



HAL
open science

Modèles pour la Création Interactive et Intuitive d'Objets Tridimensionnels

Loic Barthe

► **To cite this version:**

Loic Barthe. Modèles pour la Création Interactive et Intuitive d'Objets Tridimensionnels. Synthèse d'image et réalité virtuelle [cs.GR]. Université Paul Sabatier - Toulouse III, 2011. tel-00918452

HAL Id: tel-00918452

<https://theses.hal.science/tel-00918452v1>

Submitted on 13 Dec 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



MANUSCRIT

En vue de l'obtention de l'

HABILITATION A DIRIGER LES RECHERCHES

Délivrée par *l'Université Toulouse III - Paul Sabatier*

Présentée et soutenue par **Loïc Barthe**

Le 08 Juillet 2011

Modèles pour la Création Interactive et Intuitive d'Objets Tridimensionnels

JURY

Rapporteurs : John Hughes - Professeur, Brown University
Christophe Schlick - Professeur, INRIA Bordeaux
Dominique Faudot - Professeur, Université de Bourgogne

Examineurs : Marie-Paule Cani - Professeur, INRIA Grenoble
Bruno Lévy - Directeur de Recherche, INRIA Nancy
Mathias Paulin - Professeur, Université de Toulouse

Remerciements

Mes premiers remerciements vont à ma famille, tout d’abord ma compagne Carole et mes enfants Quentin, Clémentine, Corentin, pour leur soutien inconditionnel, la joie qu’ils m’apportent et leur patience, puis ses autres membres, Marie-Noëlle, Gérard, Nicolas, Guillaume, Antoine, Marie-Juliette, Charlotte, Thibault, Alice, Léa, Jérémy, Francine, Jean-Claude et tous les autres, tous, autant qu’ils sont.

Je remercie aussi toutes les personnes qui m’entourent (ou m’ont entouré) au quotidien et qui, de près ou de loin, ont une part dans ces travaux : Luc, David, Géraldine, Roger, Cédric, Patrice, Minica, Pierre. Avec eux, tous les “Vortexiens” : Yves, Vincent, Hervé, Alain, Jean, Romulus, Philippe, Stéphane.

Je remercie spécialement René Caubet et Véronique Gaildrat, mes premiers encadrants, ainsi que toutes les personnes qui m’ont fait confiance directement : Neil Dodgson, Malcolm A. Sabin, Leif Kobbelt, Mathias Paulin, Jean-Pierre Jessel ou indirectement : Jean Vignolle, Luis Fariñas del Cerro, Jean-Paul Bahsoun.

Je tiens aussi à remercier toutes les personnes avec qui j’ai collaboré pour leur patience, leurs suggestions avisées et leurs qualités humaines et professionnelles : Brian Wyvill, Marie-Paule Cani, Mario Botsch. Toutes les personnes qui, par un mot, une remarque, une phrase, m’ont apporté plus d’énergie et de courage qu’ils ne s’en doutent : Stephan Bischoff, Silke van Betteraey, Martin Marinov, Jianhua Wu, Mohamed Hassan, Ergun Akleman, Mathieu Desbrun, John Hughes, Dominique Bechmann, Sylvain Lefebvre.

Je remercie mes rapporteurs : John Hughes, Christophe Schlick et Dominique Faudot pour le temps qu’ils ont passé à lire et évaluer ce document et je remercie plus largement tous les membres de mon jury, Bruno Levy, Marie-Paule Cani. Je remercie aussi Claudette Cayrol qui a été mon rapporteur local et qui a présenté mon dossier lors des commissions de l’Université.

Je remercie enfin tous les étudiants qui m’ont supporté le temps d’une collaboration, d’un stage de recherche ou d’un encadrement un peu plus long . . . , Marc Cardle, Gaël Guennebaud, Anca Alexe, Vincent Forest, Olivier Gourmel, Anthony Pajot, Marion Dunyach, Adrien Bernhardt, Hong Coong Binh, François Teurlay, Azadéh Razavi, Cédric Partinico, Maya Hussein, Noura Faraj, Philippe Ercolessi, Jean-Patrick Rocchia, Rodolphe Vaillant.

Mes derniers remerciements vont aux personnes que je n’ai pas citées mais que je cotoie régulièrement et avec qui j’entretiens une relation amicale et/ou professionnelle fructueuse et enrichissante.

A ma compagne et mes enfants, ma famille, mes amis : mes joyaux.

Table des matières

| | |
|---|-----------|
| Chapitre 1 Préface | 1 |
| Chapitre 2 Introduction | 3 |
| 2.1 Contexte général | 3 |
| 2.1.1 Contexte spécifique et problématiques | 4 |
| 2.1.2 Contexte scientifique et organisation du document | 5 |
| Chapitre 3 Surfaces de subdivision | 7 |
| 3.1 Introduction | 7 |
| 3.2 Etude des comportements indésirables dans la subdivision stationnaire | 8 |
| 3.2.1 Distorsions dans la taille des faces du maillage subdivisé | 8 |
| 3.2.2 Précision linéaire | 10 |
| 3.2.3 Oscillations radiales | 10 |
| 3.3 Amélioration des schémas de subdivision autour des points irréguliers | 11 |
| 3.4 Autour de ces travaux, quelques curiosités | 13 |
| 3.5 Subdivision d’un nuage de points non organisé | 14 |
| 3.6 Positionnement dans notre contexte et conclusion | 15 |
| Chapitre 4 Modélisation par esquisses | 17 |
| 4.1 Introduction | 17 |
| 4.2 Premiers travaux et reproduction de vieux problèmes | 18 |
| 4.3 Adapter le niveau de détail de la reconstruction au zoom et éviter les oscillations | 19 |
| 4.4 Discussions et retour vers l’étude des modèles volumiques | 21 |
| Chapitre 5 Surfaces implicites | 23 |
| 5.1 Introduction | 23 |
| 5.2 Fonctions potentiel à support global | 24 |
| 5.3 Fonctions potentiel à support compact | 27 |
| 5.4 La trêve | 28 |
| 5.5 Vers une nouvelle génération d’opérateurs de composition | 29 |

| | | |
|-----------------------------------|---|------------|
| 5.6 | Discussions et perspectives : de nouveaux projets | 33 |
| Chapitre 6 Conclusion | | 37 |
| Bibliographie | | 41 |
| Chapitre 7 Publications | | 49 |
| 7.1 | Surfaces de subdivision | 51 |
| 7.1.1 | Artifacts in recursive subdivision surfaces. | 51 |
| 7.1.2 | Simple computation of the eigencomponents of a subdivision matrix in the Fourier domain. | 63 |
| 7.1.3 | Subdivision scheme tuning around extraordinary vertices. | 79 |
| 7.1.4 | Interpolatory Refinement for Real-Time Processing of Point-Based Geo- metry. | 109 |
| 7.2 | Modélisation par esquisses | 121 |
| 7.2.1 | Interactive modelling from sketches using spherical implicit functions. . . | 121 |
| 7.2.2 | Shape modelling by sketching using convolution surfaces. | 133 |
| 7.2.3 | Matisse : Painting 2D regions for Modeling Free-Form Shapes. | 139 |
| 7.3 | Surfaces implicites | 149 |
| 7.3.1 | Two-dimensional potential fields for advanced implicit modeling operators. | 149 |
| 7.3.2 | Controllable Binary CSG Operators for “Soft Objects”. | 161 |
| 7.3.3 | Fitted BVH for Fast Raytracing of Metaballs. | 183 |
| 7.3.4 | Implicit Blending Revisited. | 193 |
| 7.3.5 | Gradient-based Implicit Modeling. | 203 |
| Chapitre 8 Curriculum vitæ | | 213 |
| 8.1 | Déroulement de la carrière | 213 |
| 8.2 | Encadrement d’étudiants en thèse | 213 |
| 8.3 | Management/animation scientifique et contractuelle | 214 |
| 8.4 | Rayonnement | 214 |
| 8.5 | Activité de recherche, collaborations et production scientifique | 216 |

1

Préface

Les travaux que nous allons aborder dans ce document s'étalent sur ces neuf dernières années. Ils portent sur la modélisation géométrique de formes complexes. Sur ces travaux, j'ai tout d'abord profité de l'encadrement de trois chercheurs : Malcolm A. Sabin, Neil Dodgson et Leif Kobbelt. J'ai ensuite encadré deux étudiants en stage de Master deuxième année - Jean-Patrick Rocchia et Rodolphe Vaillant - et participé à l'encadrement de quatre étudiants en thèse : Anca Alexe, Gaël Guennebaud, Adrien Bernhardt et Olivier Gourmel. J'ai enfin collaboré étroitement avec Brian Wyvill et Marie-Paule Cani et de nouvelles collaborations naissent avec Gaël Guennebaud et Mario Botsch, Mario avec qui j'encadre Marion Dunyach qui commence une thèse en co-tutelle.

Tous ces travaux sont ceux pour lesquels ma participation a été la plus active, voire dont j'ai été l'instigateur. Ce sont aussi ceux qui touchent au plus près mes compétences principales et mon projet de recherche personnel. Grâce à la confiance que m'ont portée mes collègues Véronique Gaidrat et Mathias Paulin, et les chercheurs avec qui je collabore, Marie-Paule Cani et Brian Wyvill, j'ai encadré ou co-encadré (certains encadrements étant toujours en cours) dix étudiants de Master deuxième année et sept étudiants en thèse depuis mon recrutement à l'IRIT. J'ai ainsi pu, dès mon recrutement, participer à l'encadrement de plusieurs travaux, dont certains sont bien en dehors du cadre de ce manuscrit. Je pense notamment à des recherches menées sur des thèmes comme l'animation dirigée par la musique, le rendu de géométries définies par des nuages de points, la modélisation d'objets volumiques, la peinture sur des surfaces implicites, la génération d'ombres douces, l'intégration de Monte-Carlo pour le rendu haute qualité, les grandes déformations de maillages avec changement de topologie.

Ce document débute ainsi par une présentation scientifique (chapitres 2 à 6) suivie de sa bibliographie, puis des publications les plus significatives, auxquelles j'ai participé, correspondant aux travaux de recherche présentés (chapitre 7). Il inclut enfin une version longue de mon curriculum vitæ présentant mes activités autour de la recherche et ma liste de publications (chapitre 8).

Introduction

2.1 Contexte général

La prolifération des supports d'affichage et l'intérêt que ce qu'ils diffusent suscite auprès de la population font de l'image un élément omniprésent dans notre vie quotidienne. Nous pouvons distinguer au moins deux types de supports. Les supports statiques : les panneaux, les murs aveugles, les abribus, les bus, les métros, les tramways, les murs des stations de métro, etc, sur lesquels on affiche une image fixe, et les supports dynamiques : les écrans de cinéma, les écrans de télévision, les écrans de téléphones et d'ordinateurs fixes ou portables, etc. Tous ces supports peuvent servir à la diffusion d'informations visuelles et les afficheurs dynamiques, agrémentés de périphériques d'entrée, permettent aussi l'interaction avec un univers numérique, ceci à des fins informatives, publicitaires, ludiques, professionnelles, créatives, etc.

La production de contenus visuels est ainsi un marché en pleine expansion, regroupant un grand nombre d'industries et de métiers et sollicitant un éventail de compétences et de domaines très larges et diversifiés. Nous pouvons par exemple citer des métiers comme acteur, monteur, maquilleur, technicien/ingénieur (vidéo, réseaux, électronique, etc), infographiste, développeur, chercheur, commercial, etc.

Vu de façon très globale, le contenu visuel peut être acquis du réel ou synthétisé. L'acquisition du réel permet une retranscription relativement fidèle de notre environnement et va nécessiter l'utilisation de matériels, plus ou moins onéreux et encombrants, suivant le type d'objet que l'on souhaite acquérir : un petit objet, une pièce archéologique, une statue, un corps humain, un arbre, une rue, une ville, la topographie d'une région, etc. Ces opérations vont bien souvent nécessiter les compétences de techniciens, de développeurs (quand la suite logiciel est à compléter) et d'infographistes pour arriver à une utilisation finale de/des l'objet(s) acquis dans une scène virtuelle. La création purement synthétique de contenu nécessite, quant à elle, moins de ressources. Avec une station de travail et un logiciel de modélisation géométrique moderne, un infographiste peut produire directement du contenu pour une scène virtuelle. Ici, les compétences de l'infographiste sont les garants, aussi bien de la qualité d'une reproduction du réel, que de la création d'un objet imaginé par une autre personne ou par l'infographiste lui-même.

Dans toutes les applications qui ont été brièvement présentées jusqu'ici, un facteur non négligeable est le facteur économique. La viabilité pratique ou industrielle d'un processus dépendra aussi de son coût et de la valeur qu'un utilisateur/acheteur lui donne. Ce coût ou cette valeur peut être un prix, un rapport coût/gain (en cas de production avec valeur ajoutée), mais aussi du temps, de l'investissement, suscités par un intérêt personnel par exemple (hobby, passion, etc).

Cette présentation introduit des éléments nécessaires à la bonne compréhension du contexte général dans lequel se situent les recherches présentées dans ce document : La génération synthétique de contenu virtuel.

2.1.1 Contexte spécifique et problématiques

Dans les notions de coût et de production intervient aussi la notion de temps. Ce que cela me “coûtera” de créer un objet synthétique, ou ce que cela coûtera à une entreprise dépendra, en partie, du temps que je vais passer ou qu’un infographiste va passer à modéliser l’objet à l’aide d’un, voire plusieurs logiciels. Ceci nous amène aux logiciels de modélisation d’objets tridimensionnels. Les logiciels professionnels comme Blender, Autodesk Maya/3D-Studio-Max/AutoCAD, Rhino, Dassault-Systèmes Catia, offrent, chacun dans leur domaine (animation et jeu vidéo pour les premiers, fabrication pour les suivants), des possibilités de création presque infinies. Dans ces logiciels, un grand nombre d’outils différents, plus ou moins spécifiques, sont proposés afin de répondre “au mieux” aux différents besoins exprimés par les utilisateurs. Au final, ceci conduit à des logiciels qui sont très puissants mais aussi très complexes, longs à prendre en main et peu intuitifs dans leur interaction avec l’utilisateur. Ainsi, pour être efficace, un infographiste professionnel doit développer une expertise spécifique, devant à la fois connaître les outils proposés par un logiciel, et maintenir une habitude et une agilité manuelle pour utiliser les différents menus et contrôles nécessaires à la création d’un objet. Il doit développer une aptitude à garder en tête ce qu’il veut réaliser tout en mettant en oeuvre le plus rapidement possible toutes les “meilleures” interactions avec le logiciel. Ceci demande au départ un long apprentissage et par la suite une pratique régulière. Il est donc fréquent que les infographistes aient des connaissances générales sur l’utilisation de plusieurs logiciels de modélisation, mais n’aient un rendement optimal que sur un seul. Pour un utilisateur non-expert, seules la patience et la persévérance lui permettront d’acquérir les compétences requises à une utilisation minimale d’un logiciel de modélisation. La modélisation d’un objet complexe se révélera souvent être un processus long et, quelquefois, difficile et laborieux.

En fait, la complexité d’un processus de modélisation est telle qu’il est très difficile de proposer un logiciel à la fois différent et au moins aussi, voire plus efficace que ceux qui sont déjà en place sur le marché professionnel. L’efficacité peut cependant avoir plusieurs visages : l’intuitivité et la simplicité de l’interaction avec l’utilisateur, le nombre réduit d’outils, la manipulation naturelle des outils, la vitesse de création pour un utilisateur expert, la simplicité d’accès pour un utilisateur non-expert, etc. Ainsi, un logiciel comme Pixologic ZBrush a réussi à prendre sa place en proposant un logiciel, qui, même si il est général, est plus spécialisé dans la génération de formes organiques (humanoïdes, animaux, monstres). Il offre ainsi une interaction plus rapide et naturelle avec l’objet en cours de création. Nous pouvons ainsi constater qu’il y a de la place pour des propositions innovantes et plus efficaces, mais nous pouvons aussi constater que, dès que le logiciel veut répondre aux multiples besoins exprimés par les utilisateurs, l’interface s’enrichit, le nombre d’outils aussi et le logiciel devient très complexe (figure 2.1).

Dans ce contexte, nous allons nous intéresser aux problèmes suivants :

- Comment permettre un accès plus simple à la modélisation d’objets tridimensionnels pour un utilisateur non-expert ?
- Comment démocratiser le développement de logiciels de modélisation d’objets tridimensionnels ?

Ces problèmes restent ouverts et sont des verrous à lever dans un futur proche afin de permettre le développement de l’expression et de la création avec un maximum de diversité à travers les

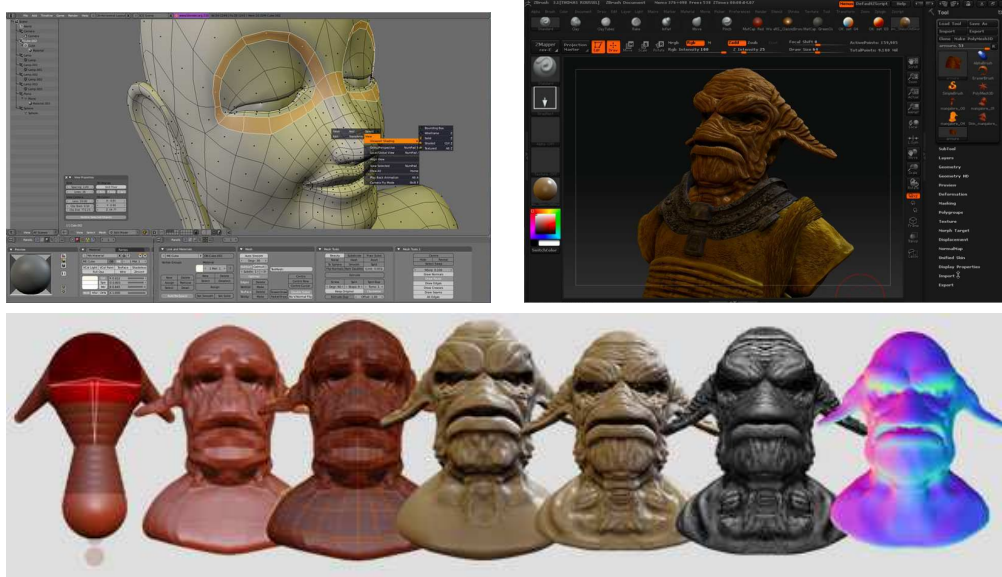


FIG. 2.1 – Illustration de l'interface de Blender (en haut à gauche) et Pixologic ZBrush (en haut à droite). En bas, illustration du principe de modélisation du logiciel Pixologic ZBrush. Sources : <http://doc.ubuntu-fr.org/blender> et <http://www.boulevard-creation.com/>

objets tridimensionnels synthétiques. Pour illustrer ce propos, nous noterons par exemple le besoin de logiciels grand public pour :

- la création d'objets tridimensionnels à imprimer en 3D,
- la génération d'environnements virtuels partagés,
- la génération d'environnements virtuels animables pour la production de minis films d'animation,
- l'utilisation de modèles tridimensionnels pour la communication, les loisirs, etc.

2.1.2 Contexte scientifique et organisation du document

Les problèmes que nous nous proposons d'aborder relèvent de plusieurs domaines scientifiques comme l'interface homme machine, l'interaction 3D, la visualisation temps réel ou interactive, la transmission via les réseaux... Nous nous concentrerons ici sur un domaine qui est au cœur de ces problématiques : les modèles de représentation d'objets tridimensionnels, leur amélioration et leur développement. Le modèle standard de représentation de surfaces pour l'affichage temps-réel est le maillage. Si très efficace pour une visualisation immédiate, cette représentation n'est pas adaptée à une manipulation simple et naturelle par l'utilisateur. Dès que l'objet se complexifie, le nombre de primitives (faces, sommets, arêtes) du maillage augmente et une manipulation directe devient vite trop laborieuse. Afin de créer un objet tridimensionnel même simple, il nous faut soit des outils haut niveau de manipulation de maillages, soit des modèles qui sont utilisés pour représenter et manipuler l'objet tridimensionnel, et qui sont rapidement maillés pour être visualisés. Dans cette catégorie, nous retrouvons les modèles surfaciques paramétriques [Far02] comme les B-splines, T-splines et les NURBS. Les surfaces de subdivision [WW02] sont un peu plus particulières mais rentrent aussi dans cette catégorie.

Les surfaces de subdivision ont en fait la particularité d'être directement définies à partir d'un maillage de variété deux quelconque. Ceci impose donc une contrainte de "bonne définition" du

maillage, mais cependant, cette contrainte garantira aussi par la suite une meilleure utilisabilité de l'objet dans les diverses applications qui pourraient l'utiliser (rendu, animation, nouvelles modifications par exemple). Elles ont aussi la propriété de définir des objets lisses à partir d'un processus de raffinement itératif du maillage initial, générant ainsi ce maillage à différentes résolutions de façon naturelle. Pour ces raisons, dès qu'elles ont pu être implantées de façon efficace, elles se sont imposées comme représentation incontournable dès lors que l'on ne fait pas de la construction (automobile, aéronautique, ...). En tant que standard, les premiers travaux que nous verrons portent sur ces surfaces. Nous verrons en quoi ces surfaces sont efficaces, où se situaient les verrous limitant leur utilisation, et des travaux que j'ai effectués sur quelques uns de ses points [SB02, BK04, BGSK05] (section 7.1). Nous verrons ensuite quelques curiosités [GBP05] (section 7.1.4). Enfin, nous discuterons de l'utilisation des surfaces de subdivision comme modèle de base pour adresser les problèmes présentés à la fin de la section précédente.

Ceci va nous amener à parler de la modélisation par esquisses. Il s'agit d'une technique qui utilise le dessin comme processus d'interaction entre l'utilisateur et le logiciel de modélisation. Sous sa version la plus simple, l'utilisateur dessine le contour de l'objet et le logiciel infère un objet tridimensionnel qui lui correspond "au mieux" [IMT99]. En dessinant ainsi plusieurs contours suivant des points de vue différents, l'utilisateur enrichit son modèle jusqu'à arriver à la forme souhaitée. Cette approche est conceptuellement très simple et surtout très naturelle, du moins, du point de vue de l'utilisateur. Dans ce domaine, j'ai participé à l'encadrement de plusieurs étudiants. Nous verrons où se situent les difficultés de génération d'une surface 3D à partir d'un contour 2D et quels sont les concepts importants que nous avons introduits [AGB04, ABCG05, BPCB08] (section 7.2). Nous mettrons ainsi en évidence les besoins que nous avons distingués d'un point de vue modèle de représentation de surface. Nous concluons avec une discussion autour des modèles et de leur applicabilité à la modélisation par esquisses. Ceci nous permettra de mettre en évidence certains avantages des représentations volumiques [Blo97b, CBC⁺01, OF02, GG07]. Ces représentations sont en quelque sorte complémentaires des représentations surfaciques, les avantages des unes étant bien souvent les désavantages des autres et vice versa. Ainsi, le lien intime entre les surfaces paramétriques et les maillages n'existe pas entre les modèles volumiques et maillages et comme nous le verrons, ceci est compensé par d'autres bonnes propriétés notamment liées à la reconstruction de surfaces et à leur composition.

Ainsi, nous arrivons à l'étude des modèles volumiques. Nous nous concentrons sur les modèles volumiques fonctionnels. Ils sont compacts et bien adaptés à la reconstruction de surfaces de révolution à partir de squelettes. Ils permettent par exemple une reconstruction naturelle d'une surface 3D à partir d'un contour 2D, donnée d'entrée d'un système par esquisse. Malgré des avantages importants, ces modèles présentent des désavantages qui les limitent dans leur utilisabilité. Cependant, très peu de recherche fondamentale a été effectuée dans ces dix dernières années, ce qui fait que ces limitations ne sont pas un fait en soi, mais plutôt des obstacles suffisamment importants pour décourager les chercheurs il y a de ça un peu moins de dix ans. Nous ferons le point sur ces obstacles et présenterons mes derniers travaux effectués, dans cette période, sur les opérateurs de composition pour les modèles volumiques [BDS⁺03, BWG04] (section 7.3). De nos jours, les ressources de stockage et de calcul sont bien plus importantes et nous verrons quelles nouvelles solutions nous avons présentées [BBCW10, OGP10, GBC⁺11] (section 7.3). Nous verrons ensuite les projets autour de leur extension, et quelles propositions nous pouvons encore envisager pour lever les limitations des modèles volumiques fonctionnels et ré-évaluer leur utilisabilité dans ce nouveau contexte scientifique et matériel.

Surfaces de subdivision

3.1 Introduction

Les surfaces de subdivision peuvent être vues comme un processus de raffinement itératif d'un maillage de variété deux quelconque convergeant vers une surface limite [WW02], potentiellement lisse, de continuité C^n sauf en quelques points isolés (appelés points singuliers ou irréguliers) où elle est naturellement C^1 , et où elle peut être maintenant reconstruite de continuité G^2 au mieux [Lev06, LS08]. L'équation de la surface limite est aussi connue sauf au niveau des points irréguliers [DS78, Sta98] pour lesquels on connaît la position limite [BK03].

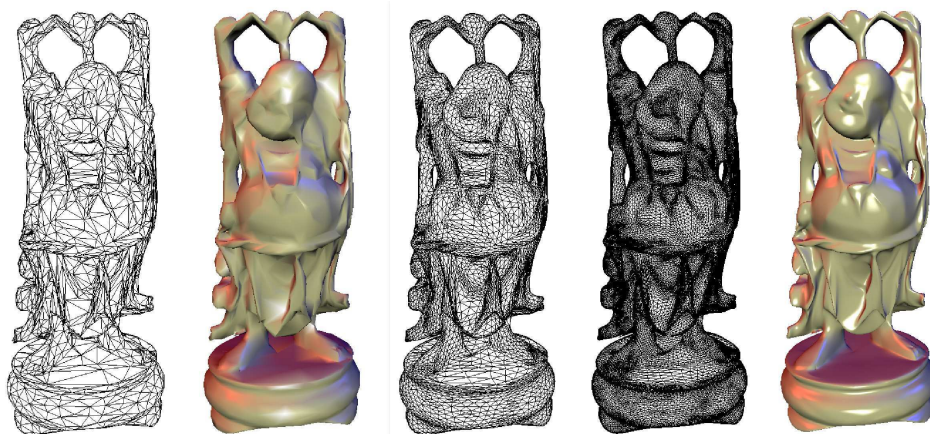


FIG. 3.1 – Illustration de la subdivision d'un maillage et de son effet sur la topologie, la géométrie et le rendu de ce maillage.

Depuis leur introduction dans le domaine de la modélisation géométrique en 1978 [CC78, DS78], ces surfaces ont tout d'abord suscité un faible engouement de la part des chercheurs et des industriels, et pour cause, le matériel informatique disponible à cette époque ne permettait pas de stocker les maillages dont la taille croît rapidement de pas de subdivision en pas de subdivision : le nombre de faces est multiplié par 4 à chaque pas pour un schéma di-adique comme les schémas Catmull-Clark [CC78] ou Loop [Loo87]. Il faudra attendre la fin des années 80 [Loo87, DLG90] puis le milieu des années 90 [HDDH94, Rei95, Kob96a, Kob96b, ZSS96, ZSS97] pour que les chercheurs anticipent l'explosion du domaine portée par Pixar et "Geri's Game" en 1997. Vont suivre un grand nombre d'avancées au niveau des surfaces de subdivision elles-mêmes [DIS03, KPR04] et au niveau de leur utilisation, en modélisation géométrique [BLZ00,

GVSS00, LLS01, BMBZ02, GKSS02] ou en rendu par exemple [Hop97, Hop98, ALSS03], jusqu'en 2005 environ. Depuis, la plupart des dernières avancées sur les surfaces de subdivision sont soit incrémentales [LS08], soit basées sur un fondement théorique plus complexe et relevant de plus en plus des mathématiques appliquées [Lev06, CADs09].

Les travaux que j'ai effectués se situent sur la période 2001-2004, c'est à dire après que des solutions efficaces aient été proposées pour le contrôle des normales, des arêtes, des bords, la modélisation multi-résolution, le copier-coller. Ils ont été initiés par Malcolm A. Sabin et Neil Dodgson puis poursuivis sous l'encadrement de Leif Kobbelt. Ils portaient sur l'étude et l'amélioration de la qualité, à proximité des points singuliers, des maillages au cours de la subdivision et de la surface limite.

3.2 Etude des comportements indésirables dans la subdivision stationnaire

Pour ces travaux, nous nous sommes cantonnés aux schémas de subdivision linéaires et stationnaires : les règles de subdivision sont des combinaisons affines des points de contrôle et sont identiques pour chaque pas de subdivision. De plus, en priorité, nous nous sommes intéressés aux schémas reproduisant les surfaces Box-splines dans les zones régulières du maillage de contrôle pour la connaissance de l'équation de la surface limite et toutes les bonnes propriétés des surfaces Box-Spline : support compact, "diminishing varying property", enveloppe convexe, etc. Ces schémas sont ainsi les plus communément utilisés [Loo87, CC78]. Cependant, même si leur utilisation est à l'origine de solutions de modélisation efficaces encore standard de nos jours, comme on peut le voir dans le logiciel Pixologic ZBrush, ces surfaces ont aussi des comportements indésirables en situation de modélisation. Dans un premier temps, j'ai participé aux travaux de Malcolm Sabin sur la mise en évidence et la caractérisation de certains des comportements indésirables [SB02] (voir la publication section 7.1.1). Ces travaux sont exploratoires et mettent en évidence des problèmes qui ont guidé les travaux qui ont suivi.

Considérons le maillage comme le polygone de contrôle de la surface limite définie par la subdivision. Les pas de subdivision intermédiaires génèrent des maillages de plus en plus denses convergeant vers la surface limite. En quelques itérations, les sommets des maillages sont très proches de la surface limite et ils peuvent ainsi être utilisés comme maillage de visualisation de la surface. Nous nous intéresserons donc aussi au comportement de ces maillages intermédiaires. Les propriétés de la surface limite ainsi que celles des maillages intermédiaires sont étroitement liées à la forme et à la connectivité du maillage initial. Dans ces travaux, nous mettons en évidence trois comportements des maillages et de la surface qui sont indésirables et qu'un utilisateur ne peut pas éviter en déplaçant les sommets du maillage de contrôle.

3.2.1 Distorsions dans la taille des faces du maillage subdivisé

Le premier affecte la taille des faces du maillage autour des sommets irréguliers (figures 3.2(a) et (b)). La subdivision n'insère que des sommets réguliers, ainsi, pas de subdivision après pas de subdivision les sommets irréguliers sont isolés au milieu d'une "mer" de sommets réguliers dont la taille des faces adjacentes devient de plus en plus homogène. Seuls restent isolés les sommets irréguliers du polygone de contrôle d'origine. Autour de ces sommets, des distorsions dans la taille des faces sont présentes : proportionnellement à la taille des faces des zones régulières, ces faces sont plus grandes autour des sommets de forte valence et plus petites autour des sommets de faible valence.

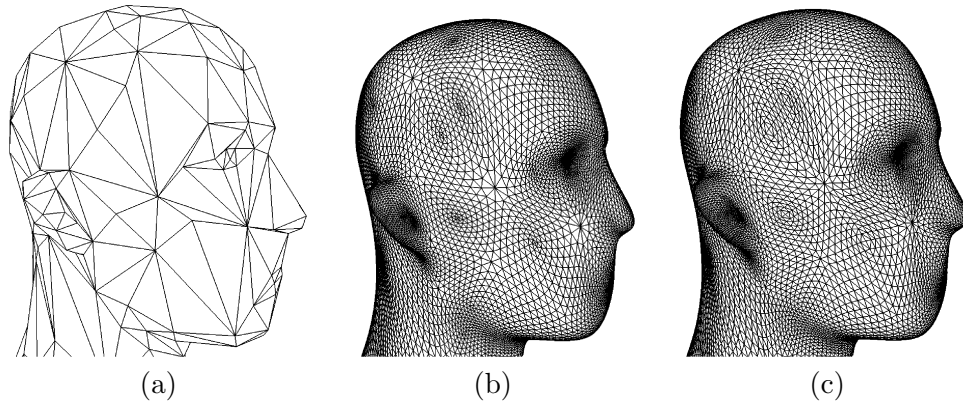


FIG. 3.2 – (a) Un maillage de contrôle initial, (b) le nouveau maillage après quatre pas de subdivision de Loop. On remarque les distortions dans la taille des faces à proximité des sommets irréguliers. (c) Une version modifiée du schéma de Loop diminuant ces distortions.

Ces différences de taille posent des problèmes de qualité de représentation de surface en modélisation, notamment en modélisation multi-résolution, quand on cherche à travailler sur une certaine taille de faces, ou en rendu, quand on souhaite effectuer un rendu où toutes les faces se projettent sur un pixel de l'image. Un exemple typique est une forte valence au pic d'une bosse de forte courbure. Après subdivision, on obtiendra une pointe au lieu d'une bosse bien lisse.

La décomposition en valeurs propres et vecteurs propres de la matrice de subdivision [DS78, BS88, Rei95, BGSK05] est un outil standard et efficace d'analyse des propriétés du maillage et de la surface limite. A partir des points des anneaux topologiques réguliers pris autour d'un sommet central irrégulier (ou régulier) P^i , la matrice de subdivision M est une matrice carrée qui contient les coefficients du calcul des mêmes anneaux topologiques après un pas de subdivision P^{i+1} avec $P^{i+1} = MP^i$. A la limite, $P^\infty = M^\infty P^i$ et les vecteurs propres de M définissent la position des sommets autour du point irrégulier, tandis que ses valeurs propres donnent le coefficient de diminution de ces vecteurs à chaque pas de subdivision. Ainsi une valeur propre que nous nommerons λ_1 , valant $\lambda_1 = 1/2$, correspondant aux vecteurs propres définissant la position des points dans le plan tangent à la surface, signifie qu'à la limite, après un pas de subdivision, les points sont deux fois plus proches du sommet irrégulier. Quand, à chaque pas de subdivision, on multiplie par quatre le nombre de sommets sur une surface (schémas di-adiques), on multiplie par deux le nombre de sommets le long d'une direction de ce maillage et ainsi, on double la densité des points sur les arêtes quittant le point irrégulier. Ceci permet d'avoir l'intuition de la raison pour laquelle $\lambda_1 = 1/2$ permet une répartition bien régulière des points du maillage. La figure 3.2(c) a été produite avec un schéma de subdivision ayant $\lambda_1 = 1/2$ pour les règles s'appliquant à tous les sommets irréguliers du maillage. En complément de ces informations, les valeurs propres permettent aussi de définir des propriétés nécessaires de convergence et de continuité C^1 , C^2 .

Dans la pratique, les schémas Box-Spline classiques [CC78, DS78, Loo87] n'ont cette valeur propre $\lambda_1 = 1/2$ que dans le cas régulier (valence quatre pour les maillages quadrilatéraux et six pour les maillages triangulaires). Pour les valences supérieures au cas régulier, on constate que $\lambda_1 > 1/2$ et pour les valences inférieures, on constate que $\lambda_1 < 1/2$ ce qui explique les distortions des faces autour des sommets irréguliers.

3.2.2 Précision linéaire

Le deuxième concerne la capacité à reproduire une fonction linéaire quand des sommets du maillage sont alignés le long d’une fonction linéaire (figure 3.3). Sur les surfaces Box-Spline, ce comportement est prévisible. La surface Box-Spline, ou la surface limite d’un schéma de subdivision convergeant vers une surface Box-spline, aura la précision linéaire dans toutes les directions du maillage dans lesquelles la fonction de base a une direction de convolution [dBHR94, PS04]. Pour les autres schémas de subdivision, la surface limite aura la précision linéaire dans toutes les directions où le masque de subdivision, écrit avec la transformée en z , est factorisable par $(1 + z)$ [Kob00, Dyn02, Bar07].

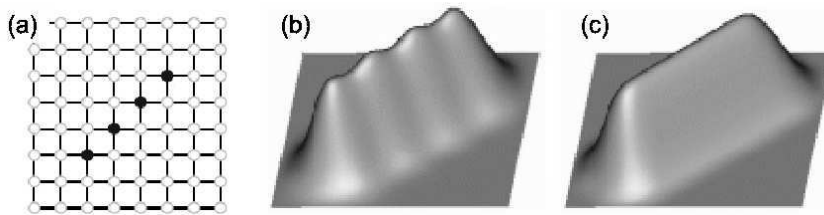


FIG. 3.3 – (a) Le maillage d’origine avec tous les sommets à l’altitude 0 sauf les sommets noirs qui sont à l’altitude 1. (b) Surface limite obtenue avec un schéma de subdivision qui n’a pas la précision linéaire dans la direction des sommets noirs : la surface limite ne reconstruit pas une fonction linéaire et oscille (ici le schéma de Catmull-Clark). (c) Surface limite obtenue avec un schéma de subdivision qui a la précision linéaire dans la direction des sommets noirs : la surface limite reconstruit une fonction linéaire (ici le schéma présenté par Peters et Shiue [PS04]). Image issue de [PS04].

Dans la pratique, les surfaces en produit tensoriel, donc définies par des maillages quadrilatéraux, ont la précision linéaire uniquement dans les directions du maillage portées par les arêtes. Pour le schéma triangulaire de Loop [Loo87], la précision linéaire est uniquement le long des arêtes du maillage. Par exemple un schéma comme le $\sqrt{3}$ de Kobbelt [Kob00] n’a pas la précision linéaire.

3.2.3 Oscillations radiales

Le troisième comportement que nous abordons vient de l’interaction entre sommets irréguliers quand ils sont voisins. La figure 3.4 illustre les oscillations radiales que l’on obtient lorsque l’on subdivise une forme simple constituée uniquement de sommets irréguliers avec un anneau de sommets de faible valence. Dans ce cas, la “diminishing varying property” n’est plus respectée et des oscillations très visibles apparaissent, et ce, dès le premier pas de subdivision.

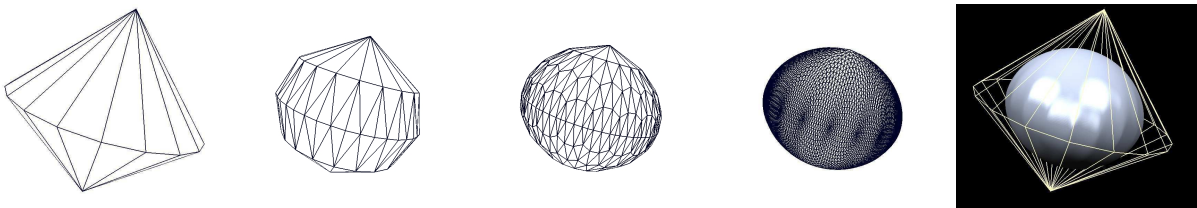


FIG. 3.4 – Subdivisions successives du maillage de gauche et surface limite à droite.

Ici, le problème vient des tout premiers pas de subdivision, voire dans certains cas, uniquement du premier pas. En effet, les règles de subdivision, en un sommet irrégulier, sont calculées en s'appuyant sur la décomposition en valeurs singulières de la matrice de subdivision. Elles sont donc calculées en faisant l'hypothèse que le sommet irrégulier est entouré d'au moins un anneau de sommets réguliers, ce qui n'est pas le cas ici lors du premier pas de subdivision. Si l'on arrive à proposer un processus de subdivision spécifique aux tout premiers pas, permettant de conserver des bonnes propriétés sur les maillages, les sommets irréguliers seront ensuite suffisamment isolés et les règles classiques ne généreront plus ce type d'oscillations.

3.3 Amélioration des schémas de subdivision autour des points irréguliers

Comme nous pouvons le constater, deux des effets indésirables dont nous venons de parler se situent au niveau des sommets irréguliers. A ce moment là, un autre problème, situé au niveau des sommets irréguliers, était aussi à considérer. Les schémas de subdivision les plus utilisés en modélisation géométrique pour le loisir ou la communication (films, jeux, ...) sont les schémas de Box-Spline, ayant des surfaces limites de continuité C^2 ([CC78, Loo87]), car c'est la continuité minimale permettant de produire des surface lisses ayant aussi des reflets lisses. Hors, au niveau des sommets irréguliers, la surface limite n'est que de continuité C^1 et la courbure peut diverger. L'impact sur le rendu des surfaces spéculaires est très perceptible, comme on peut le voir avec le schéma de Loop sur la figure 3.5(f).

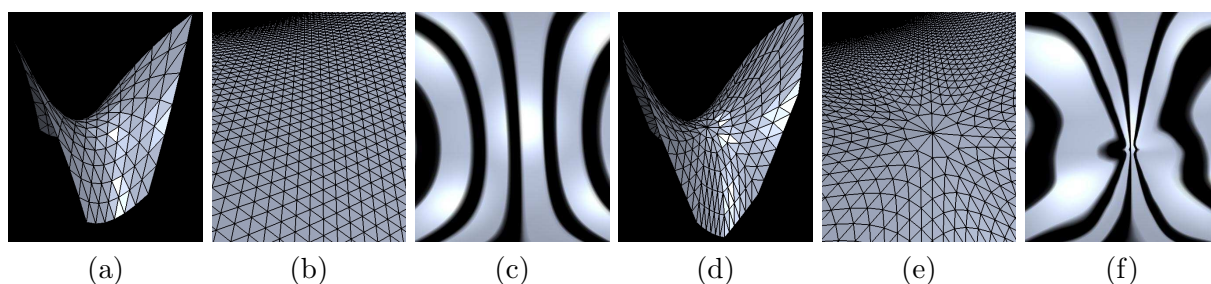


FIG. 3.5 – Moitié de gauche, illustration de la subdivision d'un maillage triangulaire régulier : (a) le maillage d'origine, (b) le maillage subdivisé un grand nombre de fois avec la subdivision de Loop (continuité C^2) et (c) le rendu de lignes de réflexion sur le maillage. Moitié de droite, la même surface de départ avec un sommet irrégulier au centre (valence douze) : (d) le maillage d'origine, (e) le maillage subdivisé un grand nombre de fois avec la subdivision de Loop (continuité C^1 et courbure divergente au sommet irrégulier) et (f) le rendu de lignes de réflexion sur le maillage. La dégradation du comportement de la courbure apparaît clairement dans les reflets autour du sommet irrégulier.

Dans les travaux présentés dans [BK04] (section 7.1.3), nous cherchons à améliorer le comportement des schémas de subdivision à proximité des sommets irréguliers en essayant de conserver l'homogénéité du processus de subdivision. Nous prenons en compte les irrégularités de taille de faces et la divergence/variation de la courbure. Pour agir sur ces irrégularités, nous utilisons la décomposition en valeurs singulières. Cette idée n'est pas nouvelle et elle a été utilisée par Peters, Reif et Prautzsch [PR98, Pra98] pour produire par exemple une courbure nulle au niveau du sommet irrégulier. On obtient ainsi une surface localement C^∞ car convergant vers un plan, mais produisant une surface plate. Le principe est de modifier directement les valeurs propres

d'intérêt et de reconstruire la matrice de subdivision correspondante qui donne directement les nouvelles règles de subdivision. Si l'on souhaite introduire des valeurs permettant de réduire les distorsions des faces et de borner la courbure plutôt que de l'annuler, l'intégralité des valeurs de la matrice de subdivision est modifiée. Ainsi, les nouvelles règles ne prennent plus en compte le comportement des règles régulières à proximité du sommet singulier. Le résultat est une surface qui a de bonnes propriétés au niveau des sommets irréguliers, mais qui n'a pas de lien particulier avec la surface régulière reconstruite autour. Ainsi, la surface exhibe de fortes oscillations incontrôlables dans la zone où les règles régulières viennent recouvrir les sommets introduits par les règles spécifiques au sommet irrégulier, pas de subdivision après pas de subdivision.

Notre contribution est, dans un premier temps, la reformulation du calcul des valeurs singulières directement en fonction des coefficients des règles de subdivision pour le sommet irrégulier (qui sont nos inconnues), appelées règles irrégulières, et des règles régulières qui sont fixées par le schéma. Puis nous proposons un processus de minimisation calculant les coefficients des règles irrégulières de façon à ce que les valeurs singulières se rapprochent au plus des bonnes valeurs ($\lambda_1 = 1/2$ par exemple). Nous pouvons ainsi choisir le nombre de coefficients que nous souhaitons utiliser pour les règles irrégulières et les valeurs singulières que nous souhaitons approcher. Nous pouvons aussi choisir de privilégier l'amélioration d'un problème plutôt qu'un autre. La figure 3.2(c) montre une subdivision avec le schéma de Loop pour lequel nous avons généré des règles avec $\lambda_1 = 1/2$ en tout point irrégulier du maillage et en faisant au mieux au niveau de la courbure. Nous avons aussi proposé des règles essayant d'améliorer les comportements de la courbure. C'est probablement le plus difficile, car le contrôle des vecteurs propres est imprécis dans la minimisation, et la formulation des contraintes est sensible. Il est très facile d'avoir l'impression que le schéma sera performant en terme de valeurs singulières et d'obtenir le résultat présenté figure 3.6(b) lorsqu'on l'applique. Au final, en nous basant sur les travaux de Gérot [GBDS05], nous sommes arrivés à améliorer localement la courbure en évitant qu'elle diverge et en évitant l'introduction de fortes oscillations dans la surface tout en diminuant les distorsions de tailles des faces (figure 3.6(d), (e) et (f)). Nous notons cependant que, dans certaines configurations de surfaces (notamment en forme de selle), il reste des oscillations de courbure autour du sommet irrégulier que nous ne sommes pas arrivés à supprimer (figure 3.6(c)).

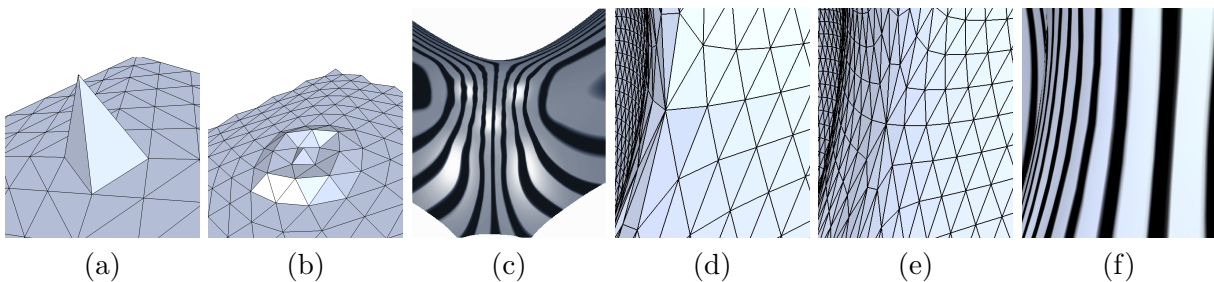


FIG. 3.6 – Illustrations d'améliorations du comportement de la courbure. (a) Un maillage d'origine et (b) une solution améliorant le comportement de la courbure de point de vue des valeurs singulières. On remarque la création d'oscillations indésirables dès le premier pas de subdivision. (c) La subdivision du maillage de la figure 3.5(d) avec des règles améliorant la courbure. La surface est bien convexe, mais il reste des oscillations de courbure. (d) Un nouveau maillage d'origine, (e) et (f) résultats obtenus avec les mêmes règles irrégulières qu'en (c).

Notre approche automatisée nous a aussi permis de proposer des règles irrégulières “acceptables” pour rendre utilisable un schéma de subdivision Box-Spline de continuité C^3 , ayant des règles de subdivision de petit support et réalisant une subdivision topologique en $\sqrt{3}$. Les règles

régulières avaient été présentées par Ron [Ron00] en 2000 et aucune règle irrégulière n'avait encore été proposée. Par la suite d'autres travaux ont traité de ce sujet [KPR04, ADS06, ADS11a, ADS11b].

3.4 Autour de ces travaux, quelques curiosités

Au cours de la réalisation de ces travaux, je me suis rendu compte qu'il y avait beaucoup de connaissances détenues par un petit nombre de personnes et très difficiles, voire impossibles à trouver dans la littérature. Ceci est à mon avis un problème important car tout chercheur, qui n'a pas accès au savoir des bonnes personnes, ne peut que très difficilement produire une recherche correcte d'un point de vue théorique, et risque d'arriver à des résultats erronés sans pouvoir l'expliquer. Par exemple, j'ai rédigé l'article présenté section 7.1.2 [BGSK05] suite à plusieurs relectures de soumissions très intéressantes présentant de nouveaux schémas de subdivision, mais pour lesquelles l'analyse pour la création des règles irrégulières était incorrecte. La notion de fréquence associée aux valeurs et vecteurs propres n'était pas prise en compte, ce qui invalidait les conclusions tirées de leurs valeurs.

Une autre connaissance, que l'on m'a enseignée comme passe-temps, est le calcul des règles de subdivision régulière pour les schémas Box-Spline, qu'ils soient sur des maillages quadrilatéraux ou triangulaires, en quelques minutes, avec juste un crayon et une feuille de papier. C'est ludique pour quiconque s'amuse des mathématiques, simple et on choisit au départ quelles seront les directions de maillage dans lesquelles notre schéma aura la précision linéaire. J'ai présenté cette méthode dans deux communications orales invitées, dans un chapitre de livre en français [Bar07] et dans certains de mes cours. Il faudra attendre 2009 pour que cette technique soit enfin présentée dans une conférence internationale [DAC09]. Un petit outil comme celui-ci peut paraître secondaire et reste finalement à l'état de curiosité. Pourtant, un schéma Box-Spline sur les maillages quadrilatéraux comme celui présenté par Peters et Shiue [PS04], a une surface limite de continuité C^2 , des règles de subdivision un peu plus petites que le schéma de Catmull-Clark [CC78], et la précision linéaire dans quatre directions (figure 3.7(b)) au lieu de deux pour Catmull-Clark (figure 3.7(a)). Avec notre petit outil, les règles régulières peuvent être calculées en deux-trois minutes, les règles irrégulières ne demandent aucune analyse spécifique autre que celle présentée dans [CC78]. Au final, il aura tout de même fallu attendre 2004 pour voir apparaître ce schéma car il est pur Box-Spline, et non résultant du produit tensoriel de courbes B-Splines. Il existe exactement le même type de schéma améliorant celui de Doo-Sabin [DS78].

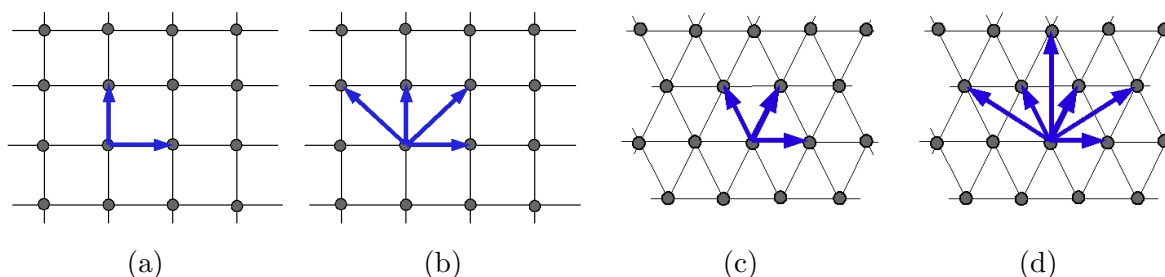


FIG. 3.7 – Directions de précisions linéaires pour les schémas de (a) Catmull-Clark et Doo-Sabin, (b) Peters et Shiue, (c) Loop. (d) Les directions de précisions linéaires, dont certaines orthogonales, pour une surface Box-Spline de continuité C^3 .

Nous pouvons ainsi fouiller un petit peu au milieu de ce que nous offrent les Box-Splines

et y trouver une Box-Spline de continuité C^3 , sur les maillages triangulaires, ayant la précision linéaire dans deux directions orthogonales (figure 3.7(d)), comme les schémas définis sur des maillages quadrilatéraux. Ce schéma permet ainsi de représenter un cylindre avec un maillage triangulaire régulier tout en ayant une surface limite qui n'oscille pas dans la direction axiale, comme le ferait le schéma de Loop ou tout autre schéma classique utilisé sur les maillages triangulaires (figure 3.7(c), figure 3.8). On donne ainsi aux maillages triangulaires les bonnes propriétés des maillages quadrilatéraux pour représenter des surfaces ayant des directions principales orthogonales ou presque.

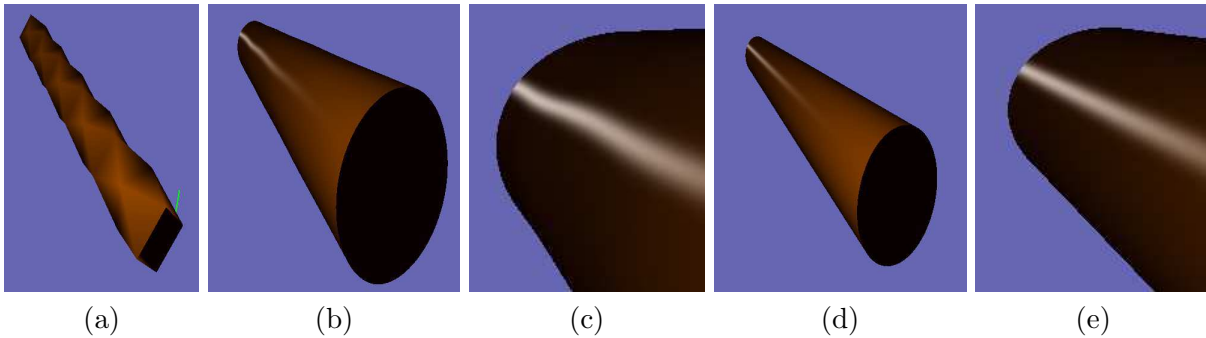


FIG. 3.8 – (a) Un cylindre représenté par un maillage triangulaire régulier. (b,c) Sa subdivision avec le schéma de Loop : on note les oscillations axiales de la surface. (d,e) Sa subdivision avec notre schéma à six directions : grâce aux directions orthogonales, la surface limite n'a pas d'oscillation axiale.

3.5 Subdivision d'un nuage de points non organisé

Au début des années 2000, la mode est aux points. Quand j'ai été recruté à l'IRIT, Gaël Guennebaud, un étudiant de Mathias Paulin en début de deuxième année de thèse finissait des travaux sur le rendu de nuages de points de tailles massives [GBP04]. J'ai pu co-encadrer Gaël et ainsi participer à ses projets de recherche. Gaël a ensuite enchaîné sur la subdivision de nuages de points, une approche naturelle permettant d'éviter la visualisation de gros disques quand on zoome sur un nuage de points, et permettant aussi la visualisation d'une surface lisse à partir d'un nuage de points de faible densité voire même de boucher des trous dans une géométrie. L'idée principale de ces travaux est d'arriver à converger vers une surface visuellement lisse sans jamais reconstruire explicitement la topologie du nuage de points, afin de garder la souplesse des points et de ne pas avoir à passer par une face de reconstruction globale du nuage de points en maillage. La méthode de raffinement est locale et dépendante de la densité locale des points. Dans [GBP05] (section 7.1.4), nous proposons de reconstruire à la volée un anneau de faces triangulaires autour de chaque point et d'insérer un point, dans les zones de densité trop faible uniquement, au centre des triangles reconstruits avec un carreau triangulaire bi-cubique interpolant les trois sommets et leurs normales.

Le résultat est un schéma interpolant générant des surfaces de meilleure qualité que le schéma interpolant Butterfly sur les maillages triangulaires [DLG90, ZSS96], car il profite de l'interpolation des normales des sommets. La technique est suffisamment efficace pour faire du raffinement en temps réel pendant le rendu d'un objet. Une originalité de cette technique est de ne pas pouvoir être portée sur les maillages triangulaires. On démontre simplement en prenant les sommets dans un plan que le schéma diverge. Le simple fait de ne pas stocker la topologie, de reconstruire

un voisinage à la volée à chaque pas de raffinement et de n'insérer des points qu'en fonction de la densité locale du nuage de points suffit à éviter de diverger et permet de converger vers une surface de continuité C^1 au moins. Le phénomène de distorsion des faces autour des sommets irréguliers se retrouve dans cette subdivision mais ici encore, travailler sans topologie explicite et en fonction de la densité locale gère naturellement le problème. Le raffinement local ou adaptatif se met aussi en oeuvre sans difficulté, les principaux problèmes dans ces cas venant de la gestion de la topologie ou du niveau de subdivision pour les schémas approximatifs.

L'intérêt applicatif de ces travaux dépend de l'utilisation des points comme représentation géométrique pour le rendu. D'un point de vue théorique, ces travaux mettent en évidence que l'absence de topologie du nuage de points permet une gestion très souple et efficace de la densité du raffinement. D'un autre côté, il n'y a pas d'auto-similarité entre les pas de subdivision successifs, ce qui complique l'analyse des propriétés de la surface limite et la mise en oeuvre du schéma : une topologie locale doit être reconstruite à la volée avec toutes les gestions d'incohérences que cela comporte.

3.6 Positionnement dans notre contexte et conclusion

Utilisés dans un système multi-résolution, les schémas de subdivision comme Catmull-Clark et Loop sont des outils très performants pour la modélisation de formes complexes. L'implantation des surfaces de subdivision seules ne pose pas de grosses difficultés. Il est déjà plus délicat de développer un système multirésolution, mais, avec des efforts, on finit par obtenir un programme stable. Dès lors que l'on veut permettre les changements de topologie dans la surface et/ou proposer des opérateurs de composition booléenne (union, intersection, différences), il faut faire face à des difficultés algorithmiques plus importantes, et, surtout, à des instabilités numériques dans la mise en oeuvre qui nécessitent des compétences techniques très fines et beaucoup de temps [JS09]. Les solutions aux problèmes que nous venons de voir reposent sur la reconstruction de carreaux paramétriques de degré élevé au niveau du sommet irrégulier [Pet02, Lev06, LS08]. Ces solutions ne sont cependant pas idéales car elles nécessitent la présence de plusieurs anneaux topologiques de maillage régulier autour du sommet irrégulier. Donc, bien souvent, il faut avoir effectué quelques pas de subdivision sur le maillage, ce qui va avoir introduit des irrégularités sur la surface qui seront conservées par le traitement local du sommet irrégulier. Ceci implique aussi de développer, en supplément de la subdivision, les carreaux paramétriques et la gestion de leur continuité pour mettre en oeuvre ces solutions.



FIG. 3.9 – Images de Torolf Sauer mann, dont les objets ont été générés avec le logiciel Top-Mod [Top], illustrant la modélisation topologique à base de surfaces de subdivision.

Il existe des pistes pour essayer de palier ces difficultés en utilisant la surface de subdivi-

sion uniquement comme un outil de raffinement et de représentation de surface lisse. L'idée est de se concentrer sur une implantation robuste d'opérateurs topologiques sur les maillages [AC00, ACS00, KCB09]. Un ensemble de travaux donnant des résultats impressionnants a été effectué par Akleman et Chen [ASC03, AC06]. Ils permettent de créer des objets à la topologie extrêmement complexe (figure 3.9) dans un environnement de modélisation très haut niveau et relativement accessible. Cependant, après plusieurs années de développement en laboratoire, le logiciel intégrant ces travaux (TopMod) reste très instable. Même si elles sont très séduisantes, je ne connais pas, actuellement, de travaux suffisamment matures pour permettre de construire des outils stables et efficaces sur ce type d'approche.



FIG. 3.10 – Image de Bein et al. [BHSF09] illustrant la modélisation par esquisses basée sur des surfaces de subdivision.

Récemment, Bein et al. [BHSF09] a proposé un système de modélisation par esquisses basé sur une représentation hybride maillage/surface de subdivision (Catmull-Clark). Pour éviter les phases sensibles de reconstruction de maillages 3D à partir de contours 2D et l'implantation des opérateurs de composition booléenne (au moins l'union), les auteurs proposent une interaction moins intuitive mais plus simple à mettre en œuvre. L'utilisateur peut dessiner des courbes de forme quelconque qui seront ensuite approximées par des B-Spline avec un faible nombre de points de contrôle. L'objet est alors construit par séquences d'extrusions et de déformations essentiellement (figure 3.10). Une idée qui me semble importante, que l'on retrouve dans les travaux d'Akleman, est l'utilisation, autant que possible, de maillages quadrilatéraux composés d'un faible nombre de faces comme support de représentation.

Au final, la puissance des surfaces de subdivision est utilisée essentiellement dans les logiciels professionnels. Il est attractif d'essayer de les utiliser dans des systèmes plus simples et plus accessibles et ceci reste un problème ouvert, à mon sens, très intéressant à aborder.

Modélisation par esquisses

4.1 Introduction

Il est très difficile de proposer à l'utilisateur un système à la fois simple, intuitif, efficace et adapté aux utilisateurs non-experts, pour modéliser des objets tridimensionnels complexes. Parmi les travaux précurseurs, on retrouve les travaux de Pugh [Pug92] ainsi que ceux de Akeo et al. [AHKS94] qui introduisent l'esquisse 2D et l'interprétation du trait comme entrée pour la reconstruction de données 3D. Vient ensuite le système présenté par Zeleznik et al. [ZHH96] qui est basé sur la reconnaissance de gestes, et un peu plus tard, "Teddy", un système de modélisation d'objets de forme libre par esquisses présenté par Igarashi et al. [IMT99]. L'intuitivité de l'interaction et le temps d'apprentissage de quelques minutes de ce système ont tout de suite séduit une grande partie de la communauté. Vont suivre des travaux reprenant ou développant ces idées [KHR02, ZS03, ONNI03, TZF04, AGB04], et en 2004, la modélisation par esquisses devient un thème de recherche à part entière avec son premier "workshop", SBIM, qui a encore lieu de nos jours (conjointement avec NPAR).

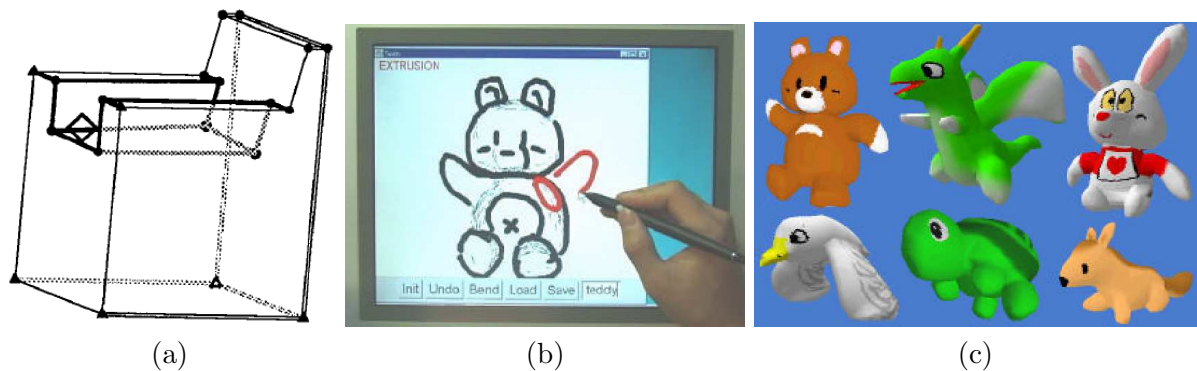


FIG. 4.1 – (a) Image de Pugh [Pug92] illustrant le type d'objets générés par son système. (b,c) Images d'Igarashi et al. [IMT99] illustrant (b) l'interaction et (c) les résultats obtenus avec "Teddy".

La modélisation par esquisses est contextuelle, ce qui permet d'interpréter avec le minimum d'ambiguïté les dessins 2D de l'utilisateur. Ainsi, un système ne reconstruira pas nécessairement une esquisse de la même façon s'il est dédié à la modélisation architecturale, à la CAO ou à la modélisation de formes libres par exemple. Dans ce chapitre, nous nous positionnons dans la modélisation de formes libres. Les recherches de modélisation par esquisses ont été initiées dans

l'équipe VORTEX par Véronique Gaildrat quand elle a commencé à encadrer Anca Alexe en Master deuxième année en 2001. Son idée était de reprendre les concepts d'interaction présentés par Igarashi et al. [IMT99] dans "Teddy" en enrichissant le système avec des opérateurs CSG (notamment la différence) avec une plus grande simplicité de mise en œuvre. Quand j'ai été recruté à l'IRIT en 2003, Véronique Gaildrat m'a tout de suite permis de co-encadrer Anca Alexe (qui commençait sa deuxième année de thèse) et je me suis donc intéressé à la modélisation par esquisses. Ces premiers travaux ont été menés à la suite de la présentation des travaux de Karpenko et al. [KHR02] et en parallèle des travaux de Tai et al. [TZF04].

4.2 Premiers travaux et reproduction de vieux problèmes

Une idée de base amenant à la simplicité de "Teddy" est la reconstruction de formes tridimensionnelles de révolution à partir de contours 2D fermés, dessinés par l'utilisateur. D'un point de vue technique, le dessin est considéré comme un ensemble de points qui sont tout d'abord triangulés en 2D à l'aide d'une triangulation de Delaunay contrainte à l'intérieur du contour [Aur91]. Une approximation de l'axe médian du contour est créée à partir de cette triangulation [Pra97]. Anca Alexe et Véronique Gaildrat proposaient alors d'utiliser une reconstruction de surface tridimensionnelle s'appuyant sur l'insertion de primitives implicites sphériques à support compact (Soft-Objects [WMW86]) centrées sur des points placés sur l'axe médian. La surface implicite finale était une iso-surface de la somme des fonctions potentiel des primitives sphériques [Bli82]. Les justifications de ce choix sont la simplicité d'implantation des surfaces implicites, si on les compare aux maillages, leur capacité naturelle à produire des surfaces de révolution et la mise en œuvre robuste des opérateurs de composition booléenne. Ce type de reconstruction de surfaces tridimensionnelles avait déjà été expérimenté dans les années 90 [Mur91, BTC95] et quand j'ai commencé à participer à ces travaux, nous étions confrontés à deux problèmes attendus. La surface implicite approxime trop grossièrement les points du contour, ce qui nous a amené, dans un premier temps, à utiliser une minimisation pour rapprocher la surface de ses contraintes. En complément du bruit dû au dessin et à l'acquisition du contour, la minimisation insère des oscillations dans la surface. Ces oscillations étaient à la fois dépendantes de l'échantillonnage du contour, de la largeur du support des primitives sphériques et de la continuité globale de la surface implicite. Dans [AGB04] (section 7.2.1) nous avons proposé l'utilisation de primitives sphériques à large support produisant des surfaces de continuité C^7 . En complément, nous avons proposé une représentation multirésolution lissant le contour afin de limiter les oscillations liées au bruit, et l'adaptation de l'échantillonnage du contour à l'erreur d'approximation locale de la surface reconstruite (figure 4.2(a)). Nous avons enfin proposé d'utiliser l'axe médian comme squelette pour implanter des opérateurs de couper/coller, copier/coller d'éléments de la surface en permettant de modifier à la fois la position et l'orientation des primitives à coller (figure 4.2(b)).

Au final, même si nous sommes arrivés à une précision d'approximation du contour raisonnable, nous n'avons pas pu éliminer complètement les oscillations de la surface (figure 4.2). Nous avons ainsi reproduit le problème auquel ont été confrontés nos prédécesseurs en reconstruisant des surfaces à l'aide de surfaces implicites sphériques. Pour éviter ce problème, une première idée était d'intégrer un terme proportionnel aux oscillations de la surface dans la fonction d'énergie à minimiser pour approximer le contour. Cette idée a été vite abandonnée quand nous avons compris que nous faisons fausse route. Dans une application de modélisation par esquisses, il faut prendre en compte l'imprécision du tracé du contour par l'utilisateur. C'est en réalité une aubaine, car c'est l'opportunité de contourner le problème en évitant de passer par un processus

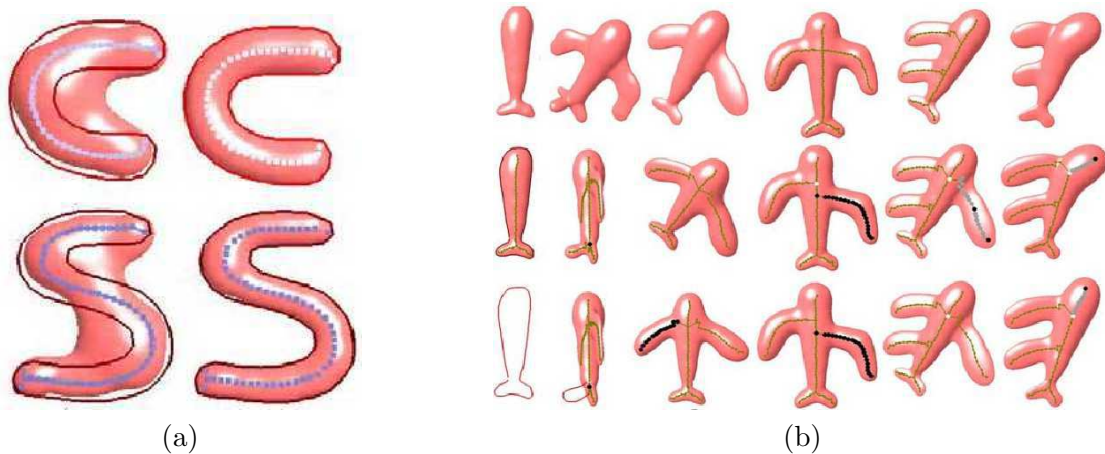


FIG. 4.2 – (a) Comparaison de l’ajustement de la surface par minimisation avec une reconstruction basée sur les surfaces de convolution utilisées dans [TZF04] à gauche et avec des “Soft-Objets” sphériques à droite. (b) Résultats obtenus par Alexe et al. dans [AGB04].

de minimisation. Non seulement la surface n’a pas à passer trop près des points du contour, mais ce n’est pas souhaitable. Il nous faut en fait définir le niveau de détail auquel l’utilisateur est en train de modéliser et reconstruire une surface lisse, au niveau de détail adéquat, intégrant éventuellement des arêtes franches. Dans nos travaux, nous nous concentrons sur la gestion du niveau de détail et la reconstruction d’une surface lisse n’oscillant pas, sans traiter la reconstruction des arêtes franches. C’est ici que commence ma collaboration avec Marie-Paule Cani sur la modélisation par esquisses.

4.3 Adapter le niveau de détail de la reconstruction au zoom et éviter les oscillations

Les travaux que nous allons voir maintenant ont été initiés par Anca Alexe dans [ABCG05] (section 7.2.2, figure 4.3(a)), puis complétés, améliorés et étendus par Adrien Bernhardt et Adeline Pihuit, deux étudiants de Marie-Paule Cani en thèse à Grenoble, dans [BPCB08] (section 7.2.3, figure 4.3(b)).

D’un point de vue modèle, nous avons tout d’abord décidé de conserver les représentations implicites définies par squelettes pour les raisons citées précédemment. Par contre, plutôt que d’utiliser des “Soft-Objects”, nous avons choisi des surfaces implicites de convolution [BS91, She99] pour leur faible tendance à osciller. Nous avons évité les problèmes d’oscillations dans la reconstruction illustrés figure 4.2(a) en localisant l’influence du noyau lors de l’intégration réalisant la convolution, comme proposé par Angelidis et al. [AJC02] et en n’effectuant pas de minimisation numérique. Nous avons plutôt proposé une fonction calculant le rayon des primitives afin qu’une fois convoluées, elles produisent une surface implicite approximant correctement le profil. Cette fonction s’appuie sur une évaluation ad-hoc locale du gonflement occasionné par l’intégration réalisant la convolution du noyau le long du squelette, et permet ainsi de le compenser. Pour implanter l’opérateur de différence, essentiel pour retirer de la matière à un objet, nous avons utilisé une adaptation des opérateurs de composition booléenne que j’ai présentés dans [BWG04]. Ces opérateurs produisent une zone de mélange lisse et arrondie entre les surfaces composées, là où elles sont raccordées. La taille de cette zone peut être contrôlée afin de produire une grande

transition arrondie entre les surfaces ou au contraire une petite transition.

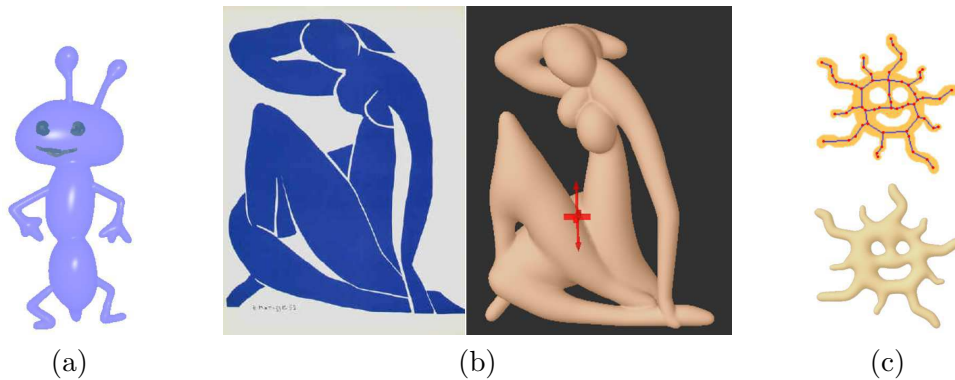


FIG. 4.3 – Modèles produits avec (a) le système d'Alexe et al. [ABCG05], et (b) le logiciel Matisse [BPCB08]. (c) Illustration de la reconstruction du squelette puis de la surface avec Matisse.

D'un point de vue contrôle du niveau de détail, nous introduisons le niveau de détail contrôlé automatiquement par le zoom. Cette idée vient du constat que, dans une session de modélisation, un utilisateur s'éloigne de l'objet pour créer sa forme grossière, ou ajouter des parties de grosse taille, tandis qu'il se rapproche de la surface pour y créer les détails. La raison vient de l'utilisation de la projection en perspective pour visualiser l'objet. L'espace de dessin est en général plan, face à l'utilisateur et de taille fixe en espace écran. Ainsi, imaginons qu'une partie de l'objet est déjà modélisée, relativement à la taille de cette partie de l'objet, un même contour produira une grosse surface si l'on est loin de la partie modélisée et un petit détail si l'on en est très proche. Le contrôle du niveau de détail doit être appliqué à la reconstruction de la surface, à la taille de la zone de mélange (un petit détail sera attaché à la surface avec une petite transition) et à la taille des facettes du maillage de la surface finale; tout cela de façon transparente pour l'utilisateur.

Le dessin du contour est acquis sous la forme d'une image qui est à la résolution de l'écran. Afin d'éliminer le bruit, nous commençons par réduire cette image en conservant les basses fréquences d'une transformée en ondelette de Haar appliquée deux fois. A partir de cette image simplifiée, nous utilisons une technique d'érosion basée image [Hal89] pour approximer l'axe médian qui servira de squelette à la surface de convolution. Conjointement, nous appliquons une transformée en distance pour reconstruire un champ de distance à partir des points du contour. Ainsi, nous connaissons la distance au contour de chaque point du squelette qui servira de support à la convolution. C'est à partir de ces données qu'est reconstruite la surface. On notera que, dans [ABCG05], le squelette peut être composé de polygones pour produire des surfaces aplaties (pour reconstruire une main par exemple) et, dans [BPCB08], cette surface est de topologie quelconque (figure 4.3(c)).

Le contrôle de la taille de la zone de mélange s'effectue simplement en fixant le paramètre de l'opérateur de composition correspondant à la taille du mélange à une valeur correspondant à une distance fixe dans l'espace écran de la caméra évaluée dans l'espace objet. Le maillage de la partie dessinée se fait en utilisant l'algorithme de Bloomenthal [Blo94] avec une taille de facette égale à la moitié de la plus petite distance entre le squelette de la surface reconstruite et le contour dans l'espace objet.

4.4 Discussions et retour vers l'étude des modèles volumiques

Dans ces travaux, nous avons proposé des solutions pour le sketching en général, comme le contrôle du niveau de détail par le zoom, et des solutions liées au choix du modèle de représentation de la surface reconstruite. Dans notre cas, des surfaces implicites définies par squelettes. Le prototype le plus abouti que nous avons obtenu suite à ces travaux est Matisse, un logiciel de modélisation par esquisses développé sous la supervision de Marie-Paule Cani à l'INRIA Rhône-Alpes. En l'état, un reproche classique est l'aspect rond et lisse des objets créés, ainsi que la difficulté de produire des détails fins sur la surface, ce qui fait que l'objet final reste en état de prototype. Parmi toutes les améliorations à apporter à ce système au niveau du modèle on peut noter : la gestion des arêtes franches et pointes dans la géométrie de l'objet, un meilleur contrôle du mélange quand de nouvelles parties sont ajoutées à l'objet (taille du mélange et localisation autour de la zone de jonction des parties assemblées), la préservation de la topologie lors de l'assemblage de nouvelles parties, l'intégration de la gestion de la coloration de la surface.

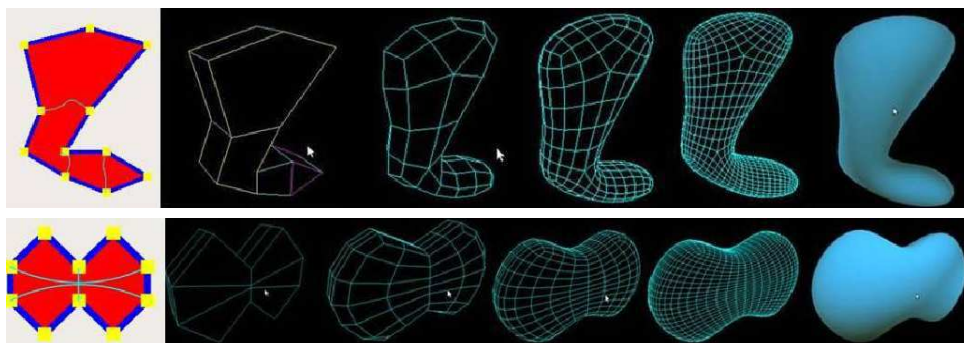


FIG. 4.4 – Image issue du rapport de stage de Master deuxième année de Jean-Patrick Roccia. Illustration de la reconstruction d'une surface de subdivision à partir de l'échantillonnage d'un contour dessiné par l'utilisateur.

D'un point de vue général, l'interaction utilisateur/système proposée par l'esquisse répond bien à notre critère initial d'accessibilité et de temps d'apprentissage réduit. Le positionnement des logiciels utilisant l'esquisse pour la modélisation de formes libres comme outils de prototypage est assez général. Même ShapeShop [Sha], un logiciel expérimental bien maintenu par Ryan Schmidt et ayant fait l'objet de plusieurs publications internationales [WFJ⁺05, SS08, SSB08], se situe dans cette catégorie. Nous sommes pendant l'année universitaire 2008-2009 et avant de chercher à aller plus en avant, il est bon de se reposer la question du modèle de représentation de surface qu'il nous semble opportun d'utiliser. Plusieurs travaux ont exploré les solutions à base de modèles volumiques : surfaces de convolution [TZF04, ABCG05, BPCB08], "Soft-Objects" [AGB04, Sha], surfaces implicites variationnelles [KHR02], "Level Sets" [EGB09]. D'autres travaux utilisent les maillages [IMT99, IH03] et j'ai encadré Jean-Patrick Roccia, un étudiant de Master deuxième année qui a étudié l'utilisation des surfaces de subdivision de Catmull-Clark en s'appuyant sur les travaux de Martinov et Kobbelt [MK06] pour générer le maillage initial à partir de l'esquisse. Le bilan est relativement explicite. Nous disposons d'importantes ressources, de travaux et de possibilité si nous utilisons les maillages et les surfaces de subdivision. Cependant, nous nous confronterons à tous les problèmes de stabilité numérique et aux difficultés d'implantation sur la plupart des algorithmes à mettre en œuvre pour réaliser un système de modélisation par esquisses. Si nous souhaitons offrir un accès au développement simple de systèmes de modélisation, les modèles volumiques semblent encore être les meilleurs

candidats. Récemment, Bein et al. [BHSF09] ont tout de même présenté un système par esquisses à base de maillages et de surfaces de subdivision (discuté à la fin de la section 3.6) et Brazil et al. [BMS⁺10] ont présenté l'utilisation des surfaces implicites variationnelles dites d'Hermite car elles interpolent des points et leur normale [MGV09]. Ceci nous ramène à développer les modèles volumiques afin d'essayer d'apporter des solutions efficaces aux différents problèmes qui se sont posés.

Surfaces implicites

5.1 Introduction

D'un point de vue général, une surface implicite est représentée par une iso-surface S dans un champ de potentiel défini par une fonction potentiel $f : \mathbb{R}^3 \rightarrow \mathbb{R}$. Ainsi, $S = \{p \in \mathbb{R}^3 \mid f(p) = c\}$ et en modélisation géométrique, on utilise de préférence des fonctions potentiel continues ce qui permet de définir explicitement le volume des objets par $V = \{p \in \mathbb{R}^3 \mid f(p) \geq c\}$ (la convention inverse où $f(p) \leq c$ peut aussi être utilisée). Qu'elles soient continues ou discrètes, c'est à ce type de représentation que nous faisons référence quand nous parlons de représentation volumique [Blo97b].

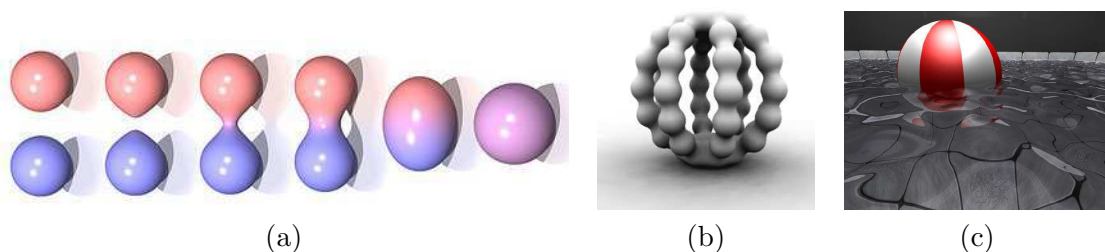


FIG. 5.1 – (a) Illustration du phénomène de mélange automatique présenté dans [Bli82] sur deux fonctions potentiel sphériques qui se rapprochent l'une de l'autre (image produite par Erwin de Groot). (b) Objet et (c) surface de l'eau produits par mélange de “Soft-Objects” sphériques [WMW86] ((b) image de www.fundza.com).

Les surfaces implicites ont été introduites en informatique graphique il y a déjà bien longtemps. Dans la fin des années soixante Sabin [Sab68] étudiait leur utilisation en CAO pour réaliser, entre autre, le mélange entre des objets assemblés. Ce mélange peut être vu comme un opérateur d'union booléenne avec une surface lisse et arrondie au niveau de la jonction des surfaces composées. Les primitives implicites utilisées sont essentiellement des quadriques dans lesquelles on retrouve le cylindre, la sphère et le cône. Les fonctions potentiel sont à support global et le volume est défini par $V = \{p \in \mathbb{R}^3 \mid f(p) \geq 0\}$. En 1973, Ricci présente une implantation des opérateurs de composition booléenne sur les surfaces implicites à partir des fonctions min et max [Ric73] (figure 5.2(a)). Ceci permet de composer entre elles des surfaces implicites dans un arbre CSG de façon très simple et efficace. Par exemple, la fonction potentiel $f = \max(f_1, f_2)$ est le résultat exact de l'union des surfaces représentées par les fonctions potentiel f_1 et f_2 . Le résultat de la composition de plusieurs surfaces implicites est une nouvelle surface implicite et si

nous disposons d'un algorithme performant permettant de visualiser une surface implicite, nous avons tout ce qu'il nous faut pour pouvoir visualiser le résultat de n'importe quelle composition de surface. Les difficultés se situent donc à trois niveaux. Proposer un/des algorithme(s) de visualisation, définir des modèles de représentation de surfaces à base de fonctions potentiel f_i et des opérateurs de composition g sous la forme $f = g(f_1, \dots, f_n)$ ($f = g(f_1, f_2)$ pour les opérateurs binaires) efficaces et pratiques.

En 1982, Blinn apporte une contribution majeure au domaine en présentant le "Blobby Model" [Bli82], des fonctions potentiel sphériques, définies positives à support global variant radialement en forme de gaussienne autour de leur centre, qui définissent une surface implicite lisse quand on les additionne : $f = \sum f_i$ et $V = \{p \in \mathbb{R}^3 \mid f(p) \geq c\}$, $c > 0$. Cette surface peut être vue comme le mélange des surfaces sphériques définies par $f_i = c$, l'opérateur g de mélange étant la somme. La première originalité de ce modèle est de permettre le mélange automatique des primitives sphériques qui sont additionnées dès qu'elles sont suffisamment proches les unes des autres (figure 5.1). Quatre ans plus tard, Wyvill et al. [WMW86] introduit les "Soft Objects" en reprenant ce principe et en utilisant des fonctions potentiel polynomiales définies positives à support compact. Il démontre une autre force de ces surfaces en présentant leur capacité à gérer automatiquement les changements de topologie quand les primitives sphériques sont animées. En règle générale, pour ces modèles, on prend $f_i(0) = 1$ et $c = 0.5$.

Nous notons dès maintenant que les fonctions potentiel à support global définissant un objet par $V = \{p \in \mathbb{R}^3 \mid f(p) \geq 0\}$ ne se composent pas de la même façon que les fonctions potentiel définies positives à support compact définissant un objet par $V = \{p \in \mathbb{R}^3 \mid f(p) \geq 0.5\}$. Par exemple, l'opérateur somme que nous venons de voir pour mélanger les "Soft-Objects" ne fonctionne pas sur des fonctions à support global telles que nous les avons définies. C'est un point important qui nous amène à différencier ces deux types de représentation.

5.2 Fonctions potentiel à support global

Ces fonctions potentiel, quand elles sont bien définies, ont une valeur maximale au "centre" de l'objet et tendent vers $-\infty$ lorsque l'on s'éloigne à l'extérieur de l'objet. Pour donner un exemple simple, la sphère de rayon r et de centre p_0 peut être définie par $V = \{p \in \mathbb{R}^3 \mid f(p) = r^2 - \|p - p_0\|_2^2 \geq 0\}$ ou par $V = \{p \in \mathbb{R}^3 \mid f(p) = r - \|p - p_0\|_2 \geq 0\}$ si l'on peut payer le surcoût de calcul que demande l'évaluation de la racine carrée. Beaucoup de primitives de ce type aux formes différentes ont été proposées dans la littérature comme les superquadriques [Bar81], les ratioquadriques [BS96], les primitives d'extrusion [CBS96, Gri99] et les primitives à squelette sous la forme $V = \{p \in \mathbb{R}^3 \mid f(p) = r - \text{dist}(p - Q) \geq 0\}$ où dist est la distance euclidienne, Q est une primitive géométrique appelée squelette (point, segment de droite, courbe, polygone, etc) et r est la distance entre la surface et le squelette. Suivent des fonctions potentiel reconstruisant globalement ou localement des nuages de points. Les surfaces implicites variationnelles utilisant des fonctions de base à support global (RBF) [TO99, Isk02], une variante interpolant des normales en complément des points reconstruits [MGV09], les "point set surfaces" (PSS) [AA03, AK04] et une variante plus stable utilisant la projection sur des sphères (APSS) [GG07].

Ainsi, des surfaces aux formes très variées peuvent être créées et pour modéliser des objets complexes, la puissance des surfaces implicites réside dans la capacité à définir des opérateurs de composition booléenne avec ou sans mélange de façon fonctionnelle. Les propriétés fondamentales des opérateurs de composition s'appliquant aux fonctions potentiel à support global ont été présentées par Rockwood [Roc89] en même temps qu'il présentait une série d'opérateurs de mélange binaire utiles en CAO. Plus tard, Pasko et al. [PASS95] introduit les "R-functions" [Rva63]

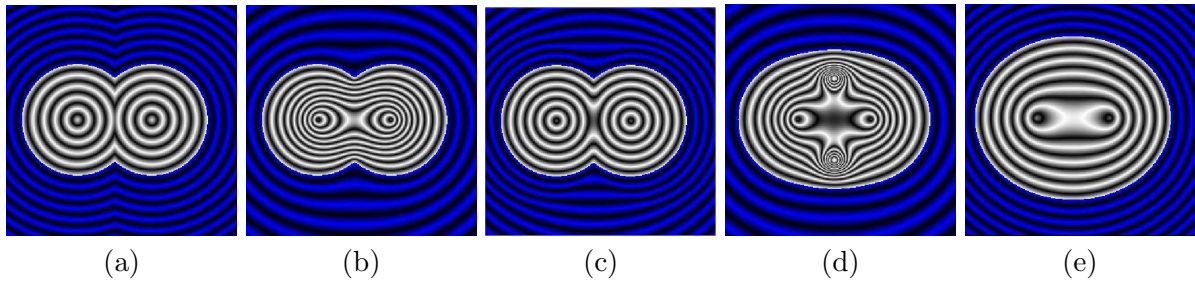


FIG. 5.2 – Illustration des iso-surfaces dans une coupe 2D après l’application d’un opérateur g (a,b,c) d’union et (d,e) de mélange sur deux fonctions potentiel sphériques à support global. La fonction résultante est positive dans la zone blanche et négative dans la zone bleue. (a) $g = \max(x, y)$ [Ric73], notons la discontinuité de dérivées premières à l’extérieur de la surface. (b) Union propre de Pasko et al. [PASS95]. Maintenant la fonction potentiel résultante est de continuité au moins C^1 en dehors de la surface. (c) Union propre de Barthe et al. [BDS⁺03]. En complément les variations des iso-surfaces de la fonction potentiel sont bien plus régulières. (d) Mélange de Pasko et al. [PASS95]. La fonction potentiel présente de fortes distorsions. (e) Mélange de Barthe et al. [BDS⁺03]. Les variations de la fonction potentiel sont régulières et fidèles à la forme de la surface finale.

comme outil pour définir des opérateurs de composition sans mélange (opérateurs booléens classiques), mais produisant des fonctions potentiel de continuité au moins C^1 partout en dehors de la surface (figure 5.2(b)). C’est l’occasion d’ouvrir une petite parenthèse - une “R-fonction” n’est pas une fonction réelle ou encore de $\mathbb{R}^n \rightarrow \mathbb{R}^m$. C’est une fonction de Rvashev (Rvashev function) qui est une fonction qui ne peut changer de signe que si et seulement si l’un de ses arguments change de signe. Par exemple $g(x, y) = \max(x, y)$ est une “R-fonction” - fermeture de la parenthèse. Cette propriété est nécessaire pour assurer que la nouvelle surface issue de la composition pourra par la suite être assemblée à une autre surface avec un mélange lisse (figure 5.3(a)-bas). En effet, la continuité de $f = g(g_0(f_1, f_2), f_3)$ sera C^0 si le mélange produit par g traverse la zone C^0 de $f_4 = g_0(f_1, f_2)$. Ceci crée une arête indésirable dans la surface correspondant au mélange (figure 5.3(a)-haut). Les opérateurs permettant d’éviter cette arête indésirable sont appelés “union (intersection ou différence) propre”. Pasko et al. [PASS95] propose aussi une extension de leurs opérateurs “propres” pour créer un mélange lors des compositions booléennes (figure 5.2(d)).

J’ai commencé à travailler sur la composition des surfaces implicites en septembre 1996, en stage de Master deuxième année sous la direction de Véronique Gaildrat, en prenant la suite des travaux de Nilo Stolte qui venait de finir sa thèse et de quitter Toulouse [Sto96]. Quand on modélise des objets complexes avec des surfaces implicites, on est amené à effectuer un grand nombre de compositions et la qualité du contrôle ainsi que la correspondance au résultat attendu par l’utilisateur dépend énormément des propriétés des fonctions potentiel générées par les opérateurs. Ce propos a par exemple été illustré figure 5.3(a). Durant ma thèse, j’ai commencé à étudier le fonctionnement théorique des opérateurs de composition binaire et le lien qu’il y a entre la forme de l’opérateur $g : \mathbb{R}^2 \rightarrow \mathbb{R}$ et la fonction potentiel obtenue lorsque l’on compose deux fonctions potentiel avec $g : f = g(f_1, f_2)$. Suite à ma soutenance de thèse en septembre 2000, j’ai continué à travailler sur ce domaine et j’ai finalisé la première partie de cette recherche pendant mon post-doctorat à Cambridge. Ce fut aussi l’occasion de commencer une collaboration avec Brian Wyvill [BDS⁺03] (section 7.3.1). Une conclusion de ce premier travail théorique est que les opérateurs tels qu’ils sont définis de façon générale permettent de

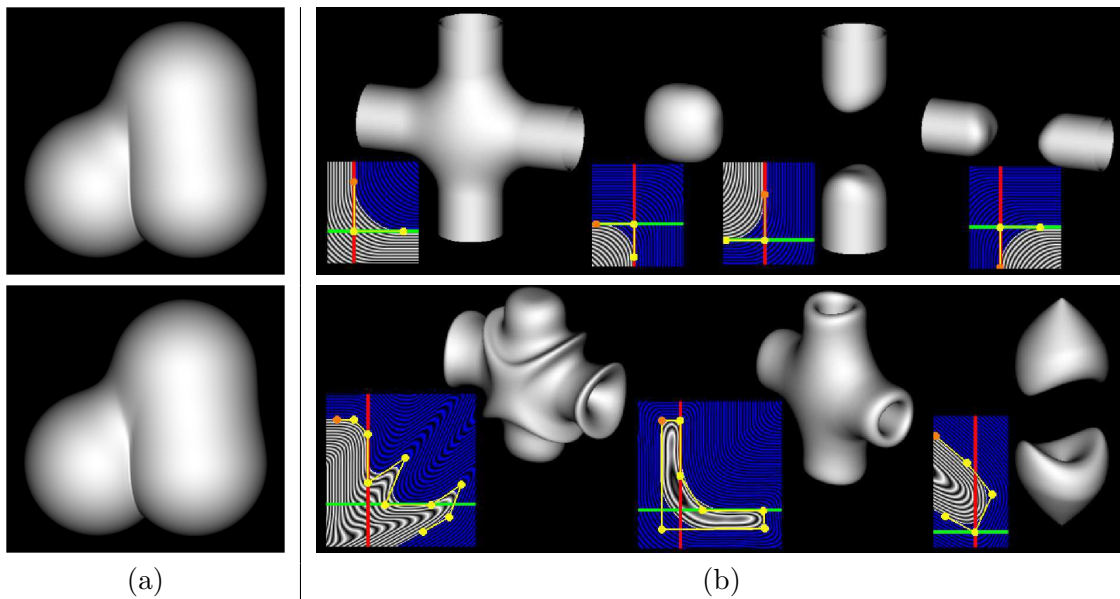


FIG. 5.3 – (a) Une troisième sphère est mélangée à l’union de 2 autres. En haut, si l’union est réalisée avec un opérateur comme celui illustré figure 5.2(a), une arête indésirable est produite au milieu du mélange. En bas, si un opérateur comme celui illustré 5.2(b,c) est utilisé, le mélange est lisse, comme désiré. (b) En blanc, l’intérieur des objets et en bleu l’extérieur. Illustration de l’application de différents opérateurs $g(x, y)$ sur deux fonctions potentiel cylindriques orthogonales. En haut, de gauche à droite, l’union avec mélange, l’intersection avec mélange, et les deux différences avec mélange. En bas, des opérations différentes produites par des opérateurs g aux formes plus exotiques.

produire des résultats bien plus variés que les simples opérateurs de composition booléenne, avec ou sans mélange (figure 5.3(b)). Plus intéressant d’un point de vue théorique, ces travaux nous apportent une compréhension très fine du lien qu’il existe entre opérateurs et fonctions potentiel produites. Ils nous permettent de travailler sur la construction d’opérateurs g , non pas à partir d’équations et du résultat qu’elles produisent, mais, dans un premier temps, à partir du graphe des opérateurs produisant les propriétés souhaitées sur les fonctions potentiel composées, puis dans un deuxième temps, sur les équations qui vont permettre de se rapprocher au mieux du graphe adéquat pour g . En complément de curiosités théoriques (figure 5.3(b)-bas), nous avons créé des opérateurs de composition booléenne avec et sans mélange dont la taille et la forme du mélange est facilement contrôlable par un utilisateur. De plus, nos opérateurs produisent des fonctions potentiel ayant peu de distorsions et des variations plus fidèles à la forme de la surface finale (figures 5.2(c),(e) et figure 5.3(b)-haut). Ces opérateurs de mélange sont ceux qui ont été utilisés dans le logiciel de modélisation par esquisses à base de fonctions implicites variationnelles proposé par Karpenko et al. [KHR02].

En finissant ces travaux, j’ai aussi eu une réflexion personnelle sur l’intérêt de travailler sur les surfaces implicites à support global. En effet, l’un des désavantages des surfaces implicites est la difficulté que l’on a à les visualiser en temps interactif lors de la modélisation surtout quand l’objet commence à être complexe. Lorsque l’on visualise une surface implicite, la majeure partie des temps de calcul est consacrée à l’évaluation de la fonction potentiel représentant l’objet final. Cette fonction est le résultat de la composition par les opérateurs g_i des primitives de base f_i et sa complexité calculatoire dépend donc de celle des f_i et des g_i . Comme il n’y a pas de

localisation des fonctions dans l'espace (ce que nous donne le support compact), la complexité de calcul est maximale en tout point de l'espace et chaque nouvelle composition dans un objet en cours de modélisation augmente cette complexité en tout point. De plus, il est impossible de garantir la localité d'un opérateur, et donc, les algorithmes de maillage incrémentaux (se concentrant uniquement sur les zones modifiées) sont difficiles à mettre en œuvre et ne sont pas robustes. Il est donc clair que si l'on veut pouvoir plus facilement augmenter l'interactivité lors de la modélisation et si l'on veut pouvoir garantir la localité des opérations, il nous faut utiliser des fonctions potentiel à support compact. En même temps que je commençais à m'intéresser aux surfaces de subdivision, je me suis aussi intéressé à la composition de fonctions potentiel à support compact.

5.3 Fonctions potentiel à support compact

Les fonctions potentiel à support compact offrent l'avantage d'être localisées dans l'espace. Ainsi, en utilisant des boîtes englobantes par exemple, nous limitons le coût d'évaluation d'une fonction potentiel, même après de nombreuses compositions. De plus, il devient bien plus aisé et pertinent de proposer des algorithmes de visualisation incrémentaux [GA00] afin d'augmenter l'interactivité lors de la modélisation. Nous sommes début 2002, et à ce moment là, il semble admis que les opérateurs de composition développés pour les fonctions potentiel à support global fonctionnent aussi sur les fonctions potentiel à support compact, dès lors que l'on positionne leur iso-surface en zéro : $f = g(f_1 - 0.5, f_2 - 0.5) + 0.5$ où f, f_1, f_2 sont des fonctions potentiel à support compact et g est un opérateur défini pour des fonctions potentiel à support global.

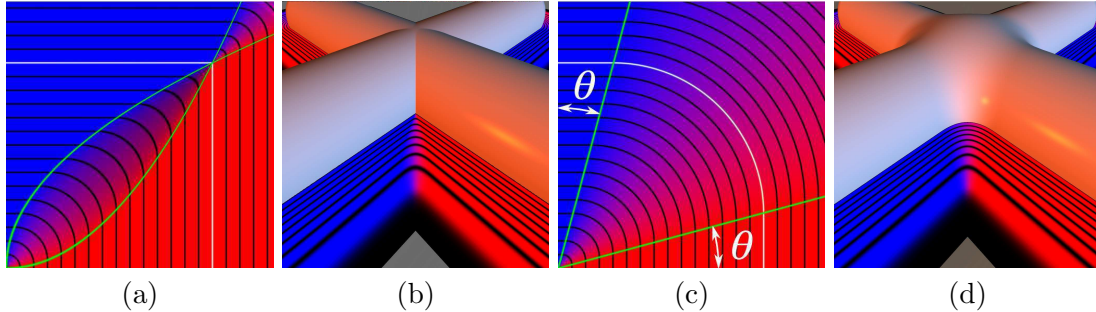


FIG. 5.4 – En (a) et (c), la ligne blanche représente la 0.5 iso-courbe de l'opérateur g et les couleurs bleu et rouge symbolisent l'orientation du gradient de g (rouge = horizontal et bleu = vertical). La zone entre l'origine et la courbe blanche correspond à l'extérieur de la surface après composition. (a) Graphe de l'opérateur d'union présenté dans [BWG04] et (b) son application dans la composition de deux cylindres orthogonaux. (c) Graphe de l'opérateur de mélange présenté dans [BWG04] et (d) son application dans la composition de deux cylindres orthogonaux.

Pourtant, si l'on regarde les travaux de Wyvill et al. [WGG99] qui intègrent les opérateurs d'union, d'intersection et de différence avec mélange dans un arbre de composition de fonctions potentiel à support compact, on s'aperçoit que l'opérateur de mélange proposé réalise la même composition mais sans mélange sur les fonctions potentiel à support global, comme expliqué en détail dans [BWG04]. En réalité, l'application de notre analyse des opérateurs et de leur impact sur les fonctions potentiel portée sur les représentations à support compact, nous montre qu'il y a une complication importante lorsqu'on doit les composer : la gestion de la frontière du support

au delà de laquelle la fonction est nulle partout. Ceci impose, en complément, de réaliser la forme de mélange souhaitée au niveau de l'iso-surface 0.5, de produire une fonction potentiel variant convenablement, et d'avoir $g(0, y) = g(x, 0) = g(0, 0) = 0$ en supplément de contraintes différentielles. De ce fait, ni les opérateurs de mélange proposés par Pasko et al. [PASS95], ni ceux que nous avons proposés dans [BDS⁺03] ne peuvent être utilisés pour mélanger des fonctions potentiel à support compact. Les seuls opérateurs réutilisables sont les opérateurs sans mélange que nous avons présentés dans [BDS⁺03]. Ils composent les bornes des supports sans mélange (union propre, ce que l'on souhaite) et mélangent localement les autres iso-surfaces (dont la 0.5 iso-surface), produisant ainsi un opérateur de mélange [BWG04] (figures 5.4(c) et (d)). Dans [BWG04], nous proposons aussi des opérateurs de composition réalisant des compositions sans mélange mais avec une fonction potentiel lisse en dehors de la surface (union, intersection et différence propre). Ceci nous amène à un opérateur dont le graphe est présenté figures 5.4(a) et (b).

5.4 La trêve

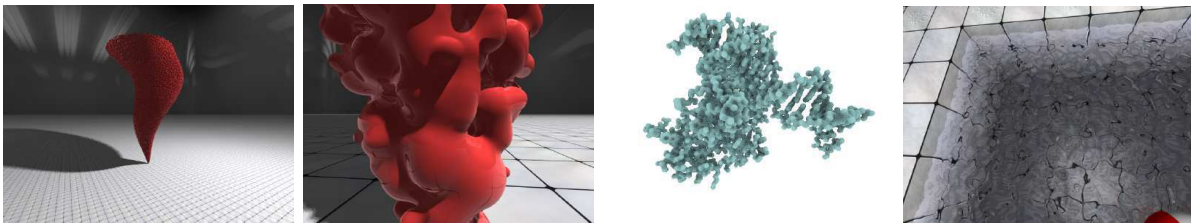


FIG. 5.5 – Exemples de scènes visualisées avec la technique présentée dans [OGP10]. Par exemple, la tornade à gauche contient 128.000 primitives sphériques et est visualisée à 1 image par seconde (avec tous les effets : ombres, plusieurs niveaux de réflexions, etc). En zoomant, le nombre de primitives visibles diminue et on arrive à environ 5 images par seconde. La piscine est composée de 10.000 primitives et est visualisée à 2 images par seconde.

J'ai arrêté toute recherche active sur les modèles de représentation implicite à la fin de l'année 2002. Après, il ne s'est agi que de rédiger la dernière publication. Il y a plusieurs raisons à cela. Tout d'abord mes travaux sur les surfaces de subdivision me prenaient la majeure partie de mon temps. De plus la communauté de modélisation géométrique manifestait un désintérêt croissant envers les représentations implicites. Parmi toutes les raisons que l'on peut évoquer, nous noterons qu'après plusieurs années de recherche active, il restait difficile de proposer une visualisation rapide de modèles de grande taille, le contrôle de la composition, notamment du mélange, n'était toujours pas satisfaisant, il était difficile de produire des modèles aux détails nombreux et très fins, l'application de textures était laborieuse. Bref, en dehors de quelques travaux [PPK05, SGW06, SdGWS08, dGWvdW09, SWS10] localisés autour de peu d'équipes de recherche, au niveau des modèles, tout allait rester plus ou moins en l'état jusqu'à nos jours. Les surfaces implicites n'étaient plus considérées, par beaucoup, comme une représentation de surface suffisamment intéressante pour être approfondie. Alors que je travaillais sur de la modélisation par esquisses avec Marie-Paule Cani et son étudiant Adrien Bernhardt, nous avons été confrontés à des problèmes de contrôle de la localité et de la taille du mélange lors de l'assemblage de nouvelles parties de l'objet dessinées par l'utilisateur. Le logiciel développé sous l'encadrement de Marie-Paule Cani [BPCB08] est à base de surfaces implicites de convolution [BS91, She99]. Même si ces fonctions potentiel ne se composent pas tout à fait comme les fonctions potentiel à

support compact, les adaptations sont suffisamment simples à effectuer (les détails sont donnés dans [BBCW10]) pour que nous les considérons dans la même famille.

Nous voilà donc début 2009, un peu plus de six ans plus tard, avec un nouveau besoin au niveau des modèles de représentation de surfaces. Prenant sérieusement en compte la discussion exposée section 4.4, et considérant la forte différence de ressources en calculs (via le CPU et le GPU) et en mémoire, j'ai pris part à ce projet. En parallèle, je co-encadrais Olivier Gourmel et Anthony Pajot avec Mathias Paulin et Pierre Poulin, sur la visualisation haute qualité, en temps-réel ou interactif, d'un nombre massif de fonctions potentiel sphériques dynamiques à support compact [OGP10] (section 7.3.3). Nous avons proposé une solution efficace utilisant, en parallèle, le CPU pour la construction d'une hiérarchie de volumes englobants (BVH) et le GPU pour réaliser un lancé de rayon sur la BVH d'abord, puis sur les fonctions potentiel d'intérêt ensuite. Le résultat de ce travail m'a conforté dans l'idée qu'effectivement, l'utilisation des fonctions potentiel à support compact couplée aux ressources matériel actuelles nous permet d'arriver à un rendu vraiment efficace (figure 5.5). Revenons-en à la modélisation. La solution proposée par Marie-Paule Cani et Adrien Bernhardt était de reprendre l'idée (manuelle) proposée par Pasko et al. [PPK05] sur des fonctions potentiel à support global et de l'adapter aux primitives de convolution avec un usage automatique. Nous voilà donc repartis à travailler au niveau modèle en s'attaquant à l'un des verrous important : le contrôle précis et automatique de la taille et de la localisation du mélange lors des compositions.

5.5 Vers une nouvelle génération d'opérateurs de composition

Pasko et al. [PPK05] définit un opérateur de composition, contrôlé par un paramètre. L'élégance de cet opérateur est de produire un mélange dont la taille est nulle quand le paramètre vaut 0 - il produit ainsi une union propre - et un mélange de plus en plus large quand le paramètre grandit, jusqu'à une taille maximale atteinte quand le paramètre vaut 1. Lors de l'utilisation de cet opérateur pour composer deux fonctions potentiel à support global, il faut positionner une fonction potentiel à support compact (variant de 0 à son support jusqu'à 1 en son centre), dans chaque zone où le mélange est désiré. La/les fonction(s) potentiel à support compact définissent directement la valeur du paramètre de l'opérateur de composition et le résultat escompté est ainsi obtenu : mélange dans les zones délimitées par les fonctions potentiel à support compact et union propre ailleurs.

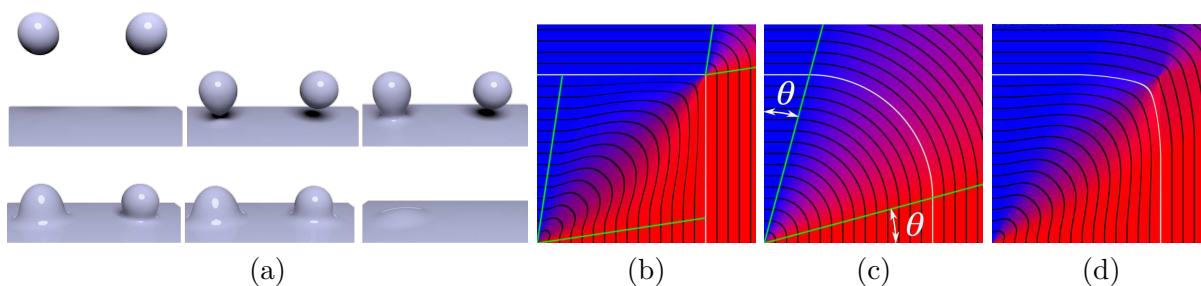


FIG. 5.6 – (a) A gauche un mélange classique et à droite l'effet obtenu avec le nouvel opérateur de Bernhardt et al. [BBCW10]. L'opérateur de Bernhardt et al. [BBCW10] interpole l'opérateur d'union propre illustré en (b) et l'opérateur de mélange de Barthe et al. [BDS⁺03] illustré en (c) pour donner l'opérateur illustré en (d) quand le paramètre d'interpolation vaut approximativement 1/2.

Il y a deux contributions importantes dans les travaux d’Adrien Bernhardt [BBCW10] (section 7.3.4). La première est la création d’un opérateur avec les mêmes propriétés que celui de Pasko et al. [PPK05], mais adapté aux fonctions potentiel à support compact. La deuxième est la génération automatique des fonctions potentiel localisant le mélange, dans la proposition d’Adrien, uniquement au niveau de l’intersection des parties de l’objet qui sont assemblées (figure 5.6(a)).

Les opérateurs que nous avons proposés sont construits à partir de ceux qui ont été présentés dans [PASS95] et [BDS⁺03]. Notre solution est basée sur l’interpolation entre un opérateur d’union propre (figure 5.6(b)) et un opérateur de mélange (figure 5.6(c)). Le paramètre d’interpolation est la valeur de la fonction potentiel de localisation du mélange. Nous n’avons pas utilisé l’opérateur d’union propre que j’avais proposé dans [BWG04] car il est bien trop coûteux en calculs. La fonction potentiel localisant le mélange est construite à partir d’un champ de distance autour des poly-lignes d’intersection des surfaces des fonctions potentiel qui sont composées. Cette poly-ligne est extraite avec un algorithme de “marching face” [LY03]. Les faces sont issues de la polygonisation des surfaces composées ou d’un “marching cubes” appliqué à des voxels situés autour de l’intersection. Ces voxels sont calculés en utilisant les fonctions d’inclusion des équations des fonctions potentiel composées [Duf92, Sny92, FSSV06].



FIG. 5.7 – Exemples d’objets réalisés avec un logiciel de modélisation par esquisses utilisant les opérateurs de composition de Bernhardt et al. [BBCW10].

Fin septembre 2009, nous disposons donc de cet outil qui est un premier pas vers une nouvelle génération d’opérateurs de composition. Ainsi, un opérateur ne se contente plus de réaliser une union propre ou un mélange. Il intègre les deux fonctionnalités et permet de passer de façon continue de l’une à l’autre. D’un point de vue pratique, ceci permet d’éviter le mélange à distance qui fait que deux primitives suffisamment proches vont commencer à se mélanger alors qu’elles ne se touchent pas (figure 5.6(a)). Ceci permet aussi de localiser le mélange dans les zones d’intersection seulement et enfin, il est possible de choisir la taille du mélange en fonction du niveau de détail local des surfaces [BBCW10]. Tout ceci de façon automatique et transparente pour l’utilisateur. La figure 5.7 montre quelques objets réalisés avec un logiciel de modélisation par esquisses intégrant ce nouvel opérateur. Les principales limitations de ce nouvel opérateur sont les distorsions dans la fonction potentiel résultante (figures 5.6(b) et (d)) et la difficulté de passer à l’échelle sur des objets très complexes intégrant beaucoup de compositions à différentes résolutions (comme un arbre par exemple) à cause de l’extraction de la poly-ligne localisant le mélange.

Nous avons ainsi enchaîné avec une idée qui avait été juste évoquée en quelques lignes par Rockwood en 1989 [Roc89] : contrôler l’interpolation entre l’opérateur d’union propre et le mélange non plus avec une fonction potentiel, mais avec l’angle des gradients des fonctions potentiel composées [GBC⁺11] (section 7.3.5). Ce nouveau projet a été mené par Olivier Gourmel, un étudiant en début de troisième année de thèse sous la direction de Mathias Paulin. Notre objectif est d’ar-

river à traiter automatiquement et sans utiliser de structure complémentaire les problèmes que nous avons résolus dans [BBCW10] (figures 5.8(b) et (c)), ainsi que d'éviter les gonflements indésirables (figures 5.8(a)) et conserver la topologie lors du mélange (figures 5.8(d)).

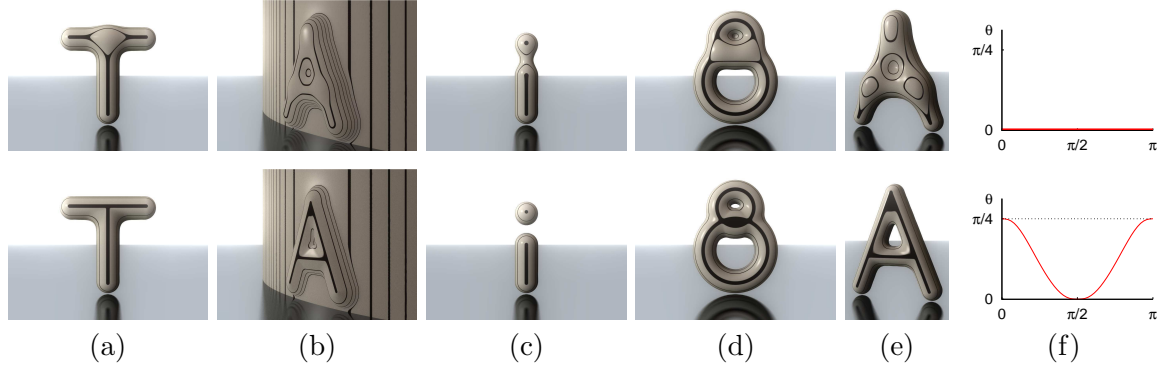


FIG. 5.8 – Illustration des problèmes de mélange résolus par notre opérateur dépendant de l'angle entre les gradients des fonctions potentiel composées [GBC⁺11]. Ligne du haut, un mélange standard et ligne du bas notre nouvel opérateur. (a) Le gonflement indésirable, (b) le voilage des détails, (c) le mélange à distance et (d) la modification de la topologie. (e) Problèmes (a,c,d) illustrés sur le même objet et (f), tracés des contrôleurs permettant d'obtenir ces différents résultats avec notre opérateur.

L'utilisation des gradients dans la définition de l'opérateur de composition introduit une contrainte supplémentaire sur sa création : à chaque composition, la continuité de la nouvelle fonction potentiel diminuera d'un degré au moins. Ainsi, si notre opérateur est de continuité G^1 , après une composition, la fonction potentiel résultante sera de continuité G^1 au mieux. Si on applique une nouvelle composition au niveau de ce mélange, la fonction potentiel résultante sera C^0 au mieux et si on réitère, on obtiendra une surface discontinue. Il nous faut donc prévoir des opérateurs de composition à la continuité suffisamment élevée. Pour ne pas faire d'hypothèse sur le nombre maximum de mélanges se chevauchant, nous proposons des opérateurs de classe C^∞ .

Une autre innovation que nous proposons est de profiter de l'utilisation de fonctions potentiel à support compact pour stocker les opérateurs de composition dans des textures sur GPU afin de ramener leur coût d'évaluation à un simple accès texture, quelle que soit la complexité des équations utilisées pour les définir. Ceci est bien plus utile qu'il n'y paraît. En effet, nous levons ainsi une contrainte très forte qui était de proposer des équations satisfaisant au mieux les contraintes qui étaient posées sur les opérateurs tout en conservant un coût d'évaluation suffisamment faible. Maintenant, nous pouvons choisir des fonctions arbitrairement complexes, même définies par des équations résolues numériquement si nécessaire. Il devient possible de définir de façon précise la forme souhaitée pour les opérateurs. Nous avons déterminé le tracé des opérateurs illustrés figure 5.9 en utilisant les connaissances théoriques présentées dans [BDS⁺03, BWG04] et nous les avons définis à l'aide des fonctions de classe C^∞ présentées dans [GBC⁺11].

Plutôt que d'interpoler entre deux opérateurs (union propre et mélange) comme nous l'avons fait dans [BBCW10], nous proposons un opérateur contrôlé directement par un angle d'ouverture (dans l'esprit de [BDS⁺03]). Quand cet angle vaut $\pi/4$, nous définissons un opérateur d'union propre (figure 5.9(a)), et quand l'angle diminue, un mélange de plus en plus large est créé (figure 5.9(b)) jusqu'à atteindre un mélange maximal quand l'angle est égal à 0 (figure 5.9(c)). Avec ce principe, nous proposons une construction générique des opérateurs permettant de

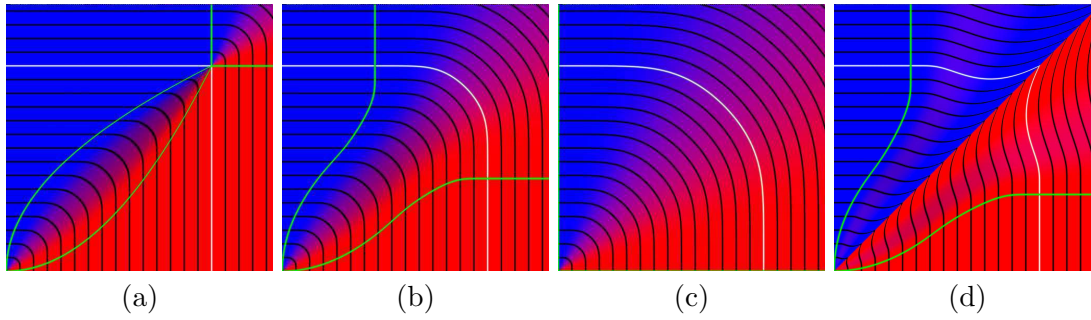


FIG. 5.9 – Illustration de notre opérateur contrôlé par l’angle des gradients [GBC⁺11] avec (a) un angle d’ouverture de $\pi/4$, (b) un angle de $\pi/8$ et (c) un angle égal à 0. (d) Illustration de l’opérateur de gonflement au contact.

définir simplement la forme de l’opérateur dans la zone de déformation. Ainsi, nous avons créé un opérateur n’introduisant qu’un minimum de distorsions dans la fonction potentiel résultant de la composition et de variation de courbure sur la surface dans la zone de mélange. Nous avons aussi créé un opérateur permettant de réaliser du gonflement au contact pour simuler l’écrasement sur des objets mous lorsqu’ils rentrent en contact, comme par exemple deux doigts de la main (figures 5.9(d), 5.11(b) et (c)).

Revenons-en au cœur théorique de notre contribution : le contrôle automatique de l’ouverture de l’opérateur par l’angle entre les gradients des fonctions potentiel composées. Nous introduisons une fonction $h : [0, \pi] \rightarrow [0, \pi/4]$ appelée “contrôleur” qui prend en entrée l’angle entre les gradients et retourne en sortie l’angle d’ouverture de l’opérateur. Par exemple, un contrôleur constant $h = 0$ (figure 5.8(f)-haut) produit un mélange maximal quel que soit l’angle entre les gradients, c’est à dire un mélange classique (figure 5.8-haut). Un contrôleur comme celui présenté figure 5.8(f)-bas permet quant à lui de régler tous les problèmes que nous avons soulevés (figure 5.8-bas). Nous proposons aussi un contrôleur spécifique pour la modélisation de formes organiques (figure 5.10) qui permet de réaliser une composition dissymétrique plus naturelle. Ainsi la cuisse du chameau, comme sa queue (figures 5.10(c) et (d)) se mélangent sur le dessus, mais forment une union dans les zones de pincement.

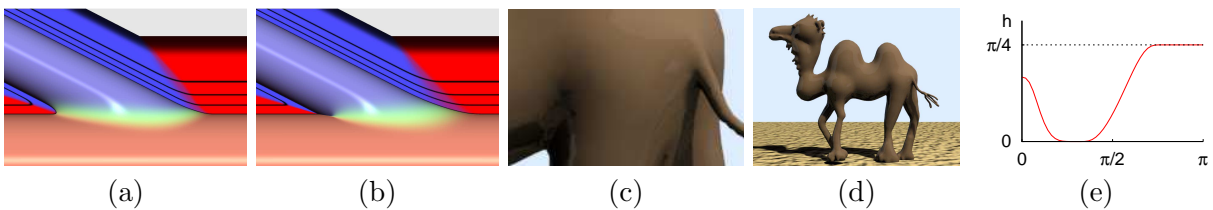


FIG. 5.10 – Illustration du résultat obtenu en composant les fonctions potentiel avec notre opérateur associé au contrôleur organique. (a) Le résultat avec le contrôleur de la figure 5.8(f)-bas et (b) la même composition avec le contrôleur organique. (c,d) Utilisation de notre contrôleur organique en modélisation de formes complexes et (e) tracé du contrôleur organique.

Pour finir, nous avons aussi proposé un contrôleur spécifique à l’utilisation de l’opérateur d’écrasement au contact (figures 5.9(d), 5.11(b) et (c)), ce qui nous a permis de produire les résultats présentés figure 5.11. On peut noter qu’en animation, il peut être intéressant de faire varier le contrôleur pour que, par exemple, des primitives ne se mélangent pas tant qu’elles ne

sont pas en contact, mais se mélangent quand elles se scindent. Il est aussi possible d'utiliser différents opérateurs dans une même animation comme nous pouvons le voir sur l'image de la lavalampe, figure 5.11(c).

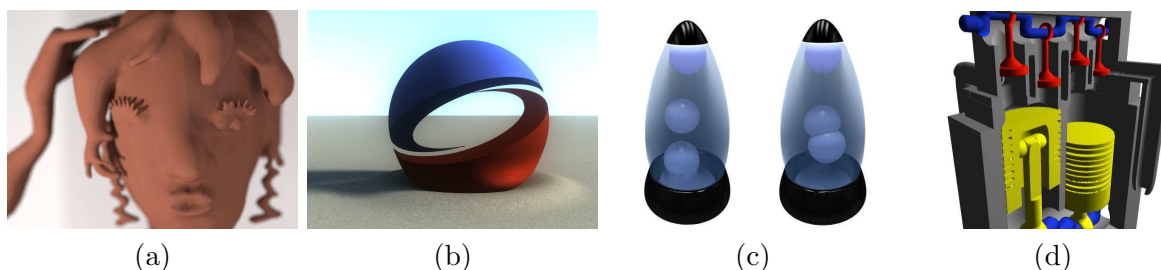


FIG. 5.11 – Différents résultats produits avec nos opérateurs et nos contrôleurs.

Nous obtenons au final des opérateurs très pratiques et efficaces. Ils apportent de nouvelles solutions et ouvrent de nouvelles voies à explorer. Leurs limitations principales sont : ils sont binaires et ne se prêtent donc pas au mélange massif d'un très grand nombre de fonctions potentiel, et ils nécessitent des fonctions potentiel au moins G^1 afin de pouvoir calculer les gradients. En cas de discontinuité de gradient dans une fonction potentiel, un traitement particulier peut tout de même être effectué (voir les détails dans [GBC⁺11], section 7.3.5).

5.6 Discussions et perspectives : de nouveaux projets

Comme nous pouvons le voir, même s'il sont découpés dans le temps en deux périodes : la suite de ma thèse puis la reprise motivée par les travaux d'Adrien Bernhardt et Marie-Paule Cani [BBCW10], tous ces travaux sur les opérateurs de composition suivent un développement qui nous fait partir de l'amélioration de l'union et du mélange standard pour arriver à des opérateurs génériques aux propriétés plus avancées et à l'utilisation bien plus efficace. Alors que les premiers travaux introduisent des outils théoriques et apportent des avancées plutôt incrémentales, nos deux dernières publications [BBCW10, GBC⁺11] exploitent ces outils pour être porteuses de nouveaux concepts et résoudre plusieurs problèmes déjà identifiés pour certains [Roc89, Can93, GW95, DC95, Blo97a, PPK05], mais jamais abordés dans leur globalité. Ces nouveaux résultats sont très prometteurs et nos opérateurs sont déjà implantés dans plusieurs logiciels de modélisation par esquisses.

D'un point de vue opérateur de composition, les opérateurs binaires commencent à être satisfaisants. La seule chose que je modifierais, c'est rendre l'ouverture de l'opérateur à la fois dépendante de l'angle entre les gradients et de la valeur des fonctions potentiel composées afin d'ouvrir rapidement l'opérateur quand on s'éloigne de la surface pour bien arrondir le champ de potentiel autour d'une arête franche. L'étape suivante me semble être le développement d'un opérateur n-aire capable de faire cohabiter tous nos opérateurs - union franche, mélange, contrôle par les gradients, écrasement au contact, évolution des contrôleurs au cours du temps - pour permettre l'animation d'un grand nombre de primitives avec un résultat plus réaliste, à moindre effort pour l'utilisateur. Je vois quelques pistes à explorer pour commencer ce projet. Tout d'abord, il me semble opportun de reprendre l'idée d'un graphe de composition introduite dans [GW95, DC95]. Les nœuds du graphe sont les fonctions potentiel, et les arêtes portent les opérateurs binaires et leurs paramètres. Ce graphe serait dynamique pour permettre de créer des animations. Pour permettre de composer toutes ces compositions binaires afin d'obtenir la fonction potentiel finale, je pense m'orienter dans un premier temps vers une partition de l'unité.

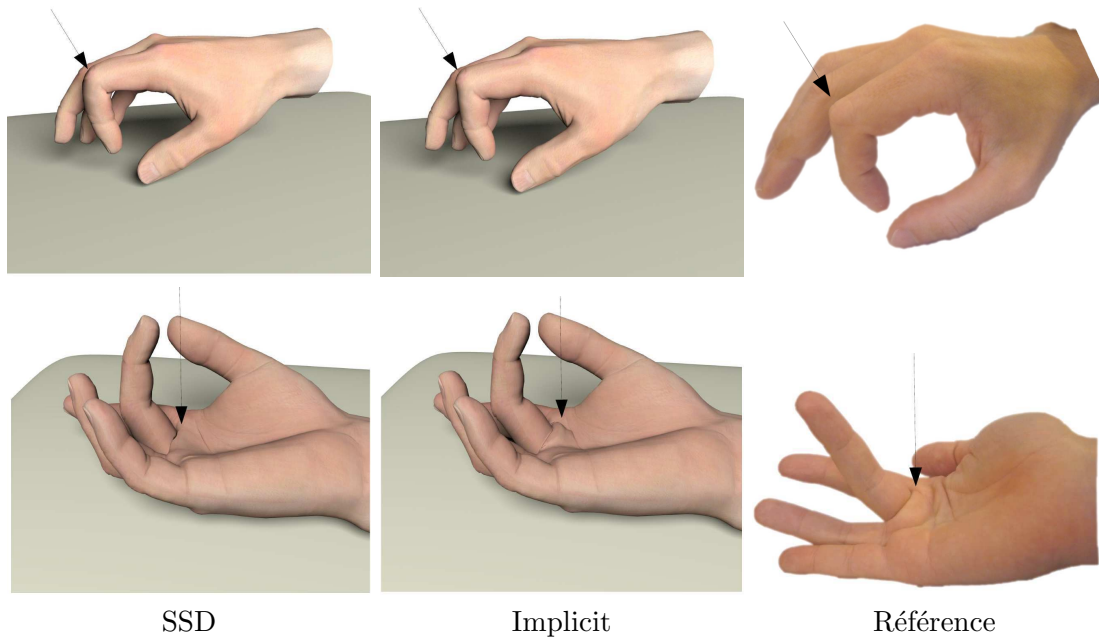


FIG. 5.12 – Comparaison du skinning d’une main guidée par la composition de cylindres implicites avec les SSD à gauche et la photo d’une vraie main à droite.

Nous avons aussi commencé un autre projet en collaboration avec Marie-Paule Cani et Gaël Guennebaud sur l’utilisation de nos opérateurs [GBC⁺11] pour guider la déformation d’un maillage dans une application de “skinning” temps-réel. Ce projet a été débuté par Olivier Gourmel et est continué par Rodolphe Vaillant qui est en stage de Master deuxième année. Nous avons déjà de très bons résultats sur des maillages cylindriques comme les doigts de la main, les bras, les jambes 5.12. Le principe est relativement simple. Nous positionnons des cylindres implicites autour des segments du squelette d’animation. Ces cylindres sont mélangés avec nos opérateurs de composition pour nous rapprocher de l’effet que l’on souhaite obtenir au niveau de l’articulation : union, voire gonflement au contact pour une articulation entre deux phalanges, mélange d’abord, puis écrasement au contact pour un coude. En position initiale, nous stockons la valeur de la fonction potentiel en chaque point du maillage. Lors de l’animation, nous appliquons d’abord une déformation par SSD classique sur le maillage, puis chaque sommet est projeté sur sa valeur de potentiel en faisant un suivi de gradient dans la fonction potentiel. La piste que l’on souhaite faire évoluer est le guidage de la déformation des maillages par des fonctions potentiel dans les situations où les surfaces implicites sont performantes : déformation avec gestion automatique de la topologie, écrasement au contact, mélange, etc.

A moyen terme, mon objectif est d’évaluer l’intérêt pratique des surfaces implicites avec les moyens actuels et d’essayer, si les résultats suivent, de raviver l’intérêt de la communauté autour de leur développement et leur utilisation.

Si nous regardons ces travaux de notre point de vue global, nous pouvons voir qu’ils améliorent l’efficacité de la modélisation par surfaces implicites, sans ajouter de complication pour l’utilisateur. Au contraire, dans un contexte défini, les résultats obtenus lors de la composition sont plus naturels et donc plus proches des attentes de l’utilisateur. De plus, de composition en composition, nos opérateurs maintiennent un mélange au comportement homogène tout en supportant l’ajout de petits détails. Nous avons ainsi apporté une brique supplémentaire à notre édifice. L’implantation de nos nouveaux opérateurs n’est pas non plus très difficile et n’est pas sujète

à des instabilités numériques, ce qui est important pour garder la simplicité et la robustesse d'implantation nécessaire au développement de petits systèmes de modélisation.

6

Conclusion

... Nous pouvons ainsi constater qu'il y a de la place pour des propositions innovantes et plus efficaces, mais nous pouvons aussi constater que dès, que le logiciel veut répondre aux multiples besoins exprimés par les utilisateurs, l'interface s'enrichit, le nombre d'outils aussi et le logiciel devient très complexe (figure 2.1).

Dans ce contexte, nous allons nous intéresser aux problèmes suivants :

- Comment permettre un accès plus simple à la modélisation d'objets tridimensionnels pour un utilisateur non-expert ?
 - Comment démocratiser le développement de logiciels de modélisation d'objets tridimensionnels ?
-

A travers ce document nous venons de suivre les travaux que j'ai effectués et encadrés sur les modèles de représentation de surfaces tridimensionnelles durant ces neuf dernières années. Ces travaux ont été présentés et discutés dans le contexte défini par les deux questions posées ci-dessus.

Nous avons commencé par des études, menées entre les années 2002 et 2004, traitant des surfaces de subdivision, une représentation standard en modélisation géométrique de formes complexes. Malgré toutes les bonnes propriétés que ces surfaces présentent, nous avons vu que des irrégularités incontrôlables par l'utilisateur sont introduites dans les maillages. Des solutions pour améliorer le comportement du maillage au cours de la subdivision ont été proposées et nous avons vu des travaux dont les objectifs sont de poser les problèmes et de diffuser les connaissances fondamentales nécessaires à leur étude. Plus généralement, les surfaces de subdivision ont probablement le potentiel d'être une représentation de surfaces à partir de laquelle nous pourrions répondre aux besoins posés par notre première question, mais elles sont difficiles à implanter de façon stable dès lors que l'on veut permettre les changements de topologie ou que l'on souhaite proposer un processus de modélisation par ajout de parties à un objet.

La modélisation par esquisses est une approche qui est reconnue pour apporter une réponse pertinente à notre première question. En travaillant sur ce thème de 2004 à 2008, j'ai pu constater qu'effectivement, la prise en main des logiciels de modélisation par esquisses est rapide et que l'utilisation est très simple et intuitive. Le constat est aussi que les objets produits restent simples et qu'il est difficile de leur ajouter des détails. Nous avons vu quelques contributions

pour améliorer la qualité des surfaces tridimensionnelles reconstruites à partir des dessins bidimensionnels de l'utilisateur. Nous avons aussi vu comment utiliser le zoom comme information pertinente pour adapter automatiquement le niveau de détail de la reconstruction de l'objet à la taille des parties modélisées. Nous avons aussi pu constater que, pour répondre à notre deuxième question, il était intéressant d'utiliser des modèles de représentation volumiques (surfaces implicites) pour définir les objets tridimensionnels. Ceci nous a amené à nous replonger dans le développement des modèles pour les surfaces implicites.

Sur la période 2008-2010, les besoins que nous avons étudiés portent sur les opérateurs de composition (union, intersection et différence avec ou sans mélange) pour les surfaces implicites. Ces opérateurs sont robustes et relativement simples à implémenter, et ils permettent de modéliser des objets de façon itérative en ajoutant petit à petit des parties à un objet. Ces travaux sur les surfaces implicites sont repris à un moment où ces représentations sont laissées de côté, considérées par une majorité de la communauté graphique comme trop coûteuses et limitées. Pour illustrer ce propos, je citerai un commentaire d'un rapporteur anonyme sur notre soumission [GBC⁺11], qui résume très bien la situation :

“... In contrast to other reviewers, I don't have a problem with a paper like this appearing at Siggraph even if we haven't seen implicit modeling papers in a while ... the Siggraph audience might need to be reminded that implicit modeling still exists and has something to offer...”.

Nous avons vérifié que la puissance de calcul actuelle est suffisante pour permettre une visualisation efficace de ces surfaces, et en utilisant les premiers résultats théoriques que j'ai obtenus en 2001-2002 ainsi que les compétences combinées d'experts dans le domaine des surfaces implicites (Marie-Paule Cani et Brian Wyvill), nous avons proposé une nouvelle génération d'opérateurs de composition. Ces opérateurs résolvent plusieurs problèmes limitant l'aspect pratique des surfaces implicites, qui avaient été identifiés au cours des années quatre-vingt-dix - deux mille, mais jamais traités dans leur globalité. Nous comptons sur ces opérateurs pour nous permettre de générer des détails fins sur les surfaces et, comme nous l'avons évoqué dans les discussions du chapitre 5, d'utiliser les surfaces implicites comme support de déformation pour les maillages dans des applications de skinning par exemple.

Plus généralement, l'association des maillages et des modèles volumiques, si nous arrivons à la rendre efficace, doit nous permettre de profiter de la simplicité d'implantation et de la robustesse de la composition des fonctions potentiel pour augmenter l'accessibilité et la mise en œuvre de certaines opérations sur les maillages. Dans ce sens, un projet débuté l'année dernière et mené par Marion Dunyach (qui est actuellement en première année de thèse) en collaboration avec Mario Botsch, porte sur l'utilisation des modèles volumiques dans la déformation de maillages avec changement de topologie. J'espère arriver assez rapidement à mieux comprendre où se situent les difficultés, les avantages, les inconvénients, les verrous et les pistes pertinentes pour faire cohabiter maillages et représentations volumiques. Il me semble que c'est un complément indispensable à mener en parallèle au développement propre des modèles volumiques, ne serait-ce que pour les nombreux avantages que ces représentations peuvent théoriquement s'apporter.

Pour finir, je serais ravi que d'ici quelques années nous arrivions à donner raison à ce rapporteur anonyme en pouvant affirmer :

“Implicit modeling exists and has many things to offer”.

Ainsi, à court terme, je vais continuer à travailler sur le skinning de maillage guidé par des fonctions potentiel ainsi que sur la déformation de maillage avec changement de topologie (les collisions étant traitées avec des fonctions potentiel). A moyen terme, je pense continuer,

d'une part sur le développement d'opérateurs de composition n-aires pour les fonctions potentiel, et d'autre part sur l'extension du skinning à une peau implicite qui permettrait de gérer non seulement les déformations du maillage au niveau des articulations, mais aussi les collisions des membres (deux bouts de doigts par exemple) et l'effet d'écrasement au contact. A ce niveau là des recherches, je devrais avoir acquis aussi bien une expertise forte en génération de modèles n-aire, avec un espoir d'arriver à une compréhension théorique aussi importante que pour les opérateurs binaires, qu'une expertise sur la déformation de maillages guidée par des fonctions potentiel. La projection à long terme est délicate et les projets dépendront de la qualité des résultats que nous aurons obtenus.

- Soit nous saurons que les surfaces implicites sont un domaine intellectuellement intéressant, mais resteront contraintes dans leur utilisation, du moins par rapport à nos propositions. Dans ce cas, je pense orienter mes recherches vers l'étude de systèmes intuitifs pour la réalisation de prototypes d'objets à topologie hautement complexe.
- Soit nous aurons proposé un ensemble de contributions suffisamment efficaces qui ouvriront des possibilités variées parmi lesquelles je peux imaginer : la modélisation d'objets à topologie très complexe (comme ceux présentés figure 3.9) à base de surfaces implicites, la visualisation interactive et incrémentale de surfaces implicites à partir de maillages dynamiques suivant les modifications des fonctions potentiel lors des compositions (reprenant l'idée de maillage par graines et éventuellement, localement par systèmes de particules), la production d'outils de modélisation et d'animation simples et intuitifs, l'étude d'outils robustes et efficaces d'édition de maillages basés sur des fonctions potentiel.

Bibliographie

- [AA03] A. Adamson and M. Alexa. Approximating and intersecting surfaces from points. In *Proceedings of the Eurographics Symposium on Geometry Processing*, pages 230–239, 2003.
- [ABCG05] A. Alexe, L. Barthe, M.P. Cani, and V. Gaildrat. Shape modelling by sketching using convolution surfaces. In *Pacific Graphics (Short Papers)*, 2005.
- [AC00] E. Akleman and J. Chen. Guaranteeing 2-manifold property for meshes by using doubly linked face list. *International Journal of Shape Modeling*, 5(2) :149–177, 2000.
- [AC06] E. Akleman and J. Chen. Regular meshes construction algorithms using regular handles. In *Proceedings of Shape Modeling International*, pages 27–35, 2006.
- [ACS00] E. Akleman, J. Chen, and V. Srinivasan. A new paradigm for changing topology during subdivision modeling. In *Proceedings of Pacific Graphics*, pages 149–177, 2000.
- [ADS06] U.H. Augsdörfer, N.A. Dodgson, and M.A. Sabin. Tuning subdivision by minimising gaussian curvature variation near extraordinary vertices. *Proc. of Eurographics, Computer Graphics Forum*, 25(3) :263–272, 2006.
- [ADS11a] U.H. Augsdörfer, N.A. Dodgson, and M.A. Sabin. Artifact analysis on b-splines, box-splines and other surfaces defined by quadrilateral polyhedra. *Computer Aided Geometric Design*, 28(3) :177–197, 2011.
- [ADS11b] U.H. Augsdörfer, N.A. Dodgson, and M.A. Sabin. Artifact analysis on triangular box-splines and subdivision surfaces defined by triangular polyhedra. *Computer Aided Geometric Design*, 28(3) :198–211, 2011.
- [AGB04] A. Alexe, V. Gaildrat, and L. Barthe. Interactive modelling from sketches using spherical implicit functions. In *AFRIGRAPH 2004*, pages 25–34, 2004.
- [AHKS94] M. Akeo, H. Hashimoto, T. Kobayashi, and T. Shibusawa. Computer graphics system for reproducing three-dimensional shape from idea sketch. *Eurographics '94 Proceedings*, 13(3) :477–488, 1994.
- [AJC02] Alexis Angelidis, Pauline Jepp, and Marie-Paule Cani. Implicit Modeling with Skeleton Curves : Controlled Blending in Contact Situations. In *International Conference on Shape Modeling*, pages 137–144, 2002.
- [AK04] N. Amenta and Y. Kil. Defining point-set surfaces. *ACM Transaction on Graphics, Proc. of SIGGRAPH 2004*, 23(3) :264–270, 2004.
- [ALSS03] P. Alliez, N. Laurent, H. Sanson, and F. Schmitt. Efficient view-dependent refinement of 3d meshes using $\sqrt{3}$ -subdivision. *the Visual Computer*, 18(4) :205 – 221, 2003.

- [ASC03] E. Akleman, V. Srinivasan, and J. Chen. Interactive rind modeling. In *Proceedings of Shape Modeling International*, pages 23–30, 2003.
- [Aur91] F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. In *ACM Computing Surveys*, volume 23, pages 345–405, 1991.
- [Bar81] A.H. Barr. Superquadrics and angle-preserving transformations. *IEEE Comput. Graph. Appl.*, 1 :11–23, January 1981.
- [Bar07] L. Barthe. Courbes et surfaces de subdivision. In D. Bechmann and B. Peroche, editors, *Informatique graphique, modélisation géométrique et animation*, pages 135–162. Hermes, 2007.
- [BBCW10] A. Bernhardt, L. Barthe, M-P. Cani, and B. Wyvill. Implicit blending revisited. *Proc. of Eurographics, Computer Graphics Forum*, 29(2) :367–376, 2010.
- [BDS⁺03] L. Barthe, N.A. Dodgson, M.A. Sabin, B. Wyvill, and V. Gaildrat. Two-dimensional potential fields for advanced implicit modeling operators. *Computer Graphics Forum*, 22(1) :23–33, 2003.
- [BGSK05] L. Barthe, C. Gerot, M.A. Sabin, and L. Kobbelt. Simple computation of the eigencomponents of a subdivision matrix in the fourier domain. In N. Dodgson, M.S. Floater, and M.A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, pages 245–257. Springer-Verlag, 2005.
- [BHSF09] M. Bein, S. Havemann, A. Stork, and D. Fellner. Sketching subdivision surfaces. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, SBIM '09, pages 61–68, New York, NY, USA, 2009. ACM.
- [BK03] L. Barthe and L. Kobbelt. Direct computation of a control vertex position on any subdivision level. In M.J. Wilson and R. Martin, editors, *Mathematics of Surfaces X*, pages 40–47. Springer, 2003.
- [BK04] L. Barthe and L. Kobbelt. Subdivision scheme tuning around extraordinary vertices. *Computer Aided Geometric Design*, 21(6) :561–583, 2004.
- [Bli82] J.F. Blinn. A generalisation of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3) :235–256, 1982.
- [Blo94] J. Bloomenthal. An implicit surface polygonizer. pages 324–349, 1994.
- [Blo97a] J. Bloomenthal. Bulge elimination in convolution surfaces. In *Computer Graphics Forum*, volume 16, pages 31–41, 1997.
- [Blo97b] J. Bloomenthal, editor. *Introduction to Implicit Surfaces*. Morgan Kaufmann, July 1997.
- [BLZ00] H. Biermann, A. Levin, and D. Zorin. Piecewise smooth subdivision surfaces with normal control. In *Proceedings of SIGGRAPH 2000, ACM*, pages 113–120, 2000.
- [BMBZ02] H. Biermann, I. Martin, F. Bernardini, and D. Zorin. Cut-and-paste editing of multiresolution surfaces. In *Proceedings of SIGGRAPH 2002, ACM*, pages 312–321, 2002.
- [BMS⁺10] E.Vital Brazil, I. Macedo, M. Costa Sousa, L.H. de Figueiredo, and L. Velho. Sketching variational hermite-rbf implicits. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, SBIM '10, pages 1–8, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

-
- [BPCB08] A. Bernhardt, A. Pihuit, M. P. Cani, and L. Barthe. Matisse : Painting 2d regions for modeling free-form shapes. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*, pages 57–64, 2008.
- [BS88] A.A. Ball and D.J.T. Storry. Conditions for tangent plane continuity over recursively generated b-spline surfaces. *ACM Transactions on Graphics*, 7(2) :183–102, 1988.
- [BS91] J. Bloomenthal and K. Shoemake. Convolution Surfaces. In *Computer Graphics (Proc. SIGGRAPH 91)*, volume 25, pages 251–256, August 1991.
- [BS96] C. Blanc and C. Schlick. Ratioquadrics : an alternative model for superquadrics. 12(8) :420–428, 1996.
- [BTC95] E. Bittar, N. Tsingos, and M.P. Cani. Reconstruction of unstructured 3d data : Combining a medial axis and implicit surfaces. *Computer Graphics Forum*, 14(3) :457–468, 1995.
- [BWG04] L. Barthe, B. Wyvill, and E. De Groot. Controllable binary csg operators for "soft objects". *International Journal of Shape Modeling*, 10(2) :135–154, 2004.
- [CAD09] T.J. Cashman, U.H. Augsdörfer, N.A. Dodgson, and M.A. Sabin. Nurbs with extraordinary points : High-degree, non-uniform, rational subdivision schemes. In *ACM Transactions on Graphics, Proc. of SIGGRAPH 2009*, volume 28, pages 1–9, 2009.
- [Can93] Marie-Paule Cani. An implicit formulation for precise contact modeling between flexible solids. In *20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1993*, pages 313–320, 1993. Published as Marie-Paule Gascuel.
- [CBC⁺01] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001*, pages 67–76, 2001.
- [CBS96] B. Crespín, C. Blanc, and C. Schlick. Implicit sweep objects. *Proc. of Eurographics, Computer Graphics Forum*, 15(3) :165–174, 1996.
- [CC78] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6) :350–355, 1978.
- [DAC09] N.A. Dodgson, U.H. Augsdörfer, T.J. Cashman, and M.A. Sabin. Deriving box-spline subdivision schemes. In E.R. Hancock, R.R. Martin, and M.A. Sabin, editors, *Mathematics of Surfaces XIII*, pages 106–123. Springer-Verlag, 2009.
- [dBHR94] C. de Boor, D. Hollig, and S. Riemenschneider. *Box Splines*. Springer-Verlag, New York, 1994.
- [DC95] M. Desbrun and M.P. Cani. Animating soft substances with implicit surfaces. In *22nd annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1995*, volume 29, pages 287–290, 1995. Published as Marie-Paule Gascuel.
- [dGWvdW09] E. de Groot, B. Wyvill, and H. van de Wetering. Locally restricted blending of blobtrees. *Computers And Graphics*, 33(6) :690–697, 2009.
- [DIS03] N.A. Dodgson, I.P. Ivrišimtzis, and M.A. Sabin. Subdivision of box-splines. In A. Cohen, J.L. Merrien, and L.L. Schumaker, editors, *Curve and Surface Fitting : Saint-Malo 2002*, pages 119–128. Nashboro Press, 2003.

- [DLG90] N. Dyn, D. Levin, and J. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transaction on Graphics*, 9(2) :160–169, 1990.
- [DS78] D. Doo and M.A. Sabin. Analysis of the behaviour of recursive subdivision surfaces near extraordinary points. *Computer Aided Design*, 10(6) :356–360, 1978.
- [Duf92] T. Duff. Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, pages 131–138, 1992.
- [Dyn02] N. Dyn. Analysis of convergence and smoothness by the formalism of laurent polynomials. In A. Iske, E. Quak, and M.S. Floater, editors, *Tutorials on multi-resolution in geometric modelling*, pages 51–68. Springer, 2002.
- [EGB09] M. Eyiyurekli, C. Grimm, and D. Breen. Editing level-set models with sketched curves. In *Proc. of Eurographics/ACM Symposium on Sketch-Based Interfaces and Modeling Tutorials on multiresolution in geometric modelling*, pages 45–52, 2009.
- [Far02] G. Farin. *Curves and surfaces for CAGD : a practical guide*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition, 2002.
- [FSSV06] J. Flórez, M. Sbert, M.A. Sainz, and J. Vehí. Improving the Interval Ray Tracing of Implicit Surfaces. In *Computer Graphics International 2006*, pages 655–664, 2006.
- [GA00] E. Galin and S. Akkouche. Incremental polygonization of implicit surfaces. *Graphical Models*, 62 :19–39, 2000.
- [GBC⁺11] O. Gourmel, L. Barthe, M.P. Cani, B. Wyvill, A. Bernhardt, M. Paulin, and H. Grasberger. Gradient-based implicit modeling. *Recommended for publication by the Siggraph 2011 committee, with modifications, at Transactions on Graphics*, 2011.
- [GBDS05] C. Gerot, L. Barthe, N.A. Dodgson, and M.A. Sabin. Subdivision as a sequence of sampled cp surfaces. In N. Dodgson, M.S. Floater, and M.A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, pages 259–270. Springer-Verlag, 2005.
- [GBP04] G. Guennebaud, L. Barthe, and M. Paulin. Deferred splatting. *Proc. of Eurographics, Computer Graphics Forum*, 23(3) :653–660, 2004.
- [GBP05] G. Guennebaud, L. Barthe, and M. Paulin. Interpolatory refinement for real-time processing of point-based geometry. *Proc. of Eurographics, Computer Graphics Forum*, 24(3) :657–666, 2005.
- [GG07] G. Guennebaud and M. Gross. Algebraic point set surfaces. *ACM Transaction on Graphics, Proc. of SIGGRAPH 2007*, 26(3) :231–239, 2007.
- [GKSS02] I. Guskov, A. Khodakovsky, P. Schröder, and W. Sweldens. Hybrid meshes : multiresolution using regular and irregular refinement. In *Proceedings of the Symposium on Computational Geometry*, pages 264–272, 2002.
- [Gri99] C.M. Grimm. Implicit generalized cylinders using profile curves. In *Proc. of Implicit Surfaces'99*, 1999.
- [GVSS00] I. Guskov, K. Vidimce, W. Sweldens, and P. Schröder. Normal meshes. In *Proceedings of SIGGRAPH 2000, ACM*, pages 95–102, 2000.

-
- [GW95] A. Guy and B. Wyvill. Controlled blending for implicit surfaces using a graph. In *Proc. of Implicit Surfaces'95*, pages 107–112, 1995.
- [Hal89] R. W. Hall. Fast parallel thinning algorithms : parallel speed and connectivity preservation. *Commun. ACM*, 32(1) :124–131, 1989.
- [HDDH94] H. Hoppe, T. DeRose, T. Duchamp, and M. Halstead. Piecewise smooth surfaces reconstruction. In *Proceedings of SIGGRAPH 1994, ACM*, pages 295–302, 1994.
- [Hop97] H. Hoppe. View-dependent refinement of progressive meshes. In *Proceedings of SIGGRAPH 1997, ACM*, pages 189–198, 1997.
- [Hop98] H. Hoppe. Smooth view-dependent level-of-detail control and its application in terrain rendering. In *IEEE Visualization*, pages 35–42, 1998.
- [IH03] T. Igarashi and J.F. Hughes. Smooth meshes for sketch-based freeform modeling. In *In Symposium on Interactive 3D Graphics*, pages 139–142, 2003.
- [IMT99] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy : A sketching interface for 3d freeform design. In *Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1999*, pages 409–416, 1999.
- [Isk02] A. Iske. Scattered data modelling using radial basis functions. In A. Iske, E. Quak, and M.S. Floater, editors, *Tutorial on multiresolution in geometric modelling*, pages 205–242. Springer-Verlag, 2002.
- [JS09] D. Jiang and N.F. Stewart. Numerical validation in current hardware architectures. chapter Robustness of Boolean Operations on Subdivision-Surface Models, pages 161–174. Springer-Verlag, Berlin, Heidelberg, 2009.
- [KCB09] P. Kraemer, D. Cazier, and D. Bechmann. Extension of half-edges for the representation of multiresolution subdivision surfaces. *The Visual Computer*, 25(2) :149–163, 2009.
- [KHR02] O. Karpenko, J.F. Hughes, and R. Raskar. Free-form sketching with variational implicit surfaces. *Proc. of Eurographics 2002*, 21 :585–594, 2002.
- [Kob96a] L. Kobbelt. Interpolatory subdivision on open quadrilateral nets with arbitrary topology. *Computer Graphics Forum*, 15(3) :409–420, 1996.
- [Kob96b] L. Kobbelt. A variational approach to subdivision. *Computer Aided Geometric Design*, 13 :743–761, 1996.
- [Kob00] L. Kobbelt. $\sqrt{3}$ -subdivision. In *Proceedings of SIGGRAPH 2000, ACM*, pages 103–112, 2000.
- [KPR04] K. Karciauskas, J. Peters, and U. Reif. Shape characterisation of subdivision surfaces - case studies. *Computer Aided Geometric Design*, 21(6) :601–614, 2004.
- [Lev06] A. Levin. Modified subdivision surfaces with continuous curvature. *ACM Transaction on Graphics, Proc. of SIGGRAPH 2006*, 25 :1035–1040, 2006.
- [LLS01] N. Litke, A. Levinand, and P. Schröder. Trimming for subdivision surfaces. *Computer Aided Geometric Design*, 18(5) :463–481, 2001.
- [Loo87] C. Loop. *Smooth subdivision surfaces based on triangles*. Master’s thesis, University of Utah, 1987.
- [LS08] C. Loop and S. Schaefer. G2 tensor product splines over extraordinary vertices. In *Proc. of the Symposium on Geometry Processing*, pages 1373–1382, 2008.

- [LY03] P. Ljung and A. Ynnerman. Extraction of intersection curves from iso-surfaces on co-located 3d grids. In *SIGRAD2003 Proceedings, ISSN 1650-3686*, pages 1650–3740. Linköping Electronic Conference Proceedings, 2003.
- [MGV09] I. Macêdo, J.P. Gois, and L. Velho. Hermite interpolation of implicit surfaces with radial basis functions. In *roc. of XXII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI '09)*, pages 1–8, 2009.
- [MK06] M. Marinov and L. Kobbelt. A robust two-step procedure for quad-dominant remeshing. *Computer Graphics Forum*, 25(3) :537–546, 2006.
- [Mur91] S. Muraki. Volumetric shape description of range data using blobby model. *Computer Graphics*, 25(4) :227–235, 1991.
- [OF02] S.J. Osher and R.P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 1 edition, October 2002.
- [OGP10] M. Paulin L. Barthe O. Gourmel, A. Pajot and P. Poulin. Fitted bvh for fast raytracing of metaballs. *Proc. of Eurographics, Computer Graphics Forum*, 29(2) :281–288, 2010.
- [ONNI03] S. Owada, F. Nielsen, K. Nakazawa, and T.A. Igarashi. Sketching interface for modeling the internal structures of 3d shapes. In *Proceedings of 3rd International Symposium on Smart Graphics*, pages 49–57, 2003.
- [PASS95] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko. Function representation in geometric modeling : concepts, implementation and applications. *The Visual Computer*, 11(8) :429–446, 1995.
- [Pet02] J. Peters. c^2 free-form surfaces of degree 3, 5. *Computer Aided Geometric Design*, 19(2) :113–126, 2002.
- [PPK05] G.I. Pasko, A.A. Pasko, and T.L. Kunii. Bounded blending for Function-Based shape modeling. *IEEE Comput. Graph. Appl.*, 25(2) :36–45, 2005.
- [PR98] J. Peters and U. Reif. Analysis of algorithms generalizing b-spline subdivision. *SIAM Journal of Numerical Analysis*, 35(2) :728–748, 1998.
- [Pra97] L. Prasad. Morphological analysis of shapes. In *CNLS Newsletter*, volume 139, pages 1–18, 1997.
- [Pra98] H. Prautzsch. Smoothness of subdivision surfaces at extraordinary points. *Adv. Comp. Math.*, 14 :377–390, 1998.
- [PS04] J. Peters and L.J. Shiue. Combining 4- and 3-direction subdivision. *ACM Transaction on Graphics*, 23(4) :980–1003, 2004.
- [Pug92] D. Pugh. Designing solid objects using interactive sketch interpretation. *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, 25(2) :117–126, 1992.
- [Rei95] U. Reif. A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Design*, 12 :153–174, 1995.
- [Ric73] A. Ricci. Constructive Geometry for Computer Graphics. *computer journal*, 16(2) :157–160, May 1973.
- [Roc89] A.P. Rockwood. The Displacement Method for Implicit Blending Surfaces in Solid Models. *CM Transaction on Graphics*, 8(4) :279–297, 1989.
- [Ron00] A. Ron. In *Private communication*, 2000.

-
- [Rva63] V.L. Rvachev. On the analytical description of some geometric objects. *Reports of Ukrainian Academy of Sciences - published in Russian*, 153(4) :765–767, 1963.
- [Sab68] M-A Sabin. The use of potential surfaces for numerical geometry. In *Tech. Report VTO/MS/153, British Aerospace Corp., Weybridge, U.K.*, 1968.
- [SB02] M.A. Sabin and L. Barthe. Artifacts in recursive subdivision surfaces. *Curve and Surface Fitting : St Malo 2002*, editors A. Cohen, J.L. Merrien and L.L. Schumaker, pages 353–362, 2002.
- [SdGWS08] M. Sugihara, E. de Groot, B. Wyvill, and R. Schmidt. A sketch-based method to control deformation in a skeletal implicit surface modeler. In *Proceedings of Sketch-Based Interfaces and Modeling Symposium, SBIM '08*, pages 65–72, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.
- [SGW06] R. Schmidt, C. Grimm, and B. Wyvill. Interactive decal compositing with discrete exponential maps. *ACM Transaction on Graphics, Proc. of SIGGRAPH 2006*, 25(3) :605–613, 2006.
- [Sha] ShapeShop. <http://www.shapeshop3d.com/>.
- [She99] Andrei Sherstyuk. Kernel functions in convolution surfaces : a comparative analysis. *The Visual Computer*, 15(4) :171–182, 1999.
- [Sny92] J.M. Snyder. Interval analysis for computer graphics. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, pages 121–130, 1992.
- [SS08] R. Schmidt and K. Singh. Sketch-based procedural surface modeling and compositing using surface trees. *Proc. of Eurographics, Computer Graphics Forum*, 27(2) :321–330, 2008.
- [SSB08] R. Schmidt, K. Singh, and R. Balakrishnan. Sketching and composing widgets for 3d manipulation. *Proc. of Eurographics, Computer Graphics Forum*, 27(2) :301–310, 2008.
- [Sta98] J. Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *Proceedings of SIGGRAPH 1998*, pages 395–404, 1998.
- [Sto96] N. Stolte. *Espaces discrets de haute résolutions : une nouvelle approche pour la modélisation et la synthèse d'images réalistes*. Phd thesis report, University Paul Sabatier of Toulouse, 1996.
- [SWS10] M. Sugihara, B. Wyvill, and R. Schmidt. Warpcurves : A tool for explicit manipulation of implicit surfaces. *Computers And Graphics, proc. of Shape Modeling International 2010*, 34(3) :282–291, 2010.
- [TO99] G. Turk and J.F. O'Brien. Shape transformation using variational implicit functions. In *Proceedings of SIGGRAPH 1999*, page 335–342, 1999.
- [Top] TopMod. <http://www.viz.tamu.edu/faculty/ergun/research/topology/download.html>.
- [TZF04] C.L. Tai, H. Zhang, and C.K. Fong. Prototype modeling from sketched silhouettes based on convolution surfaces. *Computer Graphics Forum*, 23(1) :71–83, 2004.
- [WFJ+05] B. Wyvill, K. Foster, P. Jepp, R. Schmidt, M. Costa Sousa, and J.A. Jorge. Sketch based construction and rendering of implicit models. In *Eurographics Workshop on Computational Aesthetics in Graphics, Visualization and Imaging*, pages 67–74, 2005.

- [WGG99] B. Wyvill, A. Guy, and E. Galin. Extending the CSG Tree : Warping, Blending and Boolean Operations in an Implicit Surface Modeling System. *Computer Graphics Forum*, 18(2) :149–158, 1999.
- [WMW86] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data Structure for Soft Objects. *The Visual Computer*, 2(4) :227–234, February 1986.
- [WW02] J. Warren and H. Weimer. *Subdivision methods for geometric design : a constructive approach*. San Fransisco : Morgan Kaufman, 2002.
- [ZHH96] R. Zeleznik, K. Herndon, and J. Hughes. Sketch : An interface for sketching 3d scenes. In *Proceedings of SIGGRAPH 1996*, pages 163–170, 1996.
- [ZS03] R. Zenka and P. Slavik. New dimensions for sketches. In *Proceedings of the 19th spring conference on Computer graphics*, pages 157–163, 2003.
- [ZSS96] D. Zorin, P. Schröder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *Proceedings of SIGGRAPH 1996*, pages 189–192, 1996.
- [ZSS97] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1997*, pages 259–268, 1997.

7

Publications

7.1 Surfaces de subdivision

7.1.1 Artifacts in recursive subdivision surfaces.

Auteurs : Malcolm A. Sabin, Loïc Barthe
Revue : Curve and Surface Fitting : St Malo 2002
Date : Juin 2003
ISBN : 0-9728482-1-5

Artifacts in Recursive Subdivision Surfaces

Malcolm Sabin and Loïc Barthe

Abstract. For a subdivision scheme to be effective it has to be possible for designers to provide relatively sparse data and achieve something close to their mental images. The difference between what is expected (or hoped for) from the subdivision scheme, and what actually emerges as a limit surface is an **artifact**. This aspect has not been much studied, and this paper provides an initial categorisation of five issues and identifies how much and how little we understand them. It is assumed that the reader is already familiar with eigenanalysis and z -transform methods for analysis of the limit surface.

§1. What Kinds of Artifacts Are There ?

We define an artifact to be any feature of the limit surface which cannot be controlled by the movement of control points at the current level of subdivision. This implies that the concept of spatial frequency is a key one. Spatial frequency components of a frequency greater than one cycle per two control points cannot be controlled, and so mechanisms which give them can be confidently called artifacts.

Preliminary work has identified two aspects of curve artifacts and five of surface artifacts which we can address:

- longitudinal artifacts on curves
- end-conditions on curves
- longitudinal artifacts on surfaces
- edge- and corner-conditions
- lateral artifacts
- radial artifacts
- rotational artifacts

The rest of this paper describes current knowledge, which is fairly complete, relating to the first five of these. The last two, although described here, will be addressed in much more detail in future papers.

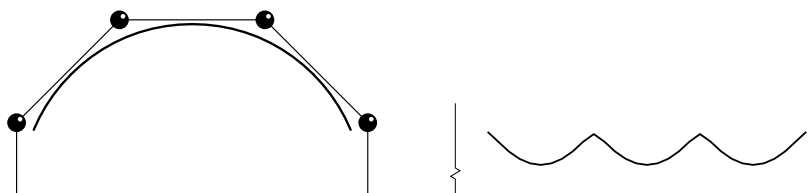


Fig. 1. (a) shows a piece of a cubic B-spline where the control points are at the vertices of a regular octagon, and (b) its curvature plot (curvature against distance). Although the curve looks smooth, the curvature plot shows up the higher frequency components. Note the broken vertical axis in the curvature plot; the peak to peak amplitude of the variation is about 0.1% of the average curvature.

§2. Spatial Frequencies

The Shannon-Whittaker theorem [15] which gives us the minimal number of samples necessary to exactly reconstruct a signal in terms of the extrema of its frequency components is just as applicable when the abscissa is a spatial dimension as when it is time.

The concept is already familiar in computer graphics under the name ‘aliasing’. The subtlety that we need for surface work is that spatial frequency has not just a magnitude but a direction. If we regard the surface as being expressed as a set of components

$$P(u, v) = \sum_i A_i \cos(a_i u + b_i v + c_i)$$

the A_i are the amplitudes, $[a_i, b_i]$ the frequencies, and c_i the phases of the various terms.

Where high-frequency artifacts appear, they are not necessarily in the same direction as the dominant low frequency variations in the data.

§3. Longitudinal Artifacts on Curves

Longitudinal artifacts occur when smoothly positioned control points give a limit curve which has spatial frequency components at a frequency higher than the Shannon limit of one cycle per two samples. For B-spline curves this is always, because the Fourier transform of the B-spline basis function is not band-limited. The exercise is one of damage limitation, but fortunately the situation is not too bad quantitatively. Indeed it is often necessary to look at curvature plots before it is visible at all.

We can analyse this either by Fourier methods, by looking at the amplitude of one-cycle-per-control point frequency for data sets generated with n points round a circle (this can be measured easily by evaluating the limit curve near the control points and half-way in between them), or by z -transforms.

| points per cycle | degrees per point | spline degree | | | |
|---------------------|----------------------|---------------|----------|----------|----------|
| | | 2 | 3 | 4 | 5 |
| 3 | 120 | 0.111111 | 0.066667 | 0.030303 | 0.015873 |
| 4 | 90 | 0.029437 | 0.014059 | 0.003799 | 0.001437 |
| 5 | 72 | 0.011146 | 0.004726 | 0.000849 | 0.000266 |
| 6 | 60 | 0.005155 | 0.002040 | 0.000261 | 0.000073 |
| 8 | 45 | 0.001565 | 0.000576 | 0.000043 | 0.000010 |
| 10 | 36 | 0.000629 | 0.000224 | 0.000011 | 0.000002 |
| 12 | 30 | 0.000300 | 0.000105 | 0.000004 | 0.000001 |
| 16 | 22.5 | 0.000094 | 0.000032 | 0.000001 | |
| 20 | 18 | 0.000038 | 0.000013 | | |
| 24 | 15 | 0.000018 | 0.000006 | | |
| 32 | 11.25 | 0.000006 | 0.000002 | | |
| 40 | 9 | 0.000002 | 0.000001 | | |
| 48 | 7.5 | 0.000001 | | | |

Tab. 1. The figures in the rightmost four columns are the amplitudes of longitudinal artifacts measured for B-splines of degrees 2 to 5, tabulated for different numbers of control points evenly spaced around a unit circle.

The z -transform approach says that this artifact varies inversely with the $(d + 1)^{\text{th}}$ power of the number of vertices per cycle, where $d + 1$ is the number of $1 + z$ factors in the symbol. Direct measurements of B-splines, tabulated in Table 1, suggest that the even degree B-splines in fact are better than this, giving variation with the $(d + 2)^{\text{th}}$ power. This is probably due to the fact that the z -transform picks up errors along the curve as well as errors across it, and these dominate for the even degrees.

Two specific points to note are

- that the number of points needed is very low if the piece of curve turns through only a small angle. Even for quadratics, the error is only 1 part in a million at one vertex per 7.5 degrees of turn;
- that for very sparse polygons, increasing the degree does not help much.

There is hope that subdivision may be able to outperform simple B-splines in terms of longitudinal artifacts, by using the concept of *geometric sensitivity*. It is certainly possible to define a variant of the four-point scheme [5] in which each new vertex is positioned in such a way that the 3-point curvature estimator gives the mean of the old curvature estimates at the adjacent old points. Such a scheme is trivially proven to have circular precision — if enough consecutive control points lie on a circle, the limit curve will contain an arc of that circle. Like the circular-precision

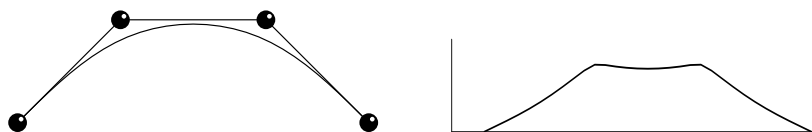


Fig. 2. The ‘natural’ end conditions have the effect of sending the curvature to zero at the ends. (a) shows a curve which uses this condition, and (b) its curvature plot. Note that the effect at the ends is much larger than the longitudinal variation in the interior of the curve.

scheme of Nasri and Farin [12] and unlike those of Dyn [4] and Warren [17], no prior information about radius or points/cycle is required.

§4. End-conditions on Curves

The usual end-condition applied to cubic subdivision curves can be described in three ways: as the extension of the control polygon by linear extrapolation of one extra control point at each end, as the modification of the rules near the end, or as the ad hoc appending of the original end control point after computation of the next polygon. All three give the same unsatisfactory result, which has zero curvature at the end points.

We need something more comparable to the (standard in practice) use of Bezier end-conditions, where an extra control point gives slope control at the end, and we retain all the convex hull properties. Unfortunately this amounts to using unequal intervals, and so does not fit into the subdivision world-view too easily.

The ‘not-a-knot’ conditions shown by Bejancu [1] to improve the approximation order in the semi-cardinal case look to be highly relevant. It is well known that this gives much better results in the interpolating spline context than the natural end conditions. A rather unsatisfactory modification of the cubic subdivision to give the not-a-knot condition is described by Nasri and Sabin [13]. Better formulations may be available, which give an effect similar to Bezier end-conditions for a spline in which the first knot interval is twice the normal, but this is still being explored.

§5. Longitudinal Artifacts on Surfaces

For quadrilateral grid schemes, these are just the tensor products of the curve artifacts. For triangle grid schemes, we can reduce the problem to a curve one by considering an initial polyhedron which is extruded in one of the mesh directions. This forces the spatial frequencies to be essentially univariate in a direction perpendicular to that of the extrusion.

What becomes interesting is what happens when the extrusion direction is not one of the mesh directions. This does not appear to have been considered at all yet. Clearly the approach of regarding the spatial frequency as a vector will be appropriate.

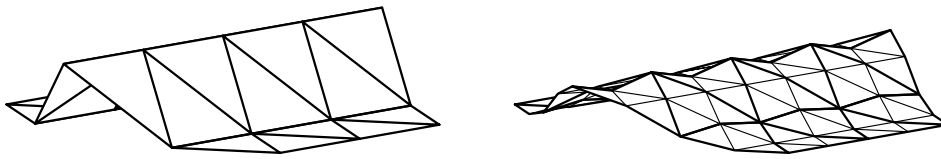


Fig. 3. (a) shows an extruded polyhedron. (b) shows the effect after one iteration. The 'dinosaur back' effect is clearly visible. There are spatial frequency components at twice the Shannon limit in the original direction of extrusion where there was no variation in the original at all.

§6. Edge- and Corner-conditions on Surfaces

In the quadrilateral grid case we merely use the tensor product structure. The main new difficulties with edge- and corner-conditions apply to schemes based on regular triangular grids, such as Loop and Butterfly [6].

For regular triangular grids there are two types of convex corner, and the two directions interact at either type of corner. Recent results indicate that it may still be possible to define 'not-a-knot' conditions on triangular grids, but the application of these to subdivision has not yet been carried out.

Levin [10] has described a scheme for controlling a triangular scheme in a way like Bezier edge-conditions, but the pattern of control points is not really intuitive, because additional vertices, interior to the surface, one per edge-vertex, are used to define the variation of tangent across the edge. Bejancu [1] has discovered that the not-a-knot condition can be applied to the 3-direction quartic box-spline (Loop) in the approximation context, and this may lead to a reasonable solution, although at present it looks as though achieving not-a-knot and the convex hull property at the same time may require changes in the rules over a region as wide as the support of the scheme. This could be unfortunate.

§7. Lateral Artifacts on Surfaces

These are an effect which has long been with us, unrecognised. It applies to the regular-grid parts of both subdivision and B-spline surfaces. While in a longitudinal artifact a spatial frequency in the original polyhedron gives an artifact component in the same direction, in lateral artifacts they cause an artifact component in a perpendicular direction.

This effect is fully understood. A lateral artifact will occur if the original polyhedron is extruded in a direction for which the symbol of the mask does not have a factor of $1 + z$. The example in Figure 3 was contrived by applying a version of the Loop scheme [11] whose mask was deliberately altered so that there was no $(1 + z)$ factor. (The mask is the set of influences of a given old vertex on the surrounding new ones.)

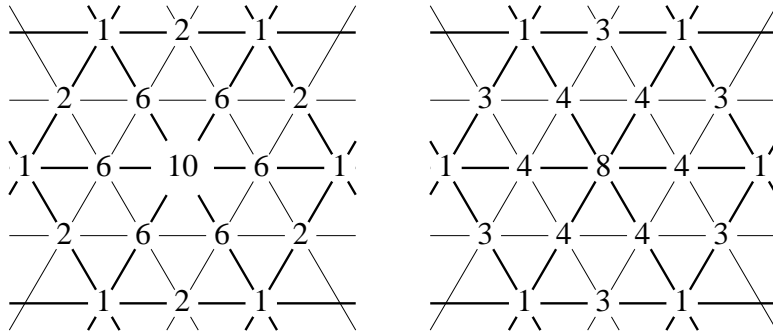


Fig. 4. The mask on the left is that of the standard Loop scheme, which has a $(1+z)^2$ factor in each mesh direction. That on the right has been altered so that although it still gives constant and linear precision, it has no $1+z$ factor in any direction.

This can be understood in the same breath as the fact that any attempt to run a feature skew to the mesh directions in a NURBS surface also causes undesired ripples. It is the same effect.

The effect of having a $(1+z)^2$ factor is that a cross-section which is linearly varying rather than merely constant gives no ripples. This is a visible difference between Simplest [14] and Velho [16].

This raises the question *'Is it better to have more directions in which $1+z$ factors exist, as in, for example the Simplest scheme, or Velho, or to have a higher power of $(1+z)$ in a few directions, as in Catmull-Clark?'*

For schemes of arity other than 2, the relevant factors are $(1-z^a)/(1-z)$ rather than just $(1+z)$. This condition is closely related to the condition for non-fractal support described by Ivriissimtziis [8], in that any scheme with fractal support, such as the $\sqrt{3}$ schemes described by Guskov [7] and by Kobbelt [9] will have no directions free from lateral artifacts.

The lateral artifact story is thus pretty well complete. It is clear that the scheme designer needs to ensure that the mask is well-endowed with directions i with $(1+z_i)$ factors in the symbol of the scheme, and the surface designer needs to run those directions of the mesh along the ridge features of the required surface. This is pretty natural anyway. It is merely necessary to suppress the urge to try to be clever. This is perhaps one of the strongest arguments for subdivision surfaces as distinct from B-splines, that the mesh can be run locally along features, with extraordinary points where necessary in relatively featureless regions.

§8. Radial Artifacts on Surfaces

The last two effects so far identified were demonstrated by Jos Stam at a Dagstuhl meeting [3]. His original polyhedron consisted of a 14-gon upper face and a 14-gon lower face, joined by 14 rectangles, and he applied a

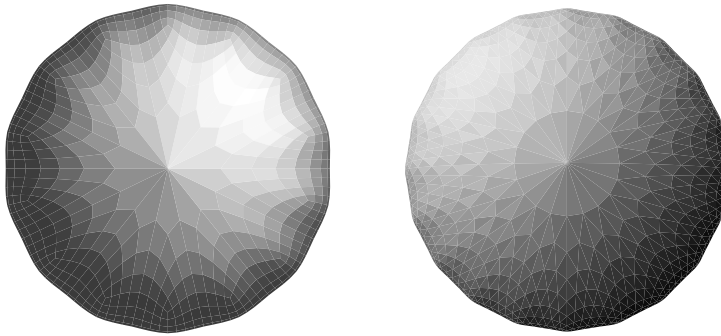


Fig. 5. The left figure shows the effect near the pole of the third iteration of the Catmull-Clark scheme on a regular 14-sided prism. The right figure shows that of the Loop scheme on an original polyhedron consisting of a vertex at each pole and 20 vertices around the equator, all joined up by 40 triangles. In both cases the facets near the pole are clearly much larger than those elsewhere. The number of vertices/faces round the equator was in each case chosen to show the artifacts most clearly.

few iterations of the standard Catmull-Clark [2] scheme. This showed two undesirable effects, long thin facets near the pole, and ridges running along lines of longitude. Very similar effects also occur in the Loop [11] scheme, and it appears likely that all schemes have the same potential problems related to extraordinary points. The question of continuity at such points is not the only important one.

Where the effect has spatial frequency components in directions radial to the extraordinary point, we call this a *radial artifact*; where it has spatial frequency components in directions around the extraordinary point, we call this a *rotational artifact*.

The specific radial artifact shown in Figure 5 is called the *polar artifact*, and is relatively easy to understand. What is happening is that while over the bulk of the shape each iteration halves the size of each triangle, near the extraordinary point the radial shrinkage factor is the value of the double eigenvalue λ associated with the natural configuration. If this is significantly larger than $1/2$, then after a few iterations the facets there are considerably larger than the average.

Interestingly, this is not a problem in the limit surface, although it gets worse at every iteration. It does mean that the rate of convergence is somewhat slower than quadratic, as was pointed out by Wang and Qin [18].

The polar artifact can be completely eliminated by adjusting the extraordinary point masks so that at every extraordinary point the second eigenvalue (λ) is the same as at ordinary vertices. Unfortunately the mask adjustments which give bounded curvature tend to increase λ .

We can also classify unbounded curvature at extraordinary points as a radial artifact.

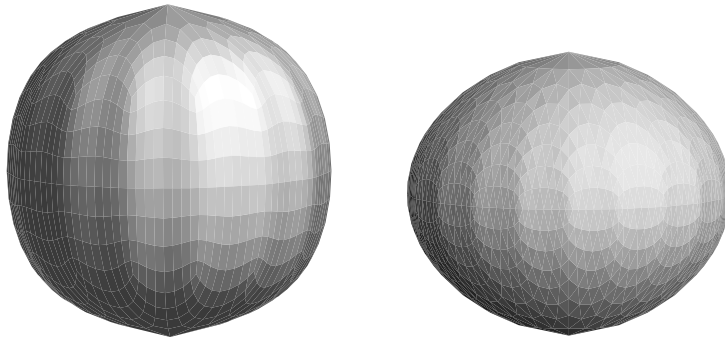


Fig. 6. The same data after 3 iterations of the two schemes (left Catmull Clark, right Loop) shows definite ridges. There is one ridge per original equatorial face/vertex, and so these ridges cannot be removed by repositioning those vertices.

§9. Rotational Artifacts on Surfaces

This effect was also shown at the Dagstuhl meeting. It is considerably harder to understand, because there are at least three potential causes:

- the high valency extraordinary vertex at the pole, and its eigenvectors.
- the low valency ones round the equator, and their eigenvectors.
- interaction between the two.

Preliminary experiments indicate that all three can contribute, and different schemes may be sensitive to different aspects. This will be reported in the detail it deserves in a separate paper.

§10. Conclusions

We are starting to understand the individual artifact mechanisms, and can in some cases control them. There still remains significant research to be done in the areas of edge conditions and rotational artifacts. More of a problem is that sometimes the adjustments we would like to make to the extraordinary point masks to control one artifact are exactly opposite to those we want to make to control another, and getting control of everything at once is the next big challenge. It may be necessary to focus on one artifact at the early stages, and on others at later stages. Geometric sensitivity may also be a useful idea to explore.

Acknowledgments. Much of the funding of the work described was provided by the EU through the Mingle project, HPRN-CT-1999-00117.

References

1. Bejancu, A., Semi-cardinal interpolation and difference equations: from cubic B-splines to a three-direction box-spline construction, Technical Report, University of Leeds, Department of Applied Mathematics; submitted for publication 2002.
2. Catmull, E. and J. Clark, Recursively generated B-spline surfaces on arbitrary topological meshes, *Computer-Aided Design* **10** (1978) 350–355.
3. Workshop on Subdivision in Geometric Modeling and Computer Graphics. Dagstuhl, Germany, 6-10 March 2000.
4. Dyn, N. and D. Levin, Stationary and non-stationary binary subdivision schemes, in *Mathematical Methods in Computer Aided Geometric Design II*, T. Lyche, and L. L. Schumaker (eds.), Academic press, 209-216 (1992).
5. Dyn, N., J. Gregory, and D. Levin, A four-point interpolatory subdivision scheme for curve design, *Comput. Aided Geom. Design* **4** (1987) 257–268.
6. Dyn, N., D. Levin and J. Gregory, A butterfly subdivision scheme for surface interpolation with tension control. *ACM Trans. on Graphics* **9** (1990) 160–169.
7. Guskov, I., Irregular subdivision and its applications, Princeton University, 1999.
8. Ivrişimţzis, I. P., M. A. Sabin and N. A. Dodgson, On the support of recursive subdivision, preprint, available as Tech. Rep't No. 544, Univ. of Cambridge Computer Laboratory (2002).
9. Kobbelt, L., $\sqrt{3}$ -Subdivision, *Proceedings of SIGGRAPH 2000, Computer Graphics Proceedings*, (2000), 103-112.
10. Levin, A., Combined subdivision schemes for the design of surfaces satisfying boundary conditions, *Comput. Aided Geom. Design* **16** (1999), 345-354.
11. Loop, C. T., Smooth subdivision surfaces based on triangles, master's thesis, University of Utah, 1987.
12. Nasri, A. H. and G. Farin, A subdivision algorithm for generating rational curves, *Journal of Graphics Tools* **3** (2001) 35–47.
13. Nasri, A. H. and M. A. Sabin, Taxonomy of interpolation conditions on recursive subdivision curves, *The Visual Computer* **18** (2002) 259–272.
14. Peters, J. and U. Reif, The simplest subdivision scheme for smoothing polyhedra, *ACM Trans. on Graphics* **16** (1997), 420–431.

15. Shannon, C. E. and W. Weaver, *The Mathematical Theory of Communication*, University of Illinois Press, Urbana, (1949)
16. Velho, L., Quasi 4–8 subdivision, *Comput. Aided Geom. Design* **18** (2001) 345–357.
17. Warren, J. and H. Weimer, *Subdivision Methods for Geometric Design* Morgan Kaufmann, San Francisco (2002), chapter 4.4 *A smooth subdivision scheme with circular precision*.
18. Wang, H. and K. Qin, Estimating Subdivision Depth of Catmull-Clark Surfaces, submitted.

Malcolm Sabin
Numerical Geometry Ltd.
26 Abbey Lane, Lode
Cambridge, England CB5 9EP
malcolm@geometry.demon.uk

Loïc Barthe
RWTH-Aachen, Informatik VIII
Ahornstrasse 55, 52074 Aachen,
Germany
barthe@informatik.
rwth-aachen.de

7.1.2 Simple computation of the eigencomponents of a subdivision matrix in the Fourier domain.

Auteurs : Loïc Barthe, Cédric Gérot, Malcolm A. Sabin, Leif Kobbelt
Revue : Advances in Multiresolution for Geometric Modelling
Date : 2005
ISBN : 3-540-21462-3

Simple computation of the eigencomponents of a subdivision matrix in the Fourier domain

Loïc Barthe¹, Cédric Géro², Malcolm A. Sabin³, and Leif Kobbelt⁴

¹ Computer Graphics Group, IRIT/UPS, 118 route de Narbonne, 31062 Toulouse Cedex4, France. lbarthe@irit.fr

² Laboratoire des Images et des Signaux, Domaine Universitaire, 38402 Saint Martin D'Herès, France. Cedric.Gerot@lis.inpg.fr

³ Numerical Geometry Ltd., 26 Abbey Lane, Lode, Cambridge, UK. malcolm@geometry.demon.co.uk

⁴ Computer Graphics Group, RWTH Aachen, Ahornstrasse 55, 52074 Aachen, Germany. kobbelt@cs.rwth-aachen.de

Summary. After demonstrating the necessity and the advantage of decomposing the subdivision matrix in the frequency domain when analysing a subdivision scheme, we present a general framework based on the method introduced in [1] which computes the Discrete Fourier Transform of a subdivision matrix. The effectivity of the technique is illustrated by performing the analysis of Kobbelt's $\sqrt{3}$ scheme in a very simple manner.

1 Introduction

Nowadays, subdivision surfaces have become a standard technique for both animation and freeform shape modeling [21]. After *one* step of subdivision, a coarse mesh is refined to a finer one and several iterations generate a sequence of incrementally refined meshes which converge to a smooth surface. The main advantage of subdivision surfaces on other freeform representations such as splines [9] is that they are defined by control meshes with arbitrary connectivity while generating smooth surfaces with arbitrary manifold topology. One of the most important stages in subdivision scheme analysis is the evaluation of the scheme's smoothness properties. This is done in *two* steps:

First, one has to study the continuity properties of the scheme in a regular lattice (composed of valence 6 vertices for triangles meshes and valence 4 vertices for quad meshes). Often the scheme is derived from the uniform knot-insertion operator of some Box-spline surface [2] which leads us to a trivial analysis: By construction the refined meshes converge to piecewise polynomial surfaces with a known degree of smoothness between the patches [3, 5, 11]. On the other hand, the scheme can be non-polynomial [8, 22, 10], i.e. it is not

derived from any known surface representation and the continuity of the limit surface is analysed using sufficient conditions based on z -Transforms [6, 10, 7].

As the second step in the analysis, one has to analyse the scheme's smoothness in the vicinity of extraordinary vertices (EVs). Up to now, the z -Transform fails to provide an efficient tool at EVs, and even though we can prove C^1 continuity for schemes derived from Box-splines by showing that the characteristic map is regular and injective [15, 12, 20], the complete analysis is performed using necessary conditions based on the eigenstructure of the subdivision matrix [5, 1, 16]. In fact, the convergence behavior of a subdivision scheme at an EV is completely determined by the eigencomponents of its subdivision matrix. The analysis of a subdivision scheme [5, 11, 19, 14, 10] in the vicinity of such irregularities of the control mesh hence requires a simple technique for identifying and computing the various eigencomponents. The standard method exploits the scheme's rotational symmetries through the use of the Fourier transform. This partitions the subdivision matrix, which size varies linearly with the valence of the EV, into a block diagonal matrix. Although the number of blocks depends on the valence, the blocks are of fixed size, and so it becomes possible to determine the eigencomponents for all valences with a single algebraic computation.

In this paper, we first illustrate by a practical example the importance of the frequency analysis and we emphasize the necessity of identifying the eigenvalues with respect to their rotational frequency. We then present the general form of Ball and Story's method [1] which performs a fast computation of the different frequency blocks. This approach is illustrated on Kobbelt's $\sqrt{3}$ scheme [10] and we show how very simple computations performed on a single subdivision iteration rather than on the square of the subdivision matrix allow us to deal with the scheme's rotation property and to find the specific subdivision rules for the EVs.

Another method computing the eigencomponents of a subdivision matrix is based on z -Transforms and it exploits the circulant structure of the subdivision matrix' blocks. This technique also leads to very simple computations and all details can be found in [12, 18]. Both methods are equivalent in terms of complexity, however our method computes the eigencomponents in the Fourier domain while the use of z -Transforms provides the eigencomponents in the spacial domain. Depending on the application, one or the other method can be preferred.

2 Frequency of the different eigenvalues

The operator which maps a central EV of valence v and its r -ring neighborhood \mathbf{P} to the same topological configuration \mathbf{p} after *one* step of subdivision is called the subdivision matrix S . The vectors of old vertices \mathbf{P} and new vertices \mathbf{p} are linked by the relation $\mathbf{p} = S\mathbf{P}$. The matrix S is square and each of its rows contains the coefficients of an affine combination of the old vertices \mathbf{P}

which computes *one* new vertex of \mathbf{p} . The convergence behavior of the subdivision scheme at the central EV is completely defined by the eigencomponents (eigenvalues and eigenvectors) of the matrix S . The matrix S is then decomposed into $S = M \Lambda M^{-1}$ where Λ is a diagonal matrix of eigenvalues $\{\lambda_j\}$ and M is a square matrix whose columns are the (*right*) eigenvectors \mathbf{m}_j . This can be well understood if we interpret the eigencomponents as a local Taylor expansion. Indeed, this interpretation allows us to associate the different geometric configurations (position, tangent plane, curvature) with the eigencomponents by which they are defined. The smoothness analysis then relies on necessary conditions for the eigencomponents of the different geometric configurations.

As we will see in Section 2.1, given the mere eigendecomposition of a subdivision matrix, we cannot directly deduce which eigenvalue corresponds to which geometric configuration so that we do not know how to apply the conditions for the scheme's smoothness analysis. In Section 2.2 we show how the decomposition of the subdivision matrix in the Fourier domain resolves this problem.

2.1 Geometric configurations and their eigencomponents

The Taylor expansion of a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ can be written as follows:

$$\begin{aligned} f(x, y) = & f + f_x x + f_y y + (f_{xx} + f_{yy}) \left(\frac{x^2}{4} + \frac{y^2}{4} \right) \\ & + (f_{xx} - f_{yy}) \left(\frac{x^2}{4} - \frac{y^2}{4} \right) + f_{xy} xy + \dots \quad , \end{aligned} \quad (1)$$

where each function expression on the right hand side is evaluated at $(0, 0)$. The point f is a position, the *two* first order partial derivatives f_x and f_y are the coefficients of x and y defining the tangent plane and the *three* second order partial derivatives f_{xx} , f_{yy} and f_{xy} are the coefficients of three quadratic configurations defining the curvature: An elliptic configuration $x^2 + y^2$ denoted as *cup* and *two* rotationally symmetric hyperbolic configurations $x^2 - y^2$ and xy denoted as *saddle*.

On the other hand, the vector of new vertices \mathbf{p} is expressed as a local Taylor expansion when it is computed as $\mathbf{p} = S \mathbf{P} = M \Lambda M^{-1} \mathbf{P} = M \Lambda \mathbf{l}$ with $\mathbf{l} = M^{-1} \mathbf{P}$ [13]. We then have:

$$\mathbf{p} = \mathbf{m}_0 \lambda_0 l_0 + \mathbf{m}_1 \lambda_1 l_1 + \mathbf{m}_2 \lambda_2 l_2 + \mathbf{m}_3 \lambda_3 l_3 + \mathbf{m}_4 \lambda_4 l_4 + \mathbf{m}_5 \lambda_5 l_5 + \dots \quad , \quad (2)$$

where the $l_j \in \mathbb{R}^3$ are the approximations of the Taylor coefficients, i.e., the successive partial derivatives (l_0 is a position, l_1 and l_2 approximate the first order derivatives, etc), the \mathbf{m}_j s correspond to the polynomials in Eq. (1),

4 Loïc Barthe, Cédric Gérot, Malcolm A. Sabin, and Leif Kobbelt

whose function values scale with a certain factor λ_j . Eq. (1) and (2) both behave like local Taylor expansions applied in different contexts. They have identical geometric interpretation and in Eq. (2), the components \mathbf{m}_j , λ_j and l_j with index $j = 0$ are responsible for the central EV's position, components with indices $j = 1, 2$ are responsible for the tangent plane and those with indices $j = 3, 4, 5$ are responsible for the curvature: $j = 3$ for the cup and $j = 4, 5$ for the *two* saddles.

Note that the components with indices $j = 1, 2$ defining the tangent plane configuration are images under rotation, yielding the property $|\lambda_1| = |\lambda_2|$. This is also the case for the *two* saddle configurations which yields $|\lambda_4| = |\lambda_5|$.

The study of the subdivision scheme's smoothness at the EV is based on necessary conditions affecting the different eigencomponents $\{\lambda_j\}$ and $\{\mathbf{m}_j\}$. For instance, the condition $1 = |\lambda_0| > |\lambda_j|$ for all $j > 0$ is necessary for convergence of the scheme, the additional conditions $1 > |\lambda_1|$, $1 > |\lambda_2|$ and $\min(|\lambda_1|, |\lambda_2|) > |\lambda_j|$ for $j > 2$ are necessary for C^1 continuity and if the scheme is C^1 . However, in practice, useful schemes are rotationally symmetric and therefore we restrict our analysis to this special case. This gives us the additional properties : $|\lambda_1| = |\lambda_2|$ and $|\lambda_4| = |\lambda_5|$. Properties like bounded curvature ($|\lambda_2|^2 \geq |\lambda_3|$, $|\lambda_2|^2 \geq |\lambda_4| = |\lambda_5|$ and $\min(|\lambda_3|, |\lambda_4|) > |\lambda_j|$ for $j > 5$) are necessary for C^2 continuity. If $|\lambda_2|^2 = |\lambda_3| = |\lambda_4| = |\lambda_5|$, the scheme has a *non-zero* bounded curvature (without flat spot at the EV) and if $|\lambda_2|^2 > |\lambda_j|$, $j = 3, 4, 5$, the scheme has *zero* curvature generating a flat spot at the EV.

A critical point in the analysis after eigendecomposition of the subdivision matrix is then to identify which index (or configuration) corresponds to which eigencomponents. Since we know which eigenvalue corresponds to which eigenvector, the task is reduced to the identification of the different eigenvalues. This is illustrated by the following example:

Let us consider a variant of Loop's subdivision scheme [11] having its n -uplet of eigenvalues $(\lambda_0, \dots, \lambda_7)$ at a valence 7 EV sorted by geometric configuration (following our Taylor notation (2)) and having their value in the set:

$$\left\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \lambda_j < \frac{1}{4}\right\} \quad (3)$$

We emphasize that the set of eigenvalues is not sorted from the greatest to the smallest as it is usually done, but by geometric configuration. The question is: How can we know which one of the eigenvalues in (3) is $\lambda_0, \lambda_1, \dots$? Indeed, following (2), each order satisfies different properties. For instance, the eigenvalues can have the following values:

$$(\lambda_0, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6, \lambda_7) = \left(1, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, < \frac{1}{4}, < \frac{1}{4}\right), \quad (4)$$

and hence one can deduce that the scheme is certainly C^1 and that it has bounded curvature. If indeed the analysis of the characteristic map proves C^1

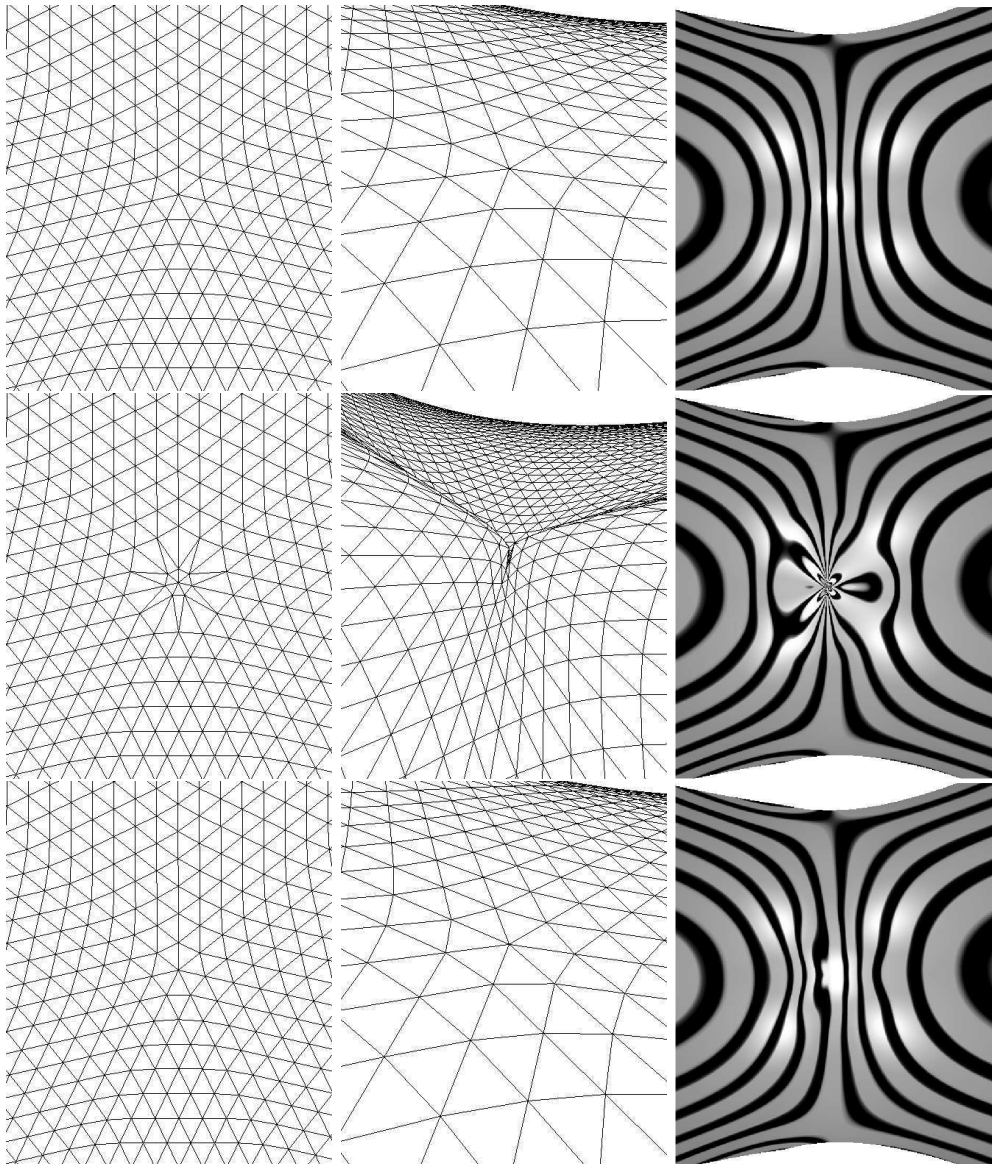


Fig. 1. Different versions of Loop's scheme at a valence 7 EV. The first column shows the subdivision of a semi-regular planar mesh with the EV in its center. The second column shows a subdivided saddle mesh with the EV in its center and the third column illustrates the curvature behavior using reflexion lines on the same saddle mesh. In the first row, the scheme has the eigenspectrum (4) and it is C^1 continuous with non-zero bounded curvature (no flat spot). In the second row, it has the eigenspectrum (5). We clearly see the shrinking factor of $\lambda_1 = \lambda_2 = \frac{1}{4}$ at the EV in the planar configuration and the C^1 discontinuity in the saddle mesh. In the third row it has the eigenspectrum (6) and the scheme is C^1 continuous with bounded curvature (flat spot in the saddle configuration). The misbehavior of the curvature is illustrated by the reflexion lines in the saddle mesh.

6 Loïc Barthe, Cédric Gérot, Malcolm A. Sabin, and Leif Kobbelt

continuity, the scheme is C^1 at the EV and it has bounded curvature without flat spot (first row in Fig. 1). However if the eigenvalues are actually:

$$(\lambda_0, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6, \lambda_7) = (1, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{2}, \frac{1}{2}, < \frac{1}{4}, < \frac{1}{4}), \quad (5)$$

the necessary condition for C^1 continuity is not satisfied ($|\lambda_2| < |\lambda_4|$) so that the scheme is not C^1 and it is not even necessary to analyse the characteristic map (second row in Fig. 1). Finally, the situation where :

$$(\lambda_0, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6, \lambda_7) = (1, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \lambda_4 < \frac{1}{4}, \lambda_5 < \frac{1}{4}, \frac{1}{4}, \frac{1}{4}), \quad (6)$$

is more problematic because if the analysis of the characteristic map proves that the scheme is C^1 , one could conclude from the eigenspectrum (3) that it has also bounded curvature without flat spot at the EV ($|\lambda_1|^2 = |\lambda_2|^2 = |\lambda_3| = |\lambda_4| = |\lambda_5|$) while it is not the case. In the saddle configuration, the curvature is bounded with a flat spot, and some curvature misbehavior can be introduced in the limit surface by the eigenvectors corresponding to the eigenvalues λ_6 and λ_7 (third row in Fig. 1).

The decomposition of the subdivision matrix in the Fourier domain will allow us to determine if we are in the situation (4), (5), (6) or in a situation where the eigenvalues are sorted in a still different manner.

2.2 Identification of the eigenvalues in the Fourier domain

The identification of the eigenvalues is based on the decomposition of the subdivision matrix S in the Fourier domain. The block circulant subdivision matrix S with n blocks $S_{i,j}$ is transformed into a block diagonal matrix \tilde{S} having v blocks \tilde{S}_ω which correspond to the *rotational frequencies* $\omega = \{0, \dots, v-1\}$ ordered in frequency in \tilde{S} (as illustrated in Equation (7)). We define the *rotational frequency* just below. The eigenvalues of the frequency blocks \tilde{S}_ω are amplitudes of the eigenvalues of the matrix S [1, 17], hence if we know which frequency represents each configuration (position, tangent plane, cup and saddle), we can identify the eigenvalues from the frequency block in which they are computed.

$$S = \begin{bmatrix} | & | & | & | \\ \hline & S_{0,0} & \cdots & S_{0,n} \\ \hline & \vdots & \ddots & \vdots \\ & \vdots & & \vdots \\ \hline S_{n,0} & \cdots & & S_{n,n} \\ \hline | & | & | & | \end{bmatrix} \longrightarrow \tilde{S} = \begin{bmatrix} [\tilde{S}_0] & & & 0 \\ & [\tilde{S}_1] & & \\ & & \ddots & \\ 0 & & & [\tilde{S}_{v-1}] \end{bmatrix} \quad (7)$$

In the Taylor expansion (1), the constant term refers to a position, the terms for x and y define the tangent plane and terms in $x^2 + y^2$, $x^2 - y^2$ and xy define the different curvature configurations (cup and *two* saddles). The expression of these configurations in a cylindrical coordinate system (Eq. (8)) as a periodic function $g(\theta) = \rho \cos(\omega\theta + \varphi)$ (where ρ is the amplitude, ω is the frequency and φ is the phase) provides directly the rotational frequency ω associated to each configuration. For instance, $x^2 - y^2 = r^2 \cos(2\theta)$ and hence this saddle configuration has a frequency component $\omega = 2$. Note that due to rotational symmetry, $\tilde{S}_\omega = \tilde{S}_{v-\omega}$ so that it is enough to consider the frequencies $\omega = \{0, \dots, \frac{v}{2}\}$.

$$(x, y, z) \rightarrow (r, \theta, z) \quad \text{with} \quad \begin{cases} x = r \cos(\theta) \\ y = r \sin(\theta) \\ z = z \end{cases} \quad (8)$$

This tells us that the position configuration has the frequency $\omega = 0$ and hence the eigenvalue $|\lambda_0|$ is the dominant eigenvalue of the frequency block \tilde{S}_0 , the tangent plane configuration has the frequency $\omega = 1$ and $|\lambda_1| = |\lambda_2|$ equals the dominant eigenvalue $\tilde{\mu}_1$ of the frequency block \tilde{S}_1 , the cup configuration has the frequency $\omega = 0$ and $|\lambda_3|$ equals the subdominant eigenvalue $\tilde{\mu}_0$ of the frequency block \tilde{S}_0 and finally the saddle configurations have the frequency $\omega = 2$ and $|\lambda_4| = |\lambda_5|$ equals the dominant eigenvalue $\tilde{\mu}_2$ of the frequency block \tilde{S}_2 . This relation between the different eigencomponents and their rotational frequencies is presented in [5].

When a scheme is convergent, its eigenvalue λ_0 equals 1 (Sect. 2.1) and since $|\lambda_0|$ is the dominant eigenvalue of the frequency block \tilde{S}_0 , it can be written as:

$$\tilde{S}_0 = \begin{bmatrix} 1 & \cdots \\ 0 & [\tilde{S}'_0] \end{bmatrix}. \quad (9)$$

In this paper, we only consider convergent subdivision schemes, and so $\lambda_0 = 1$ and the cup eigenvalue $|\lambda_3|$ is the dominant eigenvalue $\tilde{\mu}_0$ of the block \tilde{S}'_0 .

3 Computation of the frequency blocks

In this section, we present a general framework, used in [1] on Catmull-Clark's scheme [3], which computes the eigencomponents in the frequency domain. We present the procedure on a triangular lattice with a 2-ring neighborhood around the central EV and a standard dyadic refinement (see Fig. 2(a)). The adaptation to a single 1-ring neighborhood or to quad meshes [1] is straightforward. We notice that this method may not be suitable to analyse non-rotationally symmetric schemes, however, as pointed out in [4], these schemes are mainly interesting for theoretical studies and all the schemes used in practical applications are rotationally symmetric.

8 Loïc Barthe, Cédric Gérot, Malcolm A. Sabin, and Leif Kobbelt

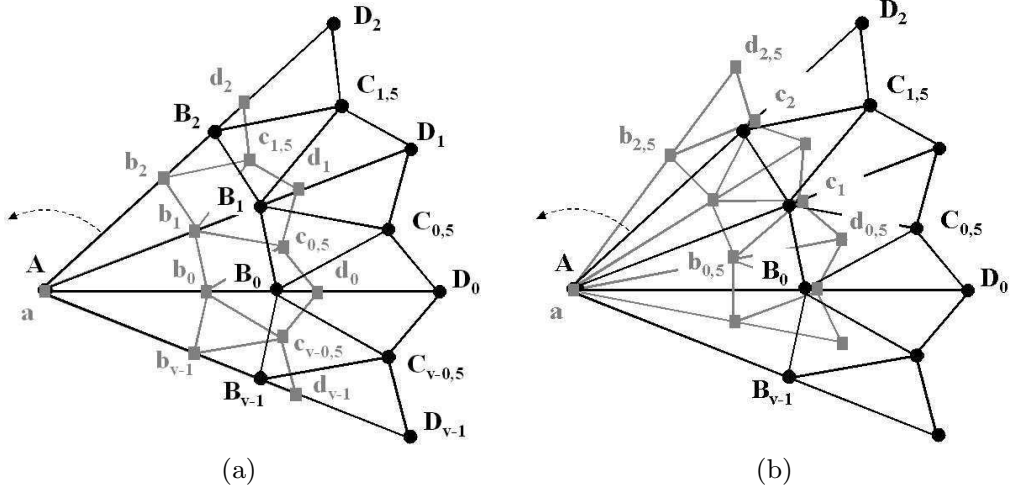


Fig. 2. Subdivision with (a) a dyadic refinement and (b) a $\sqrt{3}$ refinement, of a 2-ring configuration around a central EV A of valence v . Capital letters denote the set of old vertices \mathbf{P} and small letters the set of new vertices \mathbf{p} .

The set of old vertices \mathbf{P} is defined as follows:

$$\mathbf{P} = \{A, B_0, \dots, B_{v-1}, C_{\frac{1}{2}}, \dots, C_{v-\frac{1}{2}}, D_0, \dots, D_{v-1}\},$$

where the different letters $X = \{B, C, D\}$ denote the different sets of rotationally symmetric vertices around the central EV A and the indices give the rotational position of each vertex: Vertex X_j is at an angle of $\frac{j2\pi}{v}$ (where v is the valence of the EV) from the axis of origin. The set of new vertices \mathbf{p} :

$$\mathbf{p} = \{a, b_0, \dots, b_{v-1}, c_{\frac{1}{2}}, \dots, c_{v-\frac{1}{2}}, d_0, \dots, d_{v-1}\},$$

represents the same topological configuration, but after *one* step of subdivision and around the central EV a . The new vertices of \mathbf{p} are computed using affine combinations g_k^j of the old vertices of \mathbf{P} :

$$a = g^0(\mathbf{P}), \quad b_k = g_k^1(\mathbf{P}), \quad c_{k+\frac{1}{2}} = g_k^2(\mathbf{P}), \quad d_k = g_k^3(\mathbf{P}), \quad k = 0, \dots, v-1,$$

where the affine combinations g_k^j are the rows of the subdivision matrix S .

Using the Discrete Fourier Transform (DFT):

$$\tilde{x}_\omega = \frac{1}{v} \sum_{l=0}^{v-1} x_l \exp(-2\pi i \omega l / v), \quad x = \{a, b, c, d\}, \quad (10)$$

we express the rotational frequencies $\{\tilde{a}_\omega, \tilde{b}_\omega, \tilde{c}_\omega, \tilde{d}_\omega\}$ of each set of rotationally symmetric new vertices of \mathbf{p} in terms of the rotational frequencies $\tilde{\mathbf{P}}_\omega = \{\tilde{A}_\omega, \tilde{B}_\omega, \tilde{C}_\omega, \tilde{D}_\omega\}$ of the old vertices of \mathbf{P} :

$$\tilde{a}_\omega = \tilde{g}_\omega^0(\tilde{\mathbf{P}}_\omega), \quad \tilde{b}_\omega = \tilde{g}_\omega^1(\tilde{\mathbf{P}}_\omega), \quad \tilde{c}_\omega = \tilde{g}_\omega^2(\tilde{\mathbf{P}}_\omega), \quad \tilde{d}_\omega = \tilde{g}_\omega^3(\tilde{\mathbf{P}}_\omega),$$

where the affine combinations \tilde{g}_ω^j are the rows of the frequency blocks \tilde{S}_ω of the discrete Fourier transform \tilde{S} of the subdivision matrix S . In vertices A and a , the only frequency is the *zero* frequency, hence the terms in $\exp(-2\pi i \omega l/v)$ vanish in the expression of \tilde{a}_ω and we have: $A = \tilde{A}_0 = \tilde{A}$, $a = \tilde{a}_0 = \tilde{a}$ and $\forall \omega > 0$, $\tilde{A}_\omega = \tilde{a}_\omega = 0$. Furthermore, in order to express the frequency block \tilde{S}_0 as in Equation (9), we have to center the analysis at the central EV so that:

$$\begin{bmatrix} \tilde{a}_0 \\ \tilde{b}_0 - \tilde{a}_0 \\ \tilde{c}_0 - \tilde{a}_0 \\ \tilde{d}_0 - \tilde{a}_0 \end{bmatrix} = \begin{bmatrix} \tilde{S}_0 \end{bmatrix} \begin{bmatrix} \tilde{A}_0 \\ \tilde{B}_0 - \tilde{A}_0 \\ \tilde{C}_0 - \tilde{A}_0 \\ \tilde{D}_0 - \tilde{A}_0 \end{bmatrix} \quad \text{with} \quad [\tilde{S}_0] = \begin{bmatrix} 1 & \cdots \\ 0 & [\tilde{S}'_0] \end{bmatrix},$$

hence

$$\begin{bmatrix} \tilde{b}_0 - \tilde{a}_0 \\ \tilde{c}_0 - \tilde{a}_0 \\ \tilde{d}_0 - \tilde{a}_0 \end{bmatrix} = \begin{bmatrix} \tilde{S}'_0 \end{bmatrix} \begin{bmatrix} \tilde{B}_0 - \tilde{A}_0 \\ \tilde{C}_0 - \tilde{A}_0 \\ \tilde{D}_0 - \tilde{A}_0 \end{bmatrix},$$

and because for $\omega > 0$: $\tilde{A}_\omega = \tilde{a}_\omega = 0, \forall \omega > 0$ we have:

$$\begin{bmatrix} \tilde{b}_\omega \\ \tilde{c}_\omega \\ \tilde{d}_\omega \end{bmatrix} = \begin{bmatrix} \tilde{S}_\omega \end{bmatrix} \begin{bmatrix} \tilde{B}_\omega \\ \tilde{C}_\omega \\ \tilde{D}_\omega \end{bmatrix}.$$

The size of the frequency blocks is equal to the number of sets of rotationally symmetric vertices in the neighborhood of the central EV. Hence, these blocks are of fixed size (here they are 3×3 matrices) and they can be expressed as a function of the valence v of the central EV. The original problem of computing the eigencomponents of large $(3v + 1) \times (3v + 1)$ matrices (*in the spatial domain*) for each value of the valence v is reduced to a single eigendecomposition of small 3×3 matrices (*in the Fourier domain*), providing the different eigencomponents expressed in terms of the valence v . Depending on their complexity, the frequency blocks can be either decomposed by hand computations or using the symbolic toolbox of any mathematical software.

We note that the right eigenvectors $\tilde{\mathbf{m}}_0$, $\tilde{\mathbf{m}}_1$ and $\tilde{\mathbf{m}}_2$ associated respectively to the eigenvalues $\tilde{\mu}_0$, $\tilde{\mu}_1$ and $\tilde{\mu}_2$, can be interpreted as amplitudes of the eigenvectors $\mathbf{m}_0, \dots, \mathbf{m}_5$ of the spatial domain, e.g. the tangent plane eigenvector $\tilde{\mathbf{m}}_1 = [r_B, r_C, r_D]$ gives the radii r_X of vertices B_k , $C_{k+\frac{1}{2}}$ and D_k from the central EV A .

4 Example by Kobbelt's $\sqrt{3}$ scheme

We illustrate by the practical example of Kobbelt's $\sqrt{3}$ subdivision scheme [10] the application of the general procedure presented in Sect. 3. This scheme rotates the lattice after *one* step of subdivision due to the insertion of new vertices in the middle of the old faces (as we can see in Fig. 2(b)). If the eigen-decomposition is performed on the subdivision matrix S , we obtain complex eigencomponents generated by the scheme's rotation property. To overcome this difficulty, in [10] the scheme is rather analysed after *two* steps of subdivision so that the lattice is aligned with the *two* steps older one but rotated by *one* sector. The lattice is then rotated back using a permutation matrix R , and the matrix studied finally is $\hat{S} = R S^2$.

We first compute the frequency blocks on the 2-ring configuration shown in Fig. 2(b). The computation is performed on a single subdivision step without any back-rotation. We then reproduce the results presented in [10] using the eigencomponents in the Fourier domain. We will see that the computations are so simple that they can be done quickly and easily by hand.

Kobbelt's $\sqrt{3}$ scheme is composed of *two* refinement rules (*stencils*): One which displaces an old vertex (Eq. (11)) and one which computes a new vertex in the center of a triangle (Eq. (12)). They are defined by the following formulae:

$$a = (1 - \alpha_v)A + \frac{\alpha_v}{v} \sum_{j=0}^{v-1} B_j \quad (11)$$

$$b_{k+\frac{1}{2}} = \frac{1}{3}(A + B_k + B_{k+1}), \quad (12)$$

where α_v is a parameter which can be used in order to improve the surface smoothness at the EV for different valences v . The new vertices c_k are computed using the regular relaxation rule (valence 6 vertex) and the new vertices $d_{k+\frac{1}{2}}$ using the insertion rule as follows:

$$c_k = \frac{2}{3}B_k + \frac{1}{18}(A + B_{k-1} + B_{k+1} + C_{k-\frac{1}{2}} + C_{k+\frac{1}{2}} + D_k)$$

$$d_{k+\frac{1}{2}} = \frac{1}{3}(B_k + B_{k+1} + C_{k+\frac{1}{2}}).$$

The DFT (Eq. (10)) is then used to derive the rotational frequencies of the different sets of rotationally symmetric vertices:

$$\begin{aligned}
 \tilde{a}_0 &= 1. \sum_{j=0}^0 a e^0 = (1 - \alpha_v)A + \frac{\alpha_v}{v} \sum_{j=0}^v B_j \quad \text{using the DFT and Eq. (11)} \\
 &= (1 - \alpha_v)\tilde{A}_0 + \alpha_v\tilde{B}_0 \quad \text{because the only frequency in } a \text{ is } \omega = 0 \\
 \tilde{b}_\omega &= \frac{1}{v} \sum_{j=0}^{v-1} b_j e^{-2\pi i \omega j/v} \quad \text{DFT} \\
 &= \frac{1}{v} \sum_{j=0}^{v-1} \left[\frac{1}{3} (A + B_{j-\frac{1}{2}} + B_{j+\frac{1}{2}}) \right] e^{-2\pi i \omega j/v} \quad \text{using Eq. (12)} \\
 &= \frac{1}{3}A + \frac{1}{3v} \sum_{l=-\frac{1}{2}}^{v-\frac{3}{2}} B_l e^{-2\pi i \omega (l+\frac{1}{2})/v} + \frac{1}{3v} \sum_{l=\frac{1}{2}}^{v-\frac{1}{2}} B_l e^{-2\pi i \omega (l-\frac{1}{2})/v} \\
 &= \frac{1}{3}A + \frac{1}{3v} e^{-2\pi i \omega /2v} \sum_{l=-\frac{1}{2}}^{v-\frac{3}{2}} B_l e^{-2\pi i \omega l/v} + \frac{1}{3v} e^{2\pi i \omega /2v} \sum_{l=\frac{1}{2}}^{v-\frac{1}{2}} B_l e^{-2\pi i \omega l/v} \\
 &= \frac{1}{3}\tilde{A}_0 + \frac{1}{3} \left(e^{-\pi i \omega /v} + e^{\pi i \omega /v} \right) \tilde{B}_\omega \\
 &= \frac{1}{3}\tilde{A}_0 + \frac{2}{3}k_\omega \tilde{B}_\omega \quad \text{with } k_\omega = \cos\left(\frac{\pi\omega}{v}\right) \text{ and because } e^{-i\theta} + e^{i\theta} = 2 \cos \theta
 \end{aligned}$$

hence

$$\begin{cases} \tilde{b}_0 - \tilde{a}_0 = \left(\frac{2}{3} - \alpha_v\right) (\tilde{B}_0 - \tilde{A}_0) & \text{if } \omega = 0 \\ \tilde{b}_\omega = \frac{2}{3}k_\omega \tilde{B}_\omega & \text{otherwise.} \end{cases}$$

Using similar computations, we obtain:

$$\begin{cases} \tilde{c}_0 - \tilde{a}_0 = \left(\frac{7}{9} - \alpha_v\right) (\tilde{B}_0 - \tilde{A}_0) + \frac{1}{9} (\tilde{C}_0 - \tilde{A}_0) + \frac{1}{18} (\tilde{D}_0 - \tilde{A}_0) & \text{if } \omega = 0 \\ \tilde{c}_\omega = \left(\frac{2}{3} + \frac{1}{9}k_{2\omega}\right) \tilde{B}_\omega + \frac{1}{9}k_\omega \tilde{C}_\omega + \frac{1}{18}\tilde{D}_\omega & \text{otherwise} \\ \tilde{d}_0 - \tilde{a}_0 = \left(\frac{2}{3} - \alpha_v\right) (\tilde{B}_0 - \tilde{A}_0) + \frac{1}{3} (\tilde{C}_0 - \tilde{A}_0) & \text{if } \omega = 0 \\ \tilde{d}_\omega = \frac{2}{3}k_\omega \tilde{B}_\omega + \frac{1}{3}\tilde{C}_\omega & \text{otherwise.} \end{cases}$$

From these expressions, we directly deduce the frequency blocks:

$$\tilde{S}'_0 = \begin{bmatrix} \frac{2}{3} - \alpha_v & 0 & 0 \\ \frac{7}{9} - \alpha_v & \frac{1}{9} & \frac{1}{18} \\ \frac{2}{3} - \alpha_v & \frac{1}{3} & 0 \end{bmatrix} \quad \text{and if } \omega > 0 \quad \tilde{S}_\omega = \begin{bmatrix} \frac{2}{3}k_\omega & 0 & 0 \\ \frac{2}{3} + \frac{1}{9}k_{2\omega} & \frac{1}{9}k_\omega & \frac{1}{18} \\ \frac{2}{3}k_\omega & \frac{1}{3} & 0 \end{bmatrix}.$$

12 Loïc Barthe, Cédric Gérot, Malcolm A. Sabin, and Leif Kobbelt

Since the only non-zero coefficient of the first rows of the matrices \tilde{S}'_0 and \tilde{S}'_ω is the one in the left-hand corner, we directly obtain the dominant eigenvalues $\tilde{\mu}_0$, $\tilde{\mu}_1$ and $\tilde{\mu}_2$ as:

$$\tilde{\mu}_0 = \frac{2}{3} - \alpha_v, \quad \tilde{\mu}_1 = \frac{2}{3}k_1, \quad \tilde{\mu}_2 = \frac{2}{3}k_2. \quad (13)$$

As we see, the free parameter α_v only appears in the cup eigenvalue, hence the scheme's behavior is improved at the EV by bounding the curvature in the cup configuration using the condition: $\tilde{\mu}_0 = \tilde{\mu}_1^2$. The value of parameter α_v in Eq. (11) is then derived as follows:

$$\tilde{\mu}_0 = \tilde{\mu}_1^2 \iff \frac{2}{3} - \alpha_v = \left(\frac{2}{3}k_1\right)^2 \iff \alpha_v = \frac{2}{9} \left(2 - \cos\left(\frac{2\pi}{v}\right)\right).$$

The eigenvalues $\tilde{\mu}_i$ ($i = 1, 2, 3$) could have been computed in an even simpler manner by only considering a 1-ring neighborhood around the central EV. However, the choice of a 2-ring neighborhood is based on our wish of giving an example allowing a more complete analysis based on a larger neighborhood. It also allows us to check that more eigenvalues are adequately sorted (see the necessary conditions on the eigenvalues in Sect. 2.1).

5 Conclusion

In this paper, we have emphasized the importance of the analysis of the subdivision matrix in the Fourier domain: The analysis of large matrices in the spatial domain is reduced to the analysis of small matrices in the Fourier domain so that it becomes easier to compute the different eigencomponents. Moreover, we can determine which geometric configuration is defined by which eigencomponents. We have presented a general framework computing the subdivision matrix in the frequency domain and it has been illustrated on the practical example of Kobbelt's $\sqrt{3}$ scheme. We have shown how this computation technique allows us to analyse the scheme in a very simple manner.

There are limitations for this computation technique when the rotational position of a set of rotationally symmetric vertices with respect to the axis of origin is unknown. More investigations have to be carried out to solve this problem while keeping the computations as simple as possible. As we have demonstrated however, this approach performs on 2-ring neighborhood configurations and hence it is very well suited to analyse most of the standard subdivision schemes or any new rotationally symmetric scheme.

References

- [1] Ball, A.A., Storry, D.J.T.: Conditions for tangent plane continuity over recursively generated B-spline surfaces. *ACM Transactions on Graphics*, **7** (2), 83–102 (1988)
- [2] de Boor, C., Hollig, D., Riemenschneider, S.: *Box Splines*. Springer-Verlag, New York (1994).
- [3] Catmull, E., Clark, J.: Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, **10** (6), 350–355 (1978)
- [4] Dodgson N.A., Ivriissimtzis, I.P., Sabin, M.A.: Characteristics of dual triangular $\sqrt{3}$ subdivision. *Curves and Surface Fitting: Saint-Malo 2002*, Nashboro Press, Brentwood, 119–128 (2003)
- [5] Doo, D., Sabin, M.A.: Analysis of the behaviour of recursive subdivision surfaces near extraordinary points. *Computer Aided Design*, **10** (6), 356–360 (1978)
- [6] Dyn, N.: Subdivision schemes in Computer-Aided Geometric Design. *Advances in Numerical Analysis*, **II**, Wavelets, Subdivision Algorithms and Radial Basis Functions, W. Light (ed.), Clarendon Press, Oxford, 36–104 (1992)
- [7] Dyn, N.: Analysis of convergence and smoothness by the formalism of Laurent polynomials. *Tutorials on Multiresolution in Geometric Modelling*, A. Iske, E. Quak and M. Floater (ed.), Springer, 51–68 (2002)
- [8] Dyn, N., Levin, D., Gregory, J.: A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transaction on Graphics*, **9** (2) 160–169 (1990)
- [9] Farin, G.: *Curves and Surfaces for CAGD*. 5th Edition, Academic Press (2002)
- [10] Kobbelt, L.: $\sqrt{3}$ -Subdivision. *Proceedings of SIGGRAPH 2000, Computer Graphics Proceedings, Annual Conference Series, ACM*, 103–112 (2000)
- [11] Loop, C.: Smooth subdivision surfaces based on triangles. Master’s thesis, University of Utah (1987)
- [12] Peters, J., Reif, U.: Analysis of algorithms generalizing B-spline subdivision. *SIAM Journal of Num. Anal.*, **35** (2), 728–748 (1998)
- [13] Prautzsch, H.: Analysis of Ck-Subdivision surfaces at extraordinary points. Technical Report 98/4, Fakultät für Informatik, University of Karlsruhe, Germany (1998)
- [14] Prautzsch, H., Umlauf, G.: A G^2 -subdivision algorithm. *Geometric Modelling, Dagstuhl 1996, Computing supplement 13, Springer-Verlag*, 217–224 (1998)
- [15] Reif, U.: A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Design*, **12**, 153–174 (1995)
- [16] Sabin, M.A.: Eigenanalysis and artifacts of subdivision curves and surfaces. *Tutorials on Multiresolution in Geometric Modelling*, A. Iske, E. Quak and M. Floater (ed.), Springer, 69–97 (2002)
- [17] Stam, J.: Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. *Proceedings of SIGGRAPH 1998, ACM*, 395–404 (1998)
- [18] Warren, J., Weimer, H.: *Subdivision methods for geometric design: A constructive approach*. San Francisco: Morgan Kaufman, (2002)

- 14 Loïc Barthe, Cédric Géro, Malcolm A. Sabin, and Leif Kobbelt
- [19] Zorin, D.: Stationary subdivision and multiresolution surface representations. PhD thesis, Caltech, Pasadena, California (1997)
- [20] Zorin, D.: A method for analysis of C^1 -continuity of subdivision surfaces. *SIAM Journal of Num. Anal.*, 35 (5), 1677–1708 (2000)
- [21] Zorin, D., Schröder, P: Subdivision for modeling and animation. *SIGGRAPH 2000 Course Notes*. (2000)
- [22] Zorin, D., Schröder, P., Sweldens, W.: Interpolating subdivision for meshes with arbitrary topology. *Proceedings of SIGGRAPH 1997, ACM*, 189–192 (1996)

7.1.3 Subdivision scheme tuning around extraordinary vertices.

Auteurs : Loïc Barthe, Leif Kobbelt

Revue : Computer Aided Geometric Design, Vol 21(6)

Date : Juin 2004

Subdivision Scheme Tuning Around Extraordinary Vertices

Loïc Barthe Leif Kobbelt

*Computer Graphics Group, RWTH Aachen
Ahornstrasse 55, 52074 Aachen, Germany*

Abstract

In this paper we extend the standard method to derive and optimize subdivision rules in the vicinity of extraordinary vertices (EV). Starting from a given set of rules for regular control meshes, we tune the extraordinary rules (ER) such that the necessary conditions for C^1 continuity are satisfied along with as many necessary C^2 conditions as possible. As usually done, our approach sets up the general configuration around an EV by exploiting rotational symmetry and reformulating the subdivision rules in terms of the subdivision matrix' eigencomponents. The degrees of freedom are then successively eliminated by imposing new constraints which allows us, e.g., to improve the curvature behavior around EVs. The method is flexible enough to simultaneously optimize several subdivision rules, i.e. not only the one for the EV itself but also the rules for its direct neighbors. Moreover it allows us to prescribe the stencils for the ERs and naturally blends them with the regular rules that are applied away from the EV. All the constraints are combined in an optimization scheme that searches in the space of feasible subdivision schemes for a candidate which satisfies some necessary conditions exactly and other conditions approximately. The relative weighting of the constraints allows us to tune the properties of the subdivision scheme according to application specific requirements. We demonstrate our method by tuning the ERs for the well-known Loop scheme and by deriving ERs for a $\sqrt{3}$ -type scheme based on a 6-direction Box-spline.

Key words: Subdivision surfaces, Meshes, Extraordinary vertices, Artifacts

Email addresses: lbarthe@irit.fr (Loïc Barthe),
kobbelt@cs.rwth-aachen.de (Leif Kobbelt).

URLs: <http://www.rwth-graphics.de/> (Loïc Barthe),
<http://www.rwth-graphics.de/> (Leif Kobbelt).

1 Introduction

Subdivision schemes have become a well-established representation for freeform surface geometry in many fields of computer graphics and geometric design (Zorin and Schröder, 2000). The most important advantage of subdivision surfaces compared to other freeform representations such as splines (Farin, 2002) is that they can be defined by control meshes with arbitrary connectivity and hence can generate globally smooth surfaces with arbitrary (manifold) topology. An efficient algorithm to approximate subdivision surfaces is to apply a subdivision operator to the given control mesh. This generates a sequence of meshes which quickly converge to a smooth limit surface. A subdivision operator consists of a *splitting rule* that refines the mesh resolution by inserting new vertices and splitting faces, plus a *smoothing rule* that computes new vertex positions by weighted affine combinations of old vertex positions.

During the last years, many different schemes have been proposed (Catmull and Clark, 1978; Doo and Sabin, 1978; Loop, 1987; Dyn, 1992; Kobbelt, 1996; Peters and Reif, 1997; Kobbelt, 2000; Velho and Zorin, 2001) and attempts have been made to classify them according to various properties such as primal vs. dual splitting rules, tensorproduct smoothing rules vs. non-tensorproduct, or interpolatory vs. approximating. One common theme in the derivation of all these schemes is that one usually starts with a subdivision scheme for regular meshes and then derives special smoothing rules for the vicinity of extraordinary vertices (EV) (with valence $\neq 6$ for triangles meshes or valence $\neq 4$ for quad meshes).

Since the regular rules are often taken from the uniform knot-insertion operator of some (Box-) spline surface (de Boor et al., 1994), the analysis in the regular case is usually trivial: by construction the refined meshes converge to piecewise polynomial surfaces with a known degree of smoothness between the patches. Moreover, we can construct schemes of arbitrary smoothness by applying repeated averaging operators to the mesh (Stam, 2001; Zorin and Schröder, 2001).

What then remains is the derivation of extraordinary smoothing rules in the vicinity of EVs and the analysis of the resulting limit surface. Since the splitting rules do not introduce new EVs during subdivision, it is sufficient to check the smoothness at an isolated point. The known necessary conditions for the smoothness of a subdivision surface at an EV are typically formulated in terms of the eigenstructure of the subdivision matrix (Doo and Sabin, 1978; Ball and Storry, 1988; Reif, 1995; Warren and Weimer, 2002). This matrix is a representation of the linear operator which maps a given mesh neighborhood of an EV to the same neighborhood on the next refinement level.

The *standard* approach to find extraordinary rules (ER) is to apply some geometric heuristic that leaves just one free parameter (Doo and Sabin, 1978; Loop, 1987; Kobbelt, 2000). Then this parameter is chosen such that certain conditions on the eigenstructure of the resulting subdivision matrix are satisfied. By this approach exactly one valence dependent extraordinary rule is found and all degrees of freedom are used up by satisfying one specific necessary C^2 conditions (in addition to satisfying the C^1 conditions by construction). We are generalizing this approach by using a generic construction of feasible subdivision schemes which provides *several* free parameters that can be used to satisfy *more* constraints.

Another, purely algebraic, approach is to start with some dummy ER and then modify the eigenstructure of the resulting subdivision matrix such that the necessary C^1 (or even C^2) conditions are satisfied (Prautzsch and Umlauf, 1998). Although this approach provides more flexibility with respect to the smoothness conditions, it modifies *all* rules in the analysed neighborhood of the EV causing the stencils to cover the whole *one*-ring neighborhood after the modification. More recently, Loop (2002) proposed an improved version of his scheme, also based on the same large stencils, where Chebyshev polynomials are judiciously used in order to control the value of the different stencils' coefficients. It results in a scheme which has the convex hull property in addition to bounded curvature. However, in the different approaches presented above, no special control is exerted on the eigenvectors. As a consequence, some curvature eigenvectors can severely violate some necessary C^2 continuity conditions and although the scheme has bounded curvature, undesirable flatness can be generated at the EVs. This phenomenon can be observed in (Loop, 2002, Figure 5(d)), where after the first step of subdivision of a saddle-like control mesh, the first ring around the EV is almost flat.

Finally, Umlauf (1999) uses fairness functionals in order to determine the subdivision matrix' eigencomponents by energy minimization. However, this method leads to non-stationary schemes and does not provide control over the stencils of the subdivision rules. In contrast, our goal is to derive stationary subdivision rules with prescribed stencils.

In this paper we are proposing a construction for the ER which extends the standard approach. We use the well-established method where the subdivision matrix is setup in its general form. By representing the subdivision matrix in terms of its eigenvectors and eigenvalues, it is easy to take natural restrictions like rotational symmetry into account. From this state, we show how to derive the degrees of freedom and how to eliminate them by imposing further constraints emerging from the application of the regular smoothing rule for the vertices in the 2-ring neighborhood and from fundamental properties of the characteristic map. Even though C^1 continuity is not guaranteed theoretically, our experiments have always provided C^1 continuous schemes. The remaining

degrees of freedom are used to approximately satisfy further C^2 conditions such as the well known bounded curvature condition (Doo and Sabin, 1978) or a recently investigated quadratic precision condition (Gerot et al., 2004).

We point out that through our formulation of subdivision rules and constraints in terms of the subdivision matrix' eigenvectors, we can choose which rules are to be modified (one or several) and we can prescribe the stencils and symmetries for these rules. On the other hand, we can directly control the eigenstructure, e.g., by setting the subdominant eigenvalues appropriately.

In the following we will first review the theoretical background to setup the notation used throughout the paper. Then we will explain the general procedure that defines the eigenvectors and eigenvalues of the subdivision matrix. Finally we will demonstrate how to apply this technique to the well-known Loop subdivision scheme and a new $\sqrt{3}$ -type subdivision scheme derived from a 6-direction Box-spline. The choice of the 6-direction Box-spline emphasizes the ability of our technique to deal with complicated subdivision scheme having large support and *complex* eigenvalues. For this reason, this paper focusses on triangular lattices. The application of our approach to Catmull-Clark's scheme (Catmull and Clark, 1978) and more generally to quadrilateral lattices will be the topic of our future work.

2 Theoretical background

In recursive subdivision, the operator which maps the vicinity of an EV into the same vicinity on the next refinement level is called the *subdivision matrix*. Each row of the subdivision matrix is a smoothing rule which computes a new vertex as an affine combination of vertices of the original mesh.

The convergence behavior of a subdivision scheme at an EV is completely defined by the eigenvectors of its subdivision matrix. All standard tools to analyse the limit surface continuity at an EV are based on this eigenstructure (Reif, 1995; Zorin, 1997; Peters and Reif, 1998; Prautzsch, 1998; Zorin, 2000; Peters and Umlauf, 2001). Moreover, the eigenvectors have a nice geometric interpretation: they can be considered as a local Taylor expansion where one term is defining the limit point, two more terms are spanning the tangent plane (Ball and Storry, 1988), and yet three more define the curvature behavior (*quadratic natural configuration*) (Sabin, 2002). As a consequence the components defining the tangent plane are responsible for C^1 continuity and the curvature components are responsible for C^2 continuity.

For our purposes, we consider a *two-rings* configuration around an EV in a triangular lattice. Let $1, \lambda, \lambda, \mu_c, \mu_s, \mu_s$ be the six largest eigenvalues of the sub-

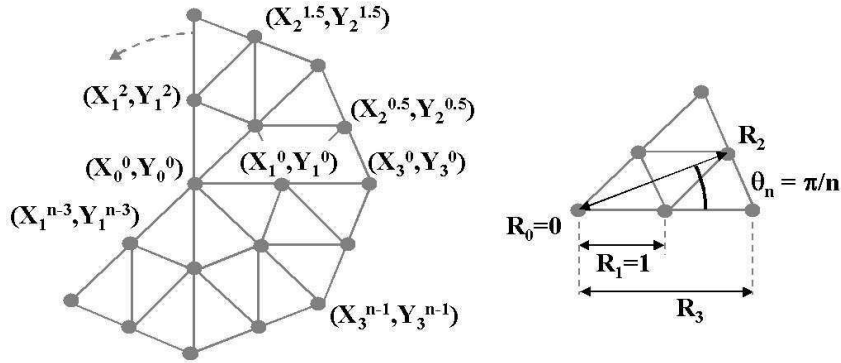


Fig. 1. Local parametrization around a valency n EV when we consider the *two-rings* neighborhood. We point out the special indexing of the second ring vertices used in order to keep notations simple in the equations $X_i^j = \cos(2j\theta_n) R_i$ and $Y_i^j = \sin(2j\theta_n) R_i$. On the left, the vertices defined by the two subdominant eigenvectors \mathbf{X} and \mathbf{Y} . On the right, the minimal set of parameters necessary to fully define the rotationally symmetric parametrization.

division matrix and let $\mathbf{X} = [X_i^j]$ and $\mathbf{Y} = [Y_i^j]$ be two orthogonal eigenvectors for the double eigenvalue λ . In our notations, the lowerscript i is the intra-sector index of the vertices and superscript j is their cyclic index (see Fig. 1). The two vectors \mathbf{X} and \mathbf{Y} induce a local parameterization (*characteristic map*) with respect to which the subdivision scheme reproduces linear polynomials. By exploiting rotational symmetry we can transform the coordinate system $\{\mathbf{X}, \mathbf{Y}\}$ into polar coordinates and it is sufficient to define the vector of radii $\mathbf{R} = [0, R_1, R_2, R_3]$ since \mathbf{R} , \mathbf{X} and \mathbf{Y} are linked by $X_i^j = \cos(2j\theta_n) R_i$ and $Y_i^j = \sin(2j\theta_n) R_i$ with $\theta_n = \pi/n$ and n is the valency of the EV (Fig. 1). The configuration can be scaled such that the first ring is at a radius of unity, i.e. $R_1 = 1$. It is then enough to consider the vector of radii $\mathbf{R} = [0, 1, R_2, R_3]$ in order to define the tangent plane configuration.

The condition $1 > |\lambda| > \max(|\mu_c|, |\mu_s|)$ is necessary for C^1 smoothness at the EV. It is even sufficient if the characteristic map is regular and injective. The subdominant eigenvalue λ defines the *shrinking factor* by which the size of the local configuration is scaled in every subdivision step (Fig. 2(a)). In the optimal setting, the value for $|\lambda|$ equals the shortening factor of the edges caused by the refinement operator, e.g. $|\lambda| = 1/2$ for binary subdivision and $|\lambda| = 1/\sqrt{3}$ for $\sqrt{3}$ subdivision. If the actual $|\lambda|$ happens to deviate significantly from that optimal value, the face shrinkage near the EV is much faster or much slower than elsewhere which leads to the so-called *polar artifact* (Sabin and Barthe, 2002) (Fig. 2(b)).

The conditions for C^2 continuity are much harder to satisfy. Hence one usually restricts to certain necessary conditions which do not imply C^2 but at least guarantee preferable behavior near the EV. First of all one has to make sure

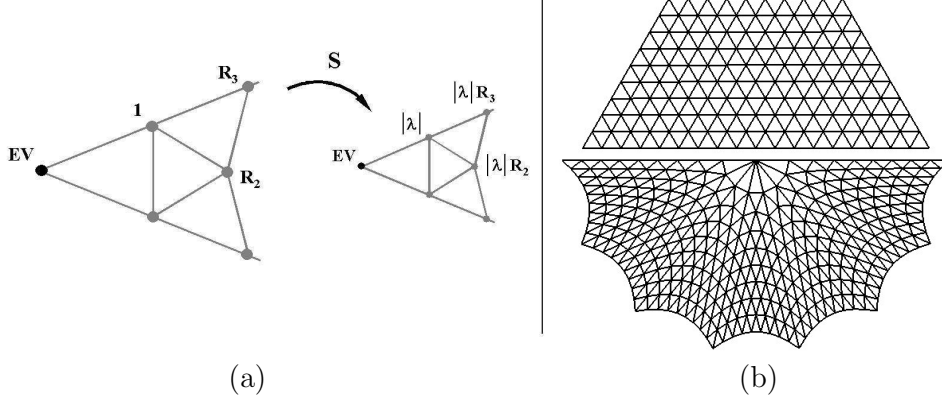


Fig. 2. (a): One sector of the *two-rings* configuration in the tangent plane and the action of the subdivision operator S . (b): Illustration of the polar artifact. The top of the figure has been defined by a regular control mesh and the bottom by regular rings around a valency 14 EV. Both meshes are shown after *three* subdivision steps using the standard Loop's scheme. While the size of the polygons remains the same in the regular case, we can clearly see its distortion in the vicinity of the high valency EV ($|\lambda| \gg 1/2$).

that only three eigenvectors influence the curvature behavior, which is guaranteed if in addition to the C^1 condition $1 > |\lambda| > \max(|\mu_c|, |\mu_s|)$ all other eigenvalues are strictly smaller than $\min(|\mu_c|, |\mu_s|)$. Additionally one would like to have *bounded curvature* (Doo and Sabin, 1978) which requires $|\mu_c| = |\mu_s| = \lambda^2$. *Local quadratic precision* (Gerot et al., 2004) with respect to the characteristic map parameterization follows from the components of the quadratic natural configurations which correspond to the eigenvalues μ_c and the *two* μ_s . We notice that there is no proof that the property here denoted as *local quadratic precision* and which is defined below actually implies the *quadratic precision* of the limit surface in the vicinity of the EV. However as shown by Gerot et al. (2004), the *local quadratic precision* is a necessary condition for C^2 continuity and it is a fundamental criterion that needs to be satisfied in order to improve the scheme's behavior at EVs. The eigenvectors associated with the eigenvalues μ_c and the *two* μ_s correspond to the quadratic polynomials $x^2 + y^2 - Z_c$ (with $Z_c > 0$), $2xy$ and $x^2 - y^2$ respectively whose coefficients are geometrically interpreted as altitudes over the tangent plane defined by \mathbf{X} and \mathbf{Y} . We distinguish the elliptic configuration, denoted as *cup configuration* with the eigenvalue μ_c and the vector of altitudes $\mathbf{C}_{\text{alt}} = [C_i]$ from the *two* hyperbolic configurations, denoted as *saddle configurations* with eigenvalues μ_s and the vectors of altitudes $\mathbf{S}_{\text{alt}}^1 = [\cos(4j\theta_n - \pi/2) S_i]$ and $\mathbf{S}_{\text{alt}}^2 = [\cos(4j\theta_n) S_i]$. The scheme has *local quadratic precision* if in addition to *bounded curvature*, it satisfies $[C_i] = [R_i^2 - Z_c]$ and $[S_i] = [R_i^2]$ (deduced respectively from the conditions $[C_i] = [(X_i^j)^2 + (Y_i^j)^2 - Z_c]$, $\mathbf{S}_{\text{alt}}^1 = [2 X_i^j Y_i^j] = [\cos(4j\theta_n - \pi/2) R_i^2]$ and $\mathbf{S}_{\text{alt}}^2 = [(X_i^j)^2 - (Y_i^j)^2] = [\cos(4j\theta_n) R_i^2]$).

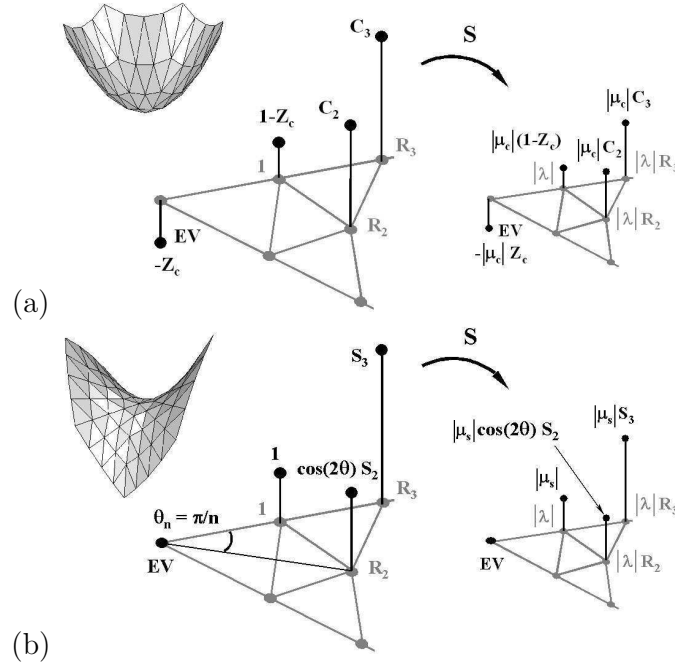


Fig. 3. One sector of the *two-rings* configuration and the action of the subdivision operator S in (a) the cup configuration and (b) a saddle configuration.

In the cup configuration, the central vertex lies at the altitude $-Z_c$ and all the vertices sharing the same ring lie at the same altitude. Hence, this configuration is defined by a vector $\mathbf{C} = [Z_c, C_1, C_2, C_3]$ and once it is scaled such that $C_1 = 1 - Z_c$, it is enough to consider the vector $\mathbf{C} = [Z_c, 1 - Z_c, C_2, C_3]$ (as illustrated in Fig. 3(a)). On the other hand, the *two* saddle configurations are rotationally symmetric for valencies greater than 4, so that they generate identical conditions except for the low valencies 3 and 4 where they have to be considered individually. In the general case (valency > 4), a vertex (X_i^j, Y_i^j) in a saddle configuration lies at the altitude $Z_i^j = \cos(4j\theta_n) S_i$ so that it is enough to consider the vector $\mathbf{S} = [0, 1, S_2, S_3]$ to define the saddle configurations (see Fig. 3(b)). In the special case of valency 4, one saddle configuration is degenerated and in the case of valency 3, both configurations are degenerated. The full study of these configurations remains an unsolved research problem which is outside the scope of this paper. However, in order to address the polar artifact for these low valencies as well, all eigenvectors coming from the degenerated configurations are not considered and the corresponding eigenvalues are set to a value which is strictly less than λ^2 so that the curvature is bounded and the miss-behavior which can be introduced by the degenerated components vanishes in the limit.

3 General approach

In the vicinity of EVs, the subdivision scheme behavior is usually improved by computing a set of stencils' coefficients such that the subdivision matrix has some prescribed eigencomponents in the tangent plane, the cup and the saddle configurations. By doing so, a subdivision scheme with the desired eigenspectrum is locally defined around the EV and subdivision step after subdivision step, it generates new rings that will be later subdivided with the regular rules. In previous work, only the vertices computed using ERs are considered in the analysis, and the irregular triangle fan is considered without taking into account the area where the regular and ERs overlap. As a result, no compatibility is ensured where the influence of these *two* schemes (regular and irregular) overlap and this often produces undesired flatness at the EV and ripples where the rules blend.

The key of our approach is to begin from the regular rules which define the outer rings around the EV and to go from outside in up to the definition of the ERs. This reverse engineering process can be understood by looking at the behavior of the subdivision scheme around a n -sided hole instead of a valency n vertex. Following this idea, we proceed in *three* steps: First, we decide which vertices are to be computed with ERs and which are to be computed with regular rules. We then continue by reformulating the regular subdivision rules in each configuration, in terms of the different eigencomponents. This gives us the set of equations controlling the overlapping area, plus the set of degrees of freedom available to improve the scheme behavior. In the second step, as done with the regular rules, we reformulate the ERs in order to express the ERs' coefficients in terms of the degrees of freedom and in the last step, we apply constraints improving the scheme behavior on the degrees of freedom. The constraints are expressed as a least square problem, which once minimized, provides a set of eigencomponents from which the corresponding ERs' coefficients are computed.

We consider *two* levels of tuning: One where all the subdivision rules but the central one are regular, and the other where the regular rules define only the *two* outer rings (leaving the rule for the *one*-ring and the central EV irregular).

3.1 Reformulation of the regular subdivision rules

As illustrated in Fig. 2(a) and 3, the application of the subdivision operator performs one step of subdivision in the different configurations (tangent plane, cup and saddle), i.e. each component of a subdivided configuration is computed from the original configuration using the suitable subdivision rule. We call *re-*

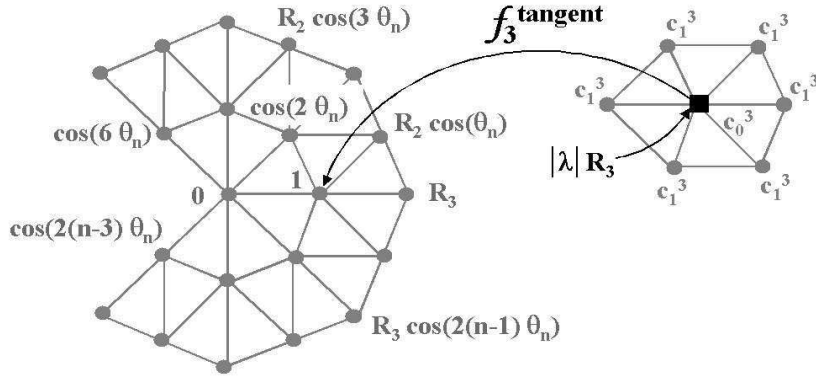


Fig. 4. Illustration of the reformulation of a regular subdivision rule f_3 in the tangent plane configuration in the case of a standard diadic scheme. This regular rule computes $|\lambda|R_3$ (see Fig. 2(a)) in terms of the tangent plane configuration's components: $|\lambda|R_3 = f_3^{\text{tangent}}(R_2, R_3)$.

formulation of a subdivision rule the expression of a subdivided configuration's component as an operator applied to the non-subdivided configuration. The subdivision rules are affine combinations denoted f_i^{conf} if the rule is regular and $\tilde{f}_i^{\text{conf}}$ if it is extraordinary. The subscript i is the intra-sector index of the new vertex generated by the application of the subdivision rule ($i = 0$ for the new central vertex, $i = 1$ for the new first ring vertices, etc) while the superscript index conf denotes the configuration in which the subdivision rule is applied. In Fig. 4 we illustrate the reformulation procedure, and in this example, the reformulation of the regular subdivision rule f_3 in the tangent plane provides the following equation:

$$\begin{aligned} |\lambda|R_3 &= f_3^{\text{tangent}}(R_2, R_3) \\ &= c_0^3 + c_1^3 (2 \cos(2\theta_n) + 2R_2 \cos(\theta_n) + R_3) \end{aligned}$$

We now reformulate the regular subdivision rules in the different configurations and we derive the set of degrees of freedom that we will be able to manipulate in order to tune the scheme at the EV, depending on the selected level of tuning.

3.1.1 Tangent plane configuration

A scheme's configuration in the tangent plane is defined by the subdominant eigenvalue λ and its associated vector of radii \mathbf{R} . In the standard techniques used to derive ERs, only the first ring is considered (Doo and Sabin, 1978; Prautzsch and Umlauf, 1998). This generic configuration has a *one*-dimensional vector $\mathbf{R} = [R_1]$ with $R_1 = 1$ when scaling the parametrization accordingly. Hence, the *one*-ring configuration provides a unique degree of

freedom: The shrinkage factor λ . When ERs are used for both the central vertex and the *one*-ring neighborhood, all the components of the configuration are set by the ERs. However the boundary of the n -sided hole defined by the regular rules is the second ring and since only the *one*-ring configuration is considered for the construction of the ERs, none of its eigencomponents can be accessed. This leaves no control on the area where the regular and the ERs overlap. Therefore, we prefer to consider the *two*-rings generic configuration. It is defined with a vector of radii $\mathbf{R} = [0, 1, R_2, R_3]$ and it provides *three* degrees of freedom which are the shrinkage factor λ and the *two* radii R_2 and R_3 (Fig. 2(a)).

Whatever the chosen level of tuning, we reformulate the regular rules generating the second ring and we obtain the following equations:

$$\begin{aligned} |\lambda|R_3 &= f_3^{\text{tangent}}(R_2, R_3) \\ |\lambda|R_2 &= f_2^{\text{tangent}}(R_2, R_3), \end{aligned}$$

We note that each application of a regular rule eliminates one degree of freedom in the configuration, hence at this stage, only *one* degree of freedom remains in the tangent plane configuration.

One has now to decide which level of tuning is to be used:

- (1) Use an ER only at the EV itself: The regular rule is then reformulated to derive eigencomponents of the first ring:

$$|\lambda| = f_1^{\text{tangent}}(R_2, R_3).$$

It leads to minimum modifications of the original scheme, but the configuration is fully set by the regular rules and it does not provide any degree of freedom.

- (2) Use ERs at the EV and for the first ring: It provides *one* degree of freedom which can be used to give more analytic properties at the EV (bounded curvature,...), but it leads to more modifications of the original scheme.

3.1.2 Cup-like configuration

The behavior of the curvature depends on a cup-like and a saddle-like configuration. We first identify the degrees of freedom given by the cup configuration. Curvature behavior in a cup configuration is defined by the subsubdominant eigenvalue μ_c and its associated vector of altitudes \mathbf{C} . We mainly apply the same procedure as the one used for the tangent plane, with the difference that the cup eigenvector has an additional component Z_c .

In the *two*-rings generic configuration, \mathbf{C} is *four*-dimensional and equals $[Z_c, 1-$

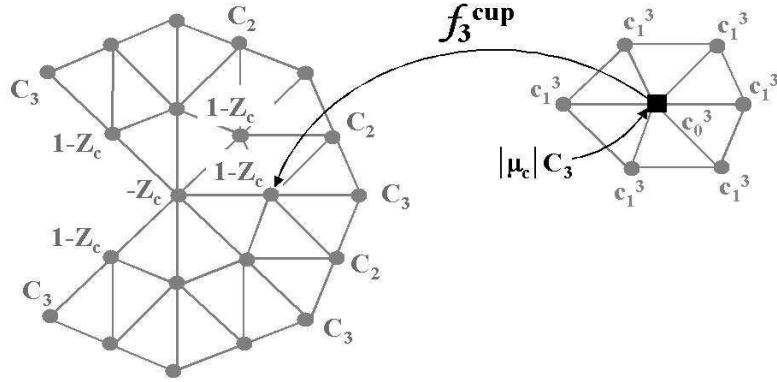


Fig. 5. Illustration of the reformulation of a regular subdivision rule f_3 in the cup configuration in the case of a standard diadic scheme. This rule computes $|\mu_c|C_3$ (see Fig. 3(a)) in terms of the cup configuration's components: $|\mu_c|C_3 = f_3^{cup}(Z_c, C_2, C_3) = c_0^3(1 - Z_c) + c_1^3(-Z_c + 2(1 - Z_c) + 2C_2 + C_3)$.

Z_c, C_2, C_3]. Hence, the configuration provides *four* degrees of freedom which are the shrinkage factor μ_c , the displacement Z_c and the altitudes C_2 and C_3 (Fig. 3(a)). As in the tangent plane, the compatibility of the ERs and the regular rules have also to be guaranteed. It is done by reformulating the regular subdivision rules affecting the second ring as illustrated in Fig. 5 and it provides the following equations:

$$\begin{aligned} |\mu_c|C_3 &= f_3^{cup}(Z_c, C_2, C_3) \\ |\mu_c|C_2 &= f_2^{cup}(Z_c, C_2, C_3). \end{aligned}$$

The *two* levels of tuning yield the following degrees of freedom:

- (1) The regular rule is reformulated to derive the eigencomponents of the first ring:

$$|\mu_c|(1 - Z_c) = f_1^{cup}(Z_c, C_2, C_3),$$

leading us to *three* equations for *four* unknowns. Hence, unlike in the tangent plane, the configuration provides *one* degree of freedom.

- (2) In this case, there are *two* degrees of freedom left.

3.1.3 Saddle-like configuration

Curvature behavior in a saddle configuration is defined by the subsubdominant eigenvalue μ_s and its associated vector of altitudes \mathbf{S} . This configuration and its treatment is very similar to the tangent plane configuration. In the *two-rings* generic configuration, the vector \mathbf{S} equals $[0, 1, S_2, S_3]$ and the con-

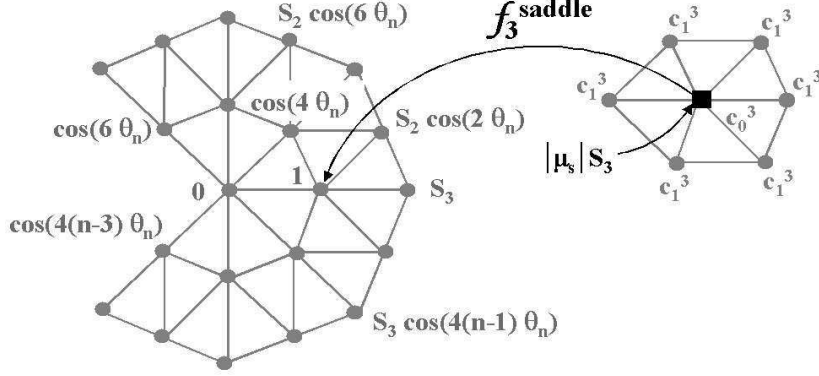


Fig. 6. Illustration of the reformulation of a regular subdivision rule f_3 in the saddle configuration in the case of a standard diadic scheme. This rule computes $|\mu_s|S_3$ (see Fig. 3(b)) in terms of the saddle configuration's components: $|\mu_s|S_3 = f_3^{saddle}(S_2, S_3) = c_0^3 + c_1^3 (2 \cos(4\theta_n) + 2S_2 \cos(2\theta_n) + S_3)$.

figuration provides *three* degrees of freedom which are the shrinkage factor μ_s and the altitudes S_2 and S_3 (Fig. 3(b)).

In this configuration again, we have to guarantee the compatibility between the ERs and the regular rules. Hence we reformulate the regular subdivision rules in terms of the degrees of freedom of the second ring as illustrated in Fig. 6. This gives us:

$$\begin{aligned} |\mu_s|S_3 &= f_3^{saddle}(S_2, S_3) \\ |\mu_s|S_2 &= f_2^{saddle}(S_2, S_3). \end{aligned}$$

The *two* levels of tuning yield the following degrees of freedom:

- (1) The regular rule is reformulated to derive the eigencomponents of the first ring:

$$|\mu_s| = f_1^{saddle}(S_2, S_3),$$

removing the last of the *three* original degrees of freedom.

- (2) When the first ring is free, there is *one* degree of freedom left.

3.2 Derive irregular rules

In order to provide a scheme whose subdivision matrix is expressed in terms of the degrees of freedom given by the different configurations (tangent plane, cup and saddle), we reformulate the ERs in terms of the eigencomponents of each configuration. When we use the ER only at the EV, the rule only affects

the cup configuration (since for the other eigenvectors, the center component is *zero*) and the configurations provide a single degree of freedom which is μ_c . The reformulation of the ER \tilde{f}_0 in terms of the cup's eigencomponents gives us the following equation:

$$-|\mu_c|Z_c = \tilde{f}_0^{cup}(Z_c, C_2, C_3), \quad (1)$$

from which we derive (we use Maple to do this) a coefficient c_0^0 of the ER \tilde{f}_0 expressed in terms of the unique degree of freedom μ_c :

$$c_0^0 = c_0^0(\mu_c).$$

When we use ERs at both the EV and the first ring, the configurations give us a set of *four* degrees of freedom denoted D . The set D has to be chosen with *one* degree of freedom coming from the tangent plane configuration, *two* coming from the cup and *one* coming from the saddle configuration. For instance, D can be set as $D = \{\lambda, \mu_c, Z_c, \mu_s\}$. The first-ring ER \tilde{f}_1 affects every configuration, and its reformulation in each of them gives us the following equations:

$$|\lambda| = \tilde{f}_1^{tangent}(R_2, R_3) \quad (2)$$

$$|\mu_c|(1 - Z_c) = \tilde{f}_1^{cup}(Z_c, C_2, C_3) \quad (3)$$

$$|\mu_s| = \tilde{f}_1^{saddle}(S_2, S_3). \quad (4)$$

This system of *three* equations allows us to express up to *three* coefficients c_i^1 ($i = 0..N_c - 1$, $N_c \leq 3$) of the ER \tilde{f}_1 in terms of the degrees of freedom D :

$$c_i^1 = c_i^1(D), \quad i = 0..N_c - 1.$$

The introduction of *three* equations (2, 3, 4) removes *three* degrees of freedom from D . However, at the same time that they remove degrees of freedom, these equations introduce the coefficients of the ER as new unknowns, and for each of them, one degree of freedom is reinserted in the system. Since the ER's coefficients are expressed in terms of the degrees of freedom D , once the ERs are reformulated, we obtain a final set D of degrees of freedom whose dimension is $N_c + 1$ (where N_c is the number of coefficients provided by the ER).

The ER \tilde{f}_0 displacing the EV acts only on the eigenvector corresponding to the cup configuration and the rule is reformulated with equation (1) in order to express a coefficient c_0^0 of the ER as:

$$c_0^0 = c_0^0(D).$$

All the eigencomponents of the *three* configurations are now expressed in terms of the final degrees of freedom D using the equations presented in Sections 3.1.1, 3.1.2 and 3.1.3. This concludes the second step of our procedure and we can now constrain the eigencomponents in order to impose more analytic properties to the scheme.

3.3 Minimization process

3.3.1 Analytic properties

Since we want to increase the quality of the scheme's curvature and reduce the polygon distortions around the EV, we propose to provide the scheme with the most of the following properties: bounded curvature (for cup “*bcc*” and saddle “*bcs*”), local quadratic precision (for cup “*lqpc*” and saddle “*lqps*”) and no polar artifact “*pa*”. These properties are detailed in Section 2. The generic configurations do not provide enough degrees of freedom to satisfy all these properties and in order to satisfy the most important ones while taking the others into account, we propose to use a least square minimization:

$$F = p_{bcc}F_{bcc} + p_{bcs}F_{bcs} + p_{lqpc}F_{lqpc} + p_{lqps}F_{lqps} + p_{pa}F_{pa}, \quad (5)$$

where map F is the function to minimize, maps F_x measure the failure to meet the property x and scalars p_x set their priority in the minimization process. Even though the minimization process is a heuristic approach which does not provide theoretical insight into the structure of the space of possible solution and into how the tuning of one property affects the others, it has the important advantage of providing a solution even when an exact approach fails to yield provable properties. The maps F_x are defined as follows (see Section 2 for the value of *Polar* in Eq. (6)):

$$\begin{aligned} F_{bcc} &= (\mu_c - \lambda^2)^2 \\ F_{bcs} &= (\mu_s - \lambda^2)^2 \\ F_{qpc} &= (C_2 + Z_c - R_2^2)^2 + (C_3 + Z_c - R_3^2)^2 \\ F_{qps} &= (S_2 - R_2^2)^2 + (S_3 - R_3^2)^2 \\ F_{pa} &= (\lambda - Polar)^2 \end{aligned} \quad (6)$$

When we use the ER at the EV, the single degree of freedom available is μ_c . Hence, we can only manipulate the components of the cup configuration and it is natural to bound the cup component of the curvature. This is done by setting $p_{bcc} = 1$ and all the other priorities $p_x = 0$. In this case, our geometric

construction leads us to the same freedom and the same analytic solutions as the standard methods used to derive the ERs.

When the ERs are used at both EV and first ring, we have more freedom and more tuning can be performed on the scheme by choosing different values for the priorities p_x .

3.3.2 Constraints

The analytic properties represented in map F (Eq. (5)) are focussed on the improvement of the subdivision scheme behavior. They do not ensure the coherent organization of the vertices in the characteristic map, the variation diminishing property, the convex hull property and the correct ordering of the eigenvalues according to the blocks in the Fourier transform of the subdivision matrix.

These *four* fundamental properties are introduced in our minimization process, adding some constraints which are formulated using “barrier” functions B_x (well known in the litterature as *exponential penalty functions* (Alvarez and Cominetti, 2002)). Our barrier functions are expressed as follows:

$$B_x = a_x \exp(-b_x(X - L_x)), \quad a_x > 0,$$

where X is the constrained variable, L_x is its minimal value if b_x is positive and its maximal value if b_x is negative. The amplitude of coefficients a_x and b_x control the barrier sharpness (in our application we use $a_x = 1/2$ and $b_x = \pm 500$).

To ensure a coherent organization of the vertices in the characteristic map and to avoid axial oscillations, we constrain the eigenvector components of the tangent plane as follows (n is the valency of the EV):

$$R_2 > \cos\left(\frac{\pi}{n}\right), \quad R_3 > 1,$$

The variation diminishing property is taken into account by preferring monotonic (radially decreasing) basis functions associated with the EV. We express this constraint by ensuring that the influence of the central EV on the new vertices decreases with increasing radius. We denote as c_0^j the coefficient of the j^{th} ring subdivision rule which weighs the EV ($j = 0$ corresponds the EV subdivision rule). These coefficients are sorted as follows:

$$c_0^0 > c_0^1 > c_0^2.$$

The convex hull property is satisfied if all ER's coefficients are positive. Subdivision rules are affine combinations and hence, the coefficients of an ER must sum up to unity. This is automatically ensured when the $N_c + 1$ coefficients of the j^{th} ER are defined by the set of free coefficients $\{c_i^j\}$, $i = 0..N_c - 1$ and the $(N_c + 1)^{th}$ coefficient is set to $1 - \sum_i c_i^j$. The conditions for the convex hull property are then:

$$c_i^j > 0 \quad \text{for} \quad i = 0..N_c - 1 \quad \text{and} \quad 1 - \sum_i c_i^j > 0.$$

Furthermore, we have to provide a correct ordering of the eigenvalues. We can directly constrain the eigenvalues of the different configurations: $1 > |\lambda| > \max(|\mu_c|, |\mu_s|)$, but it is more complicated to guarantee that $\min(|\mu_c|, |\mu_s|)$ is greater than any other eigenvalue (which is not λ and 1). To be rigorous, we have to analyse the subdivision matrix in the frequency domain (Ball and Storry, 1988; Stam, 1998) in order to derive the equations of all the different eigenvalues for the different valencies of the EV.

However, the largest eigenvalue is in general one of the dominant eigenvalues of the different frequency blocks and they can be computed using the interpretation presented in Section 2 for the tangent plane and saddle configurations. At each pair of frequencies $\pm\omega$ ($0 < |\omega| \leq n/2$) corresponds *two* rotationally symmetric generic configurations, so that we can restrict ω to the frequencies $0 < \omega \leq n/2$. For a frequency ω , the configuration has the dominant eigenvalue λ^ω and the associated vector $V^\omega = [0, 1, \cos(2\omega\theta_n) V_2^\omega, V_3^\omega]$, e.g. the tangent plane configuration corresponds to $\omega = \pm 1$ and the saddle configurations correspond to $\omega = \pm 2$. For $\omega > 2$, the eigenvalues have to be expressed in terms of the degrees of freedom D in order to be compatible to our formulation of the minimization process. This is done by solving the following system of equations:

$$|\lambda^\omega| V_2^\omega = f_2^\omega(V_2^\omega, V_3^\omega) \tag{7}$$

$$|\lambda^\omega| V_3^\omega = f_3^\omega(V_2^\omega, V_3^\omega) \tag{8}$$

$$|\lambda^\omega| = \tilde{f}_1^\omega(V_2^\omega, V_3^\omega), \tag{9}$$

and this leads us to the following conditions to provide a correct ordering of the eigenvalues:

$$|\lambda^\omega| < \min(|\mu_c|, |\mu_s|), \quad \omega = 3..n/2.$$

We strongly recommend to try these conditions first, because it avoids the computation of the subdominant and the subsubdominant eigenvalues (for all valencies) which is in general very complex due to the large matrices produced

by the *two rings* configuration and the free coefficients of the ERs. In our practical examples, these conditions always ensured adequate results.

Other constraints can be combined, like for example bound the components of the cup and saddle eigenvectors, etc.

All the inequalities presented above are formulated in terms of “barrier” functions B_x and their sum gives us the penalty function B to add in the minimization process. The final function F_{final} to minimize in terms of the degrees of freedom is then:

$$F_{final} = F + B.$$

The minimization of F_{final} provides us the values for the degrees of freedom in D and we just have to plug them into the equations defining the coefficients c_i^j to obtain the ERs.

4 Practical examples

4.1 Loop’s subdivision

We now apply our generic procedure to Loop’s subdivision (Loop, 1987) because this scheme is widely used and any improvement has immediate practical applications. As pointed out in Section 3.2, in the configuration where only the rule for the EV is used, our approach leads to the same results as the standard ones. For this reason, we focus on the configuration where the rules for both EV and first-ring are irregular. We remind that Loop’s scheme does not generate any lattice rotation through subdivision and hence, the eigenvalues of the subdivision matrix are real.

In the tangent plane, the generic configuration and the different subdivision rules are shown in Fig. 7. The reformulation of the regular subdivision rules (Section 3.1.1) gives us:

$$\begin{aligned}\lambda R_3 &= 1/16 (10 + 2 \cos(2\theta) + 2R_2 \cos(\theta) + R_3) \\ \lambda R_2 &= 1/16 (12 \cos(\theta) + 2R_2),\end{aligned}$$

where n is the valency of the EV and $\theta = \pi/n$.

The reformulation of the regular rules in the cup and the saddle configurations is done in the same maner (Sections 3.1.2 and 3.1.3) and it provides the set

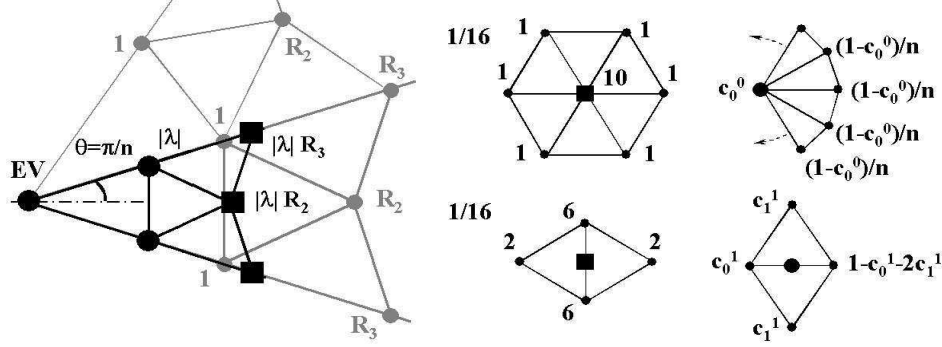


Fig. 7. On the left, one sector of the *two-rings* configuration in the tangent plane and the action of one step of subdivision, in the case of Loop's subdivision. The components derived from the regular rules are drawn with squares, and those derived from the ERs are drawn with circles. On the right, the different regular and irregular subdivision rules.

of equations:

$$\begin{aligned}\mu_c C_3 &= 1/16 (-Z_c + 12(1 - Z_c) + 2C_2 + C_3) \\ \mu_c C_2 &= 1/16 (-2Z_c + 12(1 - Z_c) + 2C_2) \\ \mu_s S_3 &= 1/16 (10 + 2 \cos(4\theta) + 2S_2 \cos(2\theta) + S_3) \\ \mu_s S_2 &= 1/16 (12 \cos(2\theta) + 2S_2).\end{aligned}$$

At this stage, the *three* generic configurations give us a set D of *four* degrees of freedom which can be chosen as $D = \{\lambda, \mu_c, Z_c, \mu_s\}$.

We now look at the ERs' coefficients (following the procedure presented in Section 3.2). The reformulation of the ER applied on the first ring gives us the *three* equations (from Eq. (2,3,4)):

$$\begin{aligned}\lambda &= 2c_1^1 \cos(2\theta) + (1 - c_0^1 - 2c_1^1) \\ \mu_c(1 - Z_c) &= -c_0^1 Z_c + (2c_1^1 + (1 - c_0^1 - 2c_1^1))(1 - Z_c) \\ \mu_s &= 2c_1^1 \cos(4\theta) + (1 - c_0^1 - 2c_1^1),\end{aligned}$$

and the reformulation of the ER applied to the central vertex gives us (Eq. (1) in Section 3.2):

$$-\mu_c Z_c = -c_0^0 Z_c + (1 - c_0^0)(1 - Z_c).$$

The first-ring rule has *two* different coefficients (Fig. 7), hence *three* degrees of freedom are removed by the equations and *two* are restored by the coefficients

c_0^1 and c_1^1 : The final dimension of D is *three* and it can be chosen as $D = \{\lambda, \mu_c, \mu_s\}$. All coefficients and eigencomponents of the different configurations are expressed in terms of the *three* degrees of freedom D . Before writing the function F which will be minimized, the system of equations controlling the ordering of the eigenvalues have to be solved (Eq. (7),(8),(9) in Section 3.3.2). For Loop's scheme, we obtain a simple closed form solution for λ^ω which is:

$$\lambda^\omega = 2c_1^1 \cos(2\omega\theta) + (1 - c_0^1 - 2c_1^1).$$

From this equation, $|\lambda^\omega|$ is expressed in terms of the degrees of freedom D for $\omega = 3..n/2$, and the scheme is ready to be tuned.

One has now to choose the value of map F priorities p_x in order to decide which analytic properties have to be satisfied. The map F_{final} is minimized and from the obtained values of the *three* degrees of freedom of D , we directly derive the ERs' coefficients.

Since we have *three* degrees of freedom, we cannot expect to be able to set all bounded curvature, quadratic precision and no polar artifact. However an improved behavior of the scheme with respect to curvature or to polar artifact can reasonably be expected. We present two different sets of priorities p_x :

$$\begin{aligned} p_{bcc} = 100, p_{bcs} = 100, p_{lqpc} = 0.01, p_{lqps} = 0.01, p_{pa} = 0 \\ p_{bcc} = 10, p_{bcs} = 1, p_{lqpc} = 0.00001, p_{lqps} = 0.000001, p_{pa} = 100 \end{aligned}$$

The first is focussed on the improvement of the curvature and allows us to provide a scheme with bounded curvature and significantly improved saddle and cup eigenvectors (Fig. 10 left). The second minimizes the polar artifact, while maintaining the cup bounded curvature (Fig. 8 and 9). The ERs' coefficients corresponding to these *two* set of priorities are respectively given in Table 1 and 2. The automatic tuning procedure is not very sensitive to slight changes in the priorities (we use powers of 10 values) and depending on the desired analytic properties, one can easily generate different versions of the scheme. For instance, Fig. 10 (right) illustrates rules optimizing the curvature while providing a reduced polar artifact.

This practical implementation revealed *two* side effects that have to be taken into account. The first comes from EVs having a valency less than 5. As noticed in Section 2, they do not behave like the other valencies and specific priorities have to be used. Here is presented a set of priorities p_x tuned to reduce the polar artifact for respectively valency 4 and 3 EVs:

$$p_{bcc} = 10, p_{bcs} = 0.1, p_{lqpc} = 0.1, p_{lqps} = 0, p_{pa} = 100$$

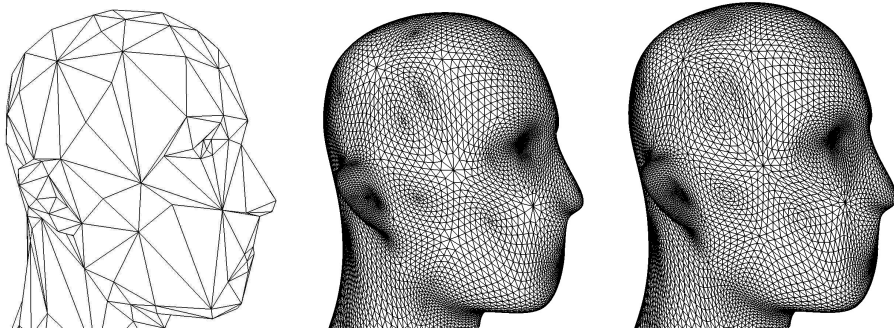


Fig. 8. Illustration of the triangle distortion improvement. On the left the original head model. In the center the effect of the standard Loop's scheme and on the right the effect of our rules reducing the polar artifact after 3 steps of subdivision.

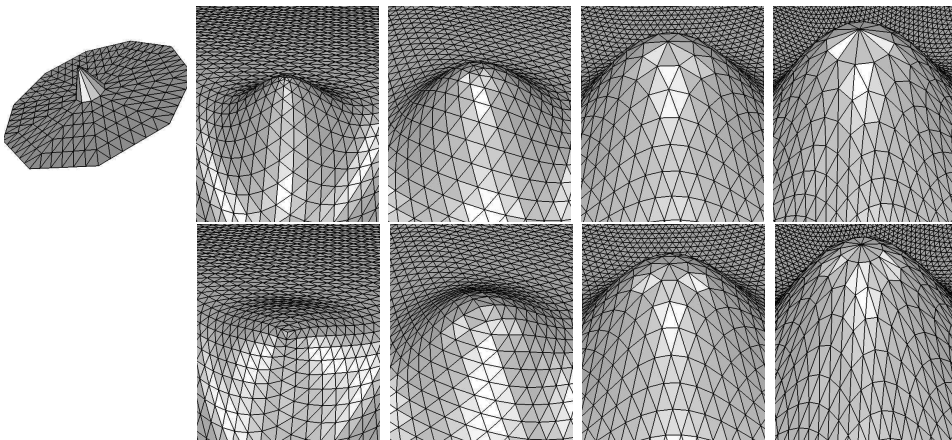


Fig. 9. Reduction of the polygon distortions illustrated on the basis functions of Loop's subdivision. The first row shows the standard Loop's scheme basis functions and the second row the ones of our version reducing the polar artifact. From left to right: Valencies 3, 4, 8, and 12.

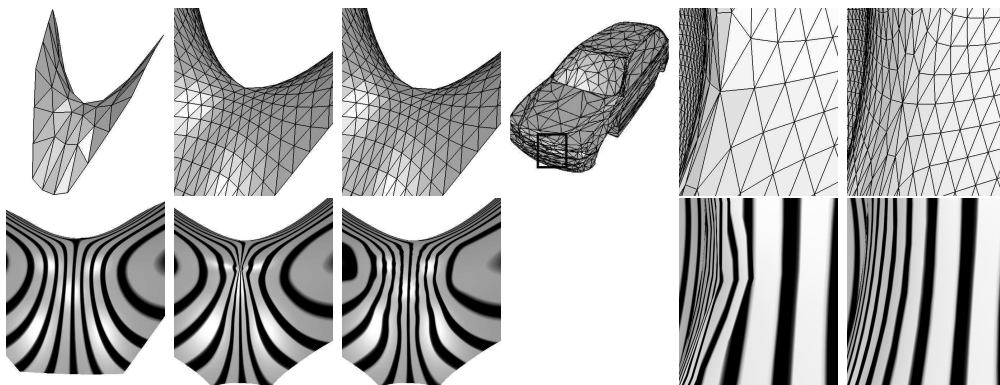


Fig. 10. Illustration of the *curvature* improvements. On the left hand side, a pure saddle control mesh with a central valency 10 EV (top, left) is subdivided using standard Loop's scheme (2nd column) and using our rules fully optimized for curvature (3rd column). The ideal curvature lines generated by a C^2 continuous scheme (bottom, left) allow us to notice the improvement brought by our rules in the curvature (bottom row) while preserving the shape (top row). On the right, a decimated car model is used to show how a rule tuned to improve the curvature while reducing the polar artifact corrects some bad behavior of the standard Loop's scheme (closeup on a feature situated near the radiator grills).

Table 1

Our Loop's subdivision ERs fully optimized to improve the curvature. The 2nd and 3rd columns provide the subdivision rule's coefficients for the central EV and the 4th, 5th and 6th columns provide the coefficients to compute the new *one*-ring vertices (see Fig. 7).

| Valency n | c_0^0 | $(1 - c_0^0)/n$ | c_0^1 | c_1^1 | $1 - c_0^1 - 2c_1^1$ |
|-------------|---------|-----------------|---------|---------|----------------------|
| 3 | 0.32517 | 0.22494 | 0.15658 | 0.14427 | 0.55488 |
| 4 | 0.50033 | 0.12492 | 0.26721 | 0.12495 | 0.48288 |
| 5 | 0.59464 | 0.081072 | 0.33539 | 0.11252 | 0.43957 |
| 6 | 0.625 | 0.0625 | 0.375 | 0.125 | 0.375 |
| 7 | 0.63903 | 0.051567 | 0.36909 | 0.14673 | 0.33745 |
| 8 | 0.67821 | 0.040224 | 0.25579 | 0.16074 | 0.42273 |
| 9 | 0.6866 | 0.034823 | 0.2521 | 0.18939 | 0.36911 |
| 10 | 0.69248 | 0.030752 | 0.24926 | 0.2222 | 0.30633 |
| 11 | 0.69678 | 0.027566 | 0.24706 | 0.25894 | 0.23506 |
| 12 | 0.70014 | 0.024988 | 0.2452 | 0.29934 | 0.15612 |

Table 2

Our Loop's subdivision ERs reducing the polar artifact.

| Valency n | c_0^0 | $(1 - c_0^0)/n$ | c_0^1 | c_1^1 | $1 - c_0^1 - 2c_1^1$ |
|-------------|---------|-----------------|---------|---------|----------------------|
| 3 | 0.32517 | 0.22494 | 0.15658 | 0.14427 | 0.55488 |
| 4 | 0.49954 | 0.12511 | 0.25029 | 0.12524 | 0.49923 |
| 5 | 0.59549 | 0.080902 | 0.34547 | 0.11182 | 0.43088 |
| 6 | 0.625 | 0.0625 | 0.375 | 0.125 | 0.375 |
| 7 | 0.63873 | 0.051609 | 0.38877 | 0.14771 | 0.3158 |
| 8 | 0.64643 | 0.044196 | 0.39644 | 0.1768 | 0.24995 |
| 9 | 0.65127 | 0.038748 | 0.40132 | 0.21092 | 0.17684 |
| 10 | 0.67358 | 0.032642 | 0.42198 | 0.20354 | 0.17094 |
| 11 | 0.68678 | 0.028475 | 0.43423 | 0.20505 | 0.15566 |
| 12 | 0.69908 | 0.025077 | 0.44579 | 0.19828 | 0.15765 |

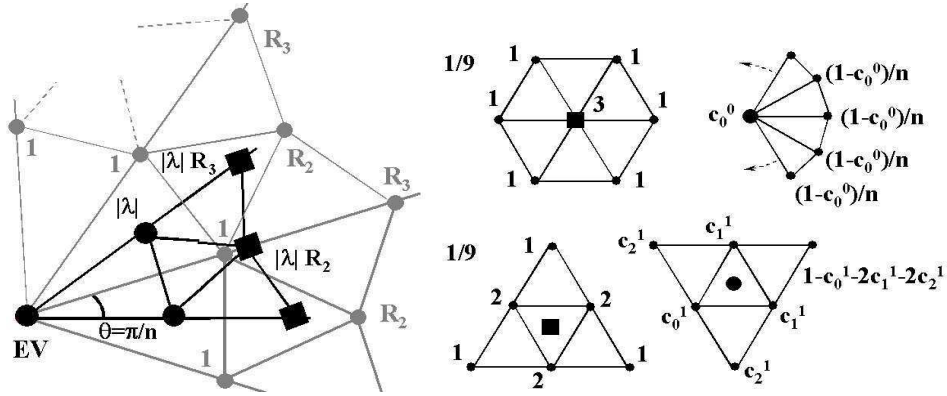


Fig. 11. On the left, one sector of the *two-rings* configuration in the tangent plane and the effect of one step of subdivision, in the case of the 6-direction Box-spline $\sqrt{3}$ subdivision. The components derived from the regular rules are drawn with squares, and those derived from the ERs are drawn with circles. On the right, the different regular and ERs.

$$p_{bcc} = 100, p_{bcs} = 0, p_{lqpc} = 0.001, p_{lqps} = 0, p_{pa} = 10$$

The second is the generation of a flat limit surface at the EV when we try to set bounded curvature without preventing the values of the saddle eigenvector components to become too small (with respect to R_2^2 and R_3^2). If bounded curvature is not a major priority, the flatness can be avoided by increasing the priority of the saddle quadratic precision and decreasing the one of either the saddle bounded curvature or the polar artifact.

4.2 6-direction Box-spline $\sqrt{3}$ subdivision

The regular rules of the 6-direction Box-spline $\sqrt{3}$ subdivision have been suggested by Ron (2000) (see Fig. 11, middle for the regular subdivision rules). This scheme is C^3 continuous in the regular case but due to its complex eigenvalues and its large support, it is very difficult to analyse, and so far, no ER have been proposed. Our procedure handles this scheme in the same way as Loop's scheme (Section 4.1) and provides practical subdivision rules.

We first derive subdivision rules in the case where only the rule for the EV is irregular. This is done using the procedure explained in Section 3. It is applied in the configuration shown in Fig. 11 with the rule for the first ring regular instead of irregular. It leads us to the standard level of tuning where we only can set the cup bounded curvature ($\mu_c = \lambda^2$). As illustrated in Fig. 12(a), the generated scheme produces very large polar artifacts. Furthermore, the constraint maintaining the variation diminishing property (Section 3.3.2) prevents

Table 3

Our 6-direction Box-spline $\sqrt{3}$ subdivision ERs. The 2nd and 3rd columns provide the subdivision rule's coefficients for the central EV and the 4th, 5th, 6th and 7th columns provide the coefficients to compute the new *one*-ring vertices (see Fig. 11).

| n | c_0^0 | $(1 - c_0^0)/n$ | c_0^1 | c_1^1 | c_2^1 | $1 - c_0^1 - 2c_1^1 - 2c_2^1$ |
|-----------|---------|-----------------|---------|---------|----------|-------------------------------|
| 3 | 0.18059 | 0.27314 | 0.14431 | 0.18751 | 0.13445 | 0.34621 |
| 4 | 0.1466 | 0.21335 | 0.12566 | 0.30087 | 0.067749 | 0.13711 |
| 5 | 0.29487 | 0.14103 | 0.27127 | 0.14951 | 0.10384 | 0.22204 |
| 6 | 0.33333 | 0.11111 | 0.22222 | 0.22222 | 0.11111 | 0.11111 |
| 7 | 0.44264 | 0.079622 | 0.22615 | 0.23274 | 0.11501 | 0.078362 |
| 8 | 0.49043 | 0.063697 | 0.22446 | 0.22303 | 0.13516 | 0.059169 |
| 9 | 0.56834 | 0.047962 | 0.23609 | 0.23381 | 0.12682 | 0.042655 |
| 10 | 0.61432 | 0.038568 | 0.23904 | 0.23737 | 0.12832 | 0.029582 |
| 11 | 0.64637 | 0.032148 | 0.2396 | 0.23693 | 0.13367 | 0.019188 |
| 12 | 0.67301 | 0.027249 | 0.23895 | 0.23741 | 0.13831 | 0.0096209 |

the scheme to have cup bounded curvature for valencies less than 6 (Fig. 12(c) and 14), producing a totally divergent curvature.

In order to improve the scheme's behavior at EVs, we apply our procedure with both EV and first ring computed with ERs, using the tangent plane configuration shown in Fig. 11. The regular rules reformulation gives us a set D of *four* degrees of freedom (Sections 3.1.1 , 3.1.2 and 3.1.3), and the reformulation of the ERs (Section 3.2) removes *three* degrees of freedom which are restored by the *three* first-ring ER's coefficients (Fig. 11).

This total of *four* degrees of freedom allows us to provide the scheme with bounded curvature and cup quadratic precision for valencies greater than 6 and bounded cup curvature for valencies lower than 6. As illustrated in Fig. 12, 13 and 14, the improved version of the scheme exhibits nice curvature properties and the corresponding ERs' coefficients are given in Table 3. It generates smooth reflection lines (Fig. 15) while maintaining a reduced polar artifact.

5 How to treat the first step of subdivision

When we use ERs to treat both EV and first ring, we can face an ambiguity at the first step of subdivision. This situation, happens when two EVs are

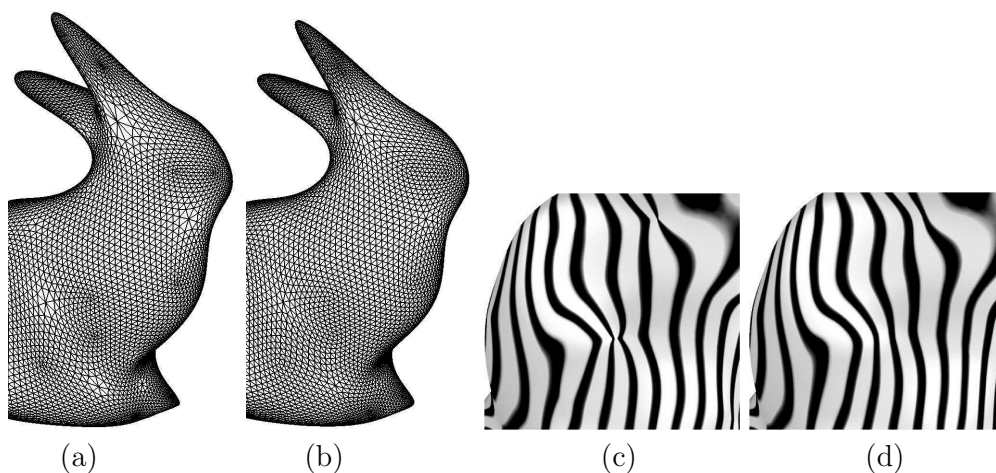


Fig. 12. (a) (respectively (c)) A mesh obtained after 4 (respectively 6) steps of the standard 6-direction Box-spline subdivision. The same mesh using our modified rules illustrates: (b) The improvement in the triangle distortions and (d) the improvement of the curvature (reflection lines).

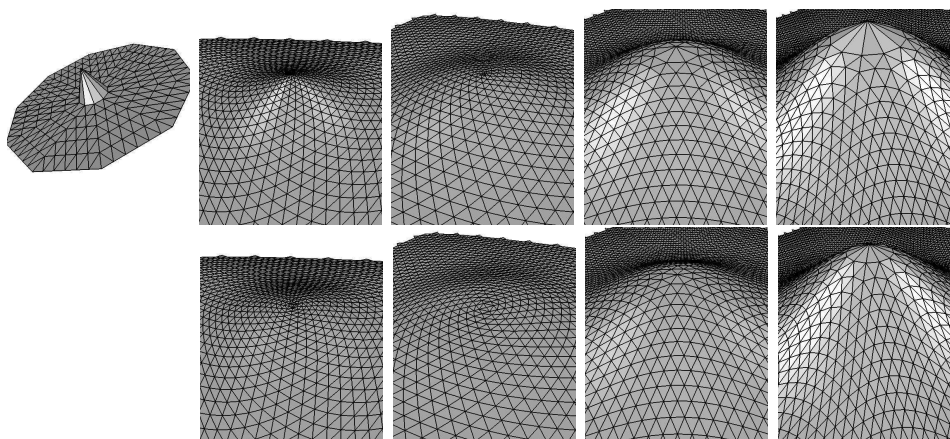


Fig. 13. Reduction of the polygon distortions illustrated on the basis functions of the 6-direction Box-spline subdivision. The first row shows the standard 6-direction Box-spline scheme basis functions and the second row the ones of our version reducing the polar artifact. From left to right: Valencies 3, 4, 8, and 12.

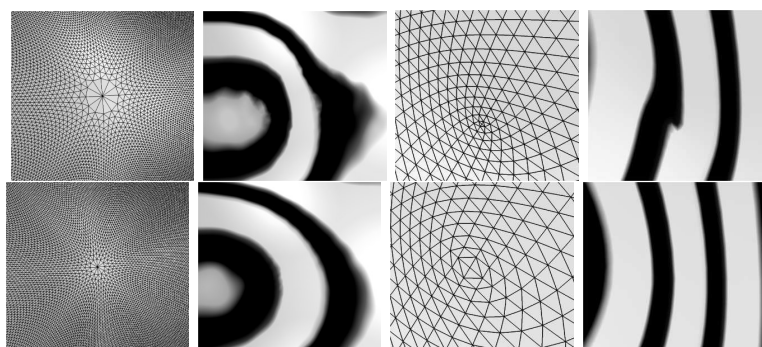


Fig. 14. The first row illustrates *two* meshes and their curvature (shown by the reflection lines) obtained at the vicinity of valencies 12 and 4 EVs after several steps of our standard 6-direction Box-spline subdivision. The second row illustrates the action of our rules improved for curvature on the same meshes.

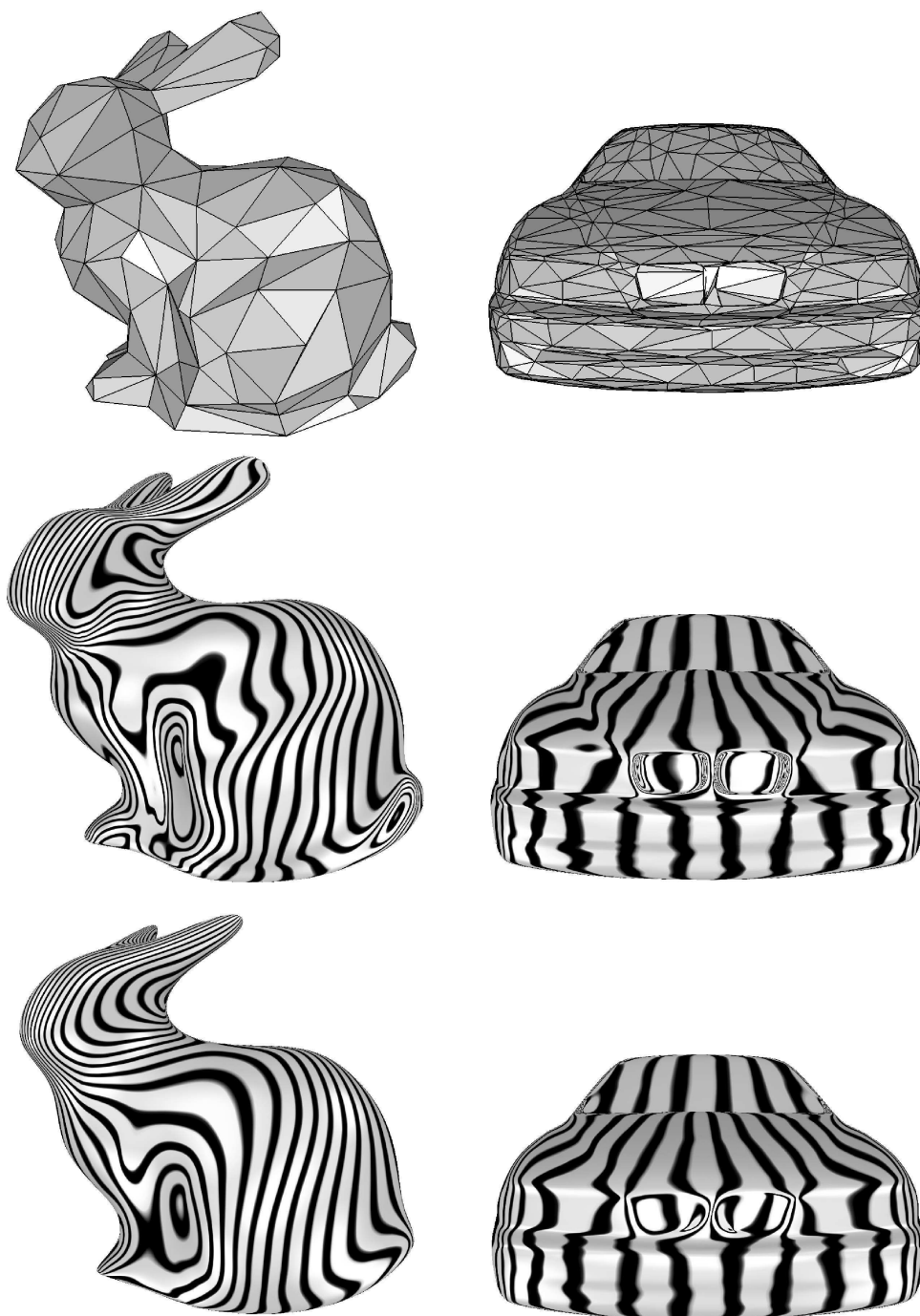


Fig. 15. Comparison of the smoothing properties of our curvature improved schemes. The first row shows the control meshes, the second row shows Loop's subdivision and the third row shows our 6-direction Box-spline $\sqrt{3}$ subdivision. Notice the improved smoothness for the 6-direction Box-spline, even for rather non-uniform control meshes.

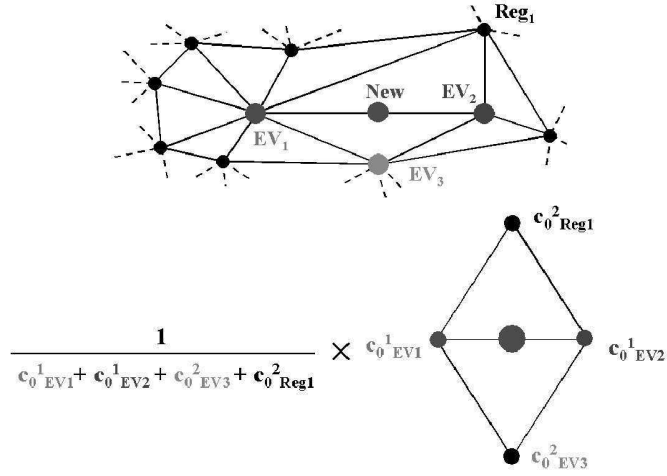


Fig. 16. Example of construction of a Loop-like *one*-ring subdivision rule from the ERs' coefficients specially computed to treat the first step of subdivision. Vertex *New* is the new vertex inserted using the subdivision rule, and the others are the vertices of the original mesh.

adjacent: Which ER should we use to compute vertices of the first ring?

In this special case, we propose to use ERs where only the coefficient c_0^j which corresponds to the EV can be modified. These rules are derived using our procedure by setting the other coefficients of the ERs to the regular values. When an ER is applied, each vertex gives its own coefficient to the subdivision rule, and then the rule is normalised (an example of such a construction is given in Fig. 16). This is an ad-hoc solution but it has the advantage to remove the indeterminacy and to provide more than one degree of freedom already at the first step of subdivision.

6 Conclusion and future work

In this paper, we have presented a generic and heuristic procedure to improve the curvature behavior and the polygon distortions when a mesh is subdivided in the vicinity of an extraordinary vertex. This method is derived from standard techniques and it allows us to automatically compute irregular subdivision rules' coefficients such that the generated subdivision scheme satisfies prescribed analytic properties (bounded curvature, quadratic precision, “no polar artifact”, etc). Two fundamental advantages of our procedure have to be noticed: It allows us to choose which subdivision rules are to be irregular (we chose the rules for the central EV and the one for the *one*-ring vertices to be irregular) and to prescribe both the *foot-print* and the free coefficients of each of them (we prescribed the *foot-prints* of our irregular rules to be

the same as those of the regular rules and all their coefficients are free). We have demonstrated the efficiency of our approach on practical examples: We provide a variant of Loop's subdivision with very little triangle distortions and bounded curvature, and we have shown the robustness of the method through the analysis of the 6-direction Box-spline $\sqrt{3}$ subdivision, providing the scheme with bounded curvature and a reduced polar artifact.

This work leads us to many directions of further investigation such as the exploration of the different scheme tuning possibilities, the treatment of boundaries and creases, the application of our generic procedure on quadrilateral meshes, a deeper study of large support subdivision schemes and additional improvements of the curvature behavior in the overlapping area.

7 Acknowledgments:

We specially thank Dr Malcolm A. Sabin for his always pertinent remarks, discussions and suggestions.

This work has been partially funded by the E.U. through the MINGLE project. Contract HPRN-CT-1999-00117

References

- Alvarez, F., Cominetti, R., 2002. Primal and dual convergence of a proximal point exponential penalty method for linear programming. *Mathematical Programming*, Springer-Verlag Heidelberg, 93 (1), pp. 87-96.
- Ball, A.A., Storry, D.J.T., 1988. Conditions for tangent plane continuity over recursively generated B-spline surfaces. *ACM Transactions on Graphics*, 7 (2), pp. 83-102.
- Catmull, E., Clark, J., 1978. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10 (6), pp. 350-355.
- de Boor, C., Hollig, D., Riemenschneider, S., 1994. *Box Splines*. Springer-Verlag, New York.
- Doo, D., Sabin, M.A., 1978. Analysis of the behaviour of recursive subdivision surfaces near extraordinary points. *Computer Aided Design*, 10 (6), pp. 356-360.
- Dyn, N., 1992. Subdivision schemes in computer-aided geometric design. *Advances in numerical analysis, II, Wavelets, Subdivision Algorithms and Radial Basis Functions*, Clarendon Press, Oxford, pp. 36-104.
- Farin, G., 2002. *CAGD a practical guide*, 5th Edition. Academic Press.
- Gerot, C., Barthe, L., Dodgson, N.A., Sabin, M.A., 2004. Subdivision as a

- sequence of sampled Cp surfaces. Proc. of MINGLE Workshop September 2003, Springer-Verlag, to appear in 2004
- Kobbelt, L., 1996. Interpolatory subdivision on open quadrilateral nets with arbitrary topology. Computer Graphics Forum 15, Eurographics '96 issue, pp. 409 - 420
- Kobbelt, L., 2000. $\sqrt{3}$ -Subdivision. Proceedings of SIGGRAPH 2000, Computer Graphics Proceedings, Annual Conference Series, ACM, pp. 103-112.
- Loop, C., 1987. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah.
- Loop, C., 2002. Bounded curvature triangular mesh subdivision with the convex hull property. The Visual Computer, 18 (5-6), pp. 316-325.
- Peters, J., Reif, U., 1997. The simplest subdivision scheme for smoothing polyhedra. ACM Transaction on Graphics, 16 (4), pp. 420-431.
- Peters, J., Reif, U., 1998. Analysis of algorithms generalizing B-spline subdivision. SIAM Journal of Num. Anal., 35 (2), pp. 728-748.
- Peters, J., Umlauf, G., 2001. Computing curvature bounds for bounded curvature subdivision. Computer Aided Geometric Design, Special issue on subdivision surfaces, 18 (5), pp. 455-461.
- Prautzsch, H., 1998. Smoothness of subdivision surfaces at extraordinary points. Adv. Comp. Math., 14, pp. 377-390.
- Prautzsch, H., Umlauf, G., 1998. A G^2 -subdivision algorithm. Geometric Modelling, Dagstuhl 1996, Computing supplement 13, Springer-Verlag, pp. 217-224.
- Reif, U., 1995. A unified approach to subdivision algorithms near extraordinary vertices. Computer Aided Design, 12, pp. 153-174.
- Ron, A., 2000. Private communication.
- Velho, L., Zorin, D., 2001. 4 – 8 Subdivision. Computer Aided Geometric Design, Special issue on subdivision surfaces, 18 (5), pp. 397-428.
- Sabin, M.A., 2002. Eigenanalysis and artifacts of subdivision curves and surfaces. Tutorial on multiresolution in geometric modelling, Springer, pp. 69-97.
- Sabin, M.A., Barthe, L., 2002. Artifact in recursive subdivision surfaces. Curves and Surfaces 2002 Proceedings, to appear.
- Stam, J., 1998: Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. Proceedings of SIGGRAPH 1998, Computer Graphics Proceedings, Annual Conference Series, ACM, pp. 395–404.
- Stam, J., 2001. On subdivision schemes generalizing uniform B-spline surfaces of arbitrary degree. Computer Aided Geometric Design, Special issue on subdivision surfaces, 18 (5), pp. 383-396.
- Umlauf, G., 1999. Glatte Freiformflächen und optimierte Unterteilungsalgorithmen. PhD thesis, University of Karlsruhe, Germany, in German.
- Warren, J., Weimer, H., 2002. Subdivision methods for geometric design : a constructive approach. San Francisco: Morgan Kaufmann.
- Zorin, D., 1997. Stationary subdivision and multiresolution surface representations. PhD thesis, Caltech, Pasadena, California.

- Zorin, D., Schröder, P., 2000. Subdivision for modeling and animation. SIGGRAPH 2000 Course Notes.
- Zorin, D., 2000. A method for analysis of C^1 -continuity of subdivision surfaces. SIAM Journal of Num. Anal., 35 (5), pp. 1677-1708.
- Zorin, D., Schröder, P., 2001. A unified framework for primal/dual quadrilateral subdivision schemes. Computer Aided Geometric Design, Special issue on subdivision surfaces, 18 (5), pp. 429-454.

7.1.4 Interpolatory Refinement for Real-Time Processing of Point-Based Geometry.

Auteurs : Gaël Guennebaud, Loïc Barthe, Mathias Paulin
Revue : Computer Graphics Forum, Vol 24(3), Proceedings of Eurographics
Date : Septembre 2005

Interpolatory Refinement for Real-Time Processing of Point-Based Geometry

G. Guennebaud and L. Barthe and M. Paulin[†]

IRIT - CNRS - Université Paul Sabatier - Toulouse - France

Abstract

The point set is a flexible surface representation suitable for both geometry processing and real-time rendering. In most applications, the control of the point cloud density is crucial and being able to refine a set of points appears to be essential. In this paper, we present a new interpolatory refinement framework for point-based geometry. First we carefully select an appropriate one-ring neighborhood around the central interpolated point. Then new points are locally inserted where the density is too low using a $\sqrt{3}$ -like refinement procedure and they are displaced on the corresponding curved Point Normal triangle. Thus, a smooth surface is reconstructed by combining the smoothing property produced by the rotational effect of $\sqrt{3}$ -like refinements with the points/normal interpolation of PN triangles. In addition we show how to handle sharp features and how our algorithm naturally fills large holes in the geometry. Finally, we illustrate the robustness of our approach, its real-time capabilities and the smoothness of the reconstructed surface on a large set of input models, including irregular and sparse point clouds.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computational Geometry and Object Modeling]: Curve, surface, solid, and object representations I.3.3 [Computer Graphics]: Viewing algorithms

1. Introduction

Compared with meshes, point-based geometries are connectivity free and no topological consistency has to be satisfied through geometry manipulations. Hence, point sets become increasingly attractive as an alternative surface representation suitable for high quality rendering as well as for flexible processing of complex 3D models [KB04].

Although it is possible to deal with dense cloud of pure points, when performance matters, points are usually enriched by some attributes such as the surface normal and an estimation of the local sampling density. Such a point is commonly represented as an oriented disk and it is called a *splat*. The visualization step is generally performed by a surface splatting based technique [ZPvBG01]: all splats are projected onto the screen and filtered by a Gaussian kernel. Such techniques perform both fast rendering since they are implementable on GPUs [BK03, GP03, ZRB*04] and high quality rendering as long as the screen space size of splats remains small enough, i.e. a few pixels. Indeed, when the point set is not dense or uniform enough, splats radii are large, making the image blurry in the inner part of the object and producing artifacts on the silhouette. “Phong Splatting” like techniques [BK04] significantly improve the inner blur

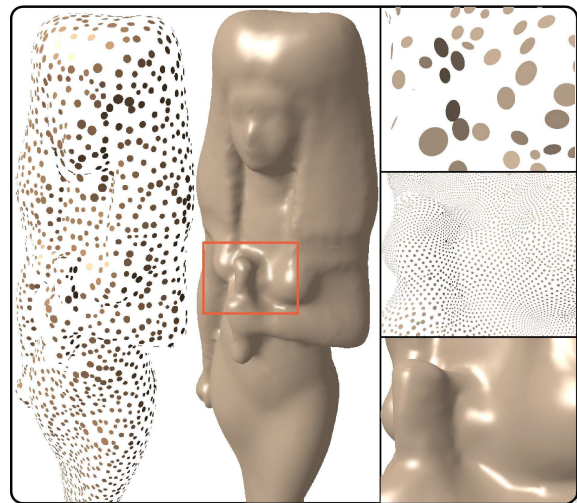


Figure 1: Illustration of the smooth reconstruction capabilities of our refinement procedure on the Isis model irregularly sampled with 3500 points (left). The right images focus on a particularly under-sampled area, from top to bottom: the initial sampling, after four, then six refinement steps.

using second order informations. However these approaches consequently increase the memory consumption, reduce the initial simplicity of points and do not solve the geometric artifacts introduced by under-sampled models.

Thus, the idea of a refinement algorithm maintaining the

[†] e-mail: {guenneba | lbarthe | paulin}@irit.fr

point set with uniform, dense sampling appears to be very relevant, and for this reason, this paper focus on point-based refinement techniques. Ideally, we would like such an algorithm to exhibit the following useful features:

- increase the sampling density,
- regularize scattered sampling,
- converge on a smooth surface,
- fill large holes in the geometry,
- handle boundary and sharp creases.

Related works: Fundamentally, the problem of point cloud refinement can be decompose in two steps: sampling (insertion operators) and displacement (smoothing operators).

In order to reduce or increase its density, the sampling is often controlled by a particle simulation procedure [Tur92, PGK02]. In [ABCO*03], Alexa et al. present an insertion procedure based on a local Voronoï diagram. Even though these two techniques can lead to a locally uniform sampling, their computational cost remains too expensive for real-time applications.

The smoothing issue is related to the problem of displacing the newly inserted points on a smooth surface defined by the original set of points. Most reconstruction techniques are based on implicit representation. For instance, radial basis functions (RBF) reconstruct a C^n implicit surface from a scattered set of point-normals [CBC*01]. Even though the global support of RBFs may be reduced by local approaches [OBA*03, TRS04], preprocessing and surface evaluation remain expensive. In [Lev03], Levin introduces a smooth point-based representation, called moving least-squares (MLS) surface, where the surface is implicitly defined by a local projection operator. Owing to the elegance of the projection idea and the relative locality of the involved computations, MLS surfaces have been used widely in applications ranging from surface editing [PKKG03] and ray-tracing [AA03] to up/down-sampling [PGK02, ABCO*03]. However, the projection operator requires a sufficiently dense input point set. Amenta and Kil [AK04] overcome this limitation by giving an explicit definition of the MLS surfaces which is able to handle a splat-based representation, but however the MLS surface is defined, the projection procedure remains expensive.

In [MMS*04], Moenning et al. present a meshless subdivision framework where mesh connectivity is replaced by intrinsic point proximity information. The subdivision operator is then based on geodesic weighted averages. However, authors simply introduce the concept and it is difficult to evaluate their method.

Targeting the real-time visualization of point clouds, Guennebaud et al. [GBP04] present a fast refinement algorithm of splat-based geometries. The input point set is iteratively refined by selecting a set of neighbors around each point and inserting new points on Bézier curves and Bézier patches. Owing to the extreme locality of these computations, their algorithm is fast enough to generate a million

points per second. However, due to the lack of robustness of the refinement procedure, their algorithm is limited to uniform sampling (otherwise, the refined surface exhibits holes) and even though the surface looks globally smooth, it generates artifacts in the form of high frequency oscillations. This is essentially due to the non homogeneity of the smoothing rules (a mix of bivariate and univariate reconstruction techniques).

Beyond these important limitations, this last work demonstrates the efficiency of a pure point-based refinement algorithm for real-time graphics applications.

Our contribution: In this paper we present a new interpolatory subdivision framework which overcomes the weakness of the Guennebaud et al. refinement procedure [GBP04]. First, in order to reconstruct a **smooth surface** we propose to combine a local and fast smooth surface reconstruction method (based on curved Point-Normal triangles [VPBM01], also called PN triangles) with an iterative $\sqrt{3}$ -like refinement scheme [Kob00]. The important features of PN triangles are their locality and their interpolation power since they interpolate both the points and their normals. For its part, the $\sqrt{3}$ scheme provides a strong global smoothing of the surface due to its rotation effect [Kob00] (figure 7d). Secondly, from the **robustness** point of view, we present two new accurate one-ring neighborhood computations and a new robust insertion procedure which optimizes the sampling uniformity and guarantees the absence of holes. Our new neighborhood computation procedures have the twin advantages that they are naturally symmetric while also being able to handle sparse point clouds through carefully analysis of the closest points. Finally, we show how to refine sharp features and we show that our approach is sufficiently fast and robust to be suitable for both large hole filling (figure 9) and real-time rendering.

2. Overview of our refinement procedure

Let $P^0 = \{\mathbf{p}_i\}$ be the initial point set defining a smooth manifold surface. We assume that we know for each point $\mathbf{p}_i \in P$, its normal \mathbf{n}_i and the local density described by a scalar r_i which must be at least greater than the distance from \mathbf{p}_i to the farthest neighbor of its natural first ring neighborhood.

In a similar fashion to subdivision surfaces, the point set is iteratively refined, leading to a sequence of point set $P^0, P^1, \dots, P^l, \dots$. Since our algorithm is interpolatory, we have $P^l \subset P^{l+1}$ (only the radius of points varies between two steps) and the refined point set P^{l+1} is the union of the set P^l itself and the set of points resulting from the local refinement of each point $\mathbf{p} \in P^l$.

The local refinement of a single point \mathbf{p} requires several operations. First, a convenient one-ring neighborhood N_p of \mathbf{p} is computed (section 4). Next, in order to match with the $\sqrt{3}$ scheme a set of triangles from which it is relevant to insert new points at their center of gravity is extracted from the implicit triangle fan formed by the sorted neighborhood

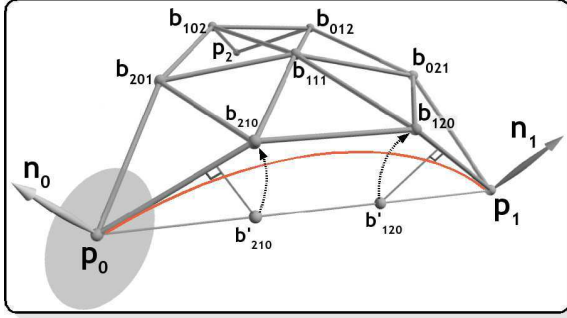


Figure 2: Construction of the control polygon of a PN triangle interpolating three splats.

of \mathbf{p} . These triangles are selected in order to optimize the uniformity of the new neighborhood of \mathbf{p} by taking into account the relative position of neighbors and the new points of $P^{l+1} - P^l$ which have been already inserted (section 5). Finally, in order to obtain a smooth surface, the centers of gravity of the selected triangles are displaced using our smoothing operator (section 3).

The paper is organized as follow: in the next section (section 3) we review the PN triangle technique from which we derive both our smoothing operator and a set of tools which are used in our neighborhood computations and during our local insertion procedure. Sections 4, 5 and 6 are respectively dedicated to the one-ring neighborhood selection, the local refinement of a single point and the refinement of sharp features. In section 7 we give some details on the data structures and the real-time rendering application. Finally, we present our results and discuss the continuity of the limit surface in section 8.

3. Point-Normal Interpolation Framework

In this section we present the set of tools used to extrapolate local geometric informations of the unknown surface S defined by the set of points with their normals. These tools are based on Point-Normal triangles [VPBM01]. A PN triangle is a Bézier triangle $B(u, v)$ of degree three interpolating three points of the point set and their normals.

$$B(u, v) = \sum_{i+j+k=3} \mathbf{b}_{ijk} \frac{3!}{i!j!k!} u^i v^j w^k, \quad w = 1 - u - v \quad (1)$$

Given three points $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$ and their respective normals $\mathbf{n}_0, \mathbf{n}_1, \mathbf{n}_2$, the nine control points \mathbf{b}_{ijk} of the patch (equation 1) are computed as follow (figure 2):

1. The three extremities $\mathbf{b}_{300}, \mathbf{b}_{030}, \mathbf{b}_{003}$ are respectively $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$.
2. The positions of the six boundary control points ($\mathbf{b}_{ijk}, i + j + k = 3, i \neq j \neq k$) only depend on the two extremities of their respective boundary and they are all computed in the same manner. For instance, the control point \mathbf{b}_{210} is the projection of $\mathbf{b}'_{210} = \mathbf{p}_0 + \frac{1}{3}\mathbf{p}_0\mathbf{p}_1$ onto the tangent plane of \mathbf{p}_0 , moved such that the length of the vector

$\mathbf{p}_0\mathbf{b}_{210}$ is equal to the third of the distance between the two extremities \mathbf{p}_0 and \mathbf{p}_1 . Let $Q_i(\mathbf{x})$ be the orthogonal projection operator, projecting the point \mathbf{x} onto the tangent plane of \mathbf{p}_i , then:

$$Q_i(\mathbf{x}) = \mathbf{x} + (\mathbf{p}_i - \mathbf{x}) \cdot \mathbf{n}_i * \mathbf{n}_i \quad (2)$$

$$\mathbf{b}_{210} = \mathbf{p}_0 + \frac{\|\mathbf{p}_0\mathbf{p}_1\|}{3} * \frac{Q_0(\mathbf{b}'_{210}) - \mathbf{p}_0}{\|Q_0(\mathbf{b}'_{210}) - \mathbf{p}_0\|}$$

3. The central point \mathbf{b}_{111} is set to reproduce quadratic polynomials by taking $\mathbf{b}_{111} = \mathbf{c} + \frac{3}{2}(\mathbf{e} - \mathbf{c})$ where \mathbf{c} is the center of gravity of the three input points and \mathbf{e} is the average of the six boundary control points.

Our construction varies from the one of Vlachos et al. [VPBM01] only in one point. After projection, we displace the boundary points \mathbf{b}_{ijk} ($i + j + k = 3, i \neq j \neq k$) while they do not. This is done to avoid the introduction of flatness in the reconstructed surface, especially in areas of high curvature.

Smoothing Operator

We define the smoothing operator ϕ as the displacement of the center of gravity (*cog*) onto the PN triangle. Thus, the position of the new point \mathbf{p}_{new} is:

$$\mathbf{p}_{new} = cog(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2) + \phi(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)$$

where ϕ is the average of the six tangent vectors t_i :

$$\phi(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2) = \frac{1}{6} \sum_{i=0}^5 t_i \quad (3)$$

with:

$$t_0 = \mathbf{b}_{210} - \mathbf{p}_0, \quad t_1 = \mathbf{b}_{120} - \mathbf{p}_1, \quad t_2 = \mathbf{b}_{021} - \mathbf{p}_1, \dots$$

The normal of the new point is the cross product of the two tangent vectors at the center of the PN triangle ($u = v = \frac{1}{3}$):

$$B^u(\frac{1}{3}, \frac{1}{3}) = 7(\mathbf{p}_1 - \mathbf{p}_0) + \mathbf{b}_{120} - \mathbf{b}_{102} + \mathbf{b}_{012} - \mathbf{b}_{210} + 2(\mathbf{b}_{021} - \mathbf{b}_{201})$$

$$B^v(\frac{1}{3}, \frac{1}{3}) = 7(\mathbf{p}_2 - \mathbf{p}_0) + \mathbf{b}_{102} - \mathbf{b}_{120} + \mathbf{b}_{021} - \mathbf{b}_{201} + 2(\mathbf{b}_{012} - \mathbf{b}_{210})$$

Geodesic Distance

With respect to the Euclidean distance, the geodesic distance is useful for evaluating the relative position of points on a surface. Following our local point-normal surface reconstruction we define the local geodesic distance $\tilde{G}(\mathbf{p}_0, \mathbf{p}_1)$ between two relatively close points \mathbf{p}_0 and \mathbf{p}_1 as the length of a cubic Bézier curve interpolating the two points and their normals. In our construction, this curve is the boundary of a PN triangle (figure 2). However the exact computation of the length of a Bézier curve is too expensive for our purpose. We rather use a sufficient approximation given by the length of the control polygon:

$$\tilde{G}(\mathbf{p}_0, \mathbf{p}_1) = \frac{2}{3} \|\mathbf{p}_0\mathbf{p}_1\| + \|b_{210} - b_{120}\| \quad (4)$$

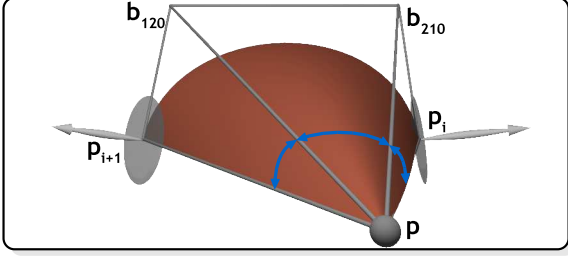


Figure 3: Our “curved angle” between two point-normals $\mathbf{p}_i, \mathbf{p}_{i+1}$ relatively two a third point \mathbf{p} is specially useful for areas of high curvature. On this example there is a ratio of two between the geometric angle and our “curved angle”.

“Curved-Angle”

Measuring the angle between two points $\mathbf{p}_0, \mathbf{p}_1$ relatively to a third point \mathbf{p} is especially useful when analyzing the neighborhood of \mathbf{p} . However when points are bound to a surface, the geometric angle may be not significant enough (figure 3). Thus, following our previous geodesic distance approximation, we define the “curved-angle” $\tilde{A}_{\mathbf{p}}(\mathbf{p}_0, \mathbf{p}_1)$ as the sum of three angles taken along the control polygon of the boundary curve interpolating $\mathbf{p}_0, \mathbf{p}_1$:

$$\tilde{A}_{\mathbf{p}}(\mathbf{p}_0, \mathbf{p}_1) = \widehat{\mathbf{p}_0\mathbf{p}b_{210}} + \widehat{b_{210}\mathbf{p}b_{120}} + \widehat{b_{120}\mathbf{p}\mathbf{p}_1} \quad (5)$$

Bounds on Point-Normal Interpolation

The construction by projection of the control points of a PN triangle boundary is not always consistent. Indeed, as illustrated in figure 4a, certain configurations of the positions and normals of the two boundary extremities yield a surface reconstruction which is inconsistent with respect to the normal’s orientation (inside/outside). This situation occurs when the point \mathbf{p}_1 is inside the infinite cone of apex \mathbf{p}_0 and axis $\mathbf{n}_0 + \mathbf{n}_1$ (figure 4b). In this case, a specific (global) treatment could be applied in order to re-establish the normal consistency. However this would mean that we try to reconstruct a highly under-sampled surface from a r -sampling P^0 with $r > 2$ [ABK98] and hence, it is more natural to consider that the points \mathbf{p}_0 and \mathbf{p}_1 are not neighbors. Thus two points are not neighbors if the following condition is not satisfied:

$$\left| (\mathbf{n}_0 + \mathbf{n}_1) \cdot \frac{\mathbf{p}_0\mathbf{p}_1}{\|\mathbf{p}_0\mathbf{p}_1\|} \right| > 1 + \mathbf{n}_0 \cdot \mathbf{n}_1 \quad (6)$$

4. One-ring Neighborhoods

The selection of a pertinent one-ring neighborhood is a critical step of the algorithm. Indeed, it is from this set of points that new points will be inserted around the refined point \mathbf{p} , and hence, the robustness of the refinement process as well as its capacity to fill large holes directly depend on the quality of this neighbor selection. A simple neighborhood definition such as the common k -nearest neighbors leads to a very poor selection and the development of a more accurate method is essential. Advanced techniques use a Voronoi diagram [UMA04] or angle criterion [LP02, GBP04] after an

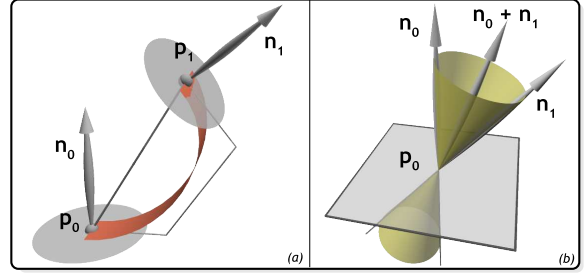


Figure 4: (a) The relative positions and orientations of the points \mathbf{p}_0 and \mathbf{p}_1 are such that the construction by projection is inconsistent. (b) Given the position \mathbf{p}_0 and the two normals $\mathbf{n}_0, \mathbf{n}_1$: the point \mathbf{p}_1 must be outside the yellow cone.

orthogonal projection of the nearest neighbors in the local tangent plane. At the same time as it simplifies the neighbor selection, the orthogonal projection to a 2D domain significantly reduces accuracy. Indeed the elevation information which is crucial for dealing with low local density and high curvature areas are lost.

For these reasons we propose two new neighborhood computation procedures significantly improving the tolerance to under-sampled and/or scattered point sets. The first has the advantage of being naturally symmetric (if \mathbf{p}_i is a neighbor of \mathbf{p}_j then \mathbf{p}_j is also a neighbor of \mathbf{p}_i) while the second improves the selection in under-sampled areas in spite of being slightly more expensive and losing the symmetry property.

4.1. Fuzzy BSP Neighborhood

The definition of our first neighborhood N_p^f is based on a BSP neighborhood [Pau03]. However, our selection is performed without projection (allowing the neighborhood relation to be symmetric) and we replace the too selective discriminant planes by more flexible *fuzzy* planes (better suited for scattered sampling). We start by computing the Euclidean neighborhood N_p^E of \mathbf{p} as the indices of all points

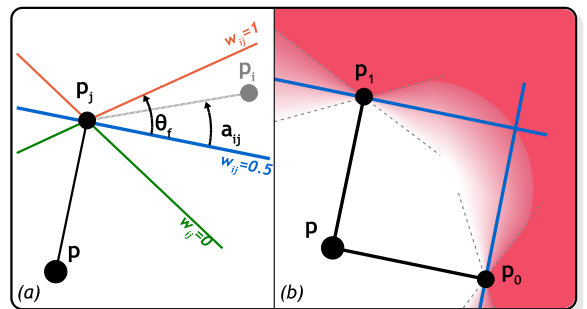


Figure 5: Illustration of Fuzzy planes. (a) Computation of the badness value w_{ij} between two points. (b) The red gradation represents the variation of the badness value w_i from 0 (white) to 1 (red), when w_{i1} (resp. w_{i0}) is the max of the successor’s (resp. predecessor’s) badness.

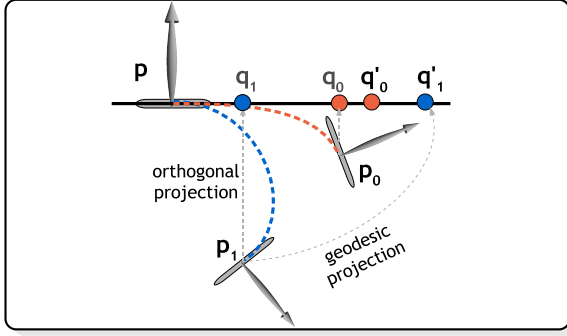


Figure 6: Illustration of our “geodesic projection”. For instance, the “geodesic projection” \mathbf{q}'_1 of \mathbf{p}_1 onto the tangent plane of \mathbf{p} is the orthogonal projection \mathbf{q}_1 of \mathbf{p}_1 moved such that its distance to \mathbf{p} is the geodesic distance between \mathbf{p} and \mathbf{p}_1 .

\mathbf{p}_i included in the ball of center \mathbf{p} and radius r :

$$N_p^\varepsilon = \{i \mid \mathbf{p}_i \in P^l, \mathbf{p}_i \neq \mathbf{p}, \|\mathbf{p} - \mathbf{p}_i\| < r\} \quad (7)$$

Following the previous section we first remove from N_p^ε all neighbors $\mathbf{p}_i, i \in N_p^\varepsilon$ which do not satisfy the condition 6.

Next, in order to get a pertinent one-ring we remove all neighbors which are strongly “behind” another one or slightly “behind” two others. For this purpose, we introduce a *badness* value w_{ij} stating to what extent the neighbor \mathbf{p}_i is “behind” the neighbor \mathbf{p}_j . This value depends on the signed angle between the vector $\mathbf{p}_j\mathbf{p}_i$ and the plane of normal \mathbf{pp}_j passing through \mathbf{p}_j (figure 5). Let α_{ij} be this angle and θ_f a given tolerance angle (typically $\frac{\pi}{6}$). Then:

$$w_{ij} = \begin{cases} 0 & \text{if } \alpha_{ij} < -\theta_f \\ \frac{1}{2} \left(\frac{\sin(\alpha_{ij})}{\sin(\theta_f)} + 1 \right) & \text{otherwise} \end{cases}$$

with $\sin(\alpha_{ij}) = \frac{\mathbf{pp}_j \cdot \mathbf{p}_j\mathbf{p}_i}{\|\mathbf{pp}_j\| \cdot \|\mathbf{p}_j\mathbf{p}_i\|}$. Let $Succ_i$ (resp. $Pred_i$) be the set of successors (resp. predecessors) of the point $\mathbf{p}_i, i \in N_p^\varepsilon$ such that $Succ_i = \{j \in N_p^\varepsilon \mid 0 < \widehat{\mathbf{q}_i\mathbf{p}\mathbf{q}_j} < \pi\}$ where \mathbf{q}_i and \mathbf{q}_j are the orthogonal projections of the points \mathbf{p}_i and \mathbf{p}_j on the tangent plane of \mathbf{p} (resp. $Pred_i = \{j \in N_p^\varepsilon \mid -\pi < \widehat{\mathbf{q}_i\mathbf{p}\mathbf{q}_j} < 0\}$). We point out that the projections are only used to sort the points as successors and predecessors. Next, we compute a *badness* value w_i for each neighbor \mathbf{p}_i as the sum of the two maximal values w_{ij} of the successors and w_{ik} of the predecessor:

$$w_i = \max_{j \in Succ_i} (w_{ij}) + \max_{k \in Pred_i} (w_{ik}) \quad (8)$$

As soon as a neighbor \mathbf{p}_i has a *badness* value w_i greater than 1 it is removed from the neighborhood of \mathbf{p} (figure 5b) yielding the final one-ring neighborhood N_p^f . Finally, the neighbors $\mathbf{p}_i, i \in N_p^f$ are sorted by increasing angles of their projection \mathbf{q}_i onto the tangent plane of \mathbf{p} , so that this neighborhood implicitly forms a triangle fan around \mathbf{p} .

4.2. Accurate Neighborhood

Our second neighborhood definition N_p^g is a variant of the previous one where the computation of the weights w_{ij} is performed after the “geodesic projection” of all neighbors $\mathbf{p}_i, i \in N_p^\varepsilon$ onto the tangent plane of \mathbf{p} . This projection technique differs significantly from the standard orthogonal projection since the geodesic distance between \mathbf{p} and its neighbors \mathbf{p}_i (equation 4) is also the distance between \mathbf{p} and the projection \mathbf{q}'_i of \mathbf{p}_i (the figure 6). Thus \mathbf{q}'_i is computed as follow:

$$\mathbf{q}'_i = \mathbf{p} + \tilde{G}(\mathbf{p}, \mathbf{p}_i) \frac{\mathbf{q}_i - \mathbf{p}}{\|\mathbf{q}_i - \mathbf{p}\|} \quad (9)$$

where, \mathbf{q}_i is the orthogonal projection of the point \mathbf{p}_i on the tangent plane of \mathbf{p} . From here, the neighborhood N_p^g is selected as N_p^f except that the weights w_{ij} are computed with the projections $\mathbf{q}'_i, \mathbf{q}'_j$ instead of the initial positions $\mathbf{p}_i, \mathbf{p}_j$. The usefulness of this projection is illustrated figure 6: if we apply directly our fuzzy plane filtering without projection, the two points \mathbf{p}_0 and \mathbf{p}_1 will be selected. An orthogonal projection is even less precise (because only \mathbf{p}_1 would be selected) while after our “geodesic projection” it clearly appears that \mathbf{p}_1 is not a neighbor of \mathbf{p} .

5. The Local Refinement Algorithm

In this section, we detail the local refinement of the current point \mathbf{p} from its neighborhood N_p computed with one of the previous methods (sections 4.1 and 4.2). The choice of the method depends on the sampling quality and we comment on this in next sections. The challenge is now to build a relevant new neighborhood N'_p around \mathbf{p} . This new neighborhood corresponds to one refinement step around \mathbf{p} and it must both fill holes and regularize the sampling.

We first initialize the set N'_p with the indices of the points of P^{l+1} which can be considered as newly inserted points i.e. the points which are at a distance from \mathbf{p} smaller than $\lambda_1 r$ with $\lambda_1 = 1/\sqrt{3}$ (figures 7a and 7b). The value of λ_1 is taken according to the scale factor of a $\sqrt{3}$ refinement in the regular case [Kob00].

We consider that the refinement of \mathbf{p} is complete as soon as the maximal “curved angle” (equation 5) between two consecutive points of N'_p is smaller than a given threshold $\theta_c = \frac{\pi}{2}$. Hence if the initialization does not provide a complete neighborhood, new points must be inserted. To do so, we define three terms (figure 7b and 7c):

- Y_p is the set of points already inserted which are sufficiently close to \mathbf{p} but not close enough to be selected in N'_p :

$$Y_p = \{h \mid \mathbf{p}_h \in P^{l+1} - P^l, \lambda_1 r < \|\mathbf{p}_h - \mathbf{p}\| < r\}, \quad (10)$$

- \mathcal{D}_p is the discard space avoiding oversampling and redundancy. It is the union of the spheres of radius $\frac{r}{2}$ centered on the points of Y_p :

$$\mathcal{D}_p = \{x; h \in Y_p, \|x - \mathbf{p}_h\| < \lambda_2 r_h\} \quad (11)$$

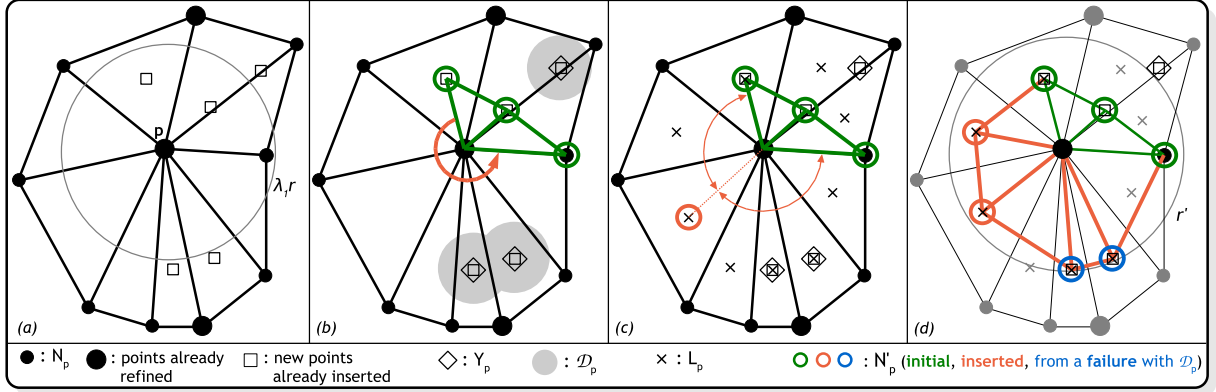


Figure 7: Local refinement of the current point \mathbf{p} . (a) The neighborhood of the point \mathbf{p} before its own refinement. It is composed of its neighbors N_p and the closest points already inserted during the previous refinement of two of its neighbors. (b) Initialization of the new neighborhood N'_p (in green) and illustration of the discard space \mathcal{D}_p around the points of Y_p . The arrow shows the area where the insertion of new points will begin in order to complete N'_p . (c) New points of L_p are computed and the best one filling the previous area is selected and inserted. (d) After the insertion of two new points (in red) and the failure of two other insertions due to the discard space (in blue) the new neighborhood N'_p of \mathbf{p} is complete. Four new points remain in L_p ; these are ignored because they are superfluous. Note the rotation effect between N_p and N'_p of our $\sqrt{3}$ -like refinement.

- L_p is the set of all possible new points, i.e. it is the set of points resulting from the application of our smoothing operator (equation 3) on the center of gravity of all triangles of the implicit triangle fan formed by the sorted neighborhood N_p :

$$L_p = \{\text{cog}(\mathbf{p}, \mathbf{p}_i, \mathbf{p}_{i+1}) + \phi(\mathbf{p}, \mathbf{p}_i, \mathbf{p}_{i+1}) \mid i \in N_p\}, \quad (12)$$

The insertion procedure is as following:

While the neighborhood N'_p is not complete **repeat**

1. Select the pair of consecutive points $\mathbf{p}_j, \mathbf{p}_{j+1}$ in N'_p having the maximal “curved angle” (figure 7b).
2. Select the new point in L_p which best balances point sampling (figure 7c). A good candidate is the point $\mathbf{p}_k \in L_p$ such that the minimum of the two angles $\widehat{\mathbf{p}_k \mathbf{p} \mathbf{p}_k}$ and $\widehat{\mathbf{p}_k \mathbf{p} \mathbf{p}_{j+1}}$, is maximal.
3. If this point is not in the discard space \mathcal{D}_p , it is inserted in N'_p and P^{l+1} , otherwise no new point is inserted in P^{l+1} and the point \mathbf{p}_k is replaced in N'_p by the closest point in Y_p (figure 7d). Hence, if the samples are locally dense enough, no new point is inserted.

When this process terminates, the radius of the point \mathbf{p} is updated according to its new neighborhood N'_p . The new radius r' is set to the maximum distance between \mathbf{p} and the points of N'_p : $r' = \max_{j \in N'_p} (\|\mathbf{p} - \mathbf{p}_j\|)$ (figure 7d). The radius r_j of each new neighbor \mathbf{p}_j , $j \in N'_p$ is set to the maximum of the four values: r_j , $\|p_j - p\|$, $\|p_j - p_{j-1}\|$ and $\|p_j - p_{j+1}\|$.

6. Sharp features

A common and efficient way to handle sharp creases with point-based geometry is to use clipped splats [ZRB*04]. Although clipped splats are sufficient for the rendering, geometry processing requires, in addition, that splats share the

same center [PKKG03, GBP04]. Thus our crease splat is just a single point with two different normals.

With our $\sqrt{3}$ -like up-sampling strategy the refinement of boundaries and creases is more fussy than with a diadic refinement as used in [GBP04] because new points are never explicitly inserted between two points. With the mesh-based $\sqrt{3}$ subdivision scheme [Kob00] Kobbelt proposes the insertion of two vertices on each boundary and crease segment at each odd refinement step only. However, with point-based geometry, there is no connectivity and after two refinement steps, the crease (or boundary) points will probably not be neighbors anymore so that no special treatment can be applied between them. Hence we explicitly store a list of crease and boundary segments; this solution has the advantage of being robust and functional. This list is computed during the first refinement step. Crease segments between two crease splats are detected in the same manner as in [GBP04]. For the boundaries, a point $\mathbf{p} \in P^0$ is said to be a potential boundary splat if there exist two consecutive points \mathbf{p}_i and \mathbf{p}_{i+1} , $i \in N_p$ such that the angle $\widehat{\mathbf{p}_i \mathbf{p} \mathbf{p}_{i+1}}$ is greater than a given threshold $\theta_b \in \left] \frac{2\pi}{3}, \pi \right]$. Then if \mathbf{p} is a potential boundary point and \mathbf{p}_i , $i \in N_p$ is also a potential boundary point, the pair $(\mathbf{p}, \mathbf{p}_i)$ is inserted in the list of boundary segments. Note that the choice of θ_b allow us to choose how strongly to smooth the boundary: if $\mathbf{p}_i, \mathbf{p}, \mathbf{p}_{i+1}$ have an angle smaller than θ_b but they are effectively on a boundary then the inserted point between them will have a larger angle, and hence the boundary will be effectively detected after a few refinement steps. The interpolation between two crease/boundary points uses a cubic Bézier curve (as in [GBP04]) except that two points are inserted at the thirds of the curve at each odd refinement step instead of inserting a single points in its middle at every step.

7. Data structures and Implementation

Closest points query

As in a lot of point-based processing methods, a critical time-consuming part of our method is the closest points query necessary to compute the Euclidean neighborhood N_p^e (equation 7). To improve efficiency, points must be spatially sorted into a data structure, like a *kd*-tree or a 3D grid, with a fine granularity.

Moreover, the local refinement step of a single point $\mathbf{p} \in P^l$ (section 5) also requires us to find the closest points already inserted into P^{l+1} (to compute the sets N_p^e and Y_p). Assuming that new points are sequentially inserted into a list of points, our solution is to associate to each point $\mathbf{p} \in P^l$ the indices of the first and last new points inserted during its own refinement. We call these points the *children* of \mathbf{p} . Thus, the set of new points already inserted into P^{l+1} close to \mathbf{p} is inferred from all children of all neighbors $\mathbf{p}_i, i \in N_p^e$ of \mathbf{p} . This solution has the advantage that it naturally creates a hierarchy of bounding spheres (a radius is associated with each point) which is also used to perform efficient closest points queries with a very low memory consumption: we only store two indices per point. We only have to structure the initial point set P^0 once: no additional data structure has to be built for other levels and no insertion has to be performed, and hence a static data structure (e.g. a *kd*-tree) can be efficiently used. A closest points query around the current point \mathbf{p} at a level l with $l \neq 0$ is done by performing a recursive traversal of the bounding spheres hierarchy while the starting bounding spheres (points of the set P^0) are found by performing a closest points query using the initial data structure.

Real-Time Refinement

On one hand, the data structure presented above is especially useful for real-time applications, i.e. when some parts of the model must be dynamically refined. For instance, in a real-time rendering system, such as the one presented in [GBP04], the refinement procedure is used to maintain a screen space splat size smaller than a given threshold (e.g. two pixels). According to the relative position of the camera, under-sampled regions of the models are refined (yielding to the insertion of new points stored in a cache) while outdated generated points are removed in order to free the memory cache. In this context, our neighbor search algorithm is particularly well suited because insertion and deletion of points is trivial, the memory cost is very low and levels are well separated. The clear separation of levels is essential when the model is not globally refined since different parts of the model are refined at different levels while new points of the level $l+1$ must be interpolated from points of the set P^l only.

On the other hand, still in the context of a rendering application, the selection of points that have to be refined is equivalent to a visibility and level-of-details (LOD) point selection which is generally performed by spatially sorting points into a hierarchy of bounding volumes (e.g. *kd*-tree or octree). Our hierarchy of bounding spheres could also be used for this purpose (in a way it looks like the *QSplat*

representation [RL00]), however, it has been shown that a too fine data structure is inefficient for high-level point selection [DVS03, EF04]; such a data structure should contain approximately one or two thousands points per cell. Moreover, no regular hierarchical data structure matches the $\sqrt{3}$ refinement. Thus, in our real-time rendering system, we have opted for a more flexible hierarchy of bounding boxes (similar to the *point-octree* [Sam89]) where the goal is to keep a constant number of points per node. We start from a set of axis aligned bounding boxes (the root nodes). Then, when a node is refined, according to its actual number of points, it is:

- not split (the node has only a single child),
- split in two or three along its maximal dimension,
- split in four along its two maximal dimensions.

With respect to our fine hierarchy of points, to memory consumption and to efficient GPU rendering requirements, the points of a node must be stored sequentially in a single chunk of video memory (shared by all nodes). Thus, a node has just to store an axis aligned box, its points as a range of indices (the indices of its first and last points) and from zero to four children. In order to respect the sequential storage of points per nodes, a node is refined as follow:

- split the current node into 1, 2, 3 or 4 children (see above),
- for each child, refine all points which are in its bounding box.

At the end of this process the bounding box of each child must be updated, i.e. contracted and/or extended in each direction in order to be as small as possible and to guarantee that it effectively contains all its points (the new points inserted during the refinement of one point of the child may be outside its initial box).

In order to maintain a real-time frame rate (above 24 fps) the time allowed for the refinement procedure must be bound (implying a breadth-first order traversal). Owing to the time consumed by the point selection and the rendering itself, the remaining time per frame for the refinement procedure is very limited, and we have measured with our implementation that a point generation rate above 300k points per second is the minimal performance required for a comfortable navigation.

Simplifications/Optimizations

The technique presented above has been designed to handle under-sampled and scattered point clouds. However, for relatively well sampled models and/or after a few refinement steps a lot of expensive tests can be safely optimized:

1. Approximate the geodesic distance by the Euclidean distance: $\tilde{G}(p_0, p_1) \approx \|p_1 - p_0\|$
2. Approximate the “curved-angle” by the simple geometric angle: $\tilde{A}_p(p_0, p_1) = \widehat{p_0 p p_1}$
3. Use the N_p^f instead of the N_p^s neighborhood.
4. Approximate the position of a new point by the center of gravity during the refinement process and apply the

smoothing operator if and only if the new point is effectively inserted.

These optimizations allow us to significantly improve the performance of the algorithm (by a speedup factor from 1.5 to 2).

8. Results and Discussion

We have tested our new approach on a wide variety of point-based models. The refinement of textured models is illustrated in figure 12: the color of a new point is linearly interpolated from the three extremities of the PN triangles. Figure 10 illustrates the refinement of sharp features.

Robustness evaluation: In order to evaluate the robustness of our method, it has been tested on several irregularly down-sampled models: for instance in figure 1 our refinement algorithm is applied to 3500 points randomly selected from a set of 150k points representing a statue of Isis. Figure 9 illustrates the use of our refinement method on an especially large hole. To fill this hole we have simply adjusted the radius of points such that they overlap the hole and we have applied our refinement algorithm several times. Figure 11 illustrates the usefulness of our “geodesic projection” and our “curved angle” on an highly under-sampled area. In this example, the boundary of the David’s eye exhibits holes if these tools are not used.

Performances: We have tested our implementation on an Athlon 3500+ system with an nNidia GeForce 6800 graphics card. Our algorithm is able to generate from 350k to 450k points per second depending of the local regularity of the sampling. The optimizations presented in the previous section allow us to reach a point generation rate around 700k points per second. The time consumption of each part of the refinement procedure is (approximately) as follow: 23% for the Euclidean neighborhood, 40% for the filtering of the neighborhood, 29% for the selection of new points and 8% for the interpolation and radii updates.

Comparisons: Figure 8 illustrates the superiority in the reconstructed surface smoothness of our new $\sqrt{3}$ refinement algorithm over the butterfly mesh based interpolatory subdivision scheme [DLG90, ZSS96] (C^1 surface but with large oscillations) and the Guennebaud et al. diadic refinement method [GBP04] (high-frequency oscillation artifacts).

8.1. Convergence and Continuity issues

From a practical point of view, our results show that our new refinement algorithm provides a high quality smooth surface and allows flexibility in the quality of the input point sampling. However, from the theoretical point of view, all the analysis mechanism developed during the last decade for meshes does not hold for point-based geometry. Hence, questions remain: what are the convergence and the continuity? The heuristic character of our refinement procedure and its dependence on the point processing order make it complicated to undertake rigorous analysis of the limit surface. Nevertheless, we can expect some good properties.

For convergence, we show that the radius of any points $p \in P^l$, noted r^l , has for limit zero when l tends to infinity. Indeed an upper bound of the radius r^l is given by the distance between \mathbf{p} and its farthest new neighbor \mathbf{p}_k which is the center of a PN triangle formed by the neighbors of \mathbf{p} (at a maximal distance r from \mathbf{p}):

$$r^{l+1} = \|p - p_k\| = \|p - \frac{1}{6}(\mathbf{b}_{210} + \mathbf{b}_{120} + \mathbf{b}_{021} + \mathbf{b}_{012} + \mathbf{b}_{201})\|$$

Since:

$$\|p - \frac{1}{2}(\mathbf{b}_{210} + \mathbf{b}_{120})\| \leq \frac{5}{6}r^l, \quad \|p - \frac{1}{2}(\mathbf{b}_{201} + \mathbf{b}_{102})\| \leq \frac{5}{6}r^l$$

$$\|p - \frac{1}{2}(\mathbf{b}_{021} + \mathbf{b}_{012})\| \leq (\cos(\beta/2) + \frac{2}{3}\sin(\beta/2))r^l$$

we have:

$$r^{l+1} \leq c_\beta r^l \quad (13)$$

$$c_\beta = \frac{1}{3}(\frac{5}{6} + \frac{5}{6} + (\cos(\beta/2) + \frac{2}{3}\sin(\beta/2))) \quad (14)$$

where β is the angle between the farthest neighbor of p and its successor (or predecessor). Whatever the value of β is, $c_\beta < 0.96$ that is strictly less than 1 and hence, the limit of r^l is zero.

For the continuity issue, considering a given point $\mathbf{p} \in P^l$, we define α_p^l as the angle of a double cone of apex \mathbf{p} and axis \mathbf{n} (the normal of \mathbf{p}) such that the complement of this double cone (co-cone) contains all the neighbors of \mathbf{p} at the level l . Moreover, the symmetry property of our first neighborhood computation guarantee that the point \mathbf{p} is one of the extremities of each built PN triangle which has yield to the insertion of a new points in its new neighborhood. Hence, assuming that after a finite number of refinement steps, no Bézier patch presents an inflexion point, we show that the angle α_p^l has for limit 0 when l tends to infinity. As above, we find a constant $c < 1$ such that $\alpha^{l+1} < c\alpha^l$. This means that, at the limit, all neighbors of \mathbf{p} are in its tangent plane that is a necessary condition for the G^1 continuity. Our assumption has always been satisfied in our experimentations, even in very distorted areas, however there is no theoretical guaranty since it has not been proved.

9. Conclusions

We have presented a fast and robust interpolatory refinement method suitable for real-time processing of point based geometry. The refinement procedure is based on a PN triangle interpolation associated with a $\sqrt{3}$ -like refinement. Our results show the efficiency of this combination which generates smooth reconstructed surfaces. We have presented a set of tools allowing the extrapolation of local geometric information from a point-normal surface representation. These tools have been particularly useful in the development of neighborhood computations handling both under-sampled and scattered point clouds. Our technique is also robust enough to smoothly fill large holes in the geometry. The connectivity free of point cloud has allowed us to develop an efficient adaptive refinement strategy with trivial reverse refinement steps.

Even though the surface looks very smooth, there is no fundamental theoretical machineries allowing a rigorous analysis of the limit surface (as for subdivision surfaces

on meshes). Nevertheless, we have proved some important properties of our refinement: refinement steps after refinement steps the inserted points converge on the central old point and at the limit, the inserted points lie in the tangent plane of the old central point. Hence, as future works, we intend to follow the analysis of the limit surface. We will also investigate the capabilities of our refinement procedure in interactive applications such as multi-resolution modeling (where the user will simply interact with oriented disks) and for the efficient compression of point sets.

Acknowledgements

We would like to thank Neil Dodgson from the University of Cambridge for proof-reading the paper.

References

- [AA03] ADAMSON A., ALEXA M.: Approximating and intersecting surfaces from points. In *Proceedings of the Eurographics Symposium on Geometry Processing* (2003), pp. 245–254. 2
- [ABCO*03] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Computing and rendering point set surface. *IEEE Transaction on Visualization and Computer Graphics* 9, 1 (2003), 3–15. 2
- [ABK98] AMENTA N., BERN M., KAMVYSSELIS M.: A new voronoi-based surface reconstruction algorithm. In *Proceedings of ACM SIGGRAPH 98* (1998). 4
- [AK04] AMENTA N., KIL Y. J.: Defining point set surfaces. In *Proceedings of ACM SIGGRAPH 2004, Computer Graphics Proceedings* (2004). 2
- [BK03] BOTSCH M., KOBBELT L.: High-Quality Point-Based Rendering on Modern GPUs. In *11th Pacific Conference on Computer Graphics and Applications* (2003), pp. 335–343. 1
- [BK04] BOTSCH M., KOBBELT L.: Phong splatting. In *Proceedings of Point-Based Graphics 2004* (2004). 1
- [CBC*01] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and representation of 3D objects with radial basis functions. In *Proceedings of ACM SIGGRAPH 2001* (2001), pp. 67–76. 2
- [DLG90] DYN N., LEVIN D., GREGORY J.: A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transaction on Graphics*, 9 (2) (1990), 160–169. 8
- [DVS03] DACHSBACHER C., VOGELGSANG C., STAMMINGER M.: Sequential point trees. In *Proceedings of ACM SIGGRAPH 2003* (2003), pp. 657–662. 7
- [EF04] ENRICO G., FABIO M.: Layered point clouds: a simple and efficient multiresolution structure for distributing and rendering gigantic point-sampled models. *Computers & Graphics* 28 (2004), 815–826. 7
- [GBP04] GUENNEBAUD G., BARTHE L., PAULIN M.: Dynamic surfel set refinement for high-quality rendering. *Computers & Graphics* 28 (2004), 827–838. 2, 4, 6, 7, 8
- [GP03] GUENNEBAUD G., PAULIN M.: Efficient screen space approach for Hardware Accelerated Surfel Rendering. In *Proceedings of Vision, Modeling and Visualization* (2003), IEEE Signal Processing Society, pp. 41–49. 1
- [KB04] KOBBELT L., BOTSCH M.: A survey of point-based techniques in computer graphics. *Computers & Graphics* 28 (2004), 801–814. 1
- [Kob00] KOBBELT L.: $\sqrt{3}$ subdivision. In *Proceedings of ACM SIGGRAPH 2000* (2000). 2, 5, 6
- [Lev03] LEVIN D.: Mesh-independent surface interpolation. In *Geometric Modeling for Data Visualization* (2003). 2
- [LP02] LINSEN L., PRAUTZSCH H.: Fan clouds - an alternative to meshes. In *Dagstuhl Seminar 02151 on Theoretical Foundations of Computer Vision - Geometry, Morphology and Computational Imaging* (2002). 4
- [MMS*04] MOENNING C., MÉMOLI F., SAPIRO G., DYN N., DODGSON N. A.: *Meshless geometric subdivision*. Tech. rep., IMA Preprint Series number #1977, 2004. 2
- [OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.-P.: Multi-level partition of unity implicits. *ACM Transactions on Graphics* 22, 3 (July 2003), 463–470. 2
- [Pau03] PAULY M.: *Point Primitives for Interactive Modeling and Processing of 3D geometry*. Master's thesis, ETH Zürich, 2003. 4
- [PGK02] PAULY M., GROSS M., KOBBELT L. P.: Efficient simplification of point-sampled surfaces. In *Proceedings of the 13th IEEE Visualization Conference* (2002), pp. 163–170. 2
- [PKKG03] PAULY M., KEISER R., KOBBELT L. P., GROSS M.: Shape modeling with point-sampled geometry. In *Proceedings of ACM SIGGRAPH 2003* (2003), pp. 641–650. 2, 6
- [RL00] RUSINKIEWICZ S., LEVOY M.: QSplat: A multiresolution point rendering system for large meshes. In *Proceedings of SIGGRAPH 2000, Computer Graphics Proceedings* (2000), pp. 343–352. 7
- [Sam89] SAMET H.: *The Design and Analysis of Spatial Data Structures*. Reading, Mass.: Addison Wesley, 1989. 7
- [TRS04] TOBOR I., REUTER P., SCHLICK C.: Multiresolution reconstruction of implicit surfaces with attributes from large unorganized point sets. In *Proceedings of Shape Modeling International 2004* (2004). 2
- [Tur92] TURK G.: Re-tiling polygonal surface. In *Proceedings of ACM SIGGRAPH 92* (1992). 2
- [UMA04] ULRICH C., MARTIN R., ALEXANDRU T.: Surface processing methods for point sets using finite elements. *Computers & Graphics* 28 (2004), 851–868. 4
- [VPBM01] VLACHOS A., PETERS J., BOYD C., MITCHELL J. L.: Curved pn triangles. In *Proceedings of the 2001 symposium on Interactive 3D graphics* (2001). 2, 3
- [ZPvBG01] ZWICKER M., PFISTER H., VAN BAAR J., GROSS M.: Surface splatting. In *Proceedings of ACM SIGGRAPH 2001* (2001), pp. 371–378. 1
- [ZRB*04] ZWICKER M., RÄSÄNEN J., BOTSCH M., DACHSBACHER C., PAULY M.: Perspective accurate splatting. In *Graphics Interface 2004* (2004). 1, 6
- [ZSS96] ZORIN D., SCHRÖDER P., SWELDENS W.: Interpolating subdivision for meshes with arbitrary topology. In *Proceedings of ACM SIGGRAPH 1996* (1996), pp. 189–192. 8



Figure 8: The Igea model uniformly sampled by 600 points is refined to 150k points with various techniques. From left to right: the butterfly (after a meshing step), the Guennebaud et al. diadic refinement and our new $\sqrt{3}$ -like refinement.

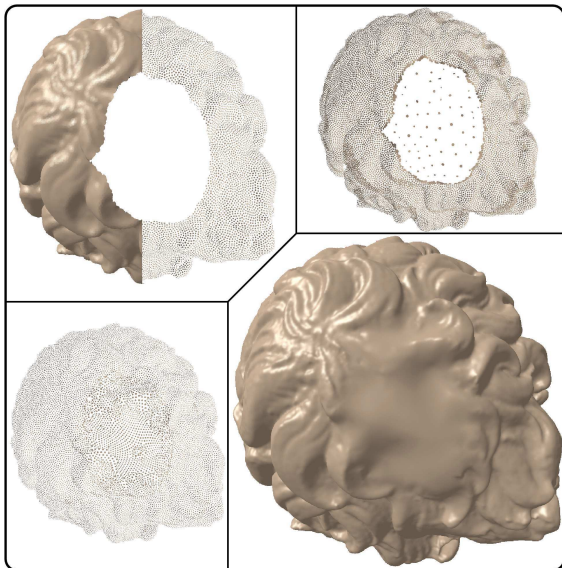


Figure 9: Illustration of the hole filling capability of our algorithm. A large hole in the David's hair is filled by adjusting the radius of boundary points such that they are greater than the hole and applying our refinement algorithm. The final image is obtained after eight refinement steps while the two others show intermediate steps.

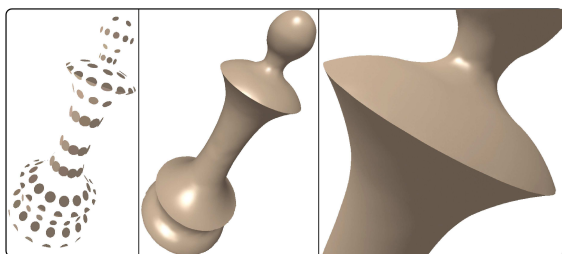


Figure 10: Illustration of the refinement of creases.

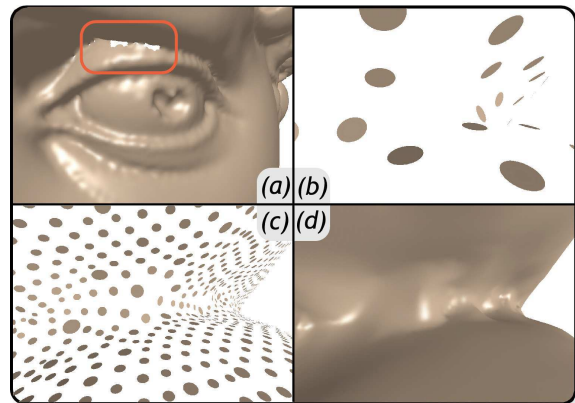


Figure 11: Illustration of the usefulness of our “geodesic projection” and our “curved angle”. (a) If they are disabled high curvature areas are not reconstructed (holes appear). The three other images are close views of the refinement process when our “geodesic projection” and “curved angle” are enabled. (b) The initial sampling. (c-d) Intermediate step and final refinement: the previous holes are smoothly reconstructed.

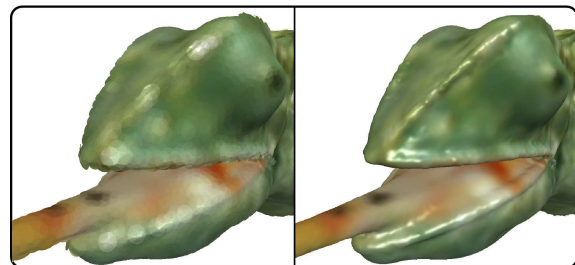


Figure 12: Refinement of a textured model. The right image is obtained after 4 refinement steps.

7.2 Modélisation par esquisses

7.2.1 Interactive modelling from sketches using spherical implicit functions.

Auteurs : Anca Alexe, Véronique Gaildrat, Loïc Barthe
Revue : AFRIGRAPH
Date : Novembre 2004

Interactive Modelling from Sketches using Spherical Implicit Functions

A. Alexe
IRIT – Paul Sabatier University
118 route de Narbonne
31062 Toulouse 04, France
33 (0)5 61 55 83 29
alex@irit.fr

V. Gaildrat
IRIT – Paul Sabatier University
118 route de Narbonne
31062 Toulouse 04, France
33 (0)5 61 55 83 29
gaildrat@irit.fr

L. Barthe
IRIT – Paul Sabatier University
118 route de Narbonne
31062 Toulouse 04, France
33 (0)5 61 55 74 31
lbarthe@irit.fr

ABSTRACT

We present an interactive modelling technique, which reconstructs three-dimensional objects from user-drawn two-dimensional strokes. We first extract a skeleton from the 2D contour, and the skeleton is used to define an implicit surface that fits the 2D contour. The reconstructed 3D shape has a natural aspect, it is very smooth and can easily be edited and modified using strokes or performing operations on the skeleton. This method is very accessible for non-specialist users and it allows fast and easy shape prototyping.

Categories and subject Descriptors:

I.3.5 [Computer Graphics] Computational Geometry and Object Modelling.

General Terms

Algorithms, Design.

Keywords

Geometric Modelling, Sketches, Implicit surfaces.

1. INTRODUCTION

When designing a three-dimensional object, one starts from a mental representation. By mental representation we understand a very high level description of the object, i.e. when the user wants to model a man he knows that he has to model four limbs and a body, etc. This mental representation has to be transcribed into the computer as a three-dimensional shape

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Conference '04, Month 1–2, 2004, City, State, Country.
Copyright 2004 ACM 1-58113-000-0/00/0004...\$5.00.

representation using some interaction device. Generally neither the interaction device nor the interaction metaphor of the modelling software are intuitive enough for non-expert users and the conception of the complete object remains a fastidious succession of shape creation and editing operations.

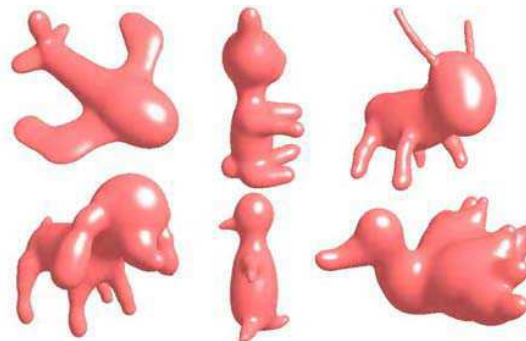


Figure 1. Examples of 3D objects designed with our system using 2D strokes, the average modelling time being 10 minutes per object.

From a general point of view the creation procedure can be decomposed into two distinct parts, which require different techniques and functionalities.

First of all, one has to build a prototype of the object i.e. a coarse approximation of the final shape. Second, the prototype is edited in order to accurately finalize the coarse representation and to add details.

Nowadays, several efficient multi-resolution techniques such as subdivision surfaces [52], [23] or normal meshes [17] allow the refinement of three-dimensional shapes once a coarse representation has been built. Concerning the prototyping procedure, actual modelling software can provide very advanced tools, allowing expert users to produce complicated and realistic prototypes, however the drawbacks are the following:

- They use complex and non-intuitive theoretical models (parametric patches, CSG, etc.). The user has to

follow a long and tedious learning process before being able to efficiently exploit these concepts.

- They use abstract notions that are too far from the real world (geometric primitives, numerical parameters, parametrical functions, key frames, etc).

The complexity of WIMP-like graphical interfaces (*Windows Icons Menus Pointer*), makes them difficult to use and increases the complexity of the user's task.

Since long time, experts showed [30], [43] that more time is spent searching through commands and menus than concentrating on the modelling task itself.

Classical modelling software still leaves unsolved some fundamental problems such as: how to quickly build mock-ups and prototypes; how to make the interaction with the user more intuitive and more natural; and finally, how to outpace the low level of abstraction.

However, we do not intent to provide a complete modelling tool, as it would be very difficult to compare our software with classical modelling software and its infinite number of modelling possibilities. We rather propose a complementing technique, which explores gesture based modelling of free-form shapes and which is mainly addressed to non-expert users.

Sketching is a simple means of expression, accessible by everyone (See [40] for a discussion on sketch-based interfaces.). It is useful for quickly materialising ideas and sharing them with the working team. It is also an excellent stimulant for creativity and innovation. For these reasons, several research projects have focused on the design of 3D shapes from 2D sketches. We will start by briefly presenting the related works on 3D shape reconstruction from a point set and then we will discuss the previous works on sketch-based modelling.

Related work on the 3D surface reconstruction: The problem can be formulated as it follows: starting from a set of points approximating a 2D contour, find a 3D closed surface with a suitable thickness which best fits the point set.

Related ideas on 3D shape reconstruction with implicit surfaces can be found in [27], [19], [8], [25], [4], [3], [31]. Other papers use globally supported RBF functions [41], [10], [44] or locally supported RBFs [28], [24], [32]. We note that all the cited papers are not meant to solve our problem, but the more general problem of 3D reconstruction from a 3D cloud of points. However they can be easily adapted to this particular case. Although RBFs seem to be the state of the art in this field, they require a large amount of data for an accurate smooth reconstruction and therefore they are not suitable for interactive use. Other inconvenients of this technique will be further discussed.

Related work on sketch-based modelling: The first modelling system that used sketches was introduced by Zeleznik [50]. SKETCH is based on gesture recognition and a "dictionary" of basic 3D primitives, and is limited to isothetic objects with sharp angles.

Teddy software [21] performs reconstruction using the chordal axis [38], from which a mesh is computed. The 2D contour is sampled regularly and the resulting planar set of points is triangulated using a constrained Delaunay triangulation [2]. A chordal axis [38], which connects the middles of the internal edges of the triangulation is then built and used as a skeleton. The final 3D shape is a polygonal mesh reconstructed by elevation of this skeleton. Teddy demonstrates the efficiency of 2D sketching for 3D free-form shape prototyping, but it also exhibits some limitations such as the difficulty to edit the mesh and to rearrange parts of the 3D object. The quality of the provided mesh is also poor, and a post-processing treatment on the mesh is necessary [20]. By using implicit surfaces (i.e. a smooth representation) we overcome these inconvenients.

Karpenko et al. [22] use variational implicit surfaces. [13] base their work on Karpenko's idea but add a more complete over-sketching utility. [51] enrich the method by the possibility to "paint" strokes on the surface or "in the air" to add more expressiveness. The surface is defined by functions that interpolate a set of points. In this context, the main problem is the extrapolation of the 3D shape thickness. The first task is to create a complete approximation of the 3D object by a set of points, which is needed by the interpolation function (the initial stroke is not sufficient). Hence, new points have to be created in both sides of the profile in order to build the 3D object thickness. This process is done by projection, which limits the form of the user-defined contours to ellipse-like shapes. The user creates several simple shapes, which are then blended in order to build the final object. The main limitation of this approach comes from the diminished range of the possible 2D contours, due to the difficult reconstruction of the 3D shape thickness. We note that the moving least squares techniques has the same drawback, so the same observations apply to it as for variational implicit surfaces. Also, in [22] the user is forced to explicitly blend the different parts of the object, whereas ideally this process would be automatic and transparent. Both of these drawbacks are avoided by our approach.

A more recent approach reconstructs a cylindrical convolution surface from the user's stroke [45]. Convolution cylindrical implicit surfaces are a smooth structure and they are very close to our approach. As we show in section 3.2.3, convolution cylinders are not adapted to reconstruct surfaces that fold onto themselves, as near the folding a visible unwanted blending appears that cannot be entirely removed by the optimisation process. Also, the authors report unwanted oscillations of the resulting surface, which are produced by the optimisation process, as it can be seen in their figures.

Owada et al. [33] use a volumetric voxel based structure, which allows them to model the intern object cavities. As done in [45], the same functionality can be realized with our approach, using negative implicit functions, and it will be developed as a future extension.

We point out that in the different methods presented so far, the user cannot control the local 3D shape thickness (except in [45]). Since the shape's thickness is computed automatically and may not correspond to the user's intention, this is an important issue.

Our approach and contributions: We prefer to keep the double representation skeleton/3D-shape and hence allow the 3D shape to be defined and modified using 2D strokes, but also edited using very simple operations on the skeleton, such as deleting, copy/cut-paste, etc. Our structure is similar to the one used in [36] to recover animation movements from video sequences. In our case there is a single image, which is the user's stroke, and we construct what would correspond to a "natural" shape from this image. The user has the possibility to modify it by various strokes and skeleton operators. Skeleton manipulations are of interest because they allow simple and intuitive rearrangements of the object's parts such as arms, legs, etc. For skeleton based editing see [49]. Moreover, a smooth shape representation is preferable to a direct polygonal mesh extraction.

Therefore we use Teddy's skeleton reconstruction procedure (modified with an adaptive sampling of the 2D stroke that allows better detail capturing than uniform rough sampling techniques) and once the skeleton is extracted, we reconstruct the 3D shape with blended implicit spheres, placed along the skeleton, which fit the 2D contour. We also preserve the skeleton structure, as a graph, the nodes being the implicit spheres.

Our reconstruction technique is based on the one used by Muraki [27] and [8], but we make it interactive by performing several simplifications which will be exposed later. The reconstructed surface is very smooth (C^7 continuous in our case; the reasons why this high class continuity is necessary will be explained in section 3.2.2), the spheres radii automatically extrapolate the 3D shape thickness and the user can edit it once the 3D shape is reconstructed, and finally simple editing operations can be performed on the skeleton.

This approach is, as the previous sketch-based approaches, suitable for modelling simple, organic-like shapes, and it can be used by non-expert users, since it does not require knowledge of the underlying surface model. Only the skeleton and the stroke tool (a digital pen, a mouse, etc.) are available to the user. It can be applied in fast prototyping and modelling for story telling. The computing times are about 2-3 seconds per stroke, with about a few dozens blobs per stroke, which is the case for all the models that we show in Figure 1. Therefore, the reconstruction times are comparable to those obtained by our predecessors, but the quality of the surface is much better (i.e. smoother and less oscillations), as we produce high-class continuity surfaces, which can be compactly stored, as a set of blobs parameters. We propose a new spherical kernel, which is well suited for our reconstruction needs. We also enriched the modelling tools with skeleton manipulation operators. Our method overcomes the limitations imposed to the input contour in [22]. Compared to cylindrical convolution functions (as used in [45]), spherical implicit functions have a smaller influence zone, and therefore they follow more faithfully the shape of the stroke, and the surface remains smooth. Besides they are faster to evaluate than cylindrical convolution functions. We also propose a technique to considerably reduce the oscillations of the resulted surface.

Paper organisation: In the next section, the adaptive 2D stroke sampling and the skeleton extraction procedure are presented. Section 3 describes the automatic reconstruction of the 3D implicit surface that fits the 2D contour and Section 4 presents the different creation/editing operations available with our system.

2. Sampling and extraction of the skeleton from the 2D stroke

First the user sketches a stroke, which is automatically closed. Then we apply an average filter, in order to remove noise from the input. In previous approaches, the contour can be then sampled following two strategies: whether the edges have uniform length ([21], [22]), or the edges length is adaptively established, between a minimum and a maximum value, whenever the angle variation is smaller than a threshold value [45]. Another solution for adaptive sampling is the popular Douglas Peucker algorithm for line simplification [14]). But the Douglas Peucker algorithm would produce a highly non-uniform repartition of samples along the contour, which will have undesirable effects on the reconstruction.

We think that an adaptive sampling technique is recommended, since it helps to capture the coarse details in the strokes and to skip the finer ones. At the same time, we need to guarantee a minimum uniformity of the samples repartition, which will result in placing the implicit spheres close enough to form a stretched smooth blending.

For these reasons, we prefer to start from a regular sampling, and then adapt it wherever this is necessary, in order to remove the oscillations in the resulting surface.

We used a wavelet-based compression of the stroke, at several levels. To compress one level, we replace every two points by one point, its coordinates being the average of the two points coordinates. The process is repeated several times (4 times in our case, which means that every 16 pixels in the contour produced a point in the sampling polygon). The coordinates are stored in tree structures, every point memorizing the two points that produced it. The leafs of the trees are the pixels of the initial contour, and the roots are the maximum compression level.

Whenever one sample is considered not sufficient, the corresponding tree expands, and the two "sons" replace the initial point. To determine if the sampling is sufficient enough, we have to compute a first approximation of the reconstructed surface, and to evaluate it. The process is described in section 3.2.2. For our computations, we start with the maximum level of compression.

The next step is the construction of a constrained Delaunay triangulation [2] (Figure 3 (b)) of the polygon points (Figure 2 (a)). This is followed by the computation of the polygon skeleton, using the chordal axis [38] (Figure 2 (c)). This is a close relative of the medial axis [11] but it is locally defined and it allows pruning of insignificant branches. We recall that the chordal axis links the middles of the internal edges of the Delaunay triangulation. A special treatment is applied to

branching points. Our skeleton extraction algorithm and pruning are based on the ones described in [21].

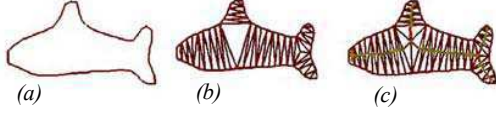


Figure 2. (a) Initial contour. (b) Delaunay triangulation. (c) Skeleton computation.

3. RECONSTRUCTION OF THE 3D IMPLICIT SURFACE

3.1 Theoretical background

The implicit surface that reconstructs the 3D shape is generated by the blend of implicit spheres. A single implicit sphere is defined by its centre c_i . Then by setting $r_i = d(p, c_i)$, the distance from a point $p \in R^3$ to the centre c_i , we define a potential function $f_i(r_i)$. Functions f_i decrease smoothly following a Gaussian-like curve, from 1 to 0, as r_i varies from 0 to infinity (Figure 3). An influence radius R_i bounds the function's contribution. When $r_i \geq R_i$: $f_i(r_i) = 0$ (bounded primitives) or $f_i(r_i) \approx 0$ i.e. $f_i(r_i) < \varepsilon, \forall r_i \geq R_i$ with ε negligible. The sphere's surface is defined by the set of points $p \in R^3$ such that $d(p, c_i) = e_i$ and $f_i(e_i) = C$, where C is a constant within the interval (0,1), generally 0.5.

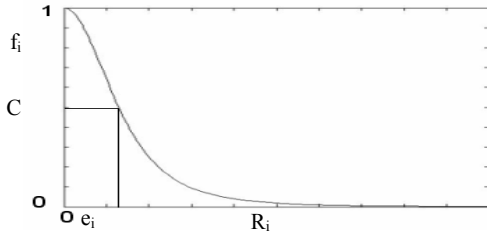


Figure 3. Plot of a Gaussian-like potential function f_i used for our reconstruction.

The volume bounded by the sphere is the point set of R^3 such that $f_i(r_i) \geq C$. The most important property of these implicit spheres is their capacity to automatically blend when their potentials are summed (N is the number of spheres):

$$f(p) = \sum_{i=0}^{N-1} f_i(p) \quad (3.1)$$

The result of the blending is a smooth surface defined by the potential function f . The first implicit model of this type was the blobby model [5], which is based on an exponential function. In order to reduce the computational cost of the exponential and to provide bounded primitives, several polynomial functions f_i have been proposed: SoftObjects [47],

Metaballs [29], W-shaped polynomial [37], the ‘‘pseudo Cauchy’’ function [39], Stolte’s function [42], keR model [8], etc. We denote these spherical potential functions f_i as ‘‘blobs’’.

Other powerful modelling by blending techniques exist, such as the blob-tree [46]. Our structure might be regarded as a single n-node from a blob-tree, the operator being the summation of the primitives. The F-Rep [35] are more general implicit functions but they are very complex to evaluate and therefore not suitable for interactive modelling.

3.2 Reconstruction and fitting procedure

3.2.1 First surface approximation

Once the skeleton has been computed, blobs are placed in every skeleton point as illustrated in Figure 4. The edges of the blobs graph are automatically computed from the skeleton’s structure. We now have to choose the potential functions f_i . For reasons that are discussed in detail in the next section, we propose the following kernel:

$$f_i(p) = \frac{(r_i^2 - R_i^2)^8}{R_i^{16}} \quad (3.2)$$

The reconstructed 3D surface is then defined by the following equation:

$$f(p) = \sum_{i=0}^{N-1} f_i(p) = \sum_{i=0}^{N-1} \frac{(r_i^2 - R_i^2)^8}{R_i^{16}} = C \quad (3.3)$$

This equation has a set of N parameters R_i (one influence radius R_i per blob f_i) that have to be determined. These parameters have to be computed so that the 3D surface best fits the drawn stroke. We cannot directly determine the parameter values for which the 3D surface is sufficiently close to the contour, but we can compute a first approximation and then use an adjustment (minimisation) procedure in order to converge to an optimal solution.

The first approximation is computed by considering each blob as being isolated. The equation that defines a single blob with a radius e_i is written as:

$$f_i(p) = \frac{(e_i^2 - R_i^2)^8}{R_i^{16}} = C \quad (3.4)$$

C being the same constant as in (3.3) and e_i being a fixed value that represents the radius in isolation of the i^{th} blob (i.e. the distance from the blob’s centre to the surface). We consider the radius e_i as being the average of the distances between the blob’s centre c_i and its neighbouring contour points (see Figure 4). The radius e_i set in this way in equation (3.4) allows us to determine the initial value of the parameter R_i . This process is applied on each blob and we obtain the initial potential function f .

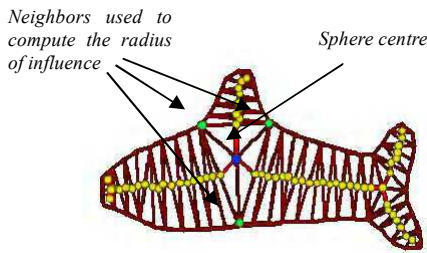


Figure 4. Computing the initial values for the radius of influence R_i .

With parameters R_i computed as described above, one could notice a slight “bloating” effect on the surface, which moves away from the contour, as shown in Figure 5 (a).

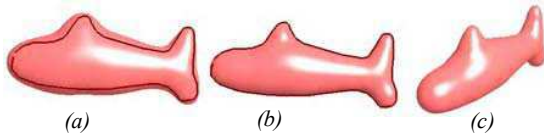


Figure 5. (a) Initial surface. (b) Surface after adjustment. (c) The same object viewed from a different angle.

This is natural since the influence radius of each blob is computed without considering the influence of the neighbouring blobs. When all the blobs’ contributions are summed (Eq. (3.3)), the potential values of functions f_i are accumulated and as the blobs blend the surface begins to bloat. Moreover, blobs are often very close, and the “bloating” effect can be quite significant. Our experiments show that a better first approximation of the contour by f is obtained by multiplying the radius e_i by a factor of $2/3$. By reducing the initial radii we reduce this bloating effect, and we win a few steps in the minimisation process, hence speeding up the process.

We notice that the distance between two neighbouring blobs never exceeds the sum of their radii and hence, they are guaranteed to blend.

3.2.2 Discussion on the smoothness of the resulting surface

Two main criteria have been taken into account to guide our kernel choice: the less oscillations of the reconstructed surface (Figure 6), and the low computational cost of the potential function.

C^1 or even C^2 continuity proved not to be sufficient for generating a perfectly smooth surface. As the Figure 6 (b) demonstrates, small oscillations are visible on the fish’s tail, which mark the transitions between blobs.

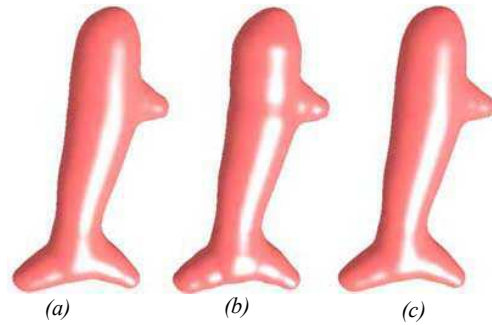


Figure 6. (a) Stroke reconstructed with blobby model [5](C^∞ function). (b) Stroke reconstructed using Soft Object function [47] (C^1 function). (c) Stroke reconstructed using our Stolte-like function (C^7).

This happens when the potential function “falls” too abruptly to zero. The oscillations vanish when the function’s continuity class increases (i.e. the potential function falls less rapidly). Our experiments produced very smooth surfaces when using computationally expensive unbounded C^∞ functions (Figure 6 (a)) and less smooth surfaces (Figure 6 (b)) when using computationally cheap C^a continuous bounded polynomial functions ($a \leq 2$). A function providing a good compromise would have a high-class continuity, while remaining computationally cheap. This has been obtained with a bounded function (Eq. 3.2) similar to the W-shaped or Stolte’s function, but with a higher degree to achieve a better smoothness (C^7 continuous) and therefore less oscillation.

As it can be observed in Figure 6, there is no visible difference between a surface generated using the blobby model, and a surface generated with our function. This is the first solution that we propose for smoothing the surface. However when the surface has tiny cylindrical regions (Figure 7), or small details, this might not be enough.

Another solution would be simply to sub-sample the stroke, in order to get a higher number of contour points, which would generate a higher number of blobs. But this solution adds more blobs everywhere on the stroke, when actually only some regions need more blobs. In this case we use a third solution.

We consider that two blobs are too far from each other if the distance between their centres exceeds the minimum of the two blobs radii, multiplied by an adjusting factor (which is 1.2 in our case). In this case the sampling contour points in that zone are expanded, every point being replaced by the two points on the superior tree level that we used for compression (see section 2). In that way we refine all the zones where blobs are not close enough. This process is done before parameters adjustment. If the trees have been expanded, the skeleton and the first approximation of the implicit spheres need to be recomputed. This process takes less than one second, and it is repeated a number of times that cannot exceed the maximum level of compression (the average is two times). This maintains the modelling time interactive.

In Figure 7 (c) the surface exhibits oscillations in tiny regions, i.e. where the blobs are not close enough to form a stretched blending. In this case the minimisation process fails to remove the oscillations and hence we use our blobs insertion procedure. After the adjustment of the local blob density, 32 contour points have been added, in less than 2 seconds and the oscillations on the surface have been considerably reduced.

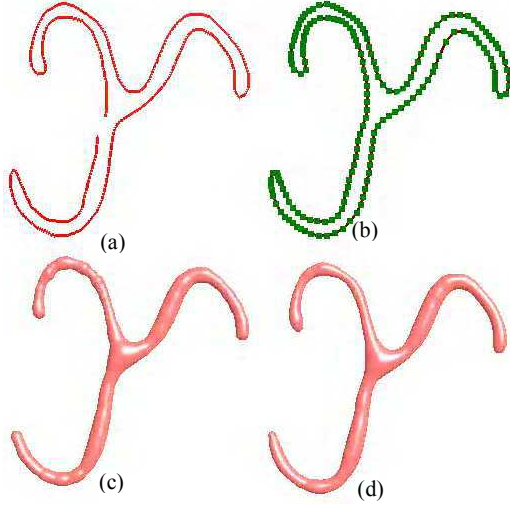


Figure 7. Smoothing the surface: (a) Initial stroke. (b) Stroke adaptively sampled by expanding the sampling trees in the tiny zones. (c) Surface reconstructed with the regular sampling. (d) Surface reconstructed with the adaptive sampling. Note the reduction of the shape oscillations.

The next step is the minimisation process, which adjusts the parameters R_i so that the 3D implicit surface fits the contour.

3.2.3 Adjusting the function parameters

As in previous approaches [27], [8], [45], the adjustment tries to minimise an energy function E that characterises the distance between the surface and the contour points. This is done by summing the squares of the difference between the value of the function f at contour points and the value C that f should return at these points in order to interpolate them. M is the number of contour samples:

$$E = \sum_{j=0}^{M-1} (f(p_j) - C)^2 \quad (3.5)$$

To perform the adjustment, several non-linear least squares minimisation methods (Gauss-Newton, Levenberg-Marquart) have been tested. The best performance in both speed and convergence were obtained using the dogleg trust region method [26]. Only two seconds are necessary on an AMD Athlon 1,3GHz for the method to find an almost zero energy

value (i.e. smaller than 0.1 which is sufficient in our case) when the number of blobs is around fifty. This number of blobs is largely enough for the models that we present. Every stroke is reconstructed individually, as the modelling process is incremental. The surface before and after the adjustment is shown in Figure 5.

A more complete energy function would also contain a tension term within the energy formula [12], which would minimize the mean curvature over the contour points. However, this would considerably slow down the minimisation process (i.e. by a factor of 100 in our tests), and it would not be suitable for interactive times. As our experiments showed, it is not necessary to use a more complete energy formula.

Finally the surface is polygonized for rendering and displayed using a classical Marching Cubes algorithm [6]. Figure 8 compares the mesh obtained with our reconstruction algorithm with the one produced from the same contour using the algorithm from Teddy [21].

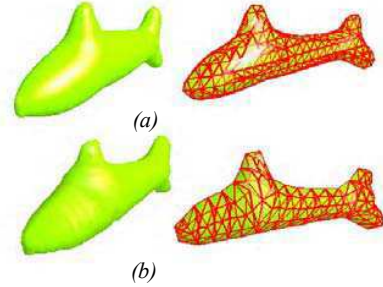


Figure 8. (a) Implicit surface produced by our algorithm. (b) Mesh produced with the algorithm described in [21]. The resultant surface is much smoother when using implicit surfaces.

One can see that our method provides a much smoother surface. A post processing of the mesh (for example, with subdivision surfaces) would also be an option, but in that case we will loose the compact structure of the implicit surfaces, and the mesh should be post processed every time the skeleton operators are applied, which is not convenient.

As we stated in the introduction, the reconstruction with convolution cylinders is not suitable for “folding” strokes. This is shown in Figure 9. We use the same convolution function as in [45], and we compare the stroke reconstructed using their method with the same stroke reconstructed with blobs. In order to remove the unwanted blending at the interior of the fold, the contribution from the cylinders in this zone is diminished, but that causes the surface to move away from the stroke at the extern part of the fold. This is the reason why we preferred to use blobs because they have smaller regions of influence, so they follow faithfully the curve of the fold, as shown in Figure 9 (b) and (d).

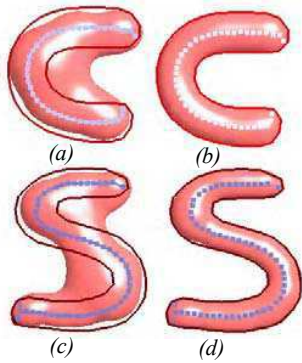


Figure 9. (a), (c) Folding stroke reconstructed with convolution cylinders as described in [45]. (b), (d) The same folding stroke reconstructed with blobs (the Stolte-like function 3.2)

4. EDITING OPERATIONS

The double shape representation (skeleton/3D-shape), gives some advantages to our method. Both of these structures are intuitive to use for shape editing. In addition to the editing possibilities provided by the use of 2D strokes, the user can perform simple operations directly on the skeleton points.

Figure 10 synthesizes most of the operations implemented so far. The automatic blending property of implicit surfaces preserves the surface smoothness and the definition of the surface by blended spheres ensures that the object is and remains solid through editing operations. Some images of objects modelled with our system are shown in Figure 1. Each object was modelled within 10 minutes average time.

4.1 Extrusion

In order to perform an extrusion, the user must first select the blob to be extruded (dark point in Figure 10 (b)). The next step is to draw the extrusion profile by sketching a stroke that is not necessarily closed, as shown in Figure 10 (b). The system then processes the stroke as in creation mode. The skeleton of the stroke is extracted and implicit spheres that approximate the stroke are produced. Finally, the newly created skeleton is connected to the existing one, and the user sees the extruded part that he created.

4.2 Changing the thickness of one or more blobs

In order to modify the thickness of one or several blobs the user first selects the centres of the blobs to be scaled using a selection rectangle (dark points in Figure 10 (c)). Then he moves the mouse or the digital pen up or down, depending on the desired effect: flattening or fattening. Inserting a factor in the squared Euclidean distance computation does scaling.

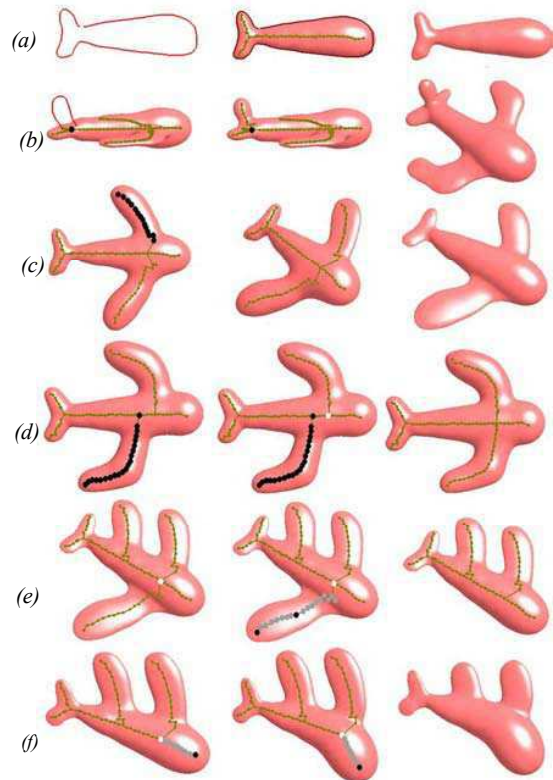


Figure 10. Operations implemented by our system: (a) Creation of a shape from its profile. (b) Creating the plane's tail by extrusion. (c) Flattening the wings and the tail by scaling the blobs. (d) Cut-paste of a skeleton part. (e) Suppression of a skeleton part. (f) Rotation around an articulation point.

For example, in order to perform scaling on the z axis (when the contour is defined in the (x,y) plane), the squared distance $r^2=x^2+y^2+z^2$ is replaced by $r^2=x^2+y^2+\gamma z^2$, where γ is the scaling factor. In order to flatten, γ is chosen to be less than one and for fattening, γ is chosen to be greater than one. When $\gamma \neq 1$ the spherical blob becomes an ellipsoid, hence providing the thickness control. Examples are shown in Figure 10 (c). As done in [45] we can think of using anisotropic distance functions [47], [34], [7], [10]. However, this obliges the user to draw the thickness for a set of blobs and we are not sure that this is an efficient tool to provide. Since we want our modelling metaphor to remain as simple and as intuitive as possible, the control of the shape's thickness in a more appropriate manner will be the topic of our future research.

4.3 Rotation around an articulation

The user selects first the point on the skeleton, which will be the rotation pivot (white point in Figure 10 (f)). Since the skeleton is a connex graph, there are at least two possibilities for the selection of the part of the skeleton to be rotated. The

second selected point identifies the part to be rotated around the pivot (dark point in Figure 10 (f)). The selected branch can now be interactively oriented in the desired direction. All the blobs located in this area (grey points Figure 10 (f)) go through the same rotation transform. The property of automatic blending of the implicit surface guarantees that the smooth surface aspect will be preserved through this transform. In order to avoid the unwanted blending (for example, an arm brought too close to the body will blend with the body) we use the technique described in [18], i.e. we start by computing the contribution from the closest blob, then we progressively add the contribution of this neighbours, and the process is repeated recursively until the contribution becomes negligible. This does not work for the folding surfaces in Figure 9 (so it would still not solve the unwanted blending that appears when using the technique from [45]).

4.4 Copy-paste

First the user has to select the skeleton branch to be copied, doing the same procedure as for rotation. In order to paste the blobs at another location, the user clicks on the blob in which he wishes to paste the branch, as shown in Figure 10 (d) (white point). The copied branch will be pasted in that position, i.e. all the blobs on the branch will be copied, and then translated with a vector, equal to the difference between the old branching point and its new paste location. In the example shown, the user performed a cut-paste operation. The skeleton graph is then updated with the new connection. Then the new branch can be reoriented in a new direction (for instance, symmetrical with the old branch, to form symmetrical arms, legs, ears, etc.).

4.5 Suppression of a skeleton part

For this operation, the selection is done in the same way as for rotation: first the pivot is selected, then the user clicks any point on the branch that he wants to select (see Figure 10 (e)). This identifies all the blobs to be deleted. Pressing the «delete» key suppresses all the selected blobs, except the pivot. The skeleton graph is updated.

5. CONCLUSION AND PERSPECTIVES

We presented a system for modelling 3D shapes from 2D user-sketched strokes. Using the Igarashi et al. skeleton extraction procedure [21] we propose reconstructing the 3D surface in a different manner. Instead of directly extracting a 3D mesh, we use the skeleton to place implicit spheres, which are automatically blended in order to reconstruct a smooth surface. This surface can then be polygonized for interactive rendering. Our method takes advantage of its double representation: the 3D shape and the skeleton. It provides the same functionalities that previous approaches have and in addition it provides a smoother, compact structure surface and it offers new editing tools such as thickness control of the 3D shape, (copy/cut)-paste operation on the skeleton, etc. It becomes possible to rearrange parts of the object without re-sketching them, and our

modelling tool remains very simple and accessible to anyone while providing an efficient shape prototyping method.

The shape can be compactly stored using the spheres' centres and radii, and it can be polygonized with the desired resolution, in order to be visualized or to be exported to other modelling software where small details can be added.

Our model is compatible with the general mesh format provided by classical modelling software and it could be integrated into a classical modelling frame using for example the HybridTree approach [1].

The main drawback of our approach is the difficulty in representing sharp edges (i.e. they are smoothed by the implicit function reconstruction).

Perspectives include the implementation of an adaptive and incremental polygonization algorithm, in order to repolygonize only the parts that have been modified [16], [15]. Now that we have demonstrated the efficiency of our reconstruction approach, we will develop more editing operators in order to provide a full panel of simple and intuitive tools that can be supported by our shape representation.

6. REFERENCES

- [1] Allègre R., Barbier A., Galin E., Akkouche S. *A Hybrid Shape Representation for Free-form Modeling*. Research Report, Liris, Lyon University, 2004, RR-2004-009.
- [2] Aurenhammer F. *Voronoi diagrams - A survey of a fundamental geometric data structure*. ACM Computing Surveys, 1991, 23: pp. 345-405.
- [3] Bernadini F., Bajaj C. L., Chen J., Schikore D.: *Automatic reconstruction of 3D CAD models from digital scans*. International Journal of Computational Geometry & Applications 9, 4, pp. 327-369.
- [4] Bajaj C. L., Bernadini F., Xu G. *Automatic reconstruction of surfaces and scalar fields from 3D scans*. Proceedings of ACM SIGGRAPH 95, pp.; 109-118.
- [5] Blinn J. F. *A Generalisation of Algebraic Surface Drawing*. ACM Trans Graphics, Vol. 1, No. 3, July 1982, pp. 235-256.
- [6] Bloomenthal J. *An Implicit Surface Polygonizer*. Graphics Gems IV (P. Heckbert, ed.), Academic Press, New York, 1994, pp. 324-349.
- [7] Blanc C., Schlick C. *Extended Field Functions for Soft Objects*. Proc of Implicit Surfaces'95 pp. 21-32.
- [8] Bittar E., Tsingos N., Cani M.-P. *Automatic Reconstruction of Unstructured 3D Data: Combining a Medial Axis and Implicit Surfaces*. Computer Graphics Forum, 14(3): pp. 457-468, August 1995.
- [9] Carr J. C., Beatson R. K., Cherrie J. B., Mitchell T. J., Fright W. R., McCallum B. C. *Reconstruction and representation of 3D objects with radial basis functions*. Computer Graphics (SIGGRAPH 2001 proceedings), pp. 67-76, August 2001.
- [10] Crespin B., Blanc C., Schlick C. *Implicit Sweep Objects*. Eurographics' 96, 15(3), pp.165-174.

- [11] Choi H. I., Choi S. W., Moon H. P. *Mathematical Theory of Medial Axis Transform*. Pacific Journal of Mathematics, Vol. 181, No. 1, pp. 57-88, November 1997.
- [12] Desbrun M., Cani M.-P. *Active Implicit Surface for Animation*. Graphics Interfac, June 1998, pp. 143-150.
- [13] De Araujo B., Jorge J. *BlobMaker: Free-form modelling with variational implicit surfaces*. Proceedings of 12^o Encontro Português de Computação Gráfica, Porto, 2003 pp. 17-26.
- [14] Douglas, D.H., Peucker, T.K. *Algorithms for the reduction of the number of points required to represent a digitized line or its caricature*. The Canadian Cartographer 10 (2), 112-122.
- [15] Ferley E., Cani M.-P., Gascuel J.-D. *Practical Volumetric Sculpting*. The Visual Computer no. 8 vol. 16 pp. 469-480, December 2000.
- [16] Galin E., Akkouché S. *Incremental Techniques for Implicit Surface Modeling*. Computer Graphics International'98 312-321, June 1998.
- [17] Guskov I., Vidimce K., Sweldens W., Schröder P. *Normal Meshes*. Computer Graphics Proceedings (SIGGRAPH 2000), pp. 95-102, 2000.
- [18] Hornus S., Angelidis A., Cani M.-P. *Implicit Modelling Using Subdivision-curves*. The Visual Computer, 2-3 (19), pp. 94-104, May, 2003.
- [19] Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W. *Surface reconstruction from unorganised points*. Proceedings of ACM SIGGRAPH 1992, pp. 71-78.
- [20] Igarashi T., Hughes J. F. *Smooth Meshes for Sketch-based Freeform Modeling*. ACM Symposium on Interactive 3D Graphics, ACM I3D'03, 2003, pp. 139-142.
- [21] Igarashi T., Matsuoka S., Tanaka H. *Teddy: A Sketching Interface for 3D Freeform Design*. ACM SIGGRAPH 1999, pp. 409-417.
- [22] Karpenko O., Hughes J. F., Raskar R. *Free-form Sketching With Variational Implicit Surfaces*. Eurographics 2002, TR2002-27, June 2002.
- [23] Kobbelt L., Campagna S., Vorsatz J., Seidel H.-P. *Interactive multi-resolution modeling on arbitrary meshes*. SIGGRAPH 98.
- [24] Kojekine N., Hagiwara I., Savchenko V. *Software tools using CSRBFs for processing scattered data*. Computers & Graphics 27, 2 (April 2003).
- [25] Lim C., Turkyyah G. M., Ganter M. A., Storti D. W. *Implicit reconstruction of solids from cloud point sets*. Proceedings of the third ACM symposium on Solid Modeling and Applications, ACM Press, 1995, pp. 393-402.
- [26] Mizutani E. *Powell's dogleg trust-region steps with the quasi-Newton augmented Hessian for neural nonlinear least-squares learning*. Proceedings of the IEEE International Conference on Neural Networks, Washington, DC, 2, pp. 1239-1244.
- [27] Muraki S. *Volumetric shape description of range data using blobby model*. Computer Graphics, 25(4): 227-235, July 1991.
- [28] Morse B. S., Yoo T. S., Rheiengans P., Chen D. T., Subramanian K. R. *Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions*. Shape Modeling International 2001, 89-98.
- [29] Nishimura H., Hirai M., Kawai T., Kawata T., Shirakawa I., Omura K. *Object Modelling by Distribution Function and a Method For Image Generation*. The Transaction for the Institute of Electronics and Communication for Engineers of Japan, 1985, Vol. J68-D, Part 4, pp. 718-725.
- [30] Norman D. A. *User Centred System Design*. Lawrence Erlbaum Associates, Inc., 1986.
- [31] Ohtake Y., Belyaev A., Alexa M., Turk G., Seidel H.-P.: *Multi-level Partition of Unity Implicit*. ACM TOG (Proc. SIGGRAPH 2003), 22(3): pp. 463-470.
- [32] Ohtake Y., Belyaev A. G., Seidel H.-P. *A multi-scale approach to 3D scattered data interpolation with compactly supported basis functions*. Shape Modeling International 2003.
- [33] Owada S., Nielsen F., Nakazawa K., Igarashi T. *A Sketching Interface for Modeling the Internal Structures of 3D Shapes*. Proceedings of 3rd International Symposium on Smart Graphics, Springer, Heidelberg, Germany, July 2-4, 2003, pp.49-57.
- [34] Parent R., SINGH K. *Polyhedral shapes as general implicit surfaces primitives*. OSU-CIS Technical Report, OSU-CISRC-5/94-TR24, 1994.
- [35] Pasko A., Adzhiev V., *Function-based shape modeling: mathematical framework and specialized language. Automated Deduction in Geometry*. Lecture Notes in Artificial Intelligence 2930, Ed. F. Winkler, Springer-Verlag, Berlin Heidelberg, 2004, pp. 132-160.
- [36] Plänkner R., Fua P. *Articulated Soft Objects for Video-based Body Modeling*. International Conference on Computer Vision, Vancouver, Canada, July 2001, pp. 394-401.
- [37] POV-Ray, <http://www.povray.org/>
- [38] Prasad L. *Morphological analysis of shapes*. CNLS Newsletter, 139: 1-18, July 1997.
- [39] Shersyuk A. *Convolution surfaces in Computer Graphics*. Ph.D. dissertation, School of Computer Sciences and Software Engineering, Monash University, Australia 2000.
- [40] Sanchis N., Jorge J. A. *Direct Modelling: from Sketches to 3D Models*. 1st Ibero-American Symposium on Computer Graphics, July 2002.
- [41] Savchenko V. V., Pasko A. A., Okunev O. G., Kunii T. L. *Function representation of solids reconstructed from scattered surface points and contours*. Computer Graphics Forum 14, 4, pp. 181-188.
- [42] Stolte N. *Espaces Discrets de Haute Résolutions: Une Nouvelle Approche pour la Modélisation et le Rendu d'Images Réalistes*. PhD thesis, University of Toulouse III, France, April 1996.

- [43] Suchman S.A. *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge Univ. Press. 1987.
- [44] Turk G., O'Brien J. *Modelling with implicit surfaces that interpolate*. ACM Transactions on Graphics 21, 4 (October 2002), pp. 855–873.
- [45] Tai C.-L., Zhang H., Fong C.-K. *Prototype Modeling from Sketched Silhouettes based on Convolution Surfaces*. Computer Graphics Forum, 2004. *To appear*.
- [46] Wyvill B., Galin E., Guy A. *The Blob Tree, Warping, Blending and Boolean Operations in an Implicit Surface Modeling System*. Implicit Surfaces' 03, June 1998.
- [47] Wyvill B., McPheeters C., Wyvill G. *Data Structure for Soft Objects*. The Visual Computer, Vol.2, No. 4 August 1986a, pp. 227-234.
- [48] Wyvill B., Wyvill G. *Field functions for implicit surfaces*. The Visual Computer, 5:75 pp. 75-82, December 1989.
- [49] Yoshizawa S., Belyaev A., Seidel H.-P. *Free-form Skeleton-driven Mesh Deformations*. Solid Modeling 2003.
- [50] Zeleznik R., Herndon K., Hughes J. *Sketch: An Interface for Sketching 3D Scenes*. Proceedings of SIGGRAPH'96, 1996, pp. 163-170.
- [51] Zenka R., Slavik P. *New Dimensions For Sketches*. Spring Conference on Computer Graphics 2003 Proceedings, Budmerice, Slovak Republic, 2003.
- [52] Zorin D., Schröder P., Sweldens W. *Interactive Multiresolution Mesh Editing*. Computer Graphics (SIGGRAPH '97 Proceedings), pp. 256-268.

7.2.2 Shape modelling by sketching using convolution surfaces.

Auteurs : Anca Alexe, Loïc Barthe, Marie-Paule Cani, Véronique Gaildrat
Revue : Pacific Graphics (Short papers)
Date : Octobre 2005

Shape Modeling by Sketching using Convolution Surfaces

Anca Alexe¹, Loic Barthe¹, Marie Paule Cani², Véronique Gaildrat¹

1. IRIT - CNRS, UPS Toulouse, France

2. GRAVIR - CNRS, INRIA, UJF, INP Grenoble, France

E-mail: alexe,lbarthe,gaildrat@irit.fr, Marie-Paule.Cani@imag.fr

Abstract

This paper proposes a user-friendly modeling system that interactively generates 3D organic-like shapes from user drawn sketches. A skeleton, in the form of a graph of branching polylines and polygons, is first extracted from the user's sketch. The 3D shape is then defined as a convolution surface generated by this skeleton. The skeleton's resolution is adapted according to the level of detail selected by the user. The subsequent 2D strokes are used to infer new object parts, which are combined with the existing shape using CSG operators. We propose an algorithm for computing a skeleton defined as a connected graph of polylines and polygons. To combine the primitives we propose precise CSG operators for a convolution surfaces blending hierarchy.

Our new formulation has the advantage of requiring no optimization step for fitting the 3D shape to the 2D contours. This yields interactive performances and avoids any non-desired oscillation of the reconstructed surface. As our results show, our system allows non-expert users to generate a wide variety of free form shapes with an easy to use sketch-based interface.

Keywords: sketch based modeling, implicit surfaces, convolution surfaces, CSG.

1. Introduction

The complexity of user interaction is the main obstacle to the use of standard modeling systems. This impacts both the user and the possibilities of expression this system provides. Achieving a simple and faithful translation of the user's idea without requiring sophisticated input and a long training process remains a challenge for the modeling software. One of the simplest and user-friendliest modeling metaphors is drawing. This kind of communication is useful in educational applications such as teaching, and already has industrial purposes such as story boarding. It is generally used in the early stages of design, because drawing a sketch is both much faster than creating a 3D model, and more convenient for expressing ideas. However, the obvious drawback of 2D sketches is their limitation to a single viewpoint. The user cannot move around the drawn object, nor view it from different

angles, except in [4] where the sketch cannot be used for extracting a 3D shape. The aim of the sketch-based modeling is to combine the simplicity of 2D sketching with powerful 3D capabilities. Since the first sketch based interface [11] the concept has been largely developed and explored, from architectural design [5] to artistic design [8] and free form shapes [6, 7, 10]. The latter are difficult to model with sketches, though among the most interesting because of the large modeling possibilities they provide. The main difficulty in reconstructing a 3D model from a 2D contour is extrapolating lacking information. There are two main approaches for constructing smooth, rounded shapes from 2D contours. The first one consists in perspective projections of the contour point samples to reconstruct the 3D geometry. These points are then interpolated using variational implicit surfaces [7, 12, 5]. The second is to construct a skeleton from the 2D contour and use it to generate a 3D shape [6, 10, 1]. The main drawback for the first approach is that the surface has to be recalculated every time it is edited and the time taken to compute the coefficients for the variational implicit surfaces increases with the number of points. Also, small details are lost when blending the object parts because preserving them would require too many constraints and too much computation. Therefore we prefer the second approach. Previous research in this field has raised some difficulties. One of these is the necessity of an optimization step to adjust the implicit surface to the drawn contour. This leads to a better contour approximation in terms of error but the surface oscillates [10, 1]. Moreover it is time consuming and in the context of sketch based interface providing very accurate reconstruction is not necessary. Indeed, the user drawn contour is seldom noisy so we rather aim at getting a smooth shape with close appearance to the contour. Removing the optimization step saves time and reconstructs a smooth non-oscillating surface. Of course the contour approximation constraints have to remain satisfied. Another drawback of most of the previous approaches is that the shape thickness is automatically inferred so the result may differ from what the user wanted. For example if the user draws the shape of the palm of a hand, the fingers will be smoothly reconstructed as cylinders, whereas the palm will look like a sphere, far from the user's expectation. In [10] the problem is addressed by asking the user to provide additional information about the cross section's profile.

This increases the complexity of the interface and for this reason the technique might not be intuitive enough for non-expert users.

Our contribution

We propose a representation that allows for a great variety of topological shapes, a richer collection of sketch-based operations, an adaptive level of detail for sketch modeling with precise control of the result up to small details, while keeping a very simple and friendly user interface. For this purpose we reconstruct the 3D shape using convolution surfaces [3] with both polylines and polygons skeletons. The primitives are composed with CSG blending in a blending hierarchy.

Section 2 presents our system from the user's point of view. Section 3 presents the application from the system's point of view, i.e. the algorithms and the techniques used. Section 4 shows and discusses some results and also draws the conclusions and perspectives of our work.

2. From the user viewpoint

The purpose of our system is to enlarge the possibilities offered by the paper-pencil 3D modeling metaphor, while keeping a simple and intuitive input interface. The modeling process iterates the following steps until modeling is complete:

1. The user draws one or several strokes
2. The strokes are interpreted to reconstruct a 3D object part
3. The part is added to the current object (or subtracted if carving)

As the user draws a stroke, its thickness and color intensity vary proportionally with the pressure on the digital pen, as to imitate the irregular density and thickness of the strokes produced by a real pen. Several strokes accumulated in the same pixel result in a darker color for that pixel. The other end of the pen is used as an eraser. As long as the stroke has not been reconstructed, the user is free to erase and modify it. This way the user's input is allowed to be noisy and irregular, as it is naturally on paper. To create a new shape, the user draws a contour on the graphic tablet using the digital pen. Once the contour has been completed the user presses the digital pen against the tablet. This produces the 3D reconstruction of the stroke (see Fig. 1 (a),(b)). To add part to an existing shape, the mechanism is the same as for creation. The first surface point hit by the user gives the depth of the shape to be constructed. When the stroke is complete, the user presses the stylus if he wishes to add the shape to the existing object, or the eraser (at the opposite pen's end) if he wishes to carve it into the object. The shape is reconstructed in such a manner that the projection of the shape on the screen fits the contour that has been drawn by the user (see Fig. 1 (c),(d),(e) and (f),(g),(h)). The user controls the thickness of the shape using the pen's

bend (see Fig. 1 (i),(j),(k)). Small details can be modeled by zooming to get closer to the object. The large object parts will smoothly blend with each other, while the small details (e.g. eyes, nose of a character) will have a sharper blending. The user can paint directly on the objects or in space next to them. In this way additional information or annotation can be added to the model.

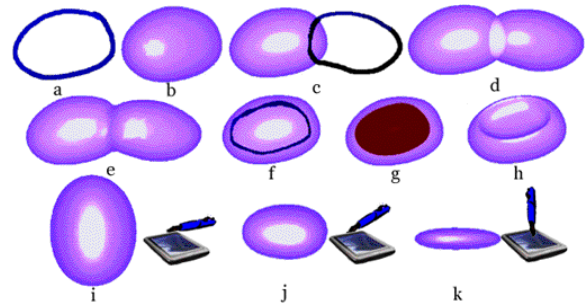


Figure 1: (a), (b) Creating a part. (c),(d),(e) Adding a part to an object. (f),(g),(h) Carving. (i),(j),(k) Thickness control (side view).

3. From the system viewpoint

A pressure threshold indicates that the drawing is finished. When the stylus pressure has reached this threshold, the strokes image is recovered as a 2D bitmap, then compressed and reduced in size using a pixel averaging technique. This also reduces the amount of computation for the skeleton. In order to perform the skeleton extraction we iteratively construct a connected pixels skeleton, which is then sampled in order to obtain a segments and triangles graph [13]. This will be used to define a convolution surface [3]. In order to obtain interactive modeling, we use the pseudo Cauchy [9] convolution kernel, which gives a closed form solution for the convolution integral for the primitives that we use (segments and triangles). See [13] for a full algorithm description.

The addition and subtraction operations are defined using CSG, for which we have adapted the composition model shown in [2] and rewrote the union and difference operators in order to allow hierarchical exact composition. The level of detail of the skeleton remains constant in the image space, but it is automatically adapted to the level of detail of the 3D shape, given by the distance between the object and the camera. The level of detail determines the blending parameters, the skeleton weights and the size of the polygonization cell for the shape to be reconstructed. The polygonization for the reconstructed stroke is computed and displayed immediately, while a process in the background computes the final surface polygonization. The final mesh is displayed as soon as it is available. This allows

maintaining interactive rates and rapid application response during the modeling process so that the user feels free to pursue his modeling activity.

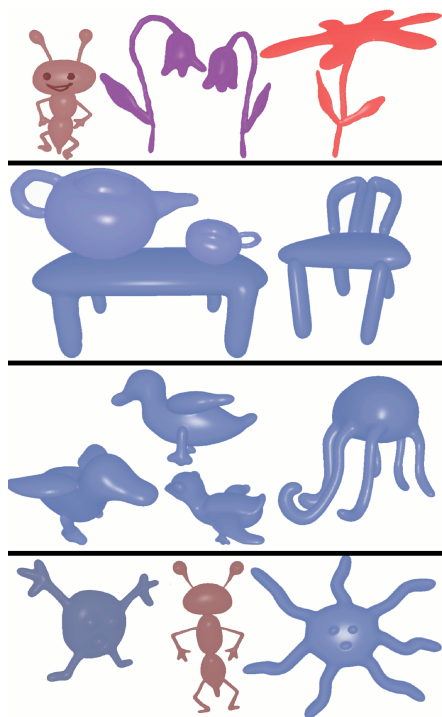


Figure 2: Objects modelled with our system. The user took 2 to 5 minutes overall modelling time and 3 to 9 strokes for each object.

5. Results and conclusions

Convolution surfaces allow much better shape representation than standard skeleton based implicit surfaces, due to their possibility to represent flat surfaces, as well as a large topological variety. Fig. 2 shows objects modeled with our system. The system provides a real simplicity for the non-expert user, for example three strokes only are necessary to create each one of the birds in Fig. 2 (with symmetry enabled for the wings and legs). The Fig. 2 also shows flat surfaces (table and chairs). The shapes have no oscillations and no bulges. The CSG composition is a generalized composition more flexible and accurate than the simple sum, allowing a better blending control, from smooth to sharp transitions. The small details of the objects are well preserved due to the parametrable CSG.

For example, the sun's eyes and mouth are small details compared to the face but they are well preserved by the blending. The shape may have various topologies (ex. chairs, teapot) and can be carved (teapot, mugs). The applications of our system are educational, but also

story boarding for films making (ex. cartoons, see Fig. 2) where the scenarios writer is not necessary a 3D designer. The system could be extended to design the internal structure of organic shapes because the composition model is suitable for this.

In the future we would also like to focus on accelerating the polygonization time for generating the final implicit surface and investigate adaptive polygonization.

References

- [1] A. Alexe, V. Gaildrat, and L. Barthe, "Interactive modeling from sketches using spherical implicit functions", *AFRIGRAPH*, Stellenbosch, November 2004.
- [2] L. Barthe, V. Gaildrat, and R. Caubet, "Combining implicit surfaces with soft blending in a CSG tree", *CSG Conference Series*, April 1998.
- [3] J. Bloomenthal and K. Shoemake, "Convolution surfaces", *Computer Graphics*, 25(4):251-256, 1991.
- [4] D. Bourguignon, M.-P. Cani, and G. Drettakis, "Drawing for illustration and annotation in 3D", *Computer Graphics Forum*, vol. 20, pages 114-122., Blackwell Publishers, September 2001.
- [5] A. Cuno, C. Esperança, P. R. Cavalcanti, R. F. Cavalcanti, and R. Farias, "3D free-form modeling with variational surfaces", *WSCG*, February 2005.
- [6] T. Igarashi, S. Matsuoka, and H. Tanaka. "Teddy: A sketching interface for 3d freeform design", *SIGGRAPH 99*, August 1999, pages 409-416.
- [7] O. Karpenko, J. F. Hughes, and R. Raskar, "Free-form sketching with variational implicit surfaces", *Computer Graphics Forum* 21(3):585-594, September 2002.
- [8] D. F. Keefe, D. Acevedo, Moscovich, Tomer, Laidlaw, H. David, and J. J. LaViola. "Cavepainting: A fully immersive 3d artistic medium and interactive experience", *Symposium on Interactive 3D Graphics*, ACM SIGGRAPH, 2001.
- [9] A. Sherstyuk, "Kernel functions in convolution surfaces: a comparative analysis", *The Visual Computer*, 15(4), 1999.
- [10] C.-L. Tai, H. Zhang, and C.-K. Fong, "Prototype modeling from sketched silhouettes based on convolution surfaces", *Computer Graphics Forum*, 23(4):855-863, 2004.
- [7] R. Zeleznik, K. Herndon, and J. Hughes, "Sketch: An interface for sketching 3d scenes", *ACM Transactions on Graphics*, Proceedings of SIGGRAPH, 1996.
- [12] R. Zenka and P. Slavik, "New dimension for sketches", *In SCCG 2003*.
- [13] A. Alexe, L. Barthe, M.P. Cani, V. Gaildrat, "A Sketch-Based Modelling system using Convolution Surfaces", *Technical Report IRIT/2005-17-R*, July 2005.

7.2.3 Matisse : Painting 2D regions for Modeling Free-Form Shapes.

Auteurs : Adrien Bernhardt, Adeline Pihuit, Marie-Paule Cani, Loïc Barthe
Revue : Eurographics Workshop on Sketch-Based Interfaces and Modeling
Date : Juin 2008

Matisse : Painting 2D regions for Modeling Free-Form Shapes

A. Bernhardt¹, A. Pihuit¹, M. P. Cani¹, L. Barthe²

¹Grenoble Universities (LJK-CNRS) & INRIA, France

²University of Toulouse (IRIT-CNRS), France

Abstract

This paper presents Matisse, an interactive modeling system aimed at providing the public with a very easy way to design free-form 3D shapes. The user progressively creates a model by painting 2D regions of arbitrary topology while freely changing the view-point and zoom factor. Each region is converted into a 3D shape, using a variant of implicit modeling that fits convolution surfaces to regions with no need of any optimization step. We use intuitive, automatic ways of inferring the thickness and position in depth of each implicit primitive, enabling the user to concentrate only on shape design. When he or she paints partly on top of an existing primitive, the shapes are blended in a local region around the intersection, avoiding some of the well known unwanted blending artifacts of implicit surfaces. The locality of the blend depends on the size of smallest feature, enabling the user to enhance large, smooth primitives with smaller details without blurring the latter away. As the results show, our system enables any unprepared user to create 3D geometry in a very intuitive way.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Object Modeling(implicit surfaces); I.3.6 [Computer Graphics]: Methodology and techniques(sketch-based interaction).

1. Introduction

Designing 3D shapes is not easily accessible to the public: sculpting the model one has in mind from a piece of clay or wood requires some skills, while sketching, if easier, will only provide a single 2D view of the shape. Digital shape modeling, supported by impressive advances in shape representation and editing, is an interesting alternative. However, standard modeling systems are far from being usable by un-trained users: they usually require some understanding of the underlying geometrical representation, are often limited to some indirect control (e.g. manipulating control structures or weight values) and require a tedious training process before mastering the interface. Most novice users would get discouraged before being able to enjoy a creative design task.

Recently, systems based on 2D sketching have been proposed to ease the modeling of 3D shapes. However, keeping the simplicity of use of real sketches - where 3D somehow emerges from the sketch - without introducing restrictions on the shape being modeled, is very challenging. This paper presents a solution where smooth free-form shapes of any topology can be constructed by progressively painting

and refining them from different viewpoints. An example is shown in Figure 1.

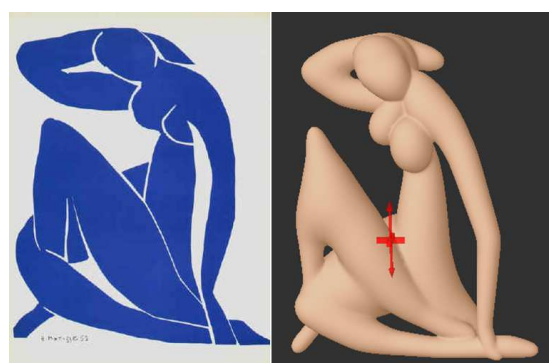


Figure 1: Left: Original art-work from Matisse. Right: 3D model created with our system using 10 sketching steps from different viewpoints. The Matisse art-work served as a loose inspiration.

1.1. Related Work

Sketch-based modeling systems can be divided in two categories. The first category groups systems dedicated to a specific application. This includes modeling clothes [TWB*06], trees [JOI06, OOI07], hair [WBC07] or architectural shapes [Ske]. In these systems, 3D is inferred using some a priori knowledge of the object being modeled. The second category, which includes the seminal paper Teddy [IMT99], groups systems for modeling general free-form shapes from the sketch only. Let us review this second category, to which our work belongs.

The basic idea is to reconstruct a 3D shape from contour curves, interpreted as silhouettes. In most cases, the silhouettes are restricted to closed curves and considered to be planar, which restricts reconstructed shapes to those with a flat medial axis. In contrast, [KH06] allows the user to draw complex contours including cusps and branching point, from which more general 3D shapes can be inferred. Most of these systems allow adding extra shape components from other viewpoints, for instance through cutting and extruding the model [IMT99]. To capture shapes of more general topological genus, some systems allow the creation of holes within the planar sketch of the shape, using conventions such as sketching a contour in clockwise order inside a counter-clockwise outer contour [SWSJ05, KH06]. However, this requires the user to learn specific sketching rules. Our painting metaphor reuses the idea of modeling planar silhouette components extended from different viewpoints. However, we use a painting instead of a sketching metaphor, making the creation of components of arbitrary genus straightforward.

Let us now discuss the methods used in previous systems to fit the sketched contours. A first group of methods belong to variational modeling, and are thus based on optimization: points on the sketched contours, projected into 3D, are used as position constraints for a 3D surface, using either variational implicit modeling [KHR02, ZS03, CEC*05] or Laplacian mesh editing [NISA07]. In [SWSJ05], a 2D variational field function is fitted to each contour, with choice between extrusion, revolution or standard inflation to get a 3D shape. Components sketched from different viewpoints are merged using the blobtree blending mechanisms.

A second approach for fitting a 3D shape is to rely on geometric analysis of the contour to reduce the need for an optimisation: a skeleton (or medial axis) is extracted from the contour and used to generate a 3D shape [IMT99, TZF04, AGB]. A difficulty in reconstructing such a shape is getting a smooth surface. Recent techniques use the skeleton for defining an implicit surface, modeled as blobs [AGB] or as a convolution primitive [TZF04, ABCG05]. On one hand, blobs often create oscillations on the shape. On the other hand, the use of convolution smoothes thickness variations so much that directly fitting the contour is difficult. [ABCG05] is supposed to solve this problem but never explains how it is done. We suspect that ad-hoc correspon-

dances tables were precomputed. [TZF04] instead introduces an extra optimization step.

In this work, we use same convolution surface model (namely, a closed-form solution for the Cauchy-kernel) but we propose a closed form formula based power transform of radius values and adapted iso-values to limit the smoothing of thickness variations. It makes our system fully interactive.

1.2. Contributions

This paper presents an interactive modeling system designed for novice users. Smooth, free-form shapes of arbitrary topological genus are progressively created by blending surfaces created from different viewing angles and zoom factors. Our main contribution is the introduction of a region-painting metaphor, that makes the creation of shape components of arbitrary topological genus straightforward. The main features of our painting system are:

- a mechanism to reconstruct a region with an implicit surface without any optimization step;
- improved, automatic ways of inferring the depth of a shape component from the elements already being drawn;
- a new local blending mechanism, *which only blends components where the user painted some overlap*, and whose parameters are automatically chosen to prevent small details from being blurred into larger shapes.
- an adaptive meshing algorithm which meshes each primitive according to the size of smallest features and locally re-meshes the surface after blending operations.

The remainder of this paper develops as follows: section 2 presents our solution for reconstructing a painted region. Section 3 details the way new shape components are combined with existing ones. Section 4 reviews the interface and presents results. We conclude and discuss future work in section 5.

2. Converting a painted region into a 3D shape

This section details the process we use for modeling a 3D shape from 2D user input. After explaining the benefits of using painted regions rather than silhouettes, we detail the way a convolution skeleton is extracted from a region, and how the convolution surface parameters are chosen so that the shape fits the region.

2.1. Painting regions instead of sketching silhouettes

In most sketch-based systems [IMT99, SWSJ05, AGB, KH06], the user is required to sketch a 2D contour, from which a 3D shape is inflated. Except in [KH06], the contour is a simple closed line, interpreted as a flat silhouette, which restricts the family of shapes one can build. Complex shapes can however be created from this mechanism, by blending flat-silhouette components built under different viewpoints.

In this work, we re-use the idea of defining shape components that all have a flat silhouette, since it makes both design by novices and 3D reconstruction much simpler than with complex contours. However, to ease the user task, we use a *painting metaphor* rather than a sketch-based one: the user selects a brush of a given size and is free to paint either a stroke (closed or not) or to fill a region, which can be done quickly using a paint bucket. An eraser is provided to edit the painting (see the mouth in Figure 2).

Using a painting rather than a sketching metaphor brings several benefits: from the user side, long and thin elements can be painted in a single gesture and edition using an eraser is very intuitive. From the system side, there is no need of monitoring the user input, such as checking that a well defined closed contour has been sketched; moreover, no extra interface is required for enabling the creation of shapes of arbitrary topological genus: complex regions such as the one in Figure 2 will be very naturally defined using the painting metaphor.

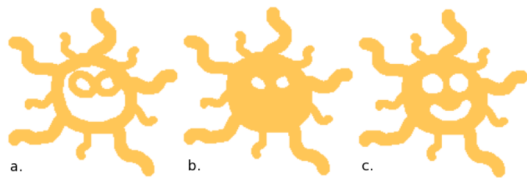


Figure 2: Painting with different sizes of brush (a), filling the surface with paint bucket (b) and erasing for adding the mouth and correcting the eyes (c).

2.2. Converting 2D regions into convolution skeletons

The region the user paints is stored in a texture image of a fixed size for further processing. The second step is to extract the *medial axis* of this region (defined as the locus of maximal circles included within the region) and convert it into a graph of branching poly-lines, usable as the skeleton for a convolution surface [AJC02].

Rather than processing a constrained Delaunay triangulation followed by a chordal axis extraction as in many previous works [IMT99, TZF04, AGB], we use a fast processing method (namely iterative erosion) to extract a set of pixels that approximates the medial axis. [Hal89]. The resulting *skeleton image* (Figure 3 (b)) is stored in a second texture. A benefit of this method is that the resolution of the skeleton is fixed with respect to the texture image: in practice, they will correspond to skeletons of various resolutions in the world referential, since the user freely zooms in and out before painting. Our method thus straightforwardly adapts the resolution of the skeletons to the size of shape features.

In addition to the skeleton image, we compute a Weighted Distance Transform (WDT) that uses a scan-line algorithm to propagate the distance from the border of the region: after

initializing distance values to zero outside the region and one inside, two passes of forward and then backward scan are applied to each pixel P , in scan-line order:

$$f_1(P) = \min(N_5 + a, N_4 + b, N_3 + a, N_2 + b)$$

$$f_2(P) = \min(P, N_1 + a, N_8 + b, N_7 + a, N_6 + b)$$

where a and b are metric weights (we use the 3-4 metric of Figure 3) and $N_i = 1..8$ are the pixel's neighbors. This provides each pixel, including those of the skeleton image, with the radius of the maximal disc centered at this pixel and included into the region (see Figure 3 (c)).

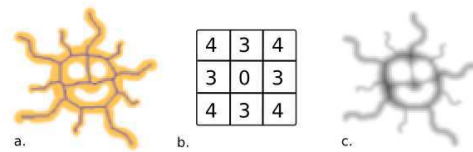


Figure 3: Skeleton image (a), Mask used for WDT (b) and Distance image (c)

The last step is to convert the skeleton and distance images to a convolution skeleton, that is a graph of branching poly-lines with adequate weight at vertices.

The pixel-based skeleton is first re-sampled and converted into poly-lines at an adequate resolution. To be sure that both the shape of curved branches and the variation of radii along branches are well captured, we use a sampling resolution which depends on *both geometry and brush size*. We first create vertices at skeleton pixel positions that correspond to branching points between three branches of more (pixels with 3 neighbors or more), and at pixels with a single neighbor, corresponding to the extremities of branches; then we refine each branch by adding vertices at skeleton pixels corresponding to the local extrema of curvature along the branch. Finally, we refine the branch further if needed, using a resolution depending on the brush radius. Experimental tests show that resampling the branch is necessary if the radius varies over a threshold of 10% of its maximal value. A result is shown in Figure 4 (a).

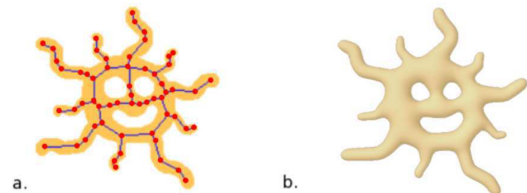


Figure 4: Convolution skeleton (with branches) extracted from a region (a) and the resulting convolution surface (b). Note that the user gets a smoothed version of his design.

This process outputs a graph of branching poly-lines, which will be used as a *convolution skeleton*. A radius value r_i corresponding to the pixel value in the WDT is stored at each vertex S_i of this skeleton.

2.3. From the skeleton to a convolution surface

Our assumption while reconstructing a 3D shape from a 2D region is that the local thickness of the shape in the third dimension should be similar to its thickness in the painting plane. The validity of this assumption will be discussed in Section 4. Implicit surfaces are a natural choice for reconstructing a smooth surface from a skeleton. Moreover, this representation will ease the blending of shape components designed from different viewpoints. To be able to use graphs of poly-lines as skeletons while avoiding the well known bulging problems at joints [BW97], we base our method on convolution surfaces. More precisely, we use the closed-form formula for the convolution integral of the Cauchy kernel, also used in [TZF04]. The implicit surface is defined as the set of points P such that: $F(P) = \mathbf{T}$ where \mathbf{T} is a given isovalue and the field function F is the integral of the kernel function h along the skeleton S_k :

$$F(P) = \int_{S_k} h_S(P) dS \quad (1)$$

Since an integral over a given support is equal to the sum of the integrals over a partition of this support, F is computed by summing the integrals along the different line-segments composing the skeleton. To get a fast yet precise computation, we use the Cauchy kernel, which yields a closed form solution for the convolution integral [She99]:

$$h_S(P) = \frac{1}{(1 + s^2 d^2(P, S))^2} \quad (2)$$

where $d(P, S)$ is the distance between the point P and the skeleton point S , and s is a tangent parameter that allows to control the kernel width, and thus the way this primitive will blend with others.

As shown in [TZF04], the closed form solution for the convolution integral of the Cauchy kernel can be extended to line-segments with a varying radius. Using the formulation from [TZF04], we express the field value at a point p as:

$$F(P) = w_H F_H(P) + \frac{w_T - w_H}{l} F_T(P) \quad (3)$$

where w_H (resp. w_T) is the weight of the segment's *head* (resp. *tail*), $F_H(P)$ (resp. $F_T(P)$) is a field value only depending on the branch's *head* (resp. *tail*) and l is the branch length.

2.4. Choosing convolution parameters

Intuitively, one would expect that in using the desired surface radii as weights w_H and w_T in the formula above, a surface of the right thickness would be constructed. Unfortunately,

some extra tuning is required: due to the integral formulation of convolution surfaces, the resulting surface is thicker than the weight values, due to the summed contribution of all points S along the skeleton. Previous work solved the problem by either manually tabulating the correspondence between weights and thicknesses and pre-inverting the table [ABCG05] or by using iterative optimization to find the weight values that best fit the region [TZF04].

We propose a simpler solution. Based on the fact that we work with texture images of fixed size, and thus that the size of the shape to reconstruct is always within the same range of values, we compute convolution weights by scaling the radius values at vertices using:

$$w_i = (C * r_i)^3 \quad (4)$$

where r_i is the desired radius along the branch, and C is a constant factor depending on the image size that normalizes distances. In practice, $C = \frac{1}{(\text{width (in pixels) of the image}) * 3}$ works well.

We first experimented with $w_i = C * r_i$ but curvatures of the reconstructed shape were much too smooth compared to the original drawing. We introduced the power of 3 to anticipate for convolution's smoothing. This formula for the weights scales the field function to the right range of values.

Our last enhancement is to insure that the surface fits the region by selecting the iso-surface of the field function: we compute the field value at points located on the contour, near the extremity of each branch. This gives us the iso-values the surface should go through. To best fit the region, we set the iso-value T to an average of these values.

The convolution surface resulting from this step is depicted in Figure 4 (b). As expected, the surface is smooth but only approximates the painted region. The quality of this reconstruction will be discussed in Section 4 (including the remaining problem in case of high thickness variations along a branch).

2.5. Meshing a surface component

The convolution surface resulting from the process we just described is tessellated into a mesh using [Blo94]. We set the grid resolution to half of the smallest radius value along the convolution skeleton, which corresponds to the size of the smallest feature to reconstruct.

To make field queries more efficient, we truncate the field function generated by each line-segment outside of a bounding region, using the fact that integrals of the Cauchy kernel quickly fall to zero at a distance from the skeleton. In our implementation, this is done by computing an axis parallel bounding box around each line segment augmented with discs corresponding to the local surface radius; then the size of the box is multiplied by a constant factor, and the field value is considered to be zero outside of this larger box.

3. Combining different shape components

3.1. Positioning new shapes into the scene

The Matisse system uses 2D painting from different viewing angles to generate 3D shapes. When someone paints or draws on a sheet of paper, he or she can imagine the relative depth position of the scene elements and thus "see" a 3D scene. To design an intuitive sketch-based system, we have to guess this relative depth position in a plausible way.

Frontal painting at the right depth : The first idea is to paint on a plane facing the camera, perpendicular to the viewing direction. When the user clicks on the scene using the painting tool a frontal painting plane is created. Our strategy for selecting a depth for this plane when no other information is provided is to always keep the same depth value in the camera's referential. This way, the distance to the painting plane adapts to the zoom level: when the user zooms in, he or she can paint closer details, while large background elements can be easily painted when he or she zooms out.

In most cases, the user has already started painting a shape and would like to add another component or some details to it from a different viewpoint. We then use a relative positioning mechanism: when the painting starts on an existing shape, we create a frontal plane at the depth of the point the user clicked on similarly to [CHZ00]. Since the skeleton of the new component is located in this plane, this results in slightly overlapping shape components, which allows for smooth blending between sub-shapes, as explained later in this section.

In order to give better feedback to the user when browsing the scene, we always display an almost transparent frontal plane which indicates where the next drawing will take place if the user clicks at the current mouse position. This virtual plane is rendered like clear plastic so that the meshes behind it are lighter (see Figure 5 (left)). Whatever the painting plane, the shape is created at the scale of the painted image. Then it is positioned in 3D at the right depth and scale, so that it superimposes with the painted region [CHZ00].



Figure 5: Semi-transparent display of the frontal plane the user has selected while sketching a nose (left) and the result (right)

Bridging between existing shapes : In some cases, the user would like to paint a connection between several existing shape components. In this case, the painting is not expected to take place in front of the camera, but in an oblique

plane connecting these elements [CHZ00]. In our implementation, we allow such a bridging mechanism (see Figure 6). It can either be used for bridging between two shapes (e.g. painting a wire between two trees) or between two points on the same shape (see the arm on the right of Figure 1). We select the two points giving the depths to interpolate as midway between the front and the back depth of the shape at the point the user clicks on. Since passing through two points is an under-constrained problem, we keep the painting plane vertical with respect to the current camera view. When the user paints on an oblique plane, his painting is

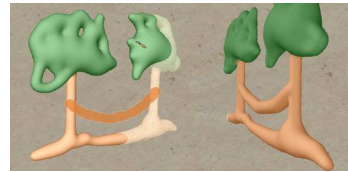


Figure 6: Bridging between two shapes: sketch (left) and rotated result (right)

distorted with respect to the shape component he is creating. To keep the benefits of the constant size texture image used in skeleton computation, we still compute the skeleton as if the region was facing the camera. Then we project it in the oblique painting plane while editing radius values so that the shape's frontal projection still corresponds to the painted region. The resulting oblique skeleton generates the convolution surface.

3.2. Soft blending

After reconstruction, new parts are blended to the object. Since our objects are reconstructed using truncated convolution surfaces, they can be considered as implicit surfaces with compact support so the whole range of blending operators can be used. This goes from a simple sum [Bli82, BW90] to more advanced arc-of-an-ellipse [BWG04] or displacement [HL03] blends.

Since our system allows the user to create large components as well as very small ones in the same framework and with an adapted resolution, the blending operator we choose must overcome the well-known problem of small objects "absorption" when blended with significantly larger ones [WW00], i.e. the small part being completely embedded into a smooth bump on the large object. A first solution to this was proposed in [BWG04]. But it only holds for a large object blended with one or several smaller ones of equivalent size, which is not usable in our case. We rather use the second version of the arc-of-an-ellipse operator also described in [BWG04]. It provides an accurate control of the transition size independently on each blended primitive, making it especially well suited to our purpose.

Let us denote M_1 and M_2 the meshes representing a pair

of objects to combine, noted O_1 (defined by function F_1) and O_2 (defined by function F_2). The operator we use restricts the blending to a local volume between two isovalues : V_1 (resp V_2) the value of function F_1 (resp F_2) at the blending boundary on O_2 (resp O_1) (see Figure 7). The idea is to carefully choose these isovalues so that the smallest surface component is not blend into the larger one. To do this we evaluate F_1 (resp F_2) in a point P'_1 (resp P'_2) at an adequate distance from object's O_1 (resp O_2) along the field gradient. So we have $V_1 = F_1(P'_1)$ and we choose :

$$P'_1 = P_1 - w_{2,min} \frac{\nabla F_1(P_1)}{\|\nabla F_1(P_1)\|} \quad (5)$$

Where P_1 is chosen on O_1 closed to the intersection and $w_{2,min}$, the minimum in the weights of object O_2 skeleton (see section 2.4), gives a good estimation of O_2 smallest detail.

3.3. Local application of the blend

A well known problem with implicit modeling is unwanted blending. When the user creates a complex model by successively blending several primitives, the later will very often blend all along. For instance, modeling an arm which blends with the body of a character at the shoulder but remains separated elsewhere is not easy. It usually requires the use of complex mechanisms such as blending graphs and decay functions [AJC02]. Fortunately, the controlled blending operators we use in our application enable us to set up a simpler, intuitive solution.

First of all, we consider that the user has expressed his wishes about regions to be blended by painting some local overlap between shapes (see Figure 7 left). Once a new shape component is constructed, we only blend it where its mesh overlaps with others. We achieve this through some processing on the meshes that represent the implicit surfaces to be blended: we compute the union of the meshes (a first approximation for the mesh of the blended shape we are looking for), together with the contour(s) of the intersection (see Figure 7 (a)), expressed as a list of segments. Note that there may be several separated such contours if the new component overlaps with the existing shape in several distinct regions (in our implementation this is done using the LGPL GTS library).

Starting on each contour, we remove from the union mesh the region affected by the local blending operations described in the last section. This is done by testing, at each vertex, the value of the field function representing the blended shape. If this value is not close to zero, the vertex belongs to the region to re-compute. Faces entirely constructed on such vertices are removed (see Figure 7 (b), (c)). Then a triangle strip is created to fill the gap between the remaining parts of the mesh. This strip is progressively refined by recursively subdividing triangles into 4 triangles where needed

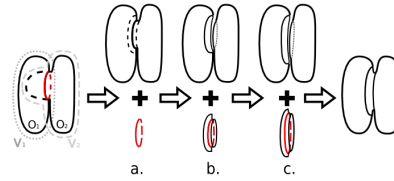


Figure 7: Local blending of two shapes : intersection contour extraction (a), intersection removal (b), blending surface remeshing and growing (c).

according to a curvature criterion: new points at the middle of edges are created and converged to the new, blended implicit surface each time their field value is smaller than the threshold.

A benefit of this mechanism is that, although we do not use any blending graph structure, blending remains local: two shapes blend in the region where the user made them overlap (slightly extended according to the parameters of the local blending operators). If other parts of the same surfaces come close to each other elsewhere, but without having their meshes overlap, they will not blend in that other region. This makes the system much more predictable than usual implicit modeling, although the resulting mesh is no longer the iso-surface of a usually defined field function.

4. Results and performances

4.1. Overview of the interface

Matisse provides the user with many tools not only for painting and for exploring the 3D scene but also for saving and re-using his work. These tools are:

- a paint bucket, a brush and an eraser of adjustable size
- a color-selection panel
- zoom, rotation and translation, usable at any time
- *undo*, *redo*, *save*, *load* and screenshot buttons

In everyday life, after drawing the global shape of an object, the average person's behavior is to rotate the sheet of paper to ease sketching in another direction, or to come closer for adding details. To keep this simplicity of use, our system does not require the user to click any button to re-construct a 3D shape before further sketching: the 3D shape is automatically computed and displayed while he or she changes the point of view by zooming or moving into the scene. This is done without freezing the interface since the computation of the surface is done in threads. Thus, as in everyday life, the user is only concerned about painting and moving his drawing for filling it out. The resulting, intuitive interface can immediately be used without a learning stage.

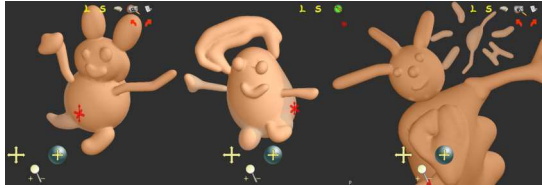


Figure 8: Creations by 18, 67 and 12 year old novice users

4.2. Performances

In Matisse adding a surface component requires 3 main steps : skeleton computation, initial meshing, and blending. Table 1 gives some performances times on a 1.8 Ghz centrino computer (times given in milliseconds). Note that the meshing mechanism we used (Section 2.5) was far from optimized. Results would be made much faster by improving this step.

| Model | Implicit Reconstruction | Skeleton Meshing | Blending with existing component |
|-----------------|-------------------------|------------------|----------------------------------|
| Sun | 120 | 6500 | - |
| Star's body | 70 | 1180 | - |
| Tree's trunk | 30 | 1040 | - |
| Tree's branches | 100 | 5360 | 750 |
| Tree's bridge | 20 | 310 | 740 |

Table 1: Performance times for the Star (Fig 5), the Sun (Fig 2), Trees and Bridge (Fig 6)

4.3. Informal validation by end-users

Since Matisse is designed for the general public, we tested the system with two kinds of users: novice users and traditional artists. The category of novice users was composed of about 10 teenagers and seniors, using the system with a simple mouse on a laptop. Their main feedback was to ask for colors, since they were provided with an earlier, mono-color version of the system. These users enjoyed both the simplicity of painting and the 3D nature of the shapes. Teenagers got accustomed very quickly to navigating through the scene and zooming to add local details. They never complained about the computed 3D shapes, although each surface component did not exactly fit their input so that they often had to bend several components to fit a slope with extreme thickness variations (see Figure 9). However, they would have liked a 3D eraser tool to remove parts of the surfaces that they disliked.

The second category of users was artists, using the system on a tablet: a CG designer and a traditional artist who never worked on computer. The traditional artist was impressed by the functionalities provided, such as *undo/redo* or *load/save*. Both asked for a more precise reconstruction of their input and for the possibility to create flat shapes, such as leaves or petals (see Figure 10). Overall, both enjoyed the simplicity

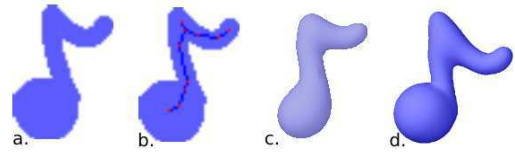


Figure 9: Our system only provides an approximated reconstruction of regions with extreme thickness variations: painting region (a), convolution skeleton (b) and resulting 3D shape (c), a shape reconstructed by blending 2 components (d).

of Matisse for creating free-form 3D shapes and asked to come back for a further test session.

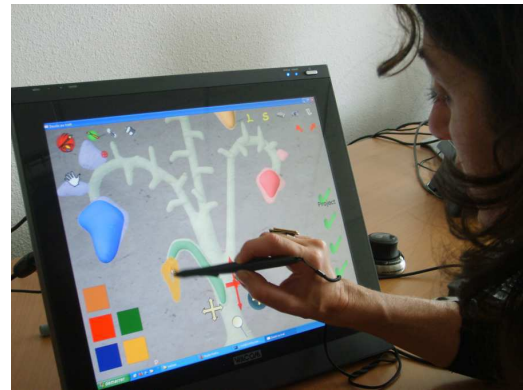


Figure 10: A traditional artist using Matisse

5. Conclusion

This paper presented an interactive modeling system designed for novice users. It enables the creation and refinement of smooth free-form shapes by progressively painting the regions they cover from different viewing angles and with adapted zoom factors. An implicit formulation based on convolution surfaces is used to reconstruct each part of the shape and blend it locally to the existing geometry. Our results shows the relevance of combining shapes of arbitrary genus, each reconstructed from a single drawing. Matisse can immediately be used by unprepared, novice users.

Being able to remove a region from a shape instead of only adding components was required by several users. This extension should be easy thanks to the implicit surface formalism we are using: we just need to include a difference operator, together with the adequate local re-computation of the mesh. Moreover, we are planning to improve the placement of details sketched over larger shapes by projecting their skeleton onto the underlying surface and extending the bridge mechanism to bridge between three scene elements, removing the ambiguity of a plane defined by two points.

Local blending was achieved thanks to an hybrid implicit/mesh representation enabling to apply the blend in regions where the meshes intersected. A useful extension would be to provide a similar mechanism while keeping a well-defined, implicit representation for the whole shape.

Future work will also include investigating a more accurate reconstruction method for the painted regions. We would, however, like to keep the method purely geometric, since optimization would reduce performance and be less controllable in terms of smoothness.

Lastly, the thickness of each reconstructed element is related to its local size in the current version of the system and cannot be edited. Generating flat surface components could be done with surface skeletons as in [ABCG05]. However, an intuitive way for the user to select the desired thickness while sketching still has to be thought of.

5.1. Acknowledgments

We are very grateful to Gregoire Aujay for his important contributions to the implementation of Matisse, and to the artist Virginia Alfonso for helping us to validate the system. We also would like to thank the company Axiatex and the PPF "Interaction multimodale" for funding this project.

References

- [ABCG05] ALEXE A., BARTHE L., CANI M.-P., GAILDRAT V.: Shape modeling by sketching using convolution surfaces. In *Pacific Graphics* (Macau, China, 2005), Short paper.
- [AGB] ALEXE A., GAILDRAT V., BARTHE L.: Interactive modelling from sketches using spherical implicit functions. In *AFRIGRAPH '04*, ACM, pp. 25–34.
- [AJC02] ANGELIDIS A., JEPPE P., CANI M.-P.: Implicit modeling with skeleton curves: Controlled blending in contact situations. In *Shape Modeling International* (2002). Banff, Canada.
- [Bli82] BLINN J.: A generalization of algebraic surface drawing. *ACM Transaction on Graphics* 1, 3 (1982), 235–256.
- [Blo94] BLOOMENTHAL J.: An implicit surface polygonizer. *Graphics gems IV* (1994), 324–349.
- [BW90] BLOOMENTHAL J., WYVILL B.: Interactive techniques for implicit modeling. *Computer Graphics (Proc. of SIGGRAPH 1990)* 24, 2 (1990), 109–116.
- [BW97] BLOOMENTHAL J., WYVILL B. (Eds.): *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.
- [BWG04] BARTHE L., WYVILL B., GROOT E. D.: Controllable binary operators for soft objects. *International Journal of Shape Modeling* 10, 2 (2004), 135–154.
- [CEC*05] CUNO A., ESPERANÇA C., CAVALCANTI P., CAVALCANTI R., FARIAS R.: 3d free free-form modeling with variational surfaces. *WSCG* (2005).
- [CHZ00] COHEN J. M., HUGHES J. F., ZELEDNIK R. C.: Harold: A world made of drawings. In *NPAR 2000 : First International Symposium on Non Photorealistic Animation and Rendering* (June 2000), pp. 83–90.
- [Hal89] HALL R. W.: Fast parallel thinning algorithms: parallel speed and connectivity preservation. *Commun. ACM* 32, 1 (1989), 124–131.
- [HL03] HSU P., LEE C.: The scale method for blending operations in functionally-based constructive geometry. *Computer Graphics Forum* 22, 2 (2003), 143–158.
- [IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: a sketching interface for 3d freeform design. In *SIGGRAPH '99* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 409–416.
- [IOI06] IJIRI T., OWADA S., IGARASHI T.: The sketch l-system: Global control of tree modeling using free-form strokes. In *Smart Graphics* (2006), pp. 138–146.
- [KH06] KARPENKO O. A., HUGHES J. F.: Smoothsketch: 3d free-form shapes from complex sketches. In *SIGGRAPH '06* (New York, NY, USA, 2006), ACM, pp. 589–598.
- [KHR02] KARPENKO O., HUGHES J. F., RASKAR R.: Free-form sketching with variational implicit surfaces. *Computer Graphics Forum* 21, 3 (2002), 585–594.
- [NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: Fibermesh: designing freeform surfaces with 3d curves. *ACM Trans. Graph.* 26, 3 (2007), 41.
- [OOI07] OKABE M., OWADA S., IGARASHI T.: Interactive design of botanical trees using freehand sketches and example-based editing. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses* (2007), ACM, p. 26.
- [She99] SHERSTYUK A.: Kernel functions in convolution surfaces: a comparative analysis. *The Visual Computer* 15, 4 (1999), 171–182.
- [Ske] Sketchup software: 3d sketching software for the conceptual phases of design. <http://www.sketchup.com>.
- [SWSJ05] SCHMIDT R., WYVILL B., SOUSA M., JORGE J.: Shapeshop: Sketch-based solid modeling with blobtrees. In *SBIM* (2005).
- [TWB*06] TURQUIN E., WITHER J., BOISSIEUX L., CANI M.-P., HUGHES J.: A sketch-based interface for clothing virtual characters. *IEEE Computer Graphics and Applications* 27 (2006), 72–81.
- [TZF04] TAI C., ZHANG H., FONG J.: Prototype modeling from sketched silhouettes based on convolution surfaces. *Computer Graphics Forum* 23 (2004), 71–83.
- [WBC07] WITHER J., BERTAILS F., CANI M.-P.: Realistic hair from a sketch. In *Shape Modeling International* (June 2007).
- [WW00] WYVILL B., WYVILL G.: Better blending of implicit objects at different scales. *SIGGRAPH Sketch* (2000).
- [ZS03] ZENKA R., SLAVIK P.: New dimension for sketches. In *Spring Conference on Computer Graphics* (Budmerice, Slovak Republic, 2003).

7.3 Surfaces implicites

7.3.1 Two-dimensional potential fields for advanced implicit modeling operators.

Auteurs : Loïc Barthe, Niel A. Dodgson, Malcolm A. Sabin, Brian Wyvill, Véronique Gaidrat

Revue : Computer Graphics Forum, Vol 22(1)

Date : Mars 2003

Two-dimensional potential fields for advanced implicit modeling operators

L. Barthe, N.A. Dodgson, M.A. Sabin, B. Wyvill and V. Gaildrat

University of Cambridge, University of Cambridge, Numerical Geometry Ltd, University of Calgary, University of Toulouse

Abstract

Current methods for building models using implicit volume techniques present problems defining accurate and controllable blend shapes between implicit primitives. We present new methods to extend the freedom and controllability of implicit volume modeling. The main idea is to use a free-form curve to define the profile of the blend region between implicit primitives.

The use of a free-form implicit curve, controlled point-by-point in the Euclidean user space, allows us to group boolean composition operators with sharp transitions or smooth free-form transitions in a single modeling metaphor. This idea is generalized for the creation, sculpting and manipulation of volume objects, while providing the user with simplicity, controllability and freedom in implicit modeling.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computational Geometry and Object Modeling]: Curve, surface, solid, and object representations;

1. Introduction

Providing interactive, precise and intuitive control of shapes is a fundamental issue in the development of three dimensional modeling techniques. Direct manipulation of meshes, parametric shape representations and, more recently, subdivision surfaces are common and useful solutions adopted by most commercial software. Implicit volume models are rapidly becoming a practical alternative to these methods due to the increase in computing power and storage capacity of modern workstations combined with the latest developments in graphics hardware. Better hardware along with improved volume visualisation algorithms^{1,2} and data structures³, allow us to interactively and accurately render iso-potential surfaces or potential variations in an implicit volume.

In this paper, we denote as volume objects or implicit volumes three-dimensional objects which are defined by a potential field $f(p)$ that associates a potential value with each point p of the Euclidean space \mathbb{E}^3 . Commonly used surface representations such as sub-division surface techniques do not provide a true three-dimensional representation of the object. A surface in a volume representation is a set of points defined with an iso-potential value (an implicit surface). Vol-

ume objects have several important advantages: inside and outside can be distinguished easily, they allow efficient collision tests, high quality triangle meshes of iso-potential surfaces⁴, classical Boolean operations⁵, blending^{6,7,8}, and more advanced sweeping by moving solid⁹, Boolean composition with soft transitions^{10,11} and Constructive Volume Geometry algebra¹².

Most of these techniques are based on the blending properties of implicit surfaces. Early work used the addition operator between field values to provide smooth transitions (blends) between implicit primitives^{6,13,14}. These transitions were approximately controlled by parameters embedded in the implicit function that defined these methods. Later work exploited the locality property of the primitives as a powerful method to build complicated objects from a small number of primitives combined with a large range of operators^{11,15}. Composition operators like the ones proposed by Hoffmann et al¹⁶ and Pasko et al¹⁰ demonstrated that smooth transitions could be obtained using Boolean operators on volumes defined by the inequality $f(p) \leq 0$ (see also ¹⁷), and Barthe et al¹⁸ showed that, in a restricted application, accurate control of the transition could be obtained. One of the big advantages over other modelling techniques is that a variety

of composition operations can be easily performed between implicit primitives, thus accurate and intuitive control over these operations is a critical step to providing interactive and efficient volume modeling software.

The goal of this work is to provide methods that will simply and accurately control the transitions in composition operators and more generally increase the freedom in the manipulation of volume objects. We use the theoretical interpretation of composition operators described by Barthe et al¹⁸. This leads us from the definition of free-form implicit curves controlled point-by-point and with regular field variations, to the generalization of Boolean composition operators, sculpture and modeling tools in a unique operator based on the manipulation of this implicit free-form curve in the Euclidean modeling space. This greatly increases the simplicity, controllability and freedom in volume modeling.

The organization of this paper is as follows: we first present a summary of the different modeling techniques commonly used to build volume objects. Bounded primitives (such as “Soft Objects”¹⁴), real-functions and sampled potential field manipulations are described. It appears clear that, whatever the model used, improvements in control are desirable when volume primitives are composed.

This is followed by the presentation of free-form implicit curves and, more generally, their two-dimensional potential fields. Since these profiles are used to combine or define volumes, particular attention is focused on the variations of the field around the iso-potential curve. In this section, we present open and closed curves and we show the progress we have made in controlling the variations of the fields.

In Section 4 we present the application of free-form curves on volumes defined by the inequality $f(p) \leq 0$. This category regroups and generalizes most of the volume primitives, as shown by Adzhiev et al¹⁷. The possibility of creating open or closed free-form curves allows us to merge the Boolean composition operators (with or without smooth transitions) and operators to sculpt or create primitives in a single modeling metaphor: the extrusion of an implicit curve in an “implicit space”. Whatever the operator, the user simply acts on the two combined potential functions f_1 and f_2 through the deformation of the implicit curve $G(f_1, f_2) = 0$. We briefly illustrate the interactive manipulation of the implicit curves in a two-dimensional section of the Euclidean modeling space using the modeling tool introduced by Barthe et al¹⁹ and we discuss the limits of the intuitiveness and accuracy of this process.

2. Related work

Two different categories of potential function can be distinguished for modelling volume objects: the first form has functions which equal *zero* outside a boundary, and the second has functions which vary over the whole of space. This last form is more expensive in evaluation. Even though our

paper only deals with non-bounded primitives, we briefly present both representations to allow the reader to clearly differentiate them.

2.1. Bounded primitives

“Metaballs”¹³ or “Soft Objects”¹⁴ are bounded objects defined by a potential function f equalling zero everywhere outside the object’s boundary. Inside this boundary, the potential varies from *zero* to *one* and the volume object is defined by the set of points of \mathbb{E}^3 for which $f(p) \geq C$ (where C is a pre-chosen value in $]0, 1[$). A wide variety of primitives are available^{7, 20}, and the blend is automatically computed by summing the potential functions of the primitives. Many different blending functions^{21, 22} and blending models^{23, 24} have been proposed to control the smoothness of the transition region, but the operators remain limited to the blending and the control of which primitives must and must not blend. The locality of the definition and the capacity to be automatically blended allow modeling techniques based on these objects to be interactive^{25, 26}.

CSG composition operators are already supported by bounded primitives (using the Ricci’s *min/max* operators⁵) but C^1 discontinuities are introduced in the potential field of the resulting object, altering the smoothness of the transition when it has to be blended. This is undesirable.

A solution using Ricci’s super-elliptic operator⁵ to apply binary union and binary intersection operators with smooth transitions to “Soft Objects” was used by Wyvill et al¹¹ (see Equation 1), and extended to n-ary operations.

$$G(f_1, f_2) = (f_1^n + f_2^n)^{\frac{1}{n}} \quad (1)$$

2.2. Real functions and sampled potential fields

R-functions (real functions) and sampled potential fields can be grouped into a second category¹⁷ where potentials are given over all the Euclidean space \mathbb{E}^3 and the volume object is defined by the inequality $f(p) \leq 0$. The opposite convention, where the volume is defined by $f(p) \geq 0$, can also be used. Bounded objects, described above, can be adapted to R-functions by considering only the distance field minus the radius of the primitive. Other primitives can be obtained from different sources when the potential field is reconstructed as a distance field^{27, 28, 29}. These objects are fundamental for volume modeling, as much for the variety of primitives as for the generalization of volumes to a unified model. For R-function operators, volume objects are defined with the convention: $f(p) \geq 0$. It has been shown by Pasko et al¹⁰ how to apply binary CSG operators (with or without smooth transitions), space mapping operators and others. Equation 2 gives the example of the union operator with smooth transitions.

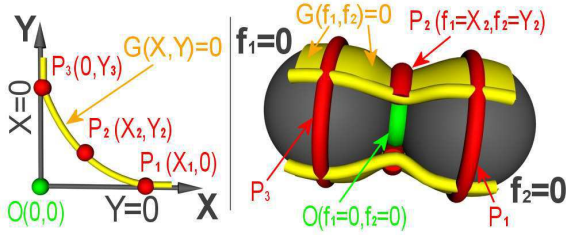


Figure 1: Representation of the same profile $G(X, Y) = 0$ in \mathbb{I}^2 on the left and in \mathbb{E}^3 on the right, when f_1 and f_2 are two spherical potential fields. The resulting iso-potential surface (in yellow on the right) has been cut away to show the details underneath.

$$G(f_1, f_2) = \left(f_1 + f_2 + \sqrt{f_1^2 + f_2^2} \right) + \frac{a_0}{1 + \left(\frac{f_1}{a_1} \right)^2 + \left(\frac{f_2}{a_2} \right)^2} \quad (2)$$

Parameters a_0, a_1, a_2 control the form of the transitions that do not have boundaries. Many other operators have been proposed in the literature^{16, 30, 8}. While advanced operators, based on R-functions and displacement functions with local area of influence, allow the user to specify the location of the blend³¹, the level of control on the transition itself remains globally equivalent. To correctly blend primitives, operator G must satisfy specific properties that are well defined⁸. A binary operator G applied on potential functions f_1 and f_2 can be written as a two-dimensional potential function $G(X, Y)$ composed with the combined three-dimensional potential functions f_1 and f_2 (Equation 3).

$$\begin{aligned} G: \mathbb{R}^3 &\rightarrow \mathbb{R} \\ (x, y, z) &\rightarrow G(f_1(x, y, z), f_2(x, y, z)) \end{aligned} \quad (3)$$

Barthe et al¹⁸ introduced the notion of implicit extrusion fields. The function G is considered as a zero iso-potential curve $G(X, Y) = 0$ defined in a two-dimensional “implicit space” \mathbb{I}^2 . A two-dimensional “implicit space” can be seen as a curvilinear space in which each coordinate is a potential field (see Section 2.3 and Equation 4).

$$\begin{aligned} G: \mathbb{I}^2 &\rightarrow \mathbb{I} \\ (X, Y) &\rightarrow Z = G(X, Y) \\ \text{or } (f_1, f_2) &\rightarrow f_3 = G(f_1, f_2) \end{aligned}$$

with $f_i: \mathbb{R}^3 \rightarrow \mathbb{R} \quad i = 1..3$

$$(x, y, z) \rightarrow f_i(x, y, z) \quad (4)$$

To clarify notations, points of an “implicit space” are denoted with capital letters and points of the Euclidean space

are denoted with small letters. A point $P(X = 3, Y = 2)$ (or $P(f_1 = 3, f_2 = 2)$) in a space \mathbb{I}^2 is then represented in the Euclidean space \mathbb{E}^3 by the intersection between the 3 iso-potential surface of the field defined by f_1 (which is the set of points p of \mathbb{E}^3 for which $f_1(p) = 3$) and the 2 iso-potential surface of the field defined by f_2 (which is the set of points p of \mathbb{E}^3 for which $f_2(p) = 2$). This is the intersection between two surfaces which is generally a curve. A point P of \mathbb{I}^2 is represented by a curve in \mathbb{E}^3 . A curve (or profile) can be seen as a continuous succession of points. The profile of \mathbb{I}^2 is then represented by a continuous succession of curves in \mathbb{E}^3 (representing each point of the profile), which gives us a surface. The profile $G(X, Y) = 0$ defined in \mathbb{I}^2 is said to be extruded in \mathbb{E}^3 along the intersections of f_1 and f_2 iso-potential surfaces (Figure 1). The surface defined by the extrusion of profile $G = 0$ is the result of combining f_1 and f_2 .

Full details are given in¹⁸. The authors explain how points and vectors defining the profile in \mathbb{I}^2 can be directly selected by the user in Euclidean space \mathbb{E}^3 , allowing accurate and intuitive control of the profile, and, therefore, of the resulting object. The profile $G = 0$ is defined in¹⁸ by a function $H: \mathbb{R} \rightarrow \mathbb{R}$ such as $Y = H(X)$ and $G = Y - H(X)$. This greatly limits the freedom given to the user, and moreover removes a part of the intuitive process. It also obliges the authors to propose specific operators for the union, intersection and difference with a “functionally-defined” transition, which is a smooth transition defined point-by-point from the Euclidean space \mathbb{E}^3 with a single valued function $H: \mathbb{R} \rightarrow \mathbb{R}$. Functions H are defined with one-dimensional cubic polynomial splines³³ to interpolate the control points. Equation 5 shows the operator used for the union operator.

$$G(f_1, f_2) = \min(f_1, f_2) - H(|f_1 - f_2|) \quad (5)$$

This union operator (first proposed by Dekkers et al³²) is built with a \min function which requires the C^1 continuity to be explicitly controlled where $f_1 = f_2$. Our conclusion is that this model represents a very interesting theoretical base and more research has to be done on profile definition to exploit the properties of profile extrusion and point-by-point control from the Euclidean modeling space in a more powerful and intuitive volume modeling tool. To build a curve $G(X, Y) = 0$ like the one shown in Figure 1, free-form implicit curves are needed (the vertical $X = 0$ can not be defined with a single valued function). This leads us to the study of implicit free-form profiles controlled point-by-point.

2.3. Definition of an implicit space

An implicit space \mathbb{I}^n is a space where each coordinate is a potential field. Its basis is defined by a set of n linearly independent potential functions $f_i: \mathbb{E}^m \rightarrow \mathbb{E}$ and a point $P(X_0, \dots, X_{n-1})$ is expressed as $P(f_0 = X_0, \dots, f_{n-1} = X_{n-1})$. In this paper $m = 3$, hence each coordinate $f_i = X_i$ ($i = 1..n - 1$) of a point P is an iso-surface in \mathbb{E}^3 and the repre-

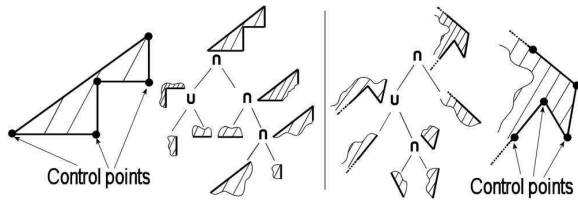


Figure 2: Example of binary composition trees to represent a polygon or an open curve by a real function.

sentation of the point P itself is defined by their intersection. More details are given in [18](#).

3. Implicit free-form curves

In this section we present our free-form implicit curves. We propose the use of homogeneous control parameters to provide an intuitive solution to the user and for this reason we use free-form implicit curves controlled point-by-point as a means of providing accurate and intuitive operators on volumes. The composition of two primitives gives a new object which can be used as a primitive in a new composition. Because the smoothness and the control of the form of the transition is highly dependent on the variational properties of the primitives' fields, potential fields used to combine them must preserve these properties as faithfully as possible. This is why particular attention has to be focused on the regularity of the field variations produced by our two-dimensional potential fields G .

3.1. Natural solutions

It appears natural to try to use methods like the implicitization of parametric free-form curves^{34,35} or the projection of the z value of a surface defined by an equation $z = f(x, y)$ to define the implicit free-form curve. But these methods provide bounded functions while we need to be able to produce infinite open curves with the control of their limits. The first solution provides complicated equations and does not ensure regular and homogeneous potential field variations, and the second requires the user to design the entire surface to create the profile and its variations, while they should be ideally concentrating on the form of the curve. For these reasons we use a different approach.

3.2. Free-form curves built with composition of lines

As shown by Pasko et al³⁶, it is possible to represent polygons with straight and curved edges by real functions. The polygon is decomposed in a binary composition tree where the leaves are lines and the nodes are union or intersection R-function operators (see Figure 2). This method also avoids internal and unwanted zeroes. We choose it as a starting point to create our profiles. Indeed, it can easily be adapted

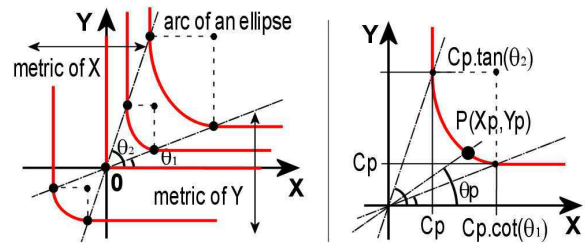


Figure 3: In red, on the left, representation of the two-dimensional field iso-potential curves defining our G^1 continuous union operator $\widehat{G}_U(X, Y)$. On the right, graph of the C_p iso-potential curve and important values for computing the equation of the arc of an ellipse.

to open profiles, and because lines are combined, the extremities are perfectly controlled: they are simply half lines. To provide a regular base for the field variations, we use lines defined by the zero iso-potential of a linear potential function $l(X, Y)$. The linear potential function l splits the space into two half spaces: one where $l(X, Y) < 0$ (inside) and one where $l(X, Y) > 0$ (outside). Function l has the one iso-potential line at a distance of one from the zero iso-potential (the one iso-potential line is in the outside half space defined by l , and the minus one iso-potential line in the inside half space). Because control points are used, line equations are defined with pairs of points. Even if R-function Boolean composition operators without smooth transition¹⁰ already provide a potential field that is C^1 continuous when both arguments are not equal to zero (they are both equal to zero at the junction between two zero iso-potential lines), they have a global impact on the line potential fields. This is a fundamental property because the variation of the combined primitives directly depends on the variation of the field of the composition operator and we want to provide volume objects with regular and smooth C^1 or at least G^1 continuous potential fields.

3.3. Composition with better field variations

To ensure more faithful conservation of the field variations, we propose a new operator that modifies line potentials only around their junction. Indeed, the line metric reproduces the metric of X and Y (as shown in Figure 3), i.e. the metric the potential functions f_1 and f_2 once used as a composition operator. For convenience, we propose the following terminology: $\widehat{G}_U, \widehat{G}_\cap, \widehat{G}_\setminus$ represent respectively our G^1 continuous Boolean union, intersection and difference operators. Operator $\widehat{G}_U(X, Y)$ (see Figure 3 and Equation 6) is defined by the fields of which it is composed outside a region bounded by two angles θ_1 and θ_2 , and by an arc of an ellipse inside it. If the angles are close to one another, the field is sharp at the transition level and if θ_1 is close to zero and θ_2 close to $\pi/2$, the field is highly smoothed. We suggest the use of

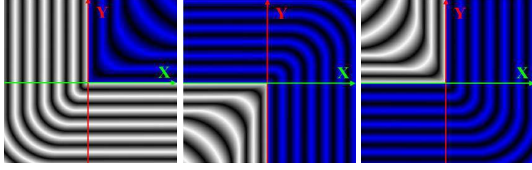


Figure 4: From left to right, graphs of our operators \widehat{G}_U , \widehat{G}_n and \widehat{G}_v in the space \mathbb{I}^2 . $G_{op} \geq 0$ in the blue area, and $G_{op} < 0$ in the white one.

$\theta_1 = \pi/8$ and $\theta_2 = 3\pi/8$, which gives a good average between a smooth field and the conservation of X and Y metrics.

$$\theta_1 \in]0, \pi/4[, \quad \theta_2 \in]\pi/4, \pi/2[$$

At a point $P(X, Y)$: $\theta = \text{angle}([OX], [OP])$. We distinguish four distinct areas:

- $P \in A_1$ if $\theta \in [\theta_2 - \pi, \theta_1]$
- $P \in A_2$ if $\theta \in [\theta_2, \theta_1 + \pi]$
- $P \in A_3$ if $\theta \in]\theta_1, \theta_2[$
- $P \in A_4$ if $\theta \in]\theta_1 + \pi, \theta_2 + \pi[$

$$\widehat{G}_U(X, Y) = \begin{cases} 0 & \text{if } X = Y = 0 \\ Y & \text{if } P \in A_1 \\ X & \text{if } P \in A_2 \\ C & \text{where } C \text{ is the solution of:} \\ & \frac{(C \cdot \cot(\theta_1) - X)^2}{(C \cdot \cot(\theta_1) - C)^2} + \frac{(C \cdot \tan(\theta_2) - Y)^2}{(C \cdot \tan(\theta_2) - C)^2} = 1 \\ & \text{if } P \in A_3 \\ C & \text{where } C \text{ is the solution of:} \\ & \frac{(X - C \cdot \cot(\theta_2))^2}{(C - C \cdot \cot(\theta_2))^2} + \frac{(Y - C \cdot \tan(\theta_1))^2}{(C - C \cdot \tan(\theta_1))^2} = 1 \\ & \text{if } P \in A_4 \end{cases} \quad (6)$$

Equation 6 appears, at first glance, to be difficult to solve, but it can be greatly optimized, and most of the terms can be pre-computed. The closed form solution for the evaluation of C is given in Appendix B. Operators \widehat{G}_n and \widehat{G}_v are built following the same construction and the same kind of equations are produced (their equations are given in Appendix A). These operators are real-valued binary CSG (Constructive Solid Geometry) operators, and their algebraic properties have been presented by Pasko et al¹⁰. However, we note that they are not symmetric ($\widehat{G}_{op}(X, Y) \neq \widehat{G}_{op}(Y, X)$) if $\theta_2 \neq \frac{\pi}{2} - \theta_1$. With these operators, or with R-functions, free-form profiles, G , can be created. They are not smooth curves, but a succession of line segments, beginning and ending with a half line if they are open. To obtain smooth curves, we use the Boolean operators proposed in ¹⁸. These operators provide a point-by-point control of a “functionally-defined” transition and, because they are derived from functions of $\mathbb{R} \rightarrow \mathbb{R}$, they provide regular and smooth field variations. The first point, the middle point and the last point of the

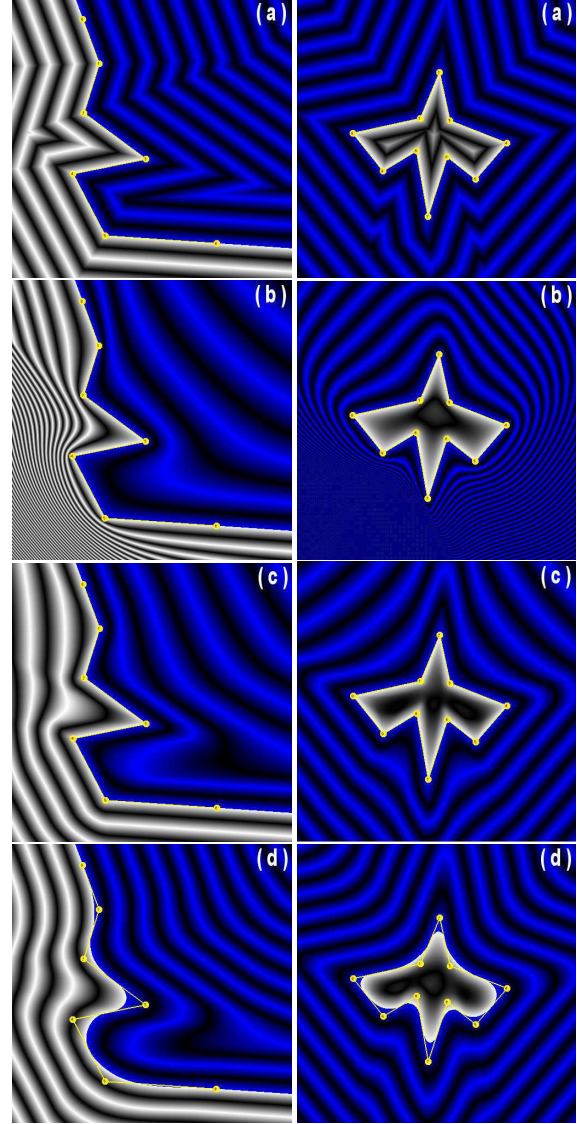


Figure 5: Open and closed implicit free-form two-dimensional potential fields G , shown in \mathbb{E}^2 , built with different operators: (a) Ricci's min and max operators, (b) R-functions, (c) \widehat{G}_U and \widehat{G}_n , and (d) Barthe et al operators with smooth transitions. The white contours correspond to $G \leq 0$ and the blue ones to $G > 0$. The control points and the corresponding lines are drawn in yellow.

| | min | max | R-functions | \widehat{G} | Barthe et al |
|--------|-------|-------|-------------|---------------|--------------|
| open | 106 | 148 | 457 | 135 | |
| closed | 178 | 254 | 802 | 227 | |

Table 1: Time in milliseconds to compute potential function values for the pictures shown in Figure 5. Picture size is 512^2 pixels, which corresponds to 262144 evaluations. The open curves are built with six line segments and the closed curves are built with ten segments.

transition are then accurately controlled, which allows us to replace the sharp transition by a simple smooth transition automatically joining the middle of each segment. An exception is made for open curves where the beginning half-line is composed from the first control point and the last one is composed to the last point. The middle point is used to select the smoothness of the transition if necessary, but we recommend fixing it at a constant value to generate a smooth transition automatically and avoid the manipulation of an additional parameter.

3.4. Results and discussion

Figure 4 illustrates operators \widehat{G}_U , \widehat{G}_\cap and \widehat{G}_\setminus individually and Figure 5 shows the difference of field variations obtained using different composition operators in both an open and a closed profile G . Ricci's min and max Boolean composition operators⁵ leave the metric of the combined primitives unchanged (Figure 5(a)). This is why the potential field computed with the evaluation of the composition tree defining the profile G with these operators gives a valid reference to evaluate the variations of the metric once the operators applied to compute the profile. Figure 5(b) shows how potential fields obtained using Pasko's R-function operators¹⁰ are degraded in some areas (bottom left corner for the open profile figure and bottom for the closed profile figure). As we see, our operators \widehat{G}_U and \widehat{G}_\cap consequently increase the fidelity of the field variation for "segment profiles" (Figure 5(c)), and Barthe et al operators allow us to produce a smooth free-form implicit curve with regular and quite homogeneous variations in its potential field (Figure 5(d)). For these reasons, our profiles satisfy the essential properties to define combination operators on volume objects in efficient and controllable modeling tools. Computing times are given in Table 1 to compare the cost of the different operators used to create free-form profiles. We can note that the use of our new operators \widehat{G} will increase the evaluation time of the curve by an average factor $\simeq 3.1$, while our smooth curves require less computations than the sharp ones produced with the R-function composition operators.

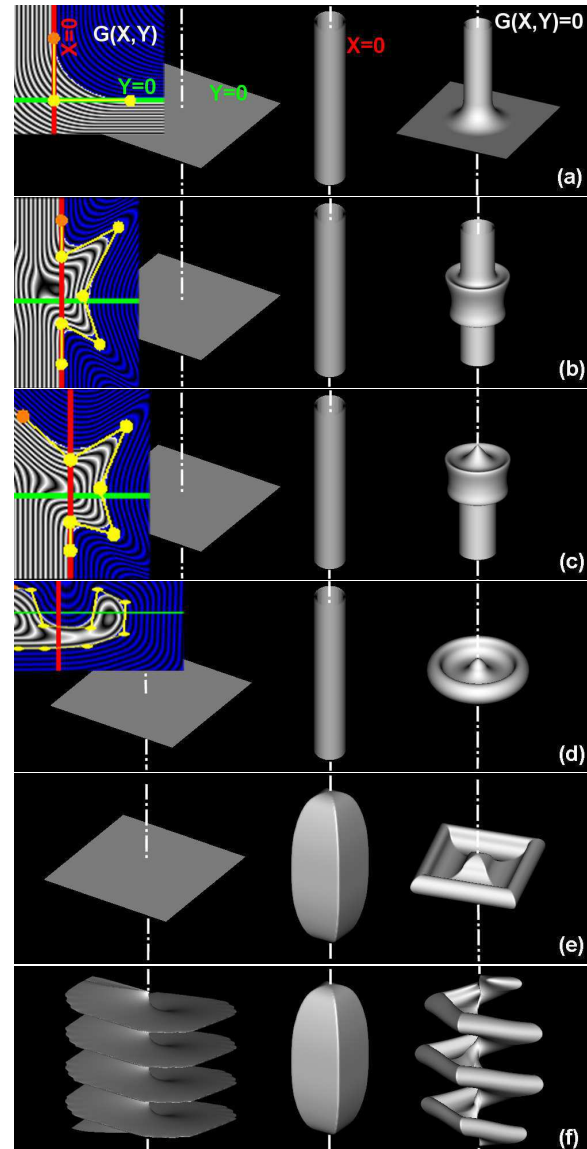


Figure 6: Figures (a) to (d) show different profiles extruded in an implicit field \mathbb{I}^2 defined by a cylindrical potential field (for the abscissa $X \equiv f_1$, in the central column) and a plane field (for the ordinate $Y \equiv f_2$, in the left column). Profiles are shown in the top left corner. They are extruded around the cylinder, following the horizontal direction given by the plane. The final object produced by the extrusion of the profile is shown in the right column. In Figures (e) and (f), the profile is the one used in Figure (d). Figure (e) illustrates the modification in the extrusion when the cylinder is replaced by a closed parallelepiped object, and Figure (f) illustrates the modification obtained when the plane is replaced by a screw-like object. The result in (f) is that the profile is extruded around the parallelepiped, following the iso-surfaces of the screw.

4. Extrusion of free-form curves

In this section we show how our free-form profiles G allow us to exploit the interpretation of “implicit space” to greatly increase freedom, accuracy and intuitive modeling in the manipulation of volumes. In fact, our method allows us to combine composition, sculpting and modeling tools in a single operator.

4.1. Extrusion properties

Following Barthe et al¹⁸, a two-dimensional function $G(X, Y)$ can be defined in an “implicit space” \mathbb{I}^2 to generate a volume object. The “implicit space” is defined by two potential functions of $\mathbb{E}^3 \rightarrow \mathbb{E}$, f_1 and f_2 . As shown in Figure 1 and explained in Section 2, the profile is said to be extruded along the intersections of f_1 and f_2 iso-potential surfaces. In effect, the choice of potential functions f_1 and f_2 define the trajectories of extrusion while the profile defines the free-form implicit curve which is going to be extruded along these trajectories (Figure 6).

The properties that a profile G must respect in order to provide the union Boolean composition operator with smooth transitions are well known⁸, and “extrusion” properties of the profile are discussed in ¹⁸. Briefly, when the profile follows the X axis (which corresponds to the line $Y = 0$), its representation in \mathbb{E}^3 follows the *zero* iso-potential surface of the field defined by f_2 and when it follows the Y axis (which corresponds to the line $X = 0$), its representation in \mathbb{E}^3 follows the *zero* iso-potential surface of the field defined by f_1 (see Figure 1). These properties allow us to integrate the *zero* iso-potential surfaces of primitives defined by f_1 and f_2 , and to realize the Boolean composition operators with smooth transitions. Furthermore, as shown in Figure 7, in addition to the classical smooth transitions, the profile can be used to sculpt the primitives and to combine them with free-form transitions.

4.2. Example of modeling interface

The link between the profile and the shape of the resulting object becomes less intuitive when profiles are complicated. However, profile control points can be directly selected from the Euclidean space \mathbb{E}^3 . Our volume objects are built and visualized using the modeling method proposed in ¹⁹. We briefly summarize this approach to illustrate the controllability of our operators. Volumes are stored in a regular grid and visualized with a ray-casting rendering using a triquadratic reconstruction³⁸. A plane section of the potential values is extracted from the grid and visualized as a picture in a new window (Figure 8 (a) and (b)). In this window, the user can interactively select the profile without the abstraction of the form of the iso-potential surfaces of potential fields f_1 and f_2 (Figure 8 (c) and (d)). This shows that being able to select the control points, and thus the profile directly in the Euclidean space \mathbb{E}^3 , restores the accuracy lost by the definition of the

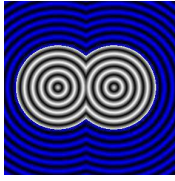
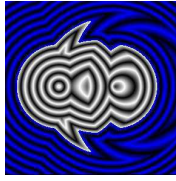
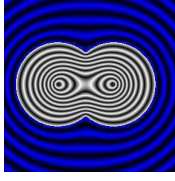
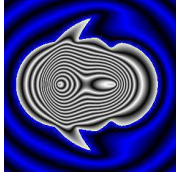
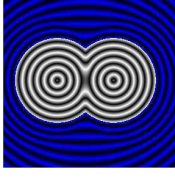

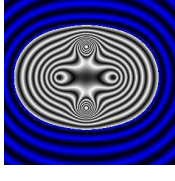
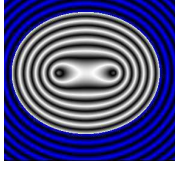
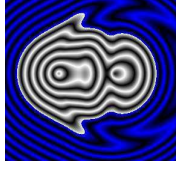
| time | 2D section | time | 2D section |
|------|--|----------|---|
| 263 |  | 1025 |  |
| 326 |  | 1448 |  |
| 1057 |  | 5238 |  |
| 369 |  | not used | |
| 322 |  | 1363 |  |

Table 2: Time in milliseconds to compute potential function values for a 128^3 grid (2097152 evaluations). In the first column: (1st row) the Ricci's min max operators, (2nd row) the sharp R-functions operators, (3rd row) our operators \hat{G} , (4th row) the blending R-functions operators and (5th row) our free-form curve operator. Time and potential field variations (in a two-dimensional plane section) are shown : On the left for the classical sharp/blending operators and on the right, for a free-form operator created with seven segments.

submitted to COMPUTER GRAPHICS Forum (1/2003).

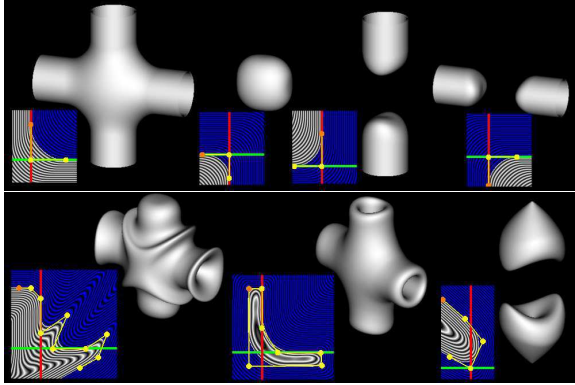


Figure 7: Different free-form profile extrusions in a “implicit space” defined by two orthogonal cylinders. The top row shows classical union, intersection and the two difference operators with smooth transitions. The following row illustrates more advanced possibilities offered by our free-form profile extrusion. In the profile pictures, the red line represents $X = 0$ and the green line $Y = 0$.

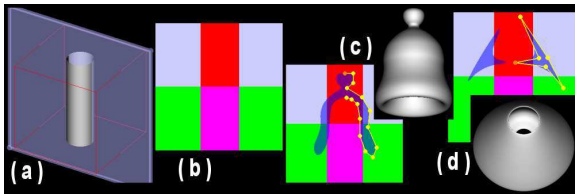


Figure 8: (a) Three dimensional visualisation of volume objects and selection of the plane section in transparent blue. (b) Visualisation of the plane section. Functions f_1 and f_2 are respectively: a vertical cylinder and an horizontal plane. Colors are used to easily identify different regions: red if $f_1 \leq 0$ and $f_2 \geq 0$, green if $f_1 \geq 0$ and $f_2 \leq 0$, purple if $f_1 < 0$ and $f_2 < 0$, bright gray if $f_1 > 0$ and $f_2 > 0$. (c) The section of the resulting object is visualized in blue with a transparency effect. We show an example of an open profile and its result. (d) Another example with a closed profile and its result.

profile in an “implicit space”. Moreover, to exactly follow the zero iso-potential surface of potential function f_1 or f_2 , the abscissa or the ordinate of a profile control points can be explicitly fixed to zero. This is important when profiles are used to combine primitives.

4.3. Results and discussion

With our approach, a wide variety of shapes and transitions can be produced, including all the ones that Barthe et al’s models¹⁸ could generate (because free-form profiles generalize “functionally defined” ones). Furthermore, instead of disparate notions of composition operators, sculpture oper-

ator or primitive creation tool, we present a unified modeling metaphor which is simply: choose the adequate potential functions f_1 and f_2 and create the profile G to generate the new object. Once this process is well understood, our method is intuitive, and it provides a lot of freedom for the user to build different “implicit spaces” and extrude profiles in them (Figure 9). In addition, profiles can be directly defined and manipulated from the user modeling space \mathbb{E}^3 . This makes our model relevant for interactive volume modeling. The limitations are essentially the complexity of the shape and the irregularity of the variations in the potential fields function f_1 and f_2 . Table 2 illustrates the simple example of the composition of two spheres, the differences of computing time and the variations in the object potential field. Times correspond to the computation of the composition operator itself. In our application (briefly presented in the previous section), grids are used to store the potential fields after each operation. The given times correspond to the computation of a new 128^3 voxels grid from the two grids representing the composed objects. Fast volume rendering methods based on ray-casting or hardware rendering can then perform the visualisation in about 1 second, and a basic marching-cube algorithm will extract the polygons from the grid in 5–10 seconds, depending on the object complexity (timings taken on an 866 MHz AMD Athlon processor). If the object has to be visualized from its equation, these times illustrate the increasing of the computation complexity. The objects shown in Figure 9 have been built with a few primitives, and the average time to design one of them is about an hour and a half. Note that the twisted corkscrew’s handle is generated by a profile swept and twisted along a line segment.

5. Conclusion and future work

Accurate and intuitive manipulation of implicit objects is the next required step to provide efficient implicit modeling software. In this paper we have presented some solutions to these requirements when implicit primitives are composed. Recent work introduced the notion of “implicit spaces” as a theoretical base to accurately control the transitions¹⁸ and we have shown how this leads us to the study of free-form implicit curves controlled point-by-point. Profiles require regular variations of their two-dimensional potential field and the control of their extremities. The adaptation of a method proposed by Pasko et al³⁶ has allowed us to provide open and closed profiles, defining line segments or smooth curves, with sufficient properties.

To manipulate implicit volumes defined by the inequality $f(p) \leq 0$, we have grouped in a single modeling tool: creation, sculpture and Boolean composition operators with sharp or smooth free-form transitions. We have shown how accuracy is ensured and explained how to understand and use our modeling metaphor in an efficient way. The techniques described in this paper generalize the models given in ¹⁸

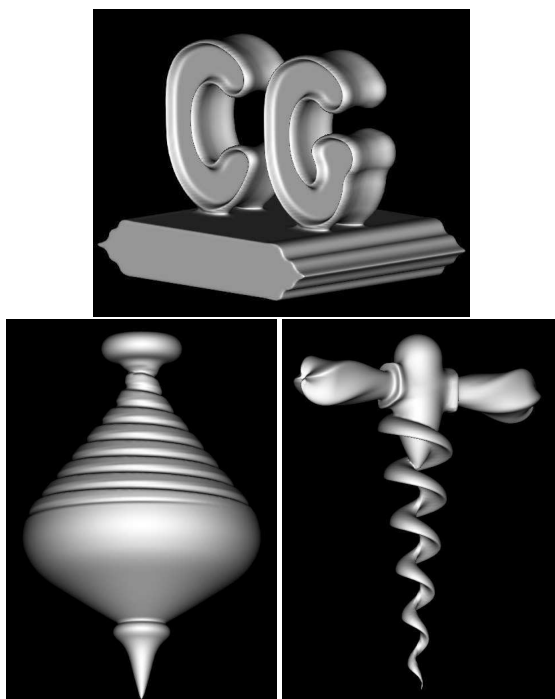


Figure 9: Examples of volume objects illustrating composition, sculpting and primitive creation using operators based on two-dimensional free-form potential fields.

and provide controllability and freedom of expression to the user and greatly extend the possibilities offered by operators on volumetric objects. The point-by-point control curve provides the possibility of interactively designing blended shapes, however more work needs to be done on volume data structure manipulation and modeling interface design to add interactivity to the modeling process. Adaptive structures like ADF³ and interactive ray casting algorithm for iso-surface visualisation¹ can be used to, respectively, store a sampled potential field and accurately render the surface. A sampled field structure stores the potential values after each operation, removing the expensive evaluation of an increasingly complicated potential field function. This technique allows us to accurately render the modeled object directly without the use of an additional data structure, such as a polygon mesh. These reasons provide a good justification for the investigation of the combination of these techniques, to provide interactive implicit modeling solutions.

Because sampled three-dimensional potential functions are large data structures, the development of multiresolution techniques to store and reconstruct potential fields represents another important step.

Free-form profiles can also be used in implicit sweep objects^{20, 37}. Indeed, only profiles defined by single valued functions of $\mathbb{R} \rightarrow \mathbb{R}$ are used in implicit modeling, while

there is no apparent reason to limit the user by the form of the profile they want to sweep. Our profiles should allow us to create swept objects defined by an inequality $f(p) \leq 0$. Since they have regular and homogeneous field variations, sweep objects can be correctly combined with other implicit objects.

6. Acknowledgments

This work was partially funded by E.U. contract HPRN-CT-1999-00117

References

1. M. Chen, A. Kaufman and R. Yagel. *Volume Graphics*, Springer, London, 2000. 1, 9
2. K. Engel, M. Krauss and T. Ertl. High-quality Preintegrated Volume rendering Using Hardware-accelerated Pixel Shading. *Proc. of EUROGRAPHICS/SIGGRAPH Graphics Hardware Workshop*, 2001. 1
3. S.F. Frisken, R.N. Perry, A.P. Rockwood and T.R. Jones. Adaptively Sampled Distance Fields: A General representation of Shape for Computer Graphics. *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pp. 249–254, 2000. 1, 9
4. L.P. Kobbelt, M. Botsch, U. Schwanecke and H.P. Seidel. Feature Sensitive surface Extraction from Volume data. 1
Proceedings of SIGGRAPH 2001, Computer Graphics Proceedings, Annual Conference Series, pp. 57–66, 2001.
5. A. Ricci. A Constructive Geometry for Computer Graphics. *The Computer Journal*, **16** (2), pp. 157–160, 1973. 1, 2, 6
6. J. F. Blinn. A Generalization of algebraic surface drawing. *ACM Transaction on Graphics*, **1** (3), pp. 235–256, 1982. 1
7. J. Bloomenthal and B. Wyvill. Interactive techniques for Implicit Modeling. *Computer Graphics (Proc. of SIGGRAPH 1990)*, **24** (2), pp. 109–116, 1990. 1, 2
8. A. P. Rockwood. The Displacement Method for Implicit Blending Surfaces in Solid Models. *ACM Transaction on Graphics*, **8** (4), pp. 279–297, 1989. 1, 3, 7
9. A. Sourin and A. Pasko. Function Representation for Sweeping by a Moving Solid. *IEEE Transaction on Visualization and Computer Graphics*, **2** (1), pp. 11–18, 1996. 1
10. A. Pasko, V. Adzhiev, A. Sourin and V. Savchenko.

- Function Representation in Geometric Modeling: Concepts, Implementation and Applications. *The Visual Computer*, **8** (2), pp. 429–446, 1995. 1, 2, 4, 5, 6
11. B. Wyvill and A. Guy and E. Galin. Extending the CSG Tree: Warping, Blending and Boolean Operations in an Implicit Surface Modeling System. *Computer Graphics Forum*, **18** (2), pp. 149–158, 1999. 1, 2
 12. M. Chen and J. Tucker. Constructive Volume Geometry. *Computer Graphics Forum*, **19** (4), pp. 281–293, 2000. 1
 13. H. Nishimura, M. Hirai, T. Kawai, I. Shirakara and K. Omura. Object Modeling by Distribution Functions. *Electronics Communications (in Japanese)*, **J68** (D), pp. 718–725, 1985. 1, 2
 14. G. Wyvill, C. McPheeters and B. Wyvill. Data Structure for Soft Objects. *The Visual Computer*, **2** (4), pp. 227–234, 1986. 1, 2
 15. C. Galbraith, P. Prusinkiewicz and B. Wyvill. Modeling Murex Cabritii Sea Shell with a Structured Implicit Modeler. *Proc. of Computer Graphics International 2000*, 2000. 1
 16. C. Hoffmann and J. Hopcroft. Automatic Surface Generation in Computer Aided Design. *The Visual Computer*, **1**, pp. 92–100, 1985. 1, 3
 17. V. Adzhiev, M. Kazakov and A. Pasko. Hybrid System Architecture for Volume Modeling. *Computer and Graphics*, **24** (1), pp. 67–78, 2000. 1, 2
 18. L. Barthe, V. Gaildrat and R. Caubet. Extrusion of 1D implicit profiles: Theory and first application. *International Journal of Shape Modeling*, **7** (2), pp. 179–199, 2001. 1, 2, 3, 4, 5, 7, 8
 19. L. Barthe, B. Mora, N. Dodgson and M. Sabin. Triquadratic reconstruction for interactive modelling of potential fields. *Proc. of Shape Modeling International 2002*, pp. 145–153, 2002. 2, 7
 20. B. Crespin, C. Blanc and C. Schlick. Implicit Sweep Objects. *EUROGRAPHICS 1996*, **15** (3), pp. 165–174, 1996. 2, 9
 21. Z. Kacic-Alesic and B. Wyvill. Controlled Blending of Procedural Implicit Surfaces. *Proc. of Graphic Interface 1991*, pp. 236–245, 1991. 2
 22. Ca. Blanc and C. Schlick. Extended Field Functions for Soft Objects. *Proc. of Implicit Surfaces 1995*, pp. 21–32, 1995. 2
 23. M. Desbrun and M.P. Gascuel. Animating Soft Substances with Implicit Surfaces. *Computer Graphics (Proc. of SIGGRAPH 1995)*, pp. 287–290, 1995. 2
 24. A. Guy and B. Wyvill. Controlled Blending for Implicit Surfaces Using a Graph. *Proc. of Implicit Surfaces 1995*, pp. 107–112, 1995. 2
 25. E. Ferley and M. P. Cani and J. D. Gascuel. Practical Volumetric Sculpting. *The Visual Computer*, **16** (8), pp. 469–480, 2000. 2
 26. E. Galin and S. Akkouche. Incremental Polygonization of Implicit Surfaces. *Graphical Models*, **62** (1), pp. 19–39, 2000. 2
 27. B.A. Payne and A.W. Toga. Distance Field Manipulation of Surface Models. *Computer Graphics and Applications*, **12** (1), pp. 65–71, 1992. 2
 28. M. Jones and R. Satherley. Shape Representation Using Space Filled Sub-voxel Distance Fields. *Proc. of Shape Modeling International 2001*, pp. 316–325, 2001. 2
 29. J.C. Carr, R.K. Beaton, J.B. Cherrie, T.J. Mitchell, W.R. Fright, B.C. McCallum and T.R. Evans. Reconstruction and Representation of 3D Objects with Radial Basis Functions. *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pp. 67–76, 2001. 2
 30. A. E. Middleditch and K. H. Sears. Blend Surfaces for Set-theoretic Volume Modelling Systems. *Computer Graphics*, **19** (3), pp. 161–170, 1985. 3
 31. G. Pasko, A. Pasko, M Ikeda and T. Kunii. Bounded Blending Operations. *Proc. of Shape Modeling International 2002*, pp. 95–103, 2002. 3
 32. D. Dekkers, K. van Overveld and R. Golsteijn. Combining CSG Modeling with Soft Blending using Lipschitz-based Implicit Surfaces. *Technical Report, Eindhoven University of Technology, Computer Graphics Group*, 1997. 3
 33. I.D. Faux and M.J. Pratt. *Computational Geometry for Design and Manufacture*, Ellis Horwood, 1979. 3
 34. R.N. Goldman, T.W. Sederberg and D.C. Anderson. Vector Elimination: A Technique for the Implicitization, Inversion, and Intersection of Planar Parametric Rational Polynomial Curves. *Computer Aided Geometric Design*, **1** (4), pp. 327–356, 1984. 4
 35. C.M. Hoffmann. Implicit Curves and Surfaces in CAGD. *IEEE Computer Graphics and Applications*, **13** (1), pp. 79–88, 1993. 4
 36. A.A. Pasko, A.V. Savchenko and V.V. Savchenko. Implicit Curved Polygons. *Technical Report 96-1-004, University of Aizu, Japon*, 1996. 4, 8
 37. C. Grimm. Implicit Generalized Cylinders Using Profile Curves. *Proc. of Implicit Surfaces 1999*, pp. 33–41, 1999. 9
 38. B. Mora, J.P. Jessel and R. Caubet. Visualization of Isosurfaces with Parametric Cubes. *EUROGRAPHICS 2001 Proc.*, **20** (3), pp. 377–384, 2001. 7

Appendix A: Operators \widehat{G}_\cap and \widehat{G}_\setminus

- Operator \widehat{G}_\cap :

$$\theta_1 \in]0, \pi/4[, \quad \theta_2 \in]\pi/4, \pi/2[.$$

At a point $P(X, Y)$: $\theta = \text{angle}([OX], [OP])$

We distinguish four distinct areas:

$$\begin{aligned} P \in A_1 & \text{ if } \theta \in [\theta_2 - \pi, \theta_1] \\ P \in A_2 & \text{ if } \theta \in [\theta_2, \theta_1 + \pi] \\ P \in A_3 & \text{ if } \theta \in]\theta_1, \theta_2[\\ P \in A_4 & \text{ if } \theta \in]\theta_1 + \pi, \theta_2 + \pi[\end{aligned}$$

$$\widehat{G}_\cap(X, Y) = \begin{cases} 0 & \text{if } X = Y = 0 \\ X & \text{if } P \in A_1 \\ Y & \text{if } P \in A_2 \\ C & \text{where } C \text{ is the solution of:} \\ & \frac{(X - C \cdot \cot(\theta_2))^2}{(C - C \cdot \cot(\theta_2))^2} + \frac{(Y - C \cdot \tan(\theta_1))^2}{(C - C \cdot \tan(\theta_1))^2} = 1 \quad (7) \\ & \text{if } P \in A_3 \\ C & \text{where } C \text{ is the solution of:} \\ & \frac{(C \cdot \cot(\theta_1) - X)^2}{(C \cdot \cot(\theta_1) - C)^2} + \frac{(C \cdot \tan(\theta_2) - Y)^2}{(C \cdot \tan(\theta_2) - C)^2} = 1 \\ & \text{if } P \in A_4 \end{cases}$$

- Operator \widehat{G}_\setminus :

Operator \widehat{G}_\setminus is directly obtained from operator \widehat{G}_\cap using the following expression:

$$\widehat{G}_\setminus(X, Y) = \widehat{G}_\cap(X, -Y) \quad (8)$$

Appendix B: Closed form solution for the evaluation of C_P in our new composition operators

- Solution for the equation:

$$\frac{(C \cdot \cot(\theta_1) - X)^2}{(C \cdot \cot(\theta_1) - C)^2} + \frac{(C \cdot \tan(\theta_2) - Y)^2}{(C \cdot \tan(\theta_2) - C)^2} = 1 \quad (9)$$

C is the greater solution of the following equation:

$$a \cdot C^2 + b \cdot C + c = 0 \quad (10)$$

with

$$\begin{aligned} a &= \frac{(\tan(\theta_2) - 1)^2}{\tan^2(\theta_1)} + \tan^2(\theta_2) \cdot \left(\frac{1}{\tan(\theta_1)} - 1 \right)^2 \\ &\quad - \left(\frac{1}{\tan(\theta_1)} - 1 \right)^2 \cdot (\tan(\theta_2) - 1)^2 \\ b &= -2 \cdot \left(X \cdot \frac{(\tan(\theta_2) - 1)^2}{\tan(\theta_1)} + Y \cdot \tan(\theta_2) \cdot \left(\frac{1}{\tan(\theta_1)} - 1 \right)^2 \right) \\ c &= X^2 \cdot (\tan(\theta_2) - 1)^2 + Y^2 \cdot \left(\frac{1}{\tan(\theta_1)} - 1 \right)^2 \end{aligned} \quad (11)$$

- Solution for the equation:

$$\frac{(X - C \cdot \cot(\theta_2))^2}{(C - C \cdot \cot(\theta_2))^2} + \frac{(Y - C \cdot \tan(\theta_1))^2}{(C - C \cdot \tan(\theta_1))^2} = 1 \quad (12)$$

C is the lower solution of the following equation:

$$a \cdot C^2 + b \cdot C + c = 0 \quad (13)$$

with

$$\begin{aligned} a &= \frac{(\tan(\theta_1) - 1)^2}{\tan^2(\theta_2)} + \tan^2(\theta_1) \cdot \left(\frac{1}{\tan(\theta_2)} - 1 \right)^2 \\ &\quad - \left(\frac{1}{\tan(\theta_2)} - 1 \right)^2 \cdot (\tan(\theta_1) - 1)^2 \\ b &= -2 \cdot \left(X \cdot \frac{(\tan(\theta_1) - 1)^2}{\tan(\theta_2)} + Y \cdot \tan(\theta_1) \cdot \left(\frac{1}{\tan(\theta_2)} - 1 \right)^2 \right) \\ c &= X^2 \cdot (\tan(\theta_1) - 1)^2 + Y^2 \cdot \left(\frac{1}{\tan(\theta_2)} - 1 \right)^2 \end{aligned} \quad (14)$$

- All the terms in θ_1 and θ_2 can be precomputed (once θ_1 and θ_2 are selected), which greatly decreases the cost of the evaluation of C . For instance, a can be totally pre-computed.

7.3.2 Controllable Binary CSG Operators for “Soft Objects”.

Auteurs : Loïc Barthe, Brian Wyvill, Erwin de Groot
Revue : International Journal of Shape Modeling, Vol 10(2)
Date : Decembre 2004

International Journal of Shape Modeling
© World Scientific Publishing Company

CONTROLLABLE BINARY CSG OPERATORS FOR “SOFT OBJECTS”

LOÏC BARTHE

*SIRV Group, IRIT/UPS Toulouse
118 route de Narbonne, 31062 Toulouse Cedex 4, France
lbarthe@irit.fr
<http://www.irit.fr/Loic.Barthe/>*

BRIAN WYVILL

ERWIN DE GROOT

*Dept. of Computer Science, University of Calgary
2500 University Dr. NW Calgary, Alberta, T2N 1N4, Canada
blob@cpsc.ucalgary.ca
erwin@erwindegroot.nl
<http://pages.cpsc.ucalgary.ca/blob/>*

Received (Day Month Year)

Revised (Day Month Year)

Accepted (Day Month Year)

Communicated by (xxxxxxxxxx)

Potential functions allow the definition of both an implicit surface and its volume. In this representation, two categories can be distinguished: bounded and unbounded representations. Boolean composition operators are standard modelling tools allowing complex objects to be built by the combination of simple volume primitives. Though they are well defined for the second category, there is no clear definition of the properties that such operators should satisfy in order to provide bounded representation with both smooth and sharp transition. In this paper, we focus on bounded implicit representation. We first present fundamental properties to create adequate composition operators. From this theoretical framework, we derive a set of Boolean operators providing union, intersection and difference with or without smooth transition. Our new operators integrate accurate point-by-point control of smooth transitions and they generate G^1 continuous potential fields even when sharp transition operators are used.

Keywords: Implicit modelling, Soft objects, Bounded representation, CSG operators, Blending.

1991 Mathematics Subject Classification: 22E46, 53C35, 57S20

1. Introduction

Providing interactive, accurate and intuitive control of shapes is a fundamental issue in the development of three-dimensional modelling techniques. Direct manipulation of meshes, parametric shape representations and, more recently, subdivision surfaces are common and useful solutions adopted by most commercial software. However, implicit volume models^{1,2} are rapidly becoming a practical alternative to these methods due to the increase

2 *Loïc Barthe and Brian Wyvill and Erwin de Groot*

in computer power and storage capacity of modern workstations combined with the latest developments in graphics hardware.

Among the advantages of implicit surfaces we notice their natural blending property^{3,4,5}, the true three-dimensional representation of volumes, the efficiency for collision tests², the Boolean composition of their volume by simple function composition^{6,8,9,10} and finally their very compact functional representation.

Two general implicit representations can be distinguished: A bounded representation where the function defining the volume returns a constant value outside the boundary (known as “Metaballs”¹¹ or “Soft Objects”¹²), and an unbounded representation (such as R -functions^{8,9}) where the function varies in the whole space. Unbounded representation provides a general implicit volume representation¹³ and therefore, a wide variety of modelling operators such as sweeping by moving solids¹⁴, Boolean composition with soft transition^{9,15,16,17} and Constructive Volume Geometry algebra¹⁸, have been proposed. Due to their global definition, it is difficult to provide both accurate and interactive surface rendering³⁰. The local representation of bounded objects is better suited for this purpose. “Soft Objects” are more popular for their automatic blending property and whereas many blending functions have been presented in the literature (see^{20,21,22} for a summary), as far as we know, since Ricci⁶, no consequent improvement has been proposed for Boolean composition operators on “Soft Objects”. Under its actual form, composition with sharp transition generates undesirable discontinuities in the potential field and smooth composition provides a very limited control of the form of the transition.

Smooth transitions in the composition operators have become a standard tool for implicit modelling. Intuitive parameters for control are necessary in order to allow the user to design the desired shape. On the other hand it has been shown that the local definition of the bounded representation is a fundamental advantage which provides complete interactive modelling tools^{23,24}. Therefore providing Boolean composition operators (with or without smooth transition) for bounded implicit primitives with properties well suited for CSG compositions remains an open, rather important problem.

In this paper, we first state the limits of the actual methods used to combine bounded objects. We then present specific properties that composition operators should satisfy in order to provide the surface resulting from a composition with, at least, a G^1 continuous potential field. We introduce a generic family of composition functions based on arc-of-ellipses. From this general function representation, we derive new operators to combine bounded primitives with both smooth transition integrating “point-by-point” control of the shape (as introduced by Barthe et al²⁵), and sharp transition with a G^1 continuous potential field everywhere outside the surface and the boundary.

2. Related works

“Metaballs”¹¹ or “Soft Objects”¹² are bounded objects defined by a potential function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$. A single primitive, also called “skeleton primitive”, is first defined from a simple geometric object S (the skeleton) such as a point, a line, a polygon, etc. Then one has to choose a metric d which is generally the Euclidean norm. The potential function f

is defined as a function of the distance with respect to the norm d from the skeleton S to points p of \mathbb{R}^3 :

$$\begin{aligned} f : \mathbb{R}^3 &\rightarrow \mathbb{R} \\ p &\rightarrow f(r) \quad \text{with } r = d(S, p). \end{aligned} \tag{1}$$

We denote r as $d(S, p)$. The primitive's boundary is defined by a chosen scalar R called "radius of influence". The function f equals 0 if $r \geq R$ and it decreases from 1 to 0 when r increases from 0 to R , following a Gaussian-like variation. The surface is defined by the set of points $p_0 \in \mathbb{R}^3$ such that $d(S, p_0) = r_0$ and $f(r_0) = C$, where C is a pre-chosen value in $(0, 1)$ (usually $1/2$), and the volume object is defined by the set of points of \mathbb{R}^3 for which $f(r) \geq C$ (see Figure 1). A wide variety of primitives are available ^{4,26}, and the blend of a set of n primitives is automatically computed by summing their potential functions f_i ($i = 1..n$):

$$F(f_i) = \sum_{i=1}^n f_i, \tag{2}$$

where F is a new potential function which has the same analytic properties as functions f_i . Many different field functions f ^{20,21,22} and blending models ^{27,28} have been proposed to control the smoothness of the transition region, but the operators remain limited to the blending and the control of which primitives must and must not blend. The locality of the definition and the capacity to be automatically blended allow modelling techniques based on these objects to be interactive ^{24,29}.

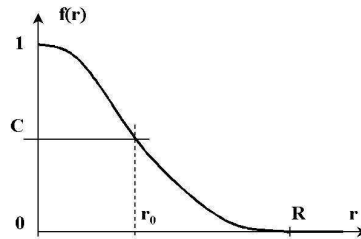


Fig. 1. Graph of a potential function f defining a "Soft Object".

CSG composition operators are already supported by bounded primitives (using the Ricci's min / max operators ⁶) but discontinuities are introduced in the gradient of the potential field of the resulting object, altering the smoothness of the transition when it has to be blended (Figure 2). This is undesirable.

A solution using Ricci's super-elliptic operator ⁶ to apply binary union and binary intersection operators with smooth transitions to "Soft Objects" was used by Wyvill et al ²³ (see Equation 3), and extended to n-ary operations.

$$G(f_1, f_2) = (f_1^n + f_2^n)^{\frac{1}{n}}. \tag{3}$$

4 Loïc Barthe and Brian Wyvill and Erwin de Groot

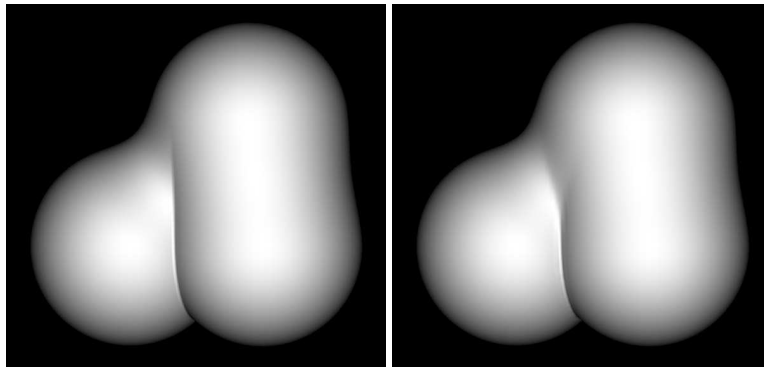


Fig. 2. On the left, the two bottom spheres are first combined (union) with sharp transition using Ricci's max operator. The resulting object is then combined with the top sphere (union) using smooth transition. We can see the undesirable discontinuity in the middle of the smooth transition. On the right we show a correct smooth transition.

This operator has a single parameter n which controls the transition sharpness. There is no explicit link with some geometric parameter allowing the user to interactively select the beginning, the end, or the form of the transition. The user is limited to approximately select either, the global sharpness of the transition, or where the transition starts on one of the combined primitives, or where it finishes on the other one (in the case of a binary operator).

Recently Barthe et al.³⁰ and Hsu et al.¹⁷ introduced new CSG operators with smooth transitions. In Hsu et al., operators can be n -ary and a lot of flexibility is provided in the choice of the function defining the operator. However, the control over the shape remains limited and the operator is very expensive to evaluate. On the other hand, Barthe et al. proposed binary operators based on¹⁶ which are cheaper to evaluate and which provide a high level of control on the shape of the transition. Since our goal is to allow the user to accurately control the shape of the smooth transition, we use operators presented in¹⁶ as a basis to develop our operators for "soft objects".

3. Fundamental properties

As far as we know, there is no clear definition of what is to be expected from a CSG Boolean composition operator on bounded implicit primitives. However, some first insights are given by the properties satisfied by the composition operators on unbounded primitives^{5,25}. The main difference between the *two* representations is the boundary. While we are expecting equivalent properties in terms of potential field variations and shape control, we also have to maintain a consistent potential field around and at boundaries. The constraints can be specified as follows:

- The potential field produced by the composition operator must be at least G^1

continuous everywhere inside the object boundary.

- The "automatic blending" property by potential summation must be conserved through the composition.
- The result of a composition must be a bounded object having its bounding box easily computed from those of the combined primitives.
- For smooth composition, the extremities of the transition should be able to be intuitively selected once the objects' boundaries (radii of influence) are fixed.

In order to introduce these properties, we first look at the expected result and work backwards, up to the composition operator. We only present the union Boolean operator because once this case is understood, properties for intersection and difference can be directly derived without major difficulty.

Figure 3 illustrates the result of the union of *two* spheres with smooth transition. Object i ($i = 1, 2$) is defined by a potential function f_i . The equation $f_i = C$ represents object's i surface, and inequalities $f_i > C$ and $f_i < C$ define the inside and outside of object i respectively. Equation $f_i = 0$ defines the part of space \mathbb{R}^3 on and outside object's i boundary (zone 4 in Figure 3).

From the graph shown in Figure 3 we plot the union binary combination operator G (Figure 4) and we deduce the definition of operator G in each zone:

- In zone 1, $f_1 > 0$ and $\forall f_2, f_2 = 0$. Therefore, in this area, operator G is a *one*-dimensional map $G(f_1, 0)$ which scales the values of function f_1 .
- In zone 2, $f_2 > 0$ and $\forall f_1, f_1 = 0$. Therefore, in this area, operator G is a *one*-dimensional map $G(0, f_2)$ which scales the values of function f_2 .
- In zone 3, $f_1 > 0$ and $f_2 > 0$. Operator G is a *two*-dimensional function $G(f_1, f_2)$ which defines a *two*-dimensional potential field.
- In zone 4, $f_1 = f_2 = 0$. Here, $G(f_1, f_2) = G(0, 0) = C^{te}$, $C^{te} \in \mathbb{R}$.

Function G can define a *two*-dimensional potential field only in zone 3. Hence, the transition between the combined primitives has to be fully defined in this zone, and no transition can be performed outside one of the objects boundary. This also implies that the continuity between the transition and the combined primitives has to be ensured in this zone, or at its boundary (as shown with small circles in Figure 4). Since we want to produce smooth potential fields, the continuity at the junction has to be at least G^1 i.e. the partial derivatives of function G must satisfy the following properties: $\partial G / \partial f_2 = 0$ at the junction between zones 1 and 3, and $\partial G / \partial f_1 = 0$ at the junction between zones 2 and 3.

In zone 1, function G scales the values of f_1 and if $G(f_1, 0) = f_1$, operator G reproduces the metric and the variations of function f_1 , and the potential field defined by f is preserved through composition. This property avoids the introduction of non-uniform variations in the potential fields which could alter the regularity of the transition and the "automatic blending" property. In zone 2, function G scales the value of f_2 and as argued for zone 1, a pertinent definition of operator G is: $G(0, f_2) = f_2$.

In zone 4, operator G is constant. It represents the outside of the resulting object boundary, and since it has to be continuously (at least G^1) joined with the other zones, an obvious

6 Loïc Barthe and Brian Wyvill and Erwin de Groot

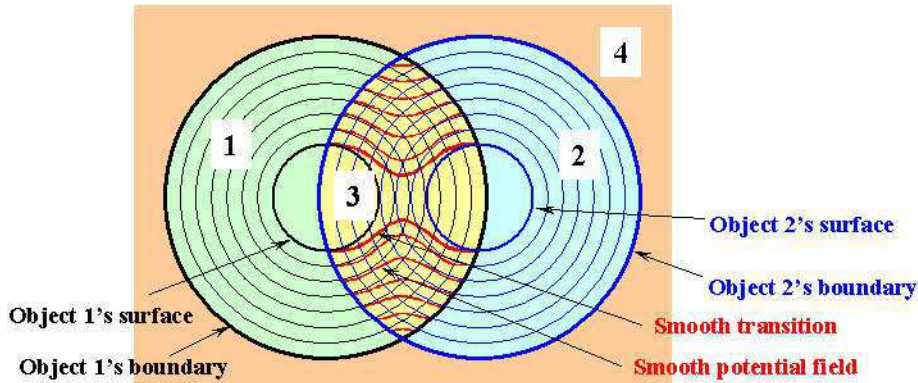


Fig. 3. Graph illustrating a 2D section of the potential fields when *two* spheres are combined (union) with smooth transition. Zone 1 is inside object 1 boundary and outside object 2 boundary, zone 2 is inside object 2 boundary and outside object 1 boundary, zone 3 is the intersection of objects' boundaries and zone 4 is outside both boundaries. Lines represent sections of iso-surfaces. In zone 3, we also show sections of the smooth transition.

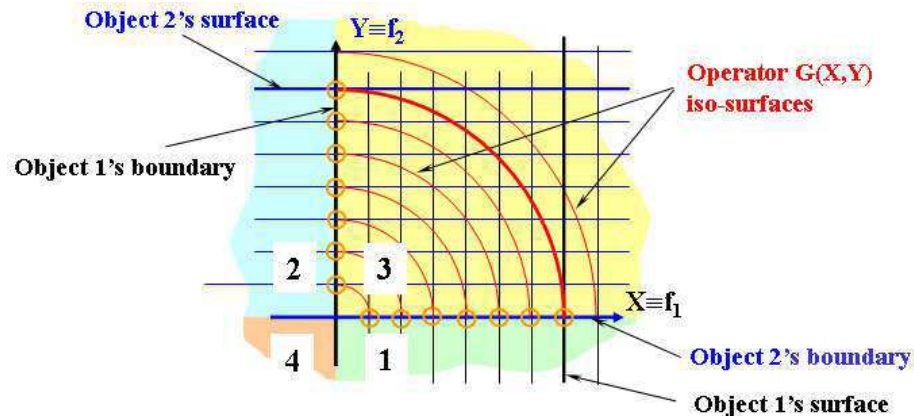


Fig. 4. Plot of the union binary composition operator G which generates the smooth composition shown in Figure 3. In order to better correspond to Figure 3, zone 1 can be reduced to the X axis ($Y = 0, \forall X$), zone 2 can be reduced to the Y axis ($X = 0, \forall Y$) and zone 4 can be reduced to the point $(0, 0)$ ($X = 0$ and $Y = 0$).

value is: $G(0, 0) = 0$.

These fundamental statements give us a theoretical basis to provide Boolean composition operators for bounded implicit primitives. Note that the bounding box of the resulting object is easy to compute. For the union operator, the box is the union of those of the two combined primitives. For the intersection operator, it is their intersection and for the

difference it is the box of the object defined by function f_1 (for the case $object_1 \setminus object_2$).

We now emphasize the link between composition operators for unbounded primitives and bounded primitives. If we look at Figure 4 carefully, we see that the operator which combines bounded objects with smooth transition also combine the zero-isosurface with sharp transition (with a smooth potential field everywhere else). Hence, as done in^{30,17}, it is desirable to study operators combining unbounded implicit primitives with sharp transition^{9,16,17}. Note that those proposed in⁹ as well as operators with smooth transition for unbounded objects do not satisfy the continuity conditions at the junction between the different zones.

4. Generic arc-of-an-ellipse function

In this Section, we introduce the general form of an operator G which satisfies the conditions presented in the previous Section. It is based on a geometric construction of function G and a specific adaptation has already been used in¹⁶ to combine unbounded implicit primitives with sharp transitions.

In order to respect the different constraints, operator G is piecewise defined. In zones 1 and 2, it returns the value of f_1 , respectively f_2 (as suggested in Section 3). In zone 4 it returns 0 and in zone 3, we propose to link the vertical iso-lines defined by $G(f_1, 0) = f_1$ to the horizontal iso-lines defined by $G(0, f_2) = f_2$ with a quarter of an ellipse (Figure 5) defined by the following equation:

$$\frac{(X_p - X_{p0})^2}{(C_p - X_{p0})^2} + \frac{(Y_p - Y_{p0})^2}{(C_p - Y_{p0})^2} = 1, \tag{4}$$

where a point $P \in \mathbb{R}^2$ has the coordinates $P(f_1 = X_p, f_2 = Y_p)$ and the potential in this point is $G(P) = G(X_p, Y_p) = C_p$. The center of the ellipse passing by the point P is the point $P_0(X_{p0}, Y_{p0})$.

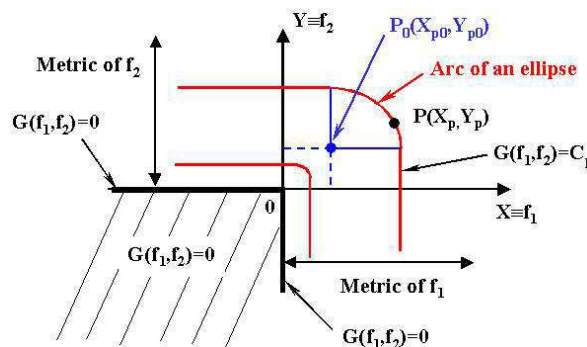


Fig. 5. Plot of the general form of a union composition operator G . The arc-of-an-ellipse links the horizontal and vertical half-lines with a G^1 continuity.

8 Loïc Barthe and Brian Wyvill and Erwin de Groot

The junction between the constant map $G(f_1, 0) = f_1$ (or $G(0, f_2) = f_2$) and the quadratic arc-of-an-ellipse is G^1 continuous and they are both internally G^∞ .

In Equation 4, unknowns X_{p0} and Y_{p0} are to be expressed in terms of C_p and the equation is solved to compute the value C_p returned by operator G at a point $P(X_p, Y_p)$. In the following Sections, we use *two* different geometric constructions based on this general definition in order to provide Boolean composition with and without smooth transition.

5. Operators with smooth transition

Our operator G is already designed to conserve the combined primitives' metrics outside the transition. All we have to do is to define boundaries for the arc-of-an-ellipse. For this purpose, we introduce *two* angles θ_1 and θ_2 , as illustrated in Figure 6, and it allows us to determine the unknowns X_{p0} and Y_{p0} :

$$X_{p0} = \frac{C_p}{\tan(\theta_2)} = C_p \cot(\theta_2) \quad \text{and} \quad Y_{p0} = C_p \tan(\theta_1). \quad (5)$$

The adaptation of operator G to bounded primitives composition operators with smooth transitions is denoted \widetilde{G}_U .

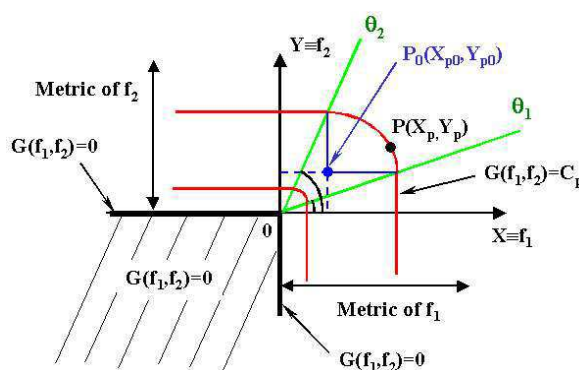


Fig. 6. Graph of our operator \widetilde{G}_U . Angles θ_1 and θ_2 are introduced in order to bound the arc-of-an-ellipse.

To allow accurate and intuitive control, the transition must be defined by control points on the C iso-potential surface. The first adaptation is to define angles θ_1 and θ_2 from the user Euclidean space \mathbb{R}^3 by selecting control points $p_1(x_1, y_1, z_1)$ and $p_2(x_2, y_2, z_2)$ on the combined objects' surface, respectively $f_1 = C$ and $f_2 = C$ (Figure 7). We notice that points p_1 and p_2 must be selected inside the intersection of the objects' boundaries because, no transition can be defined outside these limits (as explained in Section 3). Our geometric construction of operator \widetilde{G}_U leads us to the following equation (equations of operators \widetilde{G}_\cap and \widetilde{G}_\setminus are given in Appendix A):

Points $p_1 \in \mathbb{R}^3$ and $p_2 \in \mathbb{R}^3$ selected by the user correspond to points P_1 and P_2 such as: $P_1(C, f_2(p_1))$ and $P_2(f_1(p_2), C)$.

$$\theta_1 = \text{angle}([OX], [OP_1]), \quad \theta_2 = \text{angle}([OX], [OP_2])$$

$$\text{At point } P(X_p, Y_p) : \theta_p = \text{angle}([OX], [OP])$$

$$\widetilde{G}_U(X_p, Y_p) = \begin{cases} X_p & \text{if } Y_p = 0 \\ Y_p & \text{if } X_p = 0 \\ X_p & \text{if } \theta_p \leq \theta_1 \\ Y_p & \text{if } \theta_p \geq \theta_2 \\ C_p & \text{where } C_p \text{ is the solution of:} \\ & \frac{(X_p - C_p \cdot \cot(\theta_2))^2}{(C_p - C_p \cdot \cot(\theta_2))^2} + \frac{(Y_p - C_p \cdot \tan(\theta_1))^2}{(C_p - C_p \cdot \tan(\theta_1))^2} = 1 \\ & \text{if } \theta_p \in (\theta_1, \theta_2) \end{cases} \quad (6)$$

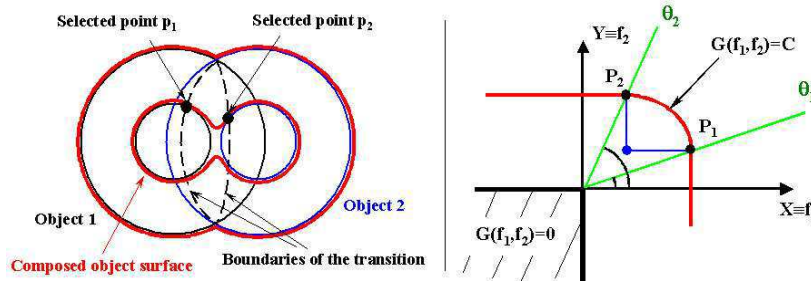


Fig. 7. Union with smooth transition controlled point-by-point in the user Euclidean space and its function representation.

The closed form solution for the evaluation of C_p in Equation 6 is given in ¹⁶. With operator \widetilde{G}_U in this form, only the boundaries of the transition can be controlled. For fixed angles θ_1 and θ_2 it is necessary to be able to choose the smoothness of the transition. This leads us to add at least one control point. In fact, adding one or more control points brings us to the same solution. To conserve the G^1 continuity in the field, we multiply $\widetilde{G}_U(P)$ by a function $m(\theta_p)$ where m is an interpolation function of $\mathbb{R} \rightarrow \mathbb{R}$ when $\theta_p \in [\theta_1, \theta_2]$ and $m(\theta_p) = 1$ otherwise. A valid graph for function m is shown in Figure 8. The link with the control points is done as follows: $m(\theta_1) = 1, m'(\theta_1) = 0$ and $m(\theta_2) = 1, m'(\theta_2) = 0$ to ensure G^1 continuity at the beginning and the end of the transition. Then k_i ($i > 2$) are computed from the control points $p_i(x_i, y_i, z_i)$ ($i > 2$) selected in the Euclidean modelling space \mathbb{R}^3 . Point p_i allows us to compute the point $P_i(f_1(p_i), f_2(p_i))$, followed by θ_{p_i} and

10 *Loïc Barthe and Brian Wyvill and Erwin de Groot*

$C_i = \widetilde{G}_U(P_i)$. The corresponding point k_i ($i > 2$), to interpolate, has then the coordinates: $k_i(\theta_i, C/C_i)$. We have chosen one-dimensional cubic polynomial splines³¹ to define function m when $\theta_p \in [\theta_1, \theta_2]$ for their adequate smoothness and oscillation properties, and for their inexpensive computation cost. We finally obtain the union Boolean operator with “functionally defined” transitions for bounded objects in Equation 7.

$$\widetilde{G}_U^{final}(P) = m(\theta_p) \cdot \widetilde{G}_U(P) \quad (7)$$

The same path has been followed to build the intersection Boolean operator with “functionally defined” transition $\widetilde{G}_\cap^{final}(P) = m(\theta_p) \cdot \widetilde{G}_\cap(P)$ from $\widetilde{G}_\cap(P)$.

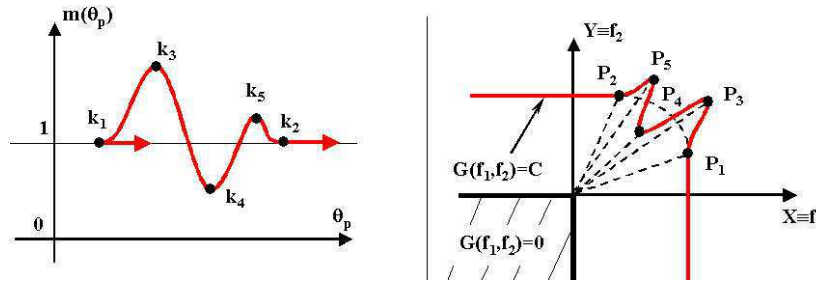
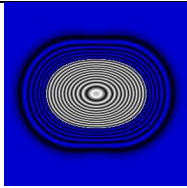
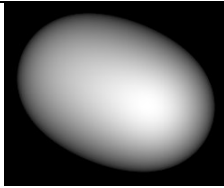
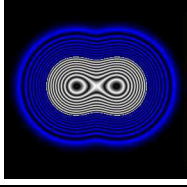
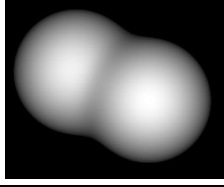
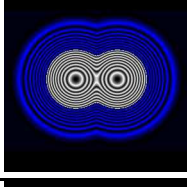
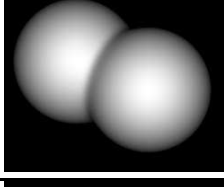
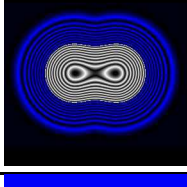
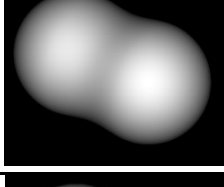
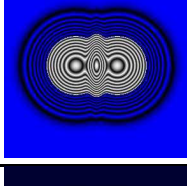
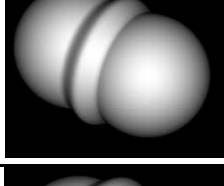
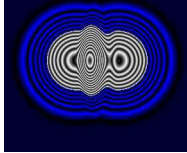
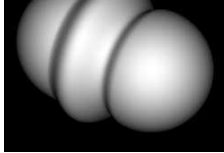


Fig. 8. Graph of an interpolation function $m(\theta_p)$ used to deform the operator \widetilde{G}_U and allow the control point-by-point of the transition.

The difference operator \widetilde{G}_\setminus cannot be directly derived with the standard method used for unbounded primitives: $\widetilde{G}_\setminus(f_1, f_2) = \widetilde{G}_\cap(f_1, -f_2)$. Ricci⁶ proposed the realization of the difference operator on bounded implicit objects using the intersection operator applied on f_1 and $(2C - f_2)$ instead of f_2 . The same method used on our intersection operator $\widetilde{G}_\cap^{final}$ gives the difference Boolean operator with “functionally defined” transition: $\widetilde{G}_\setminus^{final}$. Figure 9(c) shows a ring object built from the ring of Figure 9(a) and a gem similar to that in Figure 9(b). The gem has been further modified by two profile curves. The first profile curve modifies the smooth intersection operation used to construct the gem. The second modifies the difference operation between the gem and a sphere (implicit point primitive) at its center. Finally, another sphere has been added with a smooth union operation. The gem is then joined to the ring using another smooth union with a profile curve. Table 1 illustrates different union composition operators with smooth transition and allows us to compare of the computation times, the potential field variations and the shape produced at the transition level. The increase of the evaluation cost of our operators is compensated by the controllability of the form of the transition.

A function m of $\mathbb{R} \rightarrow \mathbb{R}$ is used to provide point-by-point control. Because such a function must be single valued, we do not obtain a true free-form control of the transition (as proposed in¹⁶ for unbounded primitives). However, we deal with bounded objects. The transition in operators like ours is then essentially used to smooth the junction of

Table 1. Time in milliseconds to compute potential function values for a 128^3 grid (2097152 evaluations), using a 1.0 GHz AMD Athlon processor with 512 Mbytes of DDR memory. (a) Using Ricci's operator with $n = 1$, (b) Ricci's operator with $n = 3$, (c) Ricci's operator with $n = 7$, (d) operator \tilde{g}_U , (e) operator \tilde{g}_U^{final} with 3 control points and (f) operator \tilde{g}_U^{final} with 5 control points. The middle column shows two-dimensional sections of the full grid.

| time | 2D section | 3D object |
|----------|---|--|
| (a) 246 |  |  |
| (b) 884 |  |  |
| (c) 940 |  |  |
| (d) 1005 |  |  |
| (e) 1933 |  |  |
| (f) 2315 |  |  |

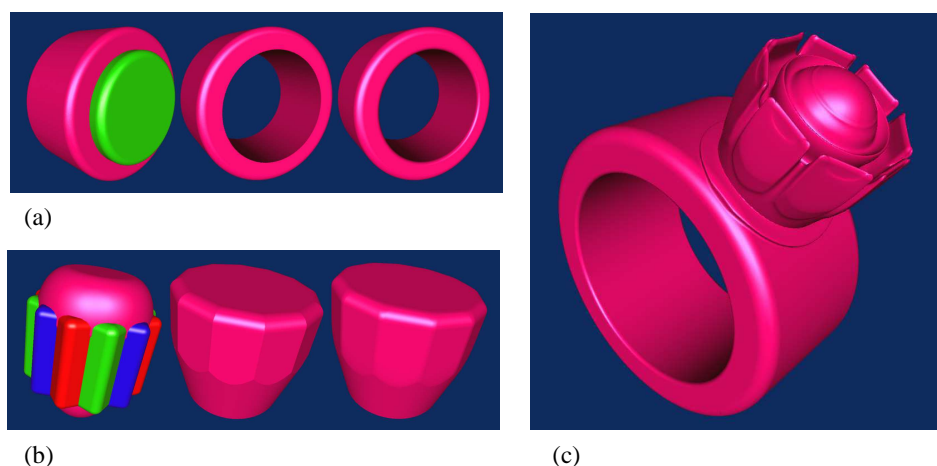
12 *Loïc Barthe and Brian Wyvill and Erwin de Groot*

Fig. 9. (a) The ring is built using two implicit cylinders and applying subtraction, the center image uses the Ricci CSG subtraction operator, the right hand image, our smooth CSG subtraction \widehat{G}_\setminus . (b) The gemstones are built from implicit *box* primitives and an implicit cone, center image uses the Ricci intersection operator, right hand image our smooth intersection \widehat{G}_\cap . (c) Applying the smooth CSG operators \widehat{G}_\cup , \widehat{G}_\cap , \widehat{G}_\setminus and profile curve operators \widehat{G}_\cap^{final} , $\widehat{G}_\setminus^{final}$ on bounded implicit objects.

two primitives when they are combined. We assume that in a general case, to create the desired smooth transition, three, four or five control points give enough flexibility. Free-form curves are needed in very specific cases, and often it remains easier to build a new primitive and to combine it with a smooth transition.

6. Operators with sharp transition

Our operator G combines the primitives' boundaries with sharp transition and generates a smooth G^1 potential field everywhere else using an arc-of-an-ellipse. Since we want to combine the object surfaces with sharp transition, we have to provide a solution where operator G does not smooth both boundaries and C iso-surfaces, but still smooth the potential field everywhere else. In order to satisfy this additional constraint, we propose the geometric construction illustrated in Figure 10 and we denote our binary union operator with sharp transition as \widehat{G}_\cup . Note that we cannot derive a simple solution using the operators proposed in ^{30,17} because they generate discontinuities in the field around the origin $(0,0)$, at the level of the X and Y axes (see Figure 4).

This geometric representation leads us to the following definition of unknowns X_{p0} and Y_{p0} :

$$\text{In zone } a: X_{p0} = Y_{p0} = \frac{C_p^2}{C} = C_a C_p^2 \quad \text{with } C_a = \frac{1}{C} \quad (8)$$

$$\text{in zone } b: X_{p0} = Y_{p0} = \sqrt{CC_p}. \quad (9)$$

Operators \widehat{G}_\cap and \widehat{G}_\setminus are presented in Appendix B and the closed form solution for the

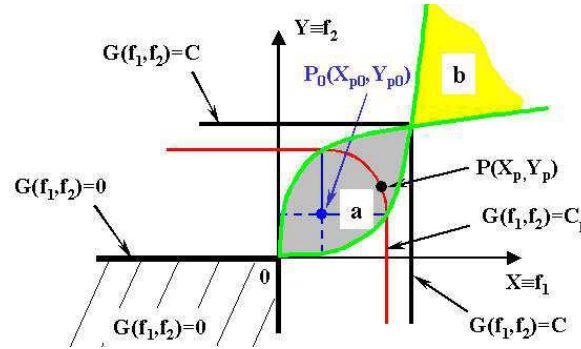


Fig. 10. Graph of our operator \widehat{G}_U . Both boundaries and C iso-surfaces are combined with sharp transition while the potential field is G^1 continuous everywhere else.

evaluation of C_p in Equation 10 is given in Appendix C. Figure 11 illustrates the effect of operator \widehat{G}_U on both shape and potential field, and Figure 2 shows its smoothing property when the resulting object is to be combined with smooth transition.

$$\widehat{G}_U(X_p, Y_p) = \begin{cases} X_p & \text{if } Y_p = 0 \\ Y_p & \text{if } X_p = 0 \\ Y_p & \text{if } Y_p \geq \sqrt{CX_p} \text{ and } Y_p \leq C \\ X_p & \text{if } Y_p \leq C_a X_p^2 \text{ and } X_p \leq C \\ Y_p & \text{if } Y_p \geq C_a X_p^2 \text{ and } Y_p > C \\ X_p & \text{if } Y_p \leq \sqrt{CX_p} \text{ and } X_p > C \\ C_p & \text{where } C_p \text{ is the solution of:} \\ & \frac{(X_p - C_a C_p^2)^2 + (Y_p - C_a C_p^2)^2}{(C_p - C_a C_p^2)^2} = 1 \\ & \text{if } P(X_p, Y_p) \text{ is in zone } a \\ C_p & \text{where } C_p \text{ is the solution of:} \\ & \frac{(X_p - \sqrt{CC_p})^2 + (Y_p - \sqrt{CC_p})^2}{(C_p - \sqrt{CC_p})^2} = 1 \\ & \text{if } P(X_p, Y_p) \text{ is in zone } b \end{cases} \quad (10)$$

The evaluations of \widehat{G}_U for a 128^3 grid (2097152 evaluations), using a 1.0 GHz AMD Athlon processor with 512 Mbytes of DDR memory takes 2010 milliseconds (to create the object of the central image in Figure 11). We notice that we first store the combined primitives in regular grids. Therefore, the given time does not depend on the complexity of the combined objects, and the composition of *two* complex primitives leads us to the same result. What is time consuming is the complexity and the expensive computation of \widehat{G}_U when it has to be evaluated in zones *a* and *b*. However, we are dealing with bounded objects, and they do not need to be evaluated at a point of \mathbb{R}^3 which is located outside their bounding-box. Moreover, if the point is inside the box, the expensive evaluation occurs

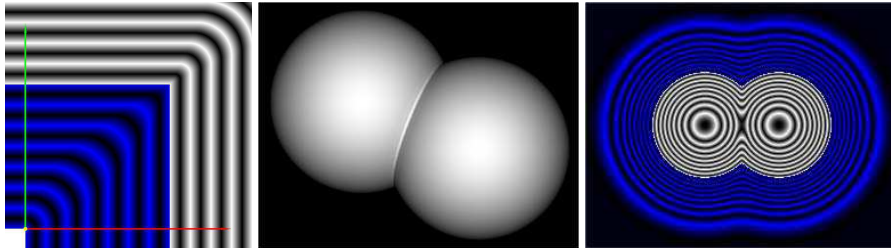
14 *Loïc Barthe and Brian Wyvill and Erwin de Groot*

Fig. 11. From left to right: Plot of our operator \widehat{G}_U , its effect on the composition of *two* implicit bounded spheres and a 2D section of the generated potential field. In the left and right Figures, the white area represents the inside of the object and the dark area, the outside.

only if the point is also inside the intersection of the combined objects boundaries. This significantly reduces the complexity brought by our operator when it is used to model a complex object.

7. Example

Figure 12 shows a teddy bear made of some simple primitives and the binary Boolean composition operators described in this paper. Figure 13 shows an arm of the teddy bear. The arm is the smooth difference between a scaled point primitive and a plane primitive. The shape of the end of the arm is created by placing controlpoints. Figure 14 shows an ear of the teddy bear. The ear is the sharp difference between a point primitive and the sharp union of a sphere and a plane. Once created, the ear is combined to the head with a smooth transition.

We point out that it is obvious that our operators are computationally expensive (as shown in Table 1). However, they have the fundamental property to be bounded, and hence, our operators have to be evaluated only in potential field areas where implicit primitives are combined. In other part of the space, the computational complexity remains the one of the combined primitives themselves.

Figures 9,12,13 and 14 are computed using a standard ray marching algorithm without any optimisation. The computation of Figure 12 took ten hours on an Intel PIII 1.3Ghz. The use of a faster raytracing technique will significantly reduce this time, but fast rendering is out of the scope of this paper.

8. Conclusion

We have presented a theoretical background which gives us a basis for the construction of binary Boolean composition operators for bounded implicit primitives. Boolean composition operators with smooth or sharp transition can be derived while preserving a G^1 continuous potential field. The operators \widetilde{G}^{final} greatly increase accuracy and freedom in the control of the transition when primitives are smoothly combined. The operators \widehat{G} limit the discontinuities in the composed objects to a minimum, so these objects can be used in



Fig. 12. Teddy bear made from simple primitives and the operations described in this paper

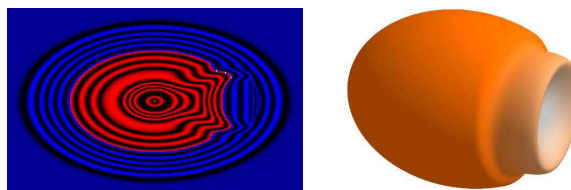


Fig. 13. An arm of the teddy bear. White circles indicate controlpoints.

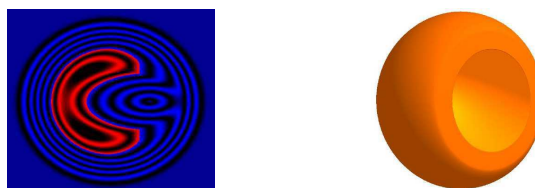


Fig. 14. An ear of the teddy bear.

16 *Loïc Barthe and Brian Wyvill and Erwin de Groot*

other operations without generating G^1 discontinuities in the transition area, so that even after several Boolean compositions, the resulting object still have the “automatic blending” property.

However, our operators remain expensive to evaluate and they are still limited to binary compositions. Hence, we can say that these results provide a basis to study new composition operators for bounded implicit primitives, and improvements can be investigated in order to propose fast evaluated operators or operators satisfying specific properties like the Lipschitz³² condition to accelerate the rendering for example. The extension from binary operators to n-ary operators is also interesting. However, we did not find any direct derivation of our operators in order to define n-ary operators and the definition of n-ary operators providing both accurate control and smooth transition seems to be a challenging problem.

Acknowledgements

This work has been partially funded by E.U. through the MINGLE project. Contract HPRN-CT-1999-00117

Appendix A. Operators \widetilde{G}_\cap and \widetilde{G}_\setminus

- Operator \widetilde{G}_\cap :

Points $p_1 \in \mathbb{R}^3$ and $p_2 \in \mathbb{R}^3$ selected by the user correspond to points P_1 and P_2 such as: $P_1(f_2(p_1), C)$ and $P_2(C, f_1(p_2))$.

$$\theta_1 = \text{angle}([OX], [OP_1]), \quad \theta_2 = \text{angle}([OX], [OP_2])$$

$$\text{At point } P(X_p, Y_p) : \theta_p = \text{angle}([OX], [OP])$$

$$\widetilde{G}_\cap(X_p, Y_p) = \begin{cases} 0 & \text{if } Y_p = 0 \\ 0 & \text{if } X_p = 0 \\ Y_p & \text{if } \theta_p \leq \theta_1 \\ X_p & \text{if } \theta_p \geq \theta_2 \\ C_p & \text{where } C_p \text{ is the solution of:} \\ & \frac{(X_p - C_p \cdot \cot(\theta_1))^2}{(C_p - C_p \cdot \cot(\theta_1))^2} + \frac{(Y_p - C_p \cdot \tan(\theta_2))^2}{(C_p - C_p \cdot \tan(\theta_2))^2} = 1 \\ & \text{if } \theta_p \in]\theta_1, \theta_2[\end{cases} \quad (\text{A.1})$$

- Operator \widetilde{G}_\setminus :

Operator \widetilde{G}_\setminus is directly obtained from operator \widetilde{G}_\cap using the following expression:

$$\widetilde{G}_\setminus(f_1, f_2) = \widetilde{G}_\cap(f_1, 2C - f_2) \quad (\text{A.2})$$

Appendix B. Operators \widehat{G}_{\cap} and \widehat{G}_{\setminus}

- Operator \widehat{G}_{\cap} :

$$\widehat{G}_{\cap}(X_p, Y_p) = \begin{cases} 0 & \text{if } Y_p = 0 \\ 0 & \text{if } X_p = 0 \\ X_p & \text{if } Y_p \geq \sqrt{CX_p} \text{ and } X_p \leq C \\ Y_p & \text{if } Y_p \leq C_a X_p^2 \text{ and } Y_p \leq C \\ X_p & \text{if } Y_p \geq C_a X_p^2 \text{ and } X_p > C \\ Y_p & \text{if } Y_p \leq \sqrt{CX_p} \text{ and } Y_p > C \\ C_p & \text{where } C_p \text{ is the solution of:} \\ & \frac{(X_p - \sqrt{CC_p})^2 + (Y_p - \sqrt{CC_p})^2}{(C_p - \sqrt{CC_p})^2} = 1 \\ & \text{if } P(X_p, Y_p) \text{ is in zone } a \\ C_p & \text{where } C_p \text{ is the solution of:} \\ & \frac{(X_p - C_a C_p^2)^2 + (Y_p - C_a C_p^2)^2}{(C_p - C_a C_p^2)^2} = 1 \\ & \text{if } P(X_p, Y_p) \text{ is in zone } b \end{cases} \quad (\text{B.1})$$

- Operator \widehat{G}_{\setminus} :

Operator \widehat{G}_{\setminus} is directly obtained from operator \widehat{G}_{\cap} using the following expression:

$$\widehat{G}_{\setminus}(f_1, f_2) = \widehat{G}_{\cap}(f_1, 2C - f_2) \quad (\text{B.2})$$

Appendix C. Closed form solution for the evaluation of C_p in our new composition operators with sharp transition

Solution for the equation:

$$\frac{(X_p - C_a C_p^2)^2 + (Y_p - C_a C_p^2)^2}{(C_p - C_a C_p^2)^2} = 1 \quad (\text{C.1})$$

C_p is one of the roots of the following equation:

$$C_a^2 C_p^4 + 2C_a C_p^3 - (2C_a X_p + 2C_a Y_p + 1)C_p^2 + X_p^2 + Y_p^2 = 0 \quad (\text{C.2})$$

C_p is computed with:

$$\begin{aligned} S_{11} &= 48C_a^3 Y_p^2 X_p + 64C_a^3 Y_p^3 + 78C_a^2 Y_p^2 + 64C_a^3 X_p^3 + 48C_a^3 X_p^2 Y_p + 78C_a^2 X_p^2 - 24C_a^2 X_p Y_p - 6C_a X_p - 6C_a Y_p - 1 \\ S_{12} &= -3(X_p^2 + Y_p^2)(-64C_a^3 X_p^3 + 16C_a^3 X_p^2 Y_p - 35C_a^2 X_p^2 + 64C_a^4 X_p^2 Y_p^2 + 72C_a^2 X_p Y_p) \\ S_{13} &= -3(X_p^2 + Y_p^2)(16C_a^3 Y_p^2 X_p + 14C_a X_p + 14C_a Y_p - 64C_a^3 Y_p^3 - 35C_a^2 Y_p^2 + 2) \\ S_1 &= S_{11} + 6 C_a \sqrt{S_{12} + S_{13}} \\ S_2 &= \frac{5 \sqrt[3]{S_1} + 4 \sqrt[3]{S_1} C_a X_p + 4 \sqrt[3]{S_1} C_a Y_p + \sqrt[3]{S_1}^2 + 16C_a^2 Y_p^2 + 16C_a^2 X_p^2 + 8C_a^2 X_p Y_p + 4C_a X_p + 4C_a Y_p + 1}{\sqrt[3]{S_1}} \\ S_3 &= \sqrt{S_2} \sqrt[3]{S_1} C_a Y_p \\ S_4 &= \sqrt{S_2} \sqrt[3]{S_1} C_a X_p \end{aligned}$$

18 *Loïc Barthe and Brian Wyvill and Erwin de Groot*

$$\begin{aligned}
S_5 &= \sqrt{S_2} \sqrt[3]{S_1} \\
S_6 &= \sqrt{S_2} \sqrt[3]{S_1^2} \\
S_7 &= \sqrt{3} \sqrt[3]{S_1} \\
R_1 &= -10S_5 - 8S_4 - 8S_3 + S_6 + 16\sqrt{S_2}C_a^2Y_p^2 + 16\sqrt{S_2}C_a^2X_p^2 + 8\sqrt{S_2}C_a^2X_pY_p + 4\sqrt{S_2}C_aX_p + 4\sqrt{S_2}C_aY_p + \sqrt{S_2} \\
R_2 &= 12S_7(C_aX_p + C_aY_p + 1) \\
C_p &= \frac{1}{6C_a} \left(-3 + \sqrt{3S_2} - \sqrt{\frac{-3(R_1 + R_2)}{S_5}} \right) \tag{C.3}
\end{aligned}$$

We notice that in the previous equation, values S_i and R_1 can be complex. Therefore, all the computations have to be done with complex numbers, even if the result C_p is real.

Solution for the equation:

$$\frac{(X_p - \sqrt{CC_p})^2 + (Y_p - \sqrt{CC_p})^2}{(C_p - \sqrt{CC_p})^2} = 1 \tag{C.4}$$

C_p is one of the roots of the following equation:

$$-C_p^2 + 2C_p\sqrt{CC_p} + CC_p - 2(X_p + Y_p)\sqrt{CC_p} + X_p^2 + Y_p^2 = 0 \tag{C.5}$$

C_p is computed with:

$$\begin{aligned}
S_{11} &= -18C^5X_p - 18C^5Y_p - 36C^4X_p^2 - 36C^4Y_p^2 + 108C^4X_pY_p - C^6 \\
S_{12} &= 18C^8X_p^3Y_p - 72C^7Y_p^3X_p^2 - 36C^7X_p^4Y_p^4 - 3C^{10}X_pY_p + 18C^8X_pY_p^3 - 48C^9X_pY_p^2 \\
S_{13} &= 36C^6X_p^2Y_p^4 - 36C^7X_p^5 + 3C^9X_p^3 + 42C^8X_p^4 + 3C^9Y_p^3 + 42C^8Y_p^4 - 36C^7Y_p^5 \\
S_{14} &= 12C^6X_p^6 + 12C^6Y_p^6 - 72C^7X_p^3Y_p^2 - 48C^9X_p^2Y_p + 165C^8X_p^2Y_p^2 + 36C^6X_p^4Y_p^2 - 36C^7Y_pX_p^4 \\
S_1 &= S_{11} + 12\sqrt{S_{12} + S_{13} + S_{14}} \\
S_3 &= \sqrt[3]{S_1} \\
S_2 &= (15C^2S_3 + 3S_3^2 + 36C^3X_p + 36C^3Y_p - 36C^2X_p^2 - 36C^2Y_p^2 + 3C^4)/S_3 \\
S_4 &= \sqrt{S_2} \\
R_1 &= 10C^2S_3S_4 - S_4S_3^2 - 12C^3X_pS_4 - 12C^3Y_pS_4 + 12C^2X_p^2S_4 + 12C^2Y_p^2S_4 - C^4S_4 \\
R_2 &= 36C^3S_3 - 36C^2X_pS_3 - 36C^2Y_pS_3 \\
C_p &= \frac{1}{C} \left(\frac{C}{2} - \frac{S_4}{6} + \frac{1}{6} \sqrt{\frac{3(R_1 - R_2)}{S_3S_4}} \right)^2 \tag{C.6}
\end{aligned}$$

We notice that in the previous equation, some values can be complex. Therefore, all the computations have to be done with complex numbers, even if the result C_p is real.

References

1. J. Bloomenthal (ed), *Introduction to Implicit Surfaces*, (Morgan-Kaufmann, 1997).
2. M. Chen, A. Kaufman and R. Yagel (eds), *Volume Graphics*, (Springer, London, 2000).

3. J.F. Blinn, "A Generalization of algebraic surface drawing", *ACM Transaction on Graphics*, **1**(3), (1982) pp. 235–256.
4. J. Bloomenthal and B. Wyvill, "Interactive techniques for Implicit Modeling", *Computer Graphics (Proc. of SIGGRAPH 1990)*, **24**(2), (1990) pp. 109–116.
5. A. P. Rockwood, "The Displacement Method for Implicit Blending Surfaces in Solid Models", *ACM Transaction on Graphics*, **8** (4), (1989) pp. 279–297.
6. A. Ricci, "A Constructive Geometry for Computer Graphics", *The Computer Journal*, **16**(2) (1973) pp. 157–160.
7. M.P. Gascuel, "An Implicit Formulation for Precise Contact Modeling Between Flexible Solids", *Computer Graphics (Proc. of SIGGRAPH 1993)*, (1993) pp. 313–320.
8. V. Shapiro, "Real functions for representation of rigid solids", *Computer Aided Geometric Design*, **11**(2) (1994) pp. 153–175.
9. A. Pasko, V. Adzhiev, A. Sourin and V. Savchenko, "Function Representation in Geometric Modeling: Concepts, Implementation and Applications", *The Visual Computer*, **8**(2) (1995) pp. 429–446.
10. B. Wyvill and K.v. Overveld, "Polygonization of Implicit Surfaces with Constructive Solid Geometry", *Journal of Shape Modelling*, **2**(4) (1996) pp. 257–274
11. H. Nishimura, M. Hirai, T. Kawai, I. Shirakara and K. Omura, "Object Modeling by Distribution Functions", *Electronics Communications (in Japanese)*, **J68**(D) (1985) pp. 718–725.
12. G. Wyvill, C. McPheeters and B. Wyvill, "Data Structure for Soft Objects", *The Visual Computer*, **2**(4) (1986) pp. 227–234.
13. V. Adzhiev, M. Kazakov and A. Pasko, "Hybrid System Architecture for Volume Modeling", *Computer and Graphics*, **24**(1) (2000) pp. 67–78.
14. A. Sourin and A. Pasko, "Function Representation for Sweeping by a Moving Solid", *IEEE Transaction on Visualization and Computer Graphics*, **2**(1) (1996) pp. 11–18.
15. G. Pasko, A. Pasko, M Ikeda and T. Kunii, "Bounded Blending Operations", *Proc. of Shape Modeling International 2002*, (2002) pp. 95–103.
16. L. Barthe, N.A. Dodgson, M.A. Sabin, B. Wyvill and V. Gaildrat, "Two-Dimensional Potential Fields for Advanced Implicit Modeling Operators", *Computer Graphics Forum*, **22**(1) (2003) pp. 23–33.
17. P.C. Hsu and C. Lee, "The Scale Method for Blending Operations in Functionally-Based Constructive Geometry", *Computer Graphics Forum*, **22**(2) (2003) pp. 143–158.
18. M. Chen and J. Tucker. "Constructive Volume Geometry", *Computer Graphics Forum*, **19**(4) (2000) pp. 281–293.
19. L. Barthe, B. Mora, N.A. Dodgson and M.A. Sabin, "Interactive Implicit Modelling Based on C^1 Continuous Reconstruction of Regular Grids", *International Journal of Shape Modeling*, **8**(2) (2002) pp. 99–117.
20. Z. Kacic-Alesic and B. Wyvill, "Controlled Blending of Procedural Implicit Surfaces", *Proc. of Graphic Interface 1991*, (1991) pp. 236–245.
21. C. Blanc and C. Schlick, "Extended Field Functions for Soft Objects", *Proc. of Implicit Surfaces 1995*, (1995) pp. 21–32.
22. P.C. Hsu and C. Lee, "Field Functions for Blending Range on Soft Objects", *Computer Graphics Forum*, Proc. of EUROGRAPHICS 2003, **22**(3) (2003).
23. B. Wyvill, A. Guy and E. Galin, "Extending the CSG Tree: Warping, Blending and Boolean Operations in an Implicit Surface Modeling System", *Computer Graphics Forum*, **18**(2) (1999) pp. 149–158.
24. E. Galin and S. Akkouche, "Incremental Polygonization of Implicit Surfaces", *Graphical Models*, **62**(1) (2000) pp. 19–39.
25. L. Barthe, V. Gaildrat and R. Caubet. "Extrusion of 1D implicit profiles: Theory and first application", *International Journal of Shape Modeling*, **7**(2) (2001) pp. 179–199.

- 20 *Loïc Barthe and Brian Wyvill and Erwin de Groot*
26. B. Crespín, C. Blanc and C. Schlick, ‘Implicit Sweep Objects’, *EUROGRAPHICS 1996*, **15**(3) (1996) pp. 165–174.
27. M. Desbrun and M.P. Gascuel, ‘Animating Soft Substances with Implicit Surfaces’, *Computer Graphics (Proc. of SIGGRAPH 1995)*, (1995) pp. 287–290.
28. A. Guy and B. Wyvill, ‘Controlled Blending for Implicit Surfaces Using a Graph’, *Proc. of Implicit Surfaces 1995*, (1995) pp. 107–112.
29. E. Ferley, M. P. Cani and J. D. Gascuel, ‘Practical Volumetric Sculpting’, *The Visual Computer*, **16**(8) (2000) pp. 469–480.
30. L. Barthe, N.A. Dodgson, M.A. Sabin, B. Wyvill and V. Gaildrat, ‘Different applications of two-dimensional potential fields for volume modeling’, *technical report UCAM-CL-TR-541, ISSN 1476-2986*, University of Cambridge, UK, (2002).
31. I.D. Faux and M.J. Pratt, ‘Computational Geometry for Design and Manufacture’, *Ellis Horwood*, (1979).
32. D. Kalra and A.H. Barr, ‘Guaranteed Ray Intersections with Implicit Surfaces’, *SIGGRAPH ’89 Proceedings*, **23**(3) (1989) pp. 297–306.

7.3.3 Fitted BVH for Fast Raytracing of Metaballs.

Auteurs : Olivier Gourmel, Anthony Pajot, Mathias Paulin, Loïc Barthe, Pierre Poulin
Revue : Computer Graphics Forum, Vol 29(2), Proceedings of Eurographics
Date : Mai 2010

Fitted BVH for Fast Raytracing of Metaballs

Olivier Gourmel¹, Anthony Pajot¹, Mathias Paulin¹, Loïc Barthe¹ and Pierre Poulin^{1,2}

¹IRIT-VORTEX, Université Paul Sabatier, Toulouse, France

²LIGUM, Dept. I.R.O., Université de Montréal

Abstract

Raytracing metaballs is a problem that has numerous applications in the rendering of dynamic soft objects such as fluids. However, current techniques are either limited in the visual effects that they can render or their performance drops as the number of metaballs and their density increase. We present a new acceleration structure based on BVH and kd-tree for efficient raytracing of a large number of metaballs. This structure is built from an adapted SAH using a fast greedy algorithm and allows the visualization of several hundreds of thousands of metaballs at interactive-to-real-time framerates. Our method can handle arbitrary rays to simulate any complex secondary effects such as reflections or soft shadows, and is robust with respect to the density of metaballs. We achieve this performance thanks to a balanced CPU-GPU (using CUDA) implementation of the animation, structure creation, and rendering.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Visible line/surface algorithms

1. Introduction

Modeling and visualizing fluids and other dynamic objects of varying topology is a difficult problem. One common representation uses a large number of implicit surfaces known as soft objects or metaballs [Blo97]. However visualizing a large number of metaballs in real time remains a difficult problem in itself due to their implicit nature.

A common method to display metaballs is ray casting, which has the benefit of rendering the surface very precisely. Since no analytical solution is known in the general case, the ray-isosurface intersection is usually computed iteratively via for instance, ray marching [Har93], Bezier clipping [NN94] or interval arithmetic [Flo08, KHK*09]. Unfortunately, computing such intersections can become problematic since the cost of evaluating the potential function of the surface increases as the number of metaballs grows. Most papers that tackle this issue try to reduce the cost and the number of evaluations of this potential function, for instance by reconstructing the potential function along the ray [WT90, NN94].

Another solution tessellates the isosurface, and then ras-

terizes the corresponding polygons [Ura06]. The main problem of this technique is that the resulting mesh is very dependent of the tessellation sampling grid, and thus some geometry can be missed if the grid resolution is too low. Moreover, the cost of meshing the surface can become another issue when dealing with large numbers of metaballs.

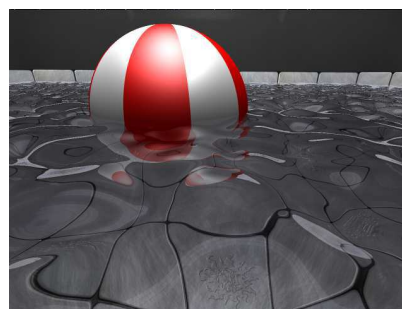


Figure 1: Screenshot of a fluid simulation composed of 10,000 metaballs rendered with reflection and refraction using our technique. Our acceleration structure makes their visualization possible at an interactive framerate.

In this paper, we present a new acceleration structure, combining aspects of a kd-tree and a BVH. This structure allows interactive raytracing of large numbers of metaballs while still capturing advanced secondary lighting effects such as true mirror reflections, translucencies and soft shadows (cf. Figure 1).

Our method aims at reducing the number of evaluations of the potential function. Our ray-isosurface intersection test works efficiently by evaluating the metaballs potential field at only a few points along the ray. The use of an acceleration structure, such as a BVH, can make our intersection tests even faster by reducing the number of metaballs accounted for. However, a BVH needs specific attention for it to work with metaball rendering, which we detail in Section 4. As successfully done with triangles [EG07, SFD09], we show that a specific Surface Area Heuristic (SAH) and the fitting of the bounding boxes of each node can make the BVH more robust according to the density of the metaballs. Our GPU implementation shows that scenes composed of a large number of metaballs can be rendered at an interactive-to-real-time framerate.

2. Related work

2.1. Density function

A metaball i is a potential field centered at a point p_i called the center of the metaball. This potential field is often described by a density function f_i , which decreases as we move away from p_i . N metaballs define a single potential field f as the sum of the density function of each metaball. Visualizing metaballs actually looks for the isosurface defined by the equation:

$$f(x) = \sum_{i=0}^N f_i(x) - T = 0$$

for a chosen threshold value T . The original potential field is Blinn's blob [Bli82], for which the density function is defined as a Gaussian function, and thus has an infinite support. Other common density functions are piecewise polynomials of the form:

$$f_i(x) = \begin{cases} \left(1 - \frac{\|x - p_i\|^2}{R_i^2}\right)^\alpha & \text{if } \|x - p_i\| \leq R_i; \\ 0 & \text{otherwise.} \end{cases}$$

The larger the value for α , the smoother the surface. These density functions have a finite support, which allows skipping metaballs that are out of the range of any point x when computing $f(x)$. The sphere centered at p_i and of radius R_i is called the *bounding sphere* of metaball i . Our method is presented using this last potential field, but it handles any finite support density function.

2.2. Raytracing metaballs on the GPU

A number of papers have addressed the issue of rendering metaballs using the GPU. Iwasaki et al. [IDYN06] perform surface reconstruction of a particle simulation on GPU.

However, since their method discretizes the surface, it might fail to render high frequency details such as thin splashes in fluid simulations. Van Kooten et al. [vKvdBT07] render metaballs on the GPU using point-based visualization of particles spread uniformly along the isosurface, but their method can miss small objects as well.

Loop and Blinn [LB06] ray cast piecewise algebraic surfaces defined by Bezier tetrahedra on the GPU using Bernstein polynomials. However their method is not really suited for a large number of metaballs as the cost of computing the coefficients of the polynomials at each vertex of the tetrahedra depends on the number of metaballs. More recently, Kanamori et al. [KSN08] efficiently ray cast a large number of metaballs on GPU using depth peeling and Bezier clipping [NN94], but since they rely on rasterization from the image, their method is limited to primary rays, thus restricting their rendering effects to the ones achievable in screen space (for instance, shadow maps or screen space ambient occlusion), contrary to our method which performs true ray tracing.

In order to render point clouds, Wald and Seidel [WS05] first reconstruct a potential field of finite support using a partition of unity, and then perform a kd-tree accelerated ray tracing of an isosurface on the CPU. Our method for rendering metaballs has similarities with their technique: we both use an acceleration structure built greedily over the bounding boxes of the primitives with the use of a SAH, and then try to reduce the size of its nodes in order to reduce the number of intersection tests. Their method for reducing the size of a node consists in slicing that node in thin layers, then checking which layers do not intersect the surface in order to remove them from the node. This last step is done in a Monte-Carlo way, by evaluating the potential field at several samples at the base of each layer. This technique increases drastically the construction time of the kd-tree as hundreds of samples are needed to safely remove a layer. This is not a problem in their context, since they consider static geometries only. The optimizations of our acceleration structure are efficient and enough to achieve interactive rendering of dynamic metaballs. In our context, our intersection test is also more accurate than the ray marching they use.

3. Intersection test

Numerous methods for ray-isosurface intersection have been studied. The Bezier clipping method, developed by Nishita and Nakamae. [NN94], is robust but requires to sort the intersections with the metaballs' bounding spheres along the ray. Sorting these intersections can be efficiently realized for primary rays, thanks to their coherency, by rasterizing the bounding spheres as did Kanamori et al. [KSN08].

Our method for intersecting the isosurface consists in two steps. First it looks for an interval $[a; b]$ along the ray containing only the first intersection. Then it uses the secant method [MKF*04, NMHW02] to narrow this interval until

sufficient precision is reached. The secant method always converges if there is only one intersection in $[a; b]$.

To find such a starting interval, we make the assumption that wherever the ray intersects the isosurface, the potential will be positive at at least one of the points p_{m_i} , corresponding to the projection of the center p_i of each metaball i on the ray. While this assumption is true for most rays, it can be false for very few rays that are nearly tangent to the surface. Fortunately, missing these tangential rays hardly causes any visual artifacts, and denser sets of metaballs create less silhouettes. We then set b as the first point p_{m_i} met by the ray at which the potential is positive and a as the origin of the ray. If $f(p_{m_i}) < 0$ for every i , we consider that our ray does not intersect anything, due to our above assumption.

Once the intersection point has been computed, the normal is found in a standard manner by computing the gradient of the potential field at that point.

4. BVH for rendering metaballs

In order to make the intersection computation more efficient and keep interactive the whole rendering process, we must build an acceleration structure that minimizes the number of metaballs for ray-metaball intersection tests.

A *Metaball Connected Set* (MCS) is defined as a set of metaballs creating a continuous surface (cf. Figure 2). As the metaballs move in the scene, the number and configuration of the MCSs are likely to vary at each frame.

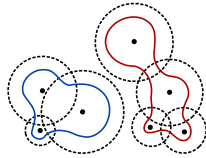


Figure 2: A set of metaballs defining two MCSs in red and blue.

A BVH is an efficient acceleration structure for many primitives, which can be built quickly either on CPU [Wal07] or GPU [LGS⁺09]. However, in our context, a classical BVH built to enclose the MCSs would not work well. Indeed, the metaballs usually define only a few MCSs, unless the metaballs are sparsely distributed, which is not the case for most fluid simulations. Therefore, such a BVH would result in a tree of low depth, whose nodes would each encompass a large number of metaballs.

Instead, we propose to compute the BVH on the metaballs' bounding spheres. However, such a structure does not contain all the necessary data to be able to perform ray-isosurface intersection tests within the nodes themselves, since the surface generated by the metaballs in a node might get modified by metaballs outside that node. We thus modified the BVH to add *split metaballs* to the nodes at the leaves.

The *split metaballs* of a node are the metaballs that do not

belong to the node, but whose bounding spheres intersect the bounding sphere of at least one metaball from that node. One can see the split metaballs of a node as the minimal set of metaballs required to compute any intersection in that node. Thus, storing the indices of those split metaballs in each leaf makes the BVH fully functional and autonomous.

The BVH accelerates intersection tests by reducing the number of metaballs accounted for: when looking for an intersection in a leaf, we explore only the projections p_{m_i} of each metaball i from that leaf and use only the metaballs and split metaballs of the leaf to compute the value of the potential field at any point. Note that it is unnecessary to compute and look for an intersection at the projection p_{m_j} of all split metaballs j , as they belong to other leaves and will be explored when looking for an intersection in those leaves anyway.

4.1. Construction

The modified BVH is built as a classical BVH, from top to bottom. As in [Wal07], at each step of the construction, several ways of splitting the latest node built are browsed using binning. The best split position is then evaluated using a greedy SAH and its cost is compared to the cost of turning that node into a leaf. However, at each step of the algorithm, we keep track of the split metaballs of each node. Each time a node is split into two children, the split metaballs of each child are computed by looking both at the split metaballs of their parent and the metaballs of the other child (cf. Algorithm 1).

Algorithm 1 Construction of a BVH for metaballs

```

1: BUILDNODE(nodeMballs, splitMballs, nodeBbox) {
2:   find best splitting plane  $s$  for nodeBbox using binning
3:   leftChildMballs, rightChildMballs  $\leftarrow \emptyset$ 
4:   leftChildBbox, rightChildBbox  $\leftarrow$  emptyBbox
5:   for each  $i$  in nodeMballs do
6:     if  $p_i$  is at the left of  $s$  then
7:       leftChildMballs  $\leftarrow$  leftChildMballs  $\cup \{i\}$ 
8:       leftChildBbox  $\leftarrow$  leftChildBbox  $\cup$  bbox( $i$ )
9:     else
10:      rightChildMballs  $\leftarrow$  rightChildMballs  $\cup \{i\}$ 
11:      rightChildBbox  $\leftarrow$  rightChildBbox  $\cup$  bbox( $i$ )
12:    end if
13:  end for
14:  leftSplitMballs  $\leftarrow \emptyset$ 
15:  for each  $i$  in splitMballs  $\cup$  rightChildMballs do
16:    if  $i$  overlaps the bounding sphere of any  $j \in$  leftChildMballs then
17:      leftSplitMballs  $\leftarrow$  leftSplitMballs  $\cup \{i\}$ 
18:    end if
19:  end for
20:  BUILDNODE(leftChildMballs, leftSplitMballs, leftChildBbox)
21:  execute the instructions 14 – 20, but for the right node }

```

5. Fitted-BVH

As the leaves of the modified BVH can overlap each other, the same intersection can be computed several times during ray traversal. This becomes an issue when dealing with dense sets of metaballs as the leaves grow bigger, therefore increasing the cost of computing a ray-isosurface intersection.

The Fitted-BVH (FBVH) reduces the number of redundant intersection tests during the traversal of the structure. When two nodes A and B of the modified BVH overlap each other, most of the metaballs from A whose bounding spheres intersect the overlapping volume $A \cap B$ are duplicated as split metaballs in B . Therefore, during ray traversal, the surface generated by those metaballs can be intersected several times as stated earlier.

Like a BVH, an FBVH is a tree in which the bounding boxes of two child nodes define a partition of the surface encompassed by their parent node. However, contrary to a BVH, the two child nodes cannot intersect each other (cf. Figure 3). Therefore, if we explore the leaves in order of traversal, we can stop as soon as an intersection is found, thus minimizing any redundant test.

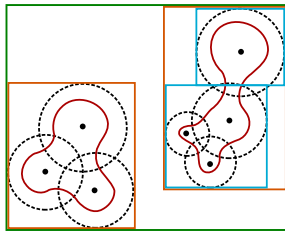


Figure 3: An FBVH is a surface partition tree where two child nodes cannot overlap each other.

Since the bounding boxes of two child nodes cannot intersect each other anymore, they no longer correspond to the bounding boxes of the metaballs they contain. Instead, they determine a region of space that contains a portion of the surface generated by the metaballs and split metaballs of the nodes. Therefore, there is not much difference between the metaballs and the split metaballs of a node: the union of both determines the set of metaballs whose bounding spheres intersect the node. However, we will continue to separate these two sets as it makes the construction of the FBVH more convenient.

Actually, the main difference with a kd-tree is the presence of a bounding box for each node, which enables a tight encompassing of the metaballs bounding spheres when they are sparse enough. This is of prime importance here as the cost of intersecting a leaf is high (cf. Section 5.1.1). Such an encompassing can be done with a kd-tree using techniques like empty space cutoff implemented using our fitting techniques (cf. Section 5.1.2) and our optimizations (cf. Section

5.3). However it would require additional nodes (up to 5 additional nodes to remove all empty space around the bounding box of a node) that would create additional branchings during the traversal of the structure. Those branchings are detrimental when using SIMD architectures such as a GPU, since they may reduce instruction coherency. This makes us believe that our FBVH should be a little faster than such a kd-tree in this context, even though we did not actually implement one. The difference of performance is probably small anyway as the cost of intersecting a leaf is much higher than the one of the ray traversal.

5.1. Construction

5.1.1. Surface Area Heuristic for metaballs

SAH is essential to build the acceleration structure in a fast, greedy way. However, it has to be carefully designed to ensure good performance during ray traversal. The usual cost considered when splitting a node N into two nodes A and B is

$$C_{\text{split}}(A, B) = \rho_A \times C_A + \rho_B \times C_B$$

where C_i is the cost of computing an intersection in node i , and ρ_i is the probability of having a ray intersect node i knowing that its parent node has been intersected. $C_{\text{split}}(A, B)$ is compared to the cost of leaving N as a leaf.

ρ_i is estimated as the ratio A_i/A_N of the surface area of the bounding box of node i over the surface area of the bounding box of its parent node N . Usually, the cost of intersecting a surface generated by a primitive is the same in each node, so C_i is in $\mathcal{O}(n)$ where n is the number of primitives in node i (we can even set $C_i = n$ if all primitives are of the same type). This is not the case for metaballs though, as the cost of evaluating the potential field depends on the number of metaballs in the current leaf.

When looking for an intersection in a leaf, we compute the projections on the ray of every center of the leaf's metaballs. The potential field is evaluated at those projection points, and each of these evaluations is done in $\mathcal{O}(n)$ operations where n is the number of metaballs in the leaf. Thus this base interval finding step is done in $\mathcal{O}(n^2)$ operations.

Starting from this analysis, our cost function should be $C_i = n^2 + \lambda \times n$ where n^2 corresponds to the base interval finding and the $\lambda \times n$ to the secant method. The value for constant λ is difficult to determine, as it depends on the number of iterations needed by the secant method to converge. However, it is clear that n^2 is the dominant term for leaves composed of many metaballs (which is the case for dense sets of metaballs). Thus we can as well set $\lambda = 0$ in those cases. In our experiments, neglecting $\lambda \times n$ and setting $C_i = n^2$ did not have much impact in terms of rendering speed in the case of sparse metaballs either.

5.1.2. Algorithm

To build the FBVH, we proceed as with the modified BVH. However, when splitting a node, we compute a first bounding

box for each child by splitting the node's bounding box at the split plane (cf. Figure 4 (a)). The metaballs from the node are distributed to either the left or the right child according to the splitting plane. Then, any metaball from the node's split metaballs or the right child's metaballs that overlaps the left child node is duplicated in its split metaballs, and the same is done for the right child node (cf. Algorithm 2).

We then try to fit the bounding box of each child node to its primitives (hence the *fitted*-BVH). This step is crucial to achieve good performance as the cost of intersecting a leaf is high ($\mathcal{O}(n^2)$, see Section 5.1.1). In order to do so, we compute the bounding box of the metaballs and split metaballs for each child (cf. Figure 4 (b)). Each resulting bounding box is then intersected with its previous bounding box (cf. Figure 4 (c)).

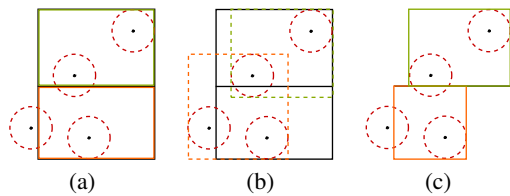


Figure 4: When splitting a node, the bounding boxes of the metaballs and split metaballs of each child are computed (b). Intersecting them with the former bounding boxes of each child (a) gives their new bounding boxes (c).

Algorithm 2 Construction of an FBVH for metaballs

```

1: BUILDNODE(nodeMballs, splitMballs, nodeBbox) {
2:   find best splitting plane  $s$  for nodeBbox using binning
3:   leftChildMballs  $\leftarrow \emptyset$ , rightChildMballs  $\leftarrow \emptyset$ 
4:   leftChildBbox  $\leftarrow$  nodeBbox.splitLeftAlong( $s$ )
5:   rightChildBbox  $\leftarrow$  nodeBbox.splitRightAlong( $s$ )
6:   leftMballBbox  $\leftarrow$  emptyBbox
7:   rightMballBbox  $\leftarrow$  emptyBbox
8:   for each  $i$  in nodeMballs do
9:     distribute  $i$  to either the left or right child node and
       update left and right metaball bounding boxes accord-
       ingly as in Algorithm 1
10:  end for
11:  leftSplitMballs  $\leftarrow \emptyset$ 
12:  for each  $i$  in splitMballs  $\cup$  rightChildMballs do
13:    if  $i$  overlaps leftChildBbox then
14:      leftSplitMballs  $\leftarrow$  leftSplitMballs  $\cup$   $\{i\}$ 
15:      leftMballBbox  $\leftarrow$  leftMballBbox  $\cup$  bbox( $i$ )
16:    end if
17:  end for
18:  leftChildBbox  $\leftarrow$  leftChildBbox  $\cap$  leftMballBbox
19:  BUILDNODE(leftChildMballs, leftSplitMballs, leftChildBbox)
20:  execute the instructions 11 – 19, but for the right node }

```

5.2. Ray traversal

The ray traversal is not much different from the one of a BVH. When a ray intersects a node, we load the bounding boxes of both child nodes and compute which one is intersected first (if any). The first one to be intersected is then traversed first. As soon as an intersection is found, we can stop the traversal since other leaves can only result in an intersection farther away.

The intersection computation in a leaf is a little different than when using a BVH. Since the split metaballs of the current leaf are not fully encompassed within another leaf, their projection along the ray has to be computed and checked as well when looking for an interval containing the first root of the potential field.

5.3. Optimizations

The bounding boxes of the nodes, built from the bounding spheres of their metaballs, might not fit as tightly to the surface they encompass after the fitting step described in Section 5.1.2. We propose a modification to the computation of the bounding box of a set of metaballs that leads to better fitted nodes. We also observed that some of the split metaballs of the nodes do not contribute to the surface encompassed by that node and propose a technique for discarding them.

5.3.1. Bounding box computation

When computing the bounding box of some metaballs, we usually compute the smallest bounding box which encompasses every metaballs' bounding sphere. However, the surface generated by the metaballs might not be so close from the bounding spheres (cf. Figure 5). Thus it is possible to compute a tighter bounding box, which would ensure better performance by reducing the number of intersection tests in each leaf.

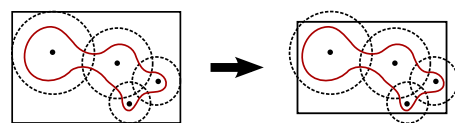


Figure 5: The bounding box encompassing the bounding spheres of the metaballs might not be the tightest one around the surface itself.

A way to do so is to consider that the largest sphere generated by the metaballs is obtained when every metaball of the leaf shares the same center, creating a sphere of radius r such that

$$\sum_{i=1}^n f_i(r) = T \quad \text{where} \quad f_i(r) = \left(1 - \frac{r^2}{R_i^2}\right)^\alpha$$

in our case. This sum is bounded by $n \times f_K(r)$ where K is the metaball with the largest bounding sphere (of radius R_K)

in the node. Thanks to this, we can easily compute an upper bound for r :

$$r_{\max} = R_K \times \sqrt{1 - \left(\frac{T}{n}\right)^{\frac{1}{\alpha}}}.$$

Therefore, $r_i = \min(R_i, r_{\max})$ can be used as the radius of metaball i 's bounding sphere when computing the bounding box of the metaballs.

5.3.2. Discarding non contributing split metaballs

The split metaballs of a node have bounding spheres overlapping their node's bounding box. However, some of the split metaballs might belong to an MCS which does not overlap the node, i.e., they do not contribute to any surface in that node (cf. Figure 6). Therefore, storing them as split metaballs is useless and harmful, as they increase computation time for the intersection tests in the node.

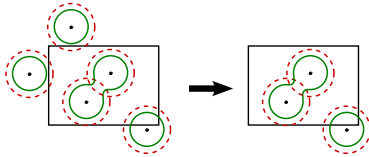


Figure 6: Some of the split metaballs whose bounding sphere (in red) intersects a node do not generate a surface (in green) intersecting that node. Storing them in that node is useless. Note that any metaball participating to the surface in a node has to be stored in that node.

Identifying such split metaballs that can thus be removed is a difficult problem. However, some conservative tests can be applied. Given split metaball i , we test:

- that the sphere centered at p_i and of radius r_{\max} (see above) does not overlap the node's bounding box. This ensures the MCS containing metaball i does not intersect the node.
- that for any other metaball j of the node,

$$\|p_i - p_j\| \geq R_i + r_{\max}.$$

This ensures metaball i does not interfere with the surface generated by any other metaball j of the node.

If both those tests succeed, the metaball can be safely discarded from the split metaballs.

6. Results

We wrote our implementation in C++ and CUDA. We perform the FBVH construction on the CPU, while ray traversals and intersection computations are done on the GPU. This allows us to compute the FBVH of the next frame on the CPU while the current frame is being rendered on the GPU, thus hiding its computation cost. In order to benefit from cache for data access, metaballs and node data are stored in

the GPU texture memory. Traversal on the GPU is done using a stack stored in local memory. Animation of metaballs is done on the GPU, then transferred to the host memory. We use streaming so that memory transfers between CPU and GPU can be hidden with computations (cf. Figure 7).

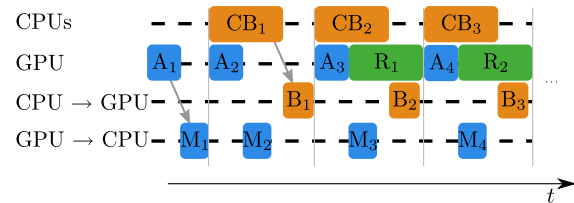


Figure 7: Runtime scheme. Animation of the metaballs (A_i , in blue) is done on the GPU, then transferred to the host (M_i). The host computes the BVH or FBVH (CB_i , in orange), then transfers it to the device memory (B_i). Rendering (R_i , in green) is done on the GPU while the CPU computes the acceleration structure for the next frame. Most of the memory transfers are hidden thanks to streaming.

Our configuration test is a PC with an intel Core 2 E6600 (2.66 GHz) and an Nvidia GTX 280. We compared the FBVH to the BVH in various scenes (cf. Table 1 and Figure 8). In most scenes, we observed a significant gain of performance over the BVH, especially with those scenes containing dense sets of metaballs. In these cases, such as the Pool scene, the FBVH can lead to more than 3 times faster renderings. The difference of performance with the BVH decreases with the density of metaballs, but as long as the metaballs are not too sparse, the FBVH still outperforms the BVH. It is also worth considering that while the times of construction of both structures are of the same magnitude, the FBVH is usually a little faster to build. This can seem surprising, as it could have been expected that fitting the bounding boxes of each node would result in a larger amount of computations. Actually, even if the bounding boxes of the nodes have a few more steps in their construction, their reduced sizes lead to fewer split metaballs to track during the construction of the structure, hence the speedup.

Considering equivalent parameter values (such as metaball radius, implicit surface polynomial degree, etc.) and hardware as those used in [KSN08], we estimate that our raytracer (acting as a ray caster in this context) would be about 30% slower than their rasterization-based method. This is a good performance for a pure raytracer that, by nature, is not limited to primary ray effects. Moreover, in our implementation, the cost of additional non-primary rays can be quite low (cf. Table 2).

We measured the average number of node traversals, intersection tests, and the average number of metaballs considered in these intersection tests per ray (cf. Table 3). As one can see, the FBVH does a clearly better job at culling unnecessary nodes and metaballs.

The effect of using the adapted SAH n^2 instead of the

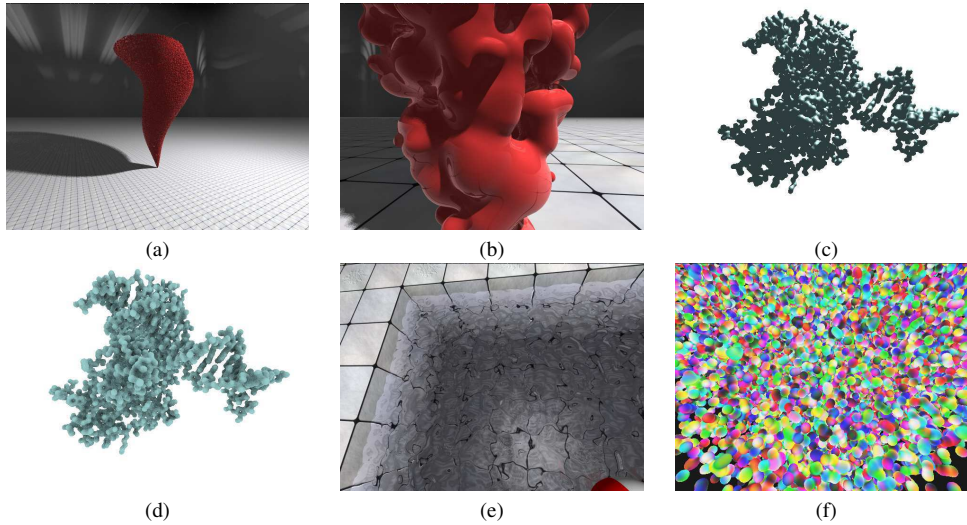


Figure 8: Screenshots of the different test scenes: (a,b) the Tornado scene, rendered using three levels of reflection; (c) the Molecule 1 scene, rendered with Phong shading; (d) the Molecule 2 scene, rendered with ambient occlusion using 64 shadow rays per pixel; (e) the Pool scene, rendered using Fresnel reflectance; (f) the Scattered metaballs scene (pure ray casting). All these scenes are rendered at a 640×480 resolution in our tests.

| Scene | number of metaballs | modified BVH | | | | FBVH | | | |
|---------------------|---------------------|--------------------------|-----|------|-----|--------------------------|-----|------|------|
| | | avg time of construction | fps | | | avg time of construction | fps | | |
| | | | min | max | avg | | min | max | avg |
| Scattered metaballs | 30,000 | 101 ms | 4.6 | 12.4 | 7.4 | 79 ms | 4.9 | 12.5 | 9.4 |
| | 100,000 | 160 ms | 1.8 | 3.9 | 3.3 | 150 ms | 2.3 | 9.1 | 4.3 |
| Tornado | 4,000 | 8.0 ms | 3.6 | 5.4 | 4.6 | 5.5 ms | 4 | 14.5 | 11 |
| | 128,000 | 500 ms | 0.6 | 1.1 | 1.0 | 290 ms | 1.9 | 2.6 | 2.4 |
| Molecule 1 | 5,440 | 8.9 ms | 3.2 | 8.6 | 5.8 | 9.9 ms | 7.5 | 24.1 | 17.7 |
| Molecule 2 | 5,440 | 8.9 ms | 0.5 | 0.7 | 0.6 | 9.9 ms | 1.2 | 1.4 | 1.3 |
| Pool | 10,000 | 41 ms | 1.5 | 2.3 | 1.9 | 40 ms | 4.1 | 11.0 | 7.1 |

Table 1: Rendering time in fps of various scenes containing metaballs using an acceleration structure (cf. Figure 8). The construction of the acceleration structure is completely hidden by the rendering in most cases.

| Scene | Effects (avg fps) | | | |
|----------|-------------------|--------------|--------------------------------------|---------------------------------------|
| | Ray cast | Hard shadows | Hard shadows + 1 level of reflection | Hard Shadows + 3 levels of reflection |
| Molecule | 22.3 | 17.7 | 12.7 | 8.2 |
| Tornado | 5.7 | 4.9 | 3.4 | 2.5 |

Table 2: Performance of our algorithm with secondary effects.

standard one n provides much better performance overall. In our tests, the Pool scene gains 40% more fps using the adapted SAH (5.1 fps with the standard SAH, 7.1 fps with ours). Our SAH leads to a tree that is better suited for our intersection test than the standard one, hence the better performance.

We also measured the effects of fitting the bounding boxes

| | Number of traversals | Number of intersection tests | No. metaballs considered |
|------|----------------------|------------------------------|--------------------------|
| BVH | 9.3 | 1.2 | 83.5 |
| FBVH | 6.2 | 0.5 | 35.1 |

Table 3: Average number of traversals, intersection tests, and metaballs considered per ray on the Tornado scene.

of the nodes to their metaballs, as explained in Section 5.1.2. To this effect, we omitted the fitting step during the construction of the FBVH so that each node's bounding box was just the result of splitting the bounding box of its parent. As one can see in Table 4, the number of intersection tests avoided by the fitting step clearly leads to better performance. This gain of performance increases with the density of metaballs, since the cost of computing an intersection in a leaf increases with the number of metaballs in that leaf. Therefore, it is a

worthy improvement since it does not impact the FBVH construction cost too much.

| Scene | Without the fitting step | | With the fitting step | |
|------------|--------------------------|---------|-----------------------|---------|
| | build time | avg fps | build time | avg fps |
| Molecule 1 | 8.7 ms | 12.7 | 9.2 ms | 14.6 |
| Pool | 35 ms | 3.2 | 37 ms | 6.5 |

Table 4: Benefits of fitting the nodes' bounding boxes after a split during the construction of the FBVH.

| Scene | Optimizations (avg fps) | | | |
|------------|-------------------------|--------------|----------------------|------|
| | None | Tighter Bbox | Discarding metaballs | Both |
| Molecule 1 | 14.6 | 15.3 | 16.7 | 17.7 |
| Pool | 6.5 | 6.7 | 6.8 | 7.1 |

Table 5: Benefits of the optimizations.

The optimizations presented in Section 5.3 bring a noticeable performance improvement (cf. Table 5), but their impact decreases as the density of the metaballs increases. This was to be expected, as the value of r_{max} increases with the number of metaballs overlapping each node. Therefore, it might be preferable to skip these optimizations in the leaves containing more than a dozen metaballs, as their cost increases with the number of metaballs (especially the discarding of the split metaballs), and since they can drastically increase the time of construction of the FBVH.

7. Conclusion and future work

We have presented a new acceleration structure for the rendering of metaballs, that supports advanced secondary effects. The resulting speedup enables the visualization of a large number of metaballs at interactive-to-real-time framerates on the GPU. The FBVH provides a consequent speedup compared to the BVH in the case of dense sets of metaballs and can be built quickly on the CPU using our SAH.

However, for extremely dense sets of a large number of metaballs, the construction time of the FBVH might become an issue, as there is a lot of split metaballs to track. In those cases, for the first levels of the tree, we do not use binning to find the best splitting plane for each node but simply split the nodes along their longest axis. This does not affect the quality of the tree too much and provides a good speedup.

Another issue is the time of computing a ray-isosurface intersection in leaves that contain many metaballs. In these cases, the leaves usually contain a single MCS whose surface is very close to the leaves' bounding boxes. We will work on approximations to accelerate intersection with the isosurface in these cases.

References

- [Bli82] BLINN J. F.: A generalization of algebraic surface drawing. *ACM Trans. Graph.* 1, 3 (1982), 235–256.
- [Blo97] BLOOMENTHAL J.: *Introduction to Implicit Surfaces*. Morgan Kaufmann, August 1997.
- [EG07] ERNST M., GREINER G.: Early split clipping for bounding volume hierarchies. In *RT '07: Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing* (2007), pp. 73–78.
- [Flo08] FLOREZ J.: *Improvements In The Ray Tracing Of Implicit Surfaces Based On Interval Arithmetic*. Ph.d. thesis, Department of Electronics, Computer Science and Control University of Girona, November 2008.
- [Har93] HART J. C.: Ray tracing implicit surfaces. In *SIGGRAPH 93 Course Notes: Design, Visualization and Animation of Implicit Surfaces* (1993), pp. 1–16.
- [IDYN06] IWASAKI K., DOBASHI Y., YOSHIMOTO F., NISHITA T.: Real-time rendering of point based water surfaces. In *Computer Graphics International 2006* (June 2006), pp. 102–114.
- [KHK*09] KNOLL A., HIJAZI Y., KENSLER A., SCHOTT M., HANSEN C., HAGEN H.: Fast Ray Tracing of Arbitrary Implicit Surfaces with Interval and Affine Arithmetic. In *Computer Graphics Forum* 28 (2009), no. 1, pp. 26–40.
- [KSN08] KANAMORI Y., SZEGO Z., NISHITA T.: GPU-based fast ray casting for a large number of metaballs. *Computer Graphics Forum (Proc. of Eurographics 2008)* 27, 3 (2008), 351–360.
- [LB06] LOOP C., BLINN J.: Real-time gpu rendering of piecewise algebraic surfaces. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (2006), ACM, pp. 664–670.
- [LGS*09] LAUTERBACH C., GARLAND M., SENGUPTA S., LUEBKE D., MANOCHA D.: Fast BVH construction on gpus. *Comput. Graph. Forum* 28, 2 (2009), 375–384.
- [MKF*04] MARMITT G., KLEER A., FRIEDRICH H., WALD I., SLUSALLEK P.: Fast and Accurate Ray-Voxel Intersection Techniques for Iso-Surface Ray Tracing. In *Vision, modeling, and visualization 2004 (VMV-04)* (2004), pp. 429–435.
- [NMHW02] NEUBAUER A., MROZ L., HAUSER H., WEGENKITTTL R.: Cell-based first-hit ray casting. In *VISSYM '02: Proceedings of the symposium on Data Visualisation* (2002), Eurographics Association, pp. 77–86.
- [NN94] NISHITA T., NAKAMAE E.: A method for displaying metaballs by using bézier clipping. *Comput. Graph. Forum (Proc. of Eurographics '94)* 13, 3 (1994), 271–280.
- [SFD09] STICH M., FRIEDRICH H., DIETRICH A.: Spatial splits in bounding volume hierarchies. In *HPG '09: Proceedings of the Conference on High Performance Graphics 2009* (New York, NY, USA, 2009), ACM, pp. 7–13.
- [Ura06] URALSKY Y.: Dx10: Practical metaballs and implicit surfaces. In *Game Developers Conference* (2006).
- [vKvdBT07] VAN KOOTEN K., VAN DEN BERGEN G., TELEA A.: Point-based visualization of metaballs on a gpu. In *GPU GEMS 3*, Nguyen H., (Ed.). Addison-Wesley, 2007, ch. 7.
- [Wal07] WALD I.: On Fast Construction of SAH based Bounding Volume Hierarchies. In *Proceedings of the 2007 Eurographics/IEEE Symposium on Interactive Ray Tracing* (2007).
- [WS05] WALD I., SEIDEL H.-P.: Interactive Ray Tracing of Point Based Models. In *Proceedings of 2005 Symposium on Point Based Graphics* (2005).
- [WT90] WYVILL G., TROTMAN A.: Ray-tracing soft objects. In *CG International '90* (1990), pp. 469–476.

7.3.4 Implicit Blending Revisited.

Auteurs : Adrien Bernhardt, Loïc Barthe, Marie-Paule Cani, Brian Wyvill
Revue : Computer Graphics Forum, Vol 29(2), Proceedings of Eurographics
Date : Mai 2010

Implicit Blending Revisited

Adrien Bernardt¹, Loïc Barthe², Marie-Paule Cani¹, Brian Wyvill³

¹Université de Grenoble, CNRS (Laboratoire Jean Kuntzmann), INRIA Grenoble Rhône-Alpes, France

²IRIT-VORTEX, Université Paul Sabatier, Toulouse, France

³Department of Computer Science, University of Victoria, Canada

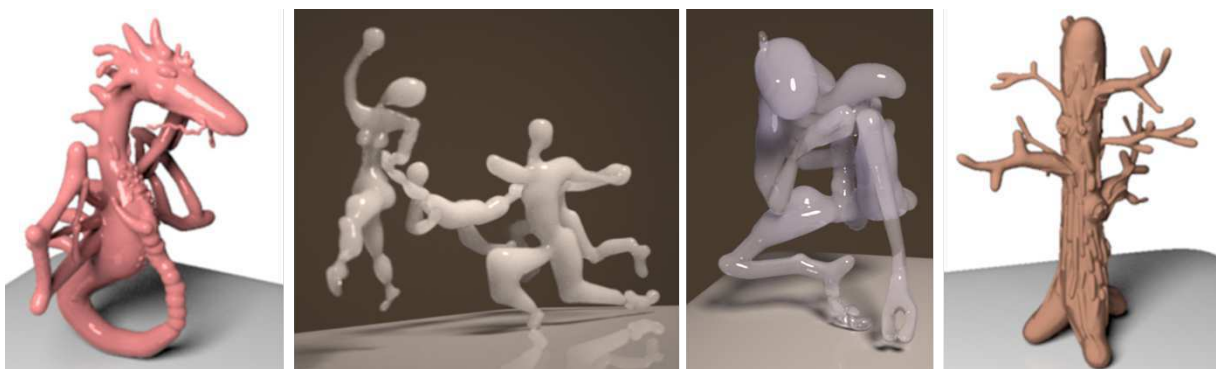


Figure 1: Models generated interactively using a new approach to implicit blending. Note the capability of the shapes to come close to each other without blending. From left to right : The dragon, a dancer loop, the alien, a tree.

Abstract

Blending is both the strength and the weakness of functionally based implicit surfaces (such as *F-reps* or soft-objects). While it gives them the unique ability to smoothly merge into a single, arbitrary shape, it makes implicit modelling hard to control since implicit surfaces blend at a distance, in a way that heavily depends on the slope of the field functions that define them. This paper presents a novel, generic solution to blending of functionally-based implicit surfaces: the insight is that to be intuitive and easy to control, blends should be located where two objects overlap, while enabling other parts of the objects to come as close to each other as desired without being deformed. Our solution relies on automatically defined blending regions around the intersection curves between two objects. Outside of these volumes, a clean union of the objects is computed thanks to a new operator that guarantees the smoothness of the resulting field function; meanwhile, a smooth blend is generated inside the blending regions. Parameters can automatically be tuned in order to prevent small objects from blurring out when blended into larger ones, and to generate a progressive blend when two animated objects come in contact.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object modelling—Solid, surface and object representation; Constructive solid geometry

1. Introduction

One of the major strength of implicit surfaces has been the capacity for objects to blend at a distance. By blending we mean that a new, seamless surface can be generated by smoothly merging two input volumes. The new surface is calculated by combining the field functions of the input objects and computing a new iso-surface. This

feature has been popular in both Computer Animation to animate topological changes (e.g. liquids, melting objects [TPF89, DG95, BGOS06]) and in modelling where a constructive approach is used to assemble object components [PASS95, WGG99, SPS04, BPCB08, dGWvdW09].

The main problem with functionally-based implicit blending is its rather unpredictable nature: implicit objects may

blend at distance, or in regions where it is not required. For instance, in simple skeleton-based implicit modelling systems (such as blobby models, meta-balls, soft objects or convolution surfaces), the blend shape is computed by taking the sum of the field functions that define the input implicit primitives. Therefore, the amount and range of blending heavily depends on the slopes of these functions, without any intuitive user control. This results in the blurring of small, sharp features when blended into large, smooth surfaces [WW00]. Moreover, this is a major problem in Computer Animation, since implicit objects that come close together will start to deform and blend at a distance; for example a falling drop of water deforms the water surface before reaching it. Lastly, this method leads to blending in unexpected regions; for instance the hand of a character will blend with its body when it comes close enough, due to the blend allowed in the shoulder region.

Although improving control and localizing blends has been attempted for years through the definition of various ways to combine field functions (see section 1.2), no simple, generally applicable solution has emerged. This may be the main reason for the relatively small spread of implicit techniques among the set of practical tools used by modelling and animation professionals.

We describe an automatic way of defining a blend between arbitrary functionally-based implicit surfaces. Contrary to previous methods, it does not allow objects to blend at a distance, but rather automatically localizes the blend near the regions where their surfaces intersect, while the extent of the blend is controlled independently of the support size of the defining field functions. This method is generic in the sense that our solution supports both local and global support implicit primitives.

1.1. Technical background

An implicit surface is defined as a set of points verifying an equation of the type $f(P) = C$ where f is a field function which can either be of global or local support. Other terminologies have been used for these functions such as *potential fields* in the case of skeleton-based implicit surfaces or *implicit functions*, which is not mathematically correct since the function is explicitly defined. We prefer *field function*, which expresses the fact that the implicit surface is an iso-surface of the scalar field they define. The field function can be defined either with a level-set, i.e. as a front propagation in a grid [Set99, OF02], or by a functionally-based representation such as f-reps [PASS95], soft objects [WMW86], convolution surfaces [She99], RBF [SPOK95], and MLS [Lev03].

For these representations, several conventions can be used: Most **global support** field functions used in solid modelling behave as a distance to the surface of interest: they are negative inside the object, and positive outside, C being equal to zero. In contrast many implicit modelling techniques rely on a **skeleton-based scalar field** (from Blinn's

objects to the most recent convolution surfaces): in which case the field function is a decreasing function of the distance to a geometric primitive called the skeleton. The field values are larger inside than outside and C is non-zero. **Local support implicit surfaces** such as meta-balls and soft objects belong to that group, with fields that vanish at a given distance to the skeleton. All these representations have specific properties that lead to different composition operators for union, difference, blending, etc. For a full explanation please see [BWd04]. This paper does not focus on discretely sampled distance fields, like level sets, but rather on the blending of functionally based implicit surfaces.

1.2. Previous work

Over the years, improving the control of implicit blends and localizing them (i.e. solving the *unwanted blending problem*) has been a major area of research in implicit modelling and animation. While efficient solutions have been proposed for level set implicit surfaces [MBWB02, BMPB08, EGB09], no fully satisfactory technique has been developed for functionally-based implicit surfaces. Up to now, two families of solutions arose: The first ones used a blending graph to define pairs of primitives allowed to blend, i.e. to sum their field functions. Since the early solutions [GW95, DG95] did not insure that the resulting shape was continuous everywhere or could lead to sudden shape changes during animation, more elaborate methods were developed [CH01, AC02]. In particular, Angelidis [AC02] added decay functions to skeletal elements to insure the smoothness of the resulting shape, making the method quite intricate to implement and restricting the method to the specific case of convolution surfaces generated by skeletons made of line-segments and triangles.

A second group of methods developed more powerful ways than simple sums to combine scalar fields. In his seminal paper on set-theoretic R-functions [PASS95], A. Pasko introduced, in the framework of the global support surfaces used in solid modelling, Rvashev's R-function union operator that generates at the same time, the exact (sharp) union surface and a C^1 continuous field function everywhere else. In the remainder of this paper, we will call such an operator a *clean union* since it ensures that no unexpected crease appears during subsequent blends [BWd04]. More recently G. Pasko [PPK05] proposed a method for defining local smooth blends between some parts of two objects, while using a clean union elsewhere. This was done through the specification of a simple user-defined implicit primitive (sphere, ellipse, torus) of local support, indicating the blending region. The field function of the primitive was used to control the amount of blending. This method was designed for global support implicit surfaces and required manual user intervention to tune each blend.

In parallel, Barthe [BDS*03, BWd04] developed controllable ways to blend field functions. The method, first introduced for global support implicit surfaces, offered intuitive shape parameters and enabled the accurate location of

the blend between two given iso-values of the input implicit functions. However this does not allow some parts of the surfaces to come close without blending.

Barthe later extended his operator for soft objects (of local support) and defined the first clean union. However, his solution based on arcs of ellipses is computationally very costly, and thus not practical in an interactive modelling system. Lastly, the smooth blend operator was used in a sketch-based modelling system [BPCB08], where blends were artificially localized by re-computing only a part of the iso-surface. This method could lead to artefacts during subsequent blends due to the lack of implicit representation for the whole surface.

1.3. Overview and contributions

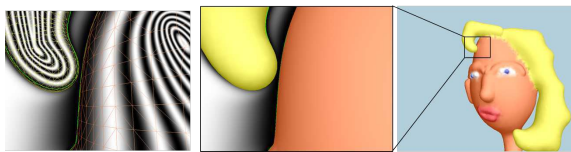


Figure 2: Implicit primitives do not interfere with each other when they come close without intersection. The field function (left) remains C^1 continuous providing a clean union.

The key idea of our method is that to be predictable and intuitive, blends should only occur where two objects overlap. Implicit primitives should not interfere with each other elsewhere, even if they come close (as illustrated in Figure 2). As suggested by Museth et al. [MBWB02] in the case of level sets, we localize blends automatically near intersections, by computing the intersection curve(s) between the two input surfaces, and use them as a skeleton to generate the region(s) where the objects are allowed to blend (see Figure 3). The thickness of the blending region is set to produce a progressive growth of the blend during animation.

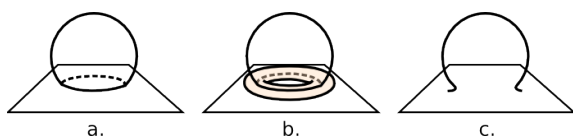


Figure 3: Overview of revisited blending: a. intersection curve extraction; b. blending region generated by the curve. c. local blending inside the blending region while a clean union is used elsewhere.

To achieve this, our contributions are to solve the 3 following issues:

- Firstly, we need an efficient yet precise method to compute the intersection curves between implicit objects and to define appropriate blending volumes around them.
- Secondly, we need an efficient clean union operator to be used outside of the blending volumes, whatever the nature of the implicit surface. While set theoretic R-functions [PASS95] provide a solution in the case of global support field functions, a more efficient operator

than Barthe's arc-of-an-ellipse union [BWd04] is required for local support field functions.

- Lastly, the smooth blend that takes place within the blending volume should automatically prevent the blur of small shape features when blended to larger shapes and ensure that the resulting field function is continuous across the border of the blending volumes.

Sections 2 to 4 respectively present our solutions to these problems. Section 5 illustrates our results on simple animation experiments and within an interactive modelling system where the user progressively creates simple shapes and blends them together. We then discuss the strengths and weaknesses of our approach.

2. Local blending volumes around intersections

Instead of asking the user to specify the region where each pair of surfaces should blend, we would like to blend them automatically where they intersect and just combine them with a union elsewhere. To achieve this, *blending volumes* must be automatically defined around each intersection. Since the field value of the blending volume will be used as an interpolation parameter between smooth blend and union, we need an implicit primitive of local support (see section 1.1) i.e. whose field function falls to zero at the border of the volume.

The intersection between two closed surfaces is a set of (possibly several) closed curves. See Figure 4. The idea is to use them as skeletons, in order to build a ring-shaped blending volume around each intersection. In the next subsection we discuss our method for an efficient, yet precise extraction of intersection curves. We then explain how the blending volumes around them are parameterized, with the goal of providing intuitive, progressive blends when animated objects come in contact.

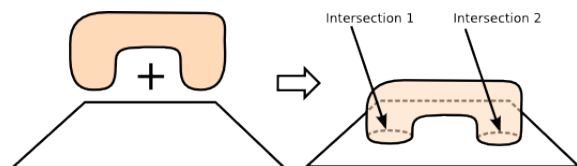


Figure 4: Dealing with multiple intersections.

2.1. Extracting intersection curves

The intersection curves between a pair of implicit surfaces may be described in two ways: as (possibly several) polylines, i.e. explicit lists of points at some approximation level, or defined analytically as a single implicit curve, i.e. as the set of points P simultaneously verifying:

$$f_1(P) = C \text{ and } f_2(P) = C \quad (1)$$

where f_1 and f_2 are the field functions of the two input objects and C is the iso-value at which the surface is computed.

The goal is to use intersection curves as skeletons, which

generate implicit volumes, i.e. to enable quick yet precise distance queries from arbitrary points in space to the curves. In [MBWB02], the intersection curves computation relies on the discrete grid representation of the potential field. For functionally-based field functions, such a discrete representation is not directly available. In the first case, a distance query can be made efficient by embedding the poly-line into a tree of bounding boxes, but the resolution being fixed, precision may be lacking. In the second case, precision is good since the curve is analytically defined, but distance queries may not be efficient, since a numeric technique must be used to find the closest curve point.

To get both precision and efficiency, we use a hybrid approach: We first extract low resolution versions of the curves, and we locally refine each curve by splitting segments in which the mid-point does not meet the constraint presented in Equation 1. The extra points are made to converge on the curve using the algorithm described in [WvO96], i.e. by alternately marching in the two field gradient directions.

The extraction of the low resolution curve can be done in one of two ways. Firstly, by computing the intersection of the meshes used to display the input implicit surfaces; this is done in an efficient way using hierarchies of bounding boxes. The second approach uses the efficient marching-faces algorithm, introduced in [LY03]: A marching-cubes is performed on the set of voxels embedding surface intersections in order to locally polygonized each surface. The marching-faces algorithm then computes the intersection lines of polygons in each voxel. This second method is more general, since it does not require any precomputed meshing of the input surface. The selection of the voxels embedding the intersection curve required by this last technique can be greatly accelerated using an adaptive octree in the intersection of the combined objects bounding boxes. The octree splitting is guided by the well designed inclusion functions of f_1 and f_2 [FSSV06, Duf92, Sny92], which provide intervals that are close to optimal.

The method produces a fast computation of explicit yet precise curves. In the case of multiple intersection volumes, our method has the advantage of separating the intersection into disjoint curve primitives, whereas the analytical description would not. This allows us to generate blending volumes of appropriate size around each individual curve, as explained next.

2.2. Setting bounding volume parameters

At this point, the intersection is described as one or several closed curve(s). We build an implicit blending volume around each of them by using a finite support skeleton-generated field function, such as Wyvill's soft object function [WMW86] used in our current implementation. However, a parameter needs to be set: the thickness of the ring-shaped blending volumes, defined as the size of the support of the associated local field function.

As already mentioned, we are looking for a method which

makes blending intuitive in both modelling and animation applications. In particular, two animated implicit shapes that come into contact need to blend progressively rather than jump immediately to a fully deformed shape (see Figure 5).

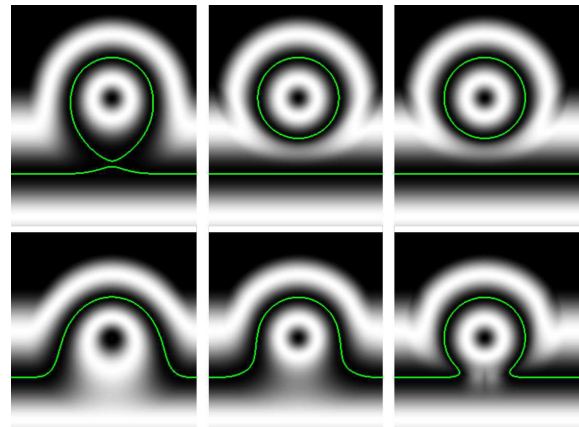


Figure 5: 2D view, with scalar field represented in grey levels, of a drop of water reaching the water surface. Left: standard blending, e.g. a sum or Barthe's operator [BWd04]. Center: our method with a constant radius blending volume: an instant large blend is generated when the drop touches the water. Right: our method with radius function of the size of intersection solves this problem by generating a continuously growing blend.

This feature can be expressed as a constraint on the thickness of the blending volume. Our experience has shown that a suitable heuristic is that the thickness should be proportional to the diameter of the associated intersection curve; defined as the largest distance between curve points. In addition to enabling animation of progressive blends, this automatic parameter setting is quite intuitive in interactive modelling sessions, where the part of the object that penetrates the most will also be those that will have the largest region of smooth blend. If the result is not satisfactory the user can manually edit the thickness parameter of a blending volume, although in our initial experiments this has not proved necessary.

3. Efficient, clean union of implicit surfaces

This section describes the way we combine field functions outside of the blending volumes where a union operator is required to prevent unwanted blending effects. However, in order to prevent artefacts during subsequent smooth blends in the area, this operator needs to be a clean-union i.e. to generate a C^1 continuous field (except on the union surface), which is not the case for simple operators such as max. As already stated, Rvashev's R-function [PASS95] provided a simple solution in the global support case, but a more efficient solution than Barthe's [BWd04] is required in the case of a local support field functions.

In this section we extend Rvachev's operator to achieve

the clean-union of local support implicit surfaces, in an efficient manner, and more generally of any skeleton-based implicit surface, including global support blobs and convolution surfaces. Figure 6 illustrates the different steps of our method.

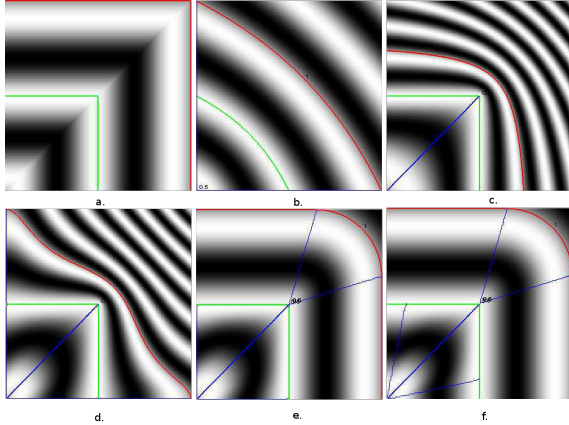


Figure 6: 2D comparison of different union operators: field values in sinusoidal grey scales are depicted in the (f_1, f_2) plane, the green and red lines being respectively the C (surface) and $2C$ (inside) iso-lines. a. The max operator is C^0 only; b. Rvachev's U_R is smooth but deforms the surface (green line) when directly applied to skeleton-based fields; c. U_{C_2} preserves the green line but contracts the field inside the shape (see the red line). d. U_{C_3} removes the contraction outside objects (see the region with small field values). e. Our final solution U_C for local support fields removes the contraction inside objects as well. f. Interpolation limits are added to properly deal with global support convolution primitives.

3.1. Local-support clean-union operator

Rvachev's clean-union operator U_R (whose positive part is depicted in Figure 6b) was created to combine functions similar to a distance: zero iso-value, negative-values (respectively positive) inside (respectively outside) and un-bounded field values. This simple and efficient operator is defined by:

$$U_R(f_1, f_2) = f_1 + f_2 + \sqrt{f_1^2 + f_2^2} \quad (2)$$

The goal is to make the operator applicable to skeleton-based scalar-fields, when surfaces are a C -iso surface with $C > 0$ and field values fall-off from inside to outside the shape. If they fall off to zero (local support), then the object resulting from the clean-union combination should also have a fall off to zero (i.e. the operator should be internal to the set of local support implicit surfaces). To achieve this, we translate and scale the U_R operator by defining:

$$U_{C_1}(f_1, f_2) = C + U_R(f_1 - C, f_2 - C) \quad (3)$$

As $U_{C_1}(0, 0) = C(\sqrt{2} - 1) \neq 0$, we need to scale U_{C_1} so as

to meet the zero value boundaries property. That gives us:

$$U_{C_2}(f_1, f_2) = C(1 - (U_R(f_1 - C, f_2 - C)/U_R(-C, -C)))$$

which can be rewritten as :

$$U_{C_2}(f_1, f_2) = \left(1 + \frac{\sqrt{2}}{2}\right) U_{C_1} - C \frac{\sqrt{2}}{2}$$

The new operator U_{C_2} (Figure 6c) respects fundamental local-support functions properties but contracts the field function. In particular, outside the boundary of f_2 (where $f_2 = 0$) we have:

$$U_{C_2}(f_1, 0) = C + \frac{f_1 - C + \sqrt{(f_1 - C)^2 + C^2}}{1 + \sqrt{2}}$$

which is different from the value of f_1 , thus altering the original field function. As a consequence, the blending quality would be altered after multiple blending.

In order to better preserve the original field properties outside the objects, we interpolate between the standard $\max(f_1, f_2)$ (which reproduces the fields of f_1 or f_2) and our function U_{C_2} as follows :

$$U_{C_3}(f_1, f_2) = (1 - \delta(f_1, f_2)) \max(f_1, f_2) + \delta(f_1, f_2) U_{C_2}(f_1, f_2)$$

with :

$$\delta(f_1, f_2) = \begin{cases} 0 & \text{when } \min(f_1, f_2) \leq 0 \\ \left(1 - \left(\frac{|f_1 - f_2|}{f_1 + f_2}\right)^2\right)^4 & \text{otherwise.} \end{cases} \quad (4)$$

Thus, as illustrated in Figure 6d the field is smooth everywhere except on the union surface $f_1 = f_2 = C$, and it tends close to f_1 (respectively f_2) when the other field function vanishes.

Unfortunately, the field inside the object (over the green line in Figure 6d) remains quite contracted, which could be a problem when objects are carved by combining them with negative-field primitives. We correct this by using Barthe's U_B operator [BDS*03] inside the objects, rather than the U_R operator in Equation 3. As shown in Figure 6e, our final clean union operator U_C , produces a high quality field function.

3.2. Adaptation to convolution surfaces

U_C can be applied to local support convolution surfaces, but not to those with infinite support. Such surfaces were made popular as in this case it is possible to compute an analytical expression of the convolution integral [She99, AC02]). As with local-support discussed above, these convolution surfaces are C -isosurfaces with $C > 0$, but rather than falling off to zero at a finite distance, the field value tends to zero at infinity. The adaptation of our compression-less operator to this case is quite straightforward:

If U_C was used directly, the interpolation applied outside the shape would never be used, since the field values never reach zero. The adaptation is just a modification of the limit

after which the operator reproduces f_1 and f_2 fields. To do so, we modify δ (Equation 4) so that $\delta_c(f_1, f_2) = 0$ when $f_1 = \alpha_1 f_2$ or $f_2 = \alpha_2 f_1$:

$$\delta_c(f_1, f_2) = \begin{cases} 0 & \text{when } f_1 \leq \alpha_1 f_2 \text{ or } f_2 \leq \alpha_2 f_1 \\ \left(\frac{4(f_1 - \alpha_1 f_2)(f_2 - \alpha_2 f_1)}{((1 - \alpha_2)f_1 + (1 - \alpha_1)f_2)} \right)^2 & \text{otherwise.} \end{cases}$$

Once this is done, we still use Barthe's U_B operator inside the objects in order to take into account the limit parameters α_1 and α_2 while providing a high quality field function (see Figure 6f).

4. Local, smooth blends

In this section we discuss our choice of operator inside the blending volume. As stressed in the introduction, the goal is to automatically calibrate a smooth blending operator in order to prevent the blurring of small shapes when blended into large, smooth ones. Our operator should also insure a seamless transition with the outer region, where the clean-union is used. The two following sections address these two problems.

4.1. Detail-preserving smooth blend

As discussed in the previous work section, a family of blending operators with some good range and shape control was already introduced by Barthe et al. [BDS*03, BWd04] in the case of both global support and local support implicit surfaces. More precisely, these operators restrict the range of blend between two, possibly different, iso-values of input implicit surfaces. The shape of the blend is then defined by a curve in the (f_1, f_2) domain, as depicted in Figure 7 (left).

To prevent small shapes from blurring, the distance at which blending is computed on the smallest shape (defined by an iso-value of the other surface) should be relatively small, while the other distance, on the larger shape, can be much larger. See Figure 7 (right). The limit iso-values chosen for the two surfaces can be set as the field values v_1 (respectively v_2) for points at the desired distance on S_1 (respectively S_2).

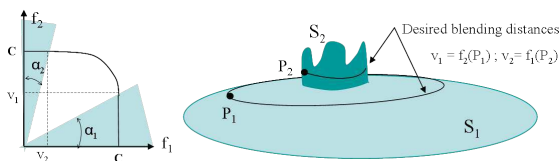


Figure 7: Parameters of the blending operator U_B used to set the blending distance on each surface S_1 and S_2 .

We thus use Barthe's U_B blending operator (designed for local support implicit surfaces) and the problem becomes: how can we automatically define these desired distances along the surfaces, to automatically blend shapes as the user expects while saving them from the burden of tuning parameters v_1 and v_2 ?

The blending distance chosen for each surface should be proportional to the size of its smallest local feature, to prevent the latter from blurring. In some cases, depending on the types of implicit surfaces, the model comes with good indicators of smallest size, such as the local radius of the shape in the case of skeleton-based implicit surfaces, or a set of local bounding boxes. However, these indicators might as well be missing. In these cases, it can be pertinent to estimate the local size of shape feature from the local curvature of each surface along the intersection curve. Curvature information can be efficiently computed as presented in [Gol05]. A last indicator for skeleton based local support primitives and convolution surfaces is given by the gradient value at surface points. Indeed, the larger the gradient, the smallest the local detail of the shape.

Whatever the technique used, we compute an approximation of the curvature radius r_1 (resp. r_2) of the small details we want to preserve on surface S_1 defined by f_1 (resp. S_2 defined by f_2). The blending distances d_1 on S_1 and d_2 on S_2 are then computed as $d_i = r_i/2$ ($i = 1, 2$). Limit field values v_1 (resp. v_2) being at distance d_1 (resp. d_2) are approximated by evaluating f_2 (resp. f_1) at a point P_1 (resp. P_2) located at distance d_1 (resp. d_2) from a point P of the intersection curve:

$$v_1 = f_2 \left(p + d_1 \cdot \frac{\nabla f_2(p)}{\|\nabla f_2(p)\|} \right), \quad v_2 = f_1 \left(p + d_2 \cdot \frac{\nabla f_1(p)}{\|\nabla f_1(p)\|} \right)$$

Thus the parameters of operator U_B are automatically set to these values. In case the progressive blend presented in Section 2.2 is not desired, the size of the support R of the ring-shaped blending volume can be set as well from the desired blending distances. Using $R = \max(d_1, d_2)$ can be too large for preventing unwanted blending; since blending at the computed distance on the largest surface is not necessary, we rather used $R = 2\min(d_1, d_2)$ in our experiments.

4.2. Seamless transition between blend and union

Our goal is now to generate an operator U , which will provide a smooth transition between the smooth blend operator parameterized as discussed above and the clean-union operator used outside of the blending volume, in order to generate a C^1 continuous field function at the border of the blending volume. Our operator U is built from the interpolation of Barthe's U_B blending operator and our U_C operator as follows:

$$U(f_1, f_2) = (1 - \beta)U_C(f_1, f_2) + \beta U_B(f_1, f_2)$$

where β is a power of the field value v of the ring-shaped blending volume.

5. Results

To demonstrate the usability of our new, local blend operator, we illustrate it in a simple animation experiment and within an interactive modelling system.

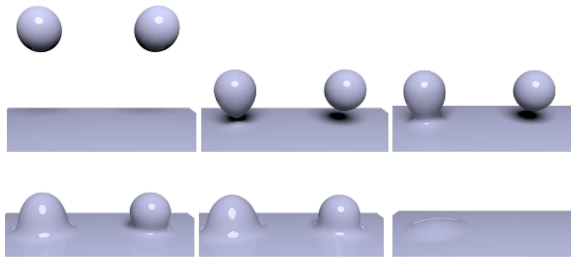


Figure 8: A drop of water reaching the water surface. In each image : Left, with a sum operator, and right, with our new blending method.

Figure 8 depicts a simple animation experiment, where a drop of water is dropped over a water surface. A local primitive (a soft object) is used for the drop and a global representation (a distance field) is used for the planar surface. Our result (the ball on the right in each of the image) validates our choice to start blending objects only when and where objects intersect. It also validates our parameterization for the size of the blending volume around the intersection curve.

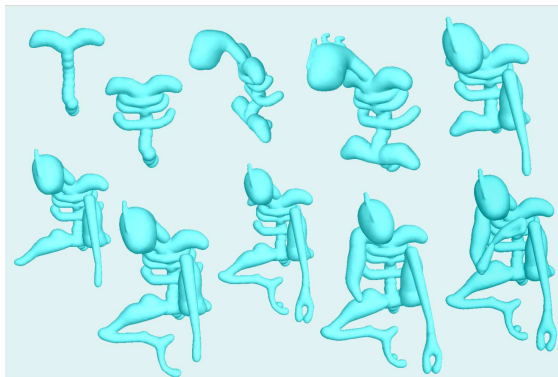


Figure 9: Some of the creation steps of the alien character from the teaser image. The hand and the knee are very close, but do not blend.

We also used our operator in an interactive modelling system where implicit primitives are progressively assembled, as illustrated on the alien in Figure 9. Primitives are convolution surfaces created interactively using skeleton curves. Four examples of different complexity were created: An alien, a group of dancers, a tree and a dragon. The two first examples illustrate the capability of the method to blend shapes only locally (such as the dancer’s hand not blending with her body in the teaser image) and the two other illustrate the fact that small shape feature are well preserved, even when blended into larger shapes (Figures 10 and 11).

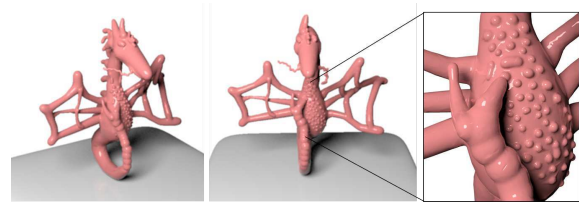


Figure 10: More views of the dragon from the teaser image, illustrating the interest of blending to create models of non-trivial topology and the fact that small primitives do not blur when blended into large ones.

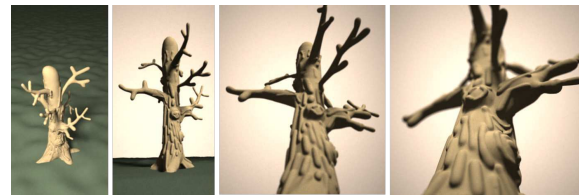


Figure 11: Some views of our tree model: blending is particularly challenging in this case, due to the complex shapes of the intersection curves between bark bumps and the trunk. Our method handles this case easily due to the automatic setting of the blending volumes.

5.1. Computational time

The designer spent 30min to an hour to create each of the four examples. This time includes the progressive creation and blending of all the primitives. To give a better idea of the computational time necessary for each blend, we stored the construction-history for each example, and we measured the portion of time spent in blending when these examples were re-loaded. This time can then be divided by the number of primitives, to get an average time for a blend. Table 1 presents these results.

We also compare in Table 2 the evaluation cost of different blending operators. As expected, the more advanced the operator, the larger its computation time. However, the great advantage of operators such as U_B and U_C is their locality. Thus in a complex model, they require less evaluations even though these evaluations are more expensive.

| U_R | U_B | U_C |
|-------|-------|-------|
| 0.08 | 0.115 | 0.95 |

Table 2: Times in seconds for the one million evaluations of different composition operators required to compute images as those presented in Figure 6.

5.2. Discussion

Our new method was found to be easy to use by our designer, since she just had to position primitives so that they slightly overlapped and to press a button for computing the blend. No

| | Dragon | Dancer Loop | Alien | Tree |
|-----------------------------------|--------|-------------|-------|------|
| Nb of primitives | 68 | 23 | 24 | 54 |
| Reconstruction time from a script | 277 | 34.3 | 41.9 | 475 |
| Average time per primitive | 2.65 | 1.24 | 1.31 | 6.3 |
| Average time per blend | 1.43 | 0.25 | 0.43 | 2.48 |

Table 1: Computation times in seconds required by the creation of models presented in the teaser. The second row gives the number of primitives built by the user and then blended by the system. The third row gives, for each model, the total computational time spent when they are directly created from a script. The fourth and the fifth rows give the average of the times spent in respectively creating individual primitives and evaluating our blending operator per transition.

extra primitives had to be created nor was it necessary to tune any parameters. The three long-standing problems identified for standard implicit blending, namely unpredictable blending at distance, lack of locality, and blurring of small details were solved by the system. The smoothness of the field function after each local blend, and the fact that our union operator does not create any unwanted contraction of the field, enabled the designer to create complex models without any problem or unexpected behaviour during successive blends.

A side benefit of our approach is the new *clean union operator* that can be used on its own for combining objects when a sharp union is desired.

We also identified some weaknesses of our methods, on which we are currently working:

Firstly, our current implementation defines blending volumes using a simple local support model (a soft object). Since the latter is based on the closest distance to the intersection curve, this model generates a field that may not be C^1 in the concavities of the curve. No artefact was observed in practice: the radius of the blending volume is often smaller than the curve radius and moreover, most curve concavities are inside objects. We can however see the problem on the bottom right of Figure 5, where the field looks strange inside the implicit shape, where the plane and the drop blend. Defining blending volumes from local-support convolution primitives would solve the problem since it would insure C^1 continuity of the scalar field everywhere and in all cases.

Secondly, the way we identify the size of small features for setting up the blending parameters may not be as general as we would like: in some cases, the two shapes we would like to blend may already have some tiny details (such as bumps on the surface), while the blend should take place at a much larger scale. In general, analysing the different scales of details in the frequency domain and enabling the user to choose the blending range among those scales (or doing it automatically from the size of the intersection curve) could be a better solution. If noisy reconstructed models from scanned real objects were used, such analyses would be necessary, however, this did not make our system less usable in practice, since objects are built from scratch and the designer naturally defined each object from coarse to fine.

Lastly, although our method is usable in interactive modelling sessions, the user sometimes has to wait for a few sec-

onds before the surface updates. This is mainly due to the lack of optimization of our polygonizer, and making blending and subsequent local re-meshing real-time would be a good practical improvement. Future work includes more accurate incremental polygonization, a GPU implementation of our method using, for instance the Intel Larabee architecture.

6. Conclusion and future work

This work has taken a new look at functionally-based implicit surface modelling. Instead of letting surfaces blend rather imprecisely at a distance as in current systems, we propose limiting the blend volume to where surfaces overlap, while enabling models to come close to each other everywhere else without deforming. This new approach combines the strength of implicit surfaces, i.e. their ability to blend seamlessly, with the ease of use of more standard models that do not blend. Our solution is generic and makes implicit surfaces much easier to use for both implicit modelling and animation.

As most previous methods [BDS*03, BWd04, PPK05], our approach defines a binary blending operator, acting on a pair of input surfaces, whereas 'sum' could be n-ary.

Consequently, if, for example, a group of implicit primitives with three overlapping blending volumes, are to be combined, the final shape will depend on the order in which the blends are performed. This is not a problem in an interactive modelling system where the user defines and adds shape components one by one, but this could be a weakness for procedural modelling, where complex, symmetric shapes with multiple branching parts are to be processed. Defining multiple-local blend operators would thus be a good topic for future work.

Also, if the user designs a very complex object composed of both large primitives and a lot of very small details, all composed using our blending operator, the correct computation of the intersection curves will become an issue and a scalable technique will have to be developed.

Lastly, our method offers a good solution to the unwanted blending problem when different primitives are combined, but does not address the self-blending of a single, complex implicit surface. For example modelling a folded snake using a convolution surface defined by a spiral curve as skeleton

is still impossible (except using intricate convolution specific methods [AJC02]), since the snake would self blend. In this case, artificially cutting the object into pieces and re-blending them would solve the problem, but this could be rather time consuming. A more general solution, that only enables blend where a complex shape self-overlaps, would be a good extension to our method.

In conclusion, we hope that our new vision of functionally-based implicit blending will be found inspiring, generate more research in the area and contribute to make implicit modelling more popular.

References

- [AC02] ANGELIDIS A., CANI M.: Adaptive implicit modeling using subdivision curves and surfaces as skeletons. In *Proceedings of the seventh ACM symposium on Solid modeling and applications* (2002), ACM New York, NY, USA, pp. 45–52.
- [AJC02] ANGELIDIS A., JEPP P., CANI M.-P.: Implicit Modeling with Skeleton Curves: Controlled Blending in Contact Situations. In *International Conference on Shape Modeling* (2002), pp. 137–144.
- [BDS*03] BARTHE L., DODGSON N. A., SABIN M. A., WYVILL B., GAILDRAT V.: Two-dimensional potential fields for advanced implicit modeling operators. *Computer Graphics Forum* 22, 1 (2003), 23–33.
- [BGOS06] BARGTEIL A. W., GOKTEKIN T., O'BRIEN J. F., STRAIN J. A.: A semi-lagrangian contouring method for fluid simulation. *ACM Trans. Graph.* 25, 1 (2006), 19–38.
- [BMPB08] BRODERSEN A., MUSETH K., PORUMBESCU S., BUDGE B.: Geometric texturing using level sets. *IEEE Transactions on Visualization and Computer Graphics* 14, 2 (2008), 277–288.
- [BPCB08] BERNHARDT A., PIHUIT A., CANI M.-P., BARTHE L.: Matisse: Painting 2D regions for modeling free-form shapes. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling, SBIM 2008, June, 2008* (Annecy, France, June 2008), Alvarado C., Cani M.-P., (Eds.), pp. 57–64.
- [BWd04] BARTHE L., WYVILL B., DE GROOT E.: Controllable binary csg operators for “soft objects”. *International Journal of Shape Modeling* 10, 2 (2004), 135–154.
- [CH01] CANI M.-P., HORNUS S.: Subdivision-curve primitives: A new solution for interactive implicit modeling. *Shape Modeling and Applications, International Conference on* (2001), 0082.
- [DG95] DESBRUN M., GASCUEL M.: Animating soft substances with implicit surfaces. In *SIGGRAPH'95 Conference Proceedings* (Los Angeles, CA, Aug. 1995), Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 287–290.
- [dGWvdW09] DE GROOT E., WYVILL B., VAN DE WETERING H.: Locally restricted blending of blobtrees. *Computers & Graphics* (2009).
- [Duf92] DUFF T.: Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry. In *Computer Graphics (Proceedings of SIGGRAPH 92)* (1992), pp. 131–138.
- [EGB09] EYIYUREKLI M., GRIMM C., BREEN D.: Editing Level-Set Models with Sketched Curves. In *Proceedings of Eurographics/ACM Symposium on Sketch-Based Interfaces and Modeling* (2009), pp. 45–52.
- [FSSV06] FLÓREZ J., SBERT M., SAINZ M. A., VEHÍ J.: Improving the Interval Ray Tracing of Implicit Surfaces. In *Computer Graphics International 2006* (2006), pp. 655–664.
- [Gol05] GOLDMAN R.: Curvature formulas for implicit curves and surfaces. *Computer Aided Geometric Design* 22, 7 (2005), 632–658.
- [GW95] GUY A., WYVILL B.: Controlled blending for implicit surfaces using a graph. In *Implicit Surfaces'95—the First Eurographics Workshop on Implicit Surfaces* (Grenoble, France, Apr. 1995), pp. 107–112.
- [Lev03] LEVIN D.: Mesh-Independent Surface Interpolation. In *Geometric Modeling for Scientific Visualization* (2003), pp. 181–187.
- [LY03] LJUNG P., YNNERMAN A.: Extraction of intersection curves from iso-surfaces on co-located 3d grids. In *SIGRAD2003 Proceedings, ISSN 1650-3686* (2003), Linköping Electronic Conference Proceedings, pp. 1650–3740.
- [MBWB02] MUSETH K., BREEN D. E., WHITAKER R. T., BARR A. H.: Level Set Surfaces Editing Operators. In *Proceedings of SIGGRAPH'02* (2002), ACM, pp. 330–338.
- [OF02] OSHER S. J., FEDKIW R. P.: *Level Set Methods and Dynamic Implicit Surfaces*, 1 ed. Springer, October 2002.
- [PASS95] PASKO A., ADZHIEV V., SOURIN A., SAVCHENKO V.: Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer* 11, 8 (1995), 429–446.
- [PPK05] PASKO G. I., PASKO A. A., KUNII T. L.: Bounded blending for Function-Based shape modeling. *IEEE Comput. Graph. Appl.* 25, 2 (2005), 36–45.
- [Set99] SETHIAN J. A.: *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, 2 ed. Cambridge University Press, June 1999.
- [She99] SHERSTYUK A.: Kernel functions in convolution surfaces: a comparative analysis. *The Visual Computer* 15, 4 (1999), 171–182.
- [Sny92] SNYDER J. M.: Interval analysis for computer graphics. In *Computer Graphics (Proceedings of SIGGRAPH 92)* (1992), pp. 121–130.
- [SPOK95] SAVCHENKO V. V., PASKO A., OKUNEV O. G., KUNII T. L.: Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum* 14, 4 (1995), 181–188.
- [SPS04] SCHMITT B., PASKO A. A., SCHLICK C.: Constructive sculpting of heterogeneous volumetric objects using trivariate b-splines. *The Visual Computer* 20, 2-3 (2004), 130–148.
- [TPF89] TERZOPOULOS D., PLATT J., FLEISHER K.: Heating and melting deformable models (from goop to gloop). In *Graphics Interface '89* (London, Ontario, June 1989), pp. 219–226.
- [WGG99] WYVILL B., GUY A., GALIN E.: Extending the csg tree - warping, blending and boolean operations in an implicit surface modeling system. *Comput. Graph. Forum* 18, 2 (1999), 149–158.
- [WMW86] WYVILL G., MCPHEETERS C., WYVILL B.: Data Structure for Soft Objects. *The Visual Computer* 2, 4 (February 1986), 227–234.
- [WvO96] WYVILL B., VAN OVERVELD K.: Polygonization of Implicit Surfaces with Constructive Solid Geometry. *Journal of Shape Modelling* 2, 4 (1996), 257–274.
- [WW00] WYVILL B., WYVILL G.: Better blending of implicit objects at different scales. In *ACM Siggraph 2000 Sketch Proceedings* (2000), ACM.

7.3.5 Gradient-based Implicit Modeling.

Auteurs : Olivier Gourmel, Loïc Barthe, Marie-Paule Cani, Brian Wyvill, Adrien Bernhardt, Mathias Paulin, Herbert Grasberger

Revue : Recommended for publication by the Siggraph 2011 committee, with modifications, at Transactions on Graphics

Gradient-based Implicit Modeling

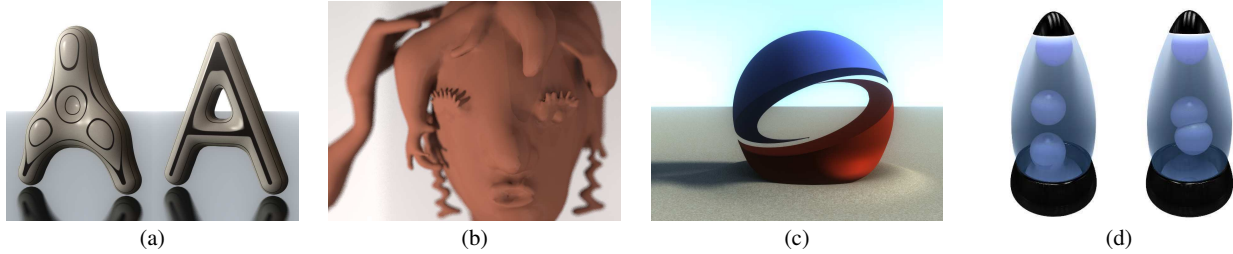


Figure 1: (a) The unwanted bulge created by standard blending is suppressed by our gradient-based method while preserving the desired topological genus. (b) We also prevent small details from blurring and primitives from blending where they do not intersect. Our generic framework allows the modeling of a wide range of composition operators such as “bulge in contact” (c) and asymmetric separation/fusion, combined to mimic a lava lamp (d).

Abstract

We introduce a new family of binary composition operators that solves four major problems of constructive implicit modeling: suppressing bulges when two shapes merge, avoiding unwanted blending at a distance, guaranteeing that the resulting shape keeps the topology of the union, and enabling sharp details to be added without being blurred. The key idea is that field functions should not only be combined based on their values, but also on their gradients. We implement this idea through a family of C^∞ composition operators evaluated on the GPU for efficiency, and illustrate it by applications to constructive modeling and animation.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations ; Constructive solid geometry (CSG)

Keywords: Geometric modeling, implicit surfaces, blending

1 Introduction

Functionally-based implicit surfaces [Bloomenthal 1997b] were introduced in the eighties as promising alternatives to mesh-based and parametric representations. They opened the way to constructive modeling systems where different object components could be smoothly blended, and enabled the production of animations where fluid-like objects could change topology over time through successive merging and splitting. Surprisingly, after being enhanced in the nineties by the introduction of convolution surfaces and of general construction trees, little advance was made on composition operators in the last decade, leaving four major problems unsolved:

1. Implicit blending creates unwanted bulging: for instance, even using convolution surfaces, a flat shape generated by blending together some implicit cylinders defined in the same plane will necessarily be thicker where the cylinders join (see Figures 1(a) and 4(a)).
2. Implicit models typically blend at a distance (Figure 4(c)). This is a major problem in modeling applications, where pre-

venting the blend between at least some parts of the models (e.g. between the hand and the head of the character in Figure 1(b)) is necessary. The issue is serious in animation too, where pieces of soft material should be allowed to deform each other and eventually blend, but only when they collide.

3. Sharp details typically blur when blended into larger implicit primitives, since they are totally included into the support region of the latter (Figure 4(b)). This is why implicit surfaces have the reputation of only producing simple, blobby shapes.
4. Lastly, although implicit modeling is an excellent way of generating arbitrary topologies, blending often produces material that will fill a hole that a designer intended (Figure 1(a) and 4(d)). The user has no easy way to control, or at least to predict, the topology he or she will get.

For instance, solving the above problems opens the way to achieving a practical solid modeling technique fast enough to apply to sketch based methods. We review the state of the art before presenting our general solution to these four problems.

1.1 Background and previous work

A functionally-based implicit surface S is defined as the C -iso-surface of a field function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$, as $S = \{p \in \mathbb{R}^3 \mid f(p) = C\}$. In this work, we use the convention that $f(p) > C$ defines the interior and f decreases outside an object.

Implicit surfaces are well known for their ability to combine shapes by simply composing the associated field functions. Except for two n -ary operators - *max* that results in a union and *sum* that generates some smooth blending between the input shapes [Sabin 1968; Ricci 1973; Blinn 1982] - composition is typically expressed as a binary operation of the form $f = g(f_1, f_2)$ where $g : \mathbb{R}^2 \rightarrow \mathbb{R}$ (see Figure 2). Improving composition operators has been a major topic of research over the years. Modeling techniques were enhanced by the introduction of *clean union* operators generating an exact union of the input shapes but with a smooth field everywhere else and of blending operators with local shape control [Pasko et al. 1995; Barthe et al. 2004]. Other operators were developed for animation applications, such as the generation of a contact surface surrounded by bulges when soft objects collide [Cani 1993].

The elimination of unwanted bulges (problem 1) was only tackled in two seminal papers: [Rockwood 1989] proposed an operator, the super-elliptic blend, where blending was parameterized by the cosine of the angle between the field gradients. Even though suppressing the unwanted bulges, this operator was C^0 only and increased problems 2 and 4, as reported in [Bloomenthal 1997a]. The

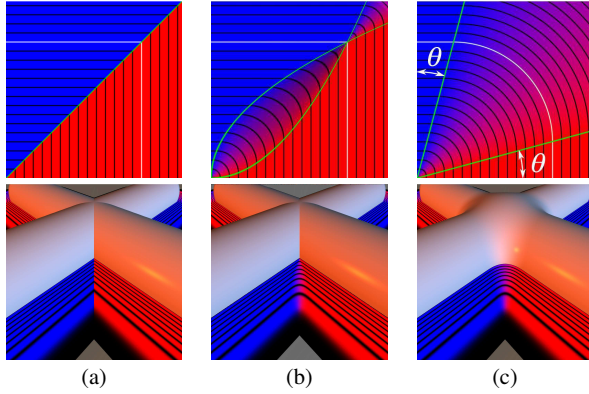


Figure 2: Three compositions applied to a pair of cylindrical implicit primitives forming a cross: (a) standard union, (b) clean union, (c) Barthe's operator parameterized by an opening angle θ . First row: plots of $g(f_1, f_2)$ where the axes are f_1 and f_2 . Black lines: some isocurves, white line: 0.5-iso-curve (surface of interest), background color: gradient direction with vertical (resp. horizontal) gradient in blue (resp. in red), green lines: boundaries out of which $g = \max(f_1, f_2)$. Second row: implicit surface $g(f_1, f_2) = 0.5$. Some iso-lines of g are depicted in a horizontal cutting plane.

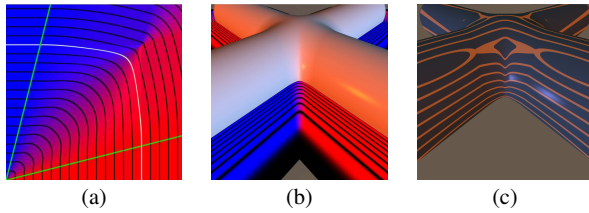


Figure 3: Bernhardt's operator interpolating between clean union and Barthe's blending. The distortions of the iso-curves of g (a) result in a G^1 -only implicit surface (b) which still has a bulge on top and exhibits a poor curvature distribution, shown by depicting reflection lines (c).

latter discussed alternative solutions for the n -ary composition of skeleton-based primitives: the kernels or the skeletons of convolution surfaces were used to parameterize a switch from sum to max operators. In addition to not being generally applicable, the lack of smoothness of these early solutions limited their usability in the case of successive blends. Our work builds on [Rockwood 1989] since our new composition operators are smooth functions of the angle between field gradients.

Although identified early [Bloomenthal 1997b], the unwanted blending problem (problem 2) was only solved recently in a general setting [Pasko et al. 2005; Bernhardt et al. 2010]. The proposed solutions are based on an extra implicit primitive, the blending volume, whose field is used to interpolate between blending and clean union (such as those of Figure 2). [Bernhardt et al. 2010] automatically generates blending volumes around intersection curves and sets parameters such that small primitives do not blur, providing a solution to problem 3 as well. The method has some drawbacks, for example only G^1 continuous field functions are produced leading to curvature artifacts (Figure 3). Also, computations involve sampling the intersection curve between the two input surfaces or extracting the curve from the intersection of the meshes used for display. This makes the solution costly and not scalable, especially in applications where many intersection curves would need to be generated at different resolutions, such as in a detailed model. Our method is a more general, scaleable approach and is able to restrict

blending to regions where the input shapes intersect. At the same time the topological genus of the resulting shape will remain that of the union (problem 4), which none of the previous solutions could achieve.

1.2 Contributions

We present a generic family of composition operators that brings a unified solution to the four major problems we mentioned. We localize blends near intersections without requiring any extra implicit primitive, which makes the method efficient and scalable. Instead, transition between blending and union is controlled through the angle between field gradients, which also enables us to avoid bulges at joints. In contrast with early work, we use this angle within a new family of C^∞ composition operators, designed to properly handle successive compositions. Our blending technique preserves sharp details and ensures that the topology of the composed object always remains that of the union, which makes composition predictable and well suited to constructive modeling frameworks. Our operators can also be tuned to animate progressive separation of soft material while preventing disjoint parts from blending at distance. They can also generate contact surfaces surrounded by bulges, possibly followed by progressive merging, when soft objects collide.

The method is generic among functionally-based implicit surfaces. Although it can be applied to discrete fields as well, where f is defined as some interpolation of sample values stored in a grid, we do not address level-set modeling frameworks since the composed field functions we output are not governed by differential equations. The remainder of this paper derives our solution for local implicit primitives $f(P) = C$, where f is local support and $C = 0.5$, such as in soft-objects and meta-balls. These surfaces are the most challenging to handle since the composed field has to maintain compact support. The same framework also applies to globally supported primitives, as we illustrate in Section 4.1 with convolution surfaces. See [Bloomenthal 1997b] for a full description of these models.

2 Gradient-based Composition

Let f_1 and f_2 be the field functions of the implicit primitives to be composed and g be the *composition operator* we are looking for. As in [Rockwood 1989], we claim that g should not only depend on f_1 and f_2 , but also on their gradients. Indeed, this appears as the only option to smoothly blend intersecting shapes where they form a sharp angle, while avoiding bulges where their surfaces are already aligned. This section first presents the key features of our solution. We then explain the field continuity issues that motivate the introduction of a family of quasi- C^∞ operators in Section 3.

2.1 Controlling opening from angle between gradients

As we just mentioned, g should be able to generate smooth blending between some parts of input shapes and union elsewhere. Following [Barthe et al. 2004] we choose g with an *opening angle* parameter $\theta \in [0, \pi/4]$ (see Figure 2(c)). θ controls the locality of the blend, i.e. the limit values in the (f_1, f_2) space out of which a union is computed. The operator g parameterized by θ can, for instance, switch from full union when $\theta = \pi/4$ to full blending when $\theta = 0$.

The key feature of gradient-based blending is to define the opening angle θ as a function of the angle between the field gradients at the query point P :

$$\theta(P) = h(\alpha(P))$$

where $\alpha = \widehat{\nabla f_1, \nabla f_2}$ is the angle between the field gradients and $h : [0, \pi] \rightarrow [0, \pi/4]$ is a continuous function called a *controller*.

While a constant controller h applies the same type of composition everywhere along the input surfaces (such as smooth blending,

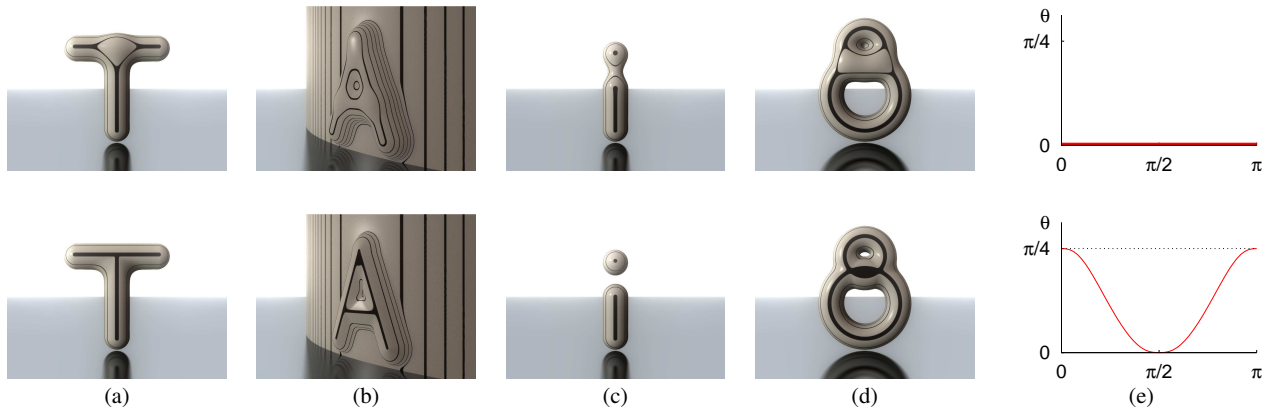


Figure 4: Comparison between standard blending (first row) - equivalent to a null controller - and our method (second row) with a controller that smoothly switches from clean union when $\alpha = 0$ or $\alpha = \pi$ to blending when $\alpha = \pi/2$. Our solutions to: (a) unwanted bulge, (b) blurring of small details, (c) blending at distance, (d) topology modification. (e) Plot of $h(\alpha)$.

158 shown in the first row of Figure 4), choosing controllers that adequately tune the opening angle according to α allows us to seamlessly solve the four challenging problems we mentioned. Figure 4 shows how this is achieved; from left to right we have:

- 162 (a) The unwanted bulge on top is the inflation produced by smooth blending where surfaces are aligned (α close to 0). To avoid this, the controller should switch to union when $\alpha = 0$. This is done by choosing h such that $h(0) = \pi/4$ (union) and $h(\pi/2) = 0$ (blending).
- 167 (b) This behavior also naturally prevents small details from blurring since the inflation produced by the blend is gradually prevented as we come close to the “top” of sharp features (where the input surfaces are aligned).
- 171 (c) Blending at distance occurs when two surfaces come close to each other without intersecting, so when α is close to π . This is easily avoided by choosing h such as $h(\pi) = \pi/4$ (union) and $h(\pi/2) = 0$ (blending).
- 175 (d) This setting also avoids the change of topological genus shown on top, since there are points inside each handle where the angle between gradients is π and which will therefore remain outside of the composed shape.

179 The controllers used to generate these compositions are depicted in 180 (e). Note that the one on the second row, which we call the *Camel controller* for its shape, solves all four challenging problems we 181 mentioned. It will be fully described in Section 4.

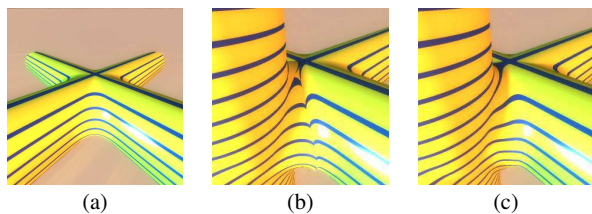


Figure 5: (a) Two cylindrical implicit primitives are combined using Barthe’s G^1 operator. The resulting shape still looks smooth. (b) A third primitive is added with the same blending operator. A C^0 sharp crease now appears. (c) Same scene using our new quasi- C^∞ blending operator.

2.2 Continuity issues

184 Computing composition based on the gradients of the input fields may locally cause a loss of one order of continuity for the resulting field. This is the case whenever piecewise polynomial field functions are combined, such as those used for defining local-support implicit primitives (e.g soft objects and meta-balls). In this case the level of continuity is finite where the polynomial parts of the function join and the composed field loses a degree of continuity. This decrease cannot be avoided, but efforts can be made such that neither the controller h nor the composition operator g (based on it) introduce any extra source of discontinuity, which could cause artifacts in case of consecutive blends (see Figure 5). In particular, Barthe’s operator [Barthe et al. 2004], depicted in Figure 2(c), cannot not be used in our framework as it is G^1 only and sets up a transition to a standard, C^0 only union. Note that its lack of smoothness also results in low quality curvature distribution in blending regions, as those shown in Figure 3(c) where it is used within Bernardt’s blending method [Bernhardt et al. 2010]. To avoid these problems, our goal is to define C^∞ operators g based on C^∞ controllers h and to set them to control transition to a clean union (with a smooth field everywhere outside the surface of interest) when $\theta = \pi/4$.

204 The constraint for g and h to be C^∞ is in fact theoretical: in practice, we just need g and h to approximate some C^∞ functions at a given precision. This has to be done without introducing oscillations, since the latter would result in curvature artifacts. We say that a function verifying this looser constraint is quasi- C^∞ continuous. Relaxing our continuity constraint to quasi- C^∞ will enable us to efficiently implement our operators on the GPU.

211 Section 3 introduces our new composition operators, which are applied to shape modeling and animation in Section 4. In these sections, the input fields are supposed to be singularity free, i.e. with smoothly varying, non-zero gradients vectors. The application of our framework to more general settings will be discussed in Section 5.

3 Quasi- C^∞ Operators

3.1 Controllers

218 We are looking for a family of C^∞ functions for defining the controllers h that convert the angle between gradients into the opening angle θ of the composition operator. This family should be flexible enough for enabling us to implement various modes of conversion,

including, but not restricted to those of Figure 4(e). Controllers designed to model organic shapes and to animate contact situations will be shown in Section 4. The family of functions needed for h should set some key values, and also provide some slope control; without which, unwanted bumps may appear, depending on the slopes of the input fields around regions where a clean union is applied. See Figure 6.

We therefore use a class of functions $h(\alpha)$, $\alpha \in [0, \pi]$ defined from eight parameters: three values of α_i ($i = 0, 1, 2$), where α_0 and α_2 are limits of the intervals $[0, \alpha_0]$ and $[\alpha_2, \pi]$ where h is set to be constant, the three associated values for h ($\theta_0 = h(\alpha_0)$, $\theta_1 = h(\alpha_1)$, $\theta_2 = h(\alpha_2)$) and two additional parameters w_0 and w_1 controlling the slope of the controller in respectively $[\alpha_0, \alpha_1]$ and $[\alpha_1, \alpha_2]$. See Figure 7. In order to ensure C^∞ continuity, h is piecewise defined from functions of class C^∞ that are flat at the boundaries of their support (i.e. all their successive derivatives equal zero). The equation we use is given in Appendix A.

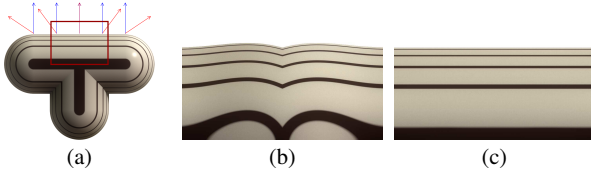


Figure 6: Role of the slope of h in blending. The red (resp. blue) vectors are gradients of the horizontal (reps. vertical) primitive. (a) Union. (b) Blending with a sharp controller ($w_0 = w_1 = 3$): unwanted bumps appear where the surfaces are almost aligned, since h falls very quickly to 0 there. (c) Tuning the slope of h according to the type of input primitives while keeping its other parameters unchanged solves the problem. Here, we set $w_0 = w_1 = 1$, a value pre-selected for soft objects.

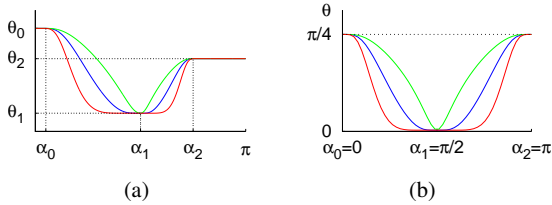


Figure 7: (a) The 8 parameters of a controller h with $w_0 = w_1$ respectively equal to 3, 1 and $1/3$ for the red, blue and green curves. (b) Three possible controllers for the blend in Figure 6, using the same values for w_0 and w_1 .

3.2 Composition operators

Let us now define another family of quasi- C^∞ functions for the composition operator g . This family should include smooth blends but can also be able to generate other useful compositions, such as bulges when soft objects come in contact. Whatever the type of composition, the operator should be parameterized by the opening angle θ , $\theta \in [0, \pi/4]$, with $\theta = h(\alpha)$. The convention is that the composition is fully applied when $\theta = 0$ (opened operator), and gradually switch to a clean union between the input shapes when $\theta = \pi/4$ (closed operator). See the first row of Figure 8.

Let $k_\theta(f_1)$ and $k_\theta(f_2)$ be the boundary curves defining the boundary of the central region, where the input field are modified (typically, blend). As shown in the second row of Figure 8, the main issue is getting the specific shape that these curves need to exhibit so that, when θ varies from 0 to $\pi/4$ (Figure 8(b)), the boundary curves smoothly interpolates between those of a standard blend (Figure 8(a)) and those of a clean union (Figures 2(b) and 8(c)).

We introduce a family of C^∞ boundary curves whose intermediate shape is shown in the second row of Figure 8 (b). The equation we use for $k_\theta(f)$ is given in Appendix B.

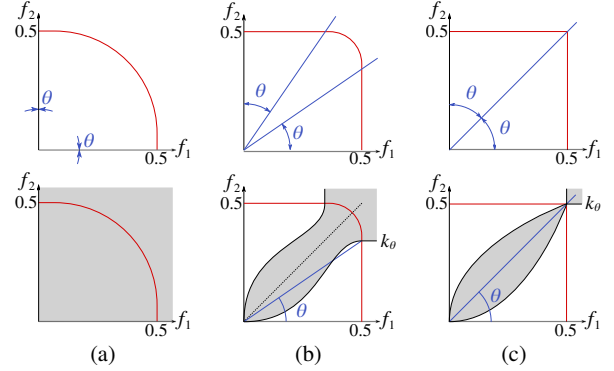


Figure 8: Our operator g . First row: The opening angle θ defines the width of the region where composition (here, smooth blend) is applied, at the level of the iso-value (red iso-curve). Second row: The boundary curves k_θ bound the region where the values of g are computed (g) (light grey area). A max operator is used outside. (a) Opened operator ($\theta = 0$). (b) Intermediate situation. (c) Closed operator ($\theta = \pi/4$).

The values of g inside the region bounded by the boundary curves k_θ (in grey on Figure 8) are defined by an operator $\bar{g} : \mathbb{R}^2 \rightarrow \mathbb{R}$, which sets the type of composition we desire. The equation of our composition operator g is thus:

$$g(f_1, f_2) = \begin{cases} \max(f_1, f_2) & \text{if } f_1 < k_\theta(f_2) \text{ or } f_2 < k_\theta(f_1) \\ \bar{g}(f_1, f_2) & \text{otherwise} \end{cases}$$

Finally, \bar{g} is defined intuitively through a silhouette curve $\bar{s} : \mathbb{R} \rightarrow \mathbb{R}$ defining the shape of the iso-curves of \bar{g} as illustrated in Figure 9. For instance, an arc-of-a-circle is simply created using $\bar{s}(\varphi) = 1$, $\forall \varphi$. Our operator $\bar{g}(f_1, f_2) = C$ is then the solution of:

$$\frac{\sqrt{(f_1 - k_\theta(C))^2 + (f_2 - k_\theta(C))^2}}{\bar{s}(\varphi)(C - k_\theta(C))} = 1, \quad \varphi = \arctan \frac{f_2 - k_\theta(C)}{f_1 - k_\theta(C)}$$

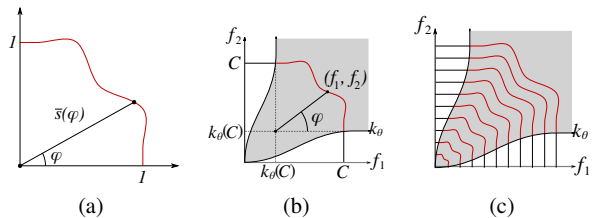


Figure 9: (a) A silhouette curve $\bar{s}(\varphi)$. (b) defines the shape of the iso-curves of \bar{g} to produce (c) the corresponding operator g .

Using any C^∞ silhouette curve \bar{s} , such that $\bar{s}(0) = \bar{s}(\pi/4) = 1$ and with flat derivatives at the level of the boundary curve (so that g 's derivatives match those of the \max function) is thus sufficient for getting a C^∞ operator g . To illustrate how different these silhouettes curves can be and the variety of resulting compositions, Section 4 will give the equations of the curves we use for smooth blending and for a "bulge-in-contact" composition.

3.3 Implementation on the GPU

Because of the smoothness we require from our families of controllers and composition parameters, complex mathematical functions have to be used. For efficiency, we developed a GPU implementation, which enables us to apply compositions in real-time.

The GPU implementation works as follows: the controller h which gives θ from the angle between gradients is stored as a 1D texture. We also pre-compute values of the operator g and its partial derivatives $\frac{\partial g}{\partial f_1}, \frac{\partial g}{\partial f_2}$ as functions of f_1, f_2 and θ in a regular grid stored as a 3D texture. Values of g are efficiently computed using the procedure presented in Appendix C and its partial derivatives are computed using finite differences. During field queries for a composed implicit surface, the values in these textures are linearly interpolated on the GPU: we first get θ from the gradients of the input fields and then $g(f_1, f_2)$ with this specific opening angle θ . Depending on the primitive, gradients are either computed using a closed form expression or finite differences. To efficiently apply successive compositions, the gradient of $g(f_1, f_2)$ is computed as:

$$\nabla g = \nabla f_1 \frac{\partial g}{\partial f_1} + \nabla f_2 \frac{\partial g}{\partial f_2}$$

A consequence of our GPU implementation is that the operators we use in practice are not exactly the C^∞ functions we developed, but quasi- C^∞ approximations of the latter: indeed, linear interpolation of values in a texture prevents unwanted oscillation and enables us to approximate the functions at any desired precision. We tested the method with $16^3, 32^3, 64^3$ and 128^3 grids for g and its gradients. Figure 10 shows the improvement of reflection lines on a close view of a composed primitive when texture resolution increases. Our current implementation uses 128^3 grids which is accurate enough to prevent any visual artifact, even after multiple compositions. Note that if a very high precision was required, e.g. for a manufacturing application, an off-line evaluation mechanism with bounded error could be set up, where binary search would be used to find the resolution needed to get a given error bound.

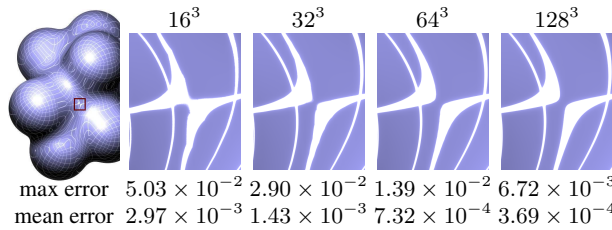


Figure 10: Close-up on the surface obtained after three successive blends when g is stored into textures of different resolutions. The piecewise linear interpolation generates artifacts in reflections (here illustrating the continuity of the fifth derivatives) with the 16^3 and 32^3 grids that are not perceptible using higher resolutions. The third and fourth rows respectively give the max and mean approximation error provided by the tri-linear interpolation of the pre-computed values.

4 Applications

In this section, we present applications of our generic composition operators to constructive modeling and animation. This leads us to define specific silhouette curves \bar{s} and the associated controllers h .

4.1 Smooth blending for constructive modeling

Being able to smoothly blend arbitrary shapes is one of the most useful compositions in constructive modeling systems. To create a good blending operator \bar{g}_b , we need a C^∞ silhouette curve \bar{s}_b ,

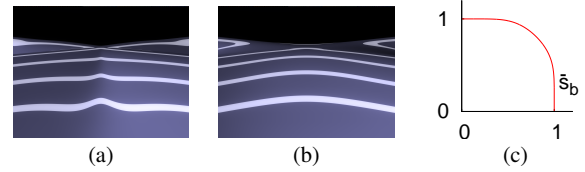


Figure 11: Close view of the top of the cross formed by two blended cylindrical primitives: (a) using Bernhard's blending; (b) using our operator g_b with a controller preventing the unwanted bulge; (c) Plot of \bar{s}_b from which g_b is derived.

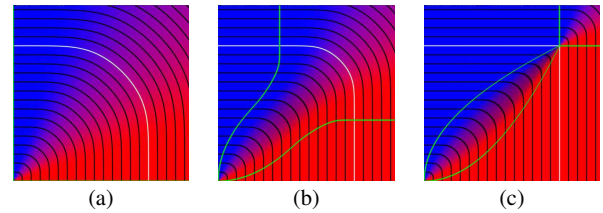


Figure 12: Plots of g_b in the (f_1, f_2) space for different values of the opening angle θ . (a) Opened operator: $\theta = 0$. (b) $\theta = \frac{\pi}{8}$. Note the improved shape of the iso-curves compared to those of Figure 3. (c) Closed operator, resulting in a clean union: $\theta = \frac{\pi}{4}$.

which, in addition to meeting the value and derivative constraints listed at the end of Section 3.2 (values 0.5 and null derivatives of every order at the extremities), avoids curvature oscillations as much as possible, as illustrated in Figure 11.

Our solution is a symmetric silhouette curve whose curvature monotonically increases, with a maximum at $\varphi = \pi/4$, then decreases back to zero (Figure 11(c)). We define \bar{s}_b as follows: We consider the function $u(x) = e^x - 1$ whose inverse is $u^{-1}(x) = \log(x + 1)$, and the transformation $T : v \mapsto u^{-1} \circ v \circ u$. Applying this transformation on the hyperbola $z(x) = 1/x$ results in a C^∞ function whose branches flatten very quickly while converging to the axes. We thus define: $\bar{s}_b(x) = 1 - e^{-1} T^2(z)(e - ex)$. For our pre-computations of g on the GPU, we express this profile function in polar form, as $\bar{s}_b(\varphi)$, by sampling the curve at points $X_i = (x_i, \bar{s}_b(x_i))$ and pre-computing the value of φ_i at each X_i . To evaluate the function $\bar{s}_b(\varphi)$, we use binary search to find the interval $[\varphi_i, \varphi_{i+1}]$ containing φ , and then use linear interpolation. Although not exactly flat at the boundaries, this silhouette curve has zero derivatives in $\varphi = 0$ and $\varphi = \pi/2$ up to an error of $\epsilon = 10^{-5}$, which is accurate enough for our needs. Its fairness leads to a clear improvement compared with Bernhard's blending solution. See Figures 11 and 12.

This blending operator g_b computed using \bar{s}_b can be combined with different controllers according to the need. Our applications to shape modeling lead us to design two specific ones (see Table 1 and Figure 14). The first one, called *camel controller* because of the shape of the associated curve, is symmetric and was especially designed to solve, for soft objects (local-support, polynomial primitives), all four problems mentioned in the introduction (Figure 4). This controller is well suited to CAGD applications, where providing smooth, predictable blending behavior while preventing bulges where two primitives join is essential.

The second one, called the *organic controller*, was developed for an interactive modeling system dedicated to the modeling of organic shapes, using global-support convolution surfaces based on Cauchy kernel [Tai et al. 2004]. Here, being able to model both smooth and sharp features in the same composition is important: for instance,

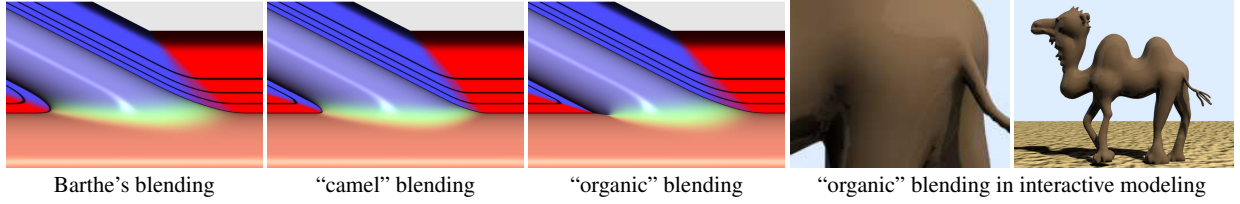


Figure 13: Comparison of Barthe's blending with our operator g_b , used either with the "camel" or with the "organic" controller.

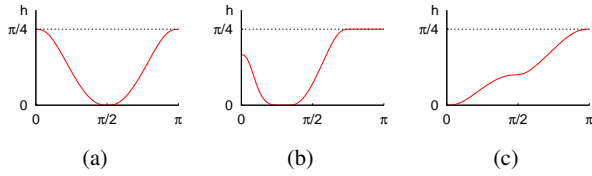


Figure 14: Three useful controllers: (a) camel and (b) organic controllers used with the blending operator g_b ; (c) contact controller used with the bulge-in-contact operator g_c .

when blending a character's nose to the head (Figure 1(b)) or the leg and tail of an animal to its body (Figure 13 (right)), the junction has to be smooth at the top but sharp at the bottom. To achieve this, our *organic controller* is set to be asymmetric: surfaces smoothly blend where they have nearly orthogonal gradients, while a clean union generating sharp creases is used when gradients tend to be in opposite directions (see Figure 13). This enables us to get detailed organic-like models with both smooth parts and sharp creases.

| | α_0 | α_1 | α_2 | θ_0 | θ_1 | θ_2 | w_0 | w_1 |
|---------|------------|------------|------------|------------|------------|------------|-------|-------|
| Camel | 0 | $\pi/2$ | π | $\pi/4$ | 0 | $\pi/4$ | 1 | 1 |
| Organic | 0 | $\pi/3$ | $3\pi/4$ | $\pi/6$ | 0 | $\pi/4$ | 3 | 1 |
| Contact | 0 | $\pi/2$ | π | 0 | $\pi/10$ | $\pi/4$ | 1 | 0.7 |

Table 1: Parameter values defining our different controllers.

4.2 Bulge in contact for soft objects animation

To illustrate the variety of compositions we can achieve, we designed a "bulge-in-contact" operator g_c , inspired by [Cani 1993], which instead of blending primitives that intersect, mimics the effect of two surfaces in contact that bulge as if made from a soft material such as rubber (Figures 15, 1(c) and (d)).

Keeping in mind that the silhouette curve \bar{s} directly gives the shape of the deformation, Figure 15(a) shows the deformation of f_1 on the right and of f_2 on top. We set \bar{s} such as to create smoothly increasing and then decreasing bulges. This is done by using values smaller than 0.5 on the silhouette curve. Figure 15(c) (top) shows the resulting point of contact (no deformation) and bottom the deformation as f_1 and f_2 overlap.

Defining these contact silhouette curves is done using the parametric expression:

$$\bar{s}_c(t) = \begin{cases} x_c(t) = \begin{cases} t & \text{if } t \geq 1 \\ 1 - Ke^{1-\frac{1}{2-t}}(t-1) & \text{otherwise} \end{cases} \\ y_c(t) = \begin{cases} 1 - Ke^{1-\frac{1}{t}}(1-t) & \text{if } t \geq 1 \\ 2-t & \text{otherwise} \end{cases} \end{cases}$$

for $t \in [0, 2]$, where $K \in [0, 1]$ is a parameter that controls the thickness of the bump. The effect of K on the iso-curves of \bar{g}_c is illustrated in Figure 15 (a). The curve \bar{s}_c is perfectly flat at the extremities and is, as desired, only C^0 continuous at $\varphi = \frac{\pi}{4}$ in order to simulate the fold to be created between the two input surfaces.

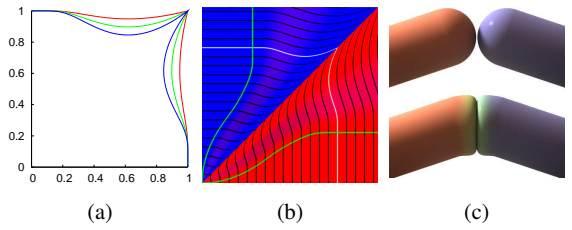


Figure 15: (a) Plots of \bar{s}_c for $K = 0.25$ (red), $K = 0.5$ (green) and $K = 0.75$ (blue). (b) The corresponding operator g_c when $h = \frac{\pi}{8}$ and $K = 0.8$. (c) The application of our operator g_c .

Again, the polar form $\bar{s}_c(\theta)$ can easily be evaluated using binary search, as done for \bar{s}_b in Section 4.1.

Finally, a *contact controller* (Table 1 and Figure 14) is designed so that when used with our bulge-in-contact operator, objects do not deform at distance but bulge when they intersect, mimicking contact between soft objects. The operator g_c is illustrated with $K = 0.8$ in Figures 1(d) and 15(c).

5 Results and Discussion

The usefulness of our generic composition framework for constructive modeling and animation applications is illustrated in Figures 16 and 1(d). In the "lava lamp" animation, g_c is used in conjunction with our blending operator to also get some blending when bubbles leave the bottom of the lamp. Other interesting animation effects can be easily produced by dynamically modifying the controller parameters during an animation. For instance, Figure 17 illustrates an animation where primitives do not blend when they are not in contact, but do blend (and thus deform each other) when they separate even though they are not in contact anymore, as in standard blending.

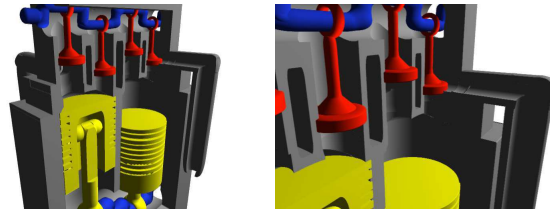


Figure 16: Constructive modeling examples.

Achieving good quality results leads us to use more complex equations for composition than in all previous methods. Therefore, our new, generic family of operators would be slow to compute without the GPU implementation. The method is based on pre-computations that only need to be performed once for each operator and for each controller, and just requires the memory space of a few textures. This implementation allows us to use arbitrarily complex expressions, not necessarily closed-form, for g and h ,

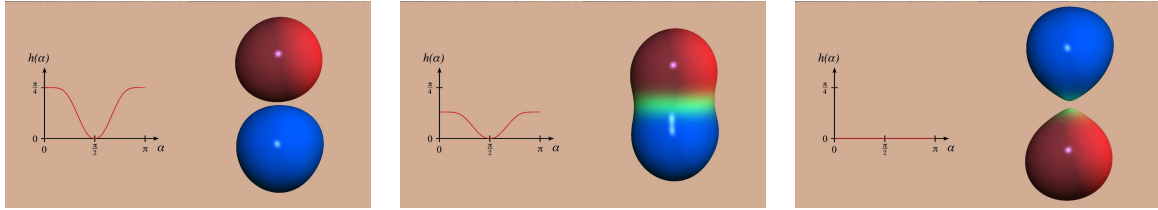


Figure 17: From left to right, the controller is modified to create different blending behaviors during an animation.

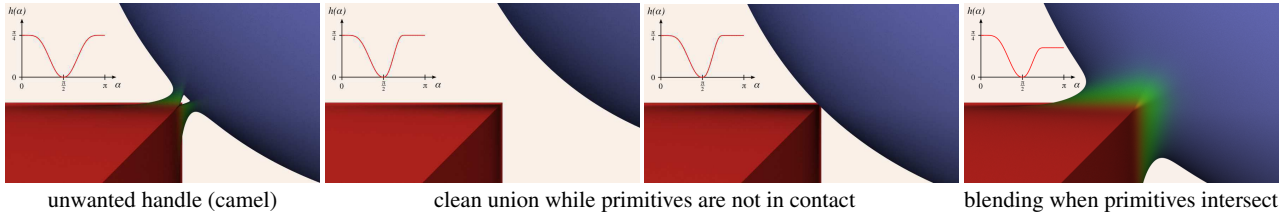


Figure 18: Composition of a cube and a sphere at a gradient discontinuity (an edge of the cube). On the left, the artifact generated when the camel controller is used for blending, and then, controllers solving the problem when the sphere moves and intersects the cube.

390 while answering field queries at the cost of a simple texture fetch. 433
 391 Therefore, when designing a new operator, one can focus on the de- 434
 392 sired effect rather than on the best properties a closed form equation 435
 393 could achieve. The precomputation of our operator and its partial 436
 394 derivatives in 128^3 grids takes 600 ms on a Core i7 950 and their 437
 395 transfer to the device memory from the host memory takes 3 ms. 438
 396 The 786 million evaluations of g required for the rendering of Fig- 439
 397 ure 10 take less than 270 ms on a NVIDIA GTX 480. 440

398 **Limitations of our method:** Firstly, as we emphasized in Section 441
 399 3.1, adequate slope for the controller has to be used, depend- 442
 400 ing on the slope of the input field functions, for not creating extra 443
 401 artifacts when removing a bulge. In practice, we use the interac- 444
 402 tive feedback of implicit modeling systems to pre-set the steepness 445
 403 parameters w_0 and w_1 of our controllers according to the family 446
 404 of field function used. Typically, we need sharper controllers to 447
 405 blend soft-objects based on smooth polynomial field functions than 448
 406 to compose sharper convolution surfaces. Once the steepness pa- 449
 407 rameters are set, we design specific controllers by adjusting the 450
 408 three input values for $(\alpha_i, h(\alpha_i))$. Providing an automatic tuning 451
 409 of the controller’s slope would be a better solution. The method, 452
 410 as currently implemented, still requires little user interaction since 453
 411 the manual tuning only needs to be done once for a given family of 454
 412 field functions, for each controller. 455

413 A drawback of our gradient-based operators is that they theoretic- 456
 414 ally require at least G^1 continuity of the field functions f_1 and 457
 415 f_2 to be combined. Using our composition framework in a more 458
 416 general setting, where the field may have a few singular points 459
 417 (with null gradients and therefore undefined angle between gra- 460
 418 dients) would however be desirable, as well as using it even for 461
 419 shapes with gradient discontinuities, e.g. for primitives with sharp 462
 420 edges. In case of null gradient vectors, we simply decrease the 463
 421 opening angle to $\min(\theta_0, \theta_1, \theta_2)$ when the gradient norm becomes 464
 422 close to null, so that the controller is constant at the vicinity of the 465
 423 singularity. This only leaves us with the case of artifacts created 466
 424 by discontinuities of the gradient field. This is illustrated by Fig- 467
 425 ure 18(left), where the camel controller is used for blending in the 468
 426 vicinity of a sharp edge. The problem can be solved in the follow- 469
 427 ing way: while primitives are not in contact, a controller preventing 470
 428 blend is required, and as soon as the primitives intersect, a blend 471
 429 can be progressively performed. This is done easily by manually 472
 430 tuning a few controller parameters as illustrated in Figure 18. In 473
 431 an animation, the controller parameters have thus to be pre-set in key 474
 432 primitive positions such as those presented in Figures 18 and then

interpolated during the animation.

434 Finally, the main limitation of this work (also true for all re- 435
 436 cent composition methods), is that gradient-based operators are re- 437
 438 stricted to binary compositions of implicit surfaces. This may limit 439
 440 their applicability in situations where many primitives need to be 441
 442 symmetrically connected at once. Generalizing the idea of using 443
 444 gradients into n-ary composition operators would be a great topic 445
 446 for future work. Similarly, although they can be used to compose 447
 448 convolution primitives, our operators cannot replace the sum used 449
 450 inside each primitive, where complex skeletons are tessellated into 451
 452 triangles or segments, whose contributions are summed. We do not 453
 454 provide a solution that prevents a shape, modeled as a single con- 455
 456 volution primitive, from blending with itself where the user did not 457
 458 intend. 459

6 Conclusion and Future Work

448 We have presented a generic family of composition operators, 449
 449 applicable to both constructive implicit modeling and animation. 450
 450 These operators, implemented in real-time on the GPU, allow pre- 451
 451 cise tuning of the transition between smooth blends and sharp union 452
 452 within a single composition operation, making compositions both 453
 453 more general and more predictable. The main feature of our ap- 454
 454 proach is the use of gradient-controlled blending: our operators are 455
 455 not only a function of the field functions to be combined, but also of 456
 456 their gradient. This avoids the well known weaknesses of implicit 457
 457 modeling, such as creating unwanted bulges and blending at dis- 458
 458 tance. The topology of the resulting shape can be set to the topology 459
 459 of the union. Moreover, our operators naturally prevent small de- 460
 460 tails from blurring even when blended into much larger primitives, 461
 461 thanks to the sharp changes of gradient values. In consequence, 462
 462 complex models with both smooth parts and sharp creases can be 463
 463 easily created, in contrast with the common idea that implicit sur- 464
 464 faces can only represent smooth, blobby shapes. 465

465 The natural extension of using the angle between field gradients 466
 466 would be the use of the information on the norm of these gradients 467
 467 to improve blends. This norm could be used to automatically set the 468
 468 slope parameters of the controllers, according to steepness of the in- 469
 469 put fields. Another extension would be to rely on the second deriva- 470
 470 tives of the input fields to detect fast local gradient variations. This 471
 471 would help us to avoid user interaction currently necessary to ad- 472
 472 just controllers in areas where the field functions are very distorted 473
 473 or at the vicinity of gradient discontinuities such as sharp edges.

474 Lastly, the general methodology we developed for the controllers, 526
475 i.e. defining C^∞ curves with some shape control parameters, could
476 be re-used to introduce local-support, C^∞ fields for implicit prim-
477 itives. This would ensure always having smooth gradient fields,
478 whatever the number of gradient-based compositions.

479 References

480 BARTHE, L., WYVILL, B., AND DE GROOT, E. 2004. Control- 527
481 lable binary csg operators for “soft objects”. *International Jour-
482 nal of Shape Modeling* 10, 2, 135–154.

483 BERNHARDT, A., BARTHE, L., CANI, M.-P., AND WYVILL, B. 528
484 2010. Implicit blending revisited. *Proc. of Eurographics, Com-
485 puter Graphics Forum* 29, 2, 367–376.

486 BLINN, J. F. 1982. A generalization of algebraic surface drawing. 531
487 *ACM Trans. Graph.* 1, 3, 235–256.

488 BLOOMENTHAL, J. 1997. Bulge elimination in convolution sur- 529
489 faces. In *Computer Graphics Forum*, vol. 16, 31–41.

490 BLOOMENTHAL, J., Ed. 1997. *Introduction to Implicit Surfaces*. 530
491 Morgan Kaufmann, July.

492 CANI, M.-P. 1993. An implicit formulation for precise contact 527
493 modeling between flexible solids. In *20th Annual Conference
494 on Computer Graphics and Interactive Techniques, SIGGRAPH
495 1993*, 313–320. Published as Marie-Paule Gascuel.

496 DESBRUN, M., AND CANI, M.-P. 1995. Animating soft sub- 528
497 stances with implicit surfaces. In *22nd annual Conference on
498 Computer Graphics and Interactive Techniques, SIGGRAPH
499 1995*, vol. 29, 287–290. Published as Marie-Paule Gascuel.

500 PASKO, A., ADZHIEV, V., SOURIN, A., AND SAVCHENKO, V. 531
501 1995. Function representation in geometric modeling: concepts,
502 implementation and applications. *The Visual Computer* 11, 8,
503 429–446.

504 PASKO, G. I., PASKO, A. A., AND KUNII, T. L. 2005. Bounded 529
505 blending for Function-Based shape modeling. *IEEE Comput.
506 Graph. Appl.* 25, 2, 36–45.

507 RICCI, A. 1973. Constructive Geometry for Computer Graphics. 530
508 *computer journal* 16, 2 (May), 157–160.

509 ROCKWOOD, A. 1989. The displacement method for implicit 527
510 blending surfaces in solid models. *ACM Transactions on Graph-
511 ics* 8, 4, 279–297.

512 SABIN, M.-A. 1968. The use of potential surfaces for numeri- 528
513 cal geometry. In *Tech. Report VTO/MS/153, British Aerospace
514 Corp., Weybridge, U.K.*

515 TAI, C., ZHANG, H., AND FONG, J. 2004. Prototype model- 529
516 ing from sketched silhouettes based on convolution surfaces. In
517 *Computer Graphics Forum*, vol. 23, 71–83.

518 A Controller Equation

519 The values of the *controller* h in $[\alpha_0, \alpha_1]$ and in $[\alpha_1, \alpha_2]$ are 534
520 computed by linearly stretching and scaling the function $\kappa(x) =$
521 $1 - \nu(1 - \nu(x))$ where $\nu(x) = e^{1-\frac{1}{x}}$, such that h meets the three
522 constraints $h(\alpha_1) = \theta_1$, $h(\alpha_2) = \theta_2$ and $h(\alpha_3) = \theta_3$. The func-
523 tion κ is flat (i.e. its successive derivatives equal zero) at both 0 and
524 1, which ensures that the *controller* is of class C^∞ . Note how w_0 535
525 and w_1 are used to control the slopes while keeping the constrained

values.

$$h(\alpha) = \begin{cases} \theta_1 & \alpha \leq \alpha_0 \\ \left(\kappa \left(\frac{\alpha - \alpha_1}{\alpha_0 - \alpha_1} \right) \right)^{w_0} (\theta_0 - \theta_1) + \theta_1 & \text{if } \alpha \in]\alpha_0, \alpha_1] \\ \left(\kappa \left(\frac{\alpha - \alpha_1}{\alpha_2 - \alpha_1} \right) \right)^{w_1} (\theta_2 - \theta_1) + \theta_1 & \text{if } \alpha \in]\alpha_1, \alpha_2] \\ \theta_2 & \text{otherwise} \end{cases}$$

526 B Equation of the Boundary Curves

527 The boundary curves are defined by C^∞ functions $k_\theta(f)$, where f
528 is either f_1 or f_2 (see Figure 8). We set $k_\theta(f)$ to a constant value
529 inside of the implicit primitives, where $f > 0.5$, using:
530

$$k_\theta(f) = \tan(\theta)/2 \quad \forall f \in \left[\frac{1}{2}, +\infty\right).$$

531 To get an expression of k_θ outside of the primitives, we first intro-
532 duce the function:

$$\lambda_\theta(f) = \begin{cases} \frac{1+\tan(\theta)}{4} & \text{if } f \geq \frac{1}{2} \\ f & \text{if } f \leq \frac{\tan(\theta)}{2} \\ \frac{1-\tan(\theta)}{4} \phi\left(2\frac{2f-\tan(\theta)}{1-\tan(\theta)}\right) + \frac{\tan(\theta)}{2} & \text{otherwise} \end{cases}$$

where ϕ is a function ensuring the C^∞ continuity of λ_θ . We use:

$$\phi^{-1}(x) = x + e^{-1}T^2(1/x)(e - ex)$$

where T is defined in Section 4.1. We use binary search to get val-
ues of ϕ at the required precision, which does not affects efficiency
since in practice, our operators are pre-computed on the GPU. k_θ is
then defined from λ_θ using:

$$k_\theta(f) = \frac{\tan(\theta)}{2} \left(\frac{4}{1 + \tan(\theta)} \lambda_\theta(f) \right)^2, \forall f \in [0, \frac{1}{2}]$$

533 C Computing values of the operators

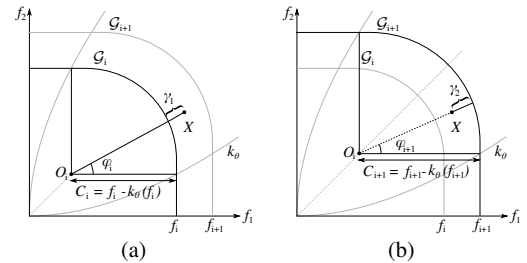


Figure 19: Computation of the interpolation parameter γ_1 (a) (re-
spectively γ_2 (b)), defined as the distances from X to G_i (respec-
tively from X to G_{i+1}), used to compute g at a sample point X .

For each grid coordinate θ_k , we compute the value of g at sampling
points $(f_1, f_2) = (f_i, f_j)$, $i, j \in [0; N]$ iteratively. For each i ,
we follow the f_i -iso-curve G_i of g and update the sample values as
follows. Outside the boundary curves, i.e. for each j such that $f_j <$
 $k_\theta(f_i)$, we set $g(f_i, f_j) = f_i$ and $g(f_j, f_i) = f_i$. Then, inside the
boundary curves, for each sampling point X located between G_i
and G_{i+1} , we compute $\bar{g}(X)$ by linearly interpolating f_i and f_{i+1}
between G_i and G_{i+1} as follows (see Figure 19):

$$g(X) = \bar{g}(X) = \frac{\gamma_2 f_i + \gamma_1 f_{i+1}}{\gamma_1 + \gamma_2}$$

with

$$\begin{aligned} \gamma_1 &= \|X - O_i\| - \bar{s}(\varphi_i)C_i \\ \gamma_2 &= \bar{s}(\varphi_{i+1})C_{i+1} - \|X - O_{i+1}\| \end{aligned}$$

where $O_i = (k_\theta(f_i), k_\theta(f_i))$, $C_i = f_i - k_\theta(f_i)$ and φ_i is computed
as presented in Section 3.2.

Curriculum vitæ

8.1 D roulement de la carri re

- **Oct. 2003** : Ma tre de Conf rence de l’Universit  Paul Sabatier de Toulouse dans l’ quipe VORTEX de l’IRIT.
- **2002-2003** : Post-doctorat au laboratoire d’informatique de l’Institut de Technologie (RWTH) d’Aix la Chapelle(Allemagne) sous la direction de Leif Kobbelt.
- **2001-2002** : Post-doctorat au laboratoire d’informatique de l’Universit  de Cambridge (UK), dans le cadre du projet europ en MINGLE «Multiresolution in Geometric Modelling» sous la direction de Neil Dodgson et Malcolm Sabin.
- **2000-2001** : A.T.E.R. «Attach  Temporaire d’Enseignement et de Recherche» (demi poste)   l’universit  Paul Sabatier de Toulouse.
- **1997-2000** : Doctorat en informatique de l’universit  Paul Sabatier de Toulouse dans l’ quipe synth se d’images et r alit  virtuelle de l’Institut de Recherche en Informatique de Toulouse sous la direction de V ronique Gaildrat.

Th me de recherche principal : Synth se d’images

- Mod lisation g om trique de formes tridimensionnelles
 - Utilisation des mod les volumiques pour la d formation des maillages surfaciques
 - Elaboration et classification d’op rateurs de composition pour les surfaces implicites
 - Etude et analyse des surfaces de subdivision
 - D veloppement de syst mes de mod lisation   base d’esquisses
 - Repr sentation de surfaces   base de points
- Algorithmes pour le rendu r aliste de sc nes tridimensionnelles
 - D veloppement de techniques temps r el pour la g n ration d’ombres douces physiquement plausibles puis pour la g n ration interactive d’ombres douces r alistes
 - Rendu interactif haute qualit  de surfaces implicites
 - Etude des mod les de rendu bas s sur Monte-Carlo

8.2 Encadrement d’ tudiants en th se

- **2001-2002** : Marc Cardle (co-encadrement de 7 mois) sur l’animation dirig e par la musique.

- **2003-2005** : Gaël Guennebaud (co-encadrement de 2 ans et 3 mois) sur la subdivision basée points et les ombres douces.
- **2003-2005** : Anca Alexe (co-encadrement de 2 ans et 3 mois) sur la modélisation par esquisses.
- **2005-2008** : Vincent Forest (co-encadrement de 3 ans et 3 mois) sur les ombres douces basées géométrie.
- **2009-2011** : Olivier Gourmel (co-encadrement de 1 an et 4 mois) sur la modélisation par surfaces implicites.
- **2008-2011** : Anthony Pajot (co-encadrement depuis septembre 2008) sur la génération d'images haute qualité basée Monte-Carlo.
- **A partir de 2010** : Marion Dunyach (dérogation de l'école doctorale pour la direction de la thèse). Début de thèse en septembre 2010. Dans le cadre d'une collaboration avec Mario Botsch (Université de Bielefeld, Allemagne), Marion Dunyach est inscrite en co-tutelle sur la déformation de maillages avec changement de topologie.

8.3 Management/animation scientifique et contractuelle

- Organisateur de la conférence AFIG 2008 : prise en charge de la soumission des papiers, gestion des reviewers et des reviews, édition des actes, choix et organisation de la venue d'un chercheur étranger de renom pour donner une présentation longue, suivi du bon déroulement de la conférence.
- Organisateur pour le Groupe de Travail Modélisation Géométrique 2009 : prise en charge de la soumission des papiers, choix du lieu de la conférence, édition des actes, suivi du bon déroulement de la conférence.
- Membre du consortium du projet ANR (05-MMSA-0004-01) NatSIM : Simulation de scènes naturelles en synthèse d'images, 2005-2008 (<http://www.irit.fr/NatSim>) participants :
 - IRIT : équipe VORTEX
 - INRIA : projets EVASION (Grenoble) et Iparla (Bordeaux)
 - CIRAD : équipe Virtual plants
 - LIAMA (Chine) : Project Greenlab

8.4 Rayonnement

- Vice-président du chapitre français d'Eurographics depuis 2009 et membre du CA depuis 2007.
- Editeur associé pour la revue internationale Graphical Models (GMOD) depuis Mai 2010.
- Membre du comité de programme de plusieurs conférences internationales :
 - Sketch-Based Interface and Modeling 2011 et 2008
 - Graphics Interface 2011
 - Siggraph Asia Technical Sketches and Posters 2011 et 2010
 - Shape Modeling International 2010 et de 2005 à 2008

-
- Symposium on Point Based Graphics de 2006 à 2009
 - Eurographics Workshop on Natural Phenomena de 2008 à 2009
 - Membre du comité de review de la conférence nationale AFIG depuis 2006.
 - Editeur pour la conférence AFIG 2008
 - Editeur pour le Groupe de Travail Modélisation Géométrique du CNRS 2009
 - Review d’une moyenne annuelle de 5 soumissions (en complément des comités) pour des revues internationales comme *Transaction on Graphics*, *Computer Graphics Forum*, *Computer and Graphics*, *Computer Aided Geometric Design* et des conférences comme *Eurographics*
 - Review d’une moyenne annuelle de 3-4 soumissions pour la *Revue Electronique Francophone d’Informatique Graphique* et la conférence nationale AFIG.
 - Expert pour l’ANR : appel «cosinus» en 2009, appels «contint» et «jeunes chercheurs» en 2010
 - Expert pour les Jeunes Entreprises Innovantes depuis 2009 (Direction Régionale pour la Recherche et pour l’Industrie)
 - Examineur pour les soutenances de thèses de :
 - Jamie Wither
 - Titre : *Sketching and annotation for the procedural modelling of complex phenomena*
 - Date de soutenance : 24 novembre 2008
 - Laboratoire Jean Kuntzmann – INRIA projet de recherche EVASION
 - Directrice de recherche : Marie-Paule Cani
 - Rapporteurs : Laurent Grisoni et John Hughes
 - Examineurs : Loïc Barthe et James Crowley (président du jury)
 - Mickaël Vermet
 - Titre : *Simulations par l’acoustique géométrique en présence de surfaces courbes avec prise en compte de la diffraction*
 - Date de soutenance : 25 mars 2010
 - Université de Poitier
 - Directeur de recherche : Rodolphe Vauzelle
 - Co-encadrants : Pierre Combeau et Nicolas Noé
 - Rapporteurs : Marc Neveu et Jean-Dominique Polack
 - Examineurs : Loïc Barthe, Philippe Jean et Patrick Vaudon
 - François Destelle
 - Titre : *Adaptation de schémas de subdivision pour la reconstruction d’objets sans artefact*
 - Date de la soutenance : 23 juin 2010
 - Institut Polytechnique de Grenoble
 - Directrice de recherche : Annick Montanvert
 - Co-encadrant : Cédric Gérot
 - Rapporteurs : Florent Dupont et Marc Neveu
 - Examineurs : Loïc Barthe et Pierre-Yves Coulon
 - Damien Rohmer

- Titre : Géométrie active pour l’animation et la modélisation
- Date de la soutenance : xx juin 2011
- Institut Polytechnique de Grenoble
- Directrices de recherche : Stefanie Hahmann et Marie-Paule Cani
- Rapporteurs : xx
- Examineurs : Loïc Barthe et xx

8.5 Activité de recherche, collaborations et production scientifique

Collaborations :

- **2001-2005** : avec Malcolm Sabin (Cambridge, UK) sur les surfaces de subdivision. Cette collaboration a donné lieu à 3 publications internationales dans des livres, 2 publications dans des revues internationales et 1 publication dans une conférence internationale
- **2003-2005** : avec Leif Kobbelt (RWTH – Aachen, Allemagne) sur les surfaces de subdivision. Cette collaboration a donné lieu à 1 publication internationale dans un livre, 1 publication dans une revue internationale et 1 publication dans une conférence internationale
- **Depuis 2005** : avec Marie-Paule Cani (INRIA – projet EVASION, Grenoble) sur la modélisation de formes complexes par esquisses et l’amélioration des modèles volumiques. Cette collaboration a donné lieu à 2 publications dans des revues internationales et 2 publications dans des conférences internationales.
- **Depuis 2003** : avec Brian Wyvill (University of Victoria, Canada) sur la modélisation de formes complexes par surfaces implicites. Cette collaboration a donné lieu à 4 publications dans des revues internationales.
- **Depuis 2009** : avec Mario Botsch (University of Bielefeld) sur la déformation de maillages avec changement de topologie.
- **Depuis 2010** : avec Gaël Guennebaud (INRIA – projet IPARLA, Bordeaux) sur la déformation de maillages guidée par des surfaces implicites.

Publications scientifiques :

Les .pdf des publications ainsi que des vidéos d’illustration des travaux de recherche sont accessibles à partir de mes pages web : www.irit.fr/~Loic.Barthe

Livres :

1. G. Guennebaud, L. Barthe and M. Paulin ”Real-Time Refinement”. Chapter in : Point Based Graphics, M. Gross and H. Pfister (eds), Morgan Kaufmann, 2007, ISBN : 978-0123706041.
2. L. Barthe ”Courbes et surfaces de subdivision”. Chapter 4 in : Informatique graphique, modélisation géométrique et animation, D. Bechmann and B. Peroche (eds), Hermes, pages 135-162, 2007, ISBN : 978-2-7462-1514-6.

3. L. Barthe, C. Gerot, M.A. Sabin and L. Kobbelt. "Simple computation of the eigencomponents of a subdivision matrix in the Fourier domain". *Advances in Multiresolution for Geometric Modelling*, N.A. Dodgson, M.S. Floater and M.A. Sabin (eds), Springer-Verlag, pages 245-257, 2005, ISBN : 3-540-21462-3.
4. C. Gerot, L. Barthe, N.A. Dodgson and M.A. Sabin. "Subdivision as a sequence of sampled C_p surfaces". *Advances in Multiresolution for Geometric Modelling*, N.A. Dodgson, M.S. Floater and M.A. Sabin (eds), Springer-Verlag, pages 259-270, 2005, ISBN : 3-540-21462-3.
5. M.A. Sabin and L. Barthe. "Artifacts in recursive subdivision surfaces", *Curve and Surface Fitting : St Malo 2002*, A. Cohen, J.L. Merrien and L.L. Schumaker (eds), Nashboro Press, Brentwood, pages 353-362, ISBN : 0-9728482-1-5.

Revue internationale :

1. O. Gourmel, L. Barthe, M.P. Cani, B. Wyvill, A. Bernhardt, M. Paulin, H. Grasberger. "Gradient-based Implicit Modeling", Recommended for publication by the Siggraph 2011 committee, with modifications, at *Transactions on Graphics*, 2011.
2. A. Pajot, L. Barthe, M. Paulin and P. Poulin. "Combinatorial Bidirectional Path-Tracing for Efficient Hybrid CPU/GPU Rendering", *Computer Graphics Forum*, 30(2), proc. of EUROGRAPHICS, pages 315-324, 2011.
3. A. Pajot, L. Barthe, M. Paulin and P. Poulin. "Representativity for Robust and Adaptive Multiple Importance Sampling", *IEEE Transactions on Visualization and Computer Graphics*, 17(8), pages 1108-1121, 2011.
4. O. Gourmel, A. Pajot, M. Paulin, L. Barthe and P. Poulin. "Fitted BVH for Fast Raytracing of Metaballs", *Computer Graphics Forum*, 29(2), proc. of EUROGRAPHICS, pages 281-288, 2010.
5. A. Bernhardt, L. Barthe, M-P. Cani and B. Wyvill. "Implicit Blending Revisited", *Computer Graphics Forum*, 29(2), proc. of EUROGRAPHICS, pages 367-376, 2010.
6. V. Forest, L. Barthe and P. Poulin. "Real-Time Hierarchical Binary-Scene Voxelization", *Journal of Graphics Tools*, 14(3), pages 21-34, 2009.
7. V. Forest, L. Barthe G. Guennebaud and M. Paulin. "Soft Textured Shadow Volume", *Computer Graphics Forum*, 28(4), proc. of Eurographics Symposium on Rendering, Girona, Spain, pages 1111-1120, June 2009.
8. V. Forest, L. Barthe and M. Paulin. "Accurate Shadows by Depth Complexity Sampling", *Computer Graphics Forum*, 27(2), proc. of EUROGRAPHICS, Crete, Greece, pages 663-674, April 2008.
9. G. Guennebaud, L. Barthe and M. Paulin. "High-Quality Adaptive Soft Shadow Mapping", *Computer Graphics Forum*, 26(3), proc. of EUROGRAPHICS, Prague, Czech Republic, pages 525-533, September 2007.
10. G. Guennebaud, L. Barthe and M. Paulin. "Interpolatory Refinement for Real-Time Processing of Point-Based Geometry", *Computer Graphics Forum*, 24(3), proc. of EUROGRAPHICS, Dublin, Ireland, pages 657-666, September 2005.

11. L. Barthe, B. Wyvill and E. De Broot. "Controllable Binary CSG Operators for "Soft Objects"", *International Journal of Shape Modeling*, 10(2), pages 135-154, December 2004.
12. G. Guennebaud, L. Barthe and M. Paulin. "Dynamic Surfel Set Refinement for High Quality Rendering", *Computers & Graphics*, 28(6), pages 827-838, 2004.
13. G. Guennebaud, L. Barthe and M. Paulin. "Deferred splatting", *Computer Graphics Forum*, 23(3), proc. of EUROGRAPHICS, Grenoble, France, pages 653-660, September 2004.
14. L. Barthe and L. Kobbelt. "Subdivision scheme tuning around extraordinary vertices", *Computer Aided Geometric Design*, 21(6), pages 561-583, June 2004.
15. L. Barthe, N.A. Dodgson, M.A. Sabin, B. Wyvill and V. Gaildrat. "Two-dimensional potential fields for advanced implicit modeling operators". *Computer Graphics Forum*, 22(1), pages 23-33, March 2003.
16. L. Barthe, B. Mora, N.A. Dodgson and M.A. Sabin. "Interactive implicit modelling based on C1 reconstruction of regular grids", *International Journal of Shape Modeling*, 8(2), pages 99-117, December 2002.
17. L. Barthe, V. Gaildrat and R. Caubet. "Extrusion of 1D implicit profiles : Theory and first application", *International Journal of Shape Modeling*, 7(2), pages 179-198, December 2001.

Conférences internationales avec sélection et publication des actes :

1. A. Pajot, L. Barthe and M. Paulin. "Sample-Space Bright Spots Removal Using Density Estimation", *Graphics Interface 2011*, 2011, à paraître.
2. O. Gourmel, A. Pajot, L. Barthe, P. Poulin and M. Paulin. "BVH for Efficient Raytracing of Dynamic Metaballs on GPU", *SIGGRAPH 2009 Sketches and Applications*, USA, New Orleans, August 2009.
3. A. Bernhardt, A. Pihuit, M. P. Cani, L. Barthe "Matisse : Painting 2D regions for Modeling Free-Form Shapes", *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*, Annecy, France, pages 57-64, June 2008.
4. L. Claustres, L. Barthe and M. Paulin "Wavelet Encoding of BRDFs for Real-Time Rendering", *Graphics Interface*, Montreal, Canada, May 2007.
5. V. Forest, L. Barthe and M. Paulin "Realistic Soft Shadows by Penumbra-Wedges Blending", *Graphics Hardware*, Vienna, Austria, September 2006.
6. G. Guennebaud, L. Barthe and M. Paulin "Splat/Mesh Blending, Perspective Rasterization and Transparency for Point-Based Rendering", *IEEE/Eurographics/ACM Symposium on Point-Based Graphics*, Boston, US, July 2006.
7. G. Guennebaud, L. Barthe and M. Paulin "Real-time soft shadow mapping by backprojection", *Eurographics Symposium on Rendering*, Nicosia, Cyprus, June 2006.
8. A. Alexe, L. Barthe, M.P. Cani and V. Gaildrat. "Shape modelling by sketching using convolution surfaces", *Pacific Graphics (Short Papers)*, Macau, China, October 2005.

9. A. Alexe, V. Gaildrat and L. Barthe. "Interactive modelling from sketches using spherical implicit functions", AFRIGRAPH 2004, Stellenbosch (Cape Town), South Africa, pages 25-34, November 2004.
10. G. Guennebaud, L. Barthe and M. Paulin. "Real-time point cloud refinement", Eurographics Symposium on Point-Based Graphics, Zurich, Switzerland, pages 41-48, June 2004.
11. L. Barthe and L. Kobbelt. "Direct computation of a control vertex position on any subdivision level", The Mathematics of Surfaces 2003, Leeds, UK, pages 40-47, September 2003.
12. L. Barthe, B. Mora, N.A. Dodgson and M.A. Sabin. "Triquadratic reconstruction for interactive modelling of potential fields", SMI'2002 Shape Modeling International, Banff, Alberta, Canada, pages 145-153, May 2002.
13. M. Cardle, S. Brooks, L. Barthe, M. Hassan and P. Robinson. "Music-Driven motion editing", SIGGRAPH 2002 Sketches and Applications, San Antonio, Texas, US, July 2002.
14. M. Cardle, L. Barthe, S. Brooks and P. Robinson. "Music-Driven motion editing : Local motion transformations guided by music analysis", EGUK'2002 Eurographics UK Conference, Leicester, UK, pages 38-44, June 2002.
15. L. Barthe, V. Gaildrat and R. Caubet. "Implicit extrusion fields : General concepts and some simple applications", SMI'2001 Shape Modeling International, Genova, Italy, pages 114-123, May 2001.
16. L. Barthe, V. Gaildrat and R. Caubet. "Implicit extrusion fields", CISST'2000 International Conference on Imaging Science, Systems and Technology, Las Vegas, U.S., pages 75-81, June 2000.
17. L. Barthe, V. Gaildrat and R. Caubet. "Combining implicit surfaces with soft blending in a CSG tree", CSG Conference Series, Ammerdown, U.K., pages 17-31. April 1998.

Reuves nationales :

1. A.Pajot, L. Barthe and M.Paulin. "Metropolis Instant Radiosity avec reconstruction sur GPU", Revue Electronique Française d'Informatique Graphique, 2009.

Conférences nationales avec comité de lecture et actes publiés :

1. A.Pajot, L. Barthe and M.Paulin. "Metropolis Instant Radiosity avec reconstruction sur GPU", 21th Journées de l'AFIG, Novembre 2008.
2. A. Alexe, V. Gaildrat et L. Barthe. "Modélisation interactive de formes complexes à partir d'esquisses", 16th Journées de l'AFIG, Paris, France, pages 175-182, December 2003.

Présentations longues dans des conférences nationales :

1. L. Barthe. "Les Surfaces de Subdivision", Presentation invitée aux journées Modélisation Géométrique et Traitement d'Images, Grenoble, France, décembre 2009.

2. L. Barthe. "Box-Splines et Surfaces de Subdivision", Presentation aux 18ème Journées de l'AFIG, Strasbourg, France, novembre 2005.

Résumé

Ce manuscrit présente principalement un ensemble de travaux de recherche, ainsi que les tâches administratives ou liées à la vie de la recherche, que j'ai effectuées. Il se termine par ma liste de publications. Les travaux présentés s'étalent sur ces neuf dernières années. Ils portent sur la modélisation géométrique de formes complexes et sont ceux pour lesquels ma participation a été la plus active, voire dont j'ai été l'instigateur.

De nos jours, l'image prend de plus en plus d'importance. Parmi tous les facteurs à l'origine de ce phénomène, nous pouvons noter la prolifération des supports d'affichage (écrans, téléphones portables, tablettes PC, ordinateurs ultra-portables, etc). La demande en création de contenu visuel est croissante et une façon de répondre à cette demande efficacement et à moindre coût passe par la génération de contenu virtuel tridimensionnel. La génération de contenu virtuel s'appuie sur la modélisation géométrique d'objets complexes. Cette phase de création des objets tridimensionnels est encore longue et laborieuse et n'est accessible qu'aux professionnels ou aux passionnés. Ceci est dû, en partie, à la difficulté de définir des modèles de représentation de surfaces qui sont à la fois rapides à visualiser, robustes et accessibles quand on édite la surface (déformation, changement de topologie, etc), simples à implanter, numériquement stables, et permettant une interaction intuitive avec l'utilisateur. En fait, il n'existe pas vraiment de solution satisfaisante au développement rapide de logiciels de modélisation performants et faciles à prendre en main par un utilisateur non-expert ; même s'ils sont contextualisés.

Nous nous intéresserons, dans un premier temps, à l'un des modèles de représentation de surfaces les plus populaires et reconnus pour sa flexibilité et son efficacité dans des applications de modélisation d'objets de formes libres : les surfaces de subdivision. Nous verrons les travaux que j'ai effectués sur ces modèles et nous discuterons de leurs avantages et leurs inconvénients dans notre contexte. Ceci nous amènera à la présentation de travaux sur la modélisation par esquisses, une technique de modélisation basée sur le tracé de contours, reconnue pour son accessibilité d'un point de vue utilisateur. Ces travaux mettront en évidence l'intérêt que l'on peut avoir à développer les modèles volumiques (surfaces implicites) et nous verrons quels travaux ont été menés récemment sur ces modèles pour améliorer leur contrôle et leur efficacité en situation de modélisation. Nous finirons en mettant en perspective le potentiel des modèles volumiques en présentant des projets de recherche qui ont ou vont commencer.

Abstract

In this document I present a summary of some of my research and also my experience with administrative research management tasks. The research presentation is followed by the related significant publications and at the end of the document, I have included the complete list of my publications.

I have conducted research over the last nine years and a unifying theme is the challenge of geometric modeling of complex shapes. Nowadays, images are becoming more and more important in our everyday life. Display devices such as screens, mobile phones, tablet-PCs, notebooks, are ubiquitous. Requests for the production of visual content are thus increasing and an efficient solution relies on the generation of virtual content. This requires the use of very efficient but intricate and complicated modeling software, only accessible to professionals or enthusiast users.

One of the main reason is that no model for representing shape supports fast and immediate rendering, robust editing and simple implementation. Some methods suffer from numerical instability or do not currently support intuitive user interaction. In fact, there is no stable modeling methodology that enables the rapid development of models that is both efficient and intuitive to use for non-expert users.

In the first chapter I present some of my research results on subdivision surfaces, a well-known and recognized surface representation for its flexibility and efficiency in geometric modeling applications. I discuss the advantages and weaknesses with respect to the problems mentioned above. This leads to my investigations into sketch based modeling, an efficient contour based technique with user-friendly interactions for shape modeling. This work demonstrates the usefulness of volumetric representations (implicit surfaces) and I present the recent research we have done for improving their control and enhancing their efficiency in modeling situations. We conclude with our perspectives and projects around volumetric shape representations.