le résultat. Il serait donc intéressant d'étudier la possibilité d'adapter ces approches pour la génération de textures géométriques sans contrainte sur le type géométrique. Ceci est certainement possible par une approche volumétrique, mais qu'en est-il pour les approches comme celle de Zhou *et al.* [2006] ou la nôtre ? Dans ces cas, le problème fondamental est de trouver quel serait l'équivalent d'un texel. Un autre problème est d'arriver à bien traiter la continuité. Une erreur entre deux texels voisins paraît beaucoup moins qu'une erreur de jonction de surfaces. Comment arriver à bien joindre tous les morceaux dans une génération par optimisation globale ou une génération parallèle ? Contrairement à la génération de textures colorées, la complexité entre la

génération par texel et par *patch* semble inversée, *i.e.* plus facile par *patch* et plus difficile par "texel"[7] .

Enfin, nous n'avons parlé jusqu'à maintenant que de géométrie. Cependant, la couleur est un autre attribut de surface important pour enrichir la qualité visuelle des objets. Seuls les travaux de Park *et al.* [2005] (§3.2.3) ont abordé la problématique de génération de géométrie et de couleur en simultané. Pour éviter de trop contraindre le choix de candidats pour les *patches*, ils séparent complètement l'aspect de géométrie et de couleur. Cependant, la plupart du temps, une corrélation existe entre les deux. Il faudrait donc explorer d'autres méthodes pour traiter ces deux aspects de sorte à préserver la corrélation, mais sans trop contraindre l'espace de recherche. Une voie d'exploration est la génération en deux passes, où la première s'occuperait de générer la texture géométrique sans se soucier de la couleur, et la seconde exploiterait des méthodes d'analogies [Hertzmann *et al.* 2001, Hertzmann *et al.* 2002] pour générer la couleur sur la texture géométrique générée, *i.e.* l'analogie se ferait entre les couleurs et les caractéristiques géométriques de la texture. Mais d'ignorer la couleur pendant la génération de la géométrie de la texture pourrait produire des régions d'ambiguïté pour l'analogie. Alors peut-être faudrait-il tout de même ajouter un facteur en fonction de la couleur dans la recherche de *patches* similaires, mais pondéré de sorte à donner priorité à la similarité géométrique.

[7]Nous excluons les approches volumétriques.

Chapitre 5

Édition multirésolution de détails géométriques

Dans le chapitre précédent, nous avons présenté une méthode de génération de textures géométriques. Ce type de méthode, ou tout autre méthode de génération de détails géométriques, ne correspond pas nécessairement à l'étape finale dans la production d'un objet. Un artiste pourrait effectuer des opérations par la suite apportant des modifications à l'objet, telles que des changements de forme, de propriétés de surface (par exemple, la couleur), etc.

Pour maintenir la qualité de l'objet en termes de détails géométriques, il est important que toute opération appliquée sur l'objet préserve autant que possible ses détails. Autrement, il serait plutôt préférable de recommencer la génération des détails après l'opération, ce qui prend plus de ressources et perd l'interactivité, en particulier si les changements apportés sont effectués en fonction des détails générés.

Ainsi, nous cherchons une méthode générique permettant d'effectuer des opérations sur un objet tout en préservant les détails, et aussi d'éditer ces derniers. Une approche classique utilise une représentation multirésolution de la surface. Il s'agit de la décomposer en une série de niveaux ayant de moins en moins de détails, où chaque niveau est encodé relativement à un niveau moins détaillé de sorte que toute opération effectuée à l'un d'eux se répercute au niveau le plus détaillé après reconstruction, *i.e.* le décodage complet.

La plupart des travaux en matière de représentations multirésolutions concernent les maillages polygonaux. Or, puisque nous avons présenté une méthode de génération de textures géométriques utilisant des surfaces définies par des ensembles de points, nous sommes plutôt inté-

ressés par les travaux concernant cette représentation. Il se trouve qu'il y en a peu, et à quelques exceptions près, la plupart des méthodes développées sont restrictives dans leurs applications.

Le reste de ce chapitre présente d'abord le principe général des surfaces multirésolutions (§5.1), suivi d'un bref survol de la littérature sur les représentations multirésolutions excluant les représentations par points (§5.2), qui sont présentées par la suite (§5.3). Enfin, nous finissons par une brève discussion sur d'autres méthodes de déformation préservant les détails géométriques (§5.4).

5.1 Principe des surfaces multirésolutions

Le principe général des représentations multirésolutions peut être illustré à la figure 5.1[1]. Il s'agit de prendre la surface initiale (le niveau L de la colonne de gauche) et de la décomposer en une série de L niveaux de *résolution* de plus en plus faible, *i.e.* de plus en grossiers ou lisses. Il y a deux opérations fondamentales : la *décomposition* (colonne de gauche) et la *reconstruction* (colonne de droite). Dans ce qui suit, notons $\mathcal{M}^{[l]}$ la surface associée au niveau l et $D^{[l]}$ les détails correspondants (colonne du centre).

La *décomposition* consiste à prendre un niveau $l \in \mathbb{Z}[1, L]$, produire un niveau $l-1$ par un opérateur Υ, et extraire la *différence* entre $\mathcal{M}^{[l]}$ et $\mathcal{M}^{[l-1]}$ pour exprimer les détails $D^{[l]}$, *i.e.*

$$\mathcal{M}^{[l-1]} = \Upsilon(\mathcal{M}^{[l]}) \tag{5.1}$$

$$D^{[l]} = \mathcal{M}^{[l]} \ominus \mathcal{M}^{[l-1]}. \tag{5.2}$$

L'opérateur Υ consiste à rendre $\mathcal{M}^{[l]}$ *moins détaillée*. La notion exacte de *moins détaillé* varie en fonction de la représentation de base. Pour les maillages, il s'agit soit de produire une surface plus grossière, *i.e.* simplifier le maillage en réduisant le nombre de sommets et de faces, ou de seulement la rendre plus lisse, *i.e.* réduire les hautes fréquences dans les détails géométriques, sans changer la connectivité du maillage.

La *reconstruction* est essentiellement l'opération inverse de la décomposition. À partir d'un niveau $l-1$, $l \in \mathbb{Z}[1, L]$, elle reconstruit le niveau l en *ajoutant* ses détails, *i.e.*

$$\mathcal{M}^{[l]} = \Upsilon^{-1}(\mathcal{M}^{[l-1]}) \oplus D^{[l]}. \tag{5.3}$$

[1] Nous devrions dire que cette illustration correspond plutôt au principe le plus fréquent. Comme nous le verrons à la prochaine section, il existe un certain nombre de variantes.

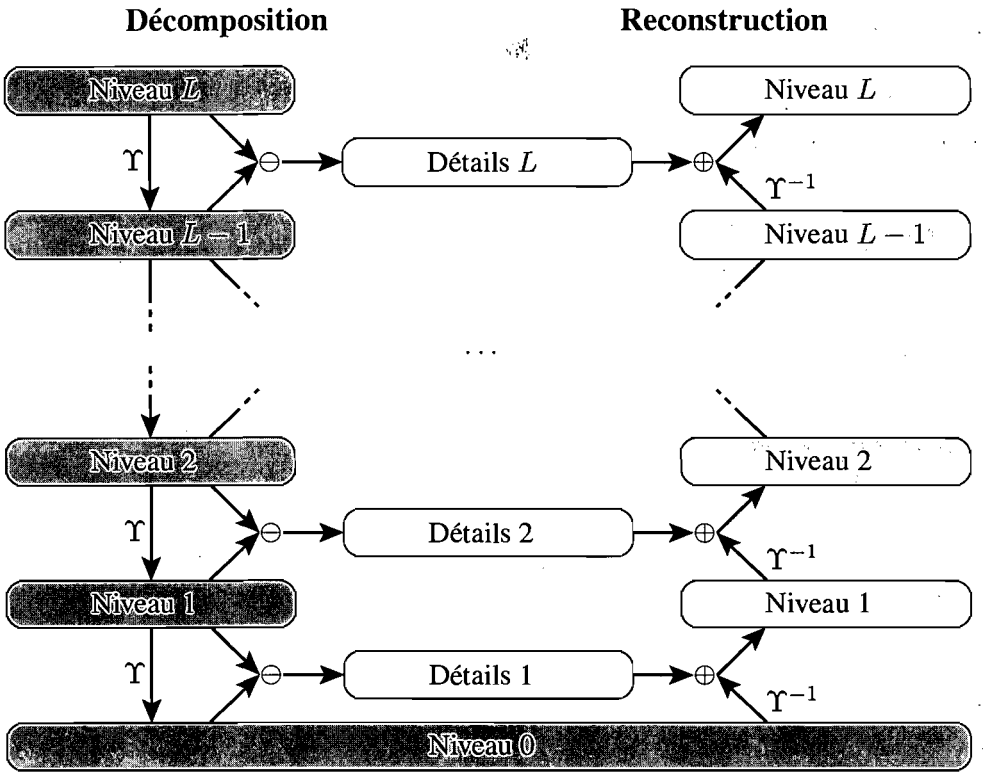
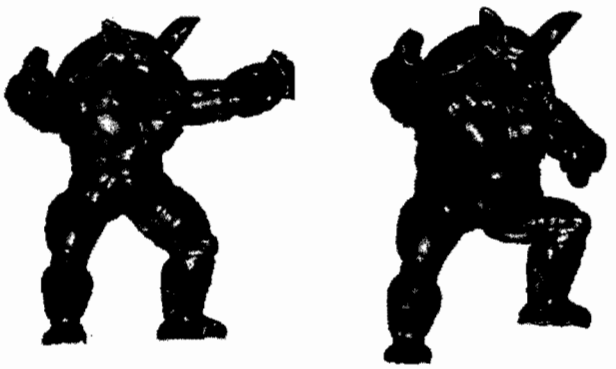


FIG. 5.1 – Opérations fondamentales des représentations multirésolutions. L'opérateur Υ transforme un niveau en un niveau de plus faible résolution (plus grossier ou seulement plus lisse). En appliquant l'opérateur $\ominus$ entre deux niveaux adjacents, un encodage relatif est obtenu (les détails) du niveau supérieur par rapport au niveau inférieur. L'opérateur $\oplus$ applique les détails à un niveau inférieur pour reconstruire le niveau supérieur.

Dans le cas des maillages, l'opérateur Υ^{-1} dénote la réinsertion des sommets et faces dans $\mathcal{M}^{[l-1]}$ si la décomposition produit un maillage plus grossier[2]. Il s'agit de l'identité si Υ ne change pas la connectivité.

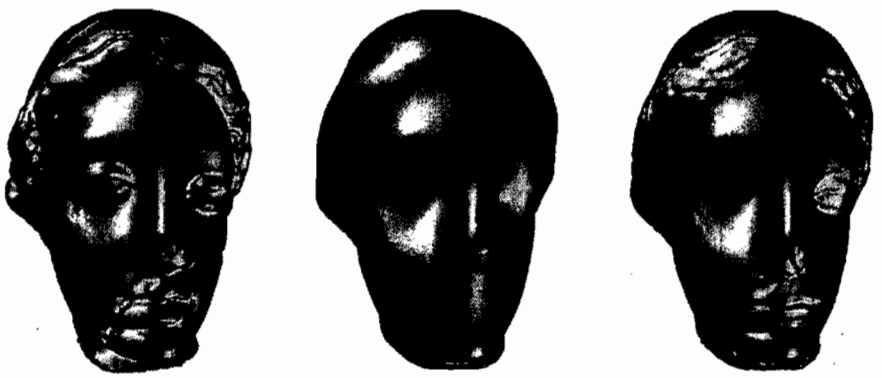
L'intérêt des représentations multirésolutions est de pouvoir effectuer des opérations sur les niveaux moins détaillés, *i.e.* principalement sur $\mathcal{M}^{[0]}$ ou alors laisser le choix libre à l'usager du niveau de détails qu'il veut affecter [Khodakovsky et Schröder 1999], et obtenir une reconstruction conséquente. Autrement dit, il s'agit de pouvoir effectuer des déformations *grossières* qui préservent les détails. Un exemple est montré à la figure 5.2 (a). Il est donc important que les détails $D^{[l]}$, $l \in \mathbb{Z}[1, L]$, soient encodés de façon non seulement relative au niveau $l-1$,

[2]Ceci est un peu un abus de notation, car Υ^{-1} n'est pas exactement l'opération inverse de Υ. Par exemple, si Υ réduit le nombre de faces et de sommets, alors les sommets insérés par Υ^{-1} n'auront pas nécessairement les mêmes positions qu'avant.



Original Déformé

(a)



Original Lissage Filtre d'arrêt (*stop band*)

(b)

FIG. 5.2 – Exemple d'éditions multirésolutions : (a) déformation préservant les détails et (b) filtrage des détails. Les images proviennent (a) de [Zorin *et al.* 1997] et (b) de [Guskov *et al.* 1999].

mais aussi de façon localisée pour que le décodage puisse exprimer les changements d'orientation et les étirements locaux en plus des translations. Les définitions des différents opérateurs dépendent de la représentation multirésolution utilisée, et ce n'est pas toutes les représentations développées qui possèdent des détails encodés selon toutes les propriétés désirées. Nous donnerons plus de détails sur les diverses représentations dans la prochaine section.

Il n'y a pas que les déformations qui peuvent profiter des niveaux plus grossiers pour être plus efficaces. Par exemple, le remaillage [Botsch et Kobbelt 2004] peut améliorer la qualité des triangles (pour un maillage triangulaire) de sorte à rendre une opération d'édition plus robuste. Pour plusieurs représentations multirésolutions, ceci n'est pas vraiment nécessaire. Un tel remaillage est plus utile lorsqu'il est question de maillages arbitraires (§5.2.4).

Puisque l'information des détails est dissociée de la surface à un niveau donné, il est alors possible d'effectuer aussi des opérations d'édition sur ces détails eux-mêmes, dans la mesure où leur représentation le permet. Une forme de traitement de signal peut ainsi être faite. Par exemple, le lissage et le filtre d'arrêt (*stop band*) de la figure 5.2 sont produits en multipliant la valeur des détails des niveaux plus hauts (lissage) ou intermédiaires (filtre d'arrêt) par zéro ou une petite valeur [Guskov *et al.* 1999]. Les représentations multirésolutions peuvent aussi aider pour le transfert les détails d'une surface à une autre [Ma et Mann 2001, Biermann *et al.* 2002]. Le découplement des détails peut aussi aider pour effectuer des métamorphoses entre deux objets [Lee *et al.* 1999].

5.2 Représentations multirésolutions

Nous présentons dans cette section un survol de la littérature sur les représentations multirésolutions pour splines, surfaces de subdivision et autres maillages.

Remarque 5.1 (Opérateurs) *Nous faisons autant que possible référence aux opérateurs et au schéma de la section précédente lorsque nous décrivons les représentations.*

5.2.1 Édition hiérarchique

Au meilleur de nos connaissances, les premiers travaux sur l'édition multirésolution de surfaces ont été effectués par Forsey et Bartels [1988] à propos de surfaces B-splines (§2.1.1.2) hiérarchiques. Dans leur cas, l'étape de décomposition n'existe pas. Il s'agit plutôt d'un système interactif partant d'une surface B-spline grossière qui permet à un usager d'insérer des nouveaux points de contrôle pour ajouter des détails. Les insertions peuvent être considérées comme l'opérateur Υ^{-1}. La particularité de leur système est que les nouveaux points de contrôle sont exprimés de façon relative à la surface parente avant insertion, *i.e.* le détail est une différence de position dans un repère local. Ainsi, l'opérateur $\oplus$ est simplement l'ajout des déplacements enregistrés.

Un problème bien connu des surfaces B-splines est le fait qu'elles peuvent adéquatement représenter seulement des surfaces homéomorphes au plan, au cylindre ou au tore [Zorin *et al.* 2000]. Suivant la ligne d'idées de Forsey et Bartels, Yvart *et al.* [2005] proposent d'utiliser plutôt des splines hiérarchiques triangulaires, alors que Pulli et Lounsbery [1997] font usage de surfaces de subdivision. Ces dernières ont d'ailleurs connu une grande popularité pour les sur-

$$\begin{pmatrix} \boldsymbol{O}^{[0]} \\ \boldsymbol{N}^{[0]} \end{pmatrix} = \left(\begin{array}{cccc} \frac{7}{16} & \frac{3}{16} & \frac{3}{16} & \frac{3}{16} \\ \frac{3}{16} & \frac{7}{16} & \frac{3}{16} & \frac{3}{16} \\ \frac{3}{16} & \frac{3}{16} & \frac{7}{16} & \frac{3}{16} \\ \frac{3}{16} & \frac{3}{16} & \frac{3}{16} & \frac{7}{16} \\ \hline \frac{3}{8} & \frac{3}{8} & \frac{1}{8} & \frac{1}{8} \\ \frac{3}{8} & \frac{1}{8} & \frac{1}{8} & \frac{3}{8} \\ \frac{1}{8} & \frac{3}{8} & \frac{1}{8} & \frac{3}{8} \\ \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \\ \frac{3}{8} & \frac{1}{8} & \frac{3}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} \end{array}\right) \begin{array}{l} \ldots\ 1 \\ \ldots\ 2 \\ \ldots\ 3 \\ \ldots\ 4 \\ \ldots\ 5 \\ \ldots\ 6 \\ \ldots\ 7 \\ \ldots\ 8 \\ \ldots\ 9 \\ \ldots\ 10 \end{array}$$

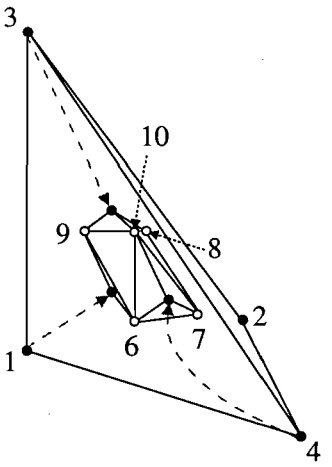


FIG. 5.3 – Exemple d'un opérateur matriciel pour la subdivision de Loop [1987] sur un tétraèdre.

faces multirésolutions, justement à cause de leurs opérations naturelles de subdivision en plus de leur capacité de représenter des objets de topologie arbitraire.

5.2.2 Surfaces de subdivision et décomposition en ondelettes

Se basant sur les surfaces de subdivision (§2.1.1.3) et les ondelettes [Stollnitz *et al.* 1996], Lounsbery *et al.* [1994, 1997] ont été essentiellement les premiers à proposer une représentation multirésolution suivant pleinement le schéma de la figure 5.1. Soient $V^{[l]}$ l'ensemble des sommets d'un maillage après l subdivisions, et $\boldsymbol{V}^{[l]}$ leur représentation sous forme matricielle[3]. L'opération de subdivision pour passer du niveau l à $l+1$ peut s'exprimer sous forme matricielle ainsi :

$$\boldsymbol{V}^{[l+1]} = \begin{pmatrix} \boldsymbol{O}^{[l]} \\ \boldsymbol{N}^{[l]} \end{pmatrix} \boldsymbol{V}^{[l]} \tag{5.4}$$

où $\boldsymbol{O}^{[l]}$ est une matrice $o^{[l]} \times o^{[l]}$, $o^{[l]} = |V^{[l]}|$, correspondant à l'opération de lissage sur les *anciens* sommets, et $\boldsymbol{N}^{[l]}$ est une matrice donnant la position des *nouveaux* sommets. Cette dernière matrice est de dimension $n^{[l]} \times o^{[l]}$, où $n^{[l]} > o^{[l]}$ et dépend du schéma de subdivision employé. La position des termes dans les colonnes des matrices dépend de la connectivité du maillage. Un exemple simple d'un opérateur de subdivision sous forme matricielle se trouve à la figure 5.3.

[3]La représentation sous forme matricielle de $V^{[l]}$ est une matrice $\boldsymbol{V}^{[l]}$ de dimensions $|V^{[l]}| \times 3$ où chaque ligne correspond à un élément de $V^{[l]}$.

Avec ceci, une simple décomposition multirésolution peut être obtenue ainsi :

$$\boldsymbol{V}^{[l-1]} = \left(\boldsymbol{O}^{[l-1]}\right)^{-1} \left(\mathbf{1}_{o^{[l-1]}} \quad \mathbf{0}_{o^{[l-1]} \times n^{[l-1]}}\right) \boldsymbol{V}^{[l]} \tag{5.5}$$

$$\boldsymbol{D}^{[l]} = \left(\mathbf{0}_{n^{[l-1]} \times o^{[l-1]}} \quad \mathbf{1}_{n^{[l-1]}}\right) \boldsymbol{V}^{[l]} - \boldsymbol{N}^{[l-1]} \left(\boldsymbol{O}^{[l-1]}\right)^{-1} \left(\mathbf{1}_{o^{[l-1]}} \quad \mathbf{0}_{o^{[l-1]} \times n^{[l-1]}}\right) \boldsymbol{V}^{[l]} \tag{5.6}$$

i.e. les sommets $V^{[l-1]}$ proviennent de l'inversion de l'opération de subdivision sur la portion des *anciens* sommets dans $V^{[l]}$, et les détails correspondent à la différence de position entre les *nouveaux* sommets dans $V^{[l]}$ et les nouveaux sommets calculés par la subdivision à partir des sommets $V^{[l-1]}$. Ainsi, l'opérateur Υ est donné par l'équation (5.5) et l'opérateur $\ominus$ par l'équation (5.6), où les détails se retrouvent aussi sous forme matricielle. Il est à noter que le nombre de rangées de $\boldsymbol{D}^{[l]}$ est égal à $n^{[l-1]}$, et donc les sommets du niveau $l-1$ et leurs détails requièrent la même quantité d'information, comme nous devrions nous attendre d'une décomposition en ondelettes. La reconstruction correspondante à cette décomposition est simplement l'application de la subdivision (l'opérateur Υ^{-1}) suivie de l'ajout des détails sur les nouveaux sommets (l'opérateur $\oplus$), *i.e.*

$$\boldsymbol{V}^{[l]} = \begin{pmatrix} \boldsymbol{O}^{[l-1]} & \mathbf{0}_{o^{[l-1]} \times n^{[l-1]}} \\ \boldsymbol{N}^{[l-1]} & \mathbf{1}_{n^{[l-1]}} \end{pmatrix} \begin{pmatrix} \boldsymbol{V}^{[l-1]} \\ \boldsymbol{D}^{[l]} \end{pmatrix}. \tag{5.7}$$

Lounsbery *et al.* [1997] ont mentionné que la simple décomposition présentée précédemment produit des surfaces grossières n'étant pas de bonnes approximations de moindres carrés des surfaces plus détaillées. Ils considèrent alors une partie de l'information des détails pour calculer les sommets $V^{[l-1]}$ ainsi :

$$\boldsymbol{V}^{[l-1]} = \left(\boldsymbol{O}^{[l-1]}\right)^{-1} \left(\mathbf{1}_{o^{[l-1]}} \quad \mathbf{0}_{o^{[l-1]} \times n^{[l-1]}}\right) \boldsymbol{V}^{[l]} + \boldsymbol{A}^{[l-1]} \boldsymbol{D}^{[l]} \tag{5.8}$$

où $\boldsymbol{A}^{[l-1]}$ est une matrice $o^{[l-1]} \times n^{[l-1]}$ exprimant essentiellement l'ajustement aux sommets calculés par le premier terme de l'équation en fonction des détails du niveau. L'équation (5.8) prend alors la place de l'équation (5.5), mais l'équation (5.6) demeure inchangée. Ceci implique que la reconstruction doit être ajustée pour retirer la portion de détails conservée dans

les sommets $V^{[l-1]}$ avant d'ajouter les détails, et alors l'équation (5.7) devient :

$$\boldsymbol{V}^{[l]} = \begin{pmatrix} \boldsymbol{O}^{[l-1]} & -\boldsymbol{O}^{[l-1]}\boldsymbol{A}^{[l-1]} \\ \boldsymbol{N}^{[l-1]} & \mathbf{1}_{n^{[l-1]}} - \boldsymbol{N}^{[l-1]}\boldsymbol{A}^{[l-1]} \end{pmatrix} \begin{pmatrix} \boldsymbol{V}^{[l-1]} \\ \boldsymbol{D}^{[l]} \end{pmatrix}. \tag{5.9}$$

Nous ne plongerons pas dans le développement du calcul de $\boldsymbol{A}^{[l-1]}$, car un exposé plus en profondeur sur les ondelettes serait requis, ce qui sort de la portée de cette thèse. Nous invitons le lecteur à consulter la référence originale [Lounsbery *et al.* 1997][4].

Un problème notoire avec l'approche précédente est que le maillage original doit avoir une connectivité de subdivision [Taubin 2002], *i.e.* il doit être semi-régulier. Une approche commune pour contourner ce problème est de reconstruire un maillage avec une connectivité telle que l'erreur d'approximation soit aussi petite que possible [Eck *et al.* 1995, Lee *et al.* 1998, Guskov *et al.* 2002, Boier-Martin 2003]. Ceci est habituellement fait en construisant d'abord un maillage très grossier qui approxime la surface originale. Cette dernière est ensuite paramétrée par rapport au maillage grossier, et un nouveau maillage est produit en subdivisant le maillage grossier et en retrouvant la position des sommets *via* la paramétrisation. Une approche plus particulière est celle de Boier-Martin [2003], qui passe par l'intermédiaire d'une image de géométrie (§3.2.1.2) pour segmenter des régions de la surface et construire le maillage grossier.

L'origine du problème de connectivité est l'usage d'un schéma de subdivision régulière. Une autre solution à ce problème est alors de développer un schéma de subdivision irrégulier inversible [Kim *et al.* 2001, Valette et Prost 2004]. Le truc est de conserver pour chaque face du maillage à un niveau donné la règle de subdivision. Un travail supplémentaire doit alors être fait pour choisir la règle à utiliser à chaque face pour définir Υ. Une fois ceci fait, les matrices $\boldsymbol{O}^{[l]}$ et $N^{[l]}$ sont composées suivant les règles choisies.

Guskov *et al.* [2000] ont présenté une autre représentation à base d'ondelettes, appelée maillages normaux (*normal meshes*). Ceux-ci sont cependant conceptuellement plus simples. La représentation se base aussi sur la subdivision, mais pas dans le même sens que Lounsbery *et al.* [1997]. La construction de la représentation s'effectue de bas en haut, et donc la décomposition suit le même axe que la reconstruction. Un maillage grossier est généré, similairement aux méthodes précédentes, mais il doit interpoler le maillage original. Puis chaque face est subdivisée en quatre, ce qui correspond à l'opération Υ^{-1}. Pour chaque nouveau sommet sur une

[4] À noter que dans la référence originale, la notation α est employée au lieu de $\boldsymbol{A}$, et une matrice $\boldsymbol{A}$ est utilisée pour décrire l'opération de l'équation (5.8). Nous avons changé la notation dans cette thèse pour rester cohérent avec notre notation.

face, la distance entre le sommet et le maillage original, le long de l'axe de la normale à ce sommet, est enregistrée, ce qui forme l'opérateur $\ominus$. L'ensemble des distances aux nouveaux sommets forme les détails du niveau courant. Les nouveaux sommets sont ensuite déplacés, ce qui correspond aussi à l'opérateur $\oplus$, et un nouveau niveau est calculé[5]. Ceci donne une représentation compacte à raison d'un seul scalaire par sommet, à l'exception des quelques sommets du maillage au niveau 0. Friedel *et al.* [2004] ont étendu la méthode de construction avec une optimisation des valeurs de distance pour être en mesure de produire des niveaux qui approximent plutôt qu'interpolent la surface originale, ce qui réduit les artefacts d'échantillonnage.

5.2.3 Surfaces de subdivision multirésolutions génériques

Il existe d'autres représentations multirésolutions basées sur la subdivision sans qu'il soit question d'ondelettes. La représentation de Zorin *et al.* [1997, 1998], à la base, se distingue de la méthode de Lounsbery *et al.* [1997] par l'utilisation d'un lissage arbitraire en simplifiant le maillage. En réutilisant commodément la notation employée pour décrire les opérations de la représentation de Lounsbery *et al.* [1997], les sommets $V^{[l-1]}$ peuvent être calculés ainsi[6] :

$$\boldsymbol{V}^{[l-1]} = \boldsymbol{\Phi}^{[l]}\boldsymbol{V}^{[l]} \tag{5.10}$$

$$\boldsymbol{D}^{[l]} = \boldsymbol{V}^{[l]} - \begin{pmatrix} \boldsymbol{O}^{[l-1]} \\ \boldsymbol{N}^{[l-1]} \end{pmatrix} \boldsymbol{V}^{[l-1]} \tag{5.11}$$

où $\boldsymbol{\Phi}^{[l]}$ est une matrice $o^{[l-1]} \times |V^{[l]}|$ correspondant au lissage des sommets dans $V^{[l]}$ correspondant aux *anciens* sommets avant subdivision. En particulier, Zorin *et al.* font usage du lissage de Taubin [1995]. Ici, contrairement aux représentations à base d'ondelettes, $|D^{[l]}| = |V^{[l]}|$, ce qui est nécessaire pour permettre l'usage d'un lissage arbitraire car dans ce cas-ci, $\begin{pmatrix} \mathbf{1}_{o^{[l-1]}} & \mathbf{0}_{o^{[l-1]} \times n^{[l-1]}} \end{pmatrix} \boldsymbol{V}^{[l]} \neq \boldsymbol{O}^{[l-1]}\boldsymbol{V}^{[l-1]}$ en général. La reconstruction découle trivialement de l'équation (5.11) en transférant le second terme du côté droit au côté gauche, *i.e.*

$$\boldsymbol{V}^{[l]} = \begin{pmatrix} \boldsymbol{O}^{[l-1]} \\ \boldsymbol{N}^{[l-1]} \end{pmatrix} \boldsymbol{V}^{[l-1]} + \boldsymbol{D}^{[l]}. \tag{5.12}$$

[5] La méthode exacte de décomposition est en réalité plus élaborée pour tenir compte des problèmes possibles d'intersection, mais l'idée essentielle est là.

[6] Nous reprenons cette notation seulement pour aider la comparaison et uniformiser les explications avec la description de la représentation de Lounsbery *et al.* [1997]. Il ne s'agit pas de la notation de l'article original, ni de la notation qui exprime une façon efficace d'effectuer les calculs.

Une autre nuance importante de la représentation de Zorin *et al.* se trouve au niveau de l'encodage des détails. Les détails tels que calculés par l'équation (5.11) sont exprimés dans le repère global, ce qui n'est pas souhaitable pour pouvoir bien effectuer des déformations aux niveaux plus grossiers (§5.1). Ainsi, suite à ce calcul, les détails sont transformés pour être exprimés dans un repère local à chaque sommet provenant du second terme de l'équation (5.11). Puis, avant le calcul de l'équation (5.12), les détails sont transformés à nouveau, mais inversement, *i.e.* à partir du repère local de chaque sommet provenant du premier terme de l'équation (5.12), qui pourraient avoir changé suite à une déformation quelconque.

Ce qui a été décrit ci-avant est la base de la représentation de Zorin *et al.* [1997]. Pour réduire les temps de calcul et le stockage, ils ont proposé une représentation adaptative qui ne conserve pas les détails dont la magnitude est sous un seuil donné. Dans les grandes lignes, pour calculer un niveau $l-l$ à partir de l, les équations (5.10) et (5.11) sont d'abord évaluées, puis les petits détails sont retirés de $D^{[l]}$. Durant la reconstruction, la subdivision et l'ajout des détails sont effectués de façon adaptative selon les informations de détails disponibles, *i.e.* les sommets aux petits détails ne sont pas reconstruits[7], ce qui donne une reconstruction plus efficace en temps et en mémoire.

Une représentation similaire mais plus compacte peut aussi être obtenue en conservant uniquement le déplacement dans l'axe de la normale à chaque sommet comme valeur de détail [Lee *et al.* 2000], ce qui n'est pas sans rappeler les maillages normaux décrits auparavant dans cette section. Par contre, ceci implique un rééchantillonnage de la surface. Lee *et al.* [2000] ont présenté en fait une représentation qui construit un niveau très grossier, qu'ils subdivisent ensuite pour obtenir un domaine lisse à partir duquel ils calculent les détails le long des normales, *i.e.* leur représentation multirésolution possède seulement deux niveaux. Cependant, nous retrouvons le concept de Zorin *et al.* [1997] de construire un maillage lissé et de calculer les détails comme une différence entre celui-ci et le niveau plus détaillé, et alors il n'est pas difficile d'imaginer une généralisation à un nombre plus grand de niveaux.

5.2.4 Représentations multirésolutions pour maillages arbitraires

Les méthodes de la section précédente ont écarté l'usage des ondelettes pour moins restreindre le choix d'opération de lissage dans la décomposition [Zorin *et al.* 1997], mais la subdivision était encore employée comme opération fondamentale de raffinement du maillage.

[7] Cette description est très sommaire et cache plusieurs subtilités quant au maintien d'informations de connectivité appropriées, mais élaborer davantage sort de la portée de cette thèse.

Kobbelt *et al.* [1998] ont écarté aussi l'usage de subdivision avec des représentations multirésolutions pour des maillages arbitraires de variété (Hubeli et Gross [2001] ont apporté des extensions pour gérer les maillages de non-variété (*non manifold*)). Kobbelt *et al.* [1998] ont considéré deux types de représentations pouvant être combinés ensemble : l'une provient d'une décomposition par décimation pour produire des surfaces plus grossières, et l'autre d'une décomposition par lissage uniquement pour capturer les détails.

La décomposition par décimation fait usage des maillages progressifs de Hoppe [1996]. Ainsi, l'opérateur Υ correspond à un groupe d'opérations de réduction d'arêtes, et les détails correspondent essentiellement à l'information requise pour défaire chaque réduction. La reconstruction est alors simplement définie par l'opérateur Υ^{-1}, l'inverse de Υ. Guskov *et al.* [1999] ont proposé de ne pas grouper les opérations, mais plutôt d'en effectuer une seule par niveau, ce qui peut produire un grand nombre de niveaux mais avec un élément de détail par niveau. Ceci permet de dynamiquement calculer des groupes en fonction de l'opération d'édition appliquée.

La décomposition par lissage, contrairement aux autres représentations décrites précédemment, ne change ni la connectivité, ni le nombre de sommets dans les maillages. La notion de résolution est alors une question de *fréquence* des détails, *i.e.* les ondulations au-dessus d'un domaine lisse. Ainsi, l'opérateur Υ^{-1} est seulement une opération de lissage des sommets, et $\ominus$ calcule les différences de position entre les sommets du niveau détaillé et du niveau plus lisse. De la même façon que Zorin *et al.* [1997], ces différences sont stockées dans un repère local. Pour la reconstruction, Υ^{-1} est l'identité alors que $\oplus$ est simplement l'addition des différences réexprimées dans le repère global.

5.3 Multirésolution et représentation par points

La notion de multirésolution pour les représentations par points a souvent été utilisée pour décrire des structures hiérarchiques, la plupart pour des fins de rendu [Pfister *et al.* 2000, Rusinkiewicz et Levoy 2000, Botsch *et al.* 2002, Coconu et Hege 2002, Dachsbacher *et al.* 2003, Guennebaud et Paulin 2003, Ohtake *et al.* 2003a, Pajarola 2003, Wand et Straßer 2003, Duguet et Drettakis 2004, Gobbetti et Marton 2004, Ji *et al.* 2004, Reuter *et al.* 2004, Park *et al.* 2004, Sainz *et al.* 2004b, Tobor *et al.* 2004, Boubekeur *et al.* 2005, Pajarola *et al.* 2005, Wu *et al.* 2005, Huang *et al.* 2006, He et Liang 2007]. Ces structures ne sont pas basées sur le principe de multirésolution énoncé à la §5.1. Elles correspondent généralement à des niveaux de détails (*levels of detail* ou *LOD*) statiques qui performent bien pour le

rendu, mais elles sont moins appropriées pour l'édition, outre l'application de transformations uniformes globales. Néanmoins, certaines d'entre elles font usage d'un encodage relatif entre les niveaux. Nous en reparlerons à la §5.3.1.

Pour définir une représentation multirésolution pour des surfaces définies par des ensembles de points selon le principe de la §5.1, nous devons surmonter certaines difficultés. En effet, les maillages ont l'avantage de maintenir explicitement la connectivité des sommets, et donc la reconstruction d'un niveau plus détaillé est déterminée par cette connectivité. Pour les surfaces définies par des ensembles de points, il n'y a pas de connectivité explicite. Elle est substituée par la notion de *voisinage* autour d'un point (§2.3.1.1). Or un voisinage peut changer si la position des points change. Pour toute méthode réduisant le nombre de points entre les niveaux, ceci crée donc un problème de cohérence lors de la reconstruction entre la génération des points d'un niveau plus détaillé et l'encodage des détails.

Un certain nombre de représentations multirésolutions ont été développées, chacune se rapprochant à divers degré du principe de la §5.1. Il y a les hiérarchies de points avec encodage relatif des positions (§5.3.1), déjà mentionnées précédemment, les représentations progressives (§5.3.2) et les surfaces multiéchelles (§5.3.3). Nous les décrivons dans les prochaines sections en s'appuyant sur le schéma de la figure 5.1.

5.3.1 Hiérarchies de points avec encodage relatif des positions

À la base, les hiérarchies de points correspondent à une arborescence quelconque ayant un point associé à chaque nœud et l'ensemble des noeuds d'une profondeur donnée forme un niveau de résolution. Dans une hiérarchie avec encodage relatif, toutes les positions des enfants d'un nœud sont exprimées de façon relative à celle du nœud plutôt qu'en coordonnées absolues. Un bon exemple d'une telle hiérarchie est celle utilisée dans *QSplat* [Rusinkiewicz et Levoy 2000]. Le principe général de ces représentations multirésolutions est décrit ci-après.

Décomposition L'opérateur Υ prend les points d'un niveau l et les réorganise en agrégats. L'ensemble des centres de ces agrégats forment les points du niveau $l-1$. Ensuite, l'opérateur $\ominus$ correspond à prendre la différence entre la position des points de chaque agrégat et le centre de ce dernier. Les détails sont encodés suivant les agrégats du niveau.

Reconstruction L'opérateur Υ^{-1} visite chaque point du niveau $l-1$ et produit un point pour chaque valeur de détail de l'agrégat associé. La position des nouveaux points est calculée en

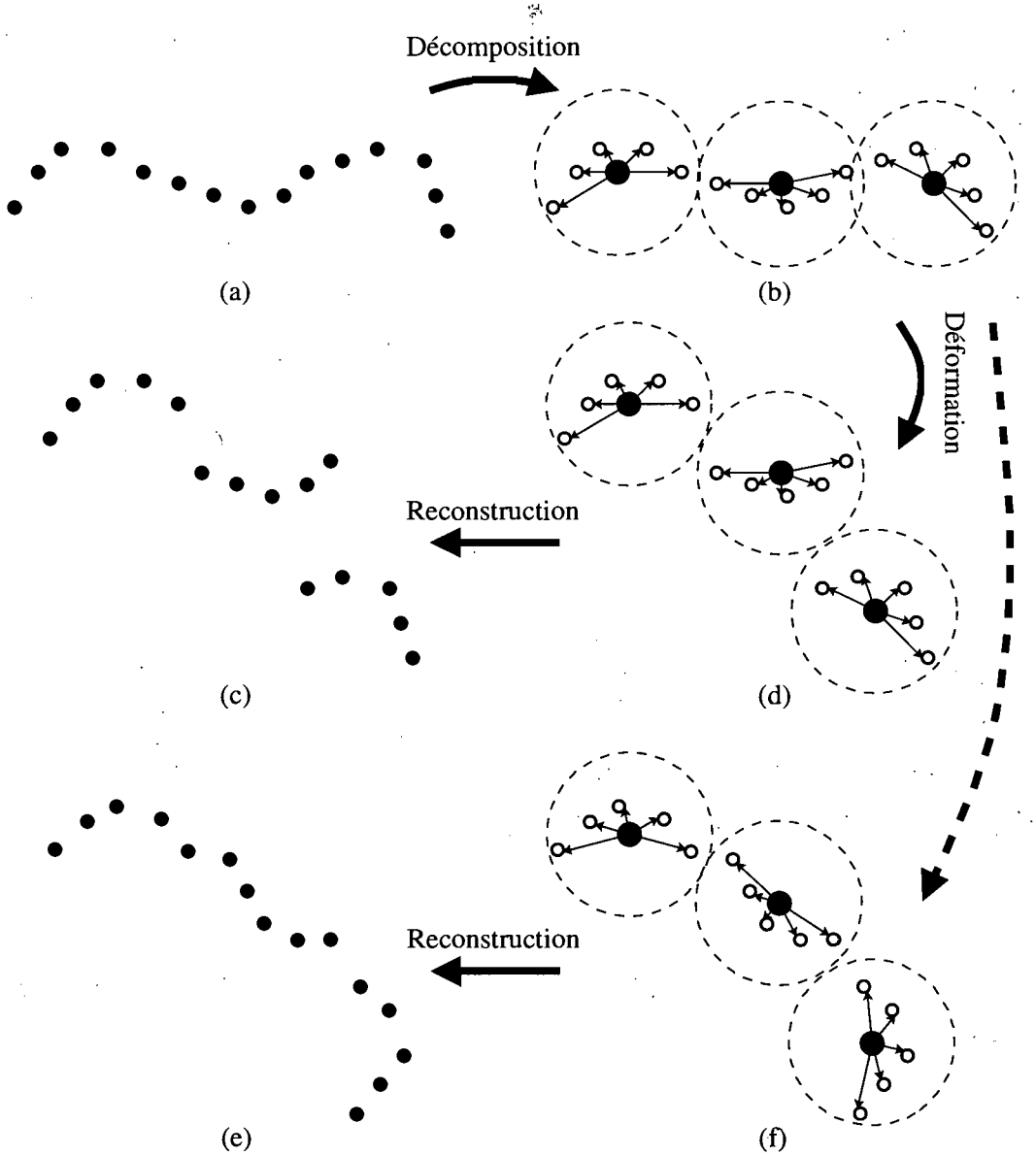


FIG. 5.4 – Déformation avec encodage global *vs.* local. (a) La courbe originale est (b) décomposée, où les points plus gros indiquent les centres des agrégats formant le niveau plus grossier et les petites flèches indiquent les détails (vecteurs de différences). L'exemple est donné pour un seul niveau de décomposition. (d) Avec un encodage global, si nous déformons le niveau grossier, (c) nous n'obtenons pas une reconstruction cohérente avec les détails originaux. (f) Avec un encodage local, suite à une déformation, les vecteurs de différences seront transformés pour tenir compte des rotations locales et (e) obtenir une reconstruction plus adéquate.

ajoutant la valeur du détail (un vecteur) à la position du point visité, ce qui définit l'opérateur $\oplus$.

La construction de la hiérarchie évite le problème de la génération cohérente des positions des points d'un niveau supérieur par rapport à l'encodage des détails. Cependant, l'encodage relatif de ce type de hiérarchie est calculé par rapport aux axes globaux, et non à un repère (*frame*) local, ce qui fait en sorte qu'une déformation d'un niveau plus grossier n'entraînera pas une déformation cohérente aux niveaux plus détaillés. La figure 5.4 (a)–(d) illustre ce problème.

Pour contrer ce problème, il faudrait associer à chaque agrégat un repère tangent à la surface et alors exprimer les vecteurs de différence dans ce repère. Alors toute manipulation au niveau 0 doit aussi transformer les repères locaux et la reconstruction doit propager ces transformations (figure 5.4 (e)–(f)), ce qui n'est pas une tâche facile. Nous pouvons contourner ce problème en recalculant ces repères tangents sur demande [Adamson et Alexa 2006b], mais en plus des calculs additionnels requis, nous n'avons pas de garantie que, suivant des changements de position, l'orientation des repères tangents restera cohérente avec la transformation subie[8]. Un autre problème de ce type de hiérarchie est que toute transformation d'un niveau $l-1$ induit une transformation linéaire par morceaux au niveau l, ce qui peut être visuellement perceptible et peut produire des fentes ou des auto-intersections (*self intersections*).

5.3.2 Représentations progressives

La particularité de ces représentations est que leur opérateur Υ^{-1} *ajoute* des points au niveau $l-1$ pour produire la base du niveau l, plutôt que de les remplacer. Trois types ont été développés : la représentation de Fleishman *et al.* [2003a], basée sur une construction de bas en haut avec rééchantillonnage (§5.3.2.1), celle de Sing et Narayanna [2006], basée sur le partitionnement (§5.3.2.2), et celle de Linsen et Prautzsch [2001, 2003], basée sur une représentation par éventails de triangles (§5.3.2.3). Nous les décrivons plus en détail dans les sections suivantes.

5.3.2.1 Construction de bas en haut avec rééchantillonnage

La représentation de Fleishman *et al.* [2003a] s'écarte légèrement du schéma de la figure 5.1 au niveau de la décomposition.

[8] Le repère tangent est calculé par analyse de covariance (§2.3.1.2), ce qui produit généralement une direction normale robuste, mais nous ne pouvons pas en dire autant pour l'orientation des autres vecteurs propres.

Décomposition Contrairement au schéma, la décomposition de cette représentation est calculée de bas en haut. Le niveau 0 est directement calculé à partir de l'ensemble de points original. Un niveau l est calculé par l'ajout de points autour de chaque point du niveau $l-1$ en échantillonnant un plan[9]. Chaque point du plan est ensuite projeté sur la surface du niveau $l-1$ et l'originale (le niveau L d'entrée), et la distance entre les deux donne la valeur du détail[10], ce qui définit l'opérateur $\ominus$. Un nouveau niveau est construit tant que l'erreur entre la surface produite et la surface originale dépasse un seuil de tolérance.

Reconstruction La reconstruction suit presque les mêmes étapes que la décomposition. L'opérateur Υ^{-1} correspond à l'ajout des points décrit précédemment, suivi de la projection sur la surface du niveau $l-1$. L'opérateur $\oplus$ ne fait qu'ajouter la valeur du détail de chaque point dans la direction de la projection.

Afin d'éviter un suréchantillonnage, l'étendue d'échantillonnage des plans est déterminée en fonction de la proximité des points voisins. Nous nous retrouvons alors avec le problème de génération *vs.* encodage. D'ailleurs, les auteurs rapportent que leur représentation n'est valable que pour des transformations rigides. Par contre, ils peuvent obtenir de bons taux de compression en quantifiant les valeurs des détails sur peu de bits.

5.3.2.2 Partitionnement

La représentation de Sing et Narayanna [2006] se rapproche beaucoup en principe des ondelettes [Stollnitz *et al.* 1996] où chaque élément d'un niveau correspond essentiellement à un représentant et une valeur de détail.

Décomposition L'opérateur Υ effectue un partitionnement des points au niveau l en deux groupes, de sorte qu'à chaque point du premier groupe est associé un point à proximité dans le second groupe. Le niveau $l-1$ correspond aux points du premier groupe. L'opérateur $\ominus$, comme pour les hiérarchies relatives (§5.3.1), correspond à prendre la différence entre les positions des points de chaque paire. Étant donné que Υ réordonne les points, il faut aussi réordonner en conséquence les détails calculés pour les niveaux supérieurs, de $l+1$ à L.

[9] Ce plan est défini selon l'équation (2.15). Le point lui-même est réutilisé, mais sa position est remplacée par sa projection sur le plan.

[10] Pour être exacte, chaque projection est calculée selon l'équation (2.17), mais en utilisant le même plan (équation (2.15)) ajusté au niveau $l-1$, et la valeur du détail est alors la différence entre les deux polynômes évalués en $(0,0)$. Il s'agit donc d'une "distance" signée.

Reconstruction Les opérateurs de la reconstruction correspondent simplement aux inverses de ceux de la décomposition, sans le réordonnement.

Cette représentation a l'avantage d'être à la base très compacte en mémoire. En effet, sans compression, elle ne prend pas plus de mémoire que l'ensemble de points original. Les auteurs décrivent aussi comment la compresser davantage. Cette représentation a aussi l'avantage de ne requérir aucun calcul de plan local, mais au prix de devoir exprimer les vecteurs de différences en coordonnées globales, ce qui amène les mêmes inconvénients que les hiérarchies avec encodage relatif sans les modifications suggérées (§5.3.1), et donc cette représentation est inappropriée pour l'édition.

5.3.2.3 Ensembles d'éventails de triangles

Une façon de contourner le problème de connectivité implicite dans les ensembles de points est de la précalculer, sans nécessairement reconstruire un maillage complet. Linsen et Prautzsch [2001, 2003] ont exploité cette idée par une représentation avec un ensemble d'éventails de triangles. En plus de l'information de position, à chaque point est associée la liste de ses k voisins les plus près[11], que nous référons plus loin comme la *liste d'éventail.* Avec une telle représentation, ils obtiennent une représentation multirésolution progressive comme suit.

Décomposition La décomposition pour cette représentation procède en supprimant un point à la fois. Le nombre L de niveaux correspond alors au nombre de points retranchés. L'opérateur Υ sélectionne un point à supprimer selon une gamme de critères et met à jour la liste d'éventail de ses voisins après la suppression. L'opérateur $\ominus$ substitue la position du point par la différence entre cette position et une pondération des positions des voisin. Pour permettre l'édition, cette différence est exprimée dans un repère local calculé par moindres carrés.

Reconstruction L'opérateur Υ^{-1} ajoute le point du niveau ayant comme position initiale la même pondération des positions des voisins utilisée à la décomposition, et met à jour la liste d'éventail de ses voisins après l'ajout. L'opérateur $\oplus$ calcule d'abord un repère local puis déplace le point par la valeur du détail dans ce repère.

[11] Un élément de cette liste est simplement un index dans la liste globale des points. Dans leurs travaux, les auteurs donnent un critère d'angle relatif dans la sélection du voisinage (les voisinages-α, décrits à la §2.3.1.1), mais pour les fins de notre présentation, ces détails n'importent pas.

En dehors des problèmes possibles associés à la cohérence de l'orientation des repères après déformation (§5.3.1), l'inconvénient principal de cette représentation est la lourdeur de la structure. Elle revient essentiellement à une structure de maillage, mais avec des sommets de valence généralement un peu plus grande.

5.3.3 Surfaces multiéchelles

Sans doute la façon la plus simple de contourner le problème de génération *vs.* encodage est de conserver le même nombre de points entre les niveaux, similairement à la représentation multirésolution basée sur le lissage de Kobbelt *et al.* [1998] (§5.2.4). C'est ce que font Pauly *et al.* [2002b, 2006] et Zhang *et al.* [2005] avec des représentations très similaires. Elles sont dites *multiéchelles* car d'un niveau à l'autre, l'échelle des détails géométriques est augmentée par lissage[12].

Décomposition Il s'agit de lisser l'ensemble des points du niveau l (opérateur Υ) et d'encoder les positions des points de ce niveau par un déplacement dans l'axe de leur projection (§2.3.3) sur la surface du niveau $l-1$ (opérateur $\ominus$).

Reconstruction Pour ce type de représentation, l'opérateur Υ^{-1} est l'identité, car le nombre de points ne change pas. L'opérateur $\oplus$ ne fait que déplacer chaque point dans l'axe de leur normale par la valeur du détail correspondante.

Puisque les travaux présentés au prochain chapitre se basent en partie sur ces représentations, plus de détails y seront donnés.

Toute forme d'édition décrite à la §5.1 peut être effectuée avec ce type de représentation. Cependant, comme le nombre de points ne change pas, toute opération effectuée au niveau 0 coûtera autant que si elle est effectuée au niveau L (hormis la considération de préservation des détails). De plus, si le nombre de niveaux est grand, la quantité de mémoire requise pour stocker cette représentation devient non négligeable.

[12]Nous employons aussi le terme *multiéchelle* pour distinguer cette approche des autres faisant usage de simplification, comme nous le mentionnons dans notre terminologie à la page xx.

5.4 Autres méthodes de déformation de surfaces avec préservation des détails géométriques

Les représentations multirésolutions font partie d'une plus grande famille de méthodes pour effectuer des déformations de surfaces qui préservent les détails géométriques. Nous invitons le lecteur à consulter l'étude d'ensemble de Botsch et Sorkine [2008] pour une bonne couverture du domaine concernant les maillages.

Parmi ces méthodes, hormis les représentations multirésolutions, les techniques d'édition laplacienne ont été adaptées aux représentations par points [Ohtake *et al.* 2006, Miao *et al.* 2006, Miao *et al.* 2008]. L'édition laplacienne se base sur la représentation différentielle des coordonnées des points, *i.e.* chaque point est exprimé par la différence entre sa position et une pondération de ses voisins (souvent le centroïde). L'information de connectivité est conservée dans une matrice d'adjacence qui, dans le cas des représentations par points, est reconstruite à partir de voisinages de l'ensemble de points original. Les déformations résultent de la résolution d'un système d'équations basé sur cette matrice et d'un ensemble de contraintes, dont les nouvelles positions et la linéarisation de contraintes pour approximer les rotations et mises à l'échelle locales.

Boubekeur *et al.* [2007] ont développé une méthode de transfert de déformations en structures libres (*free-form deformations*) sur des ensembles de points de grande taille. Leur méthode se base sur la reformulation intrinsèque locale de la position des points de l'ensemble à déformer par rapport à une version réduite. Nous reviendrons plus en détail sur cette représentation au chapitre 6 car nous nous basons sur celle-ci pour définir notre représentation multirésolution[13].

À l'exception des travaux de Miao *et al.* [2006, 2008], ces méthodes citées ci-avant sont axées sur la déformation, et non sur l'édition des détails mêmes. Comme nous l'avons présenté à la §5.1 et §5.2, les représentations multirésolutions offrent les deux possibilités à la fois. En ce qui concerne les travaux de Miao *et al.* [2006, 2008], ils parviennent à décomposer les coordonnées différentielles en bandes de fréquences et à les éditer. Cependant, les coordonnées de Laplace correspondent essentiellement à la bande de fréquences enlevées par un lissage laplacien [Taubin 1995, Desbrun *et al.* 1999]. Or, comme ce filtrage doit être appliqué beaucoup de

[13] Puisque la méthode de Boubekeur *et al.* correspond essentiellement à représenter un niveau détaillé relativement à un niveau grossier, nous devrions peut-être plutôt classer cette méthode parmi les représentations multirésolutions de la section précédente, avec la particularité que pour celle-ci, similairement à la représentation de Lee *et al.* [2000] (§5.2), nous avons toujours $L = 1$. Nous avons préféré la décrire ici pour son accent sur la déformation et, comme nous le verrons au chapitre suivant à la §6.3.2.2, nous ne pouvons pas directement la généraliser pour $L > 1$.

fois pour lisser des fréquences plus basses [Pauly *et al.* 2002b], l'étendue des fréquences touchées par l'édition laplacienne est donc plus limitée. Avec les représentations multirésolutions, l'étendue peut être plus grande, et Miao *et al.* [2006, 2008] doivent justement passer par une version simplifiée de la surface pour parvenir à éditer des détails à une échelle plus significative.

Chapitre 6

Surfaces multirésolutions représentées par des ensembles de points

Parmi toutes les représentations multirésolutions énumérées à la §5.3, les seules qui ont toutes les capacités d'édition requises (§5.1) sont les représentations avec éventails de triangles et les représentations multiéchelles. Parmi celles-ci, les plus robustes sont les dernières, *i.e.* elles peuvent généralement donner de bons résultats de reconstruction pour une gamme plus grande de déformations[1]. Nous rappelons que les approches multiéchelles imitent la représentation multirésolution basée sur le lissage de Kobbelt *et al.* [1998] (§5.2.4), *i.e.* chaque niveau contient exactement le même nombre de points (ou sommets, dans le cas des maillages), mais chacun est de plus en plus lisse.

Kobbelt *et al.* ont aussi présenté une représentation basée sur la décimation, où chaque niveau est graduellement simplifié, *i.e.* le nombre de sommets et de faces est réduit. Notre but premier est de complémenter les représentations multiéchelles de Pauly *et al.* [2006] et Zhang *et al.* [2005] par une représentation avec décimation. Cependant, en utilisant les surfaces MCM (moindres carrés mobiles, de *moving least squares* ou *MLS*), fondamentalement lisses, comme le font Pauly *et al.* et Zhang *et al.* et en exploitant l'avantage des représentations par points au niveau de la restructuration et de la consistance (§2.4), nous pouvons aisément unifier les deux aspects, lissage et décimation, en une seule représentation multirésolution. Nous présentons une

[1] Il est à noter que nous ne prétendons pas que les représentations multiéchelles sont toujours plus robustes que les représentations avec éventails de triangles. Nous en discuterons plus en profondeur à la §6.6.

telle représentation dans ce chapitre. Elle a été publiée à la conférence internationale *Graphics Interface 2008* [Duranleau *et al.* 2008].

Nous présentons d'abord les particularités de notation employées dans ce chapitre à la §6.1, avant de donner un aperçu de notre représentation à la §6.2. Puis nous présentons les deux opérations fondamentales, soient la décomposition et la reconstruction, respectivement aux §6.3 et §6.4. Nous présentons des résultats obtenus à la §6.5, puis nous finissons par une discussion sur la représentation et les extensions possibles à la §6.6.

6.1 Notation

Nous notons par $P^{[l]} \subset \mathbb{R}^3$ l'ensemble de points au niveau de résolution l. Chaque ensemble est une séquence ordonnée de points. Le i^{e} point de $P^{[l]}$ est noté $\boldsymbol{p}_i^{[l]}$, et sa normale est notée $\boldsymbol{n}_i^{[l]}$, où $\boldsymbol{n}_i^{[l]} \in \mathbb{S}^2$. Lorsque des fonctions ayant pour domaine un ensemble de points sont définies, P est utilisé comme paramètre générique, et alors le i^{e} point de ce paramètre est noté $\boldsymbol{p}_i$ et la normale associée est $\boldsymbol{n}_i$.

L'ensemble des détails du niveau l est noté par $D^{[l]}$. Il s'agit d'une séquence telle que $|D^{[l]}| = |P^{[l]}|$ et que le i^{e} élément, noté $\boldsymbol{d}_i^{[l]}$, est associé au point $\boldsymbol{p}_i^{[l]}$.

Nous réutilisons grandement les concepts et la notation de la §2.3. Cependant, lors de la décomposition ou la reconstruction d'un niveau de résolution l, puisque la plupart des calculs reliés à une surface (estimation de normale, centroïde pondéré, projection, etc.) est faite par rapport à la surface $\mathcal{S}_{P^{[l-1]}}$, pour alléger la notation, nous omettons l'ensemble en indice. Pour les cas particuliers, l'indice sera présent.

Remarque 6.1 (Illustrations) *Pour réduire la complexité des illustrations, les schémas dans les figures de ce chapitre sont en deux dimensions. Nous illustrons alors chaque point par un trait plein pour indiquer la direction de sa normale associée, i.e. sa normale est perpendiculaire au trait.*

6.2 Aperçu

Nous définissons notre représentation, illustrée à la figure 6.1, selon le même schéma que la figure 5.1. Nous débutons avec un ensemble initial de points $P^{[L]}$. La décomposition génère successivement des ensembles de points $P^{[L-1]}, \ldots, P^{[0]}$ tels que $|P^{[l-1]}| < |P^{[l]}|, \forall l \in \mathbb{Z}[1, L]$. Pour générer $P^{[l-1]}$ à partir de $P^{[l]}$, un lissage, noté Φ, est d'abord effectué sur $P^{[l]}$, suivi d'une

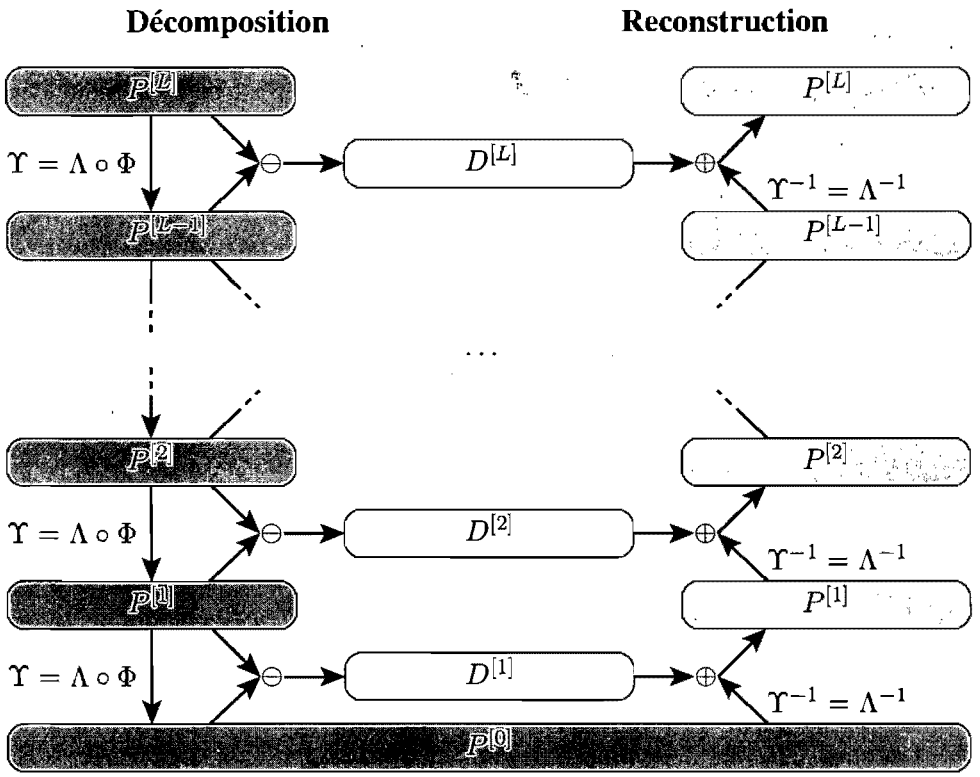


FIG. 6.1 – Opérations de notre représentation multirésolution.

simplification, notée Λ. Ainsi notre opérateur Υ correspond à $\Lambda \circ \Phi$. Les détails $D^{[l]}$ regroupent deux types d'informations, géométrique et de connectivité, et sont calculés comme suit. Chaque point $\boldsymbol{p}_i^{[l]}$ est d'abord projeté sur la surface $\mathcal{S}_{P^{[l-1]}}$. La distance signée (selon la direction de projection et celle de la normale) est la valeur du détail géométrique. Le point projeté est ensuite reformulé de façon intrinsèque dans $P^{[l-1]}$ par une méthode inspirée de celle de Boubekeur *et al.* [2007]. Pour chaque élément $\boldsymbol{d}_i^{[l]} \in D^{[l]}$, nous conservons la distance signée (le détail géométrique) et la reformulation (détail de connectivité). Les équations (5.1) et (5.2) deviennent alors

$$P^{[l-1]} = \Lambda\big(\Phi(P^{[l]})\big) \tag{6.1}$$

$$D^{[l]} = P^{[l]} \ominus P^{[l-1]}. \tag{6.2}$$

Au final, seulement $P^{[0]}$ et $D^{[1]}, \ldots, D^{[L]}$ sont conservés.

Pour reconstruire $P^{[l]}$ à partir de $P^{[l-1]}$ et $D^{[l]}$, il faut d'abord inverser la simplification en reproduisant les points reformulés intrinsèquement en utilisant les détails de connectivité dans $D^{[l]}$. Nous notons cette opération Λ^{-1}. Bien qu'il ne s'agit pas exactement de l'inverse

de Λ [2], le même nombre de points sera produit. Ensuite, la normale de chaque point reproduit est calculée, puis le point est déplacé dans cette direction par la valeur du détail géométrique. Ainsi, l'équation (5.3) devient

$$P^{[l]} = \Lambda^{-1}(P^{[l-1]}) \oplus D^{[l]}. \tag{6.3}$$

Les détails de chaque opération sont donnés dans les prochaines sections, où nous parlons seulement en termes de décomposition ou reconstruction d'un niveau. Il suffit de les répéter pour obtenir la séquence complète.

6.3 Décomposition

Une étape de la décomposition consiste à prendre $P^{[l]}$ et produire $P^{[l-1]}$ (équation (6.1)) puis $D^{[l]}$ (équation (6.2)). Ces sous-étapes sont décrites respectivement à la §6.3.1 puis à la §6.3.2. Ainsi, pour une décomposition de L niveaux, nous devrons les répéter L fois, *i.e.* pour $l = L, L-1, \ldots, 1$.

6.3.1 Génération du niveau grossier

Le calcul de $P^{[l-1]}$ est effectué en deux étapes : un opérateur de lissage Φ réduit d'abord les hautes fréquences dans les détails géométriques, puis un opérateur de simplification Λ produit un ensemble de points de cardinalité moindre représentatif de la surface lisse résultant du lissage. Nous pourrions seulement appliquer Λ, car en soi la simplification implique une réduction de hautes fréquences. Cependant, en choisissant le taux de simplification de Λ conséquemment avec le lissage de Φ, nous obtenons un contrôle plus intuitif (*i.e.* par le lissage) sur la réduction des hautes fréquences.

Comme Pauly *et al.* [2006] et Zhang *et al.* [2005] (§5.3.3), nous construisons une surface MCM multirésolution. Or, les surfaces MCM sont intrinsèquement lisses, et le taux de lissage est contrôlé par la taille du support de la fonction de pondération θ (§2.3.3.2). Donc, comme Pauly *et al.* et Zhang *et al.*, un choix naturel pour Φ est la projection MCM, *i.e.*

$$\Phi(P^{[l]}) = \psi_{P^{[l]}}(P^{[l]}) = \{\psi_{P^{[l]}}(\boldsymbol{p}) \mid \boldsymbol{p} \in P^{[l]}\}. \tag{6.4}$$

[2]Une des raisons est la dépendance sur une partie de l'information de $D^{[l]}$. À noter que nous omettons cette dépendance dans la notation de l'opérateur Λ^{-1} pour alléger la notation. Nous verrons également à la §6.4 pourquoi il n'est pas nécessaire de s'en encombrer.

Dans le cas de Pauly *et al.* et Zhang *et al.*, puisque $|P^{[l-1]}| = |P^{[l]}|$, pour produire des niveaux de plus en plus lisses, la taille du support de la fonction de pondération est augmentée pour la projection d'un niveau à l'autre. Cependant, en faisant ceci, le nombre de points avec une pondération non nulle autour d'un point $\boldsymbol{x}$ croît aussi, augmentant le temps de calcul de $\psi_{P^{[l]}}(\boldsymbol{x})$. Pour éviter ce problème, une technique d'intégration combinée à une simplification de $P^{[l]}$ peut être employée. Notons par $\tilde{\Lambda}$ l'opérateur de cette simplification. Pauly *et al.*[3] utilisent une simple technique hiérarchique[4], où le support de la fonction de pondération θ est celui d'un voisinage-k (équation (2.49)). En fixant k, la taille du support est alors augmentée en simplifiant $P^{[l]}$ avec un taux de plus en plus grand. Leur choix pour $\tilde{\Lambda}$ est une mise en agrégats hiérarchiques [Pauly *et al.* 2002a], où chaque agrégat est substitué par son centroïde. Cette méthode peut être plus formellement décrite récursivement ainsi :

$$\tilde{\Lambda}(P) = \begin{cases} \{\bar{\boldsymbol{p}}\} & \text{si} \quad |P| \leq \gamma \\ \tilde{\Lambda}\big(\{\boldsymbol{p} \in P \mid d_P(\boldsymbol{p}) < 0\}\big) \cup \tilde{\Lambda}\big(\{\boldsymbol{p} \in P \mid d_P(\boldsymbol{p}) \geq 0\}\big) & \text{si} \quad |P| > \gamma \end{cases} \tag{6.5}$$

où

$$d_P(\boldsymbol{p}) = \boldsymbol{s}_P{}^{\mathsf{T}}(\boldsymbol{p} - \bar{\boldsymbol{p}}) \tag{6.6}$$

$$\boldsymbol{s}_P = \arg\max_{\boldsymbol{s} \in \mathbb{S}^2} \boldsymbol{s}^{\mathsf{T}} \left(\sum_{i=1}^{|P|} (\boldsymbol{p}_i - \bar{\boldsymbol{p}})(\boldsymbol{p}_i - \bar{\boldsymbol{p}})^{\mathsf{T}} \right) \boldsymbol{s} \tag{6.7}$$

$$\bar{\boldsymbol{p}} = \frac{1}{|P|} \sum_{i=1}^{|P|} \boldsymbol{p}_i \tag{6.8}$$

et $\gamma \in \mathbb{N}$ est la taille maximale des agrégats. L'équation (6.7) calcule l'axe de plus grande variation de P et l'équation (6.8), le centroïde de P. La méthode de simplification procède donc en partitionnant récursivement un ensemble de points P par rapport au plan passant par le centroïde $\bar{\boldsymbol{p}}$ et perpendiculaire à $\boldsymbol{s}_P$. Dans le cas de Pauly *et al.* [2006], $\tilde{\Lambda}$ est paramétré en fonction de l et γ est substitué par γ^{L-l+1}.

Dans notre cas, $|P^{[l-1]}| < |P^{[l]}|$, et donc nous pourrions être tentés de simplement choisir $\Phi(P^{[l]}) = \psi_{P^{[l]}}(P^{[l]})$. Cependant, pour obtenir un lissage significatif entre $P^{[l]}$ et $P^{[l-1]}$, nous devrions choisir k assez grand. Ainsi, nous effectuons tout de même une simplification comme Pauly *et al.*, *i.e.* $\Phi(P^{[l]}) = \psi_{\tilde{\Lambda}(P^{[l]})}(P^{[l]})$, mais avec un taux de simplification constant (nous

[3]Zhang *et al.* ne mentionnent aucun détail. Ils ne font qu'évoquer l'idée générale.

[4]Pour ne pas dire multirésolution.

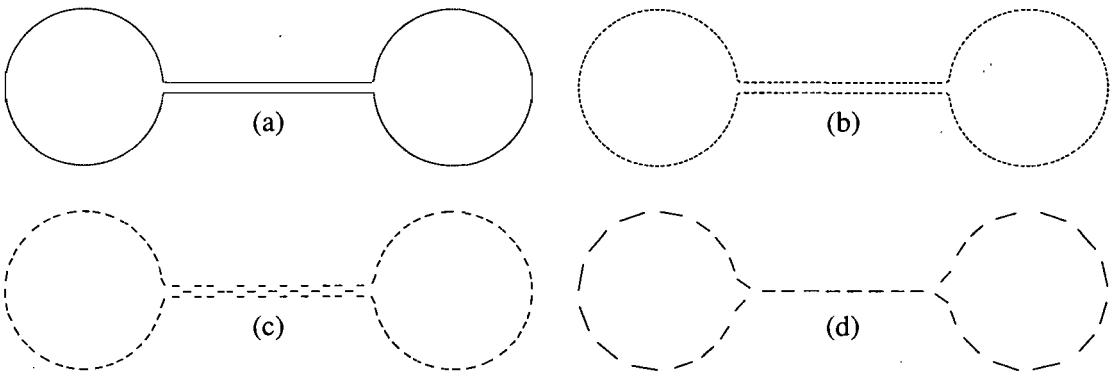


FIG. 6.2 – Dégénérescence pour une série de simplifications sans contrainte. Cet exemple a été produit par mise en agrégats hiérarchique [Pauly *et al.* 2002a] avec une taille cible pour les agrégats de trois points. (a) Ensemble de points P original. (b) $\tilde{\Lambda}(P)$. (c) $\tilde{\Lambda}(\tilde{\Lambda}(P))$. (d) $\tilde{\Lambda}(\tilde{\Lambda}(\tilde{\Lambda}(P)))$. Sans contrainte, les agrégats finissent par se former avec des points de chaque côté de la zone centrale, faisant en sorte que le tube se transforme en ligne, alors qu'aucun problème ne survient encore aux deux extrémités.

choisissons typiquement $\gamma = 4$ ou 5). Ceci nous indique en même temps le choix pour Λ. Puisque $\mathcal{S}_{\tilde{\Lambda}(P^{[l]})}$ est la surface lisse produite par Φ, alors l'ensemble de points $\tilde{\Lambda}(P^{[l]})$ est un candidat pour $P^{[l-1]}$. Cependant, nous pouvons obtenir une meilleure distribution de points en choisissant plutôt $P^{[l-1]} = \tilde{\Lambda}\big(\Phi(P^{[l]})\big)$. En effet, en simplifiant l'ensemble de points lissé, la distribution des points ne tient pas compte des hautes fréquences de $P^{[l]}$, contrairement au premier choix[5]. Ainsi, $\Lambda = \tilde{\Lambda}$, où, comme Pauly *et al.*, nous avons choisi $\tilde{\Lambda}$ de l'équation (6.5). Le calcul de $P^{[l-1]}$ est donc le suivant :

$$P^{[l-1]} = (\Lambda \circ \Phi)(P^{[l]}) = \Lambda\big(\Phi(P^{[l]})\big) = \Lambda\big(\psi_{\Lambda(P^{[l]})}(P^{[l]})\big). \tag{6.9}$$

Puisque $P^{[l-1]}$ vient indirectement de l'application consécutive de $L-l+1$ simplifications de $P^{[L]}$, *i.e.*

$$P^{[l-1]} = (\Lambda \circ \Phi)(P^{[l]}) = \big((\Lambda \circ \Phi) \circ (\Lambda \circ \Phi)\big)(P^{[l+1]}) = \big(\underbrace{(\Lambda \circ \Phi) \circ \cdots \circ (\Lambda \circ \Phi)}_{(L-l+1) \text{ fois}}\big)(P^{[L]}) \tag{6.10}$$

[5] Il est à noter cependant que $\mathcal{S}_{\tilde{\Lambda}(\Phi(P^{[l]}))}$ n'est pas identique à $\mathcal{S}_{\tilde{\Lambda}(P^{[l]})}$. La première surface est un peu plus lisse que la seconde, mais le lissage de Φ demeure dominant. Pauly *et al.* [2006] mentionnent que le lissage supplémentaire est en fait bénéfique car il réduit les artefacts de repliement (*aliasing artifacts*).

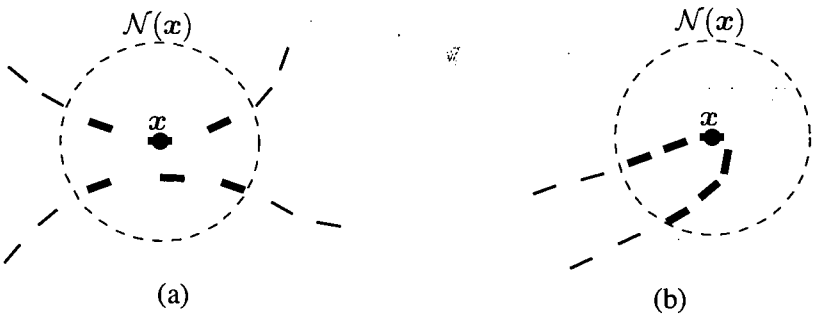


FIG. 6.3 – Cas typiques de voisinages-k posant problème pour le calcul de $\boldsymbol{n}(\boldsymbol{x})$ ou pour la simplification : (a) deux surfaces à proximité et (b) une crête fine. Pour ces exemples, $k = 6$, et $\boldsymbol{x}$ fait partie des points de la surface.

certains problèmes peuvent survenir. D'abord, les formes allongées peuvent finir par perdre complètement leur volume. Un exemple est illustré à la figure 6.2. De plus, comme nous utilisons $\psi(\boldsymbol{x})$ et $\boldsymbol{n}(\boldsymbol{x})$ (équations (2.20) ou (2.23), et (2.27), §2.3.3.1) dans des étapes subséquentes, certains problèmes peuvent survenir si nous retrouvons des cas tels qu'illustrés à la figure 6.3 après l'application de Λ. En effet, $\boldsymbol{n}(\boldsymbol{x})$ ne pointera pas nécessairement dans la direction attendue, mais possiblement dans une direction plus tangentielle. Tous ces problèmes sont reliés à la distribution des points d'un voisinage-k, où k varie habituellement entre 10 et 20 (15 dans nos expérimentations).

Pauly *et al.* [2002a] ont suggéré une variante adaptative de la mise en agrégats de l'équation (6.5) : la condition d'arrêt de la récursion devient $|P| = 1 \vee (|P| < \gamma \wedge \sigma_P(\bar{\boldsymbol{p}}) < \tau)$, *i.e.* si la variation de surface (équation (2.12)) de P dépasse un certain seuil donné τ, P est alors aussi subdivisé, pourvu qu'il le puisse (d'où $|P| = 1$ dans la condition). Le problème avec cette variante par rapport aux problèmes énumérés auparavant est qu'elle considère seulement le contenu d'un agrégat pour décider si une subdivision est nécessaire. Or, $\gamma < k$, et donc le support de toute analyse d'un agrégat n'est pas assez grand.

Nous faisons alors usage d'heuristiques évaluées pour chaque voisinage-k de l'ensemble de points simplifié, et si certains seuils ne sont pas atteints, nous raffinons davantage par subdivision les agrégats. Pour décrire notre méthode, nous reformulons d'abord la mise en agrégats ainsi :

$$\breve{\Lambda}(P) = \begin{cases} \{P\} & \text{si } |P| \leq \gamma \\ \breve{\Lambda}(\{\boldsymbol{p} \in P \mid d_P(\boldsymbol{p}) < 0\}) \cup \breve{\Lambda}(\{\boldsymbol{p} \in P \mid d_P(\boldsymbol{p}) \geq 0\}) & \text{si } |P| > \gamma. \end{cases} \tag{6.11}$$

Au lieu de produire un ensemble de points, nous produisons un ensemble d'agrégats, eux-mêmes des ensembles de points. Dans ce qui suit, si C est un tel ensemble, alors C_i dénote son i^{e} agrégat. Les agrégats C forment une partition de P, *i.e.* $P = \bigcup_{i=1}^{|C|} C_i$ et $\forall i \neq j \in \mathbb{Z}[1, |C|]\ C_i \cap C_j = \emptyset$. L'ensemble de points est alors l'ensemble des centroïdes de chaque agrégat, que nous notons en ajoutant une barre au-dessus de l'expression décrivant l'ensemble d'agrégats. Ainsi, pour C, l'ensemble de points est

$$\bar{C} = \bigcup_{i=1}^{|C|} \{\bar{\boldsymbol{c}}_i\} \tag{6.12}$$

où $\bar{\boldsymbol{c}}_i$ est le centroïde de l'agrégat C_i (équation (6.8)). La normale associée à $\bar{\boldsymbol{c}}_i$, notée $\breve{\boldsymbol{n}}_i$, est calculée par la moyenne normalisée des normales de chaque point de C_i, *i.e.*

$$\breve{\boldsymbol{n}}_i = \frac{\sum_{i \in \{j \in \mathbb{N} \mid \boldsymbol{p}_j \in C_i\}} \boldsymbol{n}_i}{\left\| \sum_{i \in \{j \in \mathbb{N} \mid \boldsymbol{p}_j \in C_i\}} \boldsymbol{n}_i \right\|}. \tag{6.13}$$

Enfin, le raffinement d'un ensemble d'agrégats C est calculé par la fonction récursive suivante :

$$\Xi(C) = \begin{cases} C & \text{si} \quad |C| = \left| \bigcup_{i=1}^{|C|} S_i \right| \\ \Xi\left(\bigcup_{i=1}^{|C|} S_i \right) & \text{sinon} \end{cases} \tag{6.14}$$

où

$$S_i = \begin{cases} \{C_i\} & \text{si } |C_i| = 1 \vee v_i \\ \left\{ \{\boldsymbol{p} \in C_i \mid d_{C_i}(\boldsymbol{p}) < 0\}, \{\boldsymbol{p} \in C_i \mid d_{C_i}(\boldsymbol{p}) \geq 0\} \right\} & \text{sinon} \end{cases} \tag{6.15}$$

$$v_i = \left(\min_{j \in \{k \in \mathbb{N} \mid \bar{\boldsymbol{c}}_k \in \mathcal{N}_{\bar{C}}(\bar{\boldsymbol{c}}_i)\}} \breve{\boldsymbol{n}}_i^{\mathsf{T}} \breve{\boldsymbol{n}}_j \right) > \tau \ \wedge \ \frac{\lambda_1}{\lambda_2} < \varsigma. \tag{6.16}$$

Dans l'équation (6.16), $\lambda_1 < \lambda_2$ sont respectivement les deux plus petites valeurs propres de la matrice décrite à l'équation (2.22) (où $P = \bar{C}$). Les scalaires τ et ς sont des seuils expliqués un peu plus loin. Exprimé autrement, le raffinement $\Xi(C)$ subdivise les agrégats de C tant que, pour tout $i \in \mathbb{Z}\,[1, |C|]$, $\mathcal{N}_{\bar{C}}(\bar{\boldsymbol{c}}_i)$ respecte certains critères, évalués par v_i. La subdivision du i^{e} agrégat est calculée par S_i (équation (6.15)).

Le critère de subdivision (équation (6.16)) se base sur deux évaluations. L'une teste l'écart de la normale de chaque point dans $\mathcal{N}_{\bar{C}}(\bar{\boldsymbol{c}}_i)$ par rapport à $\breve{\boldsymbol{n}}_i$, la normale de $\bar{\boldsymbol{c}}_i$. Cet écart est mesuré par le minimum des produits scalaires entre chaque normales et $\bar{\boldsymbol{c}}_i$. Si ce minimum dépasse

un seuil τ donné (nous utilisons typiquement une valeur près de zéro), nous considérons ce critère atteint. Ce critère vise principalement à contrôler le problème illustré à la figure 6.3 (a).

La seconde évaluation s'inspire de l'estimation de variation de surface et concerne davantage le problème des crêtes illustré à la figure 6.3 (b). Cependant, en considérant le ratio $\frac{\lambda_1}{\lambda_2}$ plutôt que celui de l'équation (2.12), nous obtenons une estimation de *variation* plus sensible aux crêtes. Nous considérons le critère atteint si cette estimation est sous un seuil ς donné (nous employons $\varsigma = 0.5$). Si nous utilisions la variation de surface originale, le seuil requis pour bien détecter les crêtes ferait en sorte que le critère serait trop sévère pour les zones courbes sans crêtes.

Remarque 6.2 (Évaluation du raffinement) *D'un point de vue implémentation, évaluer l'équation* (6.14) *naïvement est très coûteux, car nous risquons d'évaluer l'équation* (6.15) *à répétition pour des agrégats dans des régions non problématiques. Ainsi, nous traitons seulement les agrégats qui ont été subdivisés à l'itération précédente, et tous les agrégats à la première itération.*

Au final, notre opérateur Λ est défini ainsi :

$$\Lambda(P) = \overline{\Xi\big(\breve{\Lambda}(P)\big)}. \tag{6.17}$$

Puisque que $P^{[l-1]} = \Lambda(\Phi(P^{[l]})) = \overline{\Xi\big(\breve{\Lambda}(\Phi P^{[l]})\big)}$, au lieu de calculer la normale $\boldsymbol{n}_i^{[l-1]}$ de chaque point $\boldsymbol{p}_i^{[l-1]} \in P^{[l-1]}$ à partir de $P^{[l-1]}$, nous pouvons plutôt lui assigner la normale $\breve{\boldsymbol{n}}_i$ du i^{e} agrégat de $\Xi\big(\breve{\Lambda}(\Phi P^{[l]})\big)$, ce qui nous permet de sauver une étape de calcul supplémentaire, puisque cette dernière doit nécessairement être calculée.

Il est important de noter que le cœur de notre représentation (§6.3.2 et §6.4) ne dépend pas d'une technique particulière de lissage ou de simplification et nous pourrions utiliser une grande variété de techniques pour le lissage [Pauly et Gross 2001, Clarenz *et al.* 2004a, Clarenz *et al.* 2004b, Dey *et al.* 2004b, Lange et Polthier 2005, Schall *et al.* 2005, Zhang *et al.* 2005, Hu *et al.* 2006, Pauly *et al.* 2006, Xiao *et al.* 2006, Moenning *et al.* 2007] ou pour la simplification [Cohen *et al.* 2001, Pauly *et al.* 2002a, Fleishman *et al.* 2003a, Linsen 2001, Linsen et Prautzsch 2003, Moenning et Dodgson 2003, Moenning et Dodgson 2004, Wu et Kobbelt 2004, Boubekeur *et al.* 2006, Kim *et al.* 2006, Reniers et Telea 2006, Singh et Narayanan 2006, Yu et san Wong 2006, Wang *et al.* 2007]. Pour la simplification, la seule contrainte est la possibilité d'adaptation pour des conditions *a posteriori* sur l'échantillonnage résultant

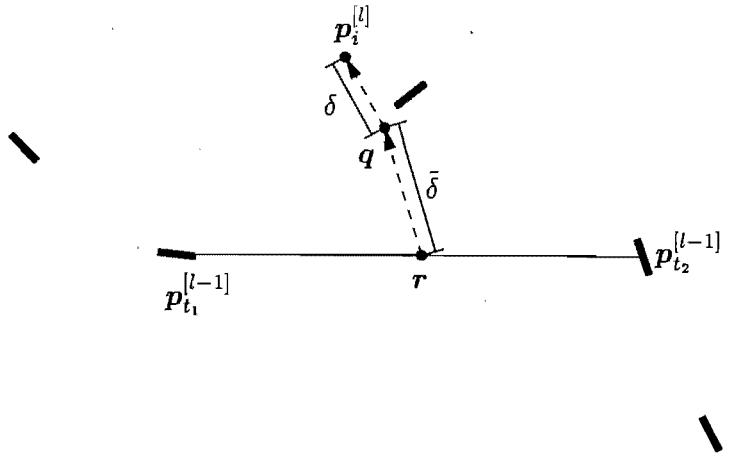


FIG. 6.4 – Calcul et représentation des détails : un point $p_i^{[l]} \in P^{[l]}$ est projeté sur $\mathcal{S}_{P^{[l-1]}}$ (le point q), donnant la valeur de détail δ ; q est ensuite reformulé sur un triangle de base $\triangle p_{t_1}^{[l-1]} p_{t_2}^{[l-1]} p_{t_3}^{[l-1]}$ (ici représenté par un segment $\overline{p_{t_1}^{[l-1]} p_{t_2}^{[l-1]}}$) tel que $q = r + \tilde{\delta} n(r)$; r est exprimé en terme de coordonnées barycentriques dans le triangle.

de la surface. Nos choix pour Φ et Λ ont été guidés par l'expérience de Pauly *et al.* [2006], la réutilisation des concepts déjà employés (projection MCM), et la simplicité des méthodes.

6.3.2 Extraction des détails

Suite à la génération du niveau grossier, nous avons en main $P^{[l]}$ et $P^{[l-1]}$. Il nous reste à exprimer $P^{[l]}$ relativement à $P^{[l-1]}$. Ceci revient à formuler un opérateur $\ominus$ pour calculer $D^{[l]}$ tel que $D^{[l]} = P^{[l]} \ominus P^{[l-1]}$ (équation (6.2)), de sorte que plus tard nous puissions retrouver $P^{[l]}$ en *additionnant* $D^{[l]}$ à $P^{[l-1]}$, si $P^{[l-1]}$ n'a pas changé. Autrement, les points $P^{[l]}$ retrouvés devraient globalement exprimer les mêmes changements que ceux effectués sur $P^{[l-1]}$.

À la base, les détails géométriques correspondent à la *différence* entre $\mathcal{S}_{P^{[l]}}$ et $\mathcal{S}_{P^{[l-1]}}$. Elle peut être exprimée par la distance signée entre chaque point de $P^{[l]}$ et leur projection sur $\mathcal{S}_{P^{[l-1]}}$, notée Q. Ceci équivaut à la notion de détails géométriques de Pauly *et al.* [2006] et Zhang *et al.* [2005]. Cette différence correspond approximativement à la bande de fréquences des détails géométriques [Pauly *et al.* 2006] entre les niveaux $l-1$ et l.

Comme toute représentation multirésolution par points (§5.3), notre problème est d'encoder les détails géométriques de façon cohérente avec l'opérateur d'échantillonnage durant la reconstruction (§6.4). Si cet opérateur correspondait à une méthode d'enrichissement d'échantillonnage (*upsampling*) basée sur les voisinages implicites [Alexa *et al.* 2003, Fleishman

et al. 2003a, Guennebaud *et al.* 2004b, Guennebaud *et al.* 2005, Moenning *et al.* 2007], nous ne pourrions pas garantir que le nombre de points générés serait le même que le nombre d'éléments dans l'ensemble des détails géométriques. Même si c'était le cas, il n'y aurait pas nécessairement correspondance entre chaque point généré et chaque élément de détail si la distribution des points générés n'est pas la même que les points $P^{[l]}$ originaux, particulièrement si $P^{[l-1]}$ a changé. Autrement dit, Q ne peut pas être implicitement reproduit par ce type d'opérateur. Les détails géométriques doivent donc être accompagnés d'informations supplémentaires, que nous appelons détails de connectivité, pour exprimer Q par rapport à $P^{[l-1]}$. Sans aller à l'extrême comme le font Linsen et Prautzsch [2003], ce qui requerrait le stockage de k indices et trois scalaires (§5.3.2.3), nous pouvons nous baser sur la reformulation intrinsèque de Boubekeur *et al.* [2007].

Pour chaque $\boldsymbol{p}_i^{[l]} \in P^{[l]}$, nous calculons $\boldsymbol{d}_i^{[l]}$ correspondant par la procédure suivante, illustrée en deux dimensions à la figure 6.4.

1. Calculer la projection $\boldsymbol{q}$ de $\boldsymbol{p}_i^{[l]}$ sur $\mathcal{S}_{P^{[l-1]}}$. La valeur de détail géométrique δ est la distance signée (selon l'orientation de la surface) entre $\boldsymbol{q}$ et $\boldsymbol{p}_i^{[l]}$.
2. Reformuler $\boldsymbol{q}$ *intrinsèquement* dans $P^{[l-1]}$. Ceci se fait en deux étapes :
   a. trouver un ensemble de trois points $\{\boldsymbol{p}_{t_1}^{[l-1]}, \boldsymbol{p}_{t_2}^{[l-1]}, \boldsymbol{p}_{t_3}^{[l-1]}\} \subset P^{[l-1]}$ formant un triangle $\mathcal{T}$ qui *circonscrit* $\boldsymbol{q}$;
   b. calculer un point $\boldsymbol{r}$ sur $\mathcal{T}$ par une projection non orthogonale de $\boldsymbol{q}$ et calculer ses coordonnées barycentriques $(\beta_1, \beta_2, \beta_3)$ dans $\mathcal{T}$.
3. Fixer
$$\boldsymbol{d}_i^{[l]} = (t_1, t_2, t_3, \beta_2, \beta_3, \delta). \tag{6.18}$$
Il n'est pas nécessaire de stocker β_1 car $\beta_1 = 1 - \beta_2 - \beta_3$. Nous aurions pu choisir de stocker une autre paire de coordonnées barycentriques ; le choix est totalement arbitraire. Optionnellement, une septième valeur peut être ajoutée au nuplet (*tuple*) $\boldsymbol{d}_i^{[l]} : \bar{\delta}$, la distance signée entre $\boldsymbol{q}$ et $\boldsymbol{r}$. Aussi optionnellement, au lieu de stocker δ, nous pouvons stocker $\frac{\delta}{\Delta_{\mathcal{T}}}$, où $\Delta_{\mathcal{T}}$ est l'aire du triangle. La raison et les détails de ces options sont expliqués à la §6.4.

Finalement, $D^{[l]} = P^{[l]} \ominus P^{[l-1]} = \left\{\boldsymbol{d}_i^{[l]}\right\}_{i=1}^{|P^{[l]}|}$. Les étapes 1, 2a et 2b de la procédure ci-avant sont détaillées dans les sections suivantes.

6.3.2.1 Différence

L'information fondamentale de détail géométrique est la différence entre chaque point de $P^{[l]}$ et $\mathcal{S}_{P^{[l-1]}}$. Pour un point $\boldsymbol{p}_i^{[l]} \in P^{[l]}$, la valeur de cette information δ est calculée ainsi :

$$\delta = \boldsymbol{n}(\boldsymbol{q})^{\mathsf{T}}(\boldsymbol{p}_i^{[l]} - \boldsymbol{q}) \tag{6.19}$$

où

$$\boldsymbol{q} = \psi(\boldsymbol{p}_i^{[l]}). \tag{6.20}$$

Idéalement, pour que δ soit vraiment représentative de la différence, ψ devrait être orthogonale, et $\boldsymbol{n}(\boldsymbol{q})$ devrait correspondre à la véritable normale de $\mathcal{S}_{P^{[l-1]}}$. Ceci est possible en utilisant la procédure de projection orthogonale de l'équation (2.29) et en remplaçant $\boldsymbol{n}(\boldsymbol{q})$ par le gradient de $\mathcal{S}_{P^{[l-1]}}$ en $\boldsymbol{q}$ [Alexa et Adamson 2004]. Cependant, selon notre expérience, le gain de précision n'est pas suffisamment significatif comparativement au temps de calcul supplémentaire requis.

Bien que nous favorisons l'usage de la moyenne pondérée pour le calcul de $\boldsymbol{n}(\boldsymbol{x})$ en un point $\boldsymbol{x}$ donné (remarque 2.2), si le calcul était plutôt fait par analyse de covariance, nous devrions faire attention pour maintenir la cohérence du signe de δ entre points voisins. Si nous ne voulions pas utiliser des approches globales comme celle de Hoppe *et al.* [1992] (§2.3.1.2), nous pourrions tout de même maintenir l'information de la normale à chaque point et déterminer l'orientation de $\boldsymbol{n}(\boldsymbol{x})$ en fonction de celle de la normale du point le plus près de $\boldsymbol{x}$. Il en irait de même dans les prochaines sections où un calcul dépend de $\boldsymbol{n}(\boldsymbol{x})$.

6.3.2.2 Sélection du triangle

Pour être en mesure de reformuler $\boldsymbol{q}$ (équation (6.20)) *intrinsèquement* dans $P^{[l-1]}$, nous cherchons un domaine local au-dessus duquel nous pouvons exprimer $\boldsymbol{q}$. Comme nous nous inspirons de la reformulation de Boubekeur *et al.* [2007], nous commençons par la décrire.

Leur domaine de base est un plan $\mathcal{H}$ perpendiculaire à $\boldsymbol{n}(\boldsymbol{q})$ et passant par $\boldsymbol{a}(\boldsymbol{q})$ (équation (2.19)). Ensuite, un ensemble de trois points $\{\boldsymbol{p}_{t_1}^{[l-1]}, \boldsymbol{p}_{t_2}^{[l-1]}, \boldsymbol{p}_{t_3}^{[l-1]}\} \subset P^{[l-1]}$ voisins de $\boldsymbol{q}$

est déterminé, où $t_1, t_2, t_3 \in \mathbb{Z}[1, |P^{[l-1]}|]$. Le point $\boldsymbol{q}$ est alors reformulé ainsi :

$$\boldsymbol{q} = \tilde{\delta}\boldsymbol{n}(\boldsymbol{q}) + \boldsymbol{r} \quad \text{où} \quad \boldsymbol{r} = \sum_{j=1}^{3} \beta_j \check{\boldsymbol{p}}_{t_j}^{[l-1]} \tag{6.21}$$

et où $\tilde{\delta}$ est la distance signée entre $\boldsymbol{q}$ et $\boldsymbol{r}$, la projection de $\boldsymbol{q}$ sur $\mathcal{H}$, $\check{\boldsymbol{p}}_{t_j}^{[l-1]}$ est la projection de $\boldsymbol{p}_{t_j}^{[l-1]}$ sur $\mathcal{H}$, $j \in \mathbb{Z}[1,3]$, et $(\beta_1, \beta_2, \beta_3)$ sont les coordonnées barycentriques de $\boldsymbol{r}$ dans le triangle $\triangle\check{\boldsymbol{p}}_{t_1}^{[l-1]}\check{\boldsymbol{p}}_{t_2}^{[l-1]}\check{\boldsymbol{p}}_{t_3}^{[l-1]}$. Nous décrirons plus loin comment $\triangle\boldsymbol{p}_{t_1}^{[l-1]}\boldsymbol{p}_{t_2}^{[l-1]}\boldsymbol{p}_{t_3}^{[l-1]}$ est choisi.

Avec cette reformulation, en conservant seulement t_1, t_2, t_3, β_2, β_3 et $\tilde{\delta}$, si $P^{[l-1]}$ a été transformé en un ensemble de points $P'^{[l-1]}$, par exemple suite à une déformation en structures libres, alors un point $\boldsymbol{q}'$ peut être reconstruit en substituant les points de $P^{[l]}$ par ceux de $P'^{[l-1]}$ dans l'équation (6.21). Cependant, les points $\{\check{\boldsymbol{p}}_{t_j}'^{[l-1]}\}_{j=1}^{3}$ dépendent de $\mathcal{H}'$, qui correspond au plan $\mathcal{H}$ suite à la transformation. Or, pour calculer $\mathcal{H}'$, il faut connaître $\boldsymbol{q}'$, qui est justement le point à reconstruire, et donc inconnu. Pour se tirer d'affaire, une solution est d'utiliser plutôt $\{\boldsymbol{p}_{t_j}'^{[l-1]}\}_{j=1}^{3}$ pour calculer $\boldsymbol{r}'$, et $\boldsymbol{n}(\boldsymbol{q}')$ est substitué par $\boldsymbol{n}(\boldsymbol{r}')$. Mais ceci fait en sorte que généralement, $\boldsymbol{q}' \neq \boldsymbol{q}$ si $P'^{[l-1]} = P^{[l-1]}$.

Boubekeur *et al.* [2007] font usage de cette reformulation pour le transfert de déformation entre un ensemble de points de très grande cardinalité et une version grandement simplifiée de cet ensemble. Ainsi, le ratio entre $\tilde{\delta}$ et l'aire du triangle sera généralement petit, et donc l'erreur de reconstruction n'aura pas un grand impact. Cependant, dans notre cas, le taux de simplification entre chaque niveau de résolution est beaucoup moindre. De plus, nous avons plus d'un niveau, et donc l'erreur peut se retrouver davantage amplifiée. Nous ne pouvons donc pas directement utiliser leur reformulation.

Dans notre cas, au lieu de reformuler $\boldsymbol{q}$ en termes d'une élévation au-dessus d'un plan et d'un triangle projeté sur ce même plan, nous utilisons directement le triangle et son plan de support. Soit $\mathcal{T}$ le triangle ayant pour sommets l'ensemble de trois points $\{\boldsymbol{p}_{t_1}^{[l-1]}, \boldsymbol{p}_{t_2}^{[l-1]}, \boldsymbol{p}_{t_3}^{[l-1]}\} \subset P^{[l-1]}$ voisins de $\boldsymbol{q}$. L'équation (6.21) devient alors

$$\boldsymbol{q} = \tilde{\delta}\boldsymbol{n}_{\mathcal{T}} + \boldsymbol{r} \quad \text{où} \quad \boldsymbol{r} = \sum_{j=1}^{3} \beta_j \boldsymbol{p}_{t_j}^{[l-1]} \tag{6.22}$$

et où $\boldsymbol{n}_{\mathcal{T}}$ est la normale de $\mathcal{T}$ et $(\beta_1, \beta_2, \beta_3)$ sont les coordonnées barycentriques de $\boldsymbol{r}$ dans $\mathcal{T}$, la projection de $\boldsymbol{q}$ sur le plan de support de $\mathcal{T}$. Nous disons que $\mathcal{T}$ *circonscrit* $\boldsymbol{q}$ si chaque coordonnée barycentrique est dans l'intervalle $[0, 1]$. Il y a des inconvénients à projeter orthogo-

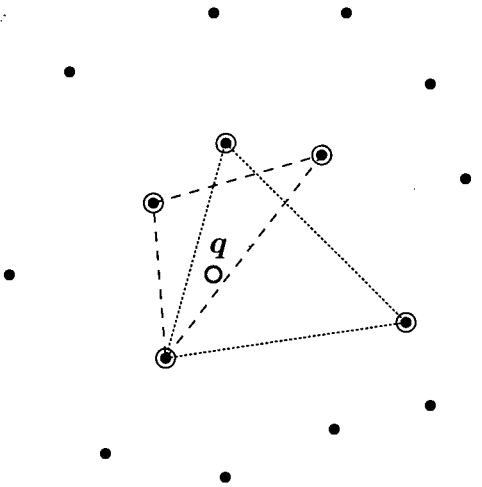


FIG. 6.5 – Ambiguïté sur la sélection du triangle de base. L'exemple est illustré par une vue perpendiculaire à la surface.

nalement q sur $\mathcal{T}$. Nous reviendrons sur ce problème à la §6.3.2.3. Pour le reste de cette section, nous nous concentrons sur la sélection des trois sommets de $\mathcal{T}$.

Comme Boubekeur *et al.* le rapportent, pour des raisons de robustesse, $\mathcal{T}$ devrait être le plus petit et le plus équilatéral possible, et devrait circonscrire q. Pour des raisons expliquées plus tard, nous requérons aussi que $n_{\mathcal{T}}$ ne soit pas très différente de l'estimation de normale à proximité de q. Toutes ces contraintes rendent le problème de sélection mal posé. Un exemple simple de conflit entre deux de ces quatre contraintes (taille et équilatéralité) est illustré à la figure 6.5. Dans cet exemple, nous avons deux triangles circonscrivant q, l'un est petit, l'autre quasi-équilatéral. Or, il n'est pas clair lequel est nécessairement le meilleur choix. Nous utilisons alors une méthode heuristique pour sélectionner les sommets de $\mathcal{T}$.

Boubekeur *et al.* utilisent une méthode simple de sélection à partir de $\mathcal{N}(q)$ en employant la notion de vote angulaire. Quand un point est choisi, il élimine de $\mathcal{N}(q)$ les points tombant dans un cône ayant pour sommet q et qui est aligné dans la direction du point choisi et d'un angle solide α, où α est initialement $\frac{2\pi}{3}$. Soit

$$C_j^{\alpha} = \left\{ c \in \mathcal{N}(q) \;\middle|\; (c-q)^{\mathsf{T}}(p_j^{[l-1]} - q) > \|c-q\|\|p_j^{[l-1]} - q\| \cos\frac{\alpha}{2} \right\}$$

les sommets tombant dans le cône du sommet j. Le premier sommet $p_{t_1}^{[l-1]}$ est fixé au point le plus près de q dans $\mathcal{N}(q)$. Le second sommet $p_{t_2}^{[l-1]}$ est choisi au hasard parmi $\mathcal{N}(q)\backslash C_{t_1}^{2\pi/3}$ et le troisième sommet $p_{t_3}^{[l-1]}$ est choisi au hasard parmi $\mathcal{N}(q)\backslash(C_{t_1}^{2\pi/3} \cup C_{t_2}^{2\pi/3})$. Si une des

différences d'ensemble s'avère être vide, la sélection est recommencée avec $\frac{\pi}{3}$, puis $\frac{\pi}{6}$, et ainsi de suite jusqu'à ce que le second et le troisième sommets puissent être choisis. Bien que cette méthode soit rapide et très simple d'implémentation, elle n'offre aucune garantie quant aux contraintes sur $\mathcal{T}$ outre sa taille limitée par l'étendue de $\mathcal{N}(\boldsymbol{q})$. Encore une fois, dans leur contexte d'application, cette sélection peut être suffisante considérant la grande différence de densité entre les ensembles de points. Dans notre cas, ce n'est pas assez robuste.

Notre procédure de sélection des sommets de $\mathcal{T}$, comme la méthode précédente, débute avec $\mathcal{N}(\boldsymbol{q})$ comme ensemble initial de candidats. Cependant, nous effectuons une bonne partie des calculs à partir de la projection des points sur un plan $\mathcal{H}$ passant par $\boldsymbol{q}$ et perpendiculaire à $\boldsymbol{n}(\boldsymbol{q})$. Ceci nous permet entre autres d'ordonner les points de façon angulaire. À partir de $\mathcal{N}(\boldsymbol{q})$, nous éliminons les points dont leur projection sur $\mathcal{H}$ est trop près de $\boldsymbol{q}$. Cette élimination peut aider à garder $\boldsymbol{n}_{\mathcal{T}}$ près de sa contrainte et réduire la possibilité d'obtenir des triangles très fins. Les points restants sont ensuite triés angulairement par rapport à leur projection sur $\mathcal{H}$. Si l'angle relatif maximal excède π, alors $\boldsymbol{p}_i^{[l]}$ est soit sur une frontière ou à proximité d'une zone de discontinuités de densité[6]. Dans le second cas, nous recommençons la sélection initiale à partir d'un voisinage euclidien dont le rayon est le rayon moyen des voisinages-k de tout point de $P^{[l-1]}$. Si malgré tout l'angle relatif maximal excède encore π, alors nous considérons $\boldsymbol{p}_i^{[l]}$ comme étant sur une frontière. Dans un tel cas, nous suivons une procédure particulière que nous décrivons plus loin.

Si le test d'angle est passé, l'ensemble des candidats est ensuite transformé en un voisinage-*BSP* (§2.3.1.1), afin de restreindre la taille finale de $\mathcal{T}$, et il est ensuite aussi trié angulairement. Si le nombre de points restants est moins que trois, ou l'angle relatif maximal excède π, nous suivons alors la procédure pour les cas frontaliers décrite plus loin. Sinon, il nous reste à choisir les trois sommets parmi les points restants. Comme nous désirons un triangle équilatéral autant que possible, nous cherchons alors trois points dont leurs angles relatifs, toujours calculés sur le plan $\mathcal{H}$, est le plus près de $\frac{2\pi}{3}$ que possible. Pour éviter une recherche exhaustive, nous fixons arbitrairement le premier sommet $\boldsymbol{p}_{t_1}^{[l-1]}$ comme étant le point le plus près de $\boldsymbol{q}$. Ensuite, $\boldsymbol{p}_{t_2}^{[l-1]}$ est le point dont l'angle relatif entre sa projection sur $\mathcal{H}$ et celle de $\boldsymbol{p}_{t_1}^{[l-1]}$ est le plus près de $\frac{2\pi}{3}$. Soit α cet angle. Le dernier sommet, $\boldsymbol{p}_{t_3}^{[l-1]}$, est le point dont l'angle relatif entre sa projection et celle de $\boldsymbol{p}_{t_2}^{[l-1]}$ est le plus près de $\frac{2\pi-\alpha}{2}$, *i.e.* l'angle bissecteur du reste.

[6]Pour les surfaces représentées par points, un de ses points peut se trouver *sur la frontière* sans que l'angle en question excède π, cependant, pour les fins de notre représentation multirésolution, si l'angle n'excède pas π, alors un triangle circonscrivant peut exister, et donc il n'est pas nécessaire de traiter spécialement le point.

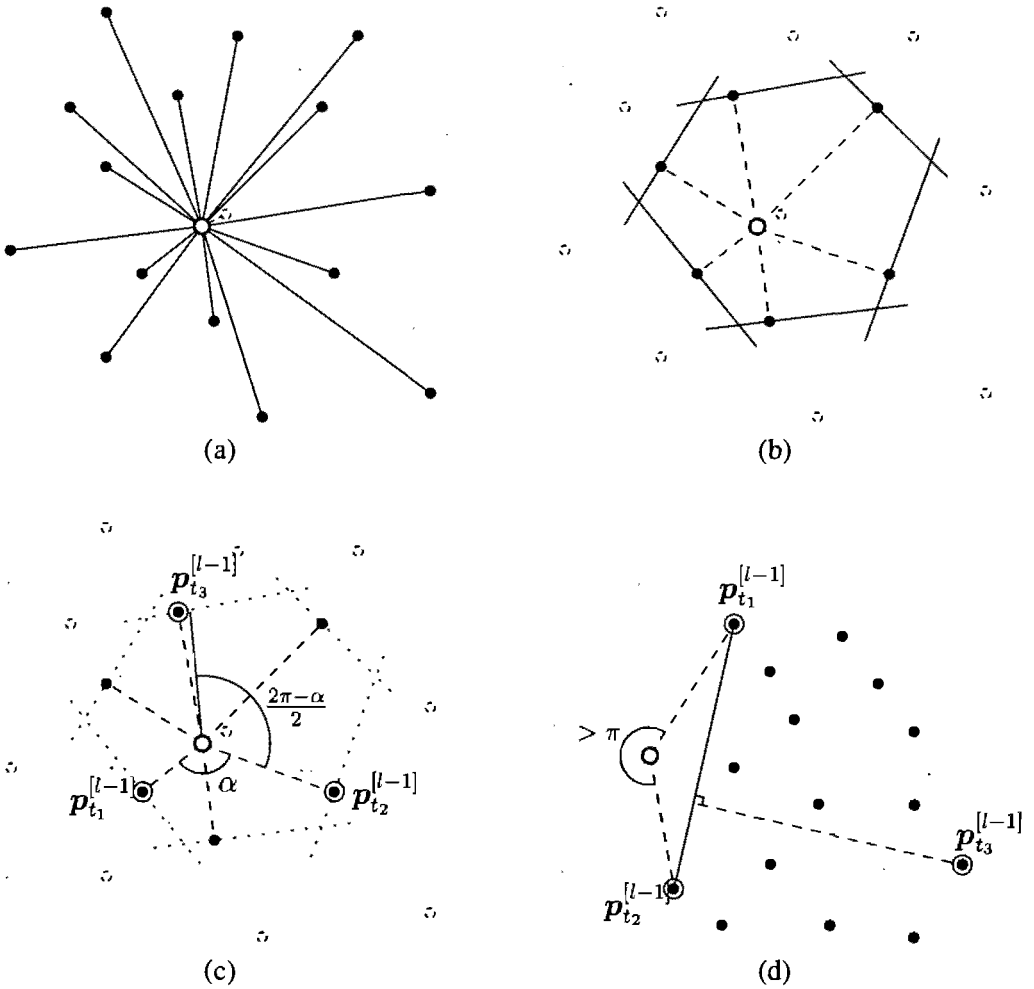


FIG. 6.6 – Sélection du triangle de base. Le cercle central ou sur la gauche en (d) représente q. (a) Candidats initiaux moins un point jugé trop près. (b) Conversion en voisinage-*BSP*. (c) Sélection de $\{p_{t_1}^{[l-1]}, p_{t_2}^{[l-1]}, p_{t_3}^{[l-1]}\}$. (d) Cas de sélection sur une frontière.

La figure 6.6 (a)-(c) illustre un exemple de la procédure. Pour simplifier l'illustration, nous supposons que la surface est localement planaire. Les étapes détaillées de la procédure de sélection des sommets de $\mathcal{T}$ sont énumérées ci-après.

1. Calculer un plan $\mathcal{H}$ passant par q et perpendiculaire à $n(q)$.

   Notation Soit un point $x \in \mathbb{R}^3$, pour le reste de cette section, $\check{x}$ dénote la projection orthogonale de x sur $\mathcal{H}$, où $\mathcal{H}$ est tel que défini ci-avant.

2. Sélectionner un ensemble de candidats :

   a. Calculer $\mathcal{N}(q)$ et éliminer les voisins dont leur projection tombe relativement trop près de q :

$$\mathcal{N}^\epsilon = \left\{ \boldsymbol{p} \in \mathcal{N}(\boldsymbol{q}) \;\middle|\; \|\check{\boldsymbol{p}} - \boldsymbol{q}\| > (1+\epsilon) h(\boldsymbol{q}) \right\} \tag{6.23}$$

où h est tel que défini à l'équation (2.49) et ϵ est la tolérance relative sur la proximité des projections, qui peut être assez grande (nous utilisons une valeur de 0.1).

b. Soit $J = \left\{ j \in \mathbb{Z}\left[1, |P^{[l-1]}|\right] \,\middle|\, \boldsymbol{p}_j^{[l-1]} \in \mathcal{N}^\epsilon \right\}$. Calculer une correspondance bijective

$$\iota : \mathbb{Z}\left[1, |\mathcal{N}^\epsilon|\right] \mapsto J$$

telle que $\forall j \in \mathbb{Z}\left[2, |\mathcal{N}^\epsilon|\right] : \angle \check{\boldsymbol{p}}_{\iota(1)}^{[l-1]}\, \boldsymbol{q}\, \check{\boldsymbol{p}}_{\iota(j-1)}^{[l-1]} \leq \angle \check{\boldsymbol{p}}_{\iota(1)}^{[l-1]}\, \boldsymbol{q}\, \check{\boldsymbol{p}}_{\iota(j)}^{[l-1]}$ [7].

c. Si

$$\left(\max_{j \in \mathbb{Z}[1, |\mathcal{N}^\epsilon|]} \angle \check{\boldsymbol{p}}_{\iota(j)}^{[l-1]}\, \boldsymbol{q}\, \check{\boldsymbol{p}}_{\iota((j+1) \bmod |\mathcal{N}^\epsilon|)}^{[l-1]} \right) > \pi, \tag{6.24}$$

recommencer les étapes 2a et 2b en utilisant le voisinage euclidien $\mathcal{N}^{\bar{r}}(\boldsymbol{q})$, où $\bar{r}$ est le rayon moyen des voisinages-k de tout point de $P^{[l-1]}$, et en gardant $h(\boldsymbol{q})$ identique à la première passe dans l'équation (6.23). Si après cette deuxième passe la condition de l'inéquation (6.24) est encore vraie, alors $\boldsymbol{p}_i^{[l]}$ est considéré sur une frontière et la procédure de sélection de $\mathcal{T}$ pour les cas frontières décrite plus loin est alors utilisée. Sinon, passer à l'étape suivante.

d. Convertir $\mathcal{N}^\epsilon$ en un voisinage-*BSP* $\mathring{\mathcal{N}}$ (équation (2.5)).

3. Sélectionner les sommets du triangle dans $\mathring{\mathcal{N}}$:

a. Soit $\mathring{J} = \left\{ j \in \mathbb{Z}\left[1, |P^{[l-1]}|\right] \,\middle|\, \boldsymbol{p}_j^{[l-1]} \in \mathring{\mathcal{N}} \right\}$. Fixer t_1 :

$$t_1 = \arg\min_{j \in \mathring{J}} \|\boldsymbol{p}_j^{[l-1]} - \boldsymbol{q}\|. \tag{6.25}$$

b. Calculer une correspondance bijective

$$\mathring{\iota} : \mathbb{Z}\left[1, |\mathring{\mathcal{N}}|\right] \mapsto \mathring{J}$$

de la même façon que ι de l'étape 2b, mais à partir de $\mathring{\mathcal{N}}$ et aussi telle que $\mathring{\iota}(1) = t_1$.

c. Si

$$|\mathring{\mathcal{N}}| < 3 \vee \left(\max_{j \in \mathbb{Z}[1, |\mathring{\mathcal{N}}|]} \angle \check{\boldsymbol{p}}_{\mathring{\iota}(j)}^{[l-1]}\, \boldsymbol{q}\, \check{\boldsymbol{p}}_{\mathring{\iota}((j+1) \bmod |\mathring{\mathcal{N}}|)}^{[l-1]} \right) > \pi \tag{6.26}$$

[7] À toute fin pratique, il s'agit de trier $\mathcal{N}^\epsilon$ angulairement par rapport à sa projection sur $\mathcal{H}$, comme le font Linsen et Prautzsch [2003] pour leur voisinage-α (§2.3.1.1). Cependant, pour des fins de notation, nous préférons exprimer ce réordonnement par l'intermédiaire d'une fonction de correspondance.

alors $\boldsymbol{p}_i^{[l]}$ est considéré sur une frontière et la procédure de sélection de $\mathcal{T}$ pour les cas frontières décrite plus loin est alors utilisée. Sinon, passer à l'étape suivante.

d. Déterminer t_2 et t_3 :

$$t_2 = \mathring{\iota}\left(\arg\min_{j\in\mathbb{Z}\left[2,|\mathring{\mathcal{N}}|-1\right]}\left|\angle\check{\boldsymbol{p}}_{t_1}^{[l-1]}\,\boldsymbol{q}\,\check{\boldsymbol{p}}_{\mathring{\iota}(j)}^{[l-1]}-\frac{2\pi}{3}\right|\right) \tag{6.27}$$

$$t_3 = \mathring{\iota}\left(\arg\min_{j\in\mathbb{Z}\left[\mathring{\iota}^{-1}(t_2)+1,|\mathring{\mathcal{N}}|\right]}\left|\angle\check{\boldsymbol{p}}_{t_2}^{[l-1]}\,\boldsymbol{q}\,\check{\boldsymbol{p}}_{\mathring{\iota}(j)}^{[l-1]}-\frac{2\pi-\angle\check{\boldsymbol{p}}_{t_1}^{[l-1]}\,\boldsymbol{q}\,\check{\boldsymbol{p}}_{t_2}^{[l-1]}}{2}\right|\right). \tag{6.28}$$

Il est à noter que parce que le plan de support de $\mathcal{T}$ n'est généralement pas le même que $\mathcal{H}$, il est possible que, malgré tout, $\mathcal{T}$ ne circonscrit pas $\boldsymbol{q}$. Ceci sera vrai si la distance entre $\boldsymbol{q}$ et $\boldsymbol{p}_{t_1}^{[l-1]}$ est petite et si $\boldsymbol{n}_{\mathcal{T}}$ diffère significativement de $\boldsymbol{n}(\boldsymbol{q})$, ce qui est peu probable par la construction de $\mathcal{T}$. Même si $\mathcal{T}$ ne circonscrit finalement pas $\boldsymbol{q}$, tant que la projection de $\boldsymbol{q}$ demeure à proximité de $\mathcal{T}$ (ce qui est généralement le cas), nous obtenons des résultats satisfaisants.

Pour un point situé sur une frontière, nous ne pouvons généralement pas obtenir un triangle $\mathcal{T}$ qui circonscrit $\boldsymbol{q}$. Nous devons donc utiliser une procédure différente de sélection. L'idée principale est de choisir un triangle tel qu'une seule coordonnée barycentrique soit négative et que la valeur des deux autres soit petite, *i.e.* si une est plus grande que un, ce ne sera pas par beaucoup. Nous voulons aussi obtenir un triangle avec un bon rapport d'aspect pour réduire l'impact d'étirement suite à une transformation de $P^{[l-1]}$.

La procédure est simple. Les deux premiers sommets sont les deux points de $\mathcal{N}^\epsilon$ ayant l'angle relatif le plus grand. Le troisième est le point le plus loin orthogonalement du segment formé par les deux autres. Plus en détail, suivant le calcul de ι à l'étape 2b de la procédure précédente, t_1, t_2 et t_3 sont calculés ainsi :

$$t_1 = \iota\left(\arg\max_{j\in\mathbb{Z}[1,|\mathcal{N}^\epsilon|]}\angle\check{\boldsymbol{p}}_{\iota(j)}^{[l-1]}\,\boldsymbol{q}\,\check{\boldsymbol{p}}_{\iota((j+1)\bmod|\mathcal{N}^\epsilon|)}^{[l-1]}\right) \tag{6.29}$$

$$t_2 = \iota\left(\left((\iota^{-1}(t_1)+1)\bmod|\mathcal{N}^\epsilon|\right)\right) \tag{6.30}$$

$$t_3 = \arg\max_{j\in J\setminus\{t_1,t_2\}}\left\|\boldsymbol{p}_j^{[l-1]}-\frac{\left(\boldsymbol{p}_j^{[l-1]}-\boldsymbol{p}_{t_1}^{[l-1]}\right)^{\mathsf{T}}\left(\boldsymbol{p}_{t_2}^{[l-1]}-\boldsymbol{p}_{t_1}^{[l-1]}\right)}{\|\boldsymbol{p}_{t_2}^{[l-1]}-\boldsymbol{p}_{t_1}^{[l-1]}\|^2}\left(\boldsymbol{p}_{t_2}^{[l-1]}-\boldsymbol{p}_{t_1}^{[l-1]}\right)\right\|^2. \tag{6.31}$$

La figure 6.6 (d) illustre un exemple.

6.3.2.3 Projection non orthogonale sur la base

Si nous reformulons $\boldsymbol{q}$ simplement selon l'équation (6.22), nous faisons face à deux problèmes principaux durant la reconstruction. D'abord, la direction de $\boldsymbol{n}_{\mathcal{T}}$ peut ne pas être fiable suite à une déformation de $P^{[l]}$. Ensuite, la projection sur $\mathcal{S}_{P^{[l-1]}}$ de certains voisins de $\boldsymbol{p}_i^{[l]}$ dans $P^{[l]}$ pourrait être reformulée sur le même triangle, et donc toute déformation de $P^{[l-1]}$ peut induire des déformations linéaires par morceaux dans la reconstruction. Soit $\mathcal{H}_{\mathcal{T}}$ le plan de support de $\mathcal{T}$. Au lieu de projeter orthogonalement $\boldsymbol{q}$ sur $\mathcal{H}_{\mathcal{T}}$, nous pourrions trouver un point $\boldsymbol{r} \in \mathcal{H}_{\mathcal{T}}$ tel que $\psi(\boldsymbol{r}) = \boldsymbol{q}$. Ainsi, $\boldsymbol{r}$ serait seulement un échantillon du domaine de ψ. Par contre, inverser ψ est loin d'être trivial. Alternativement, nous pourrions calculer $\boldsymbol{r}$ comme avant, mais à la reconstruction, utiliser la projection ψ. Cependant, l'erreur entre $\boldsymbol{q}$ et $\psi(\boldsymbol{r})$ n'est pas négligeable, particulièrement dans les zones très courbes.

Ce que nous faisons plutôt est de calculer un point $\boldsymbol{r} \in \mathcal{H}_{\mathcal{T}}$ tel que la droite passant par $\boldsymbol{q}$ de direction $\boldsymbol{n}(\boldsymbol{r})$ croise $\mathcal{H}_{\mathcal{T}}$ en $\boldsymbol{r}$, *i.e.* nous calculons un point $\boldsymbol{r}$ satisfaisant le système d'équations suivant :

$$\boldsymbol{q} - \big(\boldsymbol{n}(\boldsymbol{r})^{\mathsf{T}}(\boldsymbol{q} - \boldsymbol{r})\big)\boldsymbol{n}(\boldsymbol{r}) = \boldsymbol{r} \tag{6.32}$$

$$\boldsymbol{n}_{\mathcal{T}}{}^{\mathsf{T}}(\boldsymbol{r} - \boldsymbol{p}_{t_1}^{[l-1]}) = 0. \tag{6.33}$$

À noter que nous pouvons remplacer $\boldsymbol{p}_{t_1}^{[l-1]}$ par n'importe point de $\mathcal{H}_{\mathcal{T}}$ dans l'équation (6.33).

Nous pouvons combiner ces deux équations ensemble. Si nous substituons $\boldsymbol{r}$ dans l'équation (6.33) par l'équation (6.32), nous obtenons

$$\boldsymbol{n}_{\mathcal{T}}{}^{\mathsf{T}}(\boldsymbol{q} - \big(\boldsymbol{n}(\boldsymbol{r})^{\mathsf{T}}(\boldsymbol{q} - \boldsymbol{r})\big)\boldsymbol{n}(\boldsymbol{r}) - \boldsymbol{p}_{t_1}^{[l-1]}) = 0$$

$$\iff \qquad \boldsymbol{n}(\boldsymbol{r})^{\mathsf{T}}(\boldsymbol{q} - \boldsymbol{r}) = \frac{\boldsymbol{n}_{\mathcal{T}}{}^{\mathsf{T}}(\boldsymbol{q} - \boldsymbol{p}_{t_1}^{[l-1]})}{\boldsymbol{n}_{\mathcal{T}}{}^{\mathsf{T}}\boldsymbol{n}(\boldsymbol{r}^{(t)})}. \tag{6.34}$$

En effectuant la substitution de l'équation (6.34) dans l'équation (6.32), nous obtenons

$$\boldsymbol{q} - \frac{\boldsymbol{n}_{\mathcal{T}}{}^{\mathsf{T}}(\boldsymbol{q} - \boldsymbol{p}_{t_1}^{[l-1]})}{\boldsymbol{n}_{\mathcal{T}}{}^{\mathsf{T}}\boldsymbol{n}(\boldsymbol{r}^{(t)})}\boldsymbol{n}(\boldsymbol{r}) = \boldsymbol{r}. \tag{6.35}$$

La partie gauche de cette équation correspond à l'intersection de la droite passant par $\boldsymbol{q}$ de direction $\boldsymbol{n}(\boldsymbol{r})$ avec le plan $\mathcal{H}_{\mathcal{T}}$. En choisissant une estimation initiale appropriée pour $\boldsymbol{r}$, nous

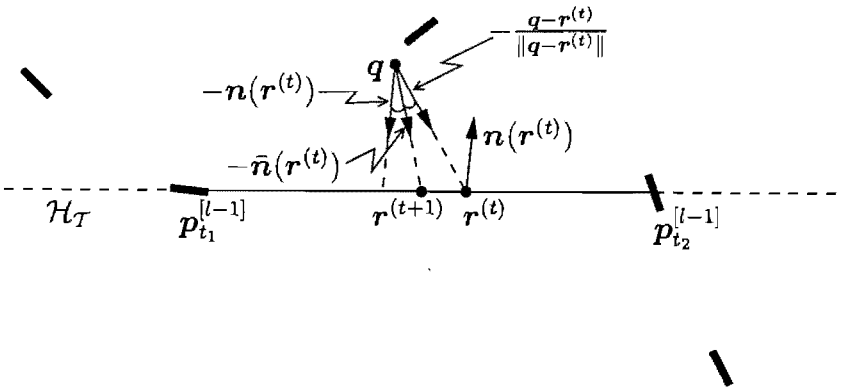


FIG. 6.7 – Interprétation géométrique de la projection non orthogonale sur le triangle de base, représenté ici en 2D par le segment $\overline{p_{t_1}^{[l-1]} p_{t_2}^{[l-1]}}$.

pouvons utiliser l'équation (6.35) comme processus itératif, *i.e.*

$$r^{(t+1)} = q - \frac{{n_\mathcal{T}}^\mathsf{T}(q - p_{t_1}^{[l-1]})}{{n_\mathcal{T}}^\mathsf{T} n(r^{(t)})} n(r^{(t)}). \tag{6.36}$$

Ce processus tend parfois à surdépasser (*overshoot*) la solution. Nous pouvons l'aider en amortissant la direction $n(r^{(t)})$ à la direction bissectrice de $(q - r^{(t)})/\|q - r^{(t)}\|$ et $n(r^{(t)})$ (ou $-n(r^{(t)})$ si les directions sont opposées). Soit $\bar{n}(r^{(t)})$ cette direction. Nous obtenons finalement le processus itératif suivant :

$$r^{(t+1)} = q - \frac{{n_\mathcal{T}}^\mathsf{T}(q - p_{t_1}^{[l-1]})}{{n_\mathcal{T}}^\mathsf{T} \bar{n}(r^{(t)})} \bar{n}(r^{(t)}) \tag{6.37}$$

où

$$\bar{n}(r^{(t)}) = \begin{cases} \frac{q-r^{(t)}}{\|q-r^{(t)}\|} + n(r^{(t)}) & \text{si} \quad n(r^{(t)})^\mathsf{T} \frac{(q-r^{(t)})}{\|q-r^{(t)}\|} > 0 \\ \frac{q-r^{(t)}}{\|q-r^{(t)}\|} - n(r^{(t)}) & \text{sinon.} \end{cases} \tag{6.38}$$

Il n'est pas nécessaire de normaliser $\bar{n}(r^{(t)})$ car le facteur de normalisation s'annulerait dans l'équation (6.37). Le processus itératif convergera si $r^{(0)}$ est raisonnablement proche de la solution et si $n(x)$ est une fonction continue d'ordre supérieur à zéro (ce qui est le cas[8]) tel que $n(r^{(t)})$ ne s'écarte pas trop de $n_\mathcal{T}$, d'où la contrainte associée à l'orientation de $\mathcal{T}$ durant la sélection de ce dernier (§6.3.2.2). Nous initialisons le processus itératif par l'intersection entre

[8]Cette continuité dépend du choix de la fonction de pondération θ (§2.3.3.2). Dans notre cas, il s'agit d'une gaussienne, qui est C^∞. Si $n(x)$ est calculée par analyse de covariance, la continuité dépend de la cohérence de la détermination de l'orientation finale. Par contre, la direction non orientée est continue.

$\mathcal{H}_\mathcal{T}$ et la droite passant par q de direction $n(q)$, *i.e.*

$$r^{(0)} = q - \frac{{n_\mathcal{T}}^\mathsf{T}(q - p_{l_1}^{[l-1]})}{{n_\mathcal{T}}^\mathsf{T} n(q)} n(q). \tag{6.39}$$

La figure 6.7 illustre un exemple d'initialisation et d'une itération du processus.

Nous arrêtons les itérations lorsque l'erreur sur l'équation (6.32) arrive sous un seuil donné, *i.e.* si

$$\|q - r^{(t)}\|^2 - \left(n(r^{(t)})^\mathsf{T}(q - r^{(t)})\right)^2 < \epsilon^2 \tag{6.40}$$

pour une tolérance d'erreur $\epsilon > 0$, ou lorsqu'un nombre maximum d'itérations est atteint. Malgré l'amortissement ajouté dans l'équation (6.37), le processus éprouve, mais très rarement, certains problèmes de convergence. Nous les détectons de deux façons :

1. à la fin de chaque itération, si l'erreur (côté gauche de l'inéquation (6.40)) est plus grande qu'à l'itération précédente ;

2. le nombre maximum d'itérations est atteint mais la condition d'arrêt de l'inéquation (6.40) n'est toujours pas satisfaite.

Dans le cas 1, nous reprenons l'itération en amortissant davantage $\bar{n}(r^{(t)})$ par la direction bissectrice entre $\bar{n}(r^{(t)})$ et $(q - r^{(t)})/\|q - r^{(t)}\|$. Cet amortissement est repris jusqu'à ce que l'erreur soit plus petite qu'à l'itération précédente ou qu'un autre maximum de fois soit atteint. Dans le premier cas, nous poursuivons le processus régulièrement. Dans l'autre cas, nous procédons selon le cas 2. Ce dernier peut arriver si $r^{(0)}$ n'était pas assez près de la solution ou, rarement, si la convergence est lente. Nous recommençons alors le processus en initialisant $r^{(0)}$ par une perturbation de $r^{(t)}$ sur $\mathcal{H}_\mathcal{T}$. À nouveau, nous permettons cette reprise complète un nombre maximum de fois. Par souci de complétude, si malgré tout l'inégalité (6.40) n'arrive pas à être satisfaite, alors nous choisissons simplement $r^{(0)}$ comme solution.

Il est à noter qu'il est possible que r tombe à l'extérieur des frontières de $\mathcal{T}$. Cependant, dans toutes nos expérimentations, ceci se produit dans au plus 2% des points traités, et dans ces cas, r n'est jamais loin des frontières.

6.4 Reconstruction

La reconstruction est l'opération inverse de la décomposition. Une étape de la reconstruction consiste à produire $P^{[l]}$ à partir de $P^{[l-1]}$ et $D^{[l]}$ (équation (6.3)). Ainsi, pour reconstruire $P^{[L]}$, nous devons la répéter L fois, *i.e.* pour $l = 1, 2, \ldots, L$.

Conceptuellement, il s'agit d'échantillonner $\mathcal{S}_{P^{[l-1]}}$ pour produire un ensemble de points Q tel que $|Q| = |D^{[l]}|$ et qui sert de base pour subséquemment appliquer les détails géométriques. Cet échantillonnage correspond à l'opérateur Λ^{-1} de l'équation (6.3), et l'application des détails géométriques correspond à l'opérateur $\oplus$ de la même équation. Parce que Λ^{-1} emploie des informations de $D^{[l]}$, et parce que tout point reconstruit peut être calculé indépendamment, nous décrivons la procédure de reconstruction en terme de traitement de chaque $\boldsymbol{d}_i^{[l]} \in D^{[l]}$ plutôt que de globalement échantillonner $\mathcal{S}_{P^{[l-1]}}$ et ensuite appliquer les détails géométriques.

Pour chaque $\boldsymbol{d}_i^{[l]} \in D^{[l]}$, nous calculons le point $\boldsymbol{p}_i^{[l]}$ correspondant selon la procédure suivante.

1. Soit $(t_1, t_2, t_3, \beta_2, \beta_3, \delta) = \boldsymbol{d}_i^{[l]}$ (équation (6.18)).
2. Calculer le point de base $\boldsymbol{r} = \sum_{j=1}^{3} \beta_j \boldsymbol{p}_{t_j}^{[l-1]}$, où $\beta_1 = 1 - \beta_2 - \beta_3$.
3. Calculer $\boldsymbol{q}$ par l'intersection de la droite passant par $\boldsymbol{r}$ de direction $\boldsymbol{n}(\boldsymbol{r})$ avec $\mathcal{S}_{P^{[l-1]}}$.
4. Fixer $\boldsymbol{p}_i^{[l]} = \boldsymbol{q} + \delta \boldsymbol{n}(\boldsymbol{q})$.

Finalement, $P^{[l]} = \{\boldsymbol{p}_i^{[l]}\}_{i=1}^{|D^{[l]}|}$. Ainsi, selon cette procédure, le résultat de l'opérateur Λ^{-1} est l'ensemble des points produits à l'étape 3, et le résultat de l'opérateur $\oplus$ est l'ensemble des points produits à l'étape 4.

Par la suite, nous devons calculer les normales de chaque point de $P^{[l]}$. Il est à noter que ce calcul ne peut pas être effectué par une moyenne pondérée de normales car l'ensemble de points de référence pour ce calcul doit être $P^{[l]}$, et non $P^{[l-1]}$ pour tenir compte des nouveaux détails géométriques. Nous utilisons la méthode par analyse de covariance (équation (2.22)).

L'intersection avec $\mathcal{S}_{P^{[l-1]}}$ de l'étape 3 est calculée en utilisant la procédure d'intersection d'Adamson et Alexa [2004]. Cependant, leur procédure considère l'intersection entre une droite quelconque et une surface MCM, ce qui requiert la construction d'un partitionnement de l'espace pour efficacement isoler des régions candidates du domaine de la surface pour l'intersection, s'il y en a une. Dans notre cas, par construction du triangle $\mathcal{T}$ formé par les points d'indices t_1, t_2 et t_3 de $P^{[l-1]}$, nous savons que $\boldsymbol{r}$ est déjà près de $\mathcal{S}_{P^{[l-1]}}$, et nous savons aussi que $\boldsymbol{n}(\boldsymbol{r})$ est une direction qui, à partir de $\boldsymbol{r}$, devrait croiser $\mathcal{S}_{P^{[l-1]}}$ à proximité de $\boldsymbol{r}$. Donc

nous n'avons pas besoin de construire un partitionnement de l'espace, et nous n'avons qu'une seule région candidate. La procédure ainsi simplifiée peut alors être formulée selon le processus itératif suivant :

$$\boldsymbol{q}^{(t+1)} = \boldsymbol{r} - \frac{\boldsymbol{n}(\boldsymbol{q}^{(t)})^{\mathsf{T}}(\boldsymbol{q}^{(t)} - \boldsymbol{a}(\boldsymbol{q}^{(t)}))}{\boldsymbol{n}(\boldsymbol{q}^{(t)})^{\mathsf{T}}\boldsymbol{n}(\boldsymbol{r})}\boldsymbol{n}(\boldsymbol{r}) \tag{6.41}$$

où $\boldsymbol{q}^{(0)} = \boldsymbol{r}$. Ce processus est très similaire à la projection orthogonale de l'équation (2.29), mais en gardant la direction de projection constante à $\boldsymbol{n}(\boldsymbol{r})$ plutôt que $\nabla f(\boldsymbol{q}^{(t)})$ (f est définie à l'équation (2.24) et correspond au numérateur de la fraction dans l'équation (6.41)). Ce processus est arrêté lorsque $|f(\boldsymbol{q}^{(t)})| < \epsilon$ pour une tolérance d'erreur $\epsilon > 0$.

6.4.1 Reconstruction plus rapide

Nous pouvons obtenir une reconstruction plus rapide si nous acceptons de stocker un scalaire supplémentaire par $\boldsymbol{d}_i^{[l]} \in D^{[l]}$. Suite à la projection non orthogonale sur $\mathcal{H}_{\mathcal{T}}$ (§6.3.2.3), nous conservons aussi la valeur $\tilde{\delta} = \boldsymbol{n}(\boldsymbol{r})^{\mathsf{T}}(\boldsymbol{q} - \boldsymbol{r})$, également illustrée à la figure 6.4. Alors, durant la reconstruction, les étapes 1 et 3 sont remplacées par

1. Soit $(t_1, t_2, t_3, \beta_2, \beta_3, \tilde{\delta}, \delta) = \boldsymbol{d}_i^{[l]}$.
3. Calculer $\boldsymbol{q} = \boldsymbol{r} + \tilde{\delta}\boldsymbol{n}(\boldsymbol{r})$.

Il est à noter que ce truc n'est pas exactement équivalent à l'approche initiale si $P^{[l-1]}$ a été déformé, particulièrement si la courbure change. Malgré tout, les résultats obtenus demeurent satisfaisants.

6.4.2 Mise à l'échelle adaptative des détails géométriques

Durant la reconstruction, si nous désirons que les détails géométriques subissent une mise à l'échelle en fonction de l'étirement local, alors, de la même façon que le font Boubekeur *et al.* [2007], au lieu de stocker δ comme dernière composante de $\boldsymbol{d}_i^{[l]}$ (équation (6.18)), nous stockons plutôt $\frac{\delta}{\Delta_{\mathcal{T}}}$, où $\Delta_{\mathcal{T}}$ est l'aire de $\mathcal{T}$ au moment de la décomposition. Ainsi, durant la reconstruction, les étapes 1 et 4 deviennent

1. Soit $(t_1, t_2, t_3, \beta_2, \beta_3, \frac{\delta}{\Delta_{\mathcal{T}}}) = \boldsymbol{d}_i^{[l]}$ [9].
4. Fixer $\boldsymbol{p}_i^{[l]} = \boldsymbol{q} + \frac{\delta}{\Delta_{\mathcal{T}}}\Delta_{\mathcal{T}'}$, où $\mathcal{T}'$ est le triangle de sommets $\{\boldsymbol{p}_{t_1}^{[l-1]}, \boldsymbol{p}_{t_2}^{[l-1]}, \boldsymbol{p}_{t_3}^{[l-1]}\}$ après une déformation possible de $P^{[l-1]}$.

[9] Si nous combinons avec l'option précédente, l'étape serait plutôt
1. Soit $(t_1, t_2, t_3, \beta_2, \beta_3, \tilde{\delta}, \frac{\delta}{\Delta_{\mathcal{T}}}) = \boldsymbol{d}_i^{[l]}$.

Nous ne pouvons pas appliquer une mise à l'échelle similaire pour $\tilde{\delta}$ car cette valeur de déplacement est calculée par rapport à un triangle qui approxime localement $\mathcal{S}_{P^{[l-1]}}$, et pour deux voisins de $P^{[l]}$, leurs triangles sélectionnés respectifs peuvent être suffisamment différents pour que les valeurs $\tilde{\delta}$ ne soient pas cohérentes.

6.5 Résultats

Nous avons utilisé notre représentation multirésolution de surfaces MCM sur un certain nombre d'ensembles de points, que nous décrivons ci-après.

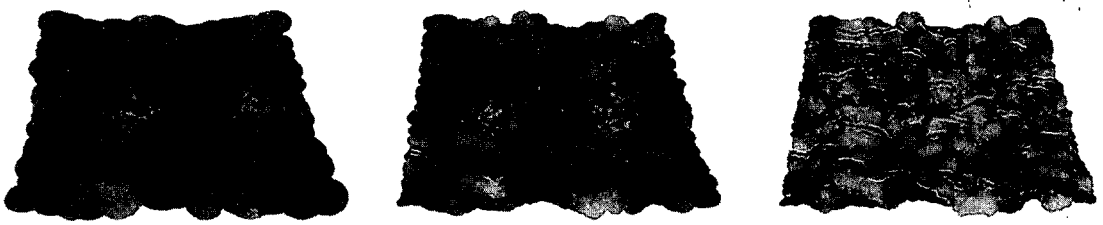
Bosses Ensemble de 97 284 points affiché à la figure 6.8 (g). Il s'agit d'une simple carte d'élévation générée à partir d'une image représentant une texture de petits cailloux. Cet exemple est riche en détails géométriques et sa forme globale simple (un carré) le rend propice pour évaluer l'effet de déformation. Il a surtout permis de nous rendre rapidement compte des problèmes que posent les frontières.

Hygie Ensemble de 134 345 points affiché à la figure 6.9 (b). Il provient de la collection d'exemples de *Cyberware* [Cyberwarea] pour leur numériseur 3D de bureau [Cyberwareb] (*Igea*). Il s'agit d'un objet 3D populaire dans la communauté graphique. Pour notre représentation, cet exemple correspond à un cas typique, *i.e.* qu'il comporte une certaine variété de détails géométriques, n'a pas de frontière (le cou est en effet fermé) et n'a pas de structures fines (minces ou allongées).

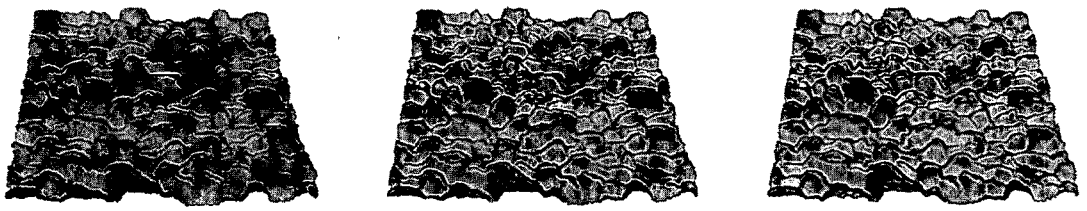
Tatou Ensemble de 172 974 points affiché à la figure 6.10 (e). Il provient de la banque de numérisations 3D de Stanford [Sta] (*Armadillo*). Il s'agit d'un exemple fréquemment utilisé dans les travaux traitant de multirésolution car la surface contient plusieurs détails géométriques de fréquences variées. Pour notre représentation, il s'agit de l'ensemble de points qui nous a fait réaliser les problèmes associés à la simplification sans contrainte car il comporte plusieurs structures fines dont les doigts, la queue et les oreilles.

Isis Ensemble de 196 256 points affiché à la figure 6.11 (f). Il provient aussi de Cyberware [Cyberwarea], mais de leur collection d'exemples pour leur numériseur 3D couleur *Model Shop* [Cyberwarec]. Il s'agit d'un exemple intermédiaire entre "Hygie" et "Tatou", qui a aussi été utilisé, entre autres, par Fleishman *et al.* [2003a] (§5.3.2.1) et dans les travaux de simplification d'ensembles de points de Pauly *et al.* [2002a].

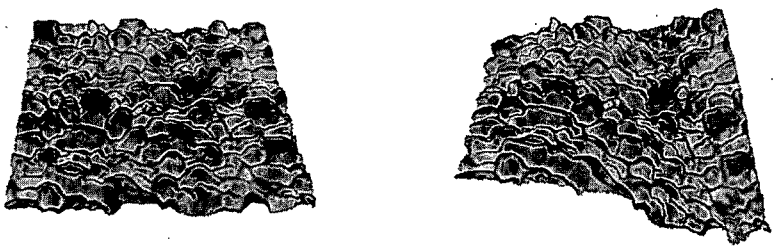
De plus, il est à noter que la dernière valeur du nuplet $\boldsymbol{d}_i^{[l]}$ est un scalaire. Nous ne pouvons plus connaître les valeurs δ et $\Delta_\mathcal{T}$ séparément. Néanmoins, nous gardons la notation $\frac{\delta}{\Delta_\mathcal{T}}$ pour exprimer ce scalaire.



(a) Niveau 0 (b) Niveau 1 (c) Niveau 2



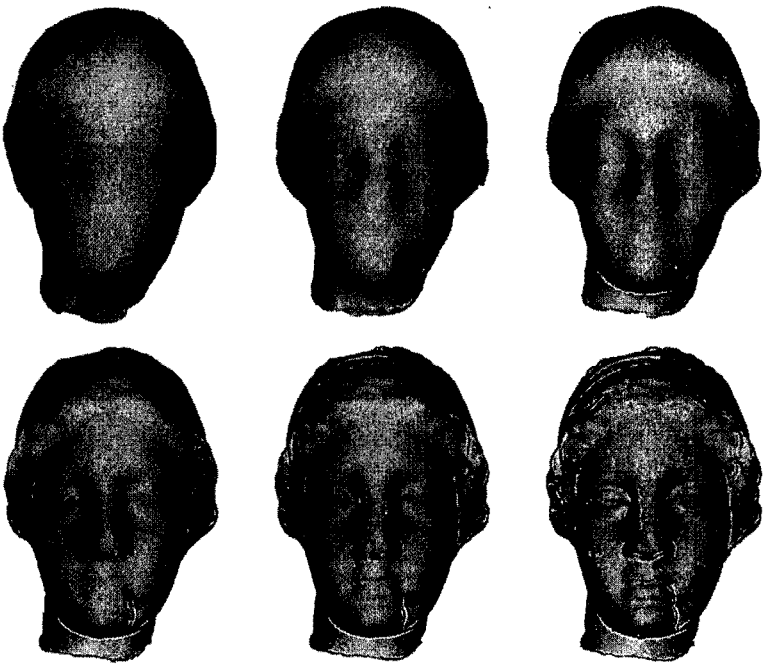
(d) Niveau 3 (e) Niveau 4 (f) Niveau 5



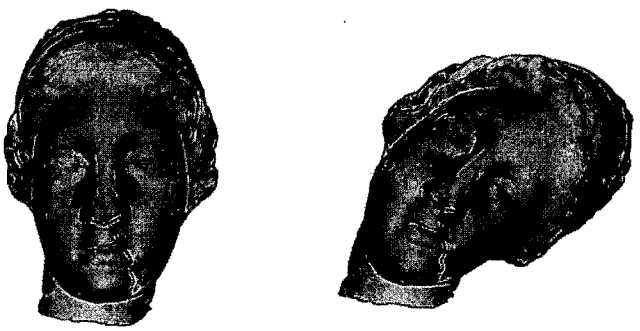
(g) Original (h) Déformé

(i) Déformé au niveau 5 (j) Déformation de (i) appliquée au niveau 0

FIG. 6.8 – Niveaux reconstruits, surface originale et quelques déformations pour l'ensemble de points "Bosses".



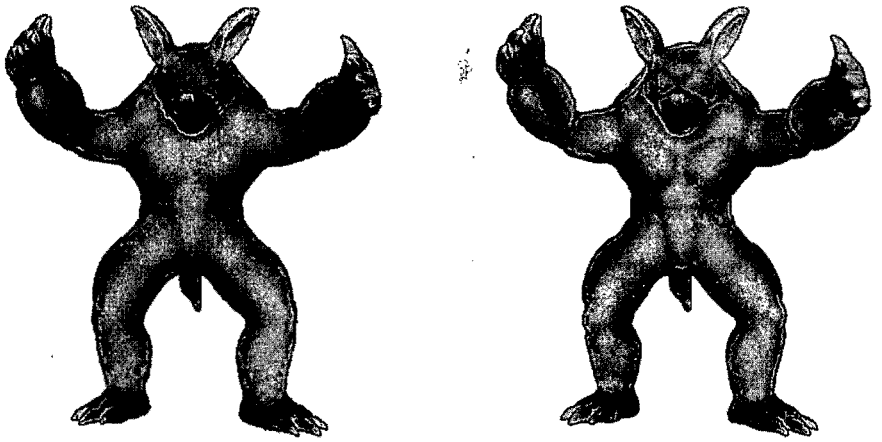
(a) Niveaux reconstruits 0 à 5.



(b) Original, déformé.

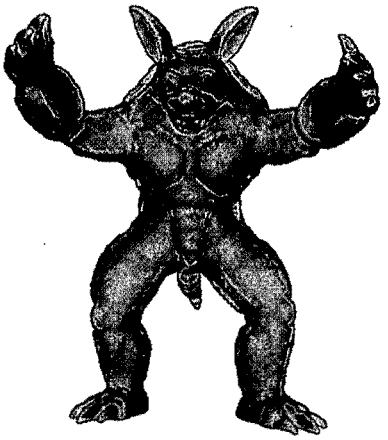
(c) Mise à l'échelle des détails géométriques : 0-0-0-1-1 ; 1-1-1-2-3 ; 2-2-1-1-1.

FIG. 6.9 – Résultats pour l'ensemble de points "Hygie".

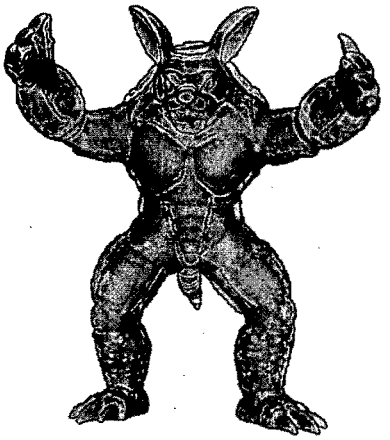


(a) Niveau 0

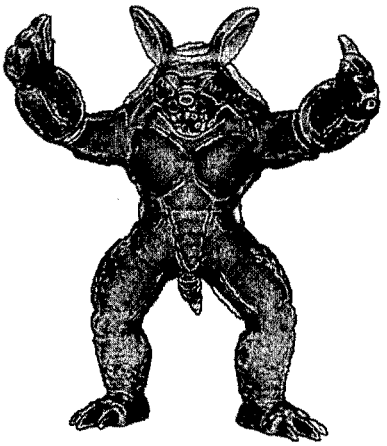
(b) Niveau 1



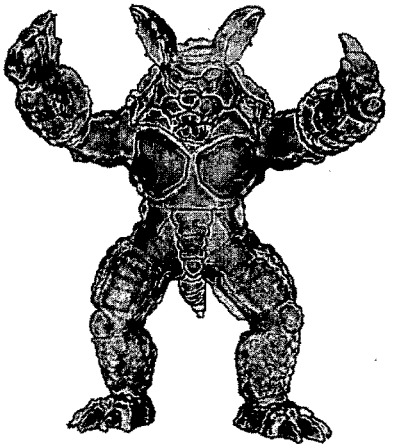
(c) Niveau 2



(d) Niveau 3

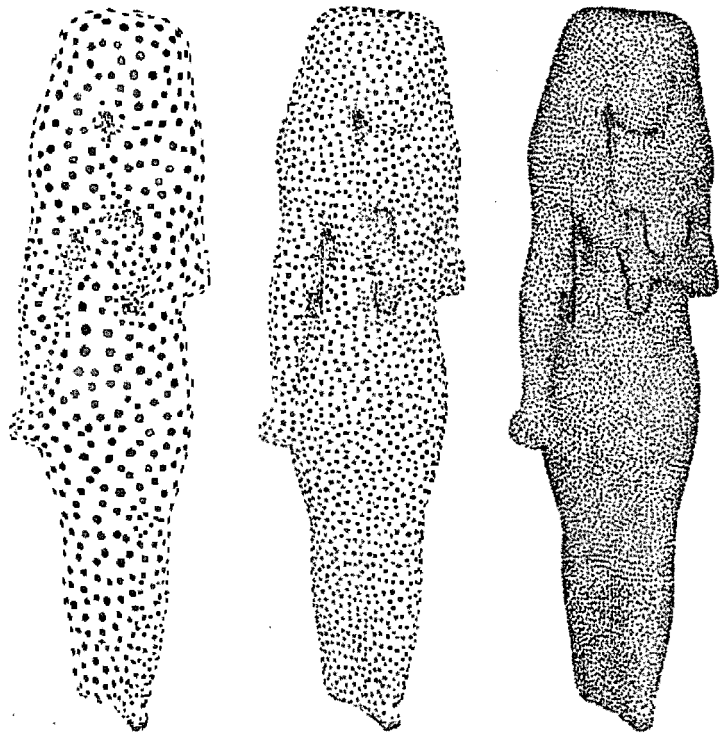


(e) Original

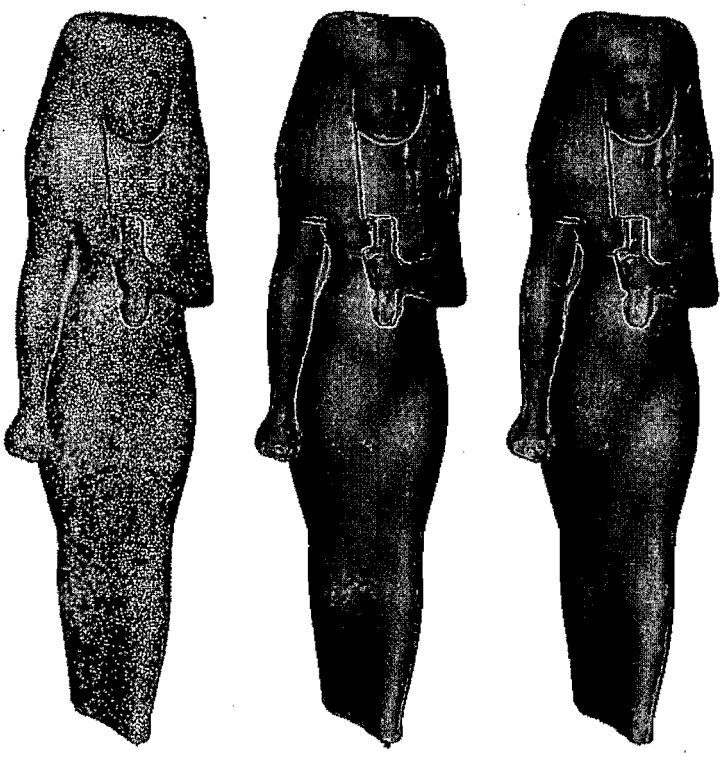


(f) Mise à l'échelle des détails géométriques (2-2-1)

FIG. 6.10 – Niveaux reconstruits, surface originale, and mise à l'échelle des détails géométriques pour l'ensemble de points "Tatou".



(a) Niveau 0 (b) Niveau 1 (c) Niveau 2



(d) Niveau 3 (e) Niveau 4 (f) Original

FIG. 6.11 – Niveaux reconstruits et surface originale pour l'ensemble de points "Isis", avec rayon d'affichage réduit pour rendre les points plus visibles.

Dans les sections suivantes, nous présentons d'abord les résultats de la décomposition et de la reconstruction simple de ces ensembles de points pour valider la base fondamentale de la représentation et extraire des statistiques sur les temps d'exécution (§6.5.1). Ensuite, nous présentons quelques résultats d'applications de notre représentation multirésolution pour l'édition d'objets (§6.5.2).

6.5.1 Décomposition et reconstruction

Les figures 6.8 (a)–(f), 6.9 (a) et 6.10 (a)–(d) montrent la reconstruction à différents niveaux[10] des ensembles "Bosses"[11], "Hygie" et "Tatou" respectivement. La figure 6.11 (a)–(e) montre des résultats similaires, mais en réduisant la taille d'affichage des points pour mieux rendre visible l'effet de simplification d'un niveau à l'autre. Cette dernière figure montre bien que, tel que décrit à la §6.3.1, plus de points sont maintenus dans les régions potentiellement problématiques, notamment dans le creux à la jonction du cou et de la tête, tel qu'à la figure 6.3 (b), et la région entre le corps et le bras droit, tel qu'à la figure 6.3 (a).

Nous avons omis de montrer les ensembles de points produits à chaque niveau durant la décomposition car nous n'avons constaté aucune différence visuelle entre ceux-ci et les niveaux reconstruits correspondants. Nous le voyons dans les figures 6.8 (f)–(g), 6.9 (a)–(b), 6.10 (d)–(e) et 6.11 (e)–(f) où nous pouvons comparer la reconstruction du niveau le plus détaillé (niveau 5, 5, 3 et 4 respectivement) avec l'ensemble de points original. En fait, l'erreur-type entre les ensembles de points reconstruits et les originaux n'est respectivement que de 9.5×10^{-5}, 2.64×10^{-4}, 4.90×10^{-4} et 1.75×10^{-4}. Soit $P^{[L]}$ l'ensemble de points original et $P'^{[L]}$ l'ensemble de points reconstruit, l'erreur-type est calculée ainsi :

$$\frac{\sqrt{\frac{1}{|P^{[L]}|}\sum_{i=1}^{|P^{[L]}|} \|\boldsymbol{p}_i^{[L]} - \boldsymbol{p}_i'^{[L]}\|^2}}{\max\limits_{j\in\mathbb{Z}[1,3]} \left(\max\limits_{i\in\mathbb{Z}[1,|P^{[L]}|]} \boldsymbol{e}_j{}^{\mathsf{T}} \boldsymbol{p}_i^{[L]} - \min\limits_{i\in\mathbb{Z}[1,|P^{[L]}|]} \boldsymbol{e}_j{}^{\mathsf{T}} \boldsymbol{p}_i^{[L]} \right)}$$

où $\boldsymbol{e}_j$ est le j^{e} vecteur unitaire de la base canonique de $\mathbb{R}^3$, *i.e.* un vecteur tel que sa j^{e} composante est 1 et les autres valent zéro. Le dénominateur est un facteur de normalisation pour ajuster l'ensemble de points à l'intérieur d'un cube unitaire de telle sorte que l'étendue la plus grande de sa boîte englobante alignée sur les axes principaux de $P^{[L]}$ devienne égale à un.

[10] Il est à noter que le niveau 0 dans ces figures ainsi qu'à la figure 6.11 n'est pas reconstruit. Il s'agit du niveau de la dernière étape de décomposition.

[11] Le peu de différences visuelles entre les niveaux 4 (figure 6.8 (e)) et 5 (figure 6.8 (f)) est dû au fait que le niveau 5 est suréchantillonné par rapport aux fréquences de ces détails géométriques.

		Bosses	Hygie	Tatou	Isis
Taux de simplification cible		4	4	5	5
Nombre de points	Niveau 0	374	500	6 510	1 600
	Niveau 1	1 140	1 512	15 218	4 368
	Niveau 2	3 583	4 551	48 312	14 953
	Niveau 3	10 395	14 503	172 974	53 883
	Niveau 4	32 028	43 636	—	196 256
	Niveau 5	97 284	134 345	—	—

TAB. 6.1 – Nombre de points pour chaque niveau reconstruit.

	Bosses	Hygie	Tatou	Isis
Décomposition	17.9 / 36.7	27.4 / 60.1	137.4 / —	39.9 / 67.3
Reconstruction	5.0 / 6.1	7.6 / 12.2	9.5 / —	10.5 / 14.4

TAB. 6.2 – Temps de calcul (en secondes) pour la décomposition et la reconstruction. Pour chaque entrée, le premier nombre est le temps pour notre méthode, et le second est pour notre implémentation de la méthode de Pauly *et al.* [2006]. Toutes implémentations sont en C++ [Stroustrup 2000] et tous les temps ont été obtenus sur un processeur *AMD Turion 64 X2* 1.8 GHz.

Le tableau 6.1 indique le taux de simplification cible et le nombre de points obtenu pour chaque niveau de résolution. Le ratio du nombre de points entre deux niveaux consécutifs n'est pas égal au taux de simplification cible car, d'abord, la méthode de mise en agrégats hiérarchiques de la §6.3.1 ne garantit pas d'atteindre exactement le taux cible, et ensuite, le raffinement (équations (6.14) à (6.16)) réduit davantage le taux obtenu.

Les temps de calcul requis pour compléter tous les niveaux de décomposition, de l'original au niveau 0, et pour compléter la reconstruction, du niveau 0 jusqu'au niveau le plus détaillé, se trouvent au tableau 6.2. Les temps de reconstruction ont été obtenus en utilisant l'option rapide décrite à la §6.4.1, *i.e.* de conserver $\tilde{\delta}$ pour chaque point. Ce tableau indique aussi les temps d'exécution de notre implémentation de la représentation de Pauly *et al.* [2006] pour les mêmes nombres de niveaux et taux de simplification (pour le lissage dans leur cas). Par contre, pour leur représentation, nous ne présentons pas de temps pour l'ensemble de points "Tatou" car leur méthode de simplification pour le lissage ne tient pas compte des conditions de voisinage (§6.3.1). Nous aurions pu l'implémenter, mais leur temps d'exécution pour la décomposition n'en serait que plus grand. Nous avons tout de même mesuré le temps d'exécution pour l'en-

semble de points "Isis" pour leur représentation[12]. Malgré que nous effectuons plus de calculs par étape de décomposition et reconstruction, les temps d'exécution pour notre représentation sont généralement moindres, ceci grâce à la réduction du nombre de points.

Dans certains cas, nous pourrions anticiper que nos temps d'exécution soient plus grands. En effet, notons le temps d'exécution plus élevé pour la décomposition de l'ensemble de points "Tatou". En comparant le nombre de points des autres ensembles par rapport à leurs temps d'exécution respectifs, nous devrions nous attendre à ce que l'ensemble "Tatou" prenne moins de temps que l'ensemble "Isis", ce qui est d'ailleurs le cas pour la reconstruction. Mais alors que se passe-t-il avec la décomposition de "Tatou" pour quelle prenne autant de temps ? La particularité de cet exemple est la présence de plusieurs structures fines dont les doigts, les orteils, la queue et les oreilles. Ceci fait en sorte que le nombre de points dans ces régions diminue très peu. Si nous comparons le nombre de points pour chaque niveau par rapport à l'ensemble "Isis", ce n'est qu'aux niveaux inférieurs que le nombre de points finit par être plus grand (il faut comparer le niveau l avec $l+1$ respectivement, $l \in \mathbb{Z}[0,3]$) ; donc l'impact sur le temps d'exécution ne peut pas être très grand. En fait, le goulot d'étranglement survient à l'extraction des détails. Les structures particulières de "Tatou" créent des zones de discontinuités de densité, ce qui ralentit la sélection du triangle (§6.3.2.2), mais surtout, la courbure de la surface à certains endroits est élevée, ce qui donne des problèmes à la convergence de la projection non orthogonale de la §6.3.2.3, car l'estimation de normales varie beaucoup.

6.5.2 Édition

Nous avons souligné auparavant (§5.1) l'intérêt des représentations multirésolutions pour l'édition d'objets, et nous présentons dans cette section quelques résultats avec l'usage de notre représentation[13].

Sans doute, leur usage principal est d'effectuer des déformations qui préservent l'apparence des détails géométriques plus fins que le niveau de détail affecté par la déformation. Par exemple, simplement bomber l'ensemble de points "Bosses" (figure 6.8 (g)) vers le haut[14] correspond à une déformation qui affecte la forme générale de l'ensemble. Ainsi, si nous avions un ensemble de points correspondant au niveau 0 (figure 6.8 (a)), nous pourrions effectuer une

[12] La reconstruction était d'ailleurs erronée dans certaines régions.

[13] Il est à noter que les applications que nous donnons en exemple dans cette section ne sont pas nouvelles. Nous les présentons dans un but de validation de notre représentation. Néanmoins, nous en profitons pour remettre en valeur l'avantage des représentations multirésolutions pour les types d'édition présentés.

[14] Il s'agit de la direction orientée vers l'observateur et perpendiculaire aux deux axes dominants de l'ensemble.

telle déformation à l'aide d'une interpolation par splines en plaques minces [Wahba 1990] avec la correspondance suivante : un point à chaque coin correspondant à lui-même, et un point au milieu correspondant à un point au-dessus de lui. En appliquant ceci au niveau le plus détaillé, nous obtenons le résultat de la figure 6.8 (i), ce qui est loin de ce à quoi nous pourrions nous attendre. Par contre, en l'appliquant au niveau 0, puis en reconstruisant le niveau le plus détaillé, nous obtenons le résultat de la figure 6.8 (j), ce qui est plus cohérent. Le résultat de la figure 6.8 (i) est obtenu car la correspondance n'est pas assez fine pour indiquer comment devraient se déplacer les points au-dessus et au-dessous du plan supportant les points initiaux de la correspondance. Autrement dit, pour réduire le travail à effectuer pour bien établir les paramètres d'une déformation, nous avons intérêt à réduire les détails géométriques à l'échelle appropriée.

Deux autres exemples de déformations utilisant l'approche multirésolution avec notre représentation sont affichés aux figures 6.8 (h) et 6.9 (b). Pour tous nos résultats, nous avons employé l'option de la mise à l'échelle adaptative des détails géométriques décrite à la §6.4.2, que nous pouvons observer, particulièrement à la figure 6.9 (b). De plus, la figure 6.8 (h) montre que notre représentation préserve assez bien les bordures.

Une autre façon d'effectuer une opération d'édition sur des objets avec une représentation multirésolution est de jouer au niveau de l'ensemble des informations de détails, *i.e.* $D^{[1]}, \ldots, D^{[L]}$. De fait, nous avons mentionné à la §6.3.2 que les détails géométriques d'un niveau donné correspondent approximativement à la bande de fréquences des détails du niveau. Comme le font Pauly *et al.* [2006] et Zhang *et al.* [2005], nous pouvons nous servir de ceci pour ajouter de l'accentuation ou lisser différentes caractéristiques de la surface en multipliant les détails géométriques par une valeur scalaire donnée[15]. Les figures 6.9 (c) et 6.10 (f) montrent certains résultats obtenus par ce type d'opération. Les valeurs de mise à l'échelle indiquent le facteur de multiplication pour chaque niveau, du plus grossier au plus fin. Par exemple, à la figure 6.10 (f), les détails géométriques des deux niveaux plus grossiers sont multipliés par deux, *i.e.* la valeur δ de chaque $\boldsymbol{d}_i^{[1]} \in D^{[1]}$ et $\boldsymbol{d}_i^{[2]} \in D^{[2]}$ (équation (6.18)) est multipliée par deux. Ceci accentue les basses fréquences, donnant ainsi l'impression que l'objet est gonflé. Comme l'indiquent Pauly *et al.* et Zhang *et al.*, cette mise à l'échelle peut aussi être utilisée pour obtenir une résolution de détails continue en multipliant les valeurs de détails géométriques progressi-

[15]Nous rappelons que les détails géométriques correspondent à la valeur δ (ou $\frac{\delta}{\Delta_\mathcal{T}}$, selon l'option) du nuplet d'information de détail (équation (6.18)).

vement par un facteur allant de zéro à un. Également, il n'y a aucune obligation d'effectuer ce type d'opération globalement sur tout un niveau.

6.6 Discussion et extensions

Pour résumer notre représentation multirésolution, nous récapitulons d'abord la représentation multiéchelle de Pauly *et al.* [2006] et Zhang *et al.* [2005]. Il s'agit de construire une série de surfaces de plus en plus lisses, où les surfaces plus détaillées sont exprimées de façon relative aux surfaces plus grossières. La reconstruction des surfaces plus détaillées s'effectue en *réadditionnant* les détails géométriques. Cependant, pour leur représentation, l'échantillonnage des surfaces ne change pas, ce qui résulte en un suréchantillonnage des surfaces plus grossières. Notre représentation multirésolution reprend les mêmes principes, mais elle ajoute une réduction du nombre de points échantillonnant les surfaces plus grossières. Afin d'éviter les problèmes associés aux méthodes d'enrichissement d'échantillonnage basées sur les voisinages implicites, notre représentation ajoute à l'information de détail géométrique des informations de connectivité calculées à partir d'une élaboration des idées de Boubekeur *et al.* [2007]. Ces informations sont utilisées lors de la reconstruction pour échantillonner une surface plus grossière de façon cohérente avec l'information de détail géométrique.

Les informations de connectivité calculées ne correspondent pas à construire un maillage, mais à identifier trois points voisins de la surface du niveau de résolution inférieur formant un plan approximant grossièrement la tangente de la surface. Ainsi aucun traitement global n'est requis, contrairement à une triangulation. Il n'est pas nécessaire non plus de stocker comme information de connectivité le voisinage complet d'un point comme le font Linsen et Prautzsch [2001, 2003]. Bien que cette dernière approche offre une grande robustesse suite aux déformations pour calculer la position initiale d'un échantillon durant la reconstruction, le coût supplémentaire en mémoire est non négligeable, sans compter les problèmes associés à la façon de représenter l'information de détail géométrique[16].

Dans les sections qui suivent, nous discutons de notre représentation, souvent en comparaison avec les représentations multiéchelles (§5.3.3), des avantages aux limites, ainsi que d'extensions possibles.

[16]Par exemple, si le point généré par l'information de voisinage est le centroïde de ce dernier et que le détail géométrique est représenté par une élévation au-dessus de la surface, des problèmes d'échantillonnage peuvent surgir car il n'est pas possible d'assurer que le point reconstruit retournera à sa position d'origine.

6.6.1 Réduction du nombre de points

Les objectifs principaux de réduire le nombre de points aux niveaux plus grossiers sont de limiter l'encombrement mémoire de la représentation et d'accélérer les traitements de géométrie. Nous devons nous poser la question si la représentation multirésolution présentée dans ce chapitre atteint ces objectifs.

D'abord, si nous comparons l'information de détail des représentations de Pauly *et al.* [2006] et de Zhang *et al.* [2005] avec la nôtre, les premières requièrent seulement un scalaire par élément de détail, alors que la nôtre requiert trois indices (des entiers) et trois scalaires (ou même quatre) par élément de détail. Plus précisément, soient L le nombre de niveaux de décomposition, s l'encombrement mémoire d'un scalaire et i celui d'un indice, alors l'encombrement mémoire m_{PZ} des méthodes de Pauly *et al.* et Zhang *et al.* est

$$m_{\mathrm{PZ}} = L|P^{[L]}|s + 6s|P^{[L]}| = (L+6)|P^{[L]}|s \tag{6.42}$$

puisque $|P^{[L]}| = |P^{[L-1]}| = \ldots = |P^{[0]}|$, et en supposant que chaque point est accompagné d'une normale. L'encombrement mémoire m de notre représentation, sans utiliser le scalaire supplémentaire par élément de détail, est

$$m = 6s|P^{[0]}| + 3(i+s)\sum_{l=1}^{L}|P^{[l]}|. \tag{6.43}$$

Afin de simplifier la comparaison, supposons que $i = s$ [17]. Nous obtenons alors

$$m = 6s\sum_{l=0}^{L}|P^{[l]}|. \tag{6.44}$$

Dans un cas idéal, $|P^{[l-1]}| = \frac{1}{\gamma}|P^{[l]}|$, où γ est la taille cible des agrégats lors de la simplification (§6.3.1, équations (6.5) et (6.11)), ce qui nous donnerait

$$m = 6s|P^{[L]}|\sum_{l=0}^{L}\frac{1}{\gamma^l} \tag{6.45}$$

$$= 6s|P^{[L]}|\frac{1-\frac{1}{\gamma^{L+1}}}{1-\frac{1}{\gamma}} \leq 6s|P^{[L]}|\frac{\gamma}{\gamma-1}. \tag{6.46}$$

[17] Il s'agit d'une simplification raisonnable, car la plupart du temps, $i \leq s$. Sur les architectures d'ordinateur 32 bits courantes, cette égalité est exacte si nous employons des nombres en virgule flottante de simple précision (quatre octets).

Ainsi, en comparant les équations (6.42) et (6.46), nous obtenons

$$\begin{aligned} m \leq 6s|P^{[L]}|\frac{\gamma}{\gamma-1} \leq m_{\mathrm{PZ}} &\Leftrightarrow 6s|P^{[L]}|\frac{\gamma}{\gamma-1} \leq (L+6)|P^{[L]}|s \\ &\Leftrightarrow 6\frac{\gamma}{\gamma-1} \leq L+6 \\ &\Leftrightarrow L \geq 6\left(\frac{\gamma}{\gamma-1}-1\right) \end{aligned} \tag{6.47}$$

ce qui nous donne une borne inférieure sur le nombre de niveaux minimum pour que notre représentation ait un encombrement mémoire moindre que celles de Pauly *et al.* et Zhang *et al.* Cependant, l'équation (6.45) est généralement plutôt

$$m \geq 6s|P^{[L]}|\sum_{l=0}^{L}\frac{1}{\gamma^l}. \tag{6.48}$$

Mais les données empiriques du tableau 6.1 nous portent à croire que nous pouvons généralement trouver un $\gamma' < \gamma$ tel que

$$m \leq 6s|P^{[L]}|\sum_{l=0}^{L}\frac{1}{\gamma'^l} \tag{6.49}$$

ce qui maintient que l'encombrement mémoire de notre représentation demeure asymptotiquement inférieur à celui des représentations de Pauly *et al.* et Zhang *et al.* en fonction du nombre de niveaux. Par exemple, si $s = 4$ [18], alors pour l'ensemble de points "Tatou", suivant l'équation (6.44), m = 5 832 336, mais selon l'équation (6.45), m = 5 180 918 ($\gamma = 5$). En choisissant $\gamma' = 3$, alors selon l'inéquation (6.49), $m \leq$ 6 150 187. Or, m_{PZ} = 6 227 064, et alors même l'estimation avec γ' est meilleure.

Une analyse similaire peut être faite si chaque élément de détail de notre représentation contenait plutôt quatre scalaires. Nous obtiendrions alors une borne inférieure légèrement plus grande.

En ce qui concerne l'accélération des traitements de géométrie, le gain que procure notre représentation se trouve au niveau de l'édition au niveau le plus grossier. Le nombre de points que nous pouvons obtenir à ce niveau peut permettre d'effectuer des opérations complexes interactivement, *i.e.* afin qu'un usager d'un système interactif puisse visualiser immédiatement

[18] Ceci correspondrait à la valeur pour la plate-forme sur laquelle nous avons effectué nos expérimentations si nous avions utilisé des nombres en virgule flottante de simple précision. Pour les nombres en double précision, $s = 8$, mais $i = 4$, et alors l'écart entre les représentations multiéchelles et la nôtre se creuse davantage en faveur de la nôtre.

les effets de ses opérations. Cependant, ceci n'est vrai qu'au niveau le plus grossier[19]. En effet, les temps de reconstruction du niveau le plus grossier au niveau le plus fin empêchent un usage pleinement interactif de notre représentation. L'usager doit patienter un peu avant de pouvoir visualiser le résultat de son opération au niveau le plus fin. Nous croyons qu'un meilleur support pour l'édition interactive peut être obtenu en ajoutant la notion de multirésolution adaptative à notre représentation, de façon similaire aux travaux de Zorin *et al.* [1997] (§5.2.3), où la décomposition et la reconstruction sont effectuées adaptativement en fonction de la courbure locale de la géométrie. De plus, pour notre représentation, chaque point est traité indépendamment à la plupart des étapes, ce qui fait en sorte que la plupart des calculs peuvent être effectués en parallèle, soit à l'aide des processeurs multi-cœurs modernes, où par l'intermédiaire de méthodes de calculs génériques sur processeur graphique [Owens *et al.* 2007]. En particulier, l'une des opérations les plus coûteuses est la projection MCM, mais des implémentations sur processeur graphique ont récemment été proposées [Guennebaud et Gross 2007, Guennebaud *et al.* 2008]. Il est à noter que ce type d'optimisation est également applicable pour les autres représentations multirésolutions par points, incluant les représentations multiéchelles [Pauly *et al.* 2006, Zhang *et al.* 2005].

6.6.2 Robustesse et améliorations possibles

Un autre aspect important à discuter est la robustesse de notre représentation. Nous avons mentionné que les représentations multiéchelles sur lesquelles se base notre représentation sont parmi les plus robustes, ce qui veut dire que la reconstruction donne généralement de meilleurs résultats que les autres représentations pour une gamme plus grande de déformations de la géométrie du niveau plus grossier ou lisse, ou de transformations des informations de détails. Ainsi, notre représentation devrait partager la même robustesse. Cependant, dans leur cas, le nombre de points ne change pas d'un niveau à l'autre, et donc elles n'ont pas, *a priori*, de problèmes dans les régions de formes allongées ou de courbures élevées (figure 6.3), pourvu que l'échantillonnage initial soit suffisant. Mais si le lissage est effectué par projection MCM, alors les mêmes problèmes pouvant survenir avec notre représentation en lien à la réduction du nombre de points existent aussi dans leur cas. Ainsi, la modification que nous avons apportée à la mise en agrégats (équation (6.17)) peut rendre notre représentation plus ou aussi robuste que les leurs[20].

[19] Nous devrions peut-être plutôt dire aux quelques niveaux les plus grossiers.

[20] Elle serait aussi robuste si nous leur appliquions la même modification.

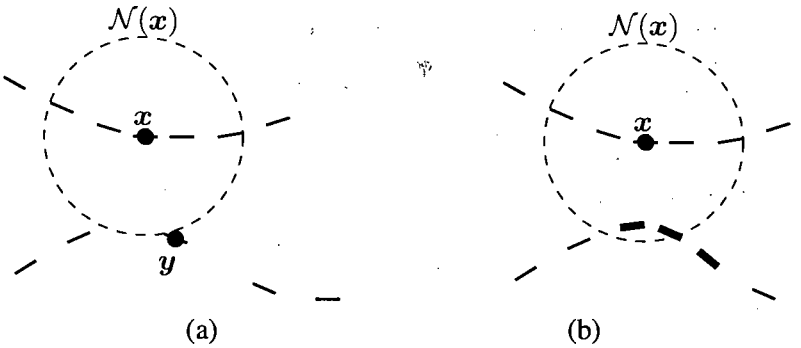


FIG. 6.12 – Subdivision d'un agrégat nuisant à un autre voisinage : (a) l'agrégat de centroïde y est subdivisé, (b) mais il est possible que ses enfants, indiqués par les trois segments plus épais, tombent dans le voisinage de x, causant soudainement une plus grande variation de surface et de normales autour x.

Le fait de partager une robustesse similaire aux représentations multiéchelles ne signifie pas qu'il n'y a pas place à amélioration. Nous avons effectué nos expérimentations de façon similaire aux autres travaux, sur des objets similaires et avec des opérations similaires. Ainsi, nos ensembles de points échantillonnent bien la surface des objets et contiennent peu ou pas de bruit, leur densité est assez uniforme, et les opérations effectuées ne sont pas extrêmes. Donc, la plupart des heuristiques que nous utilisons, notamment lors du raffinement effectué pendant la génération du niveau plus grossier (§6.3.1) et lors de la sélection du triangle (§6.3.2.2) sont bien adaptées à ces situations.

D'abord, à propos du raffinement (§6.3.1, équation (6.14)), la méthode se base sur l'hypothèse que la subdivision d'un agrégat (équation (6.15)) peut seulement améliorer la qualité des voisinages. Or, ceci peut être faux, particulièrement en considérant la remarque 6.2. La figure 6.12 montre un exemple du problème. Cependant, en pratique, il est rare que ça ne soit pas le cas, ce qui rend l'hypothèse raisonnable. De plus, dans notre implémentation, l'évaluation de l'équation (6.14) est effectuée dans un style jacobien, *i.e.* que tous les candidats à subdiviser sont déterminés d'abord, et ensuite ils sont tous subdivisés, ce qui pourrait engendrer un nombre un peu trop grand de subdivisions. Ceci réduit la possibilité de subdivisions nuisibles pour les voisins, sans toutefois les empêcher. Il n'est pas clair comment améliorer le raffinement pour éviter ce problème sans faire usage de voisinages symétriques (§2.3.1.1, équation (2.4)) ou sans accroître considérablement les temps de calcul.

Quant à la sélection de triangle (§6.3.2.2), aucune de nos heuristiques ne garantit de toujours obtenir un triangle aux propriétés désirées, soit le plus petit, équilatéral et tangent à la surface

que possible. Il est encore plus difficile de le garantir aux bordures et dans les zones de discontinuités de densité. Les discontinuités de densité, pour la plupart dues aux raffinements pendant la génération du niveau plus grossier, sont particulièrement problématiques. Nous avons décrit comment notre algorithme de sélection détecte et gère les points dans ces zones, mais le résultat, quand le point ne se fait pas considérer comme frontière, est que les calculs se font sur un voisinage de grande taille dont la distribution des points peut certainement ressembler aux figures 6.3 (a) et (b). Il faudrait plutôt procéder suivant le principe de la subdivision adaptative pour les surfaces de subdivision [Zorin *et al.* 1997, Kobbelt 2000] lors du raffinement. La sélection de triangle est aussi la partie la plus sensible au bruit[21]. Pour la rendre plus robuste, il faudrait ajouter un critère explicite de différence entre la normale du triangle sélectionné et la normale de la surface à proximité, et effectuer un retour arrière au besoin. Le problème est alors de trouver le bon compromis entre le temps de calcul et la qualité du résultat, comme c'est toujours le cas lorsqu'il est question d'heuristiques.

Un autre point à discuter au niveau de la robustesse est l'ampleur des déformations. Nous avons en effet mentionné que, de la même façon que les autres auteurs, les opérations effectuées dans nos expérimentations n'étaient pas extrêmes. Mais qu'adviendrait-il si nous effectuions des déformations plus extrêmes ? Nous n'aurions pas de problèmes pour générer la position de base (étape 2 de la reconstruction, §6.4), mais nous pourrions en avoir dans les étapes subséquentes à cause de la dépendance sur l'estimation de normales. En effet, si la déformation est très anisotropique, les conditions d'échantillonnage locales sont moins bonnes, *i.e.* nous sommes loin d'un échantillonnage$-(\epsilon, \delta)$ (équation (2.11), §2.3.1), ce qui fait en sorte qu'un voisinage-k peut avoir une distribution davantage prononcée dans l'axe perpendiculaire à l'étirement causé par la déformation. La qualité des normales estimées peut donc en être grandement réduite. Par contre, ce problème n'est pas particulier à notre représentation, mais à toutes celles qui font usage de voisinages implicites, y compris celles de Pauly *et al.* [2006] et Zhang *et al.* [2005].

La solution serait de procéder comme Pauly *et al.* [2003b] durant la déformation, *i.e.* enrichir l'échantillonnage dans les zones de grands étirements. Pour les méthodes multiéchelles (§5.3.3), l'enrichissement se ferait d'abord au niveau zéro, puis les détails géométriques des nouveaux points du niveau un seraient interpolés. En reconstruisant les autres niveaux, les détails géométriques correspondants seraient aussi interpolés en fonction des nouvelles positions obtenues. Avec notre représentation, ça serait plus compliqué car le nombre de points varie

[21] Par bruit, nous entendons des perturbations sur la position et la normale des points.

d'un niveau à l'autre. Néanmoins, nous pourrions commencer par enrichir l'échantillonnage au niveau zéro, et ensuite utiliser une technique similaire à celle de Fleishman *et al.* [2003a] (§5.3.2.1) pour localement échantillonner les trous aux niveaux supérieurs. Cependant, il n'est pas clair comment ajuster les informations de connectivité des anciens points efficacement, si toutefois cet ajustement était vraiment nécessaire.

6.6.3 Travaux futurs

Il existe plusieurs voies d'exploration pour soit raffiner notre représentation multirésolution, soit explorer d'autres façons pour obtenir une représentation multirésolution avec niveaux plus lisses et grossiers. Dans cette section, nous discutons de quelques possibilités.

6.6.3.1 Topologie

Il est possible que la surface détaillée d'un objet soit de genre[22] supérieur à zéro, tel qu'illustré par certains exemples de textures géométriques du chapitre 4. Par contre, une surface similaire mais moins détaillée pourrait avoir un genre inférieur. Une représentation multirésolution devrait donc pouvoir permettre des changements de genre entre les niveaux de résolution, comme c'est possible avec les surfaces implicites multiniveaux ou multiéchelles [Ohtake *et al.* 2003a, Ohtake *et al.* 2003b, Ohtake *et al.* 2005b], ou même avec les maillages [Guskov *et al.* 2002], quoique de façon moins naturelle puisque les changements de topologie doivent être explicitement représentés. *A priori*, il n'y a rien qui empêche les changements de genre avec les surfaces MCM. Cependant, dans le contexte de notre représentation multirésolution, il est difficile d'obtenir des changements de genre à cause du raffinement de l'équation (6.14), à moins que soudainement le lissage soit localement suffisamment fort pour enlever les poignées[23]. Il s'agit donc d'une voie intéressante à suivre pour rendre notre représentation plus puissante et robuste face à ce type de détail géométrique.

6.6.3.2 Encodage plus robuste

Une analogie peut être faite entre notre représentation multirésolution et la compression utilisant un encodage par delta (*delta encoding*) [Sayood 2000]. Dans les grandes lignes, pour un signal temporel, ce type de compression procède ainsi :

[22] Le *genre* d'une surface close est essentiellement le nombre de *trous*, *i.e.* le nombre maximum de courbes simples fermées pouvant être tracées à l'intérieur de la surface sans qu'elles n'aient de points communs [do Carmo 1976].

[23] Les *poignées* correspondent aux *trous* topologiques.

1. Soit t le temps courant. Construire un *prédicteur* pour le signal au temps $t + 1$ étant donnés les échantillons rencontrés précédemment, par exemple un prédicteur linéaire.
2. Calculer la différence entre le signal réel et le signal prédit.
3. Quantifier et stocker cette différence.
4. Recommencer à l'étape 1 à partir du temps $t + 1$, et ainsi de suite jusqu'à la fin du signal.

Pour décoder le signal, il s'agit de suivre une procédure très similaire en remplaçant les étapes 2 et 3 par les opérations inverses de chacune et d'effectuer l'étape 3 avant l'étape 2. Par contre, il y a un problème possible entre les étapes 3 et 4 dans la simple procédure d'encodage décrite ci-avant. En effet, à l'étape 3, il y a une quantification et donc une erreur est induite. Étant donné que le décodeur reconstruit seulement à partir des prédicteurs, pour éviter de propager l'erreur, la prédiction de l'étape 1 doit être faite à partir du signal reconstruit, qui inclut cette erreur, et non à partir du signal original.

Pour notre représentation, au lieu de parler de temps t, il est plutôt question de niveau l, et plutôt que de parler de signal, il est question de surface. Par contre, nous construisons l'encodage par delta en sens inverse, *i.e.* du niveau l au niveau $l - 1$. Pour que l'analogie soit plus forte, supposons que nous ayons calculé toutes les surfaces des niveaux $L - 1$ à 0 à partir de L, ce qui correspond à répéter les calculs de la §6.3.1 sans ceux de la §6.3.2. Alors, reprenant les termes de la procédure de compression du signal temporel, la procédure pour encoder notre représentation serait la suivante :

1. Soit l le niveau courant. Construire la surface au niveau l, *i.e.* le *prédicteur* pour la surface du niveau $l + 1$.
2. Calculer la différence entre la surface réelle du niveau $l + 1$ et la surface prédite.
3. Stocker cette différence.
4. Recommencer à l'étape 1 à partir du niveau $l + 1$, et ainsi de suite jusqu'au niveau L.

Pour notre représentation telle que décrite dans ce chapitre, le prédicteur de l'étape 1 est la surface produite par la décomposition. Cependant, à l'étape 3, nous n'effectuons aucune quantification[24]. En compression, cette quantification est la principale source d'erreurs. Par contre, dans notre cas, la nature des erreurs est autre : elle provient des erreurs produites par toutes les procédures itératives. Ainsi, de façon similaire à l'encodage par delta, le prédicteur de l'étape 1

[24]Il est à remarquer que nous pourrions le faire pour compresser notre représentation. Il s'agit d'autres voies à explorer.

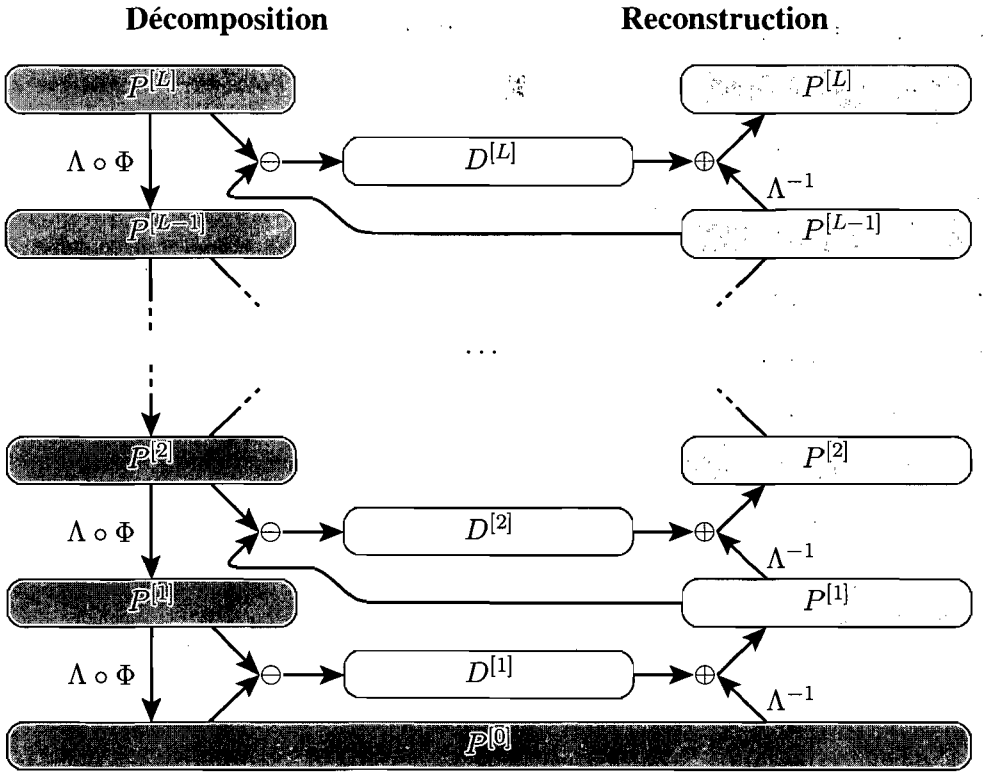


FIG. 6.13 – Encodage robuste des informations de détails.

devrait provenir de la reconstruction. La figure 6.13 montre le nouveau schéma de notre représentation suivant ce principe. Évidemment, le problème principal associé à cette modification est l'augmentation considérable du temps de calcul pour la décomposition, *i.e.* le calcul du niveau zéro et des détails $D^{[l]}$, $l \in \mathbb{Z}[1, L]$ et il reste à voir si le gain de précision en vaut le coût.

6.6.3.3 Représentation moins lourde en mémoire

La représentation multirésolution présentée dans ce chapitre est moins lourde en mémoire que celle de Linsen et Prautzsch [2003], et même que les représentations multiéchelles [Pauly *et al.* 2002b, Pauly *et al.* 2006, Zhang *et al.* 2005] pour un nombre suffisamment grand de niveaux (§6.6.1). Mais, évidemment, la question suivante se pose : pourrions-nous faire mieux, *i.e.* pourrions-nous obtenir une représentation multirésolution de surface MCM pour laquelle le nombre de points diminue pour chaque niveau plus grossier, sans faire usage de compression ? Pour chaque élément de détail géométrique dans notre représentation, les cinq sixièmes de l'information sont consacrés à la connectivité partielle (équation 6.18). Donc, pour obtenir un gain significatif d'économie de mémoire, il faut chercher à réduire cette portion.

Les représentations multiéchelles s'en tirent à raison d'un seul scalaire par élément de détail car elles n'ont pas besoin de cette information de connectivité, puisque le nombre de points ne change pas d'un niveau à l'autre. La représentation progressive de Fleishman *et al.* [2003a] (§5.3.2.1) s'en tire également avec un seul scalaire par élément de détail, et le nombre de points augmente avec la résolution. Cependant, pour en arriver là, l'opérateur d'enrichissement d'échantillonnage se base sur les voisinages implicites, et alors cette représentation ne peut pas servir pour effectuer des déformations. Mais peut-être pourrions-nous utiliser ce type de représentation couplé avec une forme de transfert de détails géométriques pour ainsi obtenir une représentation multirésolution avec un seul scalaire par élément de détail, et qui permettrait aussi d'effectuer des déformations. Nous décrivons dans ce qui suit l'idée d'une telle représentation.

La décomposition se ferait suivant une approche similaire à la représentation de Fleishman *et al.* [2003a]. Des ensembles de points $P^{[L-1]}, P^{[L-2]}, \ldots, P^{[0]}$ seraient générés à partir de l'ensemble de points initial $P^{[L]}$ par une série de simplifications, possiblement en réutilisant la méthode de la §6.3.1. Ensuite, les détails $D^{[l]}$, $l \in \mathbb{Z}[1, L]$, seraient calculés, du niveau 1 au niveau L, en appliquant un opérateur d'enrichissement d'échantillonnage et calculant une différence entre les nouveaux points et $\mathcal{S}_{P^{[l]}}$, ce qui n'est pas tellement différent de l'idée décrite à la section précédente. Il est à noter cependant que, comme c'est aussi le cas de la représentation de Fleishman *et al.*, l'ensemble de points reconstruit n'aura pas exactement le même échantillonnage que l'ensemble de points initial. Au final, la représentation conserverait les détails $D^{[l]}$, $l \in \mathbb{Z}[1, L]$, l'ensemble de points $P^{[0]}$, mais aussi une copie de $P^{[0]}$, que nous noterons $\hat{P}^{[0]}$, qui ne pourrait pas être modifiée, *i.e.* les déformations seraient effectuées sur $P^{[0]}$.

En dehors de la copie $\hat{P}^{[0]}$ dans la représentation, la différence principale avec la représentation de Fleishman *et al.* se situerait au niveau de la reconstruction. Deux ensembles de points seraient reconstruits à partir d'un niveau $l-1$, $l \in \mathbb{Z}[1, L]$: $P^{[l]}$, qui formera l'ensemble de points final au niveau L, et qui aurait pu subir une déformation quelconque, mais aussi $\hat{P}^{[l]}$, qui correspondrait à la reconstruction de l'ensemble de points non déformé et qui servirait de base pour calculer $P^{[l]}$. L'idée est donc de transférer la différence entre $\hat{P}^{[l-1]}$ et $\hat{P}^{[l]}$ pour produire $P^{[l]}$ à partir de $P^{[l-1]}$. Une façon de faire possible serait la suivante : pour chaque niveau produit par l'enrichissement d'échantillonnage de $\hat{P}^{[l-1]}$, une position correspondante serait produite à partir de $P^{[l-1]}$, et ensuite, le détail géométrique serait ajouté sur chaque point. Pour produire la position correspondante, il s'agirait de trouver une méthode rapide pour représenter le nouveau

point depuis $\hat{P}^{[l-1]}$ intrinsèquement et substituer $\hat{P}^{[l-1]}$ par $P^{[l-1]}$. Au final, seul l'ensemble de points $P^{[L]}$ est d'intérêt, le reste serait seulement un résultat temporaire.

Un avantage d'une telle représentation, en dehors de l'économie substantielle de mémoire, serait un gain en robustesse face aux déformations plus fortes, car il serait possible de toujours évaluer les voisinages à partir des ensembles de points $\hat{P}^{[l]}$ pour tout point dans les ensembles de points $P^{[l]}$, $l \in \mathbb{Z}[1, L]$. En effet, un voisinage est en réalité implémenté par une simple liste d'indices dans un ensemble, et ce voisinage peut donc être utilisé pour les deux catégories d'ensembles de points aisément, pourvu qu'il y ait une correspondance entre eux, ce qui serait le cas. Ainsi, la reconstruction des ensembles de points $\hat{P}^{[l]}$ pourrait être vue comme un domaine paramétrique, de façon très similaire à la notion de domaine paramétrique *naturel* pour les surfaces de subdivision [Zorin 1998].

Cependant, suivant la description que nous avons donnée de la reconstruction, nous devrions nous attendre à ce que les temps de calcul soient considérables et sûrement même plus grands que ceux obtenus pour notre représentation. De plus, l'espace mémoire requise *pendant* la reconstruction serait plus grande, car nous devrions maintenir deux ensembles de points. Peut-être qu'en explorant davantage cette ligne d'idées, nous pourrions arriver à obtenir une reconstruction avec moins de calculs, ou sinon, le gain d'économie en mémoire[25] pourrait malgré tout compenser pour les temps de reconstruction plus lents.

[25] Il est à noter que la copie supplémentaire $\hat{P}^{[0]}$ dans la représentation n'est pas significative au niveau de la consommation de mémoire, car le nombre de points au niveau 0 est généralement assez petit (voir le tableau 6.1).

Chapitre 7

Conclusion

Aujourd'hui, il est possible de retrouver chez un consommateur moyen un ordinateur dont la puissance de calcul surpasse celle des ordinateurs hauts de gamme d'il y a à peine quelques années[1]. Ceci implique, entre autres, qu'il est possible de livrer des jeux sur ordinateur visuellement de plus en plus riches, et que les studios de production de cinéma ou d'animation peuvent aisément s'équiper pour produire des films comportant des effets spéciaux de plus en plus impressionnants, ou des animations de grande qualité visuelle. Un des éléments importants pour évaluer la qualité visuelle d'une image repose sur la richesse des détails présents. La nature de ces détails est variée, mais un type souvent plus difficile à traiter est le détail géométrique. En effet, les outils actuels demandent souvent un travail fastidieux pour parvenir à produire un objet comportant de nombreux détails géométriques fins, ou alors ils sont plutôt spécialisés pour une catégorie plus limitée de détails.

Ainsi, nous avons présenté dans cette thèse deux contributions principales touchant la génération et l'édition de détails géométriques, sous la forme de textures géométriques, pour des surfaces représentées par des ensembles de points. Notre choix de nous baser sur les ensembles de points pour représenter la géométrie des objets est principalement justifié par la simplicité de la représentation lorsqu'il est question d'effectuer des opérations de jonction ou de rééchantillonnage, ce qui est aussi appuyé par l'expérience de certains travaux de complétion de surfaces [Sharf *et al.* 2004, Park *et al.* 2005] (§3.2.3).

Pour notre première contribution, inspirée par les méthodes de génération de textures par *patch* (§3.1.3), nous avons présenté un nouvel algorithme de génération de textures géomé-

[1] À l'exclusion des superordinateurs.

triques (chapitre 4) qui, à partir d'un exemple, génère une nouvelle texture géométrique de taille arbitraire qui est visuellement similaire à l'exemple, sans pourtant en être un pavage. Comme pour la plupart des méthodes de génération de textures, les textures géométriques que notre algorithme peut bien générer sont celles qui sont la réalisation d'un champ aléatoire de Markov (*Markov random field* ou MRF). Même s'il ne s'agit pas d'un outil de création complètement général, les champs aléatoires de Markov couvrent tout de même un très large éventail de textures, et l'usage d'algorithmes tels que le nôtre est généralement plus simple que l'usage des modèles se voulant aussi généraux, notamment en comparaison aux systèmes-L (§2.1.2.2). Les résultats que nous avons obtenus indiquent que le choix des ensembles de points est non seulement valide mais profitable en terme d'efficacité lorsque nous comparons notre algorithme avec les travaux similaires, mais basés sur les maillages, de Zhou *et al.* [2006] (§3.2.3).

Notre seconde contribution touche un autre aspect de la modélisation de détails géométriques, soit, l'édition efficace d'un objet en présence de tels détails, voire l'édition de ces détails eux-mêmes. En effet, il est généralement souhaitable de pouvoir modifier un objet à toute étape de sa production, et donc aussi lorsque tous les détails géométriques ont été générés, par exemple avec une approche comme celle de notre première contribution. Sur ce sujet, beaucoup de travaux ont été effectués pour les maillages (§5.2 et §5.4), mais peu concernant les représentations par points. Une méthode bien connue pour permettre une telle édition est d'employer des représentations multirésolutions, pour lesquelles les détails sont découplés de la surface, et ce, à plusieurs niveaux. Les représentations multirésolutions permettent de choisir le niveau de détails désiré pour une opération et, si les détails sont représentés de façon adéquate, elles permettent aussi d'éditer directement les détails sans avoir à se soucier de la géométrie de la surface sous-jacente elle-même. Nous avons donc présenté une représentation multirésolution pour des surfaces représentées par des ensembles de points (chapitre 6) qui permet d'effectuer les opérations classiques associées à la modélisation géométrique multirésolution. La plupart des travaux touchant les représentations multirésolutions par points ne permettent pas d'effectuer toute la gamme des opérations (§5.3), et en comparaison aux autres, notre représentation est généralement plus robuste, est constituée de structures moins lourdes que les ensembles d'éventails de triangles (§5.3.2.3), et est plus efficace que les surfaces multiéchelles (§5.3.3), notamment parce que le nombre de points est réduit pour les niveaux de résolution plus grossière.

Du point de vue du thème plus global de cette thèse, soit la création et l'édition de textures géométriques, nous considérons notre seconde contribution comme plus fondamentale. En effet,

plus d'information est requise pour bien représenter les détails géométriques, et alors la complexité des calculs pour les traiter est accrue. Or, l'usage de représentations multirésolutions est une approche classique pour améliorer l'efficacité des calculs, sans compter la possibilité d'améliorer la qualité des résultats, tel que sera discuté dans la prochaine section. Ainsi, un ordre de présentation plus approprié aurait été l'inverse de celui de cette thèse. Cependant, c'est le développement de la première contribution, et aussi lorsqu'il était question de traiter les textures géométriques produites, qui nous a menés à la nécessité et au développement de la seconde.

7.1 Travaux futurs

Nous avons déjà discuté des problèmes, des améliorations et des travaux futurs pour chacune de nos contributions à la fin de leur chapitre respectif, et nous n'allons pas les reprendre ici. Néanmoins, il y a quelques directions de recherche ou de travaux futurs qui touchent plus à l'ensemble des travaux de cette thèse. Nous les décrivons dans les prochaines sections. Il est à noter que la plupart des idées mentionnées ci-après sont, bien entendu, de nature plus spéculative.

7.1.1 Génération multirésolution

Une étape importante dans la plupart des algorithmes de génération de textures géométriques et de géométrie, y compris notre algorithme, est la déformation d'une *patch* pour l'ajuster à son voisinage, ce qui implique dans plusieurs cas la minimisation d'une fonction définie à partir d'un pairage entre des points de la *patch* et du voisinage. Plusieurs caractéristiques différentes des surfaces doivent être prises en compte lors de la sélection du pairage pour que la déformation résultante soit adéquate. Dans le cas de notre algorithme de génération, nous avons considéré la distance entre les points, la différence des normales et la différence de la variation de surface. Cependant, pour accommoder divers types de surfaces, la pondération entre ces termes doit être ajustée, ce qui n'est pas toujours simple et intuitif à régler. Suivant les principes de fonctionnement des algorithmes de génération de textures géométriques, cette complexité est nécessaire pour tenir compte des textures géométriques de topologie arbitraire.

D'un autre côté, les détails géométriques à petite échelle peuvent la plupart du temps être bien représentés par des cartes de déplacement. Pour ce type de texture, les méthodes de génération de textures colorées sont déjà reconnues pour être très efficaces. Or, il se trouve qu'une

bonne partie de la complexité de la sélection des poids appropriés dans le calcul du pairage provient de la présence possible de ce type de détail. C'est alors que vient l'idée d'une génération multirésolution : en découplant les détails géométriques et la surface de base plus grossière, nous pourrions grandement simplifier la génération de la texture géométrique sans ses détails. En effet, le nombre de points serait réduit, en supposant une décomposition multirésolution comparable à celle que nous avons présentée dans cette thèse, et la sensibilité aux variations locales serait considérablement moindre. Ensuite, les détails pourraient être générés suivant une des méthodes basées sur la génération de textures colorées.

Cependant, comme nous l'avons déjà souligné à la fin de la §4.6.3 pour la génération de textures géométriques avec attributs de couleur, cette génération ainsi découplée ne sera pas nécessairement cohérente avec l'exemple fourni de texture géométrique, *i.e.* la distribution des détails sur la surface de base peut être en corrélation avec ses formes locales. Nous pourrions alors suivre la ligne d'idées que nous avons mentionnée dans le cas des attributs de couleur (analogie, pondération additionnelle), ou peut-être chercher à manipuler directement la représentation multirésolution elle-même, et ajouter une étape de correction des discontinuités lors de la reconstruction.

L'idée de génération multirésolution pourrait sans doute s'étendre à la génération de géométrie, notamment dans le cas de méthodes comme celle de Merrell [2007] (§3.2.4). Par contre, dans un tel cas, la "décomposition" serait plus une relation entre *modules* de différents niveaux de résolution plutôt qu'entre détails et surfaces. Par exemple, à un niveau, il pourrait y avoir des blocs donnant la forme générale d'édifices, et au niveau de détails suivant, il y aurait des modules de fenêtres, de murs de briques, de portes, etc., le tout en relation avec leur position sur les murs. Ainsi, la génération pourrait procéder en remplissant l'espace d'abord par des modules plus grossiers, et ensuite procéder avec des modules plus fins, en tenant compte des relations préétablies.

7.1.2 Compression de textures géométriques

La plus grande partie de l'information de la surface d'un objet concerne les détails géométriques. Il y a donc un intérêt pour la compression géométrique afin de réduire l'espace requis en archivage ou pour le transfert de données.

Récemment, Wei *et al.* [2008] ont présenté une méthode pour créer un exemplaire d'une texture colorée à partir d'une grande texture, *i.e.* une méthode de génération inverse. Ceci peut

certainement être vu comme une forme de compression de textures, seulement le *décompresseur* ne reproduit pas nécessairement la texture originale mais généralement plutôt une autre qui possède les mêmes propriétés visuelles, *i.e.* le même motif, la même distribution locale de caractéristiques, etc. Cette idée de génération inverse pourrait certainement s'appliquer aux textures géométriques, quoique comme nous l'avons présenté dans cette thèse, la manipulation de ce type de texture est plus complexe à cause de la plus grande sensibilité aux discontinuités. L'idée de génération multirésolution de la section précédente pourrait contribuer à simplifier le traitement.

L'application de la génération inverse a plus ou moins de sens si les détails géométriques de l'objet proviennent de l'application d'un algorithme de génération, à moins que des opérations d'édition aient été appliquées par la suite. Cependant, puisque la génération à partir de l'exemple extrait par la génération inverse ne reproduit pas l'original, le résultat ne serait peut-être pas suffisamment cohérent avec l'original, tout dépendant de la nature des opérations effectuées. D'un autre côté, une autre source d'objets détaillés est la numérisation 3D. Une texture géométrique n'est pas toujours identifiable sur toute partie d'un objet. Par exemple, supposons que nous ayons obtenu une numérisation du dragon de jade sculpté de la figure 1.1 dans notre introduction. Celle-ci possède une texture qui pourrait être extraite par génération inverse (les écailles sur le corps), mais tel n'est pas le cas au niveau de la tête. Néanmoins, il existe des similarités entre les détails de la crinière et ceux sur le bout de la queue du dragon. Nous pourrions songer à utiliser la génération inverse sur les régions applicables, et une autre méthode de compression pour le reste, mais ceci nous paraît difficile à faire sans intervention d'un usager pour identifier les régions.

Une alternative serait d'effectuer ce que nous pourrions appeler une pseudo-génération inverse, *i.e.* au lieu de retrouver un exemplaire, nous pourrions seulement extraire une collection de *patches* représentatives de la variété des détails présents et conserver la position et l'orientation de chacune sur une surface simplifiée de l'objet. La reconstruction consisterait alors à joindre les *patches* voisines une fois transformées à leur position et orientation sur l'objet, ce qui pourrait être effectué, par exemple, suivant une méthode telle qu'utilisée pour notre algorithme de génération. Une décomposition multirésolution pourrait servir pour extraire une surface de base et les détails. La problématique principale serait d'extraire une bonne collection de *patches*. Encore une fois, nous pourrions aussi appliquer cette compression en multirésolution, de façon similaire à la génération multirésolution de la §7.1.1.

7.1.3 Génération contrôlable

Nous avons présenté une méthode de génération de textures géométriques dans le but premier de faciliter la vie d'un artiste pour produire des surfaces détaillées en évitant la répétition. Cependant, il est rare qu'un objet soit recouvert entièrement de la même texture. Ainsi, il manque la notion de contrôle dans la génération, *i.e.* l'artiste devrait être en mesure de choisir les régions sur lesquelles appliquer ses textures, de déterminer l'orientation, l'échelle de l'application, la variation de l'échelle sur la surface, le niveau de résolution d'édition, etc.

Cette notion de génération contrôlable existe déjà pour les textures colorées [Ashikhmin 2001, Hertzmann *et al.* 2001, Kwatra *et al.* 2005, Lefebvre et Hoppe 2005, Lefebvre et Hoppe 2006] par l'intermédiaire de contraintes de couleurs ou de champs d'écoulement. Il pourrait être relativement facile de reprendre le concept d'analogies de Hertzmann *et al.* [2001, 2002] pour contrôler la disposition des textures. Par contre, un soin particulier serait nécessaire pour assurer la continuité des surfaces aux frontières des différentes régions. Ce problème de jonction pourrait d'ailleurs être mal posé si les régions aux frontières n'ont pas la même topologie. Par exemple, joindre une région avec une texture géométrique de bosses (figure 4.5) et une autre avec une texture géométrique de treillis (figure 4.8) ne peut pas être fait pour obtenir un objet bien formé, *i.e.* de variété. Alors, à défaut de pouvoir assurer la continuité, d'autres moyens devraient être donnés à l'artiste pour obtenir des jonctions visuellement acceptables. Par exemple, une région tampon pourrait être utilisée pour amoindrir les discontinuités, de façon similaire à une barre métallique entre un plancher en bois et un plancher recouvert de tapis.

De ceci, la mise en garde à retenir face à la génération contrôlable est de faire attention à ne pas aller à l'encontre du but d'alléger le travail de l'artiste. Autrement dit, les problèmes causés par les outils ne devraient pas exiger beaucoup de temps à réparer ou à contourner. Dans un monde idéal, tout serait une question de quelques manipulations intuitives et interactives.

7.2 La complexité des représentations par points

À travers cette thèse, nous avons souligné les avantages des représentations géométriques par points pour le traitement de détails géométriques, mais peu les inconvénients. Ainsi nous consacrons cette dernière section à une discussion sur les représentations géométriques par points elles-mêmes.

Les surfaces MCM ont plusieurs avantages intrinsèques, en particulier leur simplicité, continuité et résistance au bruit. Cependant, les deux derniers avantages ne sont pas inconditionnels. En effet, plusieurs propriétés des surfaces MCM ont été démontrées sous l'hypothèse d'un échantillonnage-(ϵ, δ) [Andersson *et al.* 2004, Bremer et Hart 2005, Dey et Sun 2005] ou, de façon moins stricte, d'un échantillonnage-ϵ [Kolluri 2005][2]. Pauly *et al.* [2004a] ont aussi présenté des mesures quantitatives sur l'incertitude et la variabilité des surfaces induites par des ensembles de points. En bref, la qualité de l'échantillonnage est importante pour obtenir de bons résultats, et c'est alors que nous pouvons questionner l'avantage de la simplicité. Par exemple, nous avons dû apporter des modifications à l'algorithme de simplification par agrégation de Pauly *et al.* [2002a] pour tenter de contrôler les problèmes d'échantillonnage (figure 6.3) sans garantie exacte sur le résultat. Un certain nombre de travaux ont mentionné l'importance des méthodes pour changer dynamiquement l'échantillonnage suivant des opérations d'édition ou pour des besoins de qualité de rendu [Pauly *et al.* 2003b, Guennebaud *et al.* 2004c, Guennebaud *et al.* 2004b, Guennebaud *et al.* 2005]. Cependant, peu de travaux, pour ne pas dire aucun, se sont attardés sur le maintien des hypothèses, par exemple d'un échantillonnage-(ϵ, δ), ni même sur la création d'un ensemble de points initial les respectant.

Là où la simplicité des représentations par points s'arrête, commence l'avantage des maillages[3] sur ces dernières. Par l'expression explicite de la connectivité entre les sommets échantillonnant une surface, les maillages n'ont aucune difficulté à bien représenter des objets comportant des surfaces très près l'une de l'autre. Par exemple, pour représenter une longue tige fine ou le dessus et le dessous du panneau d'une table, le nombre de faces requises dans un maillage peut être considérablement moindre que le nombre de points requis pour assurer la qualité de la reconstruction de la surface MCM. Certes, les surfaces MCM dites algébriques [Guennebaud et Gross 2007, Guennebaud *et al.* 2008] peuvent s'en tirer avec un nombre plus petit de points, mais l'information des normales est requise pour reconstruire une surface moins bosselée, et alors le problème est déplacé à l'obtention d'un ensemble de points avec une bonne estimation des normales.

Malgré tout ce qui a été dit, les ensembles de points demeurent un choix certainement viable. D'abord, la modélisation géométrique par points est un domaine encore relativement

[2]Un échantillonnage-ϵ est un ensemble de points tel que pour tous ses éléments, la distance du voisin le plus près est inférieure à ϵ.

[3]Nous portons une attention particulière à la comparaison entre maillages et représentations par points car les maillages, incluant aussi les surfaces de subdivision et les surfaces paramétriques, sont la représentation standard *de facto* en modélisation géométrique au sein de la communauté graphique.

jeune. Puis, même si la complexité des algorithmes venait à augmenter pour tenir compte des problèmes d'échantillonnage, lorsqu'il est question de joindre des surfaces, nous considérons que l'avantage des ensembles de points sur les maillages surpasse significativement leurs inconvénients, en particulier parce que la consistance de la représentation est beaucoup plus facile à maintenir et est moins encline aux erreurs numériques. De plus, nous croyons que les derniers développements sur les surfaces MCM algébriques forment une bonne base pour développer des représentations par points encore plus efficaces et robustes. Ceci, avec les derniers avancements sur le matériel graphique dédié aux points [Weyrich *et al.* 2007, Heinzle *et al.* 2008a, Heinzle *et al.* 2008b], indique un avenir toujours prometteur pour les représentations par points.

Bibliographie

ADAMS, B., ET DUTRÉ, P. 2003. Interactive boolean operations on surfel-bounded solids. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) 22*, 3 (juillet), 651–656.

ADAMS, B., ET DUTRÉ, P. 2004. Boolean operations on surfel-bounded solids using programmable graphics hardware. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 19–24.

ADAMS, B., WICKE, M., DUTRÉ, P., GROSS, M., PAULY, M., ET TESCHNER, M. 2004. Interactive 3d painting on point-sampled objects. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 57–66.

ADAMSON, A., ET ALEXA, M. 2003. Approximating and intersecting surfaces from points. Dans *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 230–239.

ADAMSON, A., ET ALEXA, M. 2003. Ray tracing point set surfaces. Dans *Proceedings of Shape Modeling International 2003*, 272–279.

ADAMSON, A., ET ALEXA, M. 2004. Approximating bounded, non-orientable surfaces from points. Dans *Proceedings of IEEE International Conference on Shape Modeling and Application (SMI 2004)*, 243–252.

ADAMSON, A., ET ALEXA, M. 2006. Anisotropic point set surfaces. Dans *Proceedings of the 4th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa (AFRIGRAPH 2006)*, 7–13.

ADAMSON, A., ET ALEXA, M. 2006. Point-sampled cell complexes. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2006) 25*, 3 (juillet), 671–680.

ADOBE SYSTEMS INC. Adobe Photoshop : Professional photo editing software. `http://www.adobe.com/products/photoshop/`.

ALEXA, M., ET ADAMSON, A. 2004. On normals and projection operators for surfaces defined by point sets. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 149–155.

ALEXA, M., ET ADAMSON, A. 2008. Interpolatory point set surfaces — convexity and Hermite data. *ACM Transactions on Graphics*. À paraître.

ALEXA, M., BEHR, J., COHEN-OR, D., FLEISHMAN, S., LEVIN, D., ET SILVA, C. T. 2001. Point set surfaces. *IEEE Visualization 2001* (octobre), 21–28.

ALEXA, M., BEHR, J., COHEN-OR, D., FLEISHMAN, S., LEVIN, D., ET SILVA, C. T. 2003. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics 9*, 1 (janvier), 3–15.

ALLÈGRE, R., CHAINE, R., ET AKKOUCHE, S. 2007. A flexible framework for surface reconstruction from large point sets. *Computers & Graphics 31*, 2 (avril), 190–204.

AMENTA, N., ET BERN, M. 1999. Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry 22*, 4 (décembre), 481–504.

AMENTA, N., ET KIL, Y. J. 2004. Defining point-set surfaces. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2004) 23*, 3 (août), 264–270.

AMENTA, N., ET KIL, Y. J. 2004. The domain of a point set surface. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 138–147.

AMENTA, N., CHOI, S., ET KOLLURI, R. K. 2001. The power crust. Dans *Proceedings of 6th Symposium on Solid Modeling and Applications*, 249–266.

ANDERSSON, L.-E., ET STEWART, N. F. 2009. *An Introduction to the Mathematics of Subdivision Surfaces.* manuscrit.

ANDERSSON, M., GIESEN, J., PAULY, M., ET SPECKMANN, B. 2004. Bounds on the k-neighborhood for locally uniformly sampled surfaces. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 167–171.

APODACA, A. A., ET GRITZ, L. 1999. *Advanced RenderMan.* Morgan Kaufmann Publishers.

ARIKAN, O., ET FORSYTH, D. A. 2002. Interactive motion generation from examples. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (juillet), 483–490.

ARYA, S., MOUNT, D. M., NETANYAHU, N. S., SILVERMAN, R., ET WU, A. Y. 1998. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM 45*, 6 (novembre), 891–923.

ASHIKHMIN, M. 2001. Synthesizing natural textures. Dans *Symposium on Interactive 3D Graphics*, 217–226.

AUTODESK, INC. Autodesk 3ds Max. `http://www.autodesk.com/3dsmax/`.

AUTODESK, INC. Autodesk Maya. `http://www.autodesk.com/maya/`.

AVID TECHNOLOGY, INC. SoftimagelXSI. `http://www.softimage.com/products/xsi/`.

AYACHE, N. 1991. *Artificial Vision for Mobile Robots — Stereo-vision and Multisensor Perception.* MIT Press, Cambridge, MA.

BÆRENTZEN, A., ET CHRISTENSEN, N. J. 2003. Hardware accelerated point rendering of isosurfaces. *Journal of WSCG 11*, 1 (février), 41–48.

BALA, K., WALTER, B., ET GREENBERB, D. P. 2003. Combining edges and points for interactive high quality rendering. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) 22*, 3 (juillet), 631–640.

BANGAY, S., ET MORKEL, C. 2006. Graph matching with subdivision surfaces for texture synthesis on surfaces. Dans *Proceedings of the 4th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa (AFRIGRAPH 2006)*, 65–74.

BAR-JOSEPH, Z., EL-YANIV, R., LISCHINSKI, D., ET WERMAN, M. 2001. Texture mixing and texture movie synthesis using statistical learning. *IEEE Transactions on Visualization and Computer Graphics 7*, 2 (avril), 120–135.

BENDELS, G. H., SCHNABEL, R., ET KLEIN, R. 2006. Detecting holes in point set surfaces. *Journal of WSCG 14*, 1-3 (janvier), 89–96.

BENSON, D., ET DAVIS, J. 2002. Octree textures. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (juillet), 785–790.

BENTLEY, J. L., ET FRIEDMAN, J. H. 1979. Data structures for range searching. *ACM Computing Surveys 11*, 4 (décembre), 397–409.

BENTLEY, J. L. 1975. Multidimensional binary search trees used for associative searching. *Communications of the ACM 18*, 9 (septembre), 509–517.

BERTALMIO, M., VESE, L., SAPIRO, G., ET OSHER, S. 2003. Simultaneous structure and texture image inpainting. Dans *Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR 2003)*, 882–889.

BESL, P. J., ET MCKAY, N. D. 1992. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 2 (février), 239–256.

BHAT, K. S., SEITZ, S. M., HODGINS, J. K., ET KHOSLA, P. K. 2004. Flow-based video synthesis and editing. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2004) 23*, 3, 360–363.

BHAT, P., INGRAM, S., ET TURK, G. 2004. Geometric texture synthesis by example. Dans *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 43–46.

BIERMANN, H., MARTIN, I., BERNARDINI, F., ET ZORIN, D. 2002. Cut-and-paste editing of multiresolution surfaces. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (juillet), 312–321.

BISCHOFF, S., PAVIC, D., ET KOBBELT, L. 2005. Automatic restoration of polygon models. *ACM Transactions on Graphics 24*, 4 (octobre), 1332–1352.

BLENDER FOUNDATION. Blender. `http://www.blender.org/`.

BLINN, J. F., ET NEWELL, M. E. 1976. Texture and reflection in computer generated images. *Communications of the ACM 19*, 10 (décembre), 542–547.

BLINN, J. F. 1978. Simulation of wrinkled surfaces. Dans *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 78)*, 286–292.

BLINN, J. F. 1982. Light reflection functions for simulation of clouds and dusty surfaces. Dans *Proceedings of the 9th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 82)*, 21–29.

BLOOMENTHAL, J., Ed. 1997. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc.

BOIER-MARTIN, I. M. 2003. Domain decomposition for multiresolution analysis. Dans *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 31–40.

BOLLE, R. M., ET VEMURI, B. C. 1991. On three-dimensional surface reconstruction methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 1 (janvier), 1–13.

BOTSCH, M., ET KOBBELT, L. 2003. High-quality point-based rendering on modern GPUs. Dans *Proceedings of Pacific Graphics 2003*, 335–343.

BOTSCH, M., ET KOBBELT, L. 2004. A remeshing approach to multiresolution modeling. Dans *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 185–192.

BOTSCH, M., ET SORKINE, O. 2008. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (janvier), 213–230.

BOTSCH, M., WIRATANAYA, A., ET KOBBELT, L. 2002. Efficient high quality rendering of point-sampled geometry. Dans *13th Eurographics Workshop on Rendering*, 53–64.

BOTSCH, M., SPERNAT, M., ET KOBBELT, L. 2004. Phong splatting. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 25–32.

BOTSCH, M., HORNUNG, A., ZWICKER, M., ET KOBBELT, L. 2005. High-quality surface splatting on today's GPUs. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2005*, 17–24.

BOUBEKEUR, T., ET SCHLICK, C. 2006. Interactive out-of-core texturing with point-sampled textures. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2006*, 67–73.

BOUBEKEUR, T., REUTER, P., ET SCHLICK, C. 2005. Surfel stripping. Dans *Proceedings Conference on Computer Graphics and Interactive Techniques in Australasia and South-East Asia (GRAPHITE 2005)*, 177–186.

BOUBEKEUR, T., HEIDRICH, W., GRANIER, X., ET SCHLICK, C. 2006. Volume-surface trees. *Computer Graphics Forum (Proceedings of Eurographics 2006) 25*, 3 (septembre), 399–406.

BOUBEKEUR, T., SORKINE, O., ET SCHLICK, C. 2007. SIMOD : Making freeform deformation size-insensitive. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2007*, 47–56.

BOYER, E., LISTER, L., ET SHADER, B. L. 2000. Sphere-of-influence graphs using the sup-norm. *Mathematical and Computer Modelling 32*, 10 (novembre), 1071–1082.

BREMER, P.-T., ET HART, J. C. 2005. A sampling theorem for MLS surfaces. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2005*, 47–54.

BROOKS, S., ET DODGSON, N. 2002. Self-similarity based texture editing. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (juillet), 653–656.

BURKE, R., 2003. Image quilting, Wang tiling, and texture transfer. `http://www.robburke.net/mle/wang/`, août.

BURT, P. J., ET ADELSON, E. H. 1983. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications 31*, 4 (avril), 532–540.

CANOMA. `http://www.canoma.com/`.

CARR, N. A., ET HART, J. C. 2004. Painting detail. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2004) 23*, 3 (août), 845–852.

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 (ACM SIGGRAPH 2001)*, 67–76.

CARR, N. A., HOBEROCK, J., CRANE, K., ET HART, J. C. 2006. Rectangular multi-chart geometry images. Dans *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 181–190.

CATMULL, E. E. 1974. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. Thèse de doctorat, Computer Science Department, University of Utah, Salt Lake City.

CHEN, Y., ET MEDIONI, G. 1992. Object modelling by registration of multiple range images. *International Journal of Computer Vision and Image Understanding 10*, 3 (avril), 145–155.

CHEN, B., ET NGUYEN, M. X. 2001. POP : A hybrid point and polygon rendering system for large data. Dans *IEEE Visualization 2001*, 45–52.

CHEN, M., KAUFMAN, A. E., ET YAGEL, R., Eds. 2000. *Volume Graphics*. Springer-Verlag.

CHHUGANI, J., ET KUMAR, S. 2003. Budget sampling of parametric surface patches. Dans *Symposium on Interactive 3D Graphics*, 131–138.

CIPOLLA, R., ET BLAKE, A. 1992. Surface shape from the deformation of apparent contours. *International Journal of Computer Vision 9*, 2 (novembre), 83–112.

CLARENZ, U., RUMPF, M., ET TELEA, A. 2004. Fairing of point based surfaces. Dans *Proceedings of Computer Graphics International 2004*, 600–603.

CLARENZ, U., RUMPF, M., ET TELEA, A. 2004. Finite elements on point based surfaces. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 201–211.

CO, C. S., HAMANN, B., ET JOY, K. I. 2003. Iso-splatting : A point-based alternative to isosurface visualization. Dans *Proceedings of Pacific Graphics 2003*, 325–334.

COCONU, L., ET HEGE, H.-C. 2002. Hardware-accelerated point-based rendering of complex scenes. Dans *13th Eurographics Workshop on Rendering*, 43–52.

COHEN, J. D., ALIAGA, D. G., ET ZHANG, W. 2001. Hybrid simplification : Combining multi-resolution polygon and point rendering. Dans *IEEE Visualization 2001*, 37–44.

COHEN, M. F., SHADE, J., HILLER, S., ET DEUSSEN, O. 2003. Wang tiles for image and texture generation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) 22*, 3 (juillet), 287–294.

COLEMAN, D., HOLLAND, P., KADEN, N., KLEMA, V., ET PETERS, S. C. 1980. A system of subroutines for iteratively reweighted least squares computations. *ACM Transactions on Mathematical Software 6*, 3 (septembre), 327–336.

COLLINS, A., ZOMORODIAN, A., CARLSSON, G., ET GUIBAS, L. 2004. A barcode shape descriptor for curve point cloud data. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 181–191.

COOK, R. L. 1984. Shade trees. Dans *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 84)*, 223–231.

COUGHLAN, J. M., ET YUILLE, A. L. 2002. Algorithms from statistical physics for generative models of images. *Image and Vision Computing 21*, 1 (janvier), 29–36.

CRIMINISI, A., PEREZ, P., ET TOYAMA, K. 2003. Object removal by exemplar-based inpainting. Dans *Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR 2003)*, 721–728.

CSURI, C., HACKATHORN, R., PARENT, R., CARLSON, W., ET HOWARD, M. 1979. Towards an interactive high visual complexity animation system. Dans *Proceedings of the 6th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 79)*, 289–299.

CURLESS, B., ET LEVOY, M. 1996. A volumetric method for building complex models from range images. Dans *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 96)*, 303–312.

CYBERWARE. `http://www.cyberware.com/`.

CYBERWARE. Desktop 3D scanner samples. `http://www.cyberware.com/products/scanners/desktopSamples.html`.

CYBERWARE. Model Shop color 3D scanner samples. `http://www.cyberware.com/products/scanners/msSamples.html`.

DACHSBACHER, C., VOGELGSANG, C., ET STAMMINGER, M. 2003. Sequential point trees. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) 22*, 3 (juillet), 657–662.

DANA, K. J., VAN GINNEKEN, B., NAYAR, S. K., ET KOENDERINK, J. J. 1999. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics 18*, 1 (janvier), 1–34.

DANIELS, J. I., HA, L. K., OCHOTTA, T., ET SILVA, C. T. 2007. Robust smooth feature extraction from point clouds. Dans *Proceedings of IEEE International Conference on Shape Modeling and Application (SMI 2007)*, 123–136.

DAVIS, J., MARSCHNER, S. R., GARR, M., ET LEVOY, M. 2002. Filling holes in complex surfaces using volumetric diffusion. Dans *International Symposium on 3D Data Processing, Visualization, and Transmission*, 428–438.

DE BONET, J. S. 1997. Multiresolution sampling procedure for analysis and synthesis of texture images. Dans *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 97)*, 361–368.

DE SILVA, V., ET CARLSSON, G. 2004. Topological estimation using witness complexes. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 157–166.

DEBEVEC, P. E., TAYLOR, C. J., ET MALIK, J. 1996. Modeling and rendering architecture from photographs : A hybrid geometry- and image-based approach. Dans *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 96)*, 11–20.

DEBEVEC, P., YU, Y., ET BOSHOKOV, G. 1998. Efficient view-dependent image-based rendering with projective texture-mapping. Dans *9th Eurographics Workshop on Rendering*, 105–116.

DEBRY, D., GIBBS, J., PETTY, D. D., ET ROBINS, N. 2002. Painting and rendering textures on unparameterized models. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (juillet), 763–768.

DEDIEU, S. 2001. *Adaptation d'un système de reconstruction de modèles numériques 3D à partir de photographies face aux connaissances de l'utilisateur.* Thèse de doctorat, Université de Bordeaux I.

DEMARSIN, K., VANDERSTRAETEN, D., VOLODINE, T., ET ROOSE, D. 2007. Detection of closed sharp edges in point clouds using normal estimation and graph theory. *Computer Aided Design 39*, 4 (avril), 276–283.

DENG, Z., LEWIS, J. P., ET NEUMANN, U. 2003. Practical eye movement model using texture synthesis. Dans *Proceedings of ACM SIGGRAPH 2003 Conference on Sketches & Applications.*

DESBRUN, M., MEYER, M., SCHRÖDER, P., ET BARR, A. H. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. Dans *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 99)*, 317–324.

DESBRUN, M., MEYER, M., ET ALLIEZ, P. 2002. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum (Proceedings of Eurographics 2002) 21*, 3 (septembre), 209–209.

DEUSSEN, O., HANRAHAN, P., LINTERMANN, B., MĚCH, R., PHARR, M., ET PRUSINKIEWICZ, P. 1998. Realistic modeling and rendering of plant ecosystems. Dans *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 98)*, 275–286.

DEUSSEN, O., COLDITZ, C., STAMMINGER, M., ET DRETTAKIS, G. 2002. Interactive visualization of complex plant ecosystems. Dans *IEEE Visualization 2002*, 219–226.

DEY, T. K., ET GOSWAMI, S. 2004. Provable surface reconstruction from noisy samples. Dans *Proceedings of the 20th Annual Symposium on Computational Geometry*, 330–339.

DEY, T. K., ET HUDSON, J. 2002. PMR : Point to mesh rendering, a feature-based approach. Dans *IEEE Visualization 2002*, 155–162.

DEY, T. K., ET SUN, J. 2005. An adaptive MLS surface for reconstruction with guarantees. Dans *Proceedings of Eurographics/AMC SIGGRAPH Symposium on Geometry Processing*, 43–52.

DEY, T. K., GIESEN, J., ET HUDSON, J. 2001. Delaunay based shape reconstruction from large data. Dans *IEEE Symposium on Parallel and Large-data Visualization and Graphics (PVG 2001)*, 19–27.

DEY, T. K., GIESEN, J., ET GOSWAMI, S. 2004. Shape segmentation and matching from noisy point clouds. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 193–199.

DEY, T. K., GOSWAMI, S., ET SUN, J. 2004. Smoothing noisy point clouds with Delaunay preprocessing and MLS. Tech. Rep. OSU-CISRC-3/04-TR17, Department of Computer Science and Engineering, Ohio State University.

DEY, T. K., LI, G., ET SUN, J. 2005. Normal estimation for point clouds : A comparison study for a Voronoi based method. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2005*, 39–46.

DINH, H. Q., TURK, G., ET SLABAUGH, G. 2002. Reconstructing surfaces by volumetric regularization using radial basis functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 10 (octobre), 1358–1371.

DO CARMO, M. P. 1976. *Differential Geometry of Curves and Surfaces*. Prentice Hall.

DOBASHI, Y., YAMAMOTO, T., ET NISHITA, T. 2004. Radiosity for point-sampled geometry. Dans *Proceedings of Pacific Graphics 2004*, 152–159.

DONG, J., ET CHANTLER, M. 2005. Capture and synthesis of 3D surface texture. *International Journal of Computer Vision 62*, 1-2 (avril-mai), 177–194.

DONG, W., SUN, S., ET PAUL, J.-C. 2005. Optimal sample patches selection for tile-based texture synthesis. Dans *Proceedings of the Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG 2005)*, 503–508.

DONG, F., CLAPWORTHY, G., ET LIN, H. 2007. Pseudo surface-texture synthesis. *Computers & Graphics 31*, 2 (avril), 252–261.

DONG, W., ZHOU, N., ET PAUL, J.-C. 2008. Perspective-aware texture analysis and synthesis. *The Visual Computer 24*, 7–9 (juin), 515–523.

DONG, W., ZHOU, N., ET PAUL, J.-C. 2008. Robust tile-based texture synthesis using artificial immune system. *Neural Computing & Applications Online First* (avril).

DORETTO, G., CHIUSO, A., WU, Y. N., ET SOATTO, S. 2003. Dynamic textures. *International Journal of Computer Vision 51*, 2 (février), 91–109.

DRORI, I., COHEN-OR, D., ET YESHURUN, H. 2003. Fragment-based image completion. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) 22*, 3 (juillet), 303–312.

DUGUET, F., ET DRETTAKIS, G. 2004. Flexible point-based rendering on mobile devices. *IEEE Computer Graphics and Applications 24*, 4 (juillet), 57–63.

DURANLEAU, F., ET POULIN, P. 2006. Patch-based synthesis of geometry textures with point-set surfaces. Dans *Vision, Modeling, and Visualization 2006*, 1–8.

DURANLEAU, F., BEAUDOIN, P., ET POULIN, P. 2008. Multiresolution point-set surfaces. Dans *Proceedings of Graphics Interface 2008*, 211–218.

EBERT, D. S., MUSGRAVE, F. K., PEACHEY, D., PERLIN, K., ET WORLEY, S. 1998. *Texturing & Modeling*. AP Professional.

ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., ET STUETZLE, W. 1995. Multiresolution analysis of arbitrary meshes. Dans *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 1995)*, 173–182.

EFROS, A. A., ET FREEMAN, W. T. 2001. Image quilting for texture synthesis and transfer. Dans *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2001)*, 341–346.

EFROS, A. A., ET LEUNG, T. K. 1999. Texture synthesis by non-parametric sampling. Dans *IEEE International Conference on Computer Vision*, vol. 2, 1033–1038.

EISENACHER, C., LEFEBVRE, S., ET STAMMINGER, M. 2008. Texture synthesis from photographs. *Computer Graphics Forum (Proceedings of Eurographics 2008) 27*, 2 (avril), 419–428.

ENS, J., ET LAWRENCE, P. 1993. An investigation of methods for determining depth from focus. *IEEE Transactions on Pattern Analysis and Machine Intelligence 15*, 2, 97–108.

EOM, K. B. 2000. Synthesis of color textures for multimedia applications. *Multimedia Tools Applications 12*, 1 (septembre), 81–98.

EOS SYSTEMS INC. PhotoModeler. `http://www.photomodeler.com/`.

FANG, H., ET HART, J. C. 2004. Textureshop : Texture synthesis as a photograph editing tool. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2004) 23*, 3 (août), 354–359.

FAUGERAS, O. 1993. *Three-Dimensional Computer Vision : A Geometric Viewpoint*. MIT Press, Cambridge (MA).

FLEISCHER, K. W., LAIDLAW, D. H., CURRIN, B. L., ET BARR, A. H. 1995. Cellular texture generation. Dans *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 95)*, 239–248.

FLEISHMAN, S., COHEN-OR, D., ALEXA, M., ET SILVA, C. T. 2003. Progressive point set surfaces. *ACM Transactions on Graphics 22*, 4 (octobre), 997–1011.

FLEISHMAN, S., DRORI, I., ET COHEN-OR, D. 2003. Bilateral mesh denoising. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) 22*, 3 (juillet), 950–653.

FLEISHMAN, S., COHEN-OR, D., ET SILVA, C. T. 2005. Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2005) 24*, 3 (juillet), 544–552.

FOLEY, J. D., VAN DAM, A., FEINER, S. K., ET HUGHES, J. F. 1996. *Computer Graphics, Principles and Practice*, Second Edition in C. Addison-Wesley.

FORD, L., ET FULKERSON, D. 1962. *Flows in Networks*. Princeton University Press.

FORSEY, D. R., ET BARTELS, R. H. 1988. Hierarchical B-spline refinement. Dans *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 1988)*, 205–212.

FOURNIER, M., DISCHLER, J.-M., ET BECHMANN, D. 2003. 3D distance transform adaptive filtering for smoothing and denoising triangle meshes. Dans *Proceedings Conference on Computer Graphics and Interactive Techniques in Australasia and South-East Asia (GRAPHITE 2006)*, 407–416.

FRIEDEL, I., SCHRÖDER, P., ET KHODAKOVSKY, A. 2004. Variational normal meshes. *ACM Transactions on Graphics 23*, 4 (octobre), 1061–1073.

FRISKEN, S. F., ET PERRY, R. N. 2002. Efficient estimation of 3D Euclidean distance fields from 2D range images. Dans *IEEE Symposium on Volume Visualization and Graphics 2002*, 81–88.

FRISKEN, S. F., PERRY, R. N., ROCKWOOD, A. P., ET JONES, T. R. 2000. Adaptively sampled distance fields : a general representation of shape for computer graphics. Dans *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2000)*, 249–254.

FUNKHOUSER, T., KAZHDAN, M., SHILANE, P., MIN, P., KIEFER, W., TAL, A., RUSINKIEWICZ, S., ET DOBKIN, D. 2004. Modeling by example. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2004) 23*, 3 (août), 652–663.

GEMAN, S., ET GEMAN, D. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 6*, 6 (novembre), 721–741.

GERSHO, A., ET GRAY, R. M. 1991. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers.

GIMP — The GNU image manipulation program. `http://www.gimp.org/`.

GINGOLD, Y. I., ET ZORIN, D. 2007. Controlled-topology filtering. *Computer Aided Design 39*, 8 (août), 676–684.

GOBBETTI, E., ET MARTON, F. 2004. Layered point clouds. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 113–120.

GONZALEZ, R. C., ET WOODS, R. E. 2002. *Digital Image Processing*, deuxième édition. Prentice Hall.

GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., ET COHEN, M. F. 1996. The lumigraph. Dans *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 1996)*, 43–54.

GRANGER-PICHÉ, M., EPSTEIN, E., ET POULIN, P. 2004. Interactive hierarchical space carving with projector-based calibrations. Dans *Vision, Modeling, and Visualization 2004*, 159–166.

GROSS, M., ET PFISTER, H., Eds. 2007. *Point-Based Graphics*. The Morgan Kaufmann Series in Computer Graphics. Morgan Kaufmann, juin.

GROSSMAN, J. P., ET DALLY, W. J. 1998. Point sample rendering. Dans *9th Eurographics Workshop on Rendering*, 181–192.

GU, X., GORTLER, S. J., ET HOPPE, H. 2002. Geometry images. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (juillet), 355–361.

GUAN, X., ET MUELLER, K. 2004. Point-based surface rendering with motion blur. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 33–40.

GUENNEBAUD, G., ET GROSS, M. 2007. Algebraic point set surfaces. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2007) 26*, 3 (juillet), article 23.

GUENNEBAUD, G., ET PAULIN, M. 2003. Efficient screen space approach for hardware accelerated surfel rendering. Dans *Vision, Modeling, and Visualization 2003*, 485–493.

GUENNEBAUD, G., BARTHE, L., ET PAULIN, M. 2004. Deferred splatting. *Computer Graphics Forum (Proceedings of Eurographics 2004) 23*, 3 (septembre), 653–660.

GUENNEBAUD, G., BARTHE, L., ET PAULIN, M. 2004. Dynamic surfel set refinement for high quality rendering. *Computers & Graphics 28*, 6 (décembre), 827–838.

GUENNEBAUD, G., BARTHE, L., ET PAULIN, M. 2004. Real-time point cloud refinement. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 41–49.

GUENNEBAUD, G., BARTHE, L., ET PAULIN, M. 2005. Interpolatory refinement for real-time processing of point-based geometry. *Computer Graphics Forum (Proceedings of Eurographics 2005) 24*, 3 (septembre), 657–667.

GUENNEBAUD, G., BARTHE, L., ET PAULIN, M. 2006. Splat/mesh blending, perspective rasterization and transparency for point-based rendering. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2006*, 49–58.

GUENNEBAUD, G., GERMANN, M., ET GROSS, M. 2008. Dynamic sampling and rendering of algebraic point set surfaces. *Computer Graphics Forum (Proceedings of Eurographics 2008) 27*, 2 (avril), 653–662.

GUENNEBAUD, G., 2007. Experimental point engine. `http://graphics.ethz.ch/research/apss/tarballs/Expe-070801.tar.bz2`, août.

GUILLOU, E., MENEVEAUX, D., MAISEL, E., ET BOUATOUCH, K. 2000. Using vanishing points for camera calibration and coarse 3D reconstruction from a single image. *The Visual Computer 16*, 7, 396–410.

GUMHOLD, S., WANG, X., ET MACLEOD, R. 2001. Feature extraction from point clouds. Dans *Proceedings of the 10th International Meshing Roundtable*, 293–305.

GUO, X., HUA, J., ET QIN, H. 2004. Point set surface editing techniques based on level-sets. Dans *Proceedings of Computer Graphics International 2004*, 52–59.

GUO, X., HUA, J., ET QIN, H. 2004. Scalar-function-driven editing on point set surfaces. *IEEE Computer Graphics and Applications 24*, 4 (juillet), 43–52.

GUSKOV, I., ET WOOD, Z. J. 2001. Topological noise removal. Dans *Proceedings of Graphics Interface 2001*, 19–26.

GUSKOV, I., SWELDENS, W., ET SCHRÖDER, P. 1999. Multiresolution signal processing for meshes. Dans *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 99)*, 325–334.

GUSKOV, I., VIDIMCE, K., SWELDENS, W., ET SCHRÖDER, P. 2000. Normal meshes. Dans *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2000)*, 95–102.

GUSKOV, I., KHODAKOVSKY, A., SCHRÖDER, P., ET SWELDENS, W. 2002. Hybrid meshes : Multiresolution using regular and irregular refinement. Dans *Proceedings of the 8th Annual Symposium on Computational Geometry*, 264–272.

HAN, J., ZHOU, K., WEI, L.-Y., GONG, M., BAO, H., ZHANG, X., ET GUO, B. 2006. Fast example-based surface texture synthesis via discrete optimization. *The Visual Computer 22*, 9–11 (septembre), 918–925.

HANRAHAN, P., ET HAEBERLI, P. 1990. Direct WYSIWYG painting and texturing on 3D shapes. Dans *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 90)*, 215–223.

HE, Z., ET LIANG, X. 2007. A multiresolution object space point-based rendering approach for mobile devices. Dans *Proceedings of The 5th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa (AFRIGRAPH 2007)*, 7–13.

HEEGER, D. J., ET BERGEN, J. R. 1995. Pyramid-based texture analysis/synthesis. Dans *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 95)*, 229–238.

HEINZLE, S., GUENNEBAUD, G., BOTSCH, M., ET GROSS, M. 2008. A hardware processing unit for point sets. Dans *Proceedings of ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 21–31.

HEINZLE, S., SAURER, O., AXMANN, S., BROWARNIK, D., SCHMIDT, A., CARBOGNANI, F., LUETHI, P., FELBER, N., ET GROSS, M. 2008. A transform, lighting and setup ASIC for surface splatting. Dans *Proceedings of International Symposium on Circuits and Systems (ISCAS 2008)*, 2813–2816.

HERTZMANN, A., JACOBS, C. E., OLIVER, N., CURLESS, B., ET SALESIN, D. H. 2001. Image analogies. Dans *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2001)*, 327–340.

HERTZMANN, A., OLIVER, N., CURLESS, B., ET SEITZ, S. M. 2002. Curve analogies. Dans *13th Eurographics Workshop on Rendering*, 233–246.

HOLST, M., ET SCHUMANN, H. 2007. Surfel-based billboard hierarchies for fast rendering of 3D-objects. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2007*, 109–118.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., ET STUETZLE, W. 1992. Surface reconstruction from unorganized points. Dans *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 92)*, 71–78.

HOPPE, H., DEROSE, T., DUCHAMP, T., HALSTEAD, M., JIN, H., MCDONALD, J., SCHWEITZER, J., ET STUETZLE, W. 1994. Piecewise smooth surface reconstruction. Dans *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 1994)*, 295–302.

HOPPE, H. 1996. Progressive meshes. Dans *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 1996)*, 99–108.

HORN, B., ET BROOKS, M. 1989. *Shape from Shading*. MIT Press, Cambridge, MA.

HORNUNG, A., ET KOBBELT, L. 2006. Robust reconstruction of watertight 3D models from non-uniformly sampled point clouds without normal information. Dans *Proceedings of Eurographics Symposium on Geometry Processing*, 41–50.

HOUSTON, B., NIELSEN, M. B., BATTY, C., NILSSON, O., ET MUSETH, K. 2006. Hierarchical RLE level set : A compact and versatile deformable surface representation. *ACM Transactions on Graphics 25*, 1 (janvier), 151–175.

HU, G.-F., PENG, Q.-S., ET FORREST, A. R. 2004. Robust mesh smoothing. *Journal of Computer Science and Technology 19*, 4 (juillet), 521–528.

HU, G., PENG, Q., ET FORREST, A. R. 2006. Mean shift denoising of point-sampled surfaces. *The Visual Computer 22*, 3 (mars), 147–157.

HUANG, Y., PENG, J., KUO, C.-C. J., ET GOPI, M. 2006. Octree-based progressive geometry coding of point clouds. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2006*, 103–110.

HUANG, H.-D., TONG, X., ET WANG, W.-C. 2007. Accelerated parallel texture optimization. *Journal of Computer Science and Technology 22*, 5 (septembre), 761–769.

HUANG, K.-S., 2007. Point-based graphics resource. `http://kesen.huang.googlepages.com/PointBasedPaper.html`.

HUBELI, A., ET GROSS, M. 2001. Multiresolution methods for nonmanifold models. *IEEE Transactions on Visualization and Computer Graphics 7*, 3 (juillet), 207–221.

HÜBNER, T., ZHANG, Y., ET PAJAROLA, R. 2006. Multi-view point splatting. Dans *Proceedings Conference on Computer Graphics and Interactive Techniques in Australasia and South-East Asia (GRAPHITE 2006)*, 285–294.

HUBO, E., MERTENS, T., HABER, T., ET BEKAERT, P. 2006. The quantized kd-tree : Efficient ray tracing of compressed point clouds. Dans *IEEE Symposium on Interactive Ray Tracing*, 105–113.

HUBO, E., MERTENS, T., HABER, T., ET BEKAERT, P. 2007. Self-similarity-based compression of point clouds, with application to ray tracing. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2007*, 129–137.

IGARASHI, T., MATSUOKA, S., ET TANAKA, H. 1999. Teddy : a sketching interface for 3D freeform design. Dans *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 99)*, 409–416.

IGEHY, H., ET PEREIRA, L. 1997. Image replacement through texture synthesis. Dans *Proceedings of the International Conference on Image Processing*, vol. 3, 186–189.

IJIRI, T., MÊCH, R., IGARASHI, T., ET MILLER, G. 2008. An example-based procedural system for element arrangement. *Computer Graphics Forum (Proceedings of Eurographics 2008) 27*, 2 (avril), 429–436.

ISMERT, R. M., BALA, K., ET GREENBERG, D. P. 2003. Detail synthesis for image-based texturing. Dans *Symposium on Interactive 3D Graphics*, 171–175.

JENKE, P., WAND, M., BOKELOTH, M., SCHILLING, A., ET STRASSER, W. 2006. Bayesian point cloud reconstruction. *Computer Graphics Forum (Proceedings of Eurographics 2006) 25*, 3 (septembre), 379–388.

JEONG, W.-K., ET KIM, C.-H. 2002. Direct reconstruction of a displaced subdivision surface from unorganized points. *Graphical Models 64*, 2 (mars), 78–93.

JI, J., LI, S., LIU, X., ET WU, E. 2004. P-quadtrees : A point and polygon hybrid multiresolution rendering approach. Dans *Proceedings of Computer Graphics International 2004*, 382–385.

JODOIN, P.-M., EPSTEIN, E., GRANGER-PICHÉ, M., ET OSTROMOUKHOV, V. 2002. Hatching by example : a statistical approach. Dans *Proceedings of the 2nd International Symposium on Non-Photorealistic Animation and Rendering (NPAR 2002)*, 29–36.

JONES, T. R., DURAND, F., ET DESBRUN, M. 2003. Non-iterative, feature-preserving mesh smoothing. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) 22*, 3 (juillet), 943–949.

JONES, M. W., BÆRENTZEN, J. A., ET SRAMEK, M. 2006. 3D distance fields : A survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics 12*, 4 (juillet), 581–599.

JOYOT, P., TRUNZLER, J., ET CHINESTA, F. 2005. *Meshfree Methods for Partial Differential Equations II*, vol. 43 de *Lecture Notes in Computational Science and Engineering*. Springer, ch. Enriched Reproducing Kernel Approximation : Reproducing Functions with Discontinuous Derivatives, 93–107.

JU, T. 2004. Robust repair of polygonal models. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2004) 23*, 3 (août), 888–895.

KALAIAH, A., ET VARSHNEY, A. 2001. Differential point rendering. Dans *11th Eurographics Workshop on Rendering*, 139–150.

KALAIAH, A., ET VARSHNEY, A. 2003. Modeling and rendering of points with local geometry. *IEEE Transactions on Visualization and Computer Graphics 9*, 1 (janvier), 30–42.

KALAIAH, A., ET VARSHNEY, A. 2003. Statistical point geometry. Dans *Proceedings of Eurographics/AMC SIGGRAPH Symposium on Geometry Processing*, 107–115.

KALAIAH, A., ET VARSHNEY, A. 2005. Statistical geometry representation for efficient transmission and rendering. *ACM Transactions on Graphics 24*, 2 (avril), 348–373.

KANATANI, K. I., ET CHOU, T. C. 1989. Shape from texture : General principle. *Artificial Intelligence 38*, 1, 1–48.

KARA, L. B., ET SHIMADA, K. 2007. Sketch-based 3D-shape creation for industrial styling design. *IEEE Computer Graphics and Applications 27*, 1 (janvier), 60–71.

KARPENKO, O. A., ET HUGHES, J. F. 2006. SmoothSketch : 3D free-form shapes from complex sketches. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2006) 25*, 3 (juillet), 589–598.

KARPENKO, O. A., HUGHES, J. F., ET RASKAR, R. 2002. Free-form sketching with variational implicit surfaces. *Computer Graphics Forum (Proceedings of Eurographics 2002) 21*, 3 (septembre), 585–594.

KAWATA, H., GOUAILLARD, A., ET KANAI, T. 2004. Interactive point-based painterly rendering. Dans *Proceedings of the International Conference on Cyberworlds (CW 2004)*, 293–299.

KAZHDAN, M. 2005. Reconstruction of solid models from oriented point sets. Dans *Proceedings of Eurographics Symposium on Geometry Processing*, 73–82.

KEISER, R., MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., ET GROSS, M. 2004. Contact handling for deformable point-based objects. Dans *Vision, Modeling, and Visualization 2004*, 315–322.

KEISER, R., ADAMS, B., GASSER, D., BAZZI, P., DUTRÉ, P., ET GROSS, M. 2005. A unified Lagrangian approach to solid-fluid animation. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2005*, 125–133.

KELLEY, A. D., MALIN, M. C., ET NIELSON, G. M. 1988. Terrain simulation using a model of stream erosion. Dans *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 88)*, 263–268.

KHODAKOVSKY, A., ET SCHRÖDER, P. 1999. Fine level feature editing for subdivision surfaces. Dans *Proceedings of the 1999 ACM Symposium on Solid and Physical Modeling*, 203–211.

KIM, Y.-S., VALETTE, S., ET PROST, R. 2001. Adaptive wavelets based multiresolution modeling of irregular meshes via harmonic maps. Dans *Proceedings of the International Conference in Image Processing*, 210–213.

KIM, D. B., KANG, E. C., LEE, K. H., ET PAJAROLA, R. 2006. Framework for adaptive sampling of point-based surfaces using geometry and color attributes. Dans *Proceedings of the International Conference on Computational Science and its Applications (ICCSA 2006)*, vol. 3992 de *Lecture Notes in Computer Science*, 371–374.

KITAGO, M., ET GOPI, M. 2006. Efficient and prioritized point subsampling for CSRBF compression. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2006*, 121–128.

KLEIN, J., ET ZACHMANN, G. 2004. Point cloud collision detection. *Computer Graphics Forum (Proceedings of Eurographics 2004) 23*, 3 (août), 567–576.

KLEIN, J., ET ZACHMANN, G. 2004. Point cloud surfaces using geometric proximity graphs. *Computers & Graphics 28*, 6 (décembre), 839–850.

KLEIN, J., ET ZACHMANN, G. 2004. Proximity graphs for defining surfaces over point clouds. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 131–138.

KLEIN, J., KROKOWSKI, J., FISCHER, M., WAND, M., WANKA, R., ET MEYER AUF DER HEIDE, F. 2002. The randomized sample tree : A data structure for externally stored virtual environments. Dans *Proceedings of ACM Symposium on Virtual Reality Software and Technology*, 137–146.

KLEIN, J., KROKOWSKI, J., FISCHER, M., WAND, M., WANKA, R., ET MEYER AUF DER HEIDE, F. 2004. The randomized sample tree : A data structure for externally stored virtual environments. *Presence 13*, 6 (décembre), 617–637.

KLETTE, R., SCHLUNS, K., ET KOSCHAN, A. 1998. *Computer Vision : Three-Dimensional Data from Images*. Springer.

KOBBELT, L., ET BOTSCH, M. 2000. An interactive approach to point cloud triangulation. *Computer Graphics Forum (Proceedings of Eurographics 2000) 19*, 3 (août), 479–487.

KOBBELT, L., ET BOTSCH, M. 2004. A survey of point-based techniques in computer graphics. *Computers & Graphics 28*, 6 (décembre), 801–814.

KOBBELT, L., CAMPAGNA, S., VORSATZ, J., ET SEIDEL, H.-P. 1998. Interactive multi-resolution modeling on arbitrary meshes. Dans *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 1998)*, 105–114.

KOBBELT, L. P., BOTSCH, M., SCHWANECKE, U., ET SEIDEL, H.-P. 2001. Feature sensitive surface extraction from volume data. Dans *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2001)*, 57–66.

KOBBELT, L. 1997. Discrete fairing. Dans *Proceedings of IMA Conference on the Mathematics of Surfaces VII*, 101–131.

KOBBELT, L. 2000. $\sqrt{3}$-subdivision. Dans *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2000)*, 103–112.

KOBBELT, L., 2003. Freeform shape representations for efficient geometry processing. Présentation invitée de Eurographics 2003, septembre.
`http://www-i8.informatik.rwth-aachen.de/publications/downloads/geometry_representations.pdf`.

KOLLURI, R., SHEWCHUK, J. R., ET O'BRIEN, J. F. 2004. Spectral surface reconstruction from noisy point clouds. Dans *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 11–21.

KOLLURI, R. 2005. Provably good moving least squares. Dans *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms*, 1008–1017.

KŘIVÁNEK, J., ŽÁRA, J., ET BOUATOUCH, K. 2003. Fast depth of field rendering with surface splatting. Dans *Proceedings of Computer Graphics International 2003*, 196–201.

KRÜGER, J., SCHNEIDER, J., ET WESTERMANN, R. 2005. DUODECIM - a structure for point scan compression and rendering. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2005*, 99–107.

KULLBACK, S., ET LEIBLER, R. A. 1951. On information and sufficiency. *Annals of Mathematical Statistics 22*, 1 (mars), 79–86.

KUTULAKOS, K. N., ET SEITZ, S. M. 2000. A theory of shape by space carving. *International Journal of Computer Vision 38*, 3, 199–218.

KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., ET BOBICK, A. 2003. Graphcut textures : Image and video synthesis using graph cuts. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) 22*, 3 (juillet), 277–286.

KWATRA, V., ESSA, I., BOBICK, A., ET KWATRA, N. 2005. Texture optimization for example-based synthesis. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2005) 24*, 3 (juillet), 795–802.

LAGAE, A., ET DUTRÉ, P. 2005. A procedural object distribution function. *ACM Transactions on Graphics 24*, 4 (octobre), 1442–1461.

LAGAE, A., ET DUTRÉ, P. 2006. An alternative for Wang tiles : Colored edges versus colored corners. *ACM Transactions on Graphics 25*, 4 (octobre), 1442–1459.

LAGAE, A., DUMONT, O., ET DUTRÉ, P. 2004. Geometry synthesis. Tech. Rep. CW381, Department of Computer Science, Katholieke Universiteit Leuven, mars.

LAGAE, A., DUMONT, O., ET DUTRÉ, P. 2005. Geometry synthesis by example. Dans *Proceedings of Shape Modeling International 2005*, 176–185.

LAI, Y.-K., HU, S.-M., GU, D. X., ET MARTIN, R. R. 2005. Geometric texture synthesis and transfer via geometry images. Dans *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling*, 15–26.

LANGE, C., ET POLTHIER, K. 2005. Anisotropic smoothing of point sets. *Computer Aided Geometric Design 22*, 7 (octobre), 680–692.

LEE, A. W. F., SWELDENS, W., SCHRÖDER, P., COWSAR, L., ET DOBKIN, D. 1998. MAPS : Multiresolution adaptive parameterization of surfaces. Dans *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 1998)*, 95–104.

LEE, A. W. F., DOBKIN, D., SWELDENS, W., ET SCHRÖDER, P. 1999. Multiresolution mesh morphing. Dans *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 99)*, 343–350.

LEE, A., MORETON, H., ET HOPPE, H. 2000. Displaced subdivision surfaces. Dans *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2000)*, 85–94.

LEFEBVRE, S., ET HOPPE, H. 2005. Parallel controllable texture synthesis. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2005) 24*, 3 (juillet), 777–786.

LEFEBVRE, S., ET HOPPE, H. 2006. Appearance-space texture synthesis. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2006) 25*, 3, 541–548.

LEFEBVRE, L., ET POULIN, P. 2000. Analysis and synthesis of structural textures. Dans *Proceedings of Graphics Interface 2000*, 77–86.

LEGAKIS, J., DORSEY, J., ET GORTLER, S. 2001. Feature-based cellular texturing for architectural models. Dans *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2001)*, 309–316.

LEVIN, D. 1998. The approximation power of moving least-squares. *Mathematics of Computation 67*, 224 (octobre), 1517–1531.

LEVIN, D. 2003. *Geometric Modeling for Scientific Visualization.* Springer-Verlag, ch. Mesh-Independent Surface Interpolation, 37–49.

LEVOY, M., ET HANRAHAN, P. 1996. Light field rendering. Dans *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 1996)*, 31–42.

LEVOY, M., ET WHITTED, T. 1985. The use of points as a display primitive. Tech. Rep. 85-022, University of North Carolina Computer Science Department, Chapell Hill, janvier.

LEVOY, M., PULLI, K., CURLESS, B., RUSINKIEWICZ, S., KOLLER, D., PEREIRA, L., GINZTON, M., ANDERSON, S., DAVIS, J., GINSBERG, J., SHADE, J., ET FULK, D. 2000. The digital Michelangelo project : 3D scanning of large statues. Dans *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2000)*, 131–144.

LÉVY, B., PETITJEAN, S., RAY, N., ET MAILLOT, J. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (juillet).

LI, X., ET GUSKOV, I. 2005. Multiscale features for approximate alignment of point-based surfaces. Dans *Proceedings of Eurographics Symposium on Geometry Processing*, 217–226.

LI, Y., WANG, T., ET SHUM, H.-Y. 2002. Motion texture : a two-level statistical model for character motion synthesis. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (juillet), 465–472.

LI, G., BAO, H., ET MA, W. 2004. A unified approach for fairing arbitrary polygonal meshes. *Graphical Models 66*, 3 (mai), 160–179.

LIANG, L., LIU, C., XU, Y.-Q., GUO, B., ET SHUM, H.-Y. 2001. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics 20*, 3 (juillet), 127–150.

LIEN, J.-M. 2007. Approximate star-shaped decomposition of point set data. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2007*, 73–80.

LIEPA, P. 2003. Filling holes in meshes. Dans *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 200–205.

LINDENMAYER, A. 1968. Mathematical models for cellular interaction in development, parts I and II. *Journal of Theoretical Biology 18*, 280–315.

LINSEN, L., ET PRAUTZSCH, H. 2003. Fan clouds — an alternative to meshes. Dans *Theoretical Foundations of Computer Vision*, 39–57.

LINSEN, L. 2001. Point cloud representation. Tech. Rep. 2001-3, Fakultät für Informatik, Universität Karlsruhe.

LIPMAN, Y., COHEN-OR, D., ET LEVIN, D. 2006. Error bounds and optimal neighborhoods for MLS approximation. Dans *Proceedings of Eurographics Symposium on Geometry Processing*, 71–80.

LIPMAN, Y., COHEN-OR, D., ET LEVIN, D. 2007. Data-dependent MLS for faithful surface approximation. Dans *Proceedings of Eurographics Symposium on Geometry Processing*, 59–67.

LIPMAN, Y., COHEN-OR, D., LEVIN, D., ET TAL-EZER, H. 2007. Parameterization-free projection for geometry reconstruction. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2007) 26*, 3 (juillet), article 22.

LIU, X., BAO, H., SHUM, H.-Y., ET PENG, Q. 2002. A novel volume constrained smoothing method for meshes. *Graphical Models 64*, 3/4 (mai/juillet), 169–182.

LIU, X., HU, Y., ZHANG, J., TONG, X., GUO, B., ET SHUM, H.-Y. 2004. Synthesis and rendering of bidirectional texture functions on arbitrary surfaces. *IEEE Transactions on Visualization and Computer Graphics* (mai), 278–289.

LIU, Y., LIN, W.-C., ET HAYS, J. 2004. Near-regular texture analysis and manipulation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2004) 23*, 3 (août), 368–376.

LIU, Y.-J., TANG, K., ET YUEN, M. M.-F. 2004. Multiresolution free form object modeling with point sampled geometry. *Journal of Computer Science and Technology 19*, 5 (septembre), 607–617.

LIU, Y., TSIN, Y., ET LIN, W.-C. 2005. The promise and perils of near-regular texture. *International Journal of Computer Vision 62*, 1-2 (avril), 145–159.

LIU, F., BHAKAR, S., FEVENS, T., ET MUDUR, S. 2006. Environment lighting for point sampled geometry. Dans *Proceedings of the International Conference on Computer Graphics, Imaging and Visualisation (CGIV 2006)*, 522–527.

LONG, J., ET MOULD, D. 2007. Improved image quilting. Dans *Proceedings of Graphics Interface 2007*, 257–264.

LOOP, C. 1987. *Smooth Subdivision Surfaces Based on Triangles.* Mémoire de maîtrise, Department of Mathematics, University of Utah.

LORENSEN, W. E., ET CLINE, H. E. 1987. Marching cubes : A high resolution 3D surface construction algorithm. *Computers Graphics (ACM SIGGRAPH 1987) 21*, 4 (juillet), 163–169.

LOUNSBERY, M., DEROSE, T. D., ET WARREN, J. 1997. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics 16*, 1 (janvier), 34–73.

LOUNSBERY, J. M. 1994. *Multiresolution Analysis for Surfaces of Arbitrary Topological Type.* Thèse de doctorat, Department of Computer Science and Engineering, University of Washington.

MA, M., ET MANN, S. 2001. Multiresolution editing of pasted surfaces. Dans *Mathematical Methods for Curves and Surfaces : Oslo 2000*, Vanderbilt University, 273–282.

MAGDA, S., ET KRIEGMAN, D. 2003. Fast texture synthesis on arbitrary meshes. Dans *14th Eurographics Workshop on Rendering*, 82–89.

MANTLER, S., ET FUHRMANN, A. L. 2003. Fast approximate visible set determination for point sample clouds. Dans *Proceedings of the Workshop on Virtual Environments 2003*, 163–167.

MARKOSIAN, L., COHEN, J. M., CRULLI, T., ET HUGHES, J. 1999. Skin : a constructive approach to modeling free-form shapes. Dans *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 1999)*, 393–400.

MARROQUIM, R., KRAUS, M., ET CAVALCANTI, P. R. 2007. Efficient point-based rendering using image reconstruction. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2007*, 101–108.

MATUSIK, W., ZWICKER, M., ET DURAND, F. 2005. Texture design using a simplicial complex of morphable textures. *ACM Transactions on Graphics 24*, 3 (juillet), 787–794.

MAUCH, S. 2000. A fast algorithm for computing the closest point and distance transform. Tech. Rep. caltechASCI/2000.077, Applied and Computational Mathematics, California Institute of Technology.

MAX, N., ET OHSAKI, K. 1995. Rendering trees from precomputed z-buffer views. Dans *6th Eurographics Workshop on Rendering*, 45–54.

MCMILLAN, L., ET BISHOP, G. 1995. Plenoptic modeling : an image-based rendering system. Dans *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 1995)*, 39–46.

MĚCH, R., ET PRUSINKIEWICZ, P. 1996. Visual models of plants interacting with their environment. Dans *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 96)*, 397–410.

MÉMOLI, F., ET SAPIRO, G. 2004. Comparing point clouds. Dans *Proceedings of Eurographics Symposium on Geometry Processing*, 33–42.

MÉMOLI, F. 2007. On the use of Gromov-Hausdorff distances for shape comparison. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2007*, 81–90.

MERRELL, P. 2007. Example-based model synthesis. Dans *ACM Symposium on Interactive 3D Graphics and Games*, 105–112.

MIAO, L., HUANG, J., LIU, X., BAO, H., PENG, Q., ET GUO, B. 2005. Computing variation modes for point set surfaces. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2005*, 63–69.

MIAO, Y., FENG, J., ET PENG, Q. 2005. Curvature estimation of point-sampled surfaces and its applications. Dans *Proceedings of the International Conference on Computational Science and its Applications (ICCSA 2005)*, vol. 3482 de *Lecture Notes in Computer Science*, 1023–1032.

MIAO, Y., FENG, J., XIAO, C., LI, H., ET PENG, Q. 2006. Detail-preserving local editing for point-sampled geometry. Dans *Proceedings of Computer Graphics International 2006*, vol. 4035 de *Lecture Notes in Computer Science*, 673–681.

MIAO, Y., FENG, J., XIAO, C., LI, H., ET PENG, Q. 2008. High frequency geometric detail manipulation and editing for point-sampled surfaces. *The Visual Computer 24*, 2 (janvier), 125–138.

MICHAEL, T. S., ET QUINT, T. 2003. Sphere of influence graphs and the L_∞-metric. *Discrete Applied Mathematics 127*, 3 (mai), 447–460.

MITRA, N. J., ET NGUYEN, A. 2003. Estimating surface normals in noisy point cloud data. Dans *Proceedings of the 19th Annual Symposium on Computational Geometry*, 322–328.

MIYATA, K. 1990. A method of generating stone wall patterns. Dans *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 90)*, 387–394.

MOENNING, C., ET DODGSON, N. A. 2003. A new point cloud simplification algorithm. Dans *Proceedings of IASTED International Conference on Visualization, Imaging and Image Processing*, 1027–1033.

MOENNING, C., ET DODGSON, N. A. 2004. Intrinsic point cloud simplification. Dans *GraphiCon 2004*.

MOENNING, C., MÉMOLI, F., SAPIRO, G., DYN, N., ET DODGSON, N. A. 2007. Meshless geometric subdivision. *Graphical Models 69*, 3-4 (mai), 160–179.

MOORE, D. S., ET MCCABE, G. P. 1999. *Introduction to the Practice of Statistics*, Third Edition. W. H. Freeman and Company.

MORSE, B. S., YOO, T. S., CHEN, D. T., RHEINGANS, P., ET SUBRAMANIAN, K. R. 2001. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. Dans *Proceedings of IEEE International Conference on Shape Modeling and Application (SMI 2001)*, 89–98.

MÜLLER, M., KEISER, R., NEALEN, A., PAULY, M., GROSS, M., ET ALEXA, M. 2004. Point-based animation of elastic, plastic, and melting objects. Dans *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 141–151.

MURAKI, S. 1991. Volumetric shape description of range data using "blobby model". Dans *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 1991)*, 227–235.

MUSETH, K., ET LOMBEYDA, S. 2004. TetSplat real-time rendering and volume clipping of large unstructured tetrahedral meshes. Dans *IEEE Visualization 2004*, 433–440.

MUSGRAVE, F. K., KOLB, C. E., ET MACE, R. S. 1989. The synthesis and rendering of eroded fractal terrains. Dans *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 89)*, 41–50.

NAMANE, R., BOUMGHAR, F. O., ET BOUATOUCH, K. 2004. QSplat compression. Dans *Proceedings of the 3rd International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa (AFRIGRAPH 2004)*, 15–24.

NEALEN, A., ET ALEXA, M. 2003. Hybrid texture synthesis. Dans *14th Eurographics Workshop on Rendering*, 97–105.

NEALEN, A., ET ALEXA, M. 2004. Fast and high quality overlap repair for patch-based texture synthesis. Dans *Computer Graphics International 2004*, 582–585.

NEALEN, A., IGARASHI, T., SORKINE, O., ET ALEXA, M. 2007. FiberMesh : Designing freeform surfaces with 3D curves. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2007) 26*, 3 (juillet), article 41.

NEYRET, F. 1995. A general and multiscale model for volumetric textures. Dans *Proceedings of Graphics Interface 95*, 83–91.

NEYRET, F. 1996. *Textures volumiques pour la synthèse d'images*. Thèse de doctorat, Université Paris-XI - INRIA.

NEYRET, F. 1998. Modeling animating and rendering complex scenes using volumetric textures. *IEEE Transactions on Visualization and Computer Graphics 4*, 1 (janvier), 55–70.

NG, T.-Y., WEN, C., TAN, T.-S., ZHANG, X., ET KIM, Y. J. 2005. Generating an ω-tile set for texture synthesis. Dans *Proceedings of Computer Graphics International 2005*, 177–184.

NGUYEN, M. X., YUAN, X., ET CHEN, B. 2005. Geometry completion and detail generation by texture synthesis. *The Visual Computer (Special Issue for Pacific Graphics 2005) 21*, 8–10 (septembre), 669–678.

NIELSEN, M. B., ET MUSETH, K. 2006. Dynamic tubular grid : An efficient data structure and algorithms for high resolution level sets. *Journal of Scientific Computing 26*, 3 (mars), 261–299.

NOORUDDIN, F. S., ET TURK, G. 2003. Simplification and repair of polygonal models using volumetric techniques. *IEEE Transactions on Visualization and Computer Graphics 9*, 2 (avril), 191–205.

OCHOTTA, T., ET SAUPE, D. 2004. Compression of point-based 3D models by shape-adaptive wavelet coding of multi-height fields. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 173–182.

OHTAKE, Y., BELYAEV, A., ALEXA, M., TURK, G., ET SEIDEL, H.-P. 2003. Multi-level partition of unity implicits. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) 22*, 3 (juillet), 463–470.

OHTAKE, Y., BELYAEV, A., ET SEIDEL, H.-P. 2003. A multi-scale approach to 3D scattered data interpolation with compactly supported basis functions. Dans *Proceedings of IEEE International Conference on Shape Modeling and Application (SMI 2003)*, 153–161.

OHTAKE, Y., BELYAEV, A., ET SEIDEL, H.-P. 2005. 3D scattered data interpolation and approximation with multilevel compactly supported RBFs. *Graphical Models 67*, 3 (mai), 150–165.

OHTAKE, Y., BELYAEV, A., ET SEIDEL, H.-P. 2005. *Advances in Multiresolution for Geometric Modelling*. Springer, ch. Multi-scale and Adaptive CS-RBFs for Shape Reconstruction from Clouds of Points, 143–154.

OHTAKE, Y., KANAI, T., ET KASE, K. 2006. A Laplacian based approach for free-form deformation of sparse low-degree implicit surfaces. Dans *Proceedings of IEEE International Conference on Shape Modeling and Application (SMI 2006)*, article 30.

OSHER, S., ET PARAGIOS, N. 2003. *Geometric Level Set Methods in Imaging, Vision and Graphics*. Springer-Verlag.

OWENS, J. D., LUEBKE, D., GOVINDARAJU, N., HARRIS, M., KRÜGER, J., LEFOHN, A. E., ET PURCELL, T. J. 2007. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum 26*, 1 (mars), 80–113.

PAJAROLA, R., SAINZ, M., ET GUIDOTTI, P. 2004. Confetti : Object-space point blending and splatting. *IEEE Transactions on Visualization and Computer Graphics 10*, 5 (septembre), 598–608.

PAJAROLA, R., SAINZ, M., ET LARIO, R. 2005. XSplat : External memory multiresolution point visualization. Dans *Proceedings of IASTED International Conference on Visualization, Imaging and Image Processing*, 628–633.

PAJAROLA, R. 2003. Efficient level-of-details for point based rendering. Dans *Proceedings of IASTED Computer Graphics and Imaging*, 141–146.

PALMER, S. E. 1999. *Vision Science : Photons to Phenomenology*. The MIT Press.

PARISH, Y. I. H., ET MÜLLER, P. 2001. Procedural modeling of cities. Dans *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2001)*, 301–308.

PARK, S.-B., LEE, S.-U., ET CHOI, H. 2004. Multiscale surface representation and rendering for point clouds. Dans *Proceedings of the International Conference in Image Processing*, vol. 3, 1939–1942.

PARK, S., GUO, X., SHIN, H., ET QIN, H. 2005. Shape and appearance repair for incomplete point surfaces. Dans *Proceedings of the Tenth IEEE International Conference on Computer Vision*, 1260–1267.

PARKER, J. R., ET CHAN, S. 2003. Sound synthesis for the web, games, and virtual reality. Dans *Proceedings of ACM SIGGRAPH 2003 conference on Sketches & Applications*.

PAULY, M., ET GROSS, M. 2001. Spectral processing of point-sampled geometry. Dans *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2001)*, 379–386.

PAULY, M., GROSS, M., ET KOBBELT, L. 2002. Efficient simplification of point-sampled surfaces. Dans *IEEE Visualization 2002*, 163–170.

PAULY, M., KOBBELT, L., ET GROSS, M. 2002. Multiresolution modeling of point-sampled geometry. Tech. Rep. 378, Computer Science Department, Eidgenössische Technische Hochschule Zürich, septembre.

PAULY, M., KEISER, R., ET GROSS, M. 2003. Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum (Proceedings of Eurographics 2003) 22*, 3 (septembre), 281–289.

PAULY, M., KEISER, R., KOBBELT, L., ET GROSS, M. 2003. Shape modeling with point-sampled geometry. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) 22*, 3 (juillet), 641–650.

PAULY, M., MITRA, N., ET GUIBAS, L. 2004. Uncertainty and variability in point cloud surface data. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 77–84.

PAULY, M., PAI, D. K., ET GUIBAS, L. J. 2004. Quasi-rigid objects in contact. Dans *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 109–119.

PAULY, M., KEISER, R., ADAMS, B., DUTRÉ, P., GROSS, M., ET GUIBAS, L. J. 2005. Meshless animation of fracturing solids. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2005) 24*, 3 (juillet), 957–964.

PAULY, M., MITRA, N. J., GIESEN, J., GROSS, M., ET GUIBAS, L. J. 2005. Example-based 3D scan completion. Dans *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 23–32.

PAULY, M., KOBBELT, L., ET GROSS, M. 2006. Point-based multiscale surface representation. *ACM Transactions on Graphics 25*, 2 (avril), 177–193.

PAULY, M. 2003. *Point Primitives for Interactive Modeling and Processing of 3D Geometry*. Thèse de doctorat, Eidgenössische Technische Hochschule Zürich.

PAVIĆ, D., SCHÖNEFELD, V., ET KOBBELT, L. 2006. Interactive image completion with perspective correction. *The Visual Computer 22*, 9–11 (septembre), 671–681.

PEACHEY, D. R. 1985. Solid texturing of complex surfaces. Dans *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 85)*, 279–286.

PEDERSEN, H. K. 1994. Displacement mapping using flow fields. Dans *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 94)*, 279–286.

PENG, J., STRELA, V., ET ZORIN, D. 2001. A simple algorithm for surface denoising. Dans *IEEE Visualization 2001*, 107–112.

PÉREZ, P., GANGNET, M., ET BLAKE, A. 2003. Poisson image editing. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) 22*, 3 (juillet), 313–318.

PERLIN, K. 1985. An image synthesizer. Dans *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 85)*, 287–296.

PERLIN, K. 2002. Improving noise. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (juillet), 681–682.

PFEIFLE, R., ET SEIDEL, H.-P. 1996. Triangular B-splines for blending and filling of polygonal holes. Dans *Proceedings of Graphics Interface 1996*, 186–193.

PFISTER, H., ZWICKER, M., VAN BAAR, J., ET GROSS, M. 2000. Surfels : Surface elements as rendering primitives. Dans *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2000)*, 335–342.

PIEGL, L., ET TILLER, W. 1995. *The NURBS book*. Springer-Verlag.

POPESCU, V., SACKS, E., ET BAHMUTOV, G. 2004. Interactive point-based modeling from dense color and sparse depth. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 69–76.

PORTILLA, J., ET SIMONCELLI, E. P. 2000. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision 40*, 1 (octobre), 49–70.

PORUMBESCU, S. D., BUDGE, B., FENG, L., ET JOY, K. I. 2005. Shell maps. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2005) 24*, 3 (juillet), 626–633.

POULIN, P., OUIMET, M., ET FRASSON, M.-C. 1998. Interactively modeling with photogrammetry. Dans *Proceedings of Eurographics Workshop on Rendering 98*, 93–104.

POULIN, P., STAMMINGER, M., DURANLEAU, F., FRASSON, M.-C., ET DRETTAKIS, G. 2003. Interactive point-based modeling of complex objects from images. Dans *Proceedings of Graphics Interface 2003*, 11–20.

PRAUN, E., FINKELSTEIN, A., ET HOPPE, H. 2000. Lapped textures. Dans *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2000)*, 465–470.

PRICE, K., 2008. Annotated computer vision bibliography : Table of contents. `http://iris.usc.edu/Vision-Notes/bibliography/contents.html`.

PRUSINKIEWICZ, P., ET LINDENMAYER, A. 1990. *The Algorithmic Beauty of Plants*. Springer-Verlag.

PULLEN, K., ET BREGLER, C. 2002. Motion capture assisted animation : Texturing and synthesis. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (juillet), 501–508.

PULLI, K., ET LOUNSBERY, M. 1997. Hierarchical editing and rendering of subdivision surfaces. Tech. Rep. UW-CSE-97-04-07, Department of Computer Science and Engineering, University of Washington.

QU, L., YUAN, X., NGUYEN, M. X., MEYER, G. W., CHEN, B., ET WINDSHEIMER, J. E. 2006. Perceptually guided rendering of textured point-based models. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2006*, 95–102.

REALVIZ S.A. REALVIZ ImageModeler : Image based modeling and photogrammetry software. `http://imagemodeler.realviz.com/photomodeling-software-products/imagemodeler/modeling-software.html`.

REEVES, W. T., ET BLAU, R. 1985. Approximate and probabilistic algorithms for shading and rendering structured particle systems. Dans *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 85)*, 313–322.

REEVES, W. T. 1983. Particle systems — technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics 2*, 2 (avril), 91–108.

REN, L., ET ZWICKER, H. P. M. 2002. Object space EWA surface splatting : A hardware accelerated approach to high quality point rendering. *Computer Graphics Forum (Proceedings of Eurographics 2002) 21*, 3 (septembre), 461–470.

RENIERS, D., ET TELEA, A. 2006. Extreme simplification and rendering of point sets using algebraic multigrid. *Computing and Visualization in Science*, Online First (À paraître).

REQUICHA, A. A. G., ET VOELCKER, H. B. 1977. Constructive solid geometry. Tech. Rep. 25, University of Rochester, novembre.

REUTER, P., TOBOR, I., SCHLICK, C., ET DEDIEU, S. 2003. Point-based modelling and rendering using radial basis functions. Dans *Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, 111–118.

REUTER, P., SCHMITT, B., SCHLICK, C., ET PASKO, A. A. 2004. Interactive solid texturing using point-based multiresolution representations. *Journal of WSCG 12*, 1-3 (février), 363–370.

REUTER, P., JOYOT, P., TRUNZLER, J., BOUBEKEUR, T., ET SCHLICK, C. 2005. Surface reconstruction with enriched reproducing kernel particle approximation. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2005*, 79–87.

ROSENTHAL, P., ET LINSEN, L. 2008. Image-space point cloud rendering. Dans *Proceedings of Computer Graphics International 2008*.

RUNIONS, A., SAMAVATI, F., ET PRUSINKIEWICZ, P. 2007. Ribbons : A representation for point clouds. *The Visual Computer 23*, 9 (août), 945–954.

RUSINKIEWICZ, S., ET LEVOY, M. 2000. QSplat : A multiresolution point rendering system for large meshes. Dans *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2000)*, 343–352.

RUSINKIEWICZ, S., ET LEVOY, M. 2001. Streaming qsplat : A viewer for networked visualization of large, dense models. Dans *Symposium on Interactive 3D Graphics*, 63–68.

RUSSELL, S., ET NORVIG, P. 2002. *Artificial Intelligence — A Modern Approach*. Prentice Hall.

SABHA, M., ET DUTRÉ, P. 2005. A fast texture synthesis technique using spatial neighborhood. Tech. Rep. CW400, Department of Computer Science, Katholieke Universiteit Leuven, septembre.

SABHA, M., ET DUTRÉ, P. 2006. Image welding for texture synthesis. Dans *Vision, Modeling, and Visualization 2006*, 97–104.

SABHA, M., ET DUTRÉ, P. 2007. Feature-based texture synthesis using Voronoi diagram. Tech. Rep. CW492, Department of Computer Science, Katholieke Universiteit Leuven, mai.

SABHA, M., ET DUTRÉ, P. 2007. Texture synthesis using exact neighborhood matching. *Computer Graphics Forum 26*, 2 (juin), 131–142.

SAINZ, M., PAJAROLA, R., ET LARIO, R. 2004. Points reloaded : Point-based rendering revisited. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 121–128.

SAINZ, M., PAJAROLA, R., SUSIN, A., ET MERCADE, A. 2004. A simple approach for point-based object capturing and rendering. *IEEE Computer Graphics and Applications 24*, 4 (juillet), 24–33.

SAMET, H. 2006. *Foundations of Multidimensional and Metric Data Structures*. The Morgan Kaufmann Series in Computer Graphics. Morgan Kaufmann, août.

SANDER, P. V., WOOD, Z. J., GORTLER, S. J., SNYDER, J., ET HOPPE, H. 2003. Multi-chart geometry images. Dans *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 146–155.

SANKARANARAYANAN, J., SAMET, H., ET VARSHNEY, A. 2006. A fast k-neighborhood algorithm for large point-clouds. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2006*, 75–84.

SATO, Y., WHEELER, M. D., ET IKEUCHI, K. 1997. Object shape and reflectance modeling from observation. Dans *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 97)*, 379–388.

SAVCHENKO, V. V., PASKO, A. A., OKUNEV, O. G., ET KUNII, T. L. 1995. Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum 14*, 4 (octobre), 181–188.

SAYOOD, K. 2000. *Introduction to Data Compression (Second Edition)*. Morgan Kaufmann Publishers Inc.

SCHALL, O., BELYAEV, A., ET SEIDEL, H.-P. 2005. Robust filtering of noisy scattered point data. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2005*, 71–78.

SCHALL, O., BELYAEV, A., ET SEIDEL, H.-P. 2007. Feature-preserving non-local denoising of static and time-varying range data. Dans *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling*, 217–222.

SCHAUFLER, G., ET JENSEN, H. W. 2000. Ray tracing point sampled geometry. Dans *11th Eurographics Workshop on Rendering*, 319–328.

SCHMIDT, R., WYVILL, B., SOUSA, M. C., ET JORGE, J. A. 2005. Shapeshop : Sketch-based solid modeling with blobtrees. Dans *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, 53–62.

SCHNABEL, R., ET KLEIN, R. 2006. Octree-based point-cloud compression. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2006*, 111–120.

SCHNABEL, R., MOESER, S., ET KLEIN, R. 2007. A parallelly decodeable compression scheme for efficient point-cloud rendering. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2007*, 119–128.

SCHÖDL, A., SZELISKI, R., SALESIN, D. H., ET ESSA, I. 2000. Video textures. Dans *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2000)*, 489–498.

SEITZ, S. M., ET DYER, C. R. 1999. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision 35*, 2, 151–173.

SHADE, J., GORTLER, S., HE, L.-W., ET SZELISKI, R. 1998. Layered depth images. Dans *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 1998)*, 231–242.

SHARF, A., ALEXA, M., ET COHEN-OR, D. 2004. Context-based surface completion. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2004) 23*, 3 (août), 878–887.

SHARF, A., LEWINER, T., SHKLARSKI, G., TOLEDO, S., ET COHEN-OR, D. 2007. Interactive topology-aware surface reconstruction. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2007) 26*, 3 (juillet), article 43.

SHEN, C., O'BRIEN, J. F., ET SHEWCHUK, J. R. 2004. Interpolating and approximating implicit surfaces from polygon soup. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2004) 23*, 3 (juillet), 896–904.

SHLYAKHTER, I., ROZENOER, M., DORSEY, J., ET TELLER, S. 2001. Reconstructing 3D tree models from instrumented photographs. *IEEE Computer Graphics and Applications 21*, 3, 53–61.

SIGG, C., WEYRICH, T., BOTSCH, M., ET GROSS, M. 2006. GPU-based ray-casting of quadratic surfaces. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2006*, 59–65.

SIMS, K. 1990. Particle animation and rendering using data parallel computation. Dans *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 90)*, 405–413.

SINGH, J., ET DANA, K. J. 2004. Clustering and blending for texture synthesis. *Pattern Recognition Letters 25*, 6 (avril), 619–629.

SINGH, J. M., ET NARAYANAN, P. J. 2006. Progressive decomposition of point clouds without local planes. Dans *5th Indian Conference on Computer Vision, Graphics and Image Processing*, vol. 4338 de *Lecture Notes in Computer Science*, 364–375.

SINGH, G., MEMOLI, F., ET CARLSSON, G. 2007. Topological methods for the analysis of high dimensional data sets and 3D object recognition. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2007*, 91–100.

SMITH, A. R. 1984. Plants, fractals, and formal languages. Dans *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 84)*, 1–10.

SNOW, D., VIOLA, P., ET ZABIH, R. 2000. Exact voxel occupancy with graph cuts. Dans *Computer Vision and Pattern Recognition Conference*, vol. 1, 345–352.

SOLENTHALER, B., ZHANG, Y., ET PAJAROLA, R. 2007. Efficient refinement of dynamic point data. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2007*, 65–72.

SOLER, C., CANI, M.-P., ET ANGELIDIS, A. 2002. Hierarchical pattern mapping. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (juillet), 673–680.

SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RÖSSL, C., ET SEIDEL, H.-P. 2004. Laplacian surface editing. Dans *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 175–184.

The Stanford 3D scanning repository. `http://graphics.stanford.edu/data/3Dscanrep/`.

STAMMINGER, M., ET DRETTAKIS, G. 2001. Interactive sampling and rendering for complex and procedural geometry. Dans *12th Eurographics Workshop on Rendering*, 151–162.

STEINEMANN, D., OTADUY, M. A., ET GROSS, M. 2006. Fast arbitrary splitting of deforming objects. Dans *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 63–72.

STEINEMANN, D., OTADUY, M. A., ET GROSS, M. 2007. Efficient bounds for point-based animations. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2007*, 57–64.

STOLLNITZ, E. J., DEROSE, T. D., ET SALESIN, D. H. 1996. *Wavelets for Computer Graphics : Theory and Applications*. Morgan Kaufmann.

STROUSTRUP, B. 2000. *The C++ Programming Language*. Addison-Wesley.

SUBR, K. S., GOPI, M., PAJAROLA, R., ET SAINZ, M. 2003. Point light field for point rendering systems. Tech. Rep. UCI-ICS-03-28, Department of Computer Science, University of California Irvine.

SULLIVAN, S., ET PONCE, J. 1998. Automatic model construction and pose estimation from photographs using triangular splines. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 10 (octobre), 1091–1096.

SUN, J., YUAN, L., JIA, J., ET SHUM, H.-Y. 2005. Image completion with structure propagation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2005) 24*, 3 (juillet), 861–868.

SUN, X., ROSIN, P. L., MARTIN, R. R., ET LANGBEIN, F. C. 2007. Random walks for mesh denoising. Dans *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling*, 11–22.

SZELISKI, R., ET TONNESEN, D. 1992. Surface modeling with oriented particle systems. Dans *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 92)*, 185–194.

SZELISKI, R. 1993. Rapid octree construction from image sequences. *Computer Vision, Graphics and Image Processing 58*, 1 (juillet), 23–32.

TALTON, J. O., CARR, N. A., ET HART, J. C. 2005. Voronoi rasterization of sparse point sets. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2005*, 33–37.

TAPONECCO, F., ET ALEXA, M. 2003. Vector field visualization using Markov random field texture synthesis. Dans *Proceedings of the Symposium on Data Visualization 2003*, 195–202.

TASDIZEN, T., WHITAKER, R., BURCHARD, P., ET OSHER, S. 2002. Geometric surface smoothing via anisotropic diffusion of normals. Dans *IEEE Visualization 2002*, 125–132.

TAUBIN, G. 1995. A signal processing approach to fair surface design. Dans *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 95)*, 351–358.

TAUBIN, G. 2002. Detecting and reconstructing subdivision connectivity. *The Visual Computer 18*, 5-6 (août), 357–367.

THANGUDU, K., GADE, L., SINGH, J. M., ET NARAYANAN, P. J. 2007. Point based representations for hierarchical environments. Dans *IEEE International Conference on Computing: Theory and Applications*, 574–578.

TOBOR, I., REUTER, P., ET SCHLICK, C. 2004. Multiresolution reconstruction of implicit surfaces with attributes from large unorganized point sets. Dans *Proceedings of IEEE International Conference on Shape Modeling and Application (SMI 2004)*, 19–30.

TOBOR, I., REUTER, P., ET SCHLICK, C. 2006. Reconstructing multi-scale variational partition of unity implicit surfaces with attributes. *Graphical Models 68*, 1 (janvier), 25–41.

TOMASI, C., ET KANADE, T. 1992. Shape and motion from image streams under ortography : a factorization method. *International Journal of Computer Vision 9*, 2 (novembre), 137–154.

TONG, X., ZHANG, J., LIU, L., WANG, X., GUO, B., ET SHUM, H.-Y. 2002. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (juillet), 665–672.

TOSCANI, G., ET FAUGERAS, O. 1987. Structure from motion using the reconstruction and reprojection technique. Dans *IEEE Computer Society Workshop on Computer Vision*, 345–348.

TRUCCO, E., ET VERRI, A. 1998. *Introductory Techniques for 3-D Computer Vision.* Prentice Hall.

TURK, G., ET O'BRIEN, J. F. 2002. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics 21*, 4 (octobre), 855–873.

TURK, G. 1991. Generating textures on arbitrary surfaces using reaction-diffusion. Dans *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 91)*, 289–298.

TURK, G. 2001. Texture synthesis on surfaces. Dans *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2001)*, 347–354.

UPSTILL, S. 1989. *The RenderMan Companion : A Programmer's Guide to Realistic Computer Graphics.* Addison-Wesley Longman Publishing Co., Inc.

VALETTE, S., ET PROST, R. 2004. Wavelet-based multiresolution analysis of irregular surface meshes. *IEEE Transactions on Visualization and Computer Graphics 10*, 2 (mars), 113–122.

WAHBA, G. 1990. *Spline Models for Observational Data*, vol. 59 de *CBMS - NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics.

WALD, I., ET SEIDEL, H.-P. 2005. Interactive ray tracing of point-based models. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2005*, 9–16.

WAND, M., ET STRASSER, W. 2003. Multi-resolution point-sample raytracing. Dans *Proceedings of Graphics Interface 2003*, 139–148.

WAND, M., BERNER, A., BOKELOH, M., FLECK, A., HOFFMANN, M., JENKE, P., MAIER, B., STANEKER, D., ET SCHILLING, A. 2007. Interactive editing of large point clouds. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2007*, 37–45.

WANG, L., ET MUELLER, K. 2004. Generating sub-resolution detail in images and volumes using constrained texture synthesis. Dans *IEEE Visualization 2004*, 75–82.

WANG, X., TONG, X., LIN, S., HU, S., GUO, B., ET SHUM, H.-Y. 2004. Generalized displacement maps. Dans *15th Eurographics Workshop on Rendering*, 227–233.

WANG, Y., WANG, W., ET WU, E. 2005. Optimizing the parameters for patch-based texture synthesis. Dans *Proceedings of International Conference on Virtual Reality Continuum and Its Applications (VRCIA 2006)*, 75–82.

WANG, R., ZHANG, S., ET YE, X. 2007. A novel simplification algorithm for point-sampled surfaces. Dans *International Conference on Multimedia and Ubiquitous Engineering (MUE 2007)*, 573–578.

WANG, H., SCHEIDEGGER, C. E., ET SILVA, C. T. 2008. Optimal bandwidth selection for MLS surfaces. Dans *Proceedings of IEEE International Conference on Shape Modeling and Application (SMI 2008)*, 111–120.

WANG, H. 1961. Proving theorems by pattern recognition II. *Bell Systems Technical Journal 40* (janvier), 1–41.

WARREN, J. D., ET WEIMER, H. 2001. *Subdivision Methods for Geometric Design : A Constructive Approach*. Morgan Kaufmann Publishers Inc.

WASCHBÜSCH, M., GROSS, M., EBERHARD, F., LAMBORAY, E. C., ET WÜRMLIN, S. 2004. Progressive compression of point-sampled models. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 95–102.

WATERS, K. W., CO, C. S., ET JOY, K. I. 2006. Using difference intervals for time-varying isosurface visualization. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (septembre), 1275–1282.

WEI, L.-Y., ET LEVOY, M. 2000. Fast texture synthesis using tree-structured vector quantization. Dans *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2000)*, 479–488.

WEI, L.-Y., ET LEVOY, M. 2001. Texture synthesis over arbitrary manifold surfaces. Dans *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2001)*, 355–360.

WEI, L.-Y., ET LEVOY, M. 2002. Order-independent texture synthesis. Tech. Rep. TR-2002-01, Computer Science Department, Stanford University, avril.

WEI, L.-Y., HAN, J., ZHOU, K., BAO, H., GUO, B., ET SHUM, H.-Y. 2008. Inverse texture synthesis. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2008) 27*, 3 (août).

WEI, L.-Y. 2002. *Texture Synthesis by Fixed Neighborhood Searching.* Thèse de doctorat, Department of Electrical Engineering, Stanford University.

WEI, L.-Y. 2004. Tile-based texture mapping on graphics hardware. Dans *Proceedings of ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 55–63.

WEILER, K. 1985. Edge-based data structures for solid modeling in curved-surface environment. *IEEE Computer Graphics and Applications 5*, 1 (janvier), 21–40.

WELSH, T., ET MUELLER, K. 2003. A frequency-sensitive point hierarchy for images and volumes. Dans *IEEE Visualization 2003*, 425–432.

WENDLAND, H. 1995. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics 4*, 1 (décembre), 389–396.

WEYRICH, T., PAULY, M., KEISER, R., HEINZLE, S., SCANDELLA, S., ET GROSS, M. 2004. Post-processing of scanned 3D surface data. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 85–94.

WEYRICH, T., FLAIG, C., HEINZLE, S., MALL, S., AILA, T., ROHRER, K., FASNACHT, D. B., FELBER, N., OETIKER, S., KAESLIN, H., BOTSCH, M., ET GROSS, M. 2007. A hardware architecture for surface splatting. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2007) 26*, 3 (juillet), article 90.

WHEELER, M. K., SATO, Y., ET IKEUCHI, K. 1998. Consensus surfaces for modeling 3D objects from multiple range images. Dans *IEEE International Conference on Computer Vision*, 917–924.

WICKE, M., TESCHNER, M., ET GROSS, M. 2004. CSG tree rendering for point-sampled objects. Dans *Proceedings of Pacific Graphics 2004*, 160–168.

WICKE, M., OLIBET, S., ET GROSS, M. 2005. Conversion of point-sampled models to textured meshes. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2005*, 119–124.

WICKE, M., STEINEMANN, D., ET GROSS, M. 2005. Efficient animation of point-sampled thin shells. *Computer Graphics Forum (Proceedings of Eurographics 2005) 24*, 3 (septembre), 667–676.

WICKE, M., HATT, P., PAULY, M., MÜLLER, M., ET GROSS, M. 2006. Versatile virtual materials using implicit connectivity. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2006*, 137–144.

WILSON, B., MA, K.-L., ET MCCORMICK, P. S. 2002. A hardware-assisted hybrid rendering technique for interactive volume visualization. Dans *IEEE Symposium on Volume Visualization and Graphics 2002*, 123–130.

WIMMER, M., ET SCHEIBLAUER, C. 2006. Instant points : Fast rendering of unprocessed point clouds. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2006*, 129–136.

WOOD, Z., HOPPE, H., DESBRUN, M., ET SCHRÖDER, P. 2004. Removing excess topology from isosurfaces. *ACM Transactions on Graphics 23*, 2 (avril), 190–208.

WU, J., ET KOBBELT, L. 2004. Optimized sub-sampling of point sets for surface splatting. *Computer Graphics Forum (Proceedings of Eurographics 2004) 23*, 3 (septembre), 643–652.

WU, Q., ET YU, Y. 2004. Feature matching and deformation for texture synthesis. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2004) 23*, 3 (août), 364–367.

WU, J., ZHANG, Z., ET KOBBELT, L. 2005. Progressive splatting. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2005*, 25–32.

WU, F., ZHANG, C., ET HE, J. 2007. An evolutionary system for near-regular texture synthesis. *Pattern Recognition 40*, 8 (août), 2271–2282.

XIAO, C., MIAO, Y., LIU, S., ET PENG, Q. 2006. A dynamic balanced flow for filtering point-sampled geometry. *The Visual Computer 22*, 3 (mars), 210–219.

XIAO, C., ZHENG, W., MIAO, Y., ZHAO, Y., ET PENG, Q. 2007. A unified method for appearance and geometry completion of point set surfaces. *The Visual Computer 23*, 6 (mai), 433–443.

XU, H., ET CHEN, B. 2004. Stylized rendering of 3D scanned real world environments. Dans *Proceedings of the 3rd International Symposium on Non-Photorealistic Animation and Rendering (NPAR 2006)*, 25–34.

XU, Y.-Q., GUO, B., ET SHUM, H.-Y. 2000. Chaos mosaic : Fast and memory efficient texture synthesis. Tech. Rep. MSR-TR-2000-32, Microsoft Research, avril.

XU, Y., ZHU, S.-C., GUO, B., ET SHUM, H.-Y. 2001. Asymptotically admissible texture synthesis. Dans *International Workshop of Statistical and Computational Theory of Vision.*

XU, H., NGUYEN, M. X., YUAN, X., ET CHEN, B. 2004. Interactive silhouette rendering for point-based models. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 13–18.

YAMAZAKI, I., NATARAJAN, V., BAI, Z., ET HAMANN, B. 2006. Segmenting point sets. Dans *Proceedings of IEEE International Conference on Shape Modeling and Application (SMI 2006)*, article 6.

YANG, P., ET QIAN, X. 2007. Direct computing of surface curvatures for point-set surfaces. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2007*, 29–36.

YANG, P., ET QIAN, X. 2008. Adaptive slicing of moving least squares surfaces : Toward direct manufacturing from point cloud data. *ASME Transactions Journal of Computing and Information Science in Engineering 8*, 3 (septembre).

YANG, Z., ET WAN KIM, T. 2007. Moving parabolic approximation of point clouds. *Computer Aided Design 39*, 12 (décembre), 1091–1112.

YING, L., HERTZMANN, A., BIERMANN, H., ET ZORIN, D. 2001. Texture and shape synthesis on surfaces. Dans *12th Eurographics Workshop on Rendering*, 301–312.

YU, Z., ET SAN WONG, H. 2006. An efficient local clustering approach for simplification of 3D point-based computer graphics models. Dans *International Conference on Media & Expo*, 2065–2068.

YU, Y., ZHOU, K., XU, D., SHI, X., BAO, H., GUO, B., ET SHUM, H.-Y. 2004. Mesh editing with Poisson-based gradient field manipulation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2004) 23*, 3 (août), 644–651.

YVART, A., HAHMANN, S., ET BONNEAU, G.-P. 2005. Hierarchical triangular splines. *ACM Transactions on Graphics 24*, 4 (octobre), 1374–1391.

ZELEZNIK, R. C., HERNDON, K. P., ET HUGHES, J. F. 1996. SKETCH : an interface for sketching 3D scenes. Dans *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 96)*, 163–170.

ZELINKA, S., ET GARLAND, M. 2002. Towards real-time texture synthesis with the jump map. Dans *13th Eurographics Workshop on Rendering*, 99–104.

ZELINKA, S., ET GARLAND, M. 2003. Interactive texture synthesis on surfaces using jump maps. Dans *14th Eurographics Workshop on Rendering*, 90–96.

ZELINKA, S., ET GARLAND, M. 2004. Jump map-based interactive texture synthesis. *ACM Transactions on Graphics 23*, 4 (octobre), 930–962.

ZELINKA, S., ET GARLAND, M. 2004. Mesh modelling with curve analogies. Dans *Proceedings of Pacific Graphics 2004*, 94–98.

ZELINKA, S., ET GARLAND, M. 2004. Similarity-based surface modelling using geodesic fans. Dans *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 209–218.

ZELINKA, S., ET GARLAND, M. 2006. Surfacing by numbers. Dans *Proceedings of Graphics Interface 2006*, 107–113.

ZELINKA, S., FANG, H., GARLAND, M., ET HART, J. C. 2005. Interactive material replacement in photographs. Dans *Proceedings of Graphics Interface 2005*, 227–232.

ZHANG, H., ET KAUFMAN, A. 2007. A classification-based rendering method for point models. *Computers & Graphics 31*, 5 (octobre), 730–736.

ZHANG, Y., ET PAJAROLA, R. 2006. Single-pass point rendering and transparent shading. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2006*, 37–48.

ZHANG, Y., ET PAJAROLA, R. 2007. Deferred blending : Image composition for single-pass point rendering. *Computers & Graphics 31*, 2 (avril), 175–189.

ZHANG, R., TSAI, P. S., CRYER, J., ET SHAH, M. 1999. Shape from shading : A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence 21*, 8 (août), 690–706.

ZHANG, J., ZHOU, K., VELHO, L., GUO, B., ET SHUM, H.-Y. 2003. Synthesis of progressively-variant textures on arbitrary surfaces. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) 22*, 3 (juillet), 295–302.

ZHANG, D.-H., YUE, T., ET CHEN, Y.-H. 2005. Multi-scale surface representation of point-sampled geometry. Dans *International Symposium on Image and Signal Processing and Analysis*, 371–376.

ZHANG, N., ZHOU, X., SHA, D., YUAN, X., TAMMA, K., ET CHEN, B. 2006. Integrating mesh and meshfree methods for physics-based fracture and debris cloud simulation. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2006*, 145–154.

ZHANG, Z. 1994. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision 13*, 2 (octobre), 119–152.

ZHAO, H.-K., OSHER, S., ET FEDKIW, R. 2001. Fast surface reconstruction using the level set method. Dans *IEEE Workshop on Variational and Level Set Methods (VLSM 2001)*, 194–201.

ZHENG, W., SUN, H., BAO, H., ET PENG, Q. 2002. Rendering of virtual environments based on polygonal & point-based models. Dans *Proceedings of ACM Symposium on Virtual Reality Software and Technology*, 25–32.

ZHENG, J. Y. 1994. Acquiring 3-D models from a sequence of contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence 16*, 2 (février), 163–178.

ZHOU, K., HUANG, X., WANG, X., TONG, Y., DESBRUN, M., GUO, B., ET SHUM, H.-Y. 2006. Mesh quilting for geometric texture synthesis. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2006) 25*, 3 (juillet), 690–697.

ZHU, S. C., WU, Y., ET MUMFORD, D. 1998. Filters, random fields and maximum entropy (FRAME) : Towards a unified theory for texture modeling. *International Journal of Computer Vision 27*, 2 (avril), 107–126.

ZHU, S.-C., EN GUO, C., WANG, Y., ET XU, Z. 2005. What are textons ? *International Journal of Computer Vision 62*, 1–2 (janvier), 121–143.

ZIEGLER, G., TEVS, A., THEOBALT, C., ET SEIDEL, H.-P. 2006. On-the-fly point clouds through histogram pyramids. Dans *Vision, Modeling, and Visualization 2006*, 137–144.

ZORIN, D., SCHRÖDER, P., ET SWELDENS, W. 1997. Interactive multiresolution mesh editing. Dans *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 97)*, 259–268.

ZORIN, D., SCHRÖDER, P., DEROSE, T., KOBBELT, L., LEVIN, A., ET SWELDENS, W., 2000. Subdivision for modeling and animation. SIGGRAPH 2000 Course Notes.

ZORIN, D. 1998. *Stationary Subdivision and Multiresolution Surface Representations*. Thèse de doctorat, Department of Computer Science, California Institute of Technology, Pasadena.

ZWICKER, M., ET GOTSMAN, C. 2004. Meshing point clouds using spherical parameterization. Dans *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, 173–180.

ZWICKER, M., PFISTER, H., VAN BAAR, J., ET GROSS, M. 2001. EWA volume splatting. Dans *IEEE Visualization 2001*, 29–36.

ZWICKER, M., PFISTER, H., VAN BAAR, J., ET GROSS, M. 2001. Surface splatting. Dans *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2001)*, 371–378.

ZWICKER, M., PAULY, M., KNOLL, O., ET GROSS, M. 2002. Pointshop 3D : An interactive system for point-based surface editing. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) 21*, 3 (juillet), 323–329.

ZWICKER, M., RÄSÄNEN, J., BOTSCH, M., DACHBACHER, C., ET PAULY, M. 2004. Perspective accurate splatting. Dans *Proceedings of Graphics Interface 2004*, 247–254.