Study of the composition models of field functions in computer graphics

Florian Canezin

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *08/09/2016* **par :**

Florian CANEZIN

Study of the Composition Models of Field Functions in Computer Graphics

---

JURY

Loïc BARTHE	Professeur de l'Université Toulouse 3 Paul Sabatier	Directeur de thèse
Gaël GUENNEBAUD	Chargé de Recherche à INRIA Bordeaux–Sud-Ouest	Co-encadrant
Jean-Michel DISCHLER	Professeur de l'Université de Strasbourg	Examinateur
Mathias PAULIN	Professeur de l'Université Toulouse 3 Paul Sabatier	Examinateur
Stefanie HAHMANN	Professeur de l'Université Grenoble INP	Rapporteur
Éric GALIN	Professeur de l'Université Lumière Lyon 2	Rapporteur

---

École doctorale et spécialité :
MITT
Unité de Recherche :
Intitut de Recherche en Informatique de Toulouse
Directeurs de Thèse :
Loïc BARTHE et *Gaël GUENNEBAUD*
Rapporteurs :
Éric GALIN et *Stefanie HAHMANN*

Remerciements

Je remercie en premier lieu mon directeur de thèse Loïc Barthe pour sa confiance pour mener ce projet, de m'avoir encadré durant ces 3 ans et demi et pour son soutien durant les quelques coups de mou que j'ai pu avoir durant ma thèse. Je remercie également Gaël Guennebaud pour sa participation dans mon encadrement, notamment pour mon séjour sur Bordeaux qui fut des plus plaisant (autant professionellement que personnellement). J'en profite aussi pour les remercier, conjointement à mes rapporteurs Stefanie Hahmann et Éric Galin ainsi que mes examinateurs Jean-Michel Dischler et Mathias Paulin, pour le temps passé sur la (re)lecture de mon manuscrit.

Un grand merci bien sûr aux autres membres des équipes VORTEX de Toulouse et MANAO de Bordeaux avec qui j'ai pu passer d'agréables moments entre discussions sérieuses sur nos projets, séminaires d'équipes et autres divagations : Mathias, David, Xavier, Pierre, Pascal, Romain, Patrick. Merci aux enseignants de l'UPS, Loïc, Mathias, David, Véronique et Christine, et de l'IOGS, Xavier et Ivo, pour m'avoir donné la chance d'enseigner dans différentes matières et aux étudiants pour leur patience durant ces heures de TP. Merci à Anthony, Dorian, Andra et Rodolphe, anciens doctorants VORTEX pour m'avoir chaleureusement accueilli durant mon stage de Master et m'avoir aidé à me mettre dans le bain de la thèse. Merci aussi à Thomas, Charly, Even, Valentin, Anahid, Céline, Maurizio et Baptiste pour la bonne ambiance dans la salle Voxar de l'IRIT. Je n'oublie pas non plus de remercier les doctorants de MANAO : Boris, Brett, Carlos, John, Lois et Thibaud, pour leur accueil et les moments d'évasion au babyfoot ou derrière un verre avec d'autres doctorants de l'INRIA. J'en profite encore pour tous les remercier d'avoir supporté mes coups de colère, contre l'ordi mais surtout contre moi-même, lorsque tout ne se passait pas comme prévu.

Enfin, merci à ma famille et à mes amis pour leur soutien et leurs encouragements. Un merci tout spécial à Florine Dubreuil, ma compagne depuis 2 ans et demi et grâce à qui j'ai pu me détendre et me relaxer en dehors du labo, et à qui je dédie ce manuscrit.

Contents

Introduction

Figure 1: *L'informatique graphique permet la création et l'animation de mondes virtuels pour des applications variées dans l'industrie, les effets spéciaux, l'édition d'images et le divertissement. Images issues de [GGP+15, BMO+14, PHK11, BHK14].*

L'informatique graphique est devenue ces dernières années un domaine très populaire et actif, utilisé aussi bien par des professionnels que par le grand public. Pour les ingénieurs travaillant dans l'industrie, par exemple pour des constructeurs de voitures ou d'avions, l'informatique graphique permet de concevoir des pièces mécaniques et de simuler leur fonctionnement. Aussi, les infographistes utilisent l'informatique graphique pour créer toutes sortes d'objets 3D, allant de la carrosserie d'une voiture à des meubles, d'un environnement virtuel à des êtres qui le peuplent. L'informatique graphique est aussi très utilisée pour la communication et le divertissement, où des personnages et objets peuplent un monde virtuel dans le but de transmettre un message pour une publicité, ou encore une atmosphère particulière pour un film ou un jeu vidéo. Dans cette dernière application, les artistes ont besoin d'un contrôle intuitif des outils qu'ils manipulent pour avoir une liberté d'expression la plus grande possible dans le résultat visuel. Pour cela, l'informatique graphique propose une grande variété d'outils, incluant la modélisation géométrique, l'animation, le rendu réaliste ou non, la simulation physique, la manipulation d'images, etc. (voir Figure 1). Même si ces outils peuvent être basés sur des notions théoriques avancées provenant principalement des mathématiques et de la physique (géométrie, algèbre, statistiques, optique, mécanique ...), ces aspects sont généralement manipulés au travers d'un jeu de paramètres intuitifs et d'une interface dédiée, afin que les utilisateurs, professionnels ou non, n'aient pas à s'en soucier. En raison du besoin croissant de contrôle et de précision, mais aussi de facilité d'utilisation et de rapidité, l'informatique graphique est un vaste domaine de recherche en effervescence depuis plusieurs années.

Les objets virtuels sont la plupart du temps représentés par leur surface, dont la forme générale peut être fixe comme pour un personnage, ou dynamique comme pour un fluide. Différentes représentations pour les objets sont disponibles et certaines sont illustrées en Figure 2. Lorsque l'on souhaite représenter des objets 3D dans un environnement virtuel, les *maillages* sont sûrement la représentation la plus utilisée. Un maillage consiste en un ensemble de faces 2D, connectées entre elles par leurs arêtes et leurs sommets, qui approche la surface de l'objet par morceaux. Les maillages ont l'avantage d'être simples à exprimer et rapides à visualiser, par lancer de rayon ou rastérization. Ils restent cependant une approximation de la surface et sont donc limités par la résolution à laquelle ils l'échantillonnent. De ce fait, les maillages ne peuvent représenter une surface présentant des détails de haute fréquence ou une surface lisse qu'avec une résolution très fine. Cela empêche l'apparition d'artéfacts lors de la visualisation comme des contours plats ou des détails dégradés mais produit des maillages très coûteux en mémoire et bien plus lents à visualiser. De plus, manipuler directement un maillage pour obtenir la forme désirée pour l'objet est très fastidieux et de nombreux algorithmes ont du être conçus pour en faciliter la manipulation. Enfin, les maillages sont une représentation surfacique d'un objet, et à ce titre ne donnent aucune information explicite sur le volume de l'objet, rendant l'assemblage par opérations booléennes d'union, d'intersection et de différence extrêmement difficile et coûteux.

Un maillage peut aussi servir de base pour définir un niveau de détail de la surface d'un objet. Ce niveau de détail peut être raffiné itérativement avec un schéma de subdivision pour obtenir la résolution désirée. Ce type de représentation, appelée *surface de subdivision* est très intéressant pour la modélisation géométrique multirésolution ou dès lors que l'on souhaite garantir certaines propriétés de la surface limite comme la continuité ou la localisation. Étant donné que les surfaces de subdivision résultent en un maillage pour chaque niveau de détail, elles permettent un contrôle intuitif des détails à chaque résolution. Ceci fait d'elles une représentation très intéressante pour la visualisation d'un objet et la modélisation géométrique, où un contrôle fin de la surface d'un objet est nécessaire. Cependant, même si elles permettent de mieux approcher la surface d'un objet, les surfaces de subdivision en restent une approximation et ne proposent pas d'information volumique.

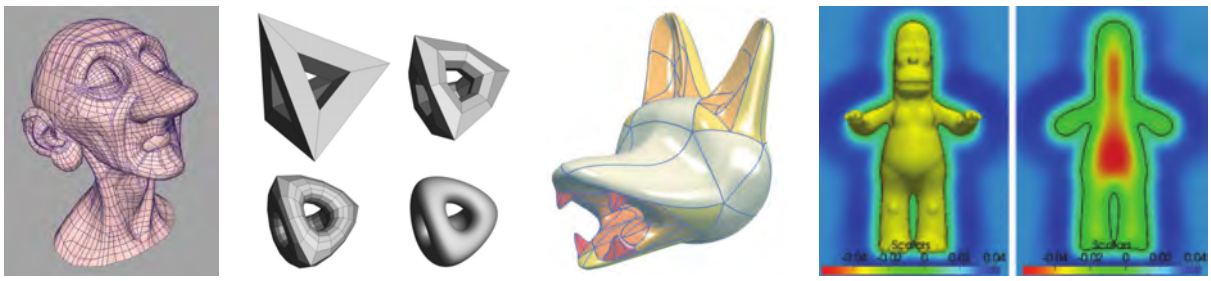


Figure 2: *Différentes représentations d'objets en informatique graphique. De gauche à droite : maillage, surface de subdivision, surface paramétrique et surface implicite. Images issues de [DKT98, YHB05, MGV11].*

Une autre représentation explicite de la surface d'un objet consiste à assembler des parties de surface définies à partir de points de contrôle et d'une équation d'en général deux paramètres. Ce sont les *surfaces paramétriques*. Elles sont surtout utilisées par les professionnels de l'industrie car elles permettent de définir de fortes contraintes sur les surfaces et offrent un contrôle fin et précis de la surface et de ses propriétés. De la même façon que pour les maillages et les surfaces de subdivision, les surfaces paramétriques n'offrent pas d'information explicite sur le volume de l'objet qu'elles représentent.

En marge des représentations pûrement surfaciques, on trouve les représentations volumiques, dont les *surfaces implicites*. Les surfaces implicites ont cet avantage d'offrir une représentation continue de la surface et des informations volumiques explicites en même temps. Une surface implicite est définie par une fonction scalaire sur les coordonnées de l'espace ambiant euclidien dans lequel est défini l'objet. Lorsque la surface de l'objet est fermée, une surface implicite la décrit comme l'interface entre son intérieur et son extérieur. Les surfaces implicites sont surtout utilisées pour la reconstruction de surface à partir de nuages de points issus de scanners, la représentation d'objets hautement dynamiques tels que les fluides, ou encore la modélisation géométrique par aggrégation de matière. En effet, elles sont par nature idéales pour des opérations booléennes d'union, d'intersection et de différence, surtout lorsque une transition dite "douce" par ajout de matière est désirée, comme pour un assemblage organique. Les surfaces implicites restent cependant une représentation peu utilisée pour la modélisation géométrique en raison de la difficulté pour les visualiser efficacement et présicément, et du manque de paramétrization 2D de la surface qui permettrait notamment de la texturer. Les surfaces implicites n'en restent pas moins une représentation puissante offrant un processus de modélisation intuitif et efficace par la combinaison des informations volumiques : des parties d'objets définies par l'utilisateur sont assemblées automatiquement afin de créer la surface de l'objet désiré, avec des arêtes franches pour les pièces mécaniques ou des jonctions douces pour des parties organiques. Bien que des travaux récents améliorent le contrôle utilisateur et offrent de meilleures propriétés de surface pour des systèmes de modélisation par assemblage, les surfaces implicites présentent encore des problèmes de stabilité devant être résolus avant de pouvoir être considérées comme une représentation pertinente d'objets en informatique graphique.

Contexte de l'étude

Le but de cette thèse est de résoudre certaines limitations encore présentes des surfaces implicites afin de déterminer si, avec le modèle de composition avantageux qui les accompagne, leur potentiel peut être totalement exploité en informatique graphique. Pour cela, nous expérimentons leur mise en application dans le cadre de la modélisation géométrique et de la simulation de fluides par particules, notamment la reconstruction de la surface du fluide.

Dans un processus de modélisation géométrique par surfaces implicites, celles-ci représentent des composants d'assemblage correspondant à des parties de l'objet final. Ces composants sont assemblés les uns avec les autres au travers d'opérations booléennes exprimées par des

opérateurs de composition. Un contrôle de plus en plus fin de la forme de l'objet final est donné à l'utilisateur à travers un certain nombre de paramètres. En général, le processus de modélisation est incrémental, assemblant les composants par paire et utilisant donc des opérateurs binaires. Cependant, il arrive que plusieurs composants doivent être assemblés en un même endroit. Dans ce cas, des opérateurs n-aires doivent être utilisés et garder un contrôle fin du résultat de l'assemblage devient très ardu lorsque le nombre de composants augmente. Aujourd'hui la définition des surfaces implicites et des modèles de composition n'est pas totalement unifiée : aucune garantie n'est donnée sur la conservation des propriétés des surfaces des composants, surtout lorsque des intersections et des différences sont appliquées. Cela rend l'ensemble du processus d'assemblage inconsistant et peut entraîner l'apparition imprévisible d'artéfacts indésirables sur la surface, tels que des trous ou des discontinuités.

En simulation de fluides par particules, ces dernières échantillonnent le volume du fluide et se déplacent selon les équations de physique qui traduisent le mouvement du fluide. Ici, les surfaces implicites représentent la contribution des particules au volume du fluide, c'est-à-dire les portions de fluide induites par les particules. Ces contributions sont en premier lieu utilisées pour calculer les interactions des particules au sein du fluide pendant la simulation, puis pour reconstruire la surface du fluide afin de la visualiser. Alors que de plus en plus de réalisme est attendu de la simulation (viscosité, tension de surface) et de la reconstruction (fine pellicule de fluide, surface lisse), ces deux étapes sont en général réalisées indépendemment à chaque pas de simulation. De ce fait des incohérences temporelles et topologiques peuvent survenir dans le comportement de la surface du fluide, comme des attractions et mélanges de composants de fluides distincts et une gestion non naturelle des fusions et séparations.

CONTRIBUTIONS

La première contribution de cette thèse concerne le problème de consistance entre les surfaces implicites et les modèles de composition tels qu'utilisés en modélisation géométrique. Nous proposons tout d'abord la prise en compte de propriétés supplémentaires par les modèles de surfaces implicites afin que celles-ci puissent supporter toutes les opérations booléennes. Ensuite, nous définissons de nouvelles contraintes pour les opérateurs de composition afin qu'ils adhèrent à ces propriétés et que les relations entre opérations booléennes puissent être retranscrites sur les opérateurs de composition. Enfin, nous introduisons un nouvel opérateur conçu spécifiquement pour la gestion de l'ajout de petits détails à un objet.

La seconde contribution concerne la gestion temporelle et topologique rencontrée dans les simulations de fluides par particules et la reconstruction de la surface des fluides. Nous commençons par introduire un nouvel opérateur de composition n-aire pour la reconstruction de la surface du fluide. Cet opérateur se base sur une structure en graphe représentant les voisinages topologiques locaux des particules, qui correspondent aux composantes du fluide restreintes à l'échelle des particules. Par l'utilisation de ce graphe et son suivi au

cours de la simulation, la reconstruction permet une cohérence temporelle et topologique du comportement de la surface du fluide. Nous expliquons ensuite comment ce graphe est mis à jour puis nous proposons de l'utiliser pour coupler la simulation avec la reconstruction afin d'éviter un comportement du fluide non naturel.

Structure du document

Dans le Chapitre 1 nous commençons par donner les notions mathématiques nécessaires à la compréhension du fonctionnement des surfaces implicites et présentons quelques familles usuelles en informatique graphique. Nous présentons ensuite le modèle général de composition qui les accompagne avant de donner un état de l'art sur les opérateurs de composition dans le Chapitre 2. Nous explorons ensuite l'utilisation des surfaces implicites en modélisation géométrique dans le Chapitre 3 et en simulation de fluides par particules dans le Chapitre 4. Enfin, le Chapitre 5 fait la synthèse des contributions de cette thèse et donne diverses directions de recherche pour des travaux futurs.

Chapter 1
Background on Implicit Surfaces

Implicit surfaces are a powerful 3D object representation for computer graphics, providing both a continuous surface definition as well as volume information. In this chapter, we present the mathematical background on implicit surfaces and the main families used in computer graphics to represent objects.

1.1 Definitions

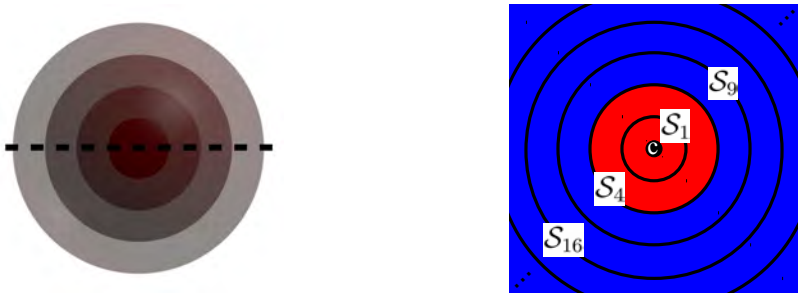
An implicit surface $\mathcal{S}_\mathcal{C}$ is defined as the isosurface, corresponding to the isovalue $\mathcal{C}$, of the scalar field generated by a field function f. Traditionally, the field function f is a scalar-valued function of the 3D Euclidean space coordinates:

$$\begin{array}{rccc} f: & \mathbb{R}^3 & \to & \mathbb{R} \\ & \mathbf{p} = (x, y, z)^t & \mapsto & f(\mathbf{p}) \end{array}$$

An implicit surface $\mathcal{S}_\mathcal{C}$ then corresponds to the set of points in the 3D Euclidean space to which the field function f associates the same isovalue $\mathcal{C}$:

$$\mathcal{S}_\mathcal{C} = \{\mathbf{p} \in \mathbb{R}^3 \mid f(\mathbf{p}) = \mathcal{C}\}$$

As an example, let us consider the following field function: $f(\mathbf{p}) = \|\mathbf{p} - \mathbf{c}\|^2$, illustrated in Figure 1.1. This field function generates a spherical distance field around the point $\mathbf{c}$ whose isosurfaces $\mathcal{S}_\mathcal{C}$ correspond to different spheres of center $\mathbf{c}$ and radius $\sqrt{\mathcal{C}}$. For example, the implicit surface $\mathcal{S}_1$ is a sphere of center $\mathbf{c}$ and radius 1, while $\mathcal{S}_4$ has a radius of 2 and $\mathcal{S}_9$ a radius of 3, as can be seen on the planar section passing through the center $\mathbf{c}$ exposed in Figure 1.1(b). Here the black circles represent some of the generated implicit surfaces while



(a) (b)

Figure 1.1: *Implicit surfaces generated by (a) the field function $f(\mathbf{p}) = \|\mathbf{p} - \mathbf{c}\|^2$ and (b) a planar section passing through $\mathbf{c}$ illustrating their location around it.*

the red colored region corresponds to the interior of the sphere described by $\mathcal{S}_4$ and the blue region to its exterior.

Implicit surfaces provide volume information about the object they represent, given that this volume $\mathcal{V}$ exists. This volume can be defined as $\mathcal{V}_\mathcal{C} = \{\mathbf{p} \in \mathbb{R}^3 \mid f(\mathbf{p}) < \mathcal{C}\}$ as for our previous example. Note that this definition depends on the convention for the field function and the opposite definition $\mathcal{V}_\mathcal{C} = \{\mathbf{p} \in \mathbb{R}^3 \mid f(\mathbf{p}) > \mathcal{C}\}$ can also be used. The implicit surface $\mathcal{S}_\mathcal{C}$ then splits the Euclidean space into two regions: the volume $\mathcal{V}_\mathcal{C}$, e.g. the interior of the object, which is enclosed in $\mathcal{S}_\mathcal{C}$, and the exterior of the object. This is illustrated in Figure 1.1(b) where the interior region corresponding to $\mathcal{V}_4$ is colored in red and the exterior region in blue. $\mathcal{S}_\mathcal{C}$ is called "implicit" because the points on the surface are not explicitly defined by a closed-form formula. All we can do is, from a given point, question the field function to know where that point lies, say if it is on the surface or, if not, on which side of it.

An interesting property of implicit surfaces is that they provide direct computation for the normal vector to an isosurface, which is used for example for lighting the object during the visualisation. The unit normal N to the implicit surface $\mathcal{S}_{\mathcal{C}_\mathbf{p}}$ given at a point $\mathbf{p}$ is computed as the normalized gradient of the field function at that point:

$$\nabla f(\mathbf{p}) = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z} \end{bmatrix}^t$$

Its direction might need to be inverted to accommodate for the inside/outside convention since the gradient points in the direction of growing values. Thus, for our previous example where $\mathcal{V}_\mathcal{C} = \{\mathbf{p} \in \mathbb{R}^3 \mid f(\mathbf{p}) < \mathcal{C}\}$, we have:

$$N(\mathbf{p}) = \frac{\nabla f(\mathbf{p})}{\|\nabla f(\mathbf{p})\|}$$

Once the object is designed, it must be placed into the virtual world. We thus need a way to move and orientate it. Directly modifying the equation of the field function f_1 into f_2 so that the object ends at the expected location can be rather cumbersome. Instead, we use a transformation matrix T encoding affine transformations such as translation, rotation and scaling. What is expected is that each point $\mathbf{p}$ of the object, either inside, outside or on the surface, is moved to the transformed point $\mathbf{p}' = T\mathbf{p}$. This means that the scalar value of f_2 at $\mathbf{p}'$ must be the same as the value of f_1 at $\mathbf{p}$. To find the value $f_2(\mathbf{p}')$, we start from point $\mathbf{p}'$ and apply the inverse transform T^{-1} to it and thus retrieve the point $\mathbf{p}$. We then evaluate the field function f_1 at $\mathbf{p}$ and set the result as the field value for f_2 at $\mathbf{p}'$ as illustrated in Figure 1.2(a). This is expressed in the following formula, along with the gradient computation, and can be seen as a change of the reference system:

$$f_2(\mathbf{p}') = f_1(\mathbf{p}) = f_1(T^{-1}\mathbf{p}') \qquad \nabla f_2(\mathbf{p}') = T^{-t}\nabla f_1(T^{-1}\mathbf{p}')$$

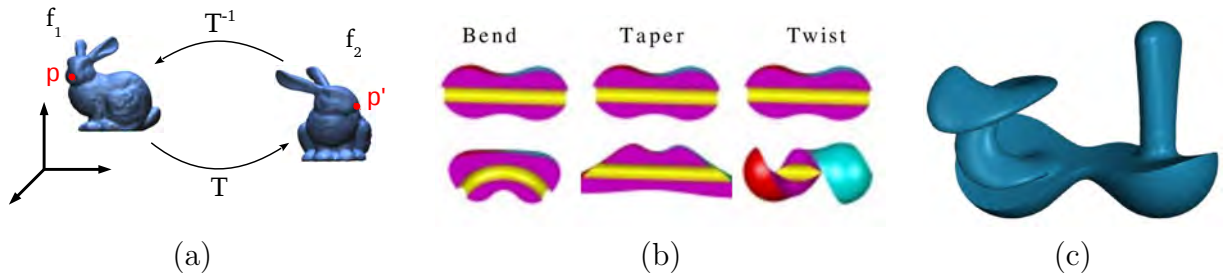


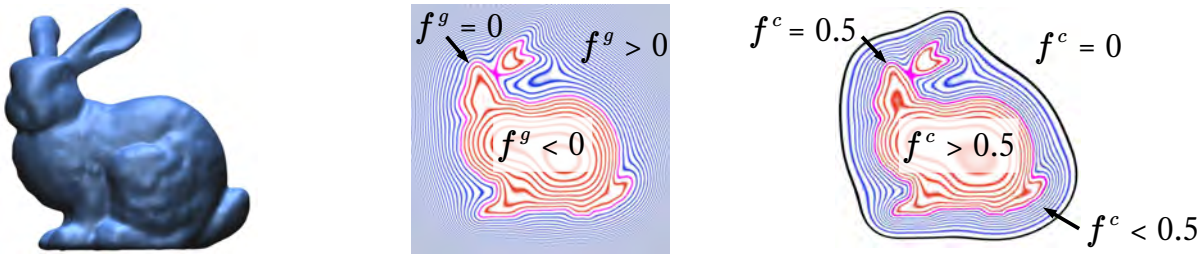
Figure 1.2: *Illustration of the transformation process for implicit surfaces. (a) The new field value after transformation for point* $\mathbf{p}'$ *is computed from point* $\mathbf{p} = T^{-1} * \mathbf{p}'$*. Using this transformation process, objects can be put and oriented in the virtual world (a) and can also be deformed using space warping transformations (b-c) [WGG99, dGWvdW09].*

Note that the normal is transformed by the inverse transpose matrix T^{-t} to conserve surface orthogonality in the case of non-uniform scaling. Any invertible transformation matrix T can be used this way, not only for affine transformations but also for more complex ones, thus leading to the deformation of the object using space warping transformations such as bend, taper and twist as shown in Figure 1.2(b).

Implicit surfaces can be classified into two main categories depending on the support of the field function they rely on: *globally* supported field functions or *compactly* supported field functions. The field generated by a globally supported field function f^g varies all over the Euclidean space, as for the previous distance example. On the other hand, the field generated by a compactly supported field function f^c only varies in a delimited region and is constant everywhere else. Conventions for each category are the following. For globally supported field functions the implicit surface is defined for $\mathcal{C} = 0$ and both volume formulations can be found depending on the model. The convention for compactly supported field functions is generally to consider $\mathcal{C} = 0.5$ with $\mathcal{V}_\mathcal{C} = \{\mathbf{p} \in \mathbb{R}^3 \mid f(\mathbf{p}) > \mathcal{C}\}$. An additional setting for compactly supported field functions is that beyond a certain distance to the surface in the "outside" region, the field function vanishes and uniformly equals 0.

The two categories are illustrated in Figure 1.3 where they define the same surface. In order to better understand the relationship between the field function f and the implicit surface $\mathcal{S}_\mathcal{C}$, let us explain the field visualisation of Figure 1.3. Consider a planar section of the object defined by $\mathcal{S}_\mathcal{C}$. The intersections of the cutting plane and the different isosurfaces of the field generated by f is a set of 2D curves in this plane. In particular, one of these curves is the one generated by the implicit surface $\mathcal{S}_\mathcal{C}$, which is colored in magenta, while others may lie inside the object, in red, or outside it, in blue. Also, the boundary of the field variation of compactly supported field functions is drawn in black.

In the remainder of this chapter, we present the main families of implicit surfaces that are used in computer graphics.



(a) (b) (c)

Figure 1.3: *Planar sections of an implicit surface representing a bunny (a). Blue curves are outside the bunny, red curves are inside it and the magenta curve is the section of the implicit surface. For globally supported field functions (b) the field varies in the whole space while for compactly supported field functions (c) its variation is limited into a region around the surface. Conventions for both globally and compactly supported field functions are also given.*

1.2 Algebraic Surfaces

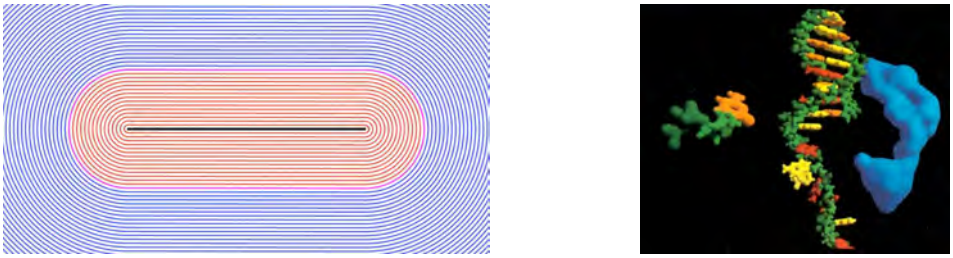
Algebraic surfaces are the most basic implicit surfaces in terms of formulation. They rely on an equation of the Euclidean coordinates that generally produces a globally supported field function. A simple example has already been given with the distance function and some more algebraic surfaces are shown in Figure 1.4 with their respective equation. Though they are capable of representing all kinds of surfaces, these formulations are not practical for the design of 3D objects which require intuitive surface control. Indeed, finding the exact equation of the desired object surface and modifying it to add details is very cumbersome. They are still useful when defining simple primitives to be assembled.

$x^4 - 5x^2 + y^4 - 5y^2 + z^4 - 5z^2 + 3$ $x^3 + y^3 + z^3 + kxyz$ $x^3y + y^3z + z^3x - a + 0.5$

Figure 1.4: *Implicit surfaces generated by algebraic functions of the Cartesian coordinates x, y and z. Figures from [RS08, HL08, Kra].*

1.3 Skeleton-based Implicit Surfaces

Considering how difficult it is to tune the equation of an algebraic surface to get the expected result, modelling a complex object this way is inefficient. Instead, field functions based on the distance to a skeleton give more control to the user through the modification of the skeleton, which is a meaningful abstraction of the overall shape of the object. Skeleton based field functions can be defined from different skeletons such as a simple point (as in



(a) (b)

Figure 1.5: *(a) Directly using the distance to a skeleton generates globally supported field functions. (b) Composing a Gaussian kernel to the distance to display molecules [Bli82].*

the spherical distance field example), poly-lines, curves, polygons or even polyhedrons, as soon as the distance to the skeleton can be computed. Directly using a distance $dist$ to a skeleton S to build the skeleton based field function f can lead to the following formula: $f(\mathbf{p}) = d_0 - dist(\mathbf{p}, S)$, where d_0 is the distance from the skeleton to the desired surface. This definition, illustrated in Figure 1.5(a), generates a globally supported field function.

In order to bring compact support and then local restriction properties into the field generated by the implicit surface and control them when using complex skeletons, the distance-based field functions are extended by composing a kernel function K to the distance:

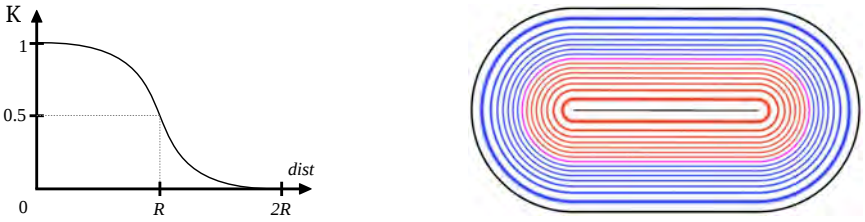
$$f(\mathbf{p}) = K(dist(\mathbf{p}, S)) \tag{1.1}$$

The first kernel-based definition of an implicit surface was introduced by Blinn [Bli82] for the visualisation of "blobby" molecules (Figure 1.5(b)), through a Gaussian kernel inspired by the electromagnetic properties of atoms:

$$K(d) = \exp\left(-\left(\frac{d}{\sigma}\right)^2\right) \tag{1.2}$$

where σ controls the slope of the Gaussian, and then the size of the atom inside the molecule. The resulting field function f is another kind of globally supported field function: it varies all over the space but is positive and $\mathcal{C} \neq 0$. Several other kernels have been designed through piecewisely defined polynomial functions, which are faster to evaluate, in order to define compactly supported field functions (see Figure 1.6), whose variation support is controlled by the parameter σ:

- Metaballs [NHK+85]: $K(d) = \begin{cases} 1 - 3\left(\frac{d}{\sigma}\right)^2 & \text{if } 0 \leq d \leq \frac{\sigma}{3} \\ \frac{3}{2}\left(1 - \frac{d}{\sigma}\right)^2 & \text{if } \frac{\sigma}{3} \leq d < \sigma \\ 0 & \text{otherwise} \end{cases}$
- Soft Objects [WMW86]: $K(d) = \begin{cases} 1 - \frac{4}{9}\left(\frac{d}{\sigma}\right)^6 + \frac{17}{9}\left(\frac{d}{\sigma}\right)^4 - \frac{22}{9}\frac{3}{2}\left(1 - \frac{d}{\sigma}\right)^2 & \text{if } d < \sigma \\ 0 & \text{otherwise} \end{cases}$



(a) (b)

Figure 1.6: *Composing a fall-off kernel K (a) with the distance allows to define compactly supported field functions (b).*

- Blobs [Blo97]: $K(d) = \begin{cases} \left(1 - \left(\frac{d}{\sigma}\right)^2\right)^3 & \text{if } d < \sigma \\ 0 & \text{otherwise} \end{cases}$

The main drawback of kernel-based distance field functions is their low degree of continuity for other skeletons than points. Indeed, for a point skeleton the field function is C^∞ continuous everywhere inside its support except at the point. In the contrary, for more complex skeletons this is not the case anymore: the field is spherical at points proximity, cylindrical at segments proximity and planar at faces proximity. Hence, the gradient of the field is continuous but not differentiable, and then the surface is only C^1 continuous.

1.4 Convolution-based Implicit Surfaces

An extension of the kernel-based distance field functions aims at increasing the field continuity and fairness regardless of the skeleton type through convolution. Here, instead of composing the kernel K with the distance, it is convolved on the skeleton, thus defining the field function as the integral of the contributions of all the points belonging to the skeleton:

$$f(\mathbf{p}) = \int_{s \in \Omega} K(\|\mathbf{p} - \Gamma(s)\|) ds \tag{1.3}$$

where Ω is the skeleton parametrisation and $\Gamma(s)$ is the skeleton point corresponding to parameter s. Thanks to the additive property of integration, convolution-based implicit surfaces are independent of the skeleton subdivision as illustrated in Figure 1.7. This property allows the design of arbitrarily complex skeletons defined from a set of primitive elements such as points, segments and triangles.

While the first convolution surfaces used Gaussian kernels [Blo97], subsequent works provided kernels with closed-form expressions [MS98, She99] for both convolution along line segment and triangle skeletons. Kernels can then be organized into three families [HC12]:

- Inverse of order i: $K(d) = \frac{1}{\left(\frac{d}{\sigma}\right)^i}$

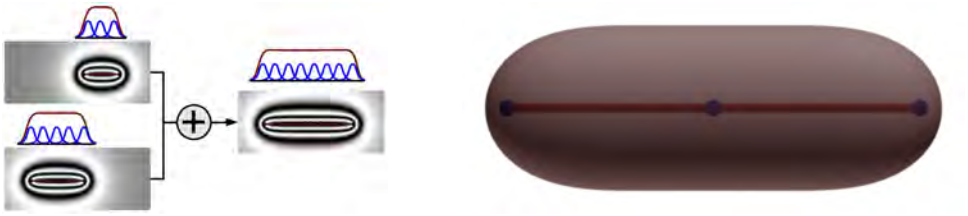


Figure 1.7: *Convolution surfaces are skeleton subdivision independent: summing contributions from skeleton parts is the same as the contribution of the skeleton as a whole. Figures from [Zan13].*

- Cauchy of order i: $K(d) = \frac{1}{\left(1+\left(\frac{d}{\sigma}\right)\right)^{\frac{i}{2}}}$
- Compact polynomial of order i: $K(d) = \begin{cases} \left(1-\left(\frac{d}{\sigma}\right)^2\right)^{\frac{i}{2}} & \text{if } d < \sigma \\ 0 & \text{otherwise} \end{cases}$

the two first ones generating C^∞ continuous globally supported field functions, and the latter generalizing the Blobs kernel to any order i.

Convolution surfaces have also been extended to handle non-constant radius along the skeleton [JT02, HAC03, ZBQC13] using a weighting function τ:

- to scale K: $f(\mathbf{p}) = \int_{s\in\Omega} \tau(s) K(\|\mathbf{p} - \Gamma(s)\|) ds$
- to scale d: $f(\mathbf{p}) = \int_{s\in\Omega} K\left(\frac{\|\mathbf{p}-\Gamma(s)\|}{\tau(s)}\right) ds$
- to scale both: $f(\mathbf{p}) = \int_{s\in\Omega} \frac{1}{\tau(s)} K\left(\frac{\|\mathbf{p}-\Gamma(s)\|}{\tau(s)}\right) ds$

Using radius variation allows the design of various shapes from the same skeleton, and in particular sharp tips as illustrated in Figure 1.8. The main advantage of convolution surfaces is the organic shape of the surface obtained thanks to the natural blending of skeleton-defined parts. However, they cannot represent sharp features other than tips, and then are not suited for the design of non-organic shapes.

Figure 1.8: *Organic shapes modelled using radius varying convolution surfaces. Figures from [JT02, ZBQC13].*

1.5 Point Set Implicit Surfaces

While implicit surfaces traditionally represent parts of the object's volume that are combined, the surface can also be defined from points on it. Point set implicit surfaces model complex free-form objects by fitting a field function from constrained surface points coming from 3D scanners or meshes with structural problems such as holes or self-intersections. Such representations are very popular for the digitisation of architectural heritage as well as augmented and virtual reality purposes. Here, the final field function $f : \mathbb{R}^3 \to \mathbb{R}$ for the object is expected to be such that $f(\mathbf{p}_i) = \mathcal{C}$ for all input surface points $\mathbf{p}_i$, with the additional constraint that f must be analogous to a distance field function.

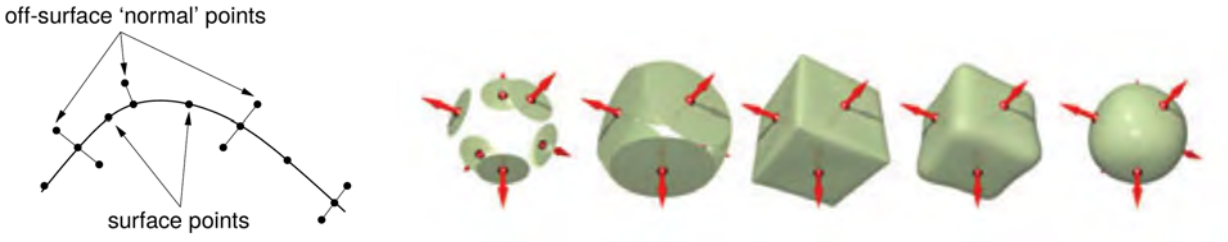
Radial Basis Functions: A first approach, called Radial Basis Functions (RBF), is to discretize the field function f as a linear combination of basis functions ϕ centered at the input surface points [CBC+01]:

$$f(\mathbf{p}) = \sum_{i}^{N} \alpha_i \phi(\|\mathbf{p} - \mathbf{p}_i\|)$$

The unknown scalar coefficients α_i are then found by solving the system of equations given by the constraints $f(\mathbf{p}_i) = 0$. However, the trivial solution $\alpha_i = 0$ leads to the zero constant function $f = 0$, thus unable to represent the object surface. In order to avoid this shortcoming, more constraints have to be set on "off-surface" points lying outside the surface as illustrated in Figure 1.9(a), and for which another field value $\mathcal{C}_i$ is expected: $f(\mathbf{p}_i + \varepsilon \mathbf{n}_i) = \mathcal{C}_i$, where $\mathbf{n}_i$ is the surface normal at $\mathbf{p}_i$. A more robust and elegant solution called Hermite RBF [Wen04, MGV11] consists in setting constraints on the gradient of the generated field so that it matches the surface normal at points $\mathbf{p}_i$ (see Figure 1.9(b)), leading to the following problem reformulation:

$$f(\mathbf{p}) = \sum_{i}^{N} \alpha_i \phi(\|\mathbf{p} - \mathbf{p}_i\|) + \beta_\mathbf{i}{}^t \nabla \phi_i(\|\mathbf{p} - \mathbf{p}_i\|)$$

with additional constraints $\nabla f(\mathbf{p}_i) = \mathbf{n}_i$.



(a) (b)

Figure 1.9: *Radial Basis Functions solve constraints defined on a set of surface points and either off-surface points (a) or their surface normal (b). Using different functions ϕ leads to different interpolation results (b). Figures from [CBC+01, MGV11].*

Several choices can be made for the basis function ϕ depending on the desired interpolation properties and the dimension of the ambient space:

- Thin plates (only in 2D): $\phi(d) = d^2 ln(d)$
- Polyharmonic splines: $\phi(d) = d^k$, $k = 1, 3, 5...$ or $\phi(d) = d^k ln(d)$, $k = 2, 4, 6...$
- Gaussian: $\phi(d) = e^{-cd^2}$
- Multiquadric and inverse quadric: $\phi(d) = \sqrt{1 + (cd)^2}$ and $\phi(d) = \frac{1}{1+(cd)^2}$

which are globally supported basis functions. The RBF formulation leads to the resolution of a huge linear system with a complexity in $O(N^3)$. A set of compactly supported basis functions have been introduced [Wen95, Wu95, Buh01] in order to make the system sparse, then faster to solve, and also reduce the number of input surface points needed for evaluation.

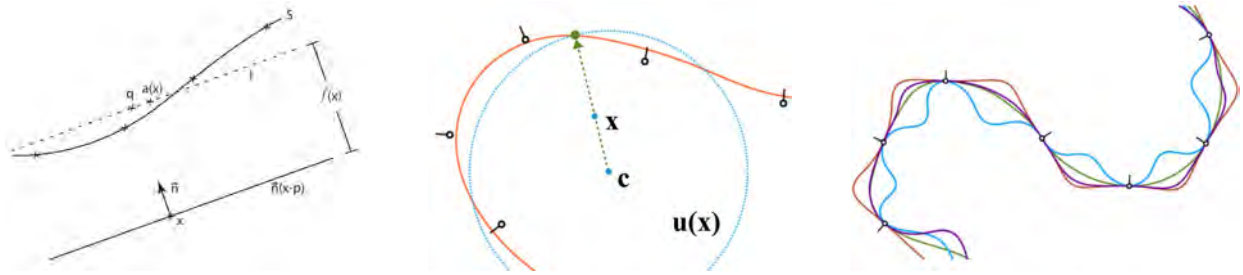
Moving Least Squares: A second approach called Moving Least Squares rather fits a non-null algebraic primitive $a_{\mathbf{p}}$:

$$f(\mathbf{p}) = a_{\mathbf{p}}(\mathbf{p})$$

approximating the local neighborhood of the evaluation point $\mathbf{p}$ in a weighted least squares way:

$$a_{\mathbf{p}} = argmin_a \sum_i^N w(\|\mathbf{p} - \mathbf{p}_i\|)\|a(\mathbf{p}_i)\|^2 \tag{1.4}$$

where w is a compactly supported weighting function whose support radius determines the degree of approximation. The larger the radius, the smoother the details and noise. Moving Least Squares methods have been applied to some algebraic primitives (see Figure 1.10), leading to the following field function expressions:



(a) (b) (c)

Figure 1.10: *Moving Least Squares methods can fit primitives such as planes (a) and spheres (b) from the local neighborhood of the evaluation point. Different interpolation results can be seen in (c) where the blue, red and purple curves fit planes while the green curve fits spheres. Figures from [SOS04, GG07].*

- plane [AA04]:

$$f(\mathbf{p}) = \frac{\sum_i^N w_i \mathbf{n}_i}{\|\sum_i^N w_i \mathbf{n}_i\|}^t (\mathbf{p} - \frac{\sum_i^N w_i \mathbf{p}_i}{\sum_i^N w_i}) = \mathbf{n}(\mathbf{p})^t(\mathbf{p} - \mathbf{a}(\mathbf{p}))$$

  where $\mathbf{a}(\mathbf{p})$ and $\mathbf{n}(\mathbf{p})$ define the fitted plane by averaging neighbor positions and normals.

- plane [SOS04]:

$$f(\mathbf{p}) = \frac{\sum_i^N w_i \mathbf{n}_i^t(\mathbf{p} - \mathbf{p}_i)}{\sum_i^N w_i}$$

  Here the function f is rather defined by averaging the planar fields defined by the neighbors and their normal.

- sphere [GG07, GGG08]:

$$a(\mathbf{p}) = [1 \quad \mathbf{p}^t \quad \mathbf{p}^t\mathbf{p}] \; u(\mathbf{p})$$

  where $u(\mathbf{p})$ is the vector of the five parameters for the algebraic sphere, which are computed as follows. u_1, u_2, u_3 and u_4 are given by minimizing the following gradient constraint: $u(\mathbf{p}) = argmin_u \sum_i^N w_i ||\nabla a(\mathbf{p}_i) - \mathbf{n}_i||^2$. u_0, which does not appear in the gradient formulation, is then retrieved by solving equation 1.4. In the case of non-oriented surface normals at input points [CGBG13], the constraint on the gradient can be replaced with a generalized eigenvalue problem: $u(\mathbf{p}) = argmax_u \sum_i^N w_i \|\nabla a(\mathbf{p}_i)^t \mathbf{n}_i\|^2$ subject to $\sum_i^N w_i \|\nabla a(\mathbf{p}_i)\|^2 = 1$.

Chapter 2
Composition Models

In the previous chapter we have seen how to represent objects with implicit surfaces. While giving the user some control over the creation of simple objects, these models are difficult to manipulate when designing a complex object made of several parts like living beings or mechanical stuff. Fortunately implicit surfaces are a volumetric representation, which makes them perfectly suited for volumetric operations. A modelling system using such operations can then be built through a composition process combining together parts of the final object. In this chapter, we present a background on this composition process followed by a state of the art on the composition models developed in computer graphics. We also list some remaining limitations to be addressed in order to fully take advantage of the power of implicit surfaces and composition models.

2.1 Definitions

The major issue with implicit surfaces is the difficulty to precisely control how the different parts of the final object behave all together. Here, implicit surfaces represent primitive components which are combined to build the final object. The composition process is formalized by a composition operator g over n primitives:

$$\begin{array}{rccc} g: & \mathbb{R}^n & \to & \mathbb{R} \\ & X=(x_1,...,x_n) & \mapsto & g(X) \end{array}$$

Composition operators must be specifically defined for each implicit surface convention. When composed with a set of field functions $\{f_1,...,f_n\}$, the operator g yields a new field function F:

$$\begin{array}{rccc} F: & \mathbb{R}^3 & \to & \mathbb{R} \\ & \mathbf{p}=(x,y,z)^t & \mapsto & F(\mathbf{p})=g(f_1(\mathbf{p}),...,f_n(\mathbf{p})) \end{array}$$

with the same convention, thus making the composition process a unified representation.

Basic and easy to understand composition operations are Boolean operations: union, intersection and difference that are volumetric operations that produce sharp features between the input surfaces as illustrated in Figure 2.1-top row. Such operators were first introduced by Ricci [Ric73] to build a Boolean composition model where "$\max_i f_i$" and "$\min_i f_i$" implement union and intersection operators according to the interior/exterior convention. The difference operation between two primitives is then defined using Boolean identities as the intersection of the first primitive with the complement of the second, i.e. $g_\setminus(f_1,f_2)=g_\cap(f_1,\neg f_2)$. The complement operation has the effect of inverting interior and exterior parts for an object

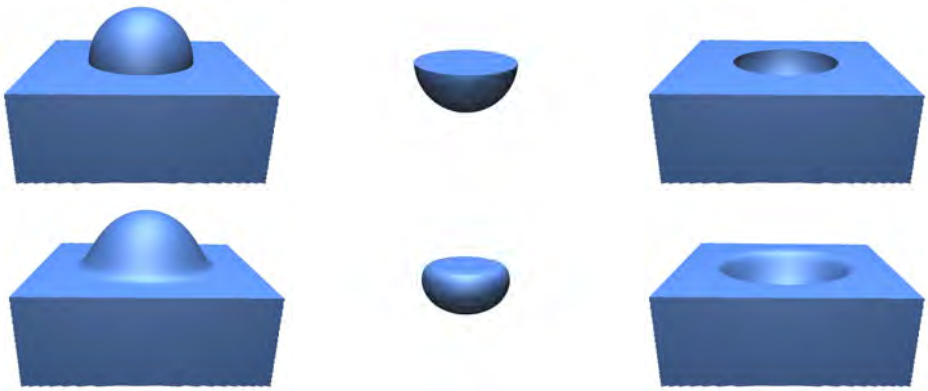


Figure 2.1: *Application of the Boolean operations (top row) of union, intersection and difference on a cube and a sphere, producing a sharp transition between the input surfaces. Blending operations (bottom row) allow for smooth transition shape for both union, intersection and difference.*

without modifying its surface nor the field properties inside these regions except the gradient direction which is also inverted. For the general case, the complement operation is just a symmetry expressed as $\neg f = 2\mathcal{C} - f$ in order to keep the same isovalue for the surface. In the case of globally supported field functions f^g, where the surface is defined for $\mathcal{C} = 0$, the complement is as simple as $\neg f^g = -f^g$. On the other hand, for compactly supported field functions f^c, for which $\mathcal{C} = 0.5$, the complement operation boils down to $\neg f^c = 1 - f^c$.

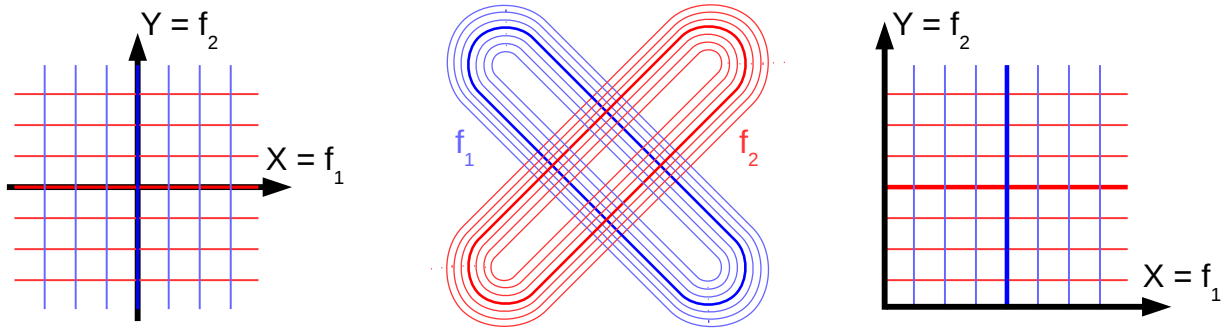
While intuitive and efficient, these operators can only produce sharp features in the vicinity of the surfaces intersection. Such sharp transitions are well desired for some mechanical parts and rigid objects (Figure 2.2(a)), but not when modelling soft objects, welding effects or attaching members of living beings together (Figure 2.2(b)). Hence, Boolean-like operators must be extended to produce a blend between the surface, e.g. a smooth transition from one to the other, as shown in Figure 2.1-bottom row. Several composition models have been designed for producing various effects and are presented in Section 2.2.

Figure 2.2: *Sharp transitions are well suited for the design of mechanical objects (a) while blending is preferred for modelling organic shapes (b). Figures from [GBC+13, BBCW10].*

2.1.1 Operator Representation

While the general composition model defines n-ary operators capable of combining several object parts, say n of them, together in one operation, such operators are rather complex to study and manipulate. Of course, the Boolean operators we presented above are simple n-ary operators that can be used easily. However, they do not provide any control on the transition shape between the combined object parts. When modelling complex objects, user control must be offered in an intuitive way, which is particularly difficult for n-ary operators. Since n-ary operators are traditionally based on the use of a graph between the input primitives, representing them intuitively and efficiently is far from straightforward.

A simpler and more intuitive manner to combine object parts together is to do so pairwisely, then adding parts to the object one after another in an incremental process. This can be done using binary operators, which only combine two object parts and are then simpler to study and design through a dedicated representation: the *operator space*. The operator space for binary operators is a 2D space formed by the two input field functions, say f_1 for the X-axis and f_2 for the Y-axis, as illustrated in Figure 2.3. Since an operator has to be designed for either globally or compactly supported field functions, the properties of the operator space vary depending on the convention. In both cases, there exists a direct relation between the operator space and the object space through f_1 and f_2. Each implicit surface $\mathcal{S}^1_\mathcal{C}$ generated by f_1 corresponds to the vertical line $x = \mathcal{C}$ in the operator space. Likewise, each implicit surface $\mathcal{S}^2_\mathcal{C}$ generated by f_2 corresponds to the horizontal line $y = \mathcal{C}$. When the binary operator is to be used with globally supported field functions, the operator space is generally the space $[-\infty, +\infty]^2$ and the lines $x = 0$ and $y = 0$ correspond to the surfaces of f_1 and f_2 respectively. On the other hand, for compactly supported field functions, the operator space is the space $[0, +\infty]^2$ and the lines $x = 0.5$ and $y = 0.5$ correspond to the surfaces of f_1 and f_2.



(a) (b) (c)

Figure 2.3: *Operator spaces for globally (a) or compactly (c) supported field functions. An example of two red and blue objects (b) is given for the relation between operator spaces and the object space. Surface lines are drawn thicker.*

Now we have this correspondence between the operator space and the object space, how to represent a binary operator in the operator space? and how does this translate into the composition of object parts? The former is done by drawing isocurves of the operator in the operator space, corresponding to the set of points in the operator space for which the operator returns the same values. The latter is done by following the transition made by the operator between the input surfaces. As an example, operators $g_\cup$, $g_\cap$ and a blending operator for both globally and compactly supported field functions are drawn in figure 2.4 with their application on two cylinders. To ease the understanding of the relation between the operator and the resulting field function, from now on we use the same color convention for both: the surface is drawn in magenta, the inside field in red and the outside field in blue.

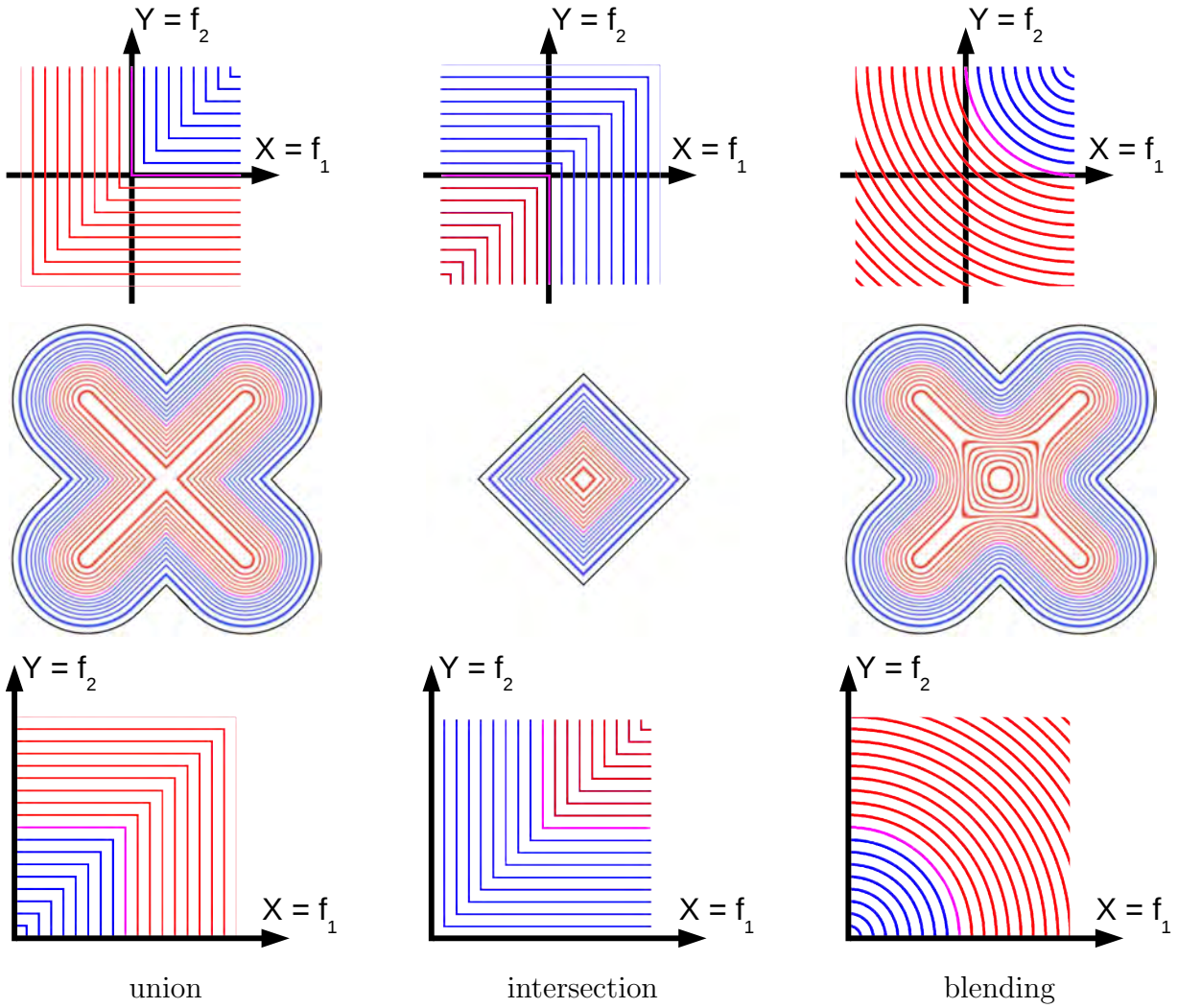


Figure 2.4: *Representation of union, intersection and blending operators in the operator space along with their application on two cylinders (middle row). Top row: operators for globally supported field functions. Bottom row: operators for compactly supported field functions.*

This useful representation intuitively translates the effect of a given binary operator and provides an immediate tool to design and control the behavior of any composition operator. What remains to be done is to find an appropriate formula for the operator, which is, however, not straightforward for complex transition shapes.

2.1.2 Composition Trees

Designing objects by parts and then assembling these parts together can be achieved because the composition model is unified: since the result of the composition of field functions by a composition operator is another field function, this one can then be used as an input for another composition. This process allows iterative and intuitive combinations of simple primitives together through simple operations for the design of complex objects. This modelling process is called Constructive Solid Geometry (CSG) and is usually implemented using a dynamic tree structure [WGG99, dGWvdW09, GDW+16], whose inner nodes are composition operators and leaves are simple primitive objects, as illustrated in Figure 2.5.

Trying to build such a complex structure from scratch to design a complex object can be rather cumbersome. The user would need to think about which operator to use first and how to balance the tree so that evaluation is faster. Instead, defining small parts of the object from simple primitives and then assembling these sub-trees together is a much more efficient way, even if it still requires some assembly plan.

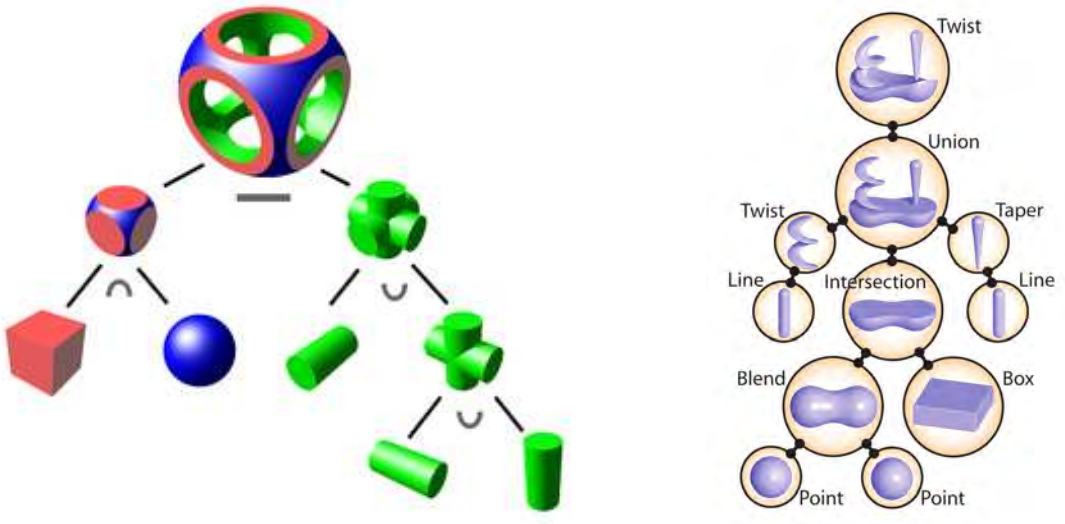


Figure 2.5: *Examples of composition trees. Leaf nodes are field functions representing basic primitives that are then combined in the inner nodes by composition operators, each leading to a new field function. Figures from [Wik, dGWvdW09].*

2.2 Composition Operators: A State of the Art

The control of the smooth transition produced by a blending operator is of great interest to provide better control of the primitives junction. In order to enhance this control and bring more diversity in the shape of the transition between the composed primitives, a large variety of composition operators have been developed, trying to address the limitations of the composition model.

2.2.1 Blending

Ricci [Ric73] was one of the first to propose a composition operator for compactly supported field functions producing a smooth transition between the input primitives, called a blending operator:

$$g_R = (\sum_i^n f_i^s)^{\frac{1}{s}} \tag{2.1}$$

where the parameter s controls the amount of blend during the composition, i.e. the amount of matter added or removed. When $s = 1$ we have $g_R = \sum_i^n f_i$, then giving a maximal blending corresponding to the popular sum operator used by Blinn [Bli82] to display molecules. When s increases we obtain a decreasing blending effect, as illustrated in Figure 2.6, down to a sharp union when $s \to \infty$ in which case $g_R \equiv g_\cup$.

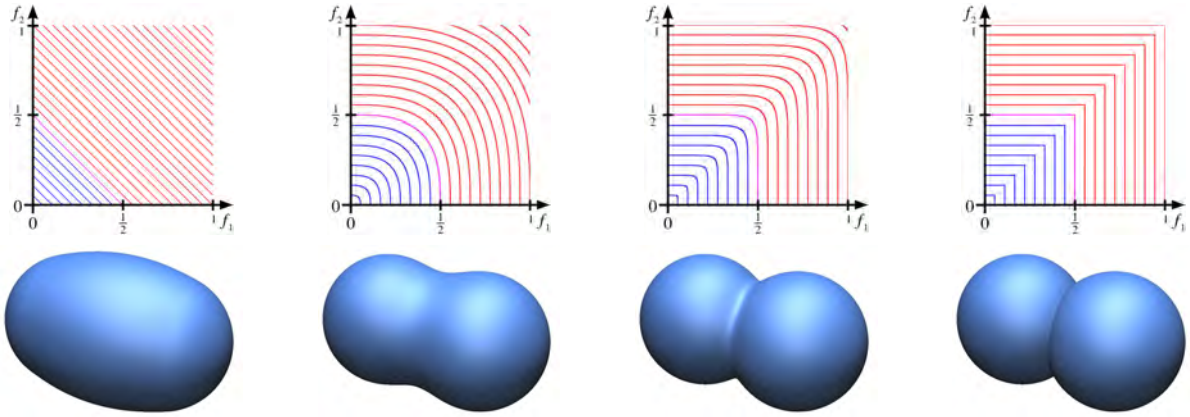


Figure 2.6: *Four applications of the operator g_R on two spheres with, from left to right, $s = 1$, $s = 3$, $s = 10$ and $s \to \infty$.*

2.2.2 Blending Issues

Blending implicit surfaces is a very useful operation when designing organic and deformable objects. However, when it comes to control the blending transition, four major issues, illustrated in Figure 2.7, arise:

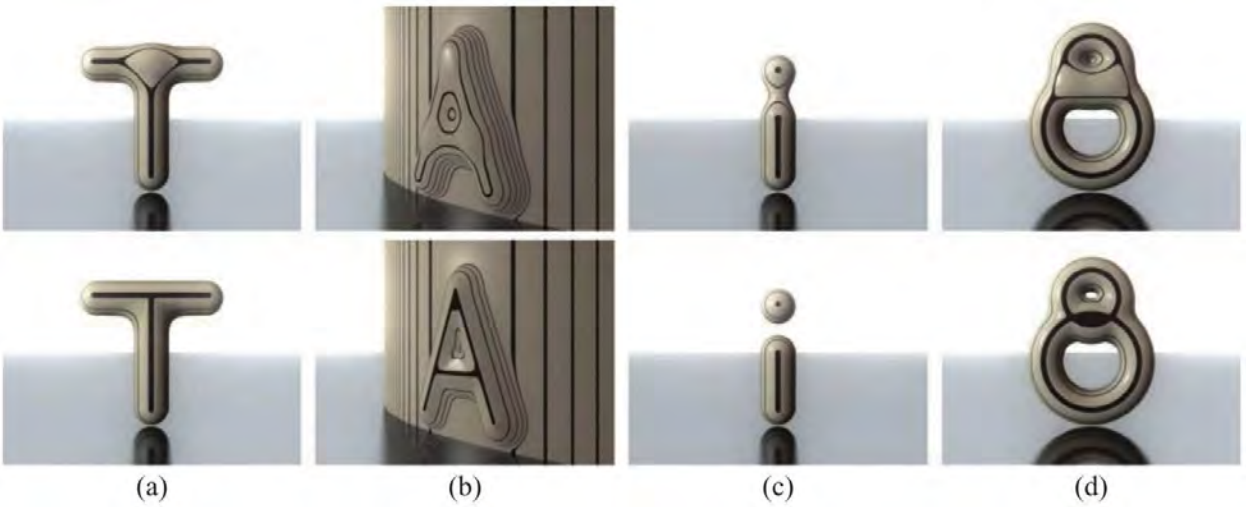


Figure 2.7: *Top row: the main issues of implicit blending: (a) bulging, (b) absorption of details, (c) blending at a distance and (d) change of topology. Bottom row: the expected surfaces. Figures from [GBC+13].*

- **Bulging:** Implicit surfaces are not independent to subdivision. Hence, the surface resulting from a blend will present a bulge in the vicinity of the input surfaces as illustrated on a T-junction in Figure 2.7(a). This can be particularly annoying where only a small blend is expected as when attaching members to the body.
- **Blurring of Details:** The amount of blending between implicit surfaces depends on their blending range. Typically, this range is coherent with the size of the object thus making small detail primitives to be highly deformed, even absorbed, by larger ones due to the difference in field metrics, as shown in Figure 2.7(b). This can be highly problematic when adding small details to enhance the final shape of an object.
- **Blending at a Distance:** Due to the influence of the external field, distinct parts of an object can be blended together if they come too close to one another as in Figure 2.7(c). This is particularly annoying when modelling living beings where union cannot be applied because it would introduce sharp features with blended parts.
- **Change of Topology:** Implicit surface composition provides an easy way to generate objects with arbitrary topology. Blending may however cause unpredictable and uncontrollable filling of holes, as illustrated on two tori in Figure 2.7(d). This loss of topology control is not welcome since parts topology defined by the user may only be changed as a result of the assembly, not of the blend.

These four blending issues are the reason why implicit surfaces have been put aside for years. In the following of this Section, we see how composition models evolved to address these limitations by defining control over the transition shape between the input field functions.

2.2.3 Anisotropic Blending

In Ricci's operator (Equation 2.1), the blend is controlled by a unique parameter that provides control on the transition shape of the blending of input primitives but without taking into account the size of the primitives nor allowing different shape transitions between some primitives in one composition. In order to improve shape control and produce anisotropic transitions between combined primitives, anisotropic blending operators have been developed.

In order to better understand blending operators and build simpler tools to control them, the work by Hoffman and Hopcroft [HH85] focuses on binary operators and leads to the useful representation for binary operators in a 2D space we presented in Section 2.1.1. For this purpose, other isovalues than $\mathcal{C}$ are considered so that the blend between the two input field functions can be expressed with respect to isosurfaces around $\mathcal{S}_\mathcal{C}$ as in Figure 2.8. Two cylinders are blended as a combination of the dotted outer surfaces (left): these surfaces are intersected in order to create the curves around the vertical cylinder. The new blue surface (right) is obtained by continuously creating these curves for all of the outer surfaces from the horizontal cylinder to the vertical one. The control of the way the outer surfaces are intersected, and thus the shape of the transition, is achieved by defining a "profile" curve, here corresponding to the dotted circle (left) tangential to both cylinders. While originally designed for the composition of algebraic implicit surfaces, this isovalue manipulation is actually practical for the composition of all implicit surfaces, whether globally or compactly supported, and leads to the operator space representation presented in Section 2.1.1.

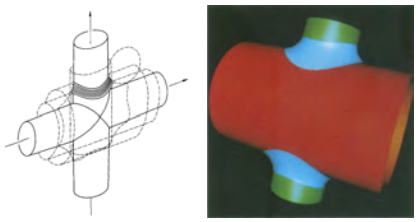


Figure 2.8: *Using outer isovalues to create blend. Figures from [HH85].*

Taking advantage of the operator representation given by the operator space, Rockwood and Owen [RO85] design a new blending operator for algebraic surfaces called superelliptic blend (which can be extended to an n-ary operator):

$$g_{SE}(f_1, f_2) = 1 - \left[1 - \frac{f_1}{r_1}\right]_+^s - \left[1 - \frac{f_2}{r_2}\right]_+^s \tag{2.2}$$

where $[x]_+ = max(0, x)$ is used to prevent issues when $r_i \to 0$ in which case union is applied, r_1 and r_2 delimit the blending zone (asymmetric blending can then be applied) and s controls the amount of blending, i.e. the shape of the $\mathcal{C}$-isocurve of g_{SE}, as illustrated in Figure 2.9.

To prevent the bulging problem, this operator is enhanced [Roc89] by taking into account the angle θ between the gradients of the input fields as a parameter for the blending zone. To this end, r_i is replaced with $R_i(\theta) = r_i(1 - \cos\theta)$. This manipulation thus enables different amounts of blending to be used in one application of the operator according to the angle θ as illustrated in Figure 2.10. When $\theta = 0$ the field gradients are collinear, meaning the surfaces

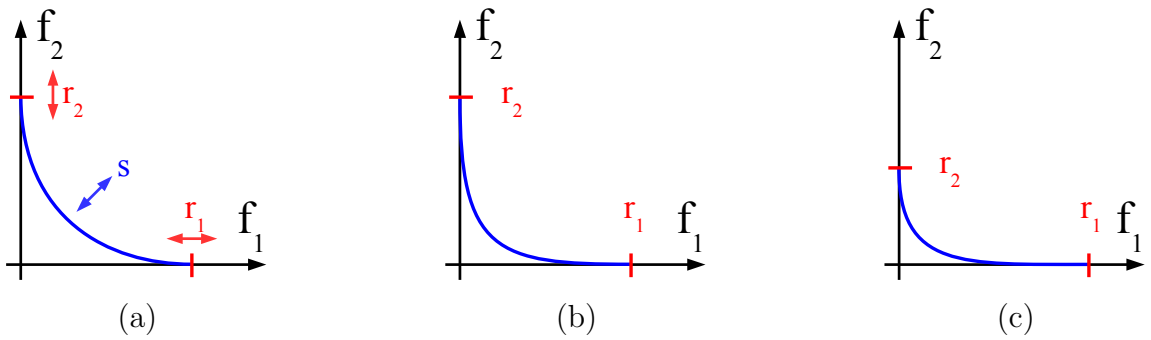


Figure 2.9: *Superelliptic blend with its control parameters (a): s for the amount of global blending (b) and r_1 and r_2 for anisotropic blending (c).*

are tangential, leading to $R_i = 0$ and then union is applied. On the contrary, when $\theta = \pi/2$ the gradients are orthogonal, meaning the surfaces are so, and a maximal blend is applied. In between, values of θ allow smooth interpolation from union to blend, thus resulting in different applications for different evaluation points, in the contrary of parameter s for Ricci's operator (Equation 2.1). This composition operator however increases the blending at a distance problem since when object parts come close to one another, $\theta = \pi$ thus resulting in an increased blending effect.

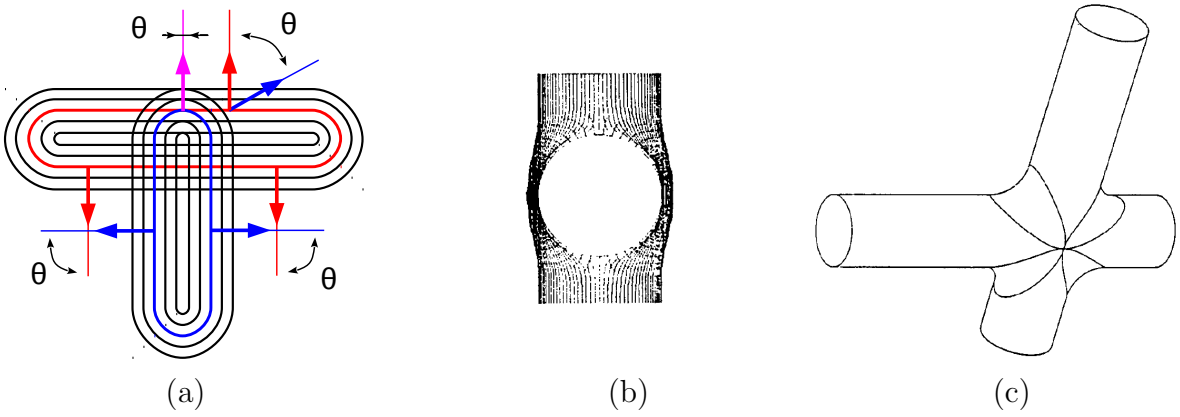


Figure 2.10: *Using a function of the angle θ between the field gradients (a) allows a different blending behavior at different positions in space. Bulge (b) can thus be avoided (c) where the surfaces are tangential.*

A similar operator to g_{SE} has been developed by Pasko [PASS95] for globally supported field functions in general:

$$g_P(f_1, f_2) = f_1 + f_2 - \sqrt{f_1^2 + f_2^2} + \frac{a_0}{1 + \left(\frac{f_1}{a_1}\right)^2 + \left(\frac{f_2}{a_2}\right)^2} \tag{2.3}$$

where a_0 controls the amount of global blending between f_1 and f_2, and a_1 and a_2 control the asymmetry in the blend. Here, the two left members refer to a specific kind of operator called "clean union", illustrated in Figure 2.11 and the fractional part corresponds to the amount of matter added to produce the blend. The clean union operator produces the union of the input surfaces while blending the fields elsewhere. It enables smooth field variations around the surface, thus making the field behave more like a distance field. It also prevents gradient discontinuities in the generated field which may cause discontinuities in the visualization of reflections on the surface or surface discontinuities after further blending compositions.

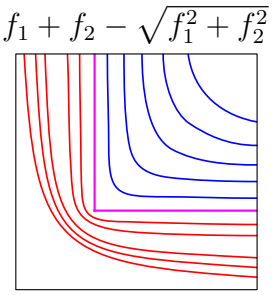


Figure 2.11: *Clean union for globally supported field functions.*

2.2.4 Graph-based Blending

Rather than using a global parameter to control the amount of blending, graph-based n-ary operators [OM93, GW95] were introduced in order to control where the blend may occur or not (see Figure 2.12). Implicit primitives connected through the graph will blend together forming their own components while unconnected primitives will not, thus resulting in the primitives to deform when colliding thanks to the introduction of a contracting function: $ct(f) = min(\mathcal{C} - f, 0)$. The contracting function ct deforms the field of the input primitive f so that, when summed with other primitives, a contact surface is created at the intersection of the surfaces. The object components are then combined using the union operator to avoid blending between separate parts. This is done even though the primitives are linked in the graph, which unfortunately composes these primitives with sharp transitions, while a blend might be expected as for the hand of Figure 2.12.

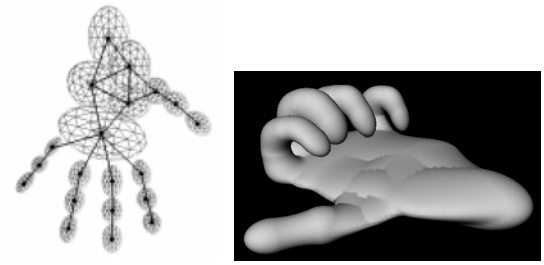
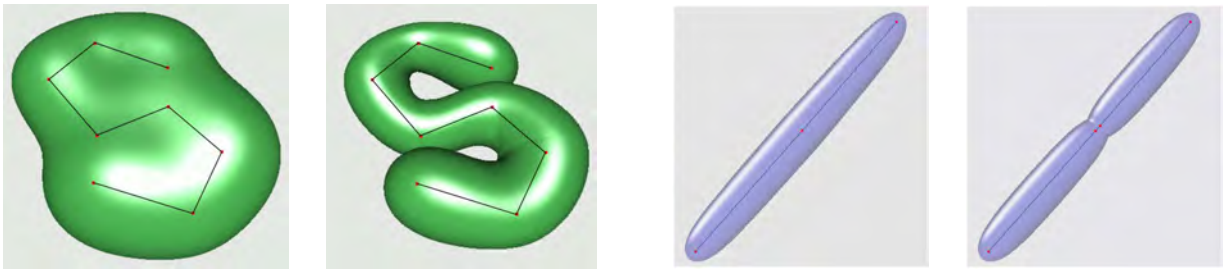


Figure 2.12: *Modelling a hand using graph-based blending of primitives to prevent fingers to blend. Figures from [OM93].*

The graph-based approach has been adapted to convolution surfaces [BS91] by Angelidis and Cani [AC02]. Unwanted blending between skeleton elements is prevented (see Figure 2.13) by using only the most influencing one and its direct neighbors in the graph. However, this composition method breaks the subdivision independence of convolution surfaces and can introduce creases as well as sharp features.

In order to preserve C^1 continuity, Hornus et al. [HAC03] propose to keep adding skeleton elements until their contributions becomes negligible. While preventing unwanted bulge of looping skeletons, this method cannot handle highly folding skeletons as shown in Figure 2.14.



(a) (b)

Figure 2.13: *Adapting graph-based method to convolution surfaces (a) prevents unwanted blending between skeleton parts but (b) breaks the independence to skeleton subdivision. Figures from [AC02].*

(a) (b)

Figure 2.14: *Another graph-based composition for convolution surfaces (a) preventing unwanted blending for loop skeletons, but (b) restricted to low skeleton folding over. Figures from [HAC03].*

A warping-based model for scale independent convolution surfaces (SCALIS) is later developed by Zanni et al. [ZBQC13] enhancing the use of varying surface radius. In order to avoid the blending problems, and inspired by the use of the gradient, a new kind of specific warping is applied on top of the SCALIS model in [ZCG14]. Here, the field resulting from the summed skeleton is remapped so that its gradient norm meets the one of a reference case (corresponding to the infinite version of the corresponding dimension skeleton). The mapping is done by using a specific 2D space taking as X-axis the field value and as Y-axis the norm of the gradient. In this space, the field and gradient norm of an evaluation point are remapped on the curve of the reference case according to a fixed angular parameter α (see Figure 2.15). This warping-based operator provides a good trade-off between union and blending, also preventing blend at a distance and preserving the skeleton topology as illustrated in Figure 2.16.

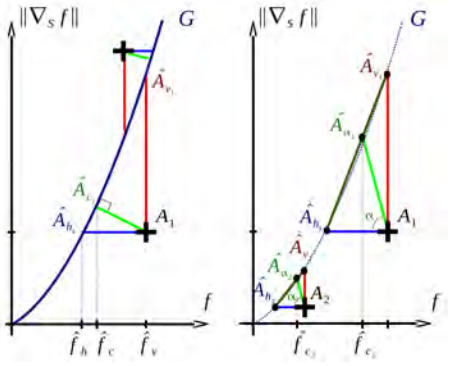


Figure 2.15: *Projecting the field value and gradient norm in A_i against the curve G for a reference case.*

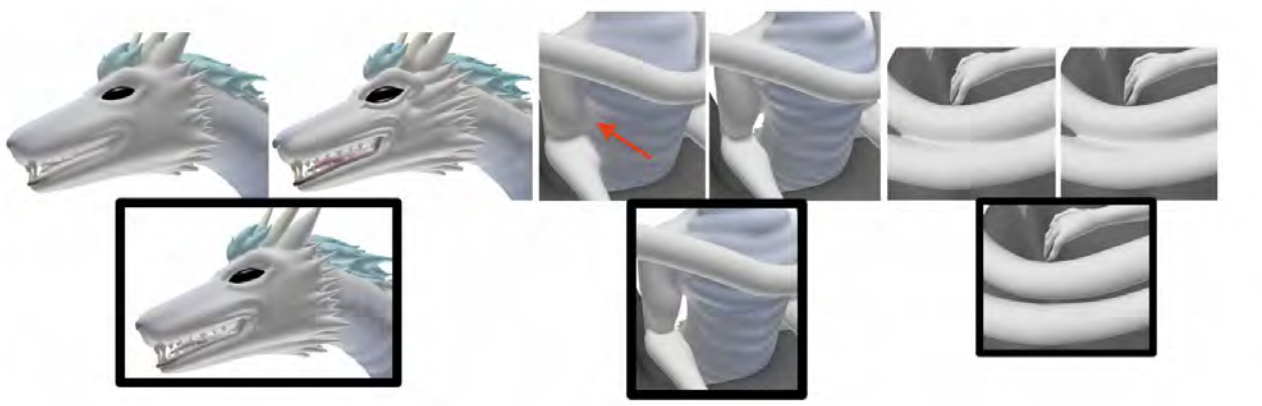


Figure 2.16: *Gradient-based warping for SCALIS prevents blend at a distance and topology modification. Figures from [ZCG14].*

2.2.5 Localized Blending

Anisotropic blending can be extended to produce a controlled restricted blend. Indeed, anisotropy parameters can be seen as blend delimiters.

Bounding curves: Hence, a new operator design method is introduced by Barthe et al. [BWdG04] for the composition of compactly supported field functions by combining the representation for binary operators with the use of some angles as parameters. Starting from the representation for binary operators, the operator space is divided in three main areas: one area under the line with angle θ_1 with the X-axis, one area above the line with angle θ_2 and the third area between the two (see green lines in Figure 2.17(a)). In the two first areas, the union operator $g_\cup$ is used to keep the input fields unchanged. The parameters θ_1 and θ_2 thus allow anisotropic blending defined in the metric of the input field functions by delimiting their blending range. In the third area, isocurves of operator G linking the same isovalues of f_1 and f_2 are created as follows. For each isovalue C, the points P_1 and P_2 are found. These points correspond to the intersections between the lines $f_1 = C$ for P_1 and $f_2 = C$ for P_2 with the lines with angle θ_1 and θ_2 respectively. Finally, the desired blending shape is created using a "profile" curve linking corresponding isocurves of the two input field functions, as illustrated in red in Figure 2.17. The desired blending shape is created using a profile function $\rho = m(\theta)$ that is finally put in place between points P_1 and P_2 using polar coordinates. This process is then applied to all isocurves of the input primitives to draw the operator. Following this construction method, a clean union operator for compactly supported field functions is also created using specific partitioning curves meeting at $P = (\mathcal{C}, \mathcal{C})$ to divide the 2D space, as illustrated in Figure 2.17(b).

In the work by Hsu and Lee [HL03b], a more detailed analysis of the profile curves is given along some theorems, properties, such as differentiability, and applications of this new kind of operators, also considering subtraction operations.

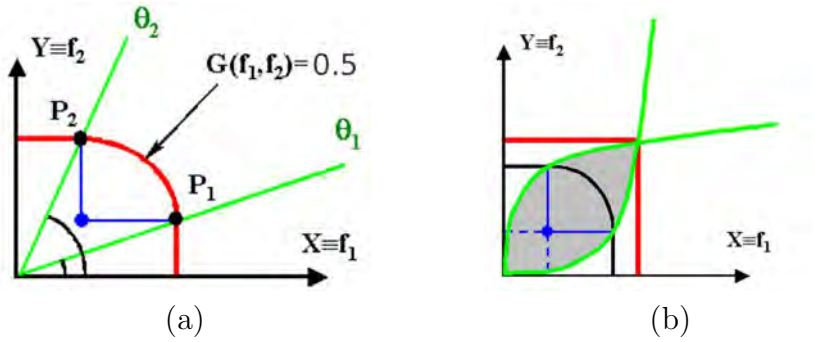


Figure 2.17: *Designing a new binary operator G using a 2D space division by angles θ_1 and θ_2 (a). A new clean union operator for compactly supported field functions is also designed (b). Figures from [BWdG04].*

Bounding primitives: The idea of bounding the application zone of blending has also been extended by Pasko et al. [PPIK02, PPK05] in order to restrict the blend for globally supported field functions by bounding the area in which the blend is applied (see Figure 2.18). Here a compactly supported implicit surface is used to bound this area: its value controls the parameter a_0 of operator g_P (Equation 2.3). At the boundary of its support $a_0 = 0$, thus producing a clean union, while its maximal inner value leads to a maximal blend. A large variety of blending behaviors is then possible depending on the bounding primitive used. Even though this method provides high user control, it requires a lot of user input in order to obtain a fair surface shape.

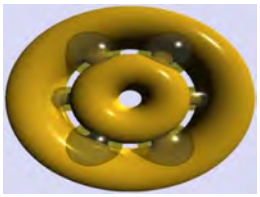
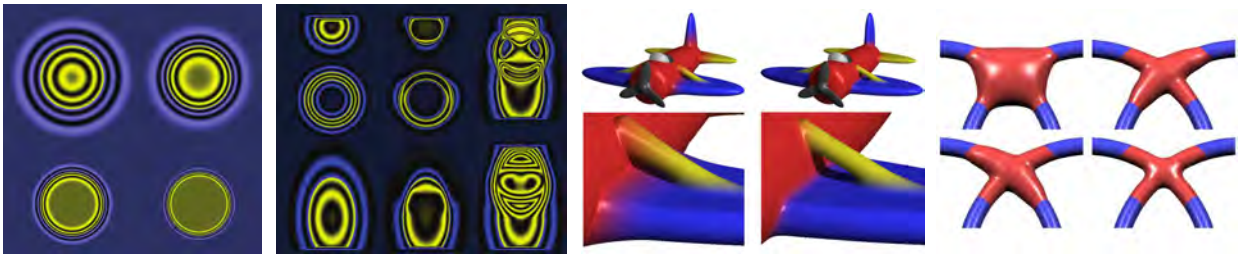


Figure 2.18: *Blending of two tori bounded by six disconnected components. Figure from [PPK05].*

Local Compression: As we have seen, unwanted blending is one of the most focused issue of implicit surfaces composition, also linked to the blending at a distance problem causing non-colliding primitives to blend. Another one that is highly problematic when modelling 3D objects is the absorption issue as illustrated in Figure 2.7(b), arising when primitives of different scales are blended as described by Wyvill and Wyvill [WW00]. This issue is then addressed in a more effective way by de Groot et al. [dGWvdW09]. The initial primitive field metrics are locally compressed around the $\mathcal{C}$-surface in order to control the amount of blending through their blending range (see Figure 2.19). This is achieved according to a set of parameters: one for isotropic compression of the primitive fields for the general blending behavior with other primitives, illustrated in Figure 2.19(a) and one for local restriction of the blend with particular primitives, as depicted in Figure 2.19(b). This operator can also be seen as a graph-based operator where each node contains a primitive and its global compression parameter, and each edge contains the local compression parameter to be applied for both end nodes. While giving advanced control on the blending of primitives and allowing anisotropic blend, see Figure 2.19(c), this operator really needs a lot of parameters to be set



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

Figure 2.19: *Local restriction of blending by (a) global compression of the base field and (b) pairwise local control of blended primitives, (c) leads to better blend control and avoids bulge artefacts. Note that (d) asymmetric blend can be achieved. Figures from [dGWvdW09].*

(i.e. the square of the number of primitives). Moreover, setting the parameters may become difficult or even impossible when multiple primitives require different blending effects in the same area because the field metrics cannot be compressed and relaxed at the same time.

Hybrid bounding: Then, in order to avoid the blending at distance and the blurring of details problems, Bernhardt et al. [BBCW10] automatically define bounding primitives around the primitives intersection curves. First, intersection curves between the surfaces are found by locally meshing the surfaces and intersecting the resulting polygons as depicted in Figure 2.20(a). These curves are then used as the skeleton of the field function defining the bounding volume with parametrizable support size. The blending range is finally adapted to the size of the primitives, i.e. a large blend on the large primitive and a small blend on the small one (see Figure 2.20(c)). This blending range is computed with respect to the local curvature of the primitives used here as a descriptor of the size of the smallest local feature. Even though this method is automatic and very effective, it can be quite slow when multiple intersections occur due to the need for intersection curves computation.

Gradient-based Composition: In order to overcome all of the blending issues shown in Figure 2.7, a new class of quasi-C^∞ continuous gradient-based binary operators are introduced by Gourmel et al. [GBC+13]. Their method takes advantage of the operator design method of Barthe et al. [BWdG04], i.e. using curves to define the blending region in which a pattern describing the shape of the composition is reproduced. Inspired by the work of Rockwood [Roc89], these curves, in yellow in Figure 2.21, are parametrized by a unique angle α. This angle is set as a function of the angle θ between the gradients of the input primitives. The function $\alpha(\theta)$ is called a controller and allows various composition behaviors depending on θ, illustrated in Figure 2.22. For instance, the main controller presented in Figure 2.22(a) produces a maximal blend when the gradients are orthogonal, a clean union when they are collinear and intermediate blend in between. This "camel" controller prevents the blending issues as shown in Figure 2.7. Gourmel et al. also proposed an "organic" controller (Figure 2.22(b)) specifically designed for assembling organic shapes, such as the legs and tail of the camel in Figure 2.22(c). Here, only a little amount of blend is required when

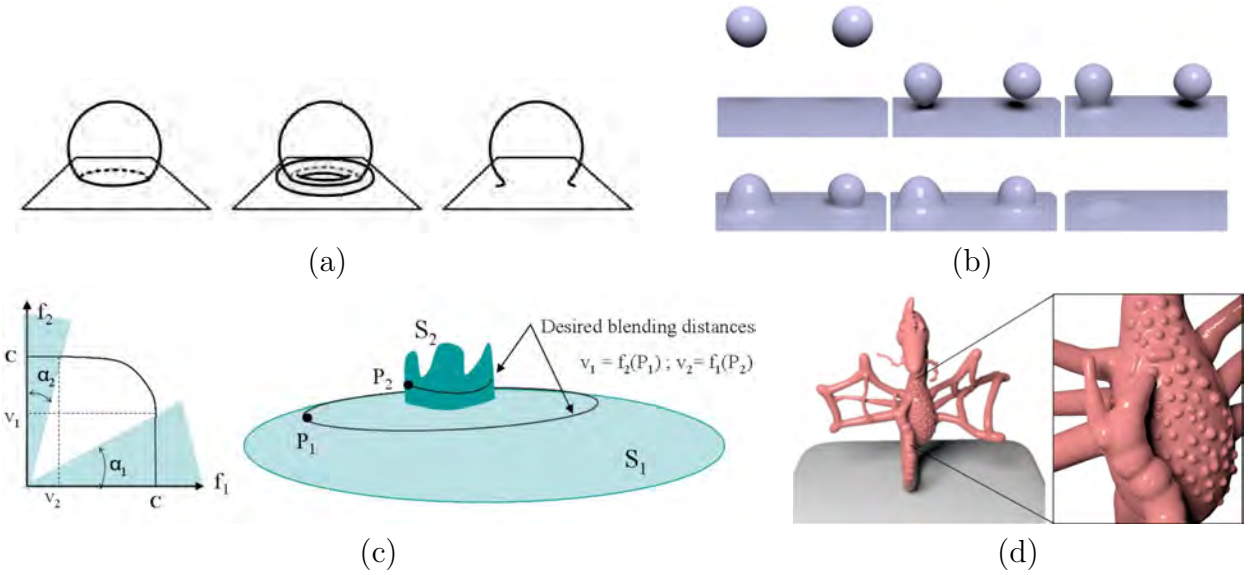


Figure 2.20: *Bounding the blend around (a) intersection curves avoids blending at a distance (b). Adapting the blending range according to the size of the primitives (c) avoids absorption when modelling details (d). Figures from [BBCW10].*

gradients are aligned. The amount of blend rises as the gradients diverge up to some angle where maximal blend is achieved to smoothly assemble parts of the organic object. Blend then decreases down to a clean union before gradients are opposite in order to design folding effects.

Whereas the controller should be adjusted with respect to the field variation of the input primitives, the common use of only one kernel for all primitives limits the required work. While similar to Rockwood's gradient based operator, this one enhances user control through the controller and prevents extra blend when the gradients are opposite. Anisotropic blending can still be achieved by using different angles for each delimiting curve as in Barthe's

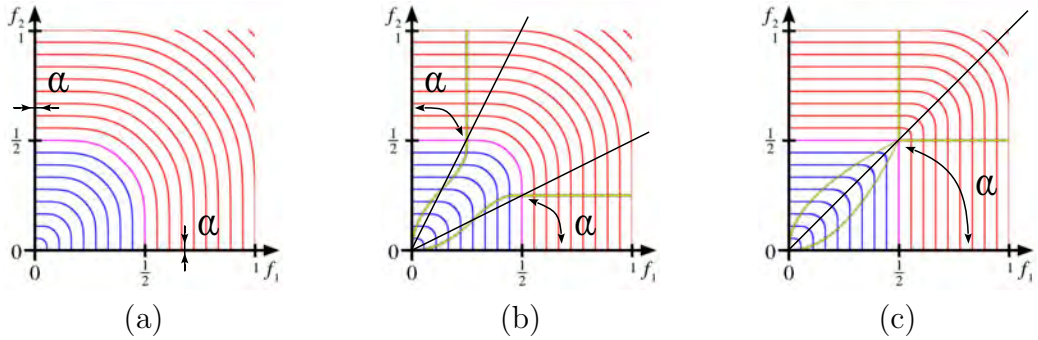


Figure 2.21: *Delimiting curves in yellow are defined according to an angle parameter α. New operators can produce a maximal blend (a), a clean union (c) or an intermediate blend (b).*

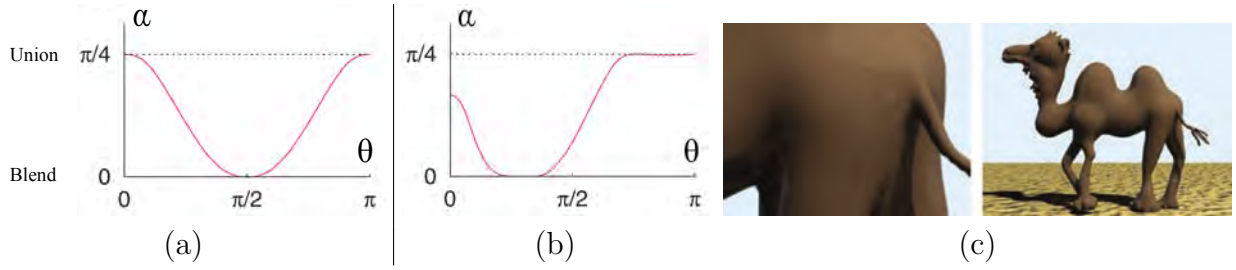


Figure 2.22: *The "Camel" controller (a) prevents the blending issues of Figure 2.7. The "organic" controller (b) is specifically designed for the blending of organic shapes. Figures from [GBC+ 13].*

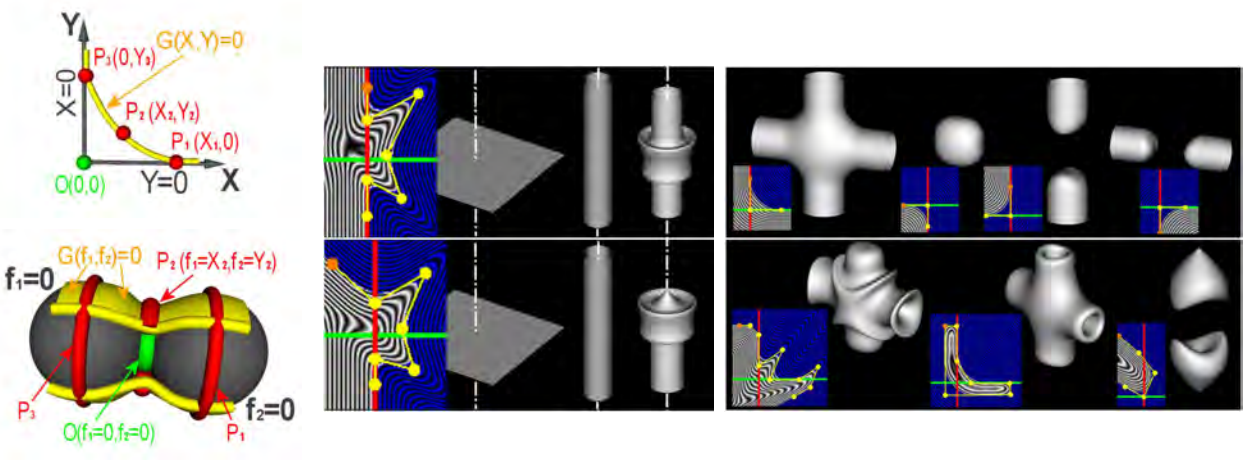
operators with one controller for each. In addition, the new set of operators can be used to define the blending behavior of primitives along an animation by modifying the controller over time.

These operators are the most advanced operators up to now and prevent the four major issues of blending to arise, as illustrated in Figure 2.7. They are however rather costly to evaluate directly since they do not admit a simple closed-form formula. Fortunately, since the design of such an operator only relies on the design of the profile and the controller, these operators can be baked into a 3D texture parametrized by the two input field function values and the gradient angle θ. The final composition field value and gradient are then retrieved by trilinear interpolation of the neighboring texture elements.

2.3 Free-Form Composition

Blending is not the only interesting effect when objects have to be combined. Actually, the desired transition shape between the input primitives may be arbitrarily complex, leading to the development of free-form composition operators.

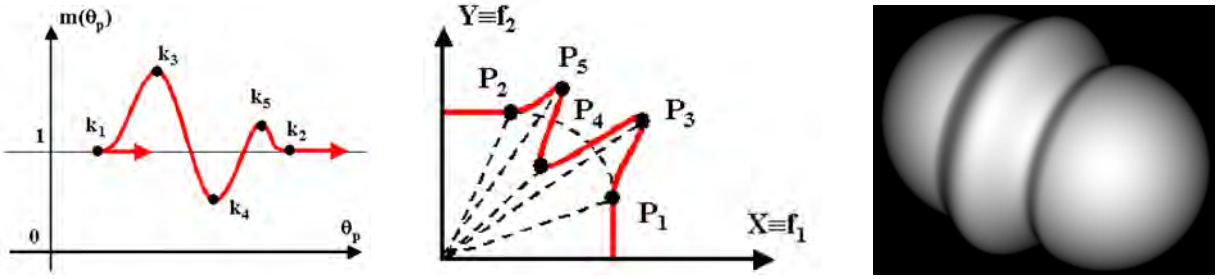
Taking advantage of the relation between the operator space of Section 2.1 for binary operators and the object space where the two input field functions are defined, Barthe et al. [BGC01] introduce the notion of implicit extrusion fields (Figure 2.23). Here, the composition operator G is represented in operator space for binary operators as a 2D field function around a 2D profile curve representing the desired shape transition between the input surfaces (Figure 2.23(b)). We have already seen the correspondence between vertical and horizontal lines in the operator space and input isosurfaces in object space, but this is not the only one. In operator space, each point $P(X, Y)$, in red in Figure 2.23(a)-top, corresponds to the intersection curve in object space between the surfaces defined by $f_1 = X$ and $f_2 = Y$, in red in Figure 2.23(a)-bottom. By extension, a curve in operator space, i.e. a continuous set of points, in yellow in Figure 2.23(a)-top, represents a surface in object space, i.e. a continuous set of curves, in yellow in Figure 2.23(a)-bottom. Using this relationship between these two spaces, the authors provide the user with direct control on the resulting shape from 2D curves defining the expected transition behavior, thus providing an interesting sculpting system illustrated in Figure 2.23(c-d).



(a) (b) (c)

Figure 2.23: *The sculpting system for implicit surfaces introduced by Barthe et al. [BGC01, BDS+03]. Profile curves describing the transition behavior are defined by the user in either the 2D space for binary operator representation (a-top) or the 3D space where the implicit surfaces lie (a-bottom). Different curve designs are given as examples and used on a plane and a cylinder (b), and on two cylinders (c).*

This idea is then improved [BDS+03] to provide a better framework and give better properties to the resulting field and surface. It is then extended for compactly supported field functions [HL03b, BWdG04] as illustrated in Figure 2.24. Here, a free-form profile curve $m(\theta_{\mathbf{p}})$, defined in local polar coordinates, is used to deform the blending curve between the delimiting points P_1 and P_2.

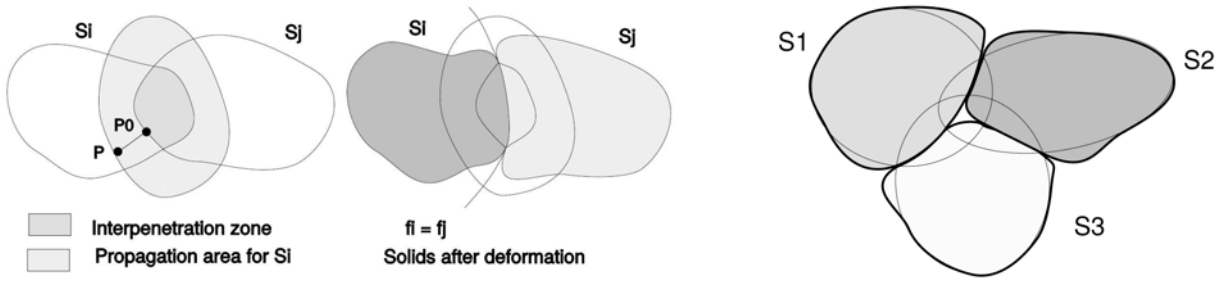


(a) (b) (c)

Figure 2.24: *Designing free-form operators using profile curves (a) describing the desired transition to link isocurves from the input field functions (b) and its application on two spherical field functions (c). Figures from [BWdG04].*

2.4 Contact

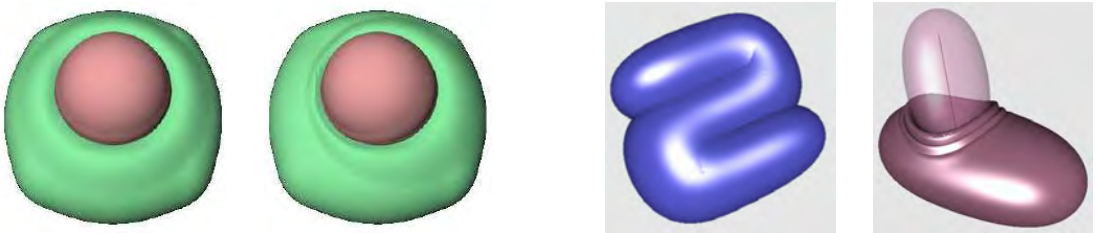
A good example of free-form composition is the bulge in contact operator. When soft objects come in contact during an animation, they are expected to deform against each other due to the impact force, then creating a bulge around the contact region. Opalach and Maddock [OM93], followed by Guy and Wyvill [GW95], used a contracting function *ct* in their graph-based blending operator to prevent blending between primitives not connected within the graph by creating a contact surface between them. This contact operation is then extended to build bulge in contact operators [Gas93, CGD97, OC97] that create bulge when soft objects are smashed against one another. On top of creating the contact surface in the interpenetration zone, an additional deformation function is applied in a so-called "propagation area" around the contact surface to create a bulge effect around the interpenetration zone (see Figure 2.25).



(a) (b)

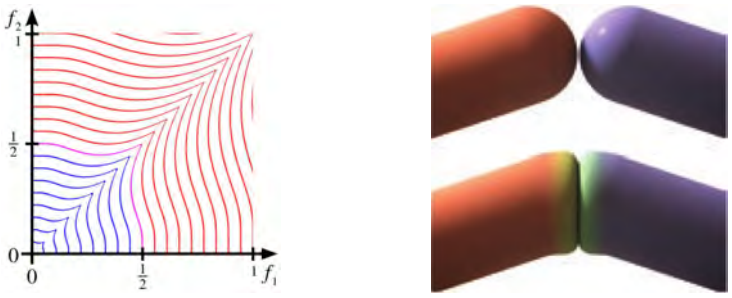
Figure 2.25: *Bulge in contact operators. (a) A contact is created in the interpenetration zone while bulge is introduced in the propagation area. (b) Resulting shapes for three soft objects deforming from the thin surface to the thick one. Figures from [Gas93].*

Further improvements of the contact operators are made by taking into account objects velocity [OC97], to adapt it to convolution surfaces [AJC02] or to model ripples as illustrated in Figure 2.26.



(a) (b)

Figure 2.26: *Taking into account objects velocity (a) to provide anisotropic bulge in contact for convolution surfaces. (b) Modelling ripples. Figures from [OC97, AJC02].*



(a) (b)

Figure 2.27: *Gradient-based contact operator for bulge at contact (a) and its application on two colliding cylinders (b). Figures from [GBC+13].*

Also, Gourmel et al. [GBC+13] proposed a gradient-based version of the bulge in contact operator using a particular profile curve, as shown in Figure 2.27. While providing a bulge in contact effect, this operator does not model the contact surface due to its construction process.

2.5 Conclusion

A wide range of composition operators have been developed in order to improve user control on shape modelling and to diversify the composition behaviors of implicit surfaces. Though the main issues concerning blending composition have been solved for the case of compactly supported field functions, whether in binary or n-ary compositions, some shortcomings are still present:

- For a modelling process, both union, intersection and difference like operations are required, but only the union has been studied due to the correspondence between the Boolean identities. Indeed, since intersection and difference can be expressed using the union and the complement operations, improving the control, intuitiveness and efficiency of the union operation has been considered enough. However, when applied on compactly supported field functions, such combination of union and complement operations can lead to composition artefacts such as creases and distorted fields if no care is taken. We address this issue in Chapter 3 by introducing new properties for compactly supported field functions and additional constraints on composition operators.

- When composing a very large number of intersecting primitives with different composition behaviors at the same time, such as in fluid simulations, using binary operators is not sustainable. However, the use of existing n-ary operators results in surfaces with sharp features or would require too much parameter tuning to get the desired composition behavior. In Chapter 4, we focus on particle-based fluid simulations and introduce a new n-ary operator based on the computation of specific particle neighborhoods tracking the fluid topology.

Chapter 3
Geometric Modelling using Field Functions

Publication: *Adequate Inner Bound For Geometric Modeling with Compact Field Functions*, F. Canezin, G. Guennebaud, L. Barthe. Computers & Graphics, Shape Modeling International (SMI) 2013 conference.

Compactly supported field functions are commonly used as primitives of modelling systems due to their localisation of the field variation, allowing better composition control and faster visualisation than globally supported field functions. Another important part of a modelling system based on field functions is the set of operators used to combine the primitives. As usual user-interfaces induce an incremental modelling process, binary operators are well suited to such modelling systems; moreover as the user gives directives to add or suppress parts of the designed object, the operators are commonly used in a Boolean context, thus performing unions, intersections and differences of parts, with an extension to deformation operations.

We thus here focus on binary operators g composing compactly supported field functions f_1 and f_2 to result in a new compactly supported field function F:

$$\begin{array}{rcl} g: & \mathbb{R}^2 & \to \mathbb{R} \\ & (X,Y) & \mapsto g(X,Y) \in \mathbb{R} \\ F: & \mathbb{R}^3 & \to \mathbb{R} \\ & \mathbf{p}=(x,y,z)^t & \mapsto F(\mathbf{p}) = g(f_1(\mathbf{p}), f_2(\mathbf{p})) \end{array}$$

Recently developed set of operators are intuitive and effective, and they only focus on the behavior of the union since intersection and difference operators can be derived from union or blend operators through the following Boolean rules:

$$g_\cap(f_1,f_2) = 1 - g_\cup(1-f_1, 1-f_2) \tag{3.1}$$

$$g_\setminus(f_1,f_2) = g_\cap(f_1, 1-f_2) = 1 - g_\cup(1-f_1, f_2)\ , \tag{3.2}$$

where $(1-f_i)$ is the complement of f_i with respect to the compactly supported field functions convention, as introduced in Chapter 1. The surface stays the same while the inside and the outside are inverted, keeping the same field variation. Thus, as illustrated in Figure 3.1(a), fundamental constraints considered for the design of composition operators for compactly supported field functions have been [BWdG04]:

- to produce the desired transition for the 0.5-isosurface, either sharp or smooth, as depicted in Figure 3.1 in the upper left and bottom right respectively;
- to preserve the boundaries of the composed field functions supports, i.e. produce the union of the external boundaries with 0 field value, as illustrated by the black curves;
- to maintain a smooth field everywhere else, either inside (red curves under the surface) or outside (blue curves), to preserve gradient continuity and avoid further composition artefacts as in Figure 3.2(a).

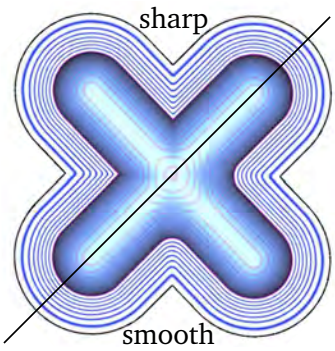
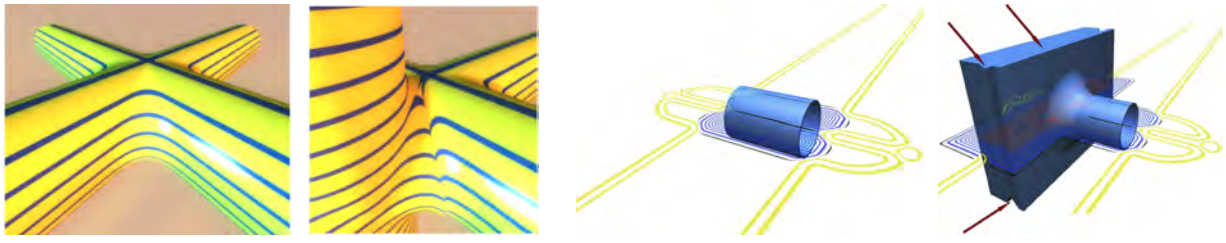


Figure 3.1: *Example of the desired field behavior after composition.*

The cases of equations 3.1 and 3.2 can lead to inconsistencies because intersection and difference operators are built on the union of the complements of the input field functions. Indeed, since the inner values of the field functions are not bound, they can exceed the value 1 and the complement operation will produce negative field values (as $1 - f < 0$ when $f > 1$), which breaks the convention for compactly supported field functions. Thus, applying compact-specific operators to the complement is not consistent nor safe and can introduce several artefacts when further compositions are applied in these unbounded regions. For example, negative values can arise in the field, as shown in yellow Figure 3.2(b)-left, thus potentially making it unsuitable for blending operations. For instance, applying further composition operations in these areas can lead to a surface with undesired cavities as pointed by the red arrows in Figure 3.2(b)-right.

To solve this problem, we first introduce in Section 3.1 an inner bound as a supplemental property for the field functions used in geometric modelling, so that the complement operation can be safely applied, i.e. guaranteeing the same properties for the result as for the input. As for the support boundary, the inner bound will delimit the region of field variation



(a) (b)

Figure 3.2: *Examples of artefacts that can be introduced when the resulting field function presents gradient discontinuities (a) or negative values (b).*

inside the object, then defining with the first one the volume around the surface where the composition operators can deform it. Secondly, we introduce in Section 3.2 new additional constraints that the operators must fulfil in order to produce field functions presenting all the same properties as the input ones. To this end, we show how to adapt recent operators to achieve conforming intersection and difference operators when derived through Equations 3.1 and 3.2.

Finally, we also propose in Section 3.3 a novel set of asymmetric operators tailored for the representation of small details on the surface. These new operators better preserve the global shape of the outside field: the field modifications introduced by the details vanish as the distance to the surface increases, thus removing depression artefacts produced by previous operators as shown in the side Figure.

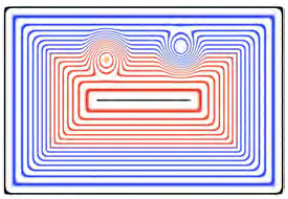


Figure 3.3: *Depression artefacts due to the addition of small details.*

3.1 Field Function Representation

In order to stabilise the stack of composition during the modelling process and to make the composition model uniform, we first present our consistent representation of compactly supported field functions and its properties. Our goal is to make the field function f and its complement $(1 - f)$ satisfy the same properties on which the definition of the composition operators rely. They must be positive, greater than the isovalue 0.5 inside the volume delimited by the surface, and lower outside with decreasing values when getting further from the surface up to a bound, outside of which they must uniformly equal zero. Generally, while these properties are true for the input field function f, these properties do not hold for its complement. However they are automatically satisfied as soon as we set an inner bound to 1 in the field of f, as illustrated in Figure 3.4. This simple manipulation can be implemented for the different families of field functions used in geometric modelling as shown below.

However, as well as the outer bound defining the region around the object where union-like operators can modify its surface, the inner bound defines the region where the difference-like operators can produce new shapes. Special care has thus to be taken on the size of the volume surrounding the surface delimited by these two regions: it has to be large enough to allow the generation of additional shape on both sides. In general, composition behaviors in both inner and outer parts are expected to be symmetric and a default solution is thus to generate symmetric field functions. If required, the size of the inner and outer bands can be set independently to accommodate for any specific requirement.

Any field function can be adapted to undertake our supplemental constraints. For a non-consistent compactly supported field function f^c, which might come from a skeleton-based, convolution or non conforming composition object, we propose the following transfer function:

$$t^c(x) = \varphi\left(\frac{\mathcal{C} - x}{r}\right), \tag{3.3}$$

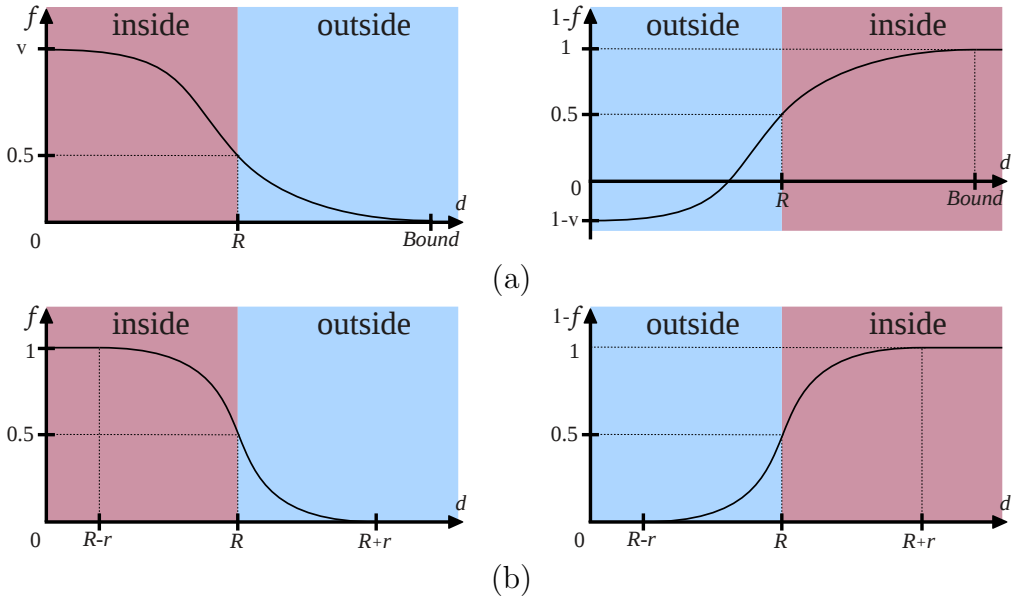


Figure 3.4: *Our consistent representation allows for safe complement operation. Without the inner bound, it can lead to negative values (a) while the inner bound solves this issue (b).*

where $\mathcal{C}$ is the isovalue of f^c, $r \in [0, \mathcal{C}]$ is the width of the symmetric band defined around the surface with respect to the input field metric and φ is a smooth-step function. For the choice of φ, any smooth-step function, such as those proposed by Li et al. [LP04] and Li [Li07], can be employed as soon as $\varphi(0) = 0.5$. In this work we use the following C^2 continuous polynomial function:

$$\varphi(x) = \begin{cases} 1 & \text{if } x \leq \text{-}1 \\ 0 & \text{if } x \geq 1 \\ \text{-}\frac{3}{16}x^5 + \frac{5}{8}x^3 - \frac{15}{16}x + \frac{1}{2} & \text{otherwise.} \end{cases} \tag{3.4}$$

This mapping is symmetric but if required, individual control on the inner and outer widths can be achieved using for instance the step function of Hsu et al. [HL03a]. For a globally supported field function f^g, which can come from object reconstruction, we use the transfer function $t^g = \varphi(x/r)$. The transfer functions t^c and t^g and the resulting shape of the field function $f = t^c(f^c)$ for two different band widths are shown in Figure 3.5.

3.2 Composition Operators

Now that we have proposed our representation for conformal field functions with respect to the complement operation ($1 - f$ has the same properties as f), we can focus on the composition operators. As already stated, intersection and difference operators are derived from union operators and the complement operation. As for the complement operation,

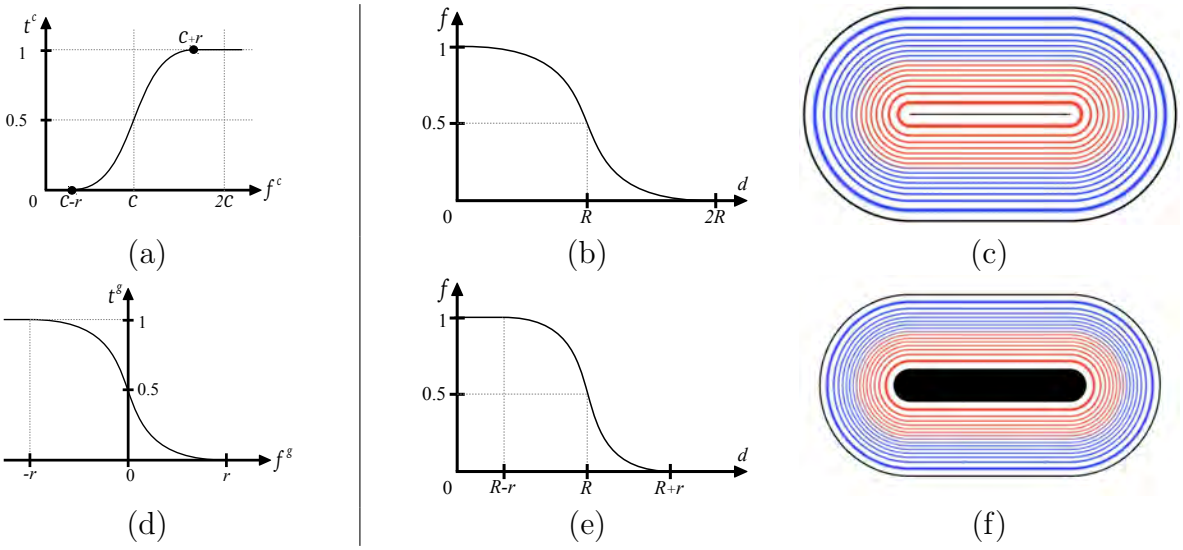


Figure 3.5: *Transfer functions for making compactly (a) and globally (d) supported field functions consistent. Varying the width of the variation band around the surface for $r = 0$ (b) to $0 < r < R$ (e) with the corresponding field for a segment-skeleton primitive (c,f).*

union operators must then be designed taking this construction into account, and so they must produce consistent field functions as output. As stated in Chapter 2, binary operators are of great interest for a constructive modelling process since they enable an advanced design of complex composition effects. Hence, we focus on binary operators, and more precisely on gradient-based operators [GBC+13] since they avoid several limitations of implicit modelling and provide a useful set of parameters.

In order to produce field functions respecting the newly defined property, i.e. field functionsbounded to 1, a new adequate operator g must satisfy several constraints. First, as stated by Barthe et al. [BWdG04], g must guarantee the continuity of the result field function $F = g(f_1, f_2)$ at the support boundaries of f_1 and f_2. Indeed, when the composition is applied, the operator must exactly reproduce f_1 outside the zone of influence and at the boundary of f_2 (i.e. where $f_2 = 0$), and respectively the other way around (where $f_1 = 0$). An observation in the operator space for compactly supported field functions as shown in the side Figure, is that $f_1 = 0$ corresponds to the Y-axis and $f_2 = 0$ to the X-axis of the operator space. Then the continuity of the operator at the support boundary is immediately obtained by imposing $g(f_1, 0) = f_1$ (along the abscissa axis) and $g(0, f_2) = f_2$ (along the ordinate axis). In previous works, g was only constrained to be positive on the $\mathbb{R}^+ \times \mathbb{R}^+$ domain. However, in order to keep the resulting field f consistent during the mod-

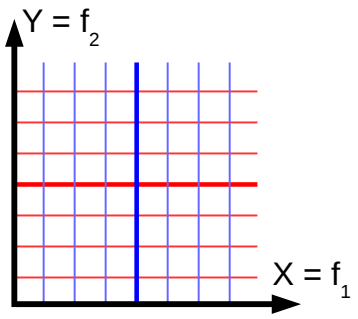


Figure 3.6: *The operator space for compactly supported field functions.*

elling process, another similar constraint has to be added along the inner boundary of the input fields (where $f_i = 0$). An analogous observation as for the support boundary can be made about the inner boundary. Indeed, $f_1 = 1$ corresponds to the vertical line $x = 1$ and $f_2 = 1$ to the horizontal line $y = 1$. The additional constraint can thus be set by imposing $g(f_1, 1) = g(1, f_2) = 1$, then making the new operator g a function $g : [0, 1]^2 \to [0, 1]$.

A naive way to enforce such a constraint on the inner bound is to clamp the resulting field function to the range $[0, 1]$. This solution, illustrated in Figure 3.7, is simple but brings two major artefacts into the resulting field. First, the clamping introduces a gradient discontinuity along the inner boundary regardless of the continuity of the input fields. Thus, composition operators with low degree of continuity cannot be used on the resulting field function. For instance, the popular sum operator from Blinn [Bli82] would be prohibited since it would lack gradient continuity.

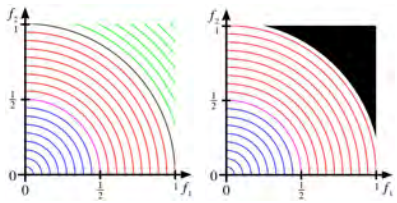


Figure 3.7: *Clamping the operators into* $[0, 1]$ *removes higher than* 1 *values (from green to black).*

Second, clamping the resulting field function will arbitrarily modify the width of the inner band of field variation around the surface. While implying a lower control over the composition during the modelling process, this undesired behavior increases with the number of overlapping compositions as shown in Figure 3.8.

We thus present a new set of binary composition operators avoiding the aforementioned problems and offering a conform composition model. These operators are adapted from the state of the art operators of Gourmel et al. [GBC+13] which are the most general and the most challenging to handle. A similar procedure can be applied to other families of binary operators such as those of Barthe et al. [BWdG04] and Bernhardt et al. [dGWvdW09].

As already mentioned in Chapter 2, the operators of Gourmel et al. are quasi-C^∞ operators interpolating between a clean-union and a smooth blend. These operators are built using the operator space for binary operators we also presented in Chapter 2. The operator space

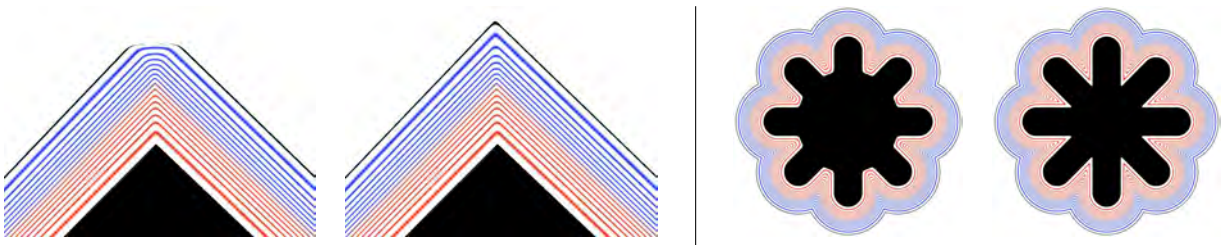


Figure 3.8: *Intersecting 2 planes (a-b) and composing 4 cylinders to build a star (c-d) using a clamped version of the clean union operator (a,c) leads to uncontrollable bound cutting, while with our operators (b,d) the bounds of the planes and cylinders are preserved.*

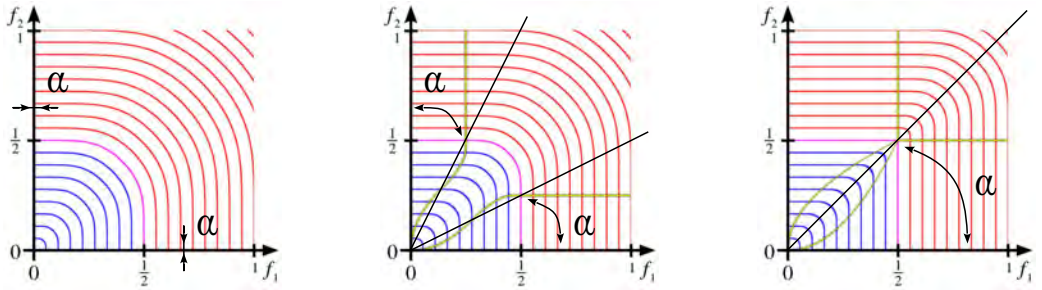
is split in two regions according to delimiting curves k_α^{base} as shown in Figure 3.9 in yellow. These curves are controlled by an angle α and are expressed as follows:

$$k_\alpha^{base}(f) = \frac{\tan(\alpha)}{2}\left(\frac{4}{1+\tan(\alpha)}\lambda_\alpha(f)\right)^2 \tag{3.5}$$

where the function λ_α is a C^∞ function that clamps the delimiting curve k_α when $f > \tan(\alpha)/2$. The use of trigonometric functions allows piecewise quasi-C^∞ continuity between the composition regions with one parameter only. Polynomials could also be used to provide high enough continuity but would require much more computations to fix the polynomial coefficients. Also to be noted, trigonometric functions allow a smooth and continuous interpolation between full composition and clean union, leading to midway operators respecting the same continuity property, which direct interpolation between operators does not provide.

In the first region between the axis and the curves k_α^{base}, a union is applied in order to keep the input field functions unmodified (see Figure 3.10(d) - black lines). The second region between the delimiting curves (see Figure 3.10(a,c,d) - gray) is where the composition is applied through a profile $\bar{s}$ describing the desired shape transition between the input primitives. This profile $\bar{s}$ is formulated in polar coordinates as $\rho = \bar{s}(\varphi)$ as illustrated in red in Figure 3.10(b). For the profile to connect the same isovalues of the input field functions, it is scaled to match the local frame corresponding to the intersection points between the delimiting curves k_α^{base} and the isoline of the input field functions (Figure 3.10(c)). The final operator is then built by reproducing the profile for each isovalue, as illustrated in Figure 3.10(d) - red curves. The only remaining parameters needed to design a new operator are the angle α and the composition profile $\bar{s}$ from which the operator is automatically built.

Setting α as a fixed parameter results in one possible operator for a given profile $\bar{s}$, meaning the exact same composition behavior is applied for every point lying in the zone of influence of both f_1 and f_2. In order to enhance user control and freedom, α can continuously vary



(a) (b) (c)

Figure 3.9: *Delimiting curves in yellow are defined according to an angle parameter α. New operators can produce maximal blend when $\alpha = 0$ (a), clean union when $\alpha = \pi/4$ (c) or intermediate blend (b).*

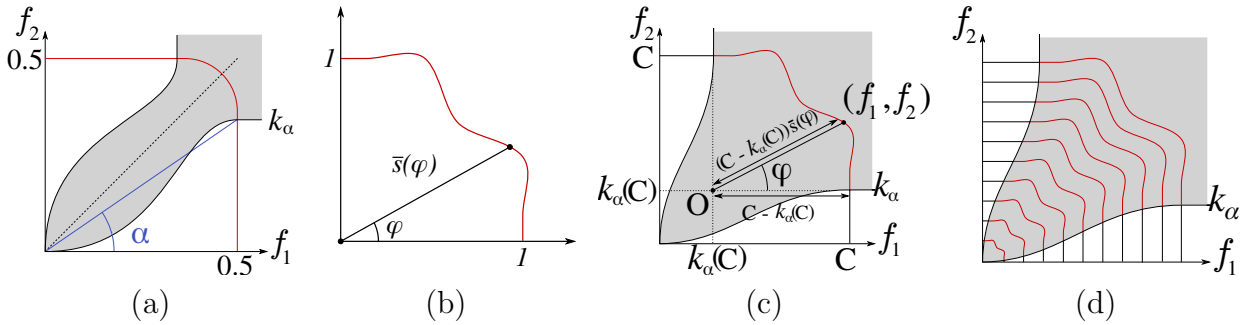


Figure 3.10: *Building composition operators by dividing the operator space (a). A profile describing the transition shape (b) connects same isovalues of the input field functions (c) in the delimited region, while a union is applied outside it. The final operator is then automatically built by connecting all the isovalues (d).*

from one evaluation point to another in order to perform different composition behaviors, in the same application, depending on some criteria. Gourmel et al. propose to make α vary according to the angle θ between the gradients of the input field functions, as presented in Figure 3.11(b). The function $\alpha(\theta)$ (for example in Figure 3.11(a)) is called a controller and provides the user with additional control over the behaviors of the operator according to the angle θ. Hence, by the mean of the controller, the operator can automatically interpolate between full blending when $\alpha(\theta) = 0$ and clean union when $\alpha(\theta) = \pi/4$ at different evaluation points. This results in different composition behaviors applied in one operation as illustrated in Figure 3.11(c), then solving all the blending issues presented in Chapter 2.

However, these operators introduce values greater than 1 in the resulting field function (in green in Figure 3.7-left), which are not consistent with our field representation and the use of the complement operation. Then, to make these operators conformal with our new representation, we need to ensure that g respects the inner bound of the input field functions, i.e. $g(f_1, 1) = g(1, f_2) = 1$. This is automatically achieved by designing the delimiting curves k_α^{base} such that they meet at $f_1 = f_2 = 1$, thus "closing" the blending region of the operator as depicted in Figure 3.12.

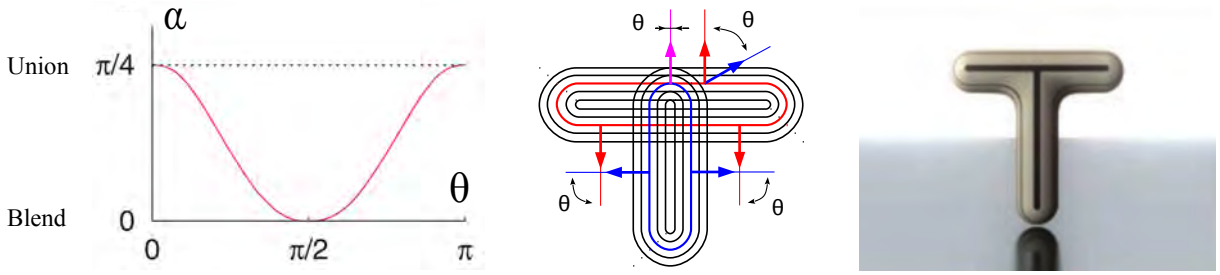


Figure 3.11: *Applying Gourmel operators on two cylinders prevents all blending issues. The presented controller function (a) maps the angle θ between the input gradients (b) to the parameter α in order to produce a nice blending only when the surfaces are not tangential. Figures from [GBC+13].*

Our first attempt to close the curves k_α^{base} was to use Hermite curves, defined in operator space as illustrated in the side Figure. These curves go from the points $\mathbf{P_1} = (0.5, k_\alpha^{base}(0.5))^t$ and $\mathbf{P_2} = (k_\alpha^{base}(0.5), 0.5)^t$ to the point $\mathbf{P_0} = (1,1)^t$ with tangent vectors T_{11}, T_{12} (in red) and T_{21}, T_{22} (in green) respectively. While closing the blending region of the operator as illustrated in Figure 3.12-top row, this formulation presents some drawbacks. First, the C^∞ continuity of the delimitation curves is lost at points $\mathbf{P_1}$and $\mathbf{P_2}$ which stand on the surface, then loosening the surface continuity. Second, tweaking the tangent vectors T_{11} and T_{21} at points $\mathbf{P_1}$ and $\mathbf{P_2}$, and T_{12} and T_{22} at point $\mathbf{P_0}$ to produce smooth transitions between successive values of α is a bit cumbersome and may produce precision issues in the computation of the operator's field in the vicinity of point $\mathbf{P_0}$.

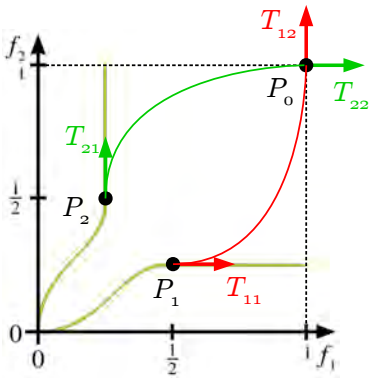


Figure 3.13: *Closing the composition region using Hermite curves.*

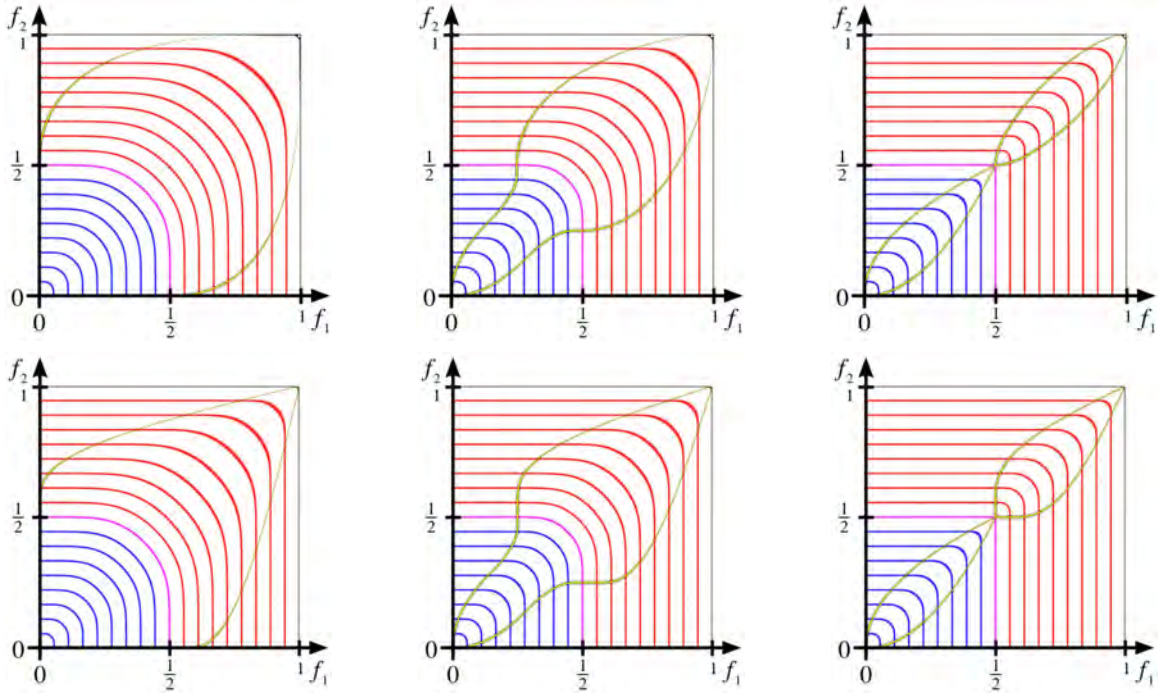


Figure 3.12: *Top row: closing Gourmel et al. operators using Hermite curves provides the expected composition behavior, but may introduce precision issues in the vicinity of the corner point. Bottom row: our closed version using trigonometric curves provides higher continuity and better precision. Three different values of θ are shown, producing (a) full blending, (b) intermediate blending, and (c) clean-union.*

We thus decided to use trigonometric and hyperbolic functions for their natural C^∞ continuity and to compose them for controlling the slope of their shape. The new design of the delimiting curves k_α is the following:

$$k_\alpha(f) = \begin{cases} k_\alpha^{base} & \text{if } f \leq 0.5 \\ \frac{1}{2}\left(\left(\frac{\tau(f)}{\tanh(1)} + 1\right)(2 - \tan(\alpha)) + \tan(\alpha)\right) & \text{otherwise} \end{cases}$$

where $\tau(f) = (\tanh \circ \tanh \circ \tan)(\pi(f-1))$, $f \in [0,1]$.

As for the operators by [GBC+13], this construction does not present a simple closed-form formulation and we precompute our operators into 3D textures to be fetched for the evaluation. Hence, the supplemental cost due to the use of trigonometric and hyperbolic functions does not matter when field evaluations are performed for rendering the designed object. Although texture fetch implies a trilinear interpolation which breaks the C^∞ continuity of the operators, using a high enough texture resolution prevents visual artefacts, even after several compositions, as shown in Figure 3.14.

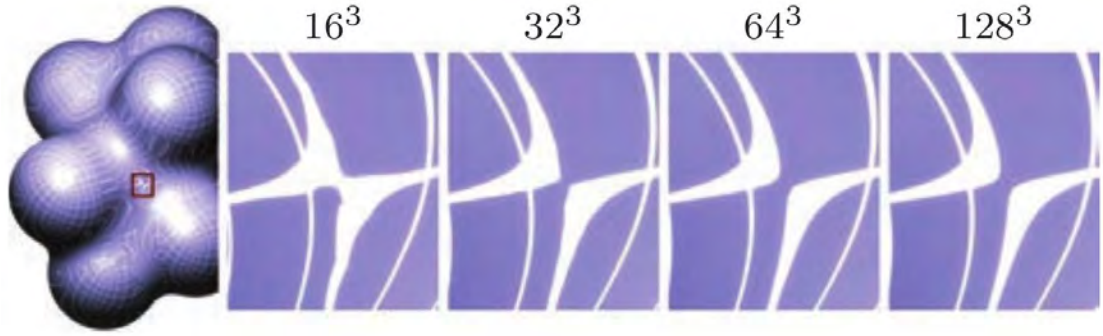
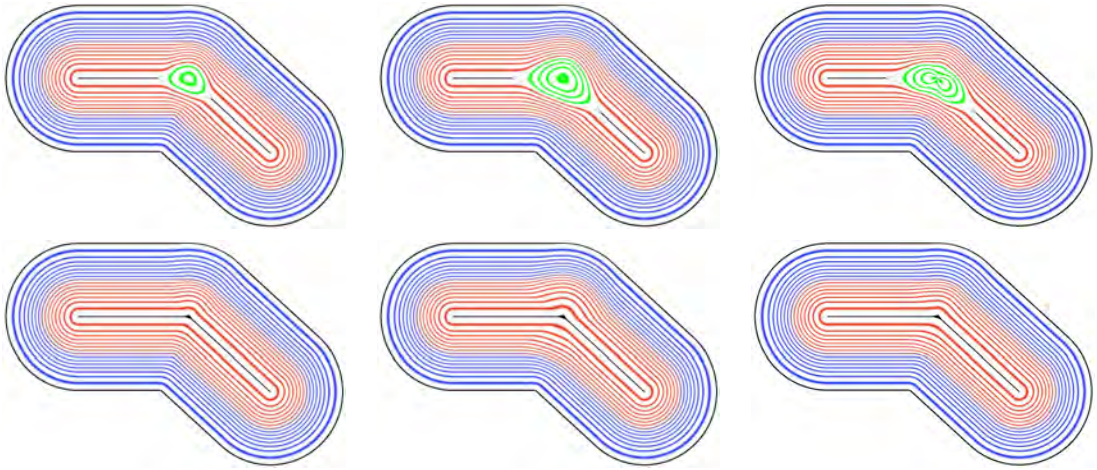


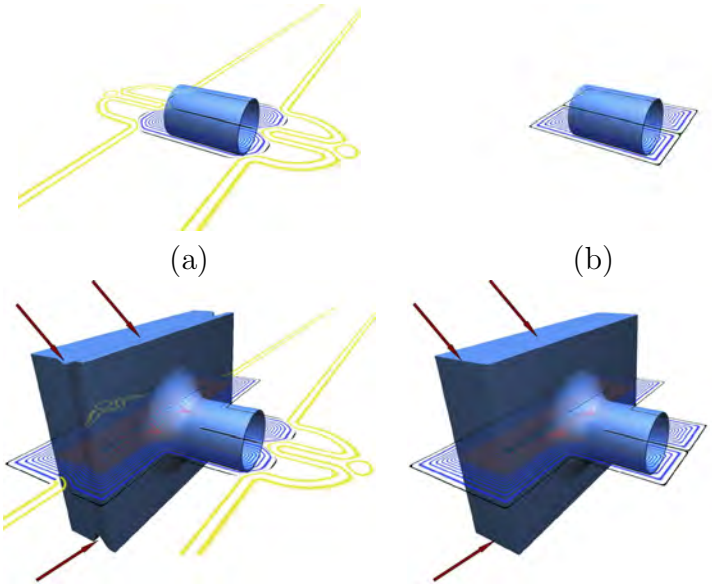
Figure 3.14: *Using a high enough texture resolution prevents visual artefacts as shown by the zoom-ins for different resolutions. Figure from [GBC+13].*

We now have all the ingredients to build artefact-free intersection and difference operators by combining our modified union and blending operators following Equations 3.1 and 3.2. The benefits of our approach are depicted in Figure 3.15 for the cases of a clean union, a blend and a gradient-based blend. Closing the operators avoids the appearance of the depression in the inner part (green isocurves). While not really problematic for the union and blending compositions, this depression becomes really disturbed when a gradient-based blending is applied. Thus, by removing these depressions and distortions, our operators can better handle subsequent gradient-based compositions. Figure 3.16 also demonstrates the effects of our modifications for the cases of intersections and differences. As we can see, the negative values (shown in yellow) produced by state of the art operators when building the cylinder are avoided by our operators. This prevents the cuboid from being unexpectedly deformed by the presence of negative values when it is blended with the cylinder and to obtain the expected result.



(a) (b) (c)

Figure 3.15: *Closing the boundary functions of a gradient-based blending operator leads to better shaped resulting scalar fields. Applications of previous operators (top row) and our operators (bottom row) using (a) union, (b) clean union and (c) "camel" blending.*



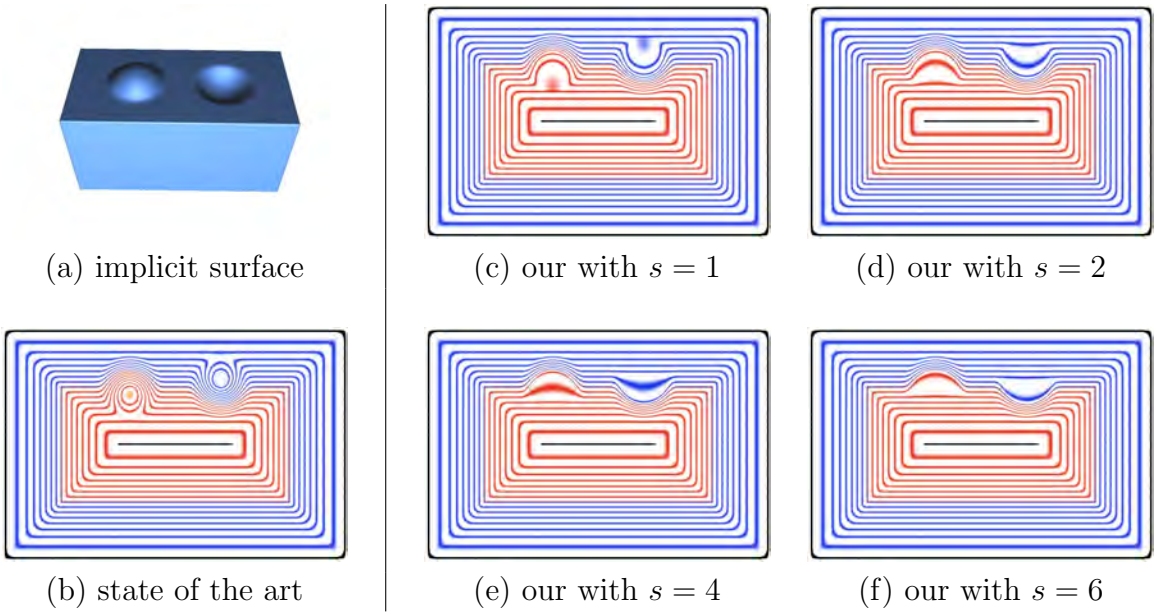
(a) (b) (c) (d)

Figure 3.16: *Top row: a hollow cylinder is built by removing a cylinder from a larger one and then intersecting with 2 planes. The use of Gourmel's operators [GBC+ 13] on the left produces negative field values (a) that lead to an inadequate deformation of the blended cuboid on its side, where pointed in (c) by the red arrows. This unexpected behavior is naturally avoided by the use of our operators as shown on the right (b-d).*

3.3 Operators for Details

Another aspect of constructive modelling is the addition of details to the object. In this case, the global shape of the surface remains the same while only specific, local portions of it are modified. Hence, since the field around the surface generally represents a distance field, its shape is expected to globally resemble the object surface and to be modified only locally around the details.

This is not the case with state of the art operators, including the ones we introduced in the previous section, which leave depressions in the field where detail material is added or subtracted, as shown in Figure 3.17(b). These depressions are undesired from both a theoretical point of view since they disturb the metric of the input fields, and from the practical point of view as they introduce unpredictable shape deformations when crossed by a blend (see Figure 3.20-left).



(a) implicit surface (c) our with $s = 1$ (d) our with $s = 2$

(b) state of the art (e) our with $s = 4$ (f) our with $s = 6$

Figure 3.17: *A cuboid (a) on which two spheres have been added (using a blend) and removed (using a difference with smooth transition). While this surface is the same for all operators, the resulting field has different local variations where the spheres are combined. This is illustrated in the planar section of the resulting field function passing by the centers of the added and removed spheres. State of the art operators (b) create field depressions. Our new detail operator (c-f) absorbs the blended/subtracted field functions. The absorption is modulated by the parameter s.*

In order to handle the addition of details during the modelling process, we designed a new set of specific operators $\tilde{g}_\theta$ according to the following reasoning. Where a detail is added to a surface, the resulting field is expected to progressively vary from the one of the detail to the field of the main object when moving away from the surface (as shown in Figure 3.17(c-f)). Assuming f_1 to be the main object and f_2 the detail to be added or removed, the operator $\tilde{g}_\theta$

must preserve the properties of f_1 while continuously vanishing f_2 so that its field is smoothly absorbed. This means that in the specific case of details, the constraints $\tilde{g}_\theta(f_1, 0) = f_1$ and $\tilde{g}_\theta(1, f_2) = 1$ must still hold, but the constraints $\tilde{g}_\theta(0, f_2) = f_2$ and $\tilde{g}_\theta(f_1, 1) = 1$ may not be respected for $f_2 > 0.5$.

Then, in order to keep high control on the way the details are added, our new dedicated operators are gradient-based. They reproduce the behavior of our previously defined gradient-based operators g_θ in the outer part of both input field functions f_1 and f_2 but progressively reproduce f_1 when moving away from the 0.5 isovalue. This behavior of such an operator $\tilde{g}_\theta$ is depicted in Figure 3.18, where its isolines are the ones of a standard gradient-based operator g_θ where $\tilde{g}_\theta(f_1, f_2) \leq 0.5$, but progressively "interpolates" between the 0.5-isoline to the straight vertical line at $f_1 = 1$.

In order to obtain this type of behavior, we first tried to define an interpolation parameter $\lambda(f_1, f_2)$ allowing the interpolation between the fields of $\tilde{g}_\theta$ and f_1. However, designing λ to obtain the desired operator and controlling it appeared to be too complicated and we rather decided to take advantage of the 2D representation for binary operators. We propose the following procedure to design the operator $\tilde{g}_\theta$ directly in the 2D space (see Figure 3.19) by precisely defining the shape of each individual isocurve. Here again, we do not provide a simple closed-form formulation and the operators are baked into 3D textures.

Given a point $\mathbf{x} = (f_1, f_2)^t$, $c = \tilde{g}_\theta(\mathbf{x})$ is the value we want to compute. If $\mathbf{x}$ is in the outer part of the field, i.e. $g_\theta(\mathbf{x}) \leq 0.5$ (under the magenta curve in Figures 3.18 and 3.19), then we want the initial operator, i.e. $c = g_\theta(\mathbf{x})$. Likewise, if $\mathbf{x}$ is outside the blending

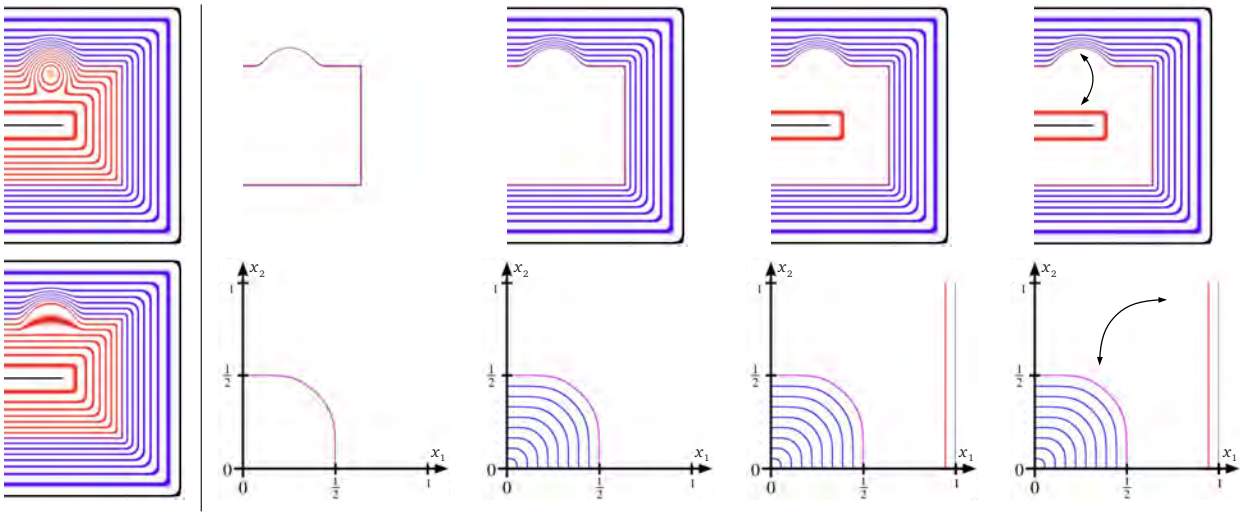


Figure 3.18: *Desired behavior for a detail-specific operator. Far left: current behavior and desired behavior of the resulting field. Top-row: decomposition of the desired behaviour. We want the same surface, with the same exterior field, the reproduction of the global object in the most inner part and interpolation between the first and the latter. Bottom-row: translation of the desired field behavior in the operator space.*

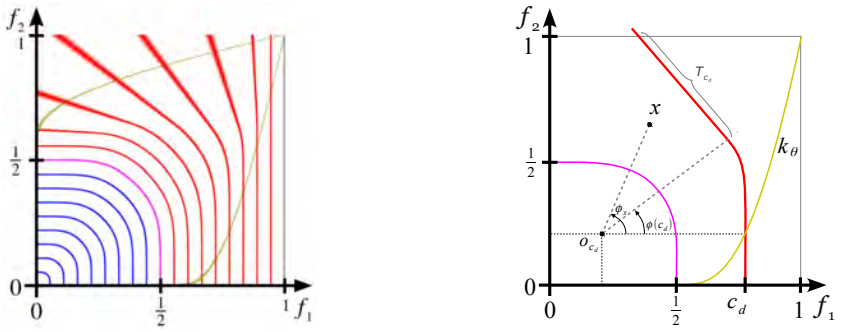


Figure 3.19: *(a) Illustration of our asymmetric operator for details $\tilde{g}_\theta$ with $\theta = 0$ and $s = 1$. (b) Illustration of the construction of this operator. The c_d-isocurve shown in red is obtained by cutting the original isocurve of g_θ at a polar angle $\phi(c_d)$ and prolonging it by a straight line T_{c_d}. The evaluation of $\tilde{g}_\theta$ at an arbitrary position $\mathbf{x}$ is performed by iteratively determining whether $\mathbf{x}$ is above or below the isocurve of the current guess c_d. The profile curve k_θ is shown in yellow, and the 0.5-isocurve in magenta.*

region, i.e. under the curve k_θ shown in yellow, then by construction we want $c = f_1$ to maintain continuity. Otherwise, c is numerically evaluated with a dichotomic search in the range $[0.5, 1]$, starting with $c_d = 0.75$ as the initial guess. If $\mathbf{x}$ is below the c_d-isocurve shown in red in Figure 3.19-right, we continue the search within the range $[0.5, c_d]$. In the opposite case, we continue the search within the range $[c_d, 1]$.

To determine the position of $\mathbf{x}$ with respect to the c_d-isocurve, we express its position in polar coordinates $(\phi_\mathbf{x}, \rho_\mathbf{x})$ in the local frame centered at $\mathbf{o}_{c_d} = (k_\theta(c_d), k_\theta(c_d))^t$. Then three cases occur:

- If $\phi_\mathbf{x} \leq 0$, then $\mathbf{x}$ is below the c_d-isocurve.
- If $\phi_\mathbf{x} \leq \phi(c_d)$, then $\mathbf{x}$ is below the c_d-isocurve if $\rho(\phi_\mathbf{x}) < \rho_\mathbf{x}$, where $\rho(\phi)$ is the profile curve used to define the blending operator g_θ, and above otherwise.
- Otherwise we directly check whether $\mathbf{x}$ is above or below the tangential half-line T_{c_d}.

This procedure makes the c_d-isocurves of g_θ straight as soon as they reach the angle $\phi(c_d)$. This angle ϕ is designed to smoothly vary between $\pi/2$ and 0: when $\phi = \pi/2$ we reproduce g_θ while f_1 is reproduced when $\phi = 0$. We thus take the following formulation for ϕ:

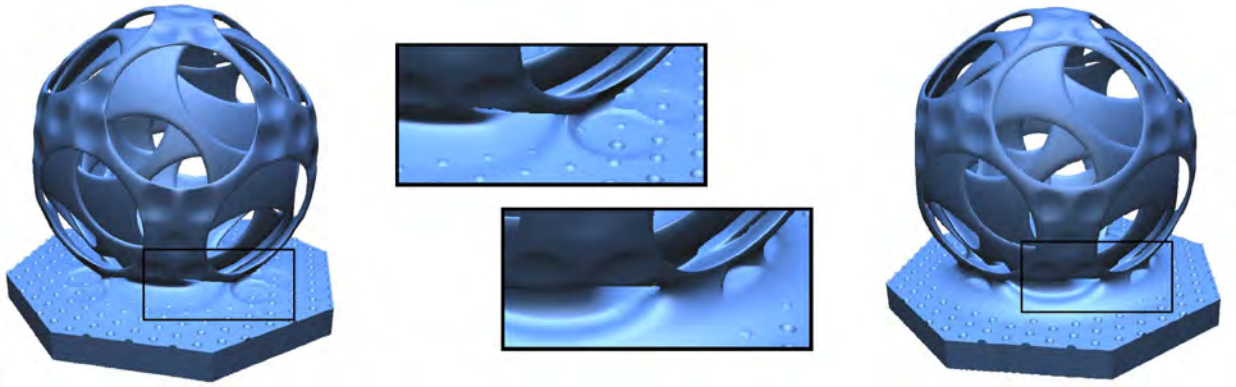
$$\phi(t) = \frac{\pi}{2}(2 - 2t)^s, \tag{3.6}$$

where the exponent s adjusts the interpolation speed between the 0.5-isovalue of g_θ and the field of f_1. Applications of this operator are shown in Figure 3.17 for different values of the parameter s which modulates the absorption of the blended/subtracted field. In practice, we suggest $s = 4$ which has been used for all our experiments and our results figures.

3.4 Results

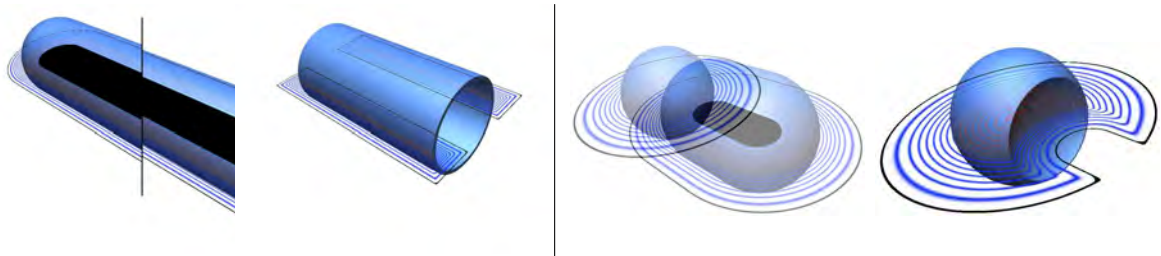
Using both our compact support field representation and our new operators, we can now design complex objects with adequate field variations and metrics in their inside part. These objects can be drilled with a guaranty that the resulting object is well shaped. Figure 3.20 illustrates a complex object built with several differences and blends. As we can see on the left of Figure 3.20, despite all their nice properties, Gourmel et al. operators fail in preserving field variations and metrics when several difference operations at a different scale are applied. The field of the large drilled spheres has been altered and a consequence is the asymmetry in the blend between the large spheres and the pedestal. As shown in the right of Figure 3.20, using of our operators prevents these alterations of the field and the blend is symmetric.

When applying a difference operation, a very interesting observation is that the inner field of the subtracted primitive defines a part of the outer field of the result. Another important observation is that after a composition, we expect the resulting field to follow the shape of the resulting surface as if approximating the variations of a distance field. Without inner field control, this is not the case in the inner field part when clean union and blending are used as illustrated in Figure 3.15 where unexpected field depressions arise. This is also usually not the case for the difference. The use of an inner bound together with the adapted operators avoids this problem. Reducing the inner radius of the combined objects enables the generation of a band around the implicit surface, as for the tubes in Figure 3.21 (a-b). In this band, the field smoothly approximates a distance field with the metric of the composed field functions. Figure 3.21 (c-d) shows a similar control on a capsule subtracted from a sphere.



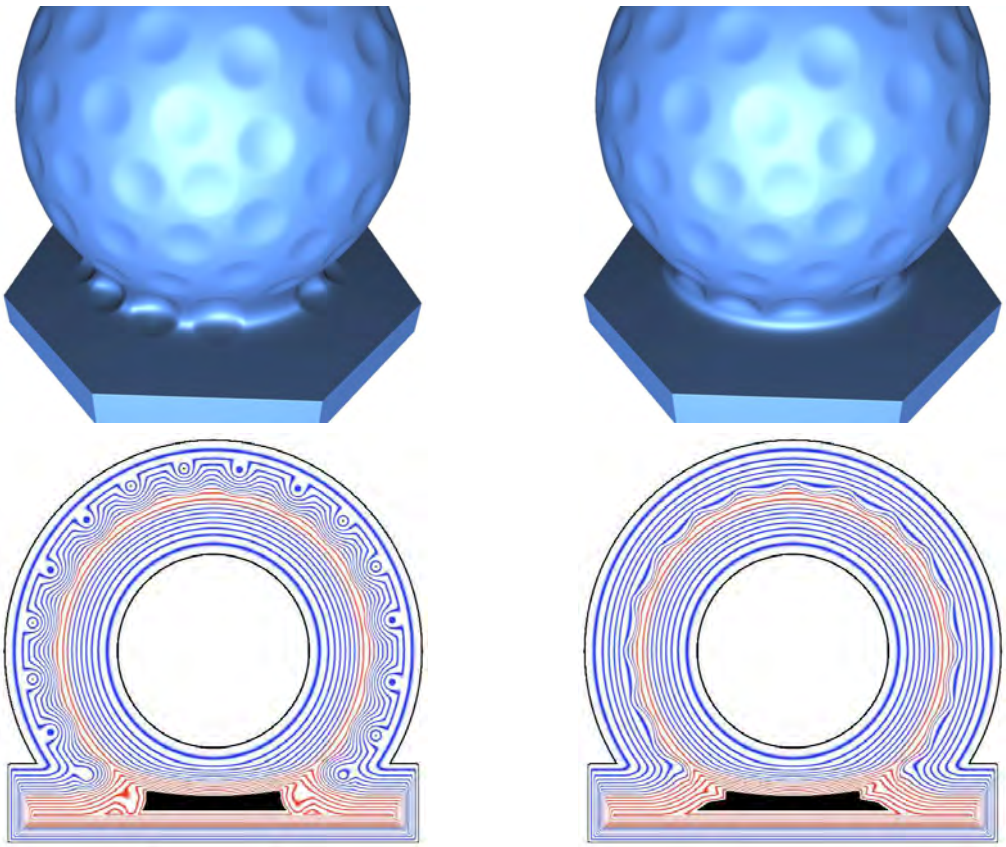
state of the art operators our operators

Figure 3.20: *A complex model built (left) with Gourmel et al. gradient-based operators and (right) with our novel operators. As we can see in the zoom in the middle-top, details on the spheres consequently deform the blend between the spheres and the pedestal while in the middle-bottom our new operators preserve the blending shape. In addition, in the middle-top the same blending shape is smooth on the pedestal and unexpectedly sharp on the sphere, while in the middle-bottom it is nicely smooth on both objects with our operators.*



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

Figure 3.21: *Adjusting the external boundary using the inner bound and subtracting field functions. A cylinder is properly removed from a slightly larger one (a) and intersection with planes are used to get the final result (b). (c-d) A capsule is removed from a sphere. Observe in (b) and (d) how the field approximates a smooth distance field around the implicit surface.*



(a) state of the art (b) ours

Figure 3.22: *Illustration of our new operator for details. (a) The details are created using our difference operator and (b) using our new detail-specific difference operator. Note the field depressions introduced (a)-bottom and the resulting blend deformation between the ball and the pedestal (a)-top that are avoided (b) with our detail-specific operator.*

While symmetric operators are well suited for large-scale compositions (Section 3.2), when modelling small features, our detail-aware operator becomes preferable (Section 3.3). This is demonstrated in Figure 3.22 where a golf-ball like shape is obtained by removing small spheres from a large one, and then blending the result with a pedestal.

Using symmetric difference operators, the depressions introduced in the field when removing the small spheres distort the blend between the ball and the pedestal (Figure 3.22 (a)). This behavior is undesired and unexpected as these small spheres just represent a detail and the blend should mostly be as the one linking the ball and the pedestal. Figure 3.22 (b) illustrates the improvement obtained by our new detail specific operator. These behaviors are also illustrated in Figure 3.20 and 3.23. Figure 3.23 also illustrates the field variations generated when objects are built using our compactly supported field representation together with our composition operators.

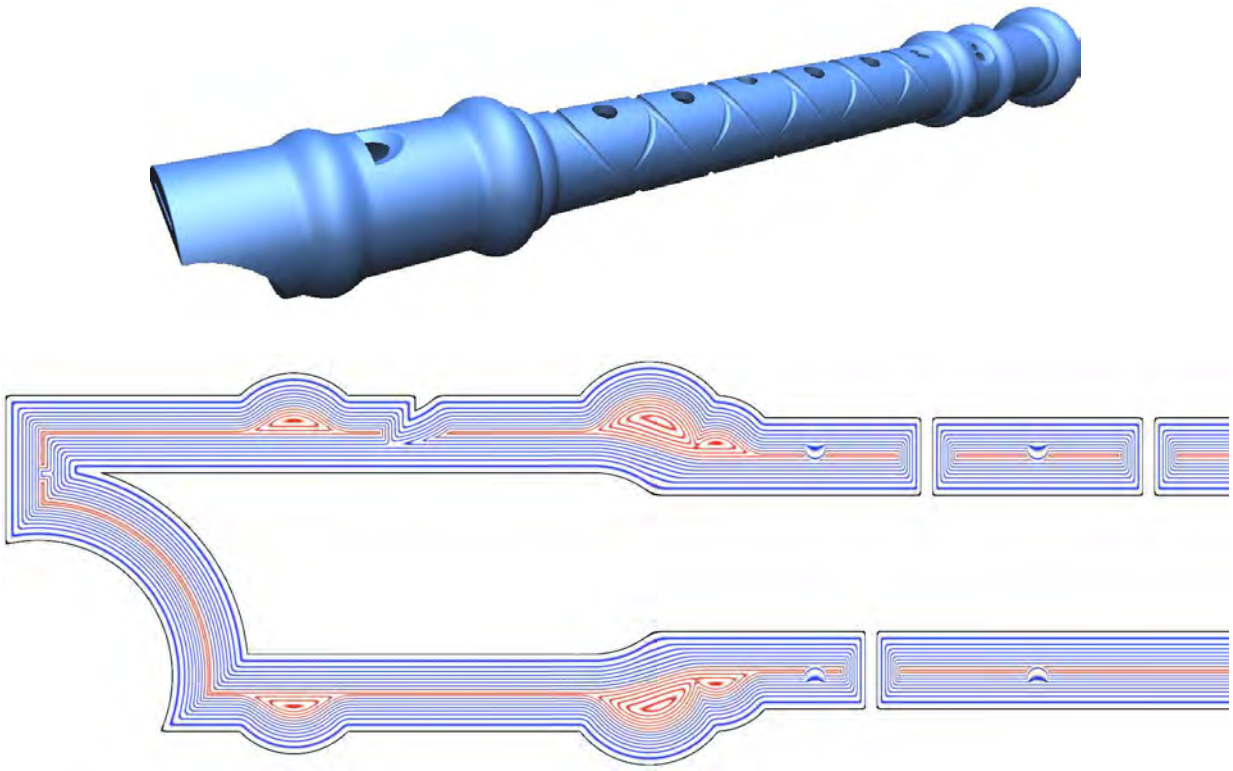


Figure 3.23: *A flute model built using our compactly supported field functions and our adapted composition operators including detail-specific operators. The quality of the field variations is illustrated on a vertical section of the left side of the field function. Note that unexpected depressions are avoided and the field approximates a distance field in bands located on each side of the implicit surface.*

3.5 Conclusion

In this chapter, we have introduced new constraints on field functions so that intersection and difference composition operators are applied in a consistent manner, avoiding field distortions and discontinuities. We also provide a method to build composition operators satisfying those constraints when intersection and difference operators are derived from union or blending. This method takes advantage of the 2D representation for binary operators which allows a better understanding of what would happen when an operator is applied and facilitates the design of new operators. Combining these contributions allows, when applying difference operators, to dig the outer bound of the resulting field function so that its shape follows the shape of the surface.

Finally, we have introduced a new specific composition operator for the modelling of thin details on a surface. This operator smoothly absorbs the removed field, thus avoiding the introduction of undesired depressions in the resulting field function that would degrade the shape of subsequent smooth transitions. These advanced operators do not yield analytic formulas and have to be precomputed into tables to enable fast evaluations.

As future work, it might be interesting to derive analytic formulas reproducing our operators, even if that means losing C^∞ continuity. The study of the transcription of the desired operator shape in the 2D space into an effective operator could also be of great interest. For example, defining some constraints as in [BGC01, BDS+03] and diffusing them in a grid sampling the 2D space is an interesting alternative to look at. The study of interactive visualizations, especially for the creation of small details and field function based micro geometries would also be of interest. Finally, the way the details could be positioned and repeated on the surface is another direction to investigate.

Chapter 4
Handling topology issues in particle-based Fluid Simulation

Accepted Submission: *Topology-Aware Neighborhoods for Point-Based Simulation and Reconstruction*, F. Canezin, G. Guennebaud, L. Barthe. ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA) 2016.

4.1 Fluid Simulations

Fluid simulations are an important component of computer graphics, used in real-time applications such as video games and offline applications such as visual effects in movies and simulations. Going from liquids to foam or gases (Figure 4.1(a-b)), fluid simulation is a very active research area for which more and more realistic solutions are expected. Fluid simulations are based on the Navier-Stokes equations describing incompressible fluid dynamics:

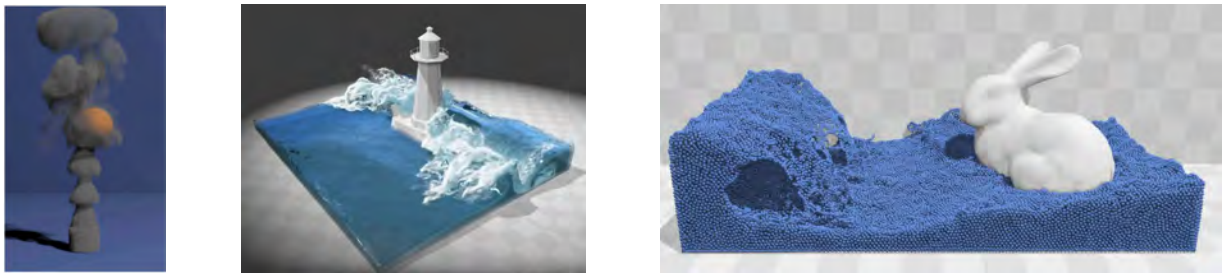
$$\rho\left(\frac{\partial v}{\partial t} + v.\nabla v\right) = -\nabla p + \rho g + \mu\nabla^2 v \tag{4.1}$$

where v is the fluid's velocity field, ρ its density, μ its viscosity, p the pressure and g an external force field (usually the gravity). The additional continuity equation is needed to ensure mass conservation:

$$\frac{\partial \rho}{\partial t} + \nabla(\rho v) = 0 \tag{4.2}$$

as well as a state equation for energy conservation, for which numerous formulations have been developed according to the properties of the simulated fluid (compressible or not, gas, liquid, etc.).

The different methods for modelling and simulating fluids can be classified in three categories: Eulerian methods, Lagrangian methods and hybrid methods. Eulerian methods are based on a regular discretization of the simulation domain tracking the fluid properties, such as density and velocity, in the vicinity of the discretization points. The surface of the fluid, e.g. the interface between its volume and the environment, is then retrieved using Level Set Methods [OF06], Volume of Fluid Methods [PAB+97] or Density-based Approaches [MMTD07]. All these surface reconstruction methods build an implicit surface from the fluid properties at the discretization points.



(a) (b) (c)

Figure 4.1: *Fluid simulations can be used for modelling both gases (a) or liquids (b) dynamics. In particle-based fluid simulations, the particles sample the volume of the fluid (c). Figures from [WYY13, MM13].*

On the other hand, Lagrangian methods rely on a set of particles sampling the volume of the simulated fluid and carrying its properties (mass, velocity, density) when advected through the physics equations, as illustrated in Figure 4.1(c). These properties then evolve during the simulation according to the influence of the local environment (collision with obstacles, external forces, etc.) and the particle interactions within the fluid. Among the different methods, the popular Smoothed Particle Hydrodynamics (SPH) method was first introduced for astronomical simulations [Luc77, GM77]. They became very popular in computer graphics for their efficiency when they are applied to fluid simulations [MP89, MCG03, MSKG05, BTT09] and are now widely used by the animation industry. In order to extract the surface of the fluid, particles contributions to the volume of the fluid are combined, thus generating the final implicit surface for the whole fluid. In the work presented here we tackle problems in surface reconstruction and focus on particle-based fluid simulations [IOS+14].

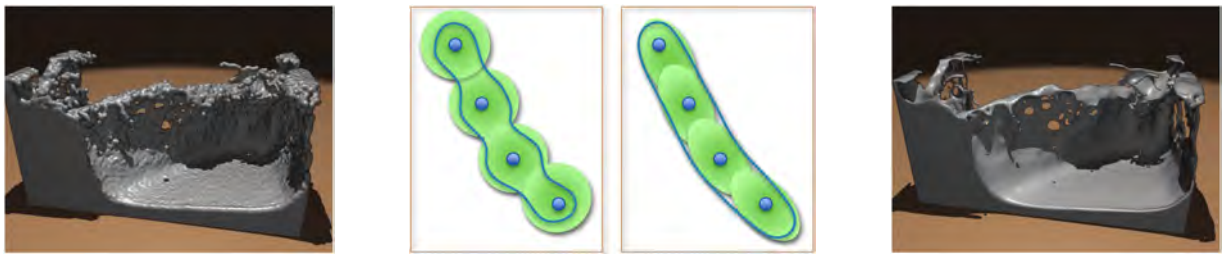
4.1.1 Fluid Surface Reconstruction

In particle-based simulations, the fluid surface is defined as an isosurface in a 3D scalar field computed from the particles properties. A first method to build such a scalar field, originated from the SPH formulation, is to sum density field functions attached to the particles and representing their contribution to the volume of the simulated fluid:

$$\phi(\mathbf{x}) = \sum_i f_i(\mathbf{x}) = \sum_i \frac{m_i}{\rho_i} W(\mathbf{x} - \mathbf{p}_i), \tag{4.3}$$

where $\mathbf{p}_i$ is the particle position, m_i its mass, ρ_i its density and W is a compactly supported kernel of radius twice the average particle distance h [Mon92]:

$$W(\mathbf{r}) = \frac{\sigma}{h^d} P\left(\frac{\|\mathbf{r}\|}{h}\right) = \frac{\sigma}{h^d} \begin{cases} 1 - \frac{3}{2}v^2 + \frac{3}{4}v^3 & \text{if } 0 \leq v \leq 1 \\ \frac{1}{4}(2-v)^3 & \text{if } 1 \leq v \leq 2 \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$



(a) (b) (c)

Figure 4.2: *Using isotropic spheres (a) lead to a "blobby" surface while anisotropic ellipsoids (b-c) produce a smooth surface. Figures from [YT10, Mül11].*

where d is the dimension of the ambient space, σ is a normalizing factor and $v = \|\mathbf{r}\|/h$. The main limitation of this method is that the sum generates an oscillating surface exhibiting bumps around the particles (see Figure 4.2(a)). These oscillations are due to the spherical shape of the density functions attached to the particles. Addressing these oscillations is one of the main concern of recent fluid surface reconstruction methods we present in the following. In addition, as explained in Section 2.2.2, the sum suffers from the blending at a distance effect which is unnatural for a fluid surface and results in a reconstructed surface with an incorrect topology.

As a way to address this limitation, Yu and Turk [YT10] propose to use anisotropic density functions based on the neighborhood particles distribution around their barycenter:

$$W(\mathbf{r}, A) = \sigma det(A) P(\|A\mathbf{r}\|) \tag{4.5}$$

where A is a $d \times d$ matrix encoding the anisotropy coming from the neighbors distribution. Hence, particles contributions are transformed from spheres to ellipsoids stretched in the surface direction and elongated in the tangential directions, as shown in Figure 4.2(b). When summed, these anisotropic density functions result in a fluid surface that still oscillates but at an indistinguishable level, as presented in Figure 4.2(c).

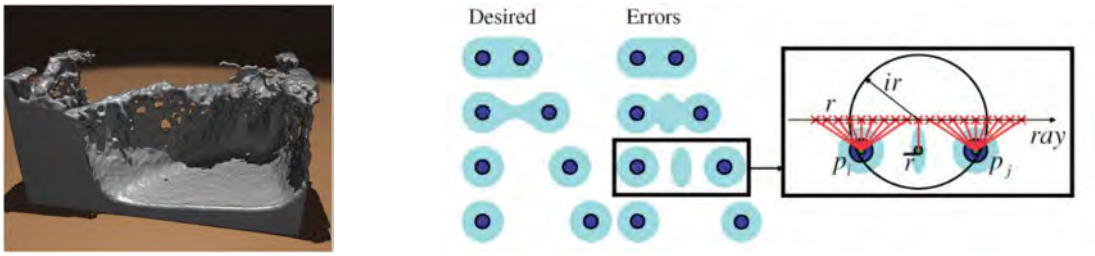
A second family of approaches tries to build a scalar field as smooth as possible through distance field functions. Zhu and Bridson [ZB05] define a distance field function from a weighted average of both the neighboring particles positions $\bar{\mathbf{x}}$ and their radius $\bar{r}$:

$$\phi(\mathbf{x}) = \|\mathbf{x} - \bar{\mathbf{x}}\| - \bar{r} \tag{4.6}$$

$$\bar{\mathbf{x}} = \sum_i w_i x_i \tag{4.7}$$

$$\bar{r} = \sum_i w_i r_i \tag{4.8}$$

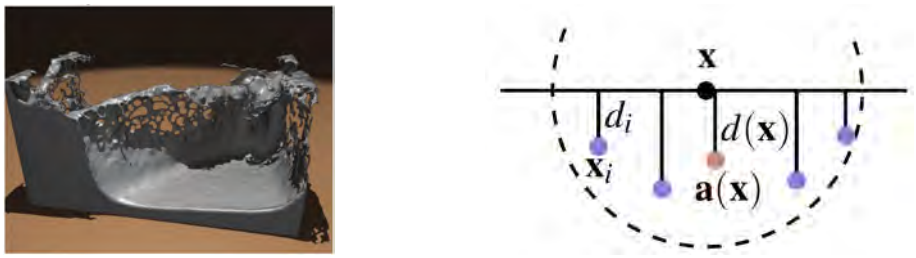
$$w_i = \frac{K(\|\mathbf{x} - \mathbf{x}_i\|/2h)}{\sum_j K(\|\mathbf{x} - \mathbf{x}_j\|/2h)} \tag{4.9}$$



(a) (b)

Figure 4.3: *Averaging neighboring particles positions and radii [ZB05] creates a smooth distance field (a) that is enhanced by using their derivatives to avoid separation artefacts (b). Figures from [YT10, SSP07].*

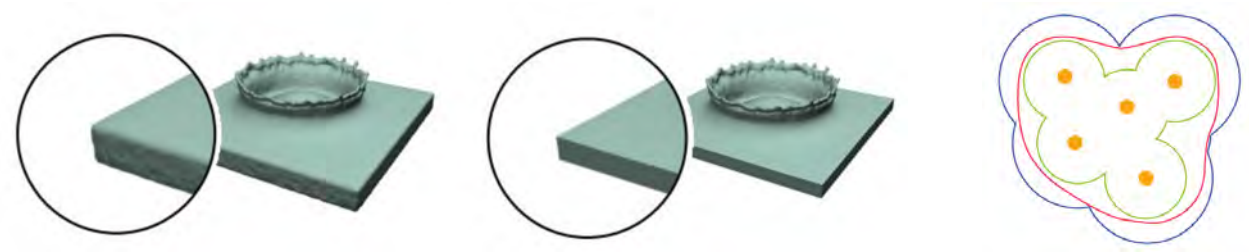
where K is a smooth-step function. While smoothing the surface, this method produces artefacts in the reconstruction occurring when fluid components split, as illustrated in Figure 4.3. Solenthaler et al. [SSP07] then improve this method and remove these artefacts using the derivatives of $\bar{\mathbf{x}}$ to smooth its variation for small $\mathbf{x}$ variations. Extending this approach, Adams et al. [APKG07] propose to use particle-to-surface distances d_i, which are computed at surface particles and propagated into the fluid, instead of particles radii, as illustrated in Figure 4.4. Moreover, these distances are recomputed after each simulation step by projecting particles near the surface onto it and propagating the new distance into the fluid. This brings temporal coherence into the reconstructed surface by avoiding sudden unnaturally large modifications.



(a) (b)

Figure 4.4: *Defining the surface as a distance field generated by tracking the particle-to-surface distance [APKG07] (b) also smooths the resulting fluid surface (a). Figures from [YT10, APKG07].*

A last family of surface reconstruction approaches applies a post-treatment to smooth the resulting surface. Once the mesh surface is extracted by means of Marching Cubes [WMW86, LC87], it can be smoothed via multiple iterations of Laplacian and Bilapacian mesh smoothing methods as proposed by Williams [Wil08]. The other way to perform this post-treatment is to directly smooth the scalar field before extraction by minimisation of the biharmonic energy as proposed by Bhatacharya et al. [BGB11, BGB15]. Both approaches aim at smoothing the surface with the constraint that it should remain between two unions of spheres of different radii centered at the particles positions, as illustrated in Figure 4.5.



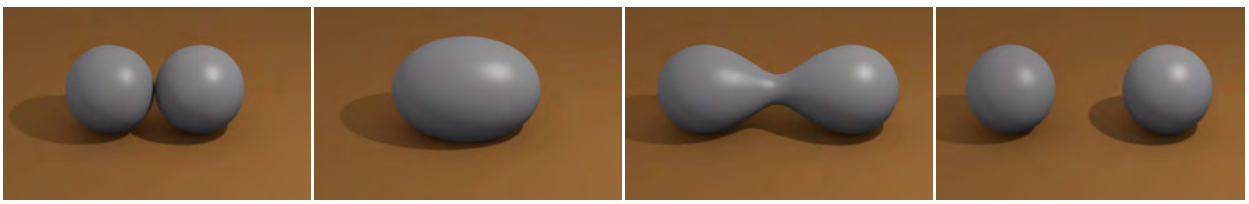
(a) (b)

Figure 4.5: *Smoothing the result surface as a post treatment (a) with the constraint (b) it (red curve) should stay between two unions of spheres (blue and green). Figures from [Wil08, BGB11].*

4.1.2 Fluid Topology

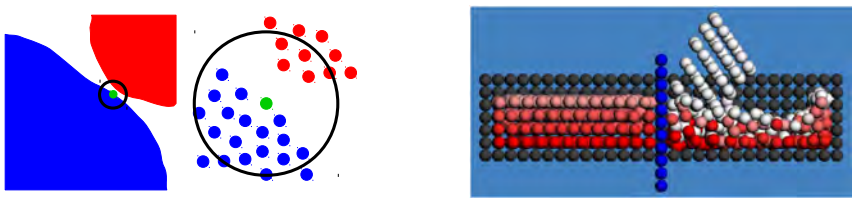
In all these methods the reconstruction is performed independently at each frame, and thus does not bring temporal coherence in the surface topology (except for Adams et al. [APKG07]) nor consistency between the simulated fluid and the reconstructed surface. Currently, no blending operator nor reconstruction method is able to reconstruct a smooth fluid surface respecting the fluid topology and the asymmetry of the fusion/separation effects in a particle-based simulation. Indeed, when fluid components come close to one another, they do not attract nor blend with each other until contact is made, in which case they immediately merge as illustrated in Figure 4.6(a-b). On the opposite, when fluid components are going to split, they still blend together until the very moment of the separation, then leading to a thin trickle connecting them as in Figure 4.6(c-d). When separation is complete the trickle is split and each component brings back its part of it. Hence, a different blending behavior is expected between fluid components according to the involved topological events. Our solution to the surface reconstruction topological issue solves both problems and in addition, it can be used to enhance the simulation realism and maintain a consistent behavior between the fluid simulation and the surface reconstruction.

When computing the internal interactions within the fluid, a common and critical step is the computation of the neighborhood of a particle that defines which particles influence



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

Figure 4.6: *Illustration of the blending asymmetry of merge (a-b) and split events (c-d) with two particles.*



(a) (b)

Figure 4.7: *Euclidean neighborhoods do not handle the fluid topology, thus introducing incorrect particles behavior. When disconnected components, in blue and red, come close by (a), the Euclidean neighborhood of the green particle includes red particles. Even when a thin wall must separate the fluid (b)-in blue, the components still interact through it.*

the evolution of the fluid's physical properties it carries. Although only the particles within a locally common fluid component physically interact, current neighborhood computations do not consider the fluid topology and they simply return the set of particles located at a distance lower than a certain threshold. In general, the computation of fluid interactions requires this distance to include at least three rings of neighbors around a particle. On the one hand, this Euclidean neighborhood is computationally very efficient and representative for the fluid inner parts. On the other hand, in the vicinity of the fluid boundaries, particles belonging to disconnected fluid components can be neighbors and for all of them, the physics is solved as if they are all within the same contiguous part of the fluid.

This approximation introduces inaccuracies in the simulation. For example, when two crossing disjoint fluid parts moving in opposite directions influence each other because they are close enough at a particular time step to be considered in the same neighborhood as illustrated in Figure 4.7(a). This introduces rotations on the fluid simulation and possibly attraction and fusions between the components while they should just cross without any interaction. The situation is the same when a thin wall is meant to separate the fluid into two distinct components: Euclidean neighborhoods make the components interact through the wall (see Figure 4.7(b)). When such inaccurate behaviors happen at different locations, repetitively over time during a simulation, their impact on the result becomes more and more prominent.

At each time step, once the simulation is computed, the fluid surface has to be reconstructed in order to be visualized. In this reconstruction, a neighborhood computation is also required to blend the contributions of the nearby particles, and thus produce a smooth reconstruction. When using Euclidean neighborhoods, all particles are automatically blended together to form the reconstructed surface. This produces an inaccurate final surface, which is not exactly representative of the simulated fluid.

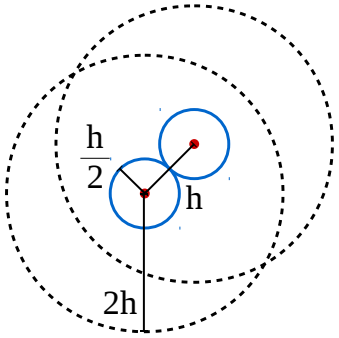
To our knowledge, the only work tackling the topology issue proposes to group the particles globally by connected components and to blend only the particles within the same component for rendering [YT13]. In this method, particles are considered in the same component if they are close enough. Even though it provides a partial solution for improving the fluid

rendering, this global approach does not allow the detection of disjoint, but close parts of the same component. Neither is the method able to detect the local fusion and separation of particles in an accurate fashion nor produce the asymmetric behavior of fluid fusion and separation. Indeed, two fluid components are to be fused only when they collide, while surface tension forces maintain two fluid components connected during separation up to a split when the fluid junction becomes too thin.

4.2 Contribution And Overview

In our work, inspired by the blending graphs introduced to control the compositions in soft object modelling [OM93,DG95], we propose to manage the fluid topology using a graph. The fluid topology is represented at the level of each particle by its list of neighbors within its local fluid component. The main contributions are then the temporally coherent neighborhood updates during the simulation with a detection and treatment of the particles fusion and separation together with a dedicated surface reconstruction. We take into account the asymmetric behavior of particles fusion and separation, while maintaining the coherence between the physical simulation and the reconstructed surface. Thereby, we avoid the introduction of the inconsistent particle behaviors caused by the use of Euclidean neighborhoods, and we bring a solution to an extremely complex problem in the context of particle-based fluid simulation.

Our neighborhood computation can be used in most particle-based fluid simulation and without loss of generality, it is exposed and illustrated in an SPH simulation. Each particle of index i is associated with a position $\mathbf{p}_i$, a density ρ_i varying over time, and a constant mass m_i. For simplifying the exposition, we assume that all particles have the same radius of influence $2h$ and their radius is $h/2$ as illustrated in the side Figure. At each time step, the density and position of each particle i are updated from the set N_i of particles j within a given distance $2h$ from i, that is $N_i = \{j \mid \|\mathbf{p}_j - \mathbf{p}_i\| < 2h\}$.



As motivated above, this set of Euclidean neighboring particles might improperly include particles belonging to a different component of the simulated fluid, which is problematic for both the simulation and the surface reconstruction. Our main objective in this work is therefore to filter these neighborhoods such that only the particles that are locally part of the same fluid component interact. As illustrated in Figure 4.8, we intuitively define this notion of local components by considering the pairs of particles that are directly connected by a "piece" of fluid. These so-called *topological* neighborhoods $G_i \subseteq N_i$ are computed and stored for each particle i and maintained throughout the simulation such that both the simulation and the reconstruction remain consistent.

The two main steps of our algorithm are depicted in Figure 4.8. The left figures illustrate the surface reconstruction with blending at a distance obtained in the case of a standard

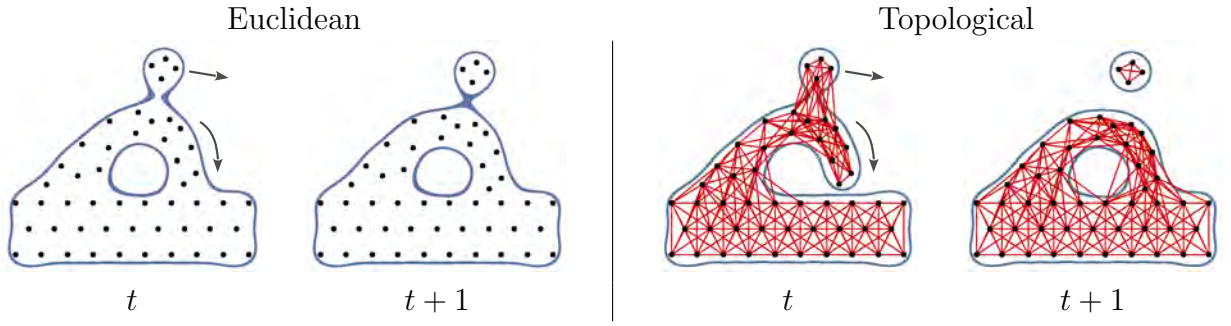


Figure 4.8: *Overview of our approach at two successive time steps. Particles are shown in black, and the reconstructed fluid surface is in blue. Left: standard Euclidean neighborhoods and reconstruction by the sum lead to unwanted blend at a distance. Right: simulation and reconstruction using our topological neighborhoods (in red) produce a surface with a more consistent topology.*

simulation, and the right figures show both the surface and the graph with topology control produced with our approach. Given the set of n particles indexed by $i \in [1, n]$ (black dots) and the set $\{G_i\}$ of their topological neighborhoods at time t (red lines in Figure 4.8 right), we start by reconstructing a fluid surface presenting an adequate topology (in blue in Figure 4.8 middle right and Section 4.1.1).

Then, the particles at the next time step $t + 1$ are updated by restricting the simulation interactions in the fluid to the current topological neighborhoods $\{G_i\}$. This integration of our neighborhoods in the underlying SPH simulation is discussed in Section 4.5.1. From these new positions (Figure 4.8 far right), the neighborhoods (red lines) are updated by detecting merges and splits according to the surface reconstruction as detailed in Section 4.4. This provides the particles with their new topological neighborhoods that are used for the next surface reconstruction at time $t + 1$ (in blue in Figure 4.8 far right). The neighborhood management is however computationally expensive and we show how computations can be efficiently accelerated and reduced in Section 4.5.

4.3 Topology-Aware Surface Reconstruction

Our reconstruction method is based on two steps. We first compute local reconstructions from the topological neighborhoods of particles, thus corresponding to the local fluid components particles lie in. We then assemble these local reconstructions together through a global composition keeping connected components together and preventing blending at a distance to maintain topology consistency.

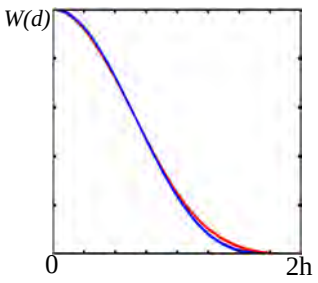
4.3.1 Local Reconstruction

We now propose a surface reconstruction respecting these topological neighborhoods. For the following we assume that each particle i knows its topological neighborhood G_i.

Our proposition is based on the summation of field functions f_i associated with each particle i, for which we chose the following polynomial P:

$$P(d) = \max\left(0, \left(1 - (\tfrac{d}{2h})^2\right)^5\right) \tag{4.10}$$

which is a very close approximation (in blue in the side Figure) of the standard SPH kernel of Equation 4.4(in red), having the advantage of being faster to evaluate.



The particle density ρ_i is computed in a standard way in SPH by integrating the mass m_i over the topological neighborhood G_i:

$$\rho_i = m_i + \sum_{j \in G_i} m_j \, W(\mathbf{p}_i - \mathbf{p}_j) \tag{4.11}$$

Whereas the masses, the kernel and its radius match the physical simulation, we emphasize that these densities are computed for reconstruction purpose only. They usually do not coincide with the densities computed in the physical simulation, as discussed in Section 4.5.1.

At this stage, our goal is to avoid the reconstruction artefacts produced by the sum of all f_i, i.e. the blending at a distance that generates surface attraction and unwanted connections as illustrated in Figure 4.9. We thus want a field function $\phi : \mathbb{R}^3 \mapsto \mathbb{R}$ reproducing in each point $\mathbf{x} \in \mathbb{R}^3$ the sum of the functions f_i of the particles that are in the same local fluid component only, while not blending those in disconnected components.

The blend between particles is generated by the summation of their respective field contributions. Thus, the blending size is controlled by the radial slope and radius of the field functions defining the particle contributions to the fluid volume. For instance, the higher the field values and the larger the radius, the larger the blending size. Some approaches try to adapt these slope and influence by particle [BS95, BGC98, WW00, HL03a] so that the blending size between a particle and its neighborhood can be locally adjusted. However, this prevents particles to blend differently with their neighbors according to their local fluid component and represent to much parameter tuning, say one per particle.

sum of particles contributions expected result

Figure 4.9: *Blending the particles with a sum leads to either unwanted blending at a distance (left red arrow) or bulge (right red arrow) when fluid components should not interact.*

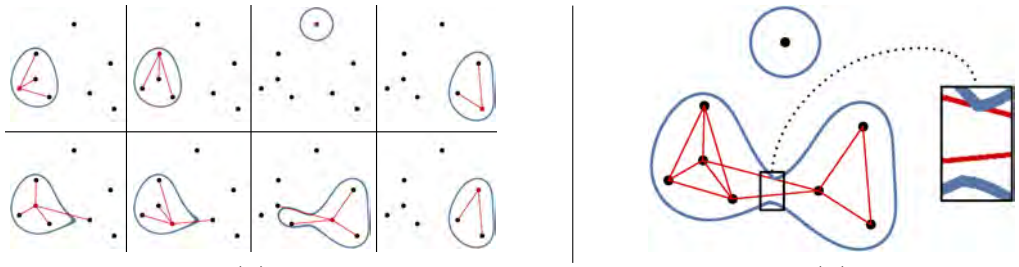
In order to overcome these limitations, blending graphs can be used [OM93, GW95]. In these structures particles are sorted by component such that all particles within the same component blend together while those in different components collide and generate a contact surface. In these graphs, components can be connected by duplicating, in each component, the particles by which they are linked.

This is the direction we are following. For each particle i we first define the field function g_i used to reconstruct its local component as follows:

$$g_i(\mathbf{x}) = f_i(\mathbf{x}) + \sum_{j \in G_i} f_j(\mathbf{x}) \tag{4.12}$$

Figure 4.10(a) illustrates the set of such blended particles from the ones depicted in Figure 4.9. Particles i are in red and the neighbor particles in G_i are linked with a red edge. By construction, each field function g_i respects the neighborhood G_i and taking the union of these functions, for instance using $\max_i g_i$ [Sab68, Ric73], yields a reconstruction with an adequate topology as shown in Figure 4.10(b). Note that some particles remain outside the reconstructed surface due to the multiplication by the ratio between the particle mass and density in Equation 4.3 or to graph update heuristics used in Section 4.4. This Figure also illustrates that this topologically coherent reconstruction only produces C^0 continuous surfaces. This is due to the union of the functions g_i that generates sharp edges where they intersect, which is undesired for the reconstruction of a fluid surface.

From there, we need to find a way to automatically combine the local topological reconstructions g_i so that the resulting surface presents an adequate topology and does not exhibit sharp features, i.e. is continuous enough, nor bulge where the local reconstructions intersect. However, using the initial graph blending [GW95] between the g_i will boil down to a simple union. In the following Section we present approaches we tried by adapting state of the art operators. Our final topology-aware surface reconstruction solution is presented in Section 4.3.3.



(a) (b)

Figure 4.10: *Illustration of our topological neighborhoods in 2D with union composition. Particles are shown with black dots, and the reconstructed surface is in blue. In our approach each particle stores its list of topological neighbors (a - red connections) from which local reconstructions are defined (a - blue surfaces), one per particle. Taking their union (b) yields the expected topology, but the surface is only C^0 thus exhibiting sharp edges (close-up).*

4.3.2 First Attempts For Combining Local Fluid Components

Our first attempt was to take advantage of the work by de Groot et al. [dGWvdW09] (see Chapter 2), that extends the idea of blending range manipulation in order to allow the same particle to blend differently with different particles around it via anisotropic field contraction. This approach is effective in the case of a low number of neighbors, when the local field deformation only influences the desired particles and is then not practical for particle-based simulations where a single particle has often a very large number of neighbors. The idea of this first attempt was to apply this blending range manipulation on the g_i instead of the particles, then reducing the perturbations it introduces within the global reconstruction. However, the local field modifications do not only affect the desired neighbors but also a set of nearby particles for which a different blending behavior is expected, making this approach ineffective. Indeed, since particles contribute not only to their own local reconstruction but also to their neighbors', then the local reconstructions largely overlap, thus introducing a large bulging effect, unless the blending range is totally contracted around the surface, then once again leading to the union of the g_i.

The second attempt we made was to use the recent gradient-based operators [GBC+13] to benefit from the automatic blend control. Here again, instead of composing particles we compose their local reconstructions g_i in two steps. First, when the g_i share particles we want a slight blend near their intersection, i.e. where the angle between their gradients is large enough, and a union elsewhere, then producing an expected result close to the one of the "camel" controller of Section 2.2.5. We hence build global components by iteratively blending such g_i. Second, these global components are combined using a union to keep the global topology. This approach is however ineffective since such global components do not handle self-proximity, as the breaking wave of Figure 4.8, and then still present blending at a distance. Also, the use of binary operators makes the reconstruction unstable since the resulting surface depends on pairwise compositions and is not suitable for the composition of the largely overlapping local reconstructions g_i (≈ 100 contributing neighbors).

Our last unsuccessful attempt was inspired by the Moving Least Squares method [SOS04] and the use of gradient based composition to control the amount of blending between the local reconstructions g_i. We start by projecting the evaluation point $\mathbf{p}$ onto the surface of the local reconstruction g_i to get the projected point $\tilde{\mathbf{p}}_i$. This allows to build a nearly Euclidean distance field $\|\mathbf{p} - \tilde{\mathbf{p}}_i\|$ between $\mathbf{p}$ and g_i. The projected point $\tilde{\mathbf{p}}_i$ is iteratively computed by semi-orthogonal gradient projection following ∇g_i, as detailed in Algorithm 1 and illustrated in Figure 4.11. The gradient descent is controlled by the surface isovalue $\mathcal{C}$, the value of the local reconstruction g_i and the norm of is gradient ∇g_i at the current position $\mathbf{x}$. We ensure convergence by setting a maximal step of half the gradient norm. We thus get the intermediary point $\mathbf{x}'$. In order to have an as orthogonal as possible projection, we then project the evaluation point $\mathbf{p}$ on the plane defined by the intermediary point $\mathbf{x}'$ and the gradient $\nabla g_i(\mathbf{x}')$. We thus get a second intermediary point $\mathbf{x}''$. Finally, the new iteration point is given as the barycenter of $\mathbf{x}$ and $\mathbf{x}''$ to avoid divergence in convex zones.

Algorithm 1 Computation of $\tilde{\mathbf{p}}_i$ = projection of $\mathbf{p}$ on g_i

$\mathbf{x} \leftarrow \mathbf{p}$
while $\|\nabla g_i(\mathbf{x})\| > \epsilon$ and $|g_i(\mathbf{x}) - C| > \epsilon$ **do**
  // Gradient descent
  $\mathbf{x}' \leftarrow \mathbf{x} - \frac{\nabla g_i(\mathbf{x})}{\|\nabla g_i(\mathbf{x})\|}.(g_i(\mathbf{x}) - C).min(\frac{g_i(\mathbf{x})}{\|\nabla g_i(\mathbf{x})\|}, \frac{1}{2})$
  // Semi-orthogonal projection
  $\mathbf{x}'' \leftarrow \mathbf{p} + [(\mathbf{x}' - \mathbf{p})^T \frac{\nabla g_i(\mathbf{x}')}{\|\nabla g_i(\mathbf{x}')\|}].\frac{\nabla g_i(\mathbf{x}')}{\|\nabla g_i(\mathbf{x}')\|}$
  $\mathbf{x} \leftarrow \frac{\mathbf{x}' + \mathbf{x}''}{2}$
end while
$\tilde{\mathbf{p}}_i \leftarrow \mathbf{x}$

From there, the final surface is defined as an isosurface of the following distance field:

$$\phi(\mathbf{p}) = \frac{\sum_{i=0}^{N} \|\mathbf{p} - \tilde{\mathbf{p}}_i\| \; \omega_i(\mathbf{p}) \; \theta_i(\mathbf{p})}{\sum_{i=0}^{N} \omega_i(\mathbf{p}) \; \theta_i(\mathbf{p})} \tag{4.13}$$

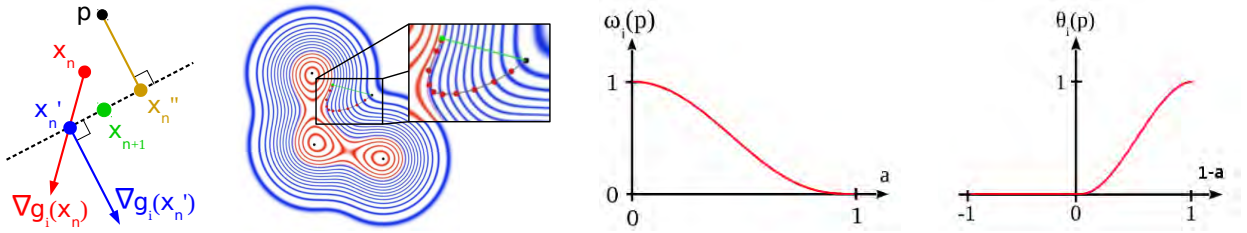
where ω_i is a distance based weighting function:

$$\omega_i(\mathbf{p}) = 2a^3 - 3a^2 + 1, a = \frac{\mathbf{p} - \tilde{\mathbf{p}}_i}{h_i} \tag{4.14}$$

and θ_i is a gradient based weighting function:

$$\theta_i(\mathbf{p}) = 2b^3 - 3b^2 + 1, b = 1 - \frac{\nabla g_i(\tilde{\mathbf{p}}_i)}{\|\nabla g_i(\tilde{\mathbf{p}}_i)\|}^T . \frac{\nabla g_k(\tilde{\mathbf{p}}_k)}{\|\nabla g_i(\tilde{\mathbf{p}}_k)\|} \tag{4.15}$$

where $k = argmin_j \|\mathbf{p} - \tilde{\mathbf{p}}_j\|$, allowing for control of the blending based on the gradient of the sharp union. Desired shapes for functions ω_i and θ_i are also illustrated in Figure 4.11.



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

Figure 4.11: *Projection of the evaluation point* $\mathbf{p}$ *on a local reconstruction surface. Particles and the projected point are drawn in black, intermediary points are in red, and the final projected point in green. (a) One iteration process, (b) full iteration process. (c) and (d): general behavior of weighting functions* ω_i*, for distance weight, and* θ_i*, for gradient weight, used to control the final surface reconstruction.*

This representation of the fluid surface is however quite complex, very slow to compute due to the multiple projections, $\approx 500s$ for $20K$ particles and a 200^3 evaluation grid, and does not provide a smooth enough surface.

4.3.3 Our Composition Model For Combining Local Fluid Components

We thus came back to traditional n-ary operators for implicit surfaces and take advantage of the following observation. By construction of our neighborhoods (see Section 4.4), when a particle j is in G_i then i is in G_j, meaning that the volumes described by the functions g_i and g_j largely overlap each other (≈ 100 neighbors). We thus need to slightly blend the functions g_i (Figure 4.10), just enough to avoid sharp edges but without generating bulge nor blending at a distance (Figure 4.9). This is done by weighting Ricci's blending operator [Ric73] to define a new graph-based composition operator as follows:

$$\phi(\mathbf{x}) = \left(\sum_i \frac{g_i(\mathbf{x})^s}{|G_i| + 1} \right)^{\frac{1}{s}}, \tag{4.16}$$

where s is the parameter for controlling the amount of blending as explained in Section 2.2.1. The normalization factor $|G_i| + 1$ compensates the multiple occurrences of the same particle field function, say f_i, in the different g_j, thus avoiding the introduction of bulges. The multiple occurrences of field functions f_i come from the overlapping of the different neighborhoods G_j (Figure 4.10). We found that taking $s = 20$ provides a good tradeoff for maintaining the expected topology while smoothing the edges as illustrated in Figure 4.12(b). The final fluid surface is then reconstructed as an isosurface defined as the set $\{\mathbf{x} \in \mathbb{R}^3 \mid \phi(\mathbf{x}) = \mathcal{C}\}$, where $\mathcal{C}$ is taken such that the radius of an isolated particle is $h/2$, i.e. $\mathcal{C} = W(h/2)$. This formulation is much simpler than the previous attempts and the final surface (Figure 4.12(b)) is close to the result of the union (Figure 4.12(a)), but with higher continuity.

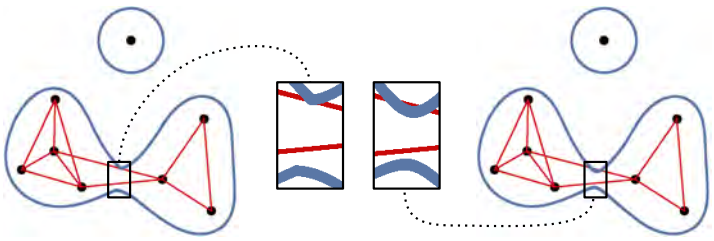


Figure 4.12: *Comparison of surface reconstructions in 2D using a union composition (a) and our new operator (b). Our new operator allows small enough blending between local reconstructions to smooth out sharp features produced by the union while preventing bulge and blend at a distance.*

4.4 Topological Neighborhoods

Given the surface reconstruction described in the previous section, we explain how the set of topological neighborhoods $\{G_i^t\}$ at a time step t is updated from the previous simulation step $t-1$. This mainly consists in detecting fusion and separation in the fluid surface in order to maintain an adequate topology. Note that our topological neighborhoods can be paired with any other reconstruction method that handles the topology issues mentioned at the beginning of Section 4.1.1. Throughout these updates, we require that the neighbor relation is symmetric: $j \in G_i \Leftrightarrow i \in G_j$. We also assume that the topological neighborhoods are consistent with respect to the fluid components. This means that particles within the kernel support of each other and that are parts of the same local fluid component must remain topological neighbors even though a non-merge or split event has been detected (see Sections 4.4.1 and 4.4.2). This property boils down to the following local transitive-closure of the neighbor relation:

$$\begin{aligned} &\forall i,j \text{ s.t. } i \neq j \text{ and } \|\mathbf{p}_i - \mathbf{p}_j\| < 2h, \\ &\text{if } \exists k \in G_i^t \cap G_j^t \text{ s.t. } \max(\|\mathbf{p}_i - \mathbf{p}_k\|, \|\mathbf{p}_j - \mathbf{p}_k\|) \leq \alpha h \\ &\text{then } i \in G_j^t \text{ and } j \in G_i^t \ . \end{aligned} \tag{4.17}$$

In contrast to classical transitive-closure, our local variant restricts the transitivity condition in two ways. Firstly, it applies only to the pairs of particles that are less than $2h$ apart (first line in Equation 4.17). Secondly, it can be inferred only from existing pairs of particles that are close enough to each other. This proximity is controlled by the parameter α in the second line of Equation 4.17. We control this proximity because in SPH simulation and reconstruction, the kernel support of the field functions is usually very large, it approximately matches a three-ring neighborhood. Without this restriction, a transitive-closure would connect distant particles that would not be connected by the surface reconstruction, which is inconsistent. A typical example is the one presented in Figure 4.8 for which connections would be created across the fluid handle. Our experiments showed that taking $\alpha = 5/4$ is an effective choice.

Since any neighborhood change modifies the reconstructed surface, satisfying all the aforementioned constraints might lead to a chicken-egg problem. Our solution to avoid this involves the following three steps which are summarized in algorithm 2.

Firstly: the particle positions are updated through the SPH simulation routine using the neighborhoods $\{G_i^{t-1}\}$ (Algorithm 2 line 1), where only topological neighbors interact. For the initial time step, these neighborhoods are initialized with all particles within the radius of influence $2h$ (i.e. $G_i^0 = N_i^0$). As the particles have moved, their surface reconstruction densities have to be updated using Equation 4.11 (Algorithm 2 line 2).

Secondly: particle fusions are detected among the pairs of particles whose supports intersect and which are not already neighbors in $\{G_i^{t-1}\}$ (Algorithm 2 line 3). As detailed in Section 4.4.1, this step yields intermediate neighborhoods $\{G_i'\}$. Before going any further,

Algorithm 2 Simulation step

1: $\{\mathbf{p}_i^t\} \leftarrow$ sph_update($\{\mathbf{p}_i^{t-1}\}$, $\{\rho_i^{t-1}\}$, $\{G_i^{t-1}\}$)
2: $\{\rho_i^t\} \leftarrow$ reconstruction_density_update($\{\mathbf{p}_i^t\}$, $\{G_i^{t-1}\}$)

Merging stage:

3: $\{G'_i\} \leftarrow$ merge_update($\{\mathbf{p}_i^t\}$, $\{\rho_i^t\}$, $\{G_i^{t-1}\}$)
4: local_transitive_closure($\{G'_i\}$)
5: $\{\rho_i^t\} \leftarrow$ reconstruction_density_update($\{\mathbf{p}_i^t\}$, $\{G'_i\}$)

Splitting stage:

6: $\{G_i^t\} \leftarrow$ split_update($\{\mathbf{p}_i^t\}$, $\{\rho_i^t\}$, $\{G'_i\}$)
7: local_transitive_closure($\{G_i^t\}$)
8: $\{\rho_i^t\} \leftarrow$ reconstruction_density_update($\{\mathbf{p}_i^t\}$, $\{G_i^t\}$)
9: surface_reconstruction($\{\mathbf{p}_i^t\}$, $\{\rho_i^t\}$,$\{G_i^t\}$)

these neighborhoods have to be completed to satisfy the local transitive-closure property (Algorithm 2 line 4). Also, surface reconstruction densities have to be recomputed to take into account the novel connections (Algorithm 2 line 5).

Thirdly: pairs of particles which are connected in $\{G'_i\}$ but that appear to be locally disconnected with respect to the fluid surface are removed (Algorithm 2 line 6) as detailed in Section 4.4.2. Again, the local transitive-closure property has to be ensured (Algorithm 2 line 7) to obtain the final updated neighborhoods $\{G_i^t\}$. Those are used to update the densities one more time (Algorithm 2 line 8) before performing the surface reconstruction (Algorithm 2 line 9).

4.4.1 Component Fusion

For each particle i, we compute an intermediate neighborhood G'_i as the union of the topological neighborhood G_i^t and the set of particles j within its kernel support for which their respective local fluid components collide or interpenetrate the one of i. We thus consider each pair of particles i-j such that $j \notin G_i^{t-1}$, $i \notin G_j^{t-1}$ and $\|\mathbf{p}_i - \mathbf{p}_j\| \leq 2h$. Since our reconstructed surface is very close to the union of the blended neighborhoods (i.e. $\phi \approx \max_i g_i$), we can assume that each of the two blended neighborhoods g_i and g_j well represent the local fluid components around the particles i and j respectively. Our problem is then to detect whether these two pieces of fluid intersect.

We detect the fusion by searching along a 1D parametric line connecting the two particles. Let r_{ij} be the signed distance between $\mathbf{p}_i$ and the fluid surface defined by g_i in the direction of $\mathbf{p}_j$, that is, r_{ij} is the largest real value such that $g_i\left(\mathbf{p}_i + r_{ij}\frac{(\mathbf{p}_j-\mathbf{p}_i)}{\|(\mathbf{p}_j-\mathbf{p}_i)\|}\right) = \mathcal{C}$. By defining r_{ji} analogously, our fusion condition becomes:

$$\|\mathbf{p}_i - \mathbf{p}_j\| < \beta(r_{ij} + r_{ji})\,,$$

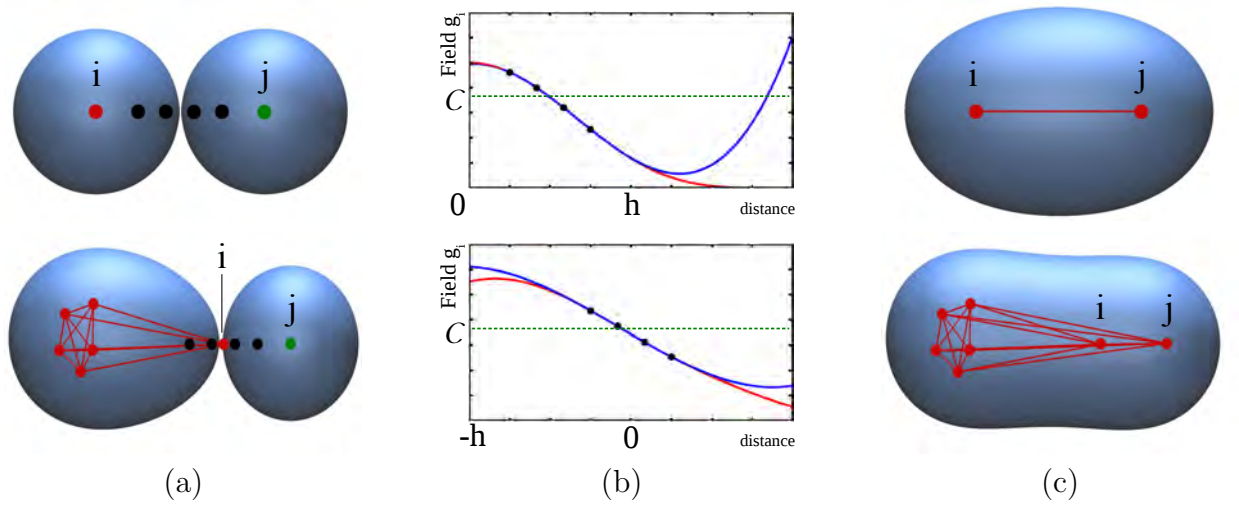


Figure 4.13: *Illustration of the merge detection mechanism between two primitives i and j formed by the red and green particles respectively. Two cases are depicted here for the approximation of r_{ij}. Top row: the position of the isosurface is found using the first tested range. Bottom row: for an uneven sampling, a second range around the particle i has to be tested. (a) The current particles and reconstruction. (b) The plot of the field function g_i (red curve) of particle i, and its local approximation by a cubic polynomial (blue curve) passing through the four sampled positions (black dots). (c) Novel connections and reconstruction after merging.*

where β is a small tolerance factor compensating for the small blending produced by Equation 4.16 and favoring early over late fusions. We always use $\beta = 1.01$.

Since each field function evaluation has a computational cost, we estimate the values r_{ij} with cubic polynomial approximations. As depicted in Figure 4.13 top row, we first consider the interval $[h/4, 3h/4]$ along the given parametric line, and construct the cubic polynomial that interpolates four sample values of g_i uniformly taken within this range. In most cases, this strategy succeeds in providing the expected result. In some rare cases, as in Figure 4.13 bottom row, an uneven particle sampling might significantly shrink the surface. The current particle might even lie outside its own local fluid component. In this case, the first sampled value is below the isovalue (i.e. $g_i(h/4) < \mathcal{C}$), and the search interval is shifted to $[-h/4, h/4]$. The same fitting procedure is then applied. Finally, if the last sampled value is within the fluid component (i.e. $g_i(3h/4) > \mathcal{C}$) then the particle cannot be at the boundary of its component, and no merge is explicitly detected for this pair. A connection might eventually be established later through transitive-closure as explained in Section 4.4.3.

We emphasize that this detection of fusion does not depend on the look-up order of the particles as all primitive evaluations are carried out according to the fixed neighborhoods $\{G_i^{t-1}\}$, whereas newly detected neighbors produce $\{G_i'\}$. Implementation-wise, the use of four sample values to fit the cubic polynomials enables to fully exploit the SIMD vector instruction sets of current CPUs: these four evaluations are carried out at the cost of a single evaluation.

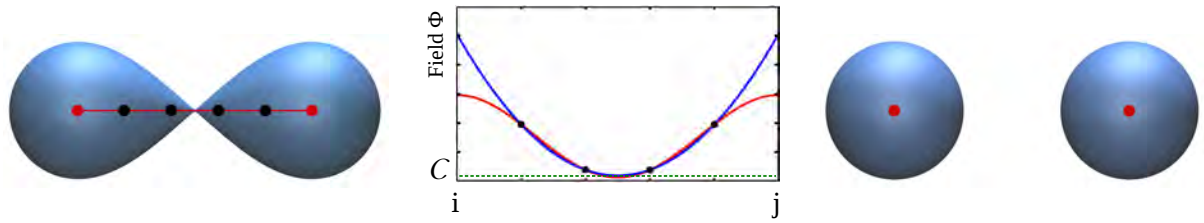
4.4.2 Component Splitting

Component separation or split occurs when particles of the same fluid component move apart from each other. Each pair i-j of neighbor particles (i.e. $j \in G'_i$ and $i \in G'_j$) is checked in case splitting is required once all fusions have been performed. Two neighbor particles i and j are split only if the segment $[\mathbf{p}_i, \mathbf{p}_j]$ joining them has a part lying outside the local fluid component defined by the union of their respective blended neighborhoods g_i and g_j (see Figure 4.14).

More formally, let $\bar{g}_{ij}$ be the minimum of $\max(g_i, g_j)$ along the segment $[\mathbf{p}_i, \mathbf{p}_j]$. If $\bar{g}_{ij} < \mathcal{C}$, then the pair i-j is split. Otherwise, the particle i (resp. j) is inserted into G^t_j (resp. G^t_i). In practice, we quickly estimate $\bar{g}$ by fitting a univariate quadratic polynomial to a given number of sampled values of $\max(g_i, g_j)$ uniformly taken on the segment $[\mathbf{p}_i, \mathbf{p}_j]$, as illustrated in Figure 4.14. As for detecting fusions, we found that taking four samples is accurate enough in practice while enabling fast SIMD evaluations.

This procedure requires every pair of connected particles to be tested, which is very expensive as the number of such pairs is two orders of magnitude larger than the number of particles. The number of splitting tests can be drastically reduced by observing that if two connected particles are close enough to each other, then a separation is very unlikely to occur. Each connected pair i-j such that $\|\mathbf{p}_i - \mathbf{p}_j\| < \alpha h$ are thus preserved and ignored by the splitting test, where $\alpha = 5/4$ as for the local transitive-closure in Equation 4.17 since it plays the same role.

We can now take advantage of our local transitive-closure property to further reduce the number of splitting tests. Indeed, given a connected pair of particles i-j that can be potentially split, if there exists a third particle k satisfying the local transitive-closure property of Equation 4.17, then we know that the pairs i-k, and j-k will not be split, and thus the pair i-j will be set back by the transitive-closure property. The pair i-j has then to be preserved bypassing the splitting test.



(a) (b) (c)

Figure 4.14: *Illustration of the split detection mechanism between two connected particles i and j. (a) Current configuration and reconstruction. (b) The plot of the union of the two field functions g_i, g_j (red curve) along the segment i-j, and its approximation by a quadratic polynomial (blue curve) fitted on four sample points (black dots). (c) Since the minimum is below $\mathcal{C}$, the pair is split, resulting in disconnected particles.*

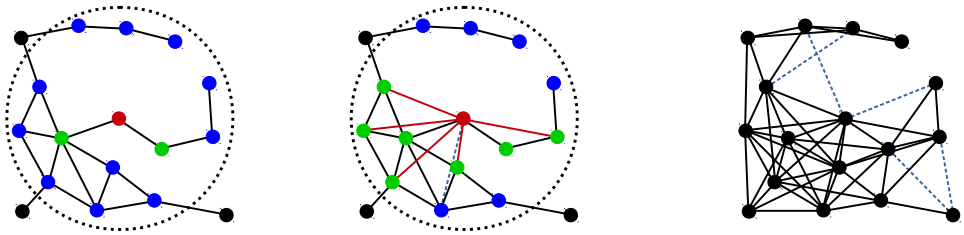
Finally, the condition for a pair i-j, $i \in G'_j$ and $j \in G'_i$, to be inserted into G^t_i and G^t_j can be summarized as follows:

$$\begin{array}{ll} & \|\mathbf{p}_i - \mathbf{p}_j\| < \alpha h \\ \text{or} & \exists k \in G'_i \cap G'_j \text{ s.t. } \max(\|\mathbf{p}_i - \mathbf{p}_k\|, \|\mathbf{p}_j - \mathbf{p}_k\|) \leq \alpha h \\ \text{or} & \bar{g}_{ij} \geq \mathcal{C} \end{array}$$

4.4.3 Transitive-Closure

As explained earlier, the local transitive-closure property (Eq. 4.17) of our topological neighborhoods has to be satisfied before their use for density estimation or local surface reconstruction. This explains why passes of transitive-closure update have to be performed both after detecting merge and split events (lines 4 and 7 of Algorithm 2). Transitive-closure is usually computed through repeated depth-first or breath-first traversals. However, in our context the number of pairs that can be added through local transitive-closure is considerably smaller than the number of existing pairs. Therefore, we found that a much faster strategy consists of looping over each pair of potentially miss-connected particles, i.e. each pair i-j such that $j \notin G^{t-1}_i$ while $\|\mathbf{p}_i - \mathbf{p}_j\| \leq 2h$, and search for a common and close enough neighbor particle. This step has to be repeated until convergence is achieved, that is until no novel connection is established. This procedure is illustrated in Figure 4.15. Notice that thanks to our double locality restriction, the cavity is well preserved.

During the merging stage we loop over the set of potentially missing pairs using the following method. Each pair for which no merge has been detected is appended to a list. Then, during transitive-closure updates, it is enough to loop over this list from which a pair



(a) (b) (c)

Figure 4.15: *Local transitive-closure. (a) A given particle in red with its current neighbors in green, its kernel support (large circle), and the set of particles that can be potentially connected to it in blue. (b) The first sweep of transitive-closure update for this given particle yields the addition of the five connections in red. The dashed blue line indicates a potential connection that has not been already established because their respective shared particle was too far away from them. This one will be established during the next sweep. (c) Resulting connections after applying this step to all particles repeatedly until convergence is achieved. Again, the dashed blue lines indicate a few connections that are not introduced (on purpose), because the edges that could infer them are above our threshold length αh.*

is removed if and only if it is added to the set of neighborhoods. For the second pass of transitive-closure (after the splitting step), it is enough to consider only the list of pairs that have been split.

4.4.4 Temporal Coherence

During a fusion near a particle i, new particles are inserted into its topological neighborhood. These new particles are usually close to i meaning that they immediately exhibit a significant contribution to both the density ρ_i and the blended neighborhood g_i of the given particle i. As a result, popping might occur in the reconstructed surface, as show in Figures 4.13 and 4.16. In the second Figure, a falling droplet gets immediately absorbed when contact is detected. We address this issue by tracking the "age" a_{ij} of each neighbor relation. For each newly connected particle i-j, a_{ij} is initialized to zero and updated at each frame as follows:

$$a_{ij}^t = \min\left(1, a_{ij}^{t-1} + \Delta t/\gamma\right) \tag{4.18}$$

where Δt is the time in seconds between two frames, and γ is the duration in which the age of a relation saturates to 1. For the first frame, the age is initialized to one for all pairs (i.e. $a_{ij}^0 = 1$). This age is then used in the computation of both densities and primitives as follows:

$$\rho_i = m_i + \sum_{j \in G_i} a_{ij} m_j W(\mathbf{p}_i - \mathbf{p}_j) \tag{4.19}$$

and

$$g_i(\mathbf{x}) = f_i(\mathbf{x}) + \sum_{j \in G_i} \left(1\text{-}(1\text{-}a_{ij})^3\right) f_j(\mathbf{x}) \tag{4.20}$$

The difference in weighting enables a better balance between the temporal variations of the density versus blending. The behavior produced by such a temporal weighting scheme is depicted in Figure 4.16.

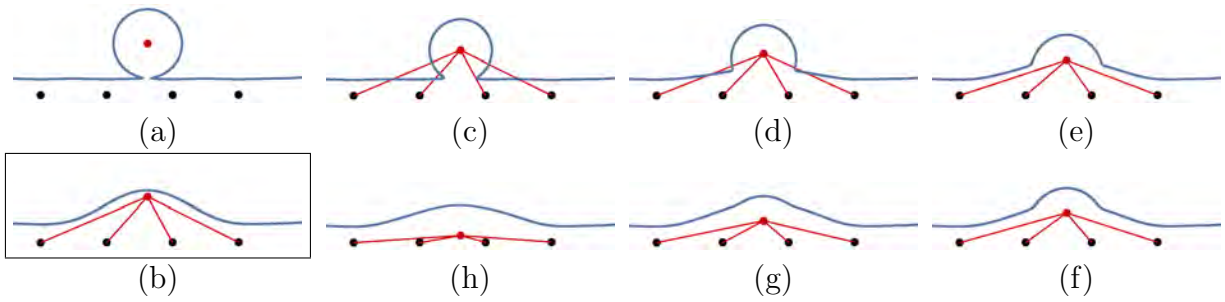


Figure 4.16: *Illustration of our temporal coherence mechanism on a falling particle (in red) coming into contact with another fluid component (a). (b) The newly established connections lead to a quick change in the reconstructed isosurface. The sequence (c) to (h) shows the progressive absorption produced by our temporal weighting.*

4.5 Practical Implementation

4.5.1 Integration in a Particle-Based Simulation

Integrating our approach within an existing simulation code only requires to replace loops over Euclidean neighbors by loops over our topological ones. We implemented our prototype using DualSPHysics [CDR+15] for the particle simulation, for which we enabled the Shepard density filter for adjusting densities in the vicinity of the fluid surface. From the physical aspect, some precautions must be taken however.

The standard SPH integration kernel assumes that the ambient space is full of particles whereas in general only fluid particles are simulated, resulting in a bias in the density estimation at the proximity of the fluid surface. On the other hand, removing particles from a Euclidean neighborhood in these areas, even though they belong to a separate fluid component, also leads to physically incorrect density computations. This situation changes as soon as such under-sampled particle neighborhoods are numerically compensated, using for instance adjusted integration kernels [BK02]. In that case, the use of our topological neighborhood allows to compute densities and forces consistently and independently of the presence of a nearby disconnected fluid component. Thus, inadequate fluid interferences coming from disconnected components are avoided. We also point out that the use of our topological neighborhood would naturally handle very thin walls between parts of a fluid since particles in each side would belong to their own topological fluid component. This avoids particles interactions through the wall and only particle-to-wall interactions remain to be simulated. We do not show such an example case because the DualSPHysics fluid simulation enforces the use of large enough walls to avoid the unexpected fluids interactions, thus preventing particles on one side of the wall to come close enough to penetrate the influence radius of particles on the other side.

4.5.2 Topological Neighborhood Implementation

Neighborhood updates (steps 2 to 8 of algorithm 2) are implemented on the CPU. All these steps but the transitive-closure passes are accelerated taking advantage of multi-threading with OpenMP. In order to avoid memory reallocation during neighborhood updates, each particle stores the list of all neighbors within the range $2h$, say $\mathcal{N}^t$. These lists are updated once per frame using a 3D grid after the particle positions have been updated. Each list is kept sorted with respect to indices and such that topological neighbors appear first. Flags are used to distinguish between the different stages of the update (i.e. G^{t-1}, G' or G^t). Sorted lists enable fast searches and set intersections during transitive closure updates.

4.5.3 Efficient Surface Evaluation

Our reconstructed isosurface is extracted as a mesh from a uniform grid which is filled by evaluating ϕ through a CUDA implementation. Each evaluation of $\phi(\mathbf{x})$ (Eq. 4.16) involves a pair of nested loops on each nearest primitive and each particle of the current primitive.

These loops are required to find all particles i such that $g_i(\mathbf{x})$ is non null, i.e. we have to find all particles within a sphere centered at $\mathbf{x}$ and of radius $4h$.

In practice, these search and evaluation can be greatly accelerated by reducing this radius. Indeed, due to the use of exponentiation with a large number, that is g_i^{20} in Equation 4.16, the contribution of a given field function g_i quickly becomes negligible when moving away, and only the nearest ones have a real impact on the result. Therefore, thanks to the very large overlap between the field functions g_i, we found that it is always sufficient to consider only the particles within a radius of $3.25h$ for a gain of about $\times 1.5$. Since the isosurface is expected to stand at a distance $h/2$ of the particles, this is a rather conservative choice, and the search radius can be aggressively shrunk without impacting the reconstruction.

A second optimization consists in stopping the sum over the primitives as soon as it exceeds $\mathcal{C}^s$, meaning that the evaluation point $\mathbf{x}$ is within the fluid. As shown in Table 4.1, this *early stop* optimization significantly reduces the grid filling cost, especially where the fluid covers a large volume.

In addition, our computation of the global field ϕ is only required in the vicinity of adjusted neighborhoods. Everywhere else, where $\{G_i\} = \{\mathcal{N}_i\}$, it is enough to apply the standard reconstruction, i.e. summing the fields f_i of the Euclidean neighbors of $\mathbf{x}$. Finally, the number of overall evaluations can be greatly reduced by evaluating the field function at the proximity of surface particles, as explained by Akinci et al. [AIAT12].

When implementing the grid filling on a GPU, additional care must be taken to maximize parallelism among the threads of the same warp, i.e. among the packet of typically 32 threads

Scene		**Table**		**Splash**
#particles		3.8k	22.8k	56k
SPH simulation		0.11	1.47	0.3
$\{G_i\}$ **update**		0.02	0.17	0.49
GPU Eval	*Grid resolution*	462×264×216		216^2×334
	no optimization	*0.68*	*0.83*	*66.33*
	+ early stop	*0.68*	*0.84*	*14.09*
	+locality of adjusted neighborhoods	*0.40*	*0.70*	*1.29*
	Final eval time (*+CUDA warps*)	0.33	0.54	0.99
	Standard sum of the f_i	*0.15*	*0.23*	*0.28*

Table 4.1: *Average timings in seconds for the update of one frame for the two scenes shown in Figure 4.17. Reported timings include the SPH simulation using the DualSPHysics library, the update of our topological neighborhoods $\{G_i\}$, both on the CPU, and the filling of the full marching cube grid on the GPU using either our reconstruction method or a standard sum of the f_i. The timings for our reconstruction method are reported with different level of optimization, starting from the naive version evaluating the full field function ϕ everywhere, then successively adding early-stop, the restriction to area containing adjusted neighborhoods, and finally the CUDA warp coherence.*

that follow the same execution flow. Indeed, because of the nested loops, threads attached to nearby grid points can quickly diverge. We enforce a coherent evaluation by attaching blocks of $4 \times 4 \times 2$ grid points to the same warp that performs a common traversal of the grid to query the primitives within the union of individual queries. However, branch divergence still exists because some primitives which have to be processed by at least one thread might have to be skipped for the others. Further acceleration is thus obtained by skipping the farthest primitives in a coherent manner using the following pseudo-code algorithm where $\{i_1, \ldots, i_m\}$ denotes the set of primitive indices processed by the current warp:

$k = 1$
while $k \leq m$ **do**
  while $k \leq m$ **and** $\|\mathbf{x} - \mathbf{p}_{i_k}\| > 3.25h$ **do** $k = k + 1$;
  if $k \leq m$ **then** accumulate the contribution of g_{i_k} ;
  $k = k + 1$

This algorithm has the effect of re-synchronizing the threads by making them wait until all threads have a primitive to work on. This yields an additional $\times 1.3$ speed-up factor for large grid resolution.

Thus, thanks to our various optimization mechanisms, the overhead induced by the reconstruction of the fluid surface with our approach compared to a standard sum over the Euclidean neighbors ranges from a factor $\times 2$ to $\times 3$.

4.6 Results and Limitations

We have evaluated our approach on two test scenes (see Figure 4.17) by comparing our results with those obtained by using Euclidean neighborhoods for the SPH simulation, and the sum of the field functions f_i for the surface reconstruction (denoted as standard).

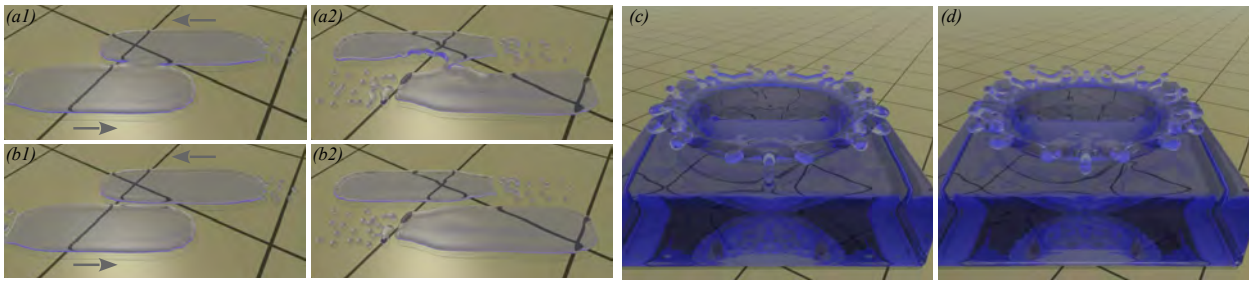


Figure 4.17: *Two SPH fluid simulations using a standard Euclidean particle neighborhood (a,c), and our new topological neighborhood (b,d). On the left, two fluid components are crossing while moving in opposite directions. Our topological neighborhoods perform accurate merge detection and avoid both unwanted fusions in the reconstruction and incorrect fluid interactions in the simulation. On the right, our topologically accurate neighborhoods lead to different shape of the splash, and enable the reconstruction of the fluid with an adequate topology while avoiding bulging at a distance.*

4.6.1 Quality

The first scene, called "Table" (Figure 4.17-left), consists of two pieces of water labelled A and B, moving in opposite directions along a flat table, and passing nearby each other. The particles of each component have been initialized such that the two respective fluids do not intersect if simulating them separately. For evaluation purpose, we added an isolated copy of the second component labelled C, which is initialized with the exact same conditions. Figure 4.18 shows short sequences for two different simulation resolutions. When using 1260 particles per component, Euclidean neighborhoods quickly merge the two nearby components in both the simulation and reconstruction, creating small splashes. As a result, the components become significantly distorted. On the contrary, our topological neighborhoods properly solve the reconstruction ambiguity and thus prevent the interactions between the two disconnected components within the simulation. As a result, the component B remains identical to its isolated duplicate C. The effect of distant interactions of Euclidean neighborhoods can be diminished by increasing the number of particles sampling the fluid. Nonetheless, as shown in Figure 4.18-right, even after increasing the number of particles by a factor 6, some unwanted fusions and distortions are still present.

In the second test scene (Figure 4.17-right), a droplet composed of about 180 particles hits a box of still water producing a splash. As can be seen in Figure 4.19, the two simulations exhibit differences. Figure 4.21 shows the effect of our temporal merging mechanism: as expected, it can be seen that the falling droplet gets smoothly absorbed by the larger fluid component. In order to ease the evaluation of our reconstruction method in Figure 4.20, we thus compare it to reconstruction results obtained using a standard sum of the f_i over the Euclidean neighbors, but using the same particle simulation as in our method. Unwanted bulging and merging at a distance can be observed throughout the sequence produced by the Euclidean neighborhoods, whereas our approach successfully tracks the expected topology of the fluid until components get very close to each other.

4.6.2 Performance

We have measured the performance of our prototype implementation on a computer equipped with a 3.4GHz Intel Core-i7 processor and an NVIDIA Geforce GTX 580 GPU. Table 4.1 reports average costs of the different steps of our algorithms for three different simulations. This table also details the impact of the aforementioned optimizations when filling a grid on the GPU for our simulations. It can be seen that the relative overhead to update our topological neighborhood highly depends on the simulation. Indeed, for the "Table" scene, the overhead of our approach is marginal because this scene requires several intermediate simulation steps to avoid numerical instabilities, as automatically determined by DualSPHysics. On the other hand, DualSPHysics can simulate the "Splash" scene at a much higher rate, even though it contains more particles. For this scene, the running time of our neighborhood update is of the same order as the SPH simulation itself.

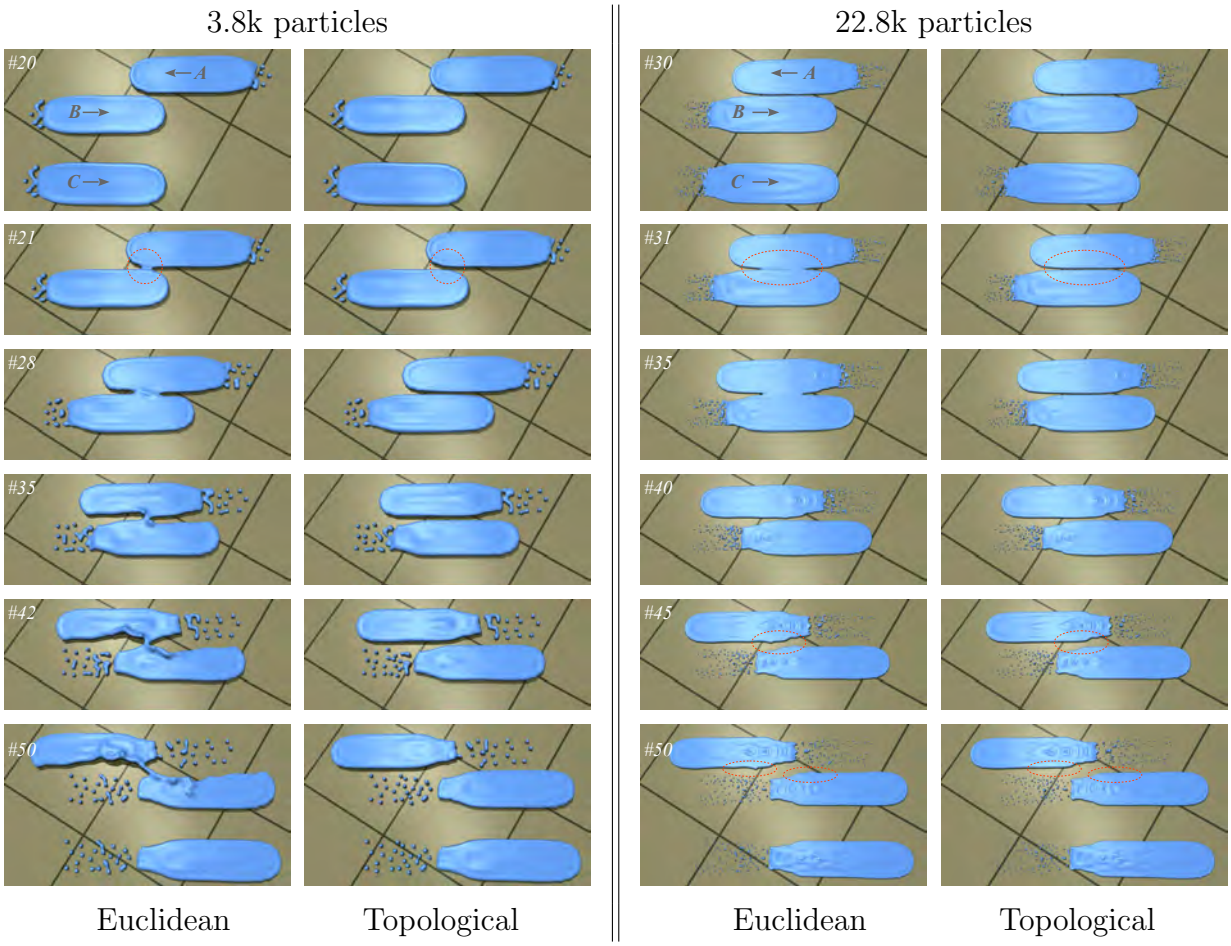


Figure 4.18: *Short sequences of the "Table" simulations using either Euclidean or our topological neighborhoods. The white labels indicate the respective frame number. The two fluid components A and B incorrectly interact and merge when using Euclidean neighborhoods, while they naturally cross without interacting when using our approach (component B remains identical to its isolated copy C).*

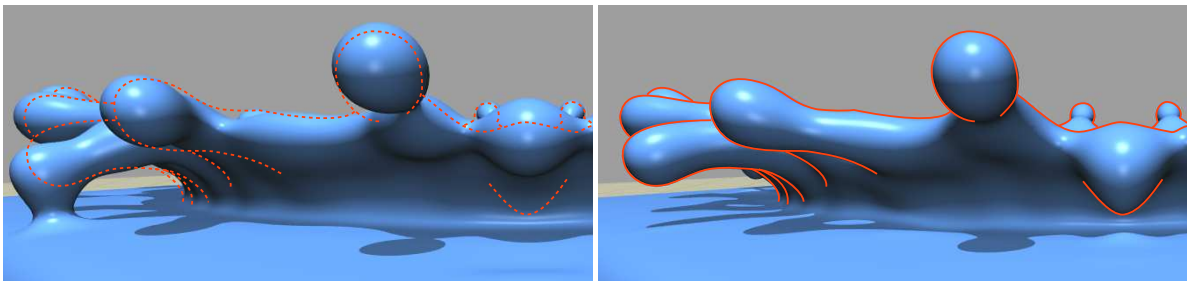


Figure 4.19: *Illustration of the diverging behavior between a standard simulation (left), and our approach (right). To highlight the simulation differences, both simulations have been reconstructed using Euclidean neighborhoods and the silhouette of the right image is reported to the left one.*

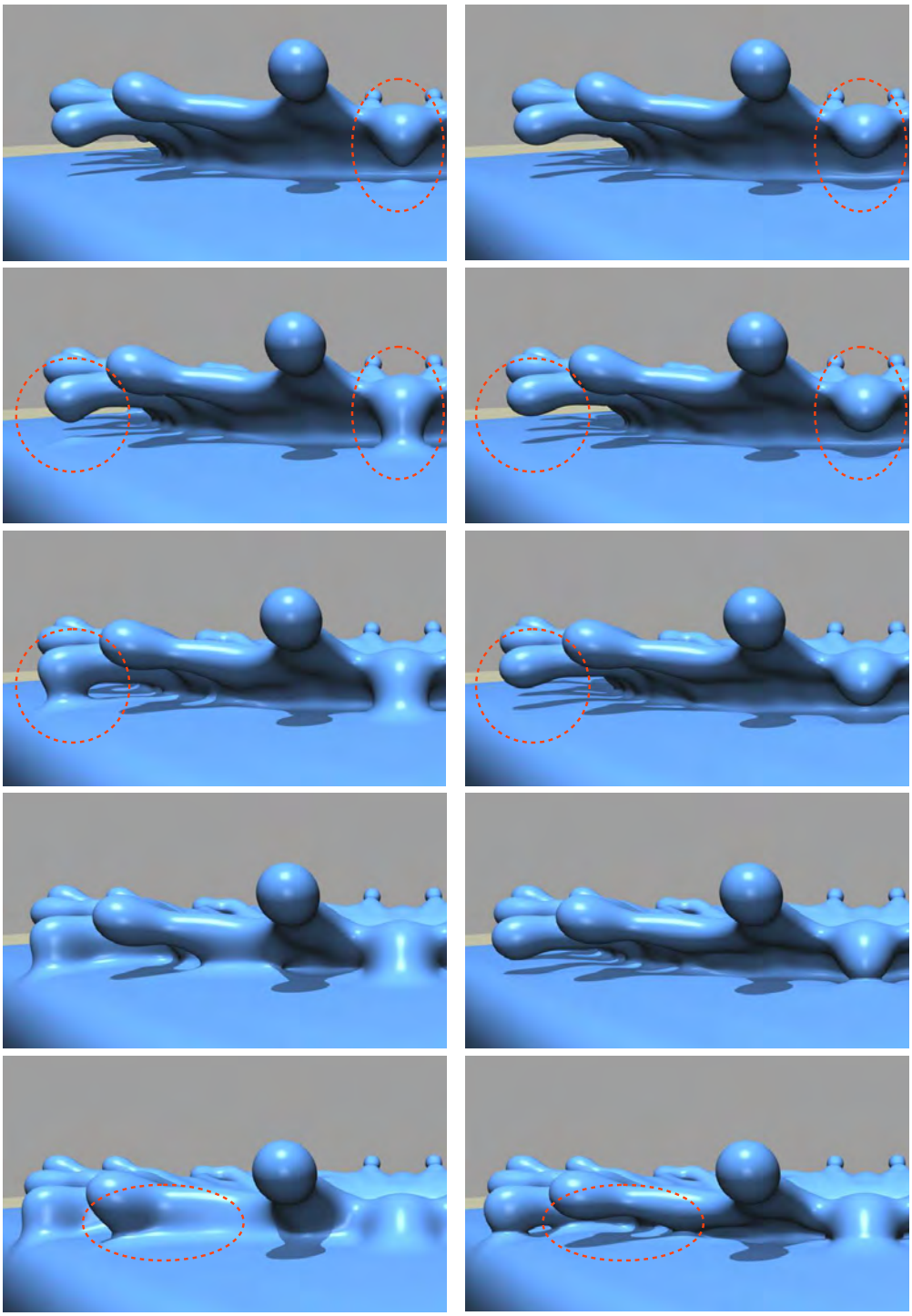


Figure 4.20: *Five consecutive frames of the "Splash" simulation using our neighborhoods. This figure compares the reconstruction with a standard Euclidean sum of the field functions f_i (left), to our reconstruction method (right). Notice how the water surface is deformed and merged prior to the actual contact event when using Euclidean neighborhoods.*

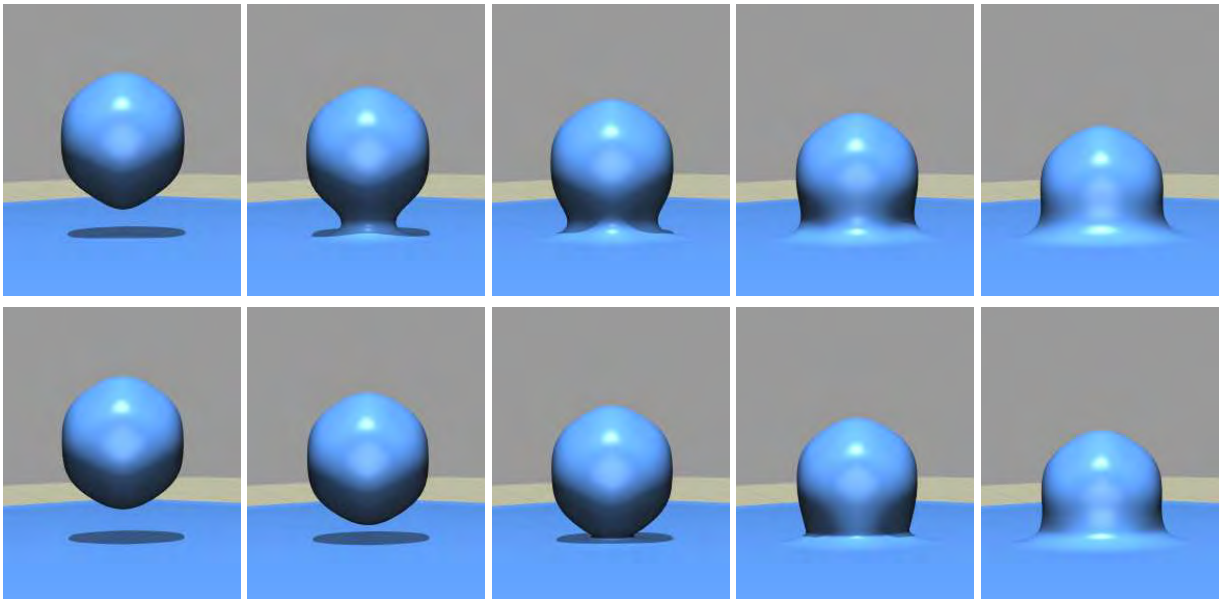


Figure 4.21: *Comparison of the standard approach (top row), with our approach (bottom row) on the "Splash" simulation. Notice how our approach successfully solves the contact event while enabling a smooth transition through temporal weights. The falling droplet is composed of about* 180 *particles.*

4.6.3 Limitations

The detection of fusion and separation relies on some heuristics, meaning that they can be detected slightly too early or too late. Nonetheless, the effects of these approximations are seldom perceptible and they are considerably less prominent than in standard approaches.

In this work we focused on the preservation of the fluid topology, although we have neglected the fairness of the final fluid surface. Although mesh based smoothing techniques can be used, it would be interesting to investigate the extension of our method to take advantage of recent advances in the reconstruction of a fluid surface with better tension properties from SPH simulations.

For instance, incorporating anisotropic primitives [YT10] within our method is straightforward. Another approach would be to define the individual clusters g_i using any implicit reconstruction method, for instance Solenthaler et al.'s smooth distance field [SSP07], convert it to a compactly supported field function (see Section 3.1), and combine them using the original Ricci's blending operator [Ric73].

4.7 Conclusion

In this chapter, we presented a new n-ary composition operator, capable of representing asymmetric blending between primitives and supporting a large number of them. This new

n-ary operator relies on a composition graph to cluster the primitives by local contribution components that can largely overlap or may not be blend.

For the case of particle-based fluid simulation, our new operator is able to represent the fluid surface with adequate control of both blend at a distance and surface topology. Here, our graph-based neighborhoods track the surface topology events such as merges and splits so that our composition operator produces a final surface with the adequate topology, thus preventing incorrect bulges and blends. Also, our new neighborhood computation can benefit to the fluid simulation by preventing incorrect particles behavior when disconnected components come close one to another but do not intersect. Our results show that our neighborhoods solve some incorrect behaviors in the simulation and can lead to significantly different particles motion. Finally, our neighborhoods can be trivially integrated into an existing particle-based fluid simulation system as it simply replaces the classical Euclidean neighborhoods.

As future works, we would like to continue to diminish the computational overhead brought by the computation of our neighborhoods, for instance by exploring a GPU implementation of their update, and alternative reconstruction strategies exhibiting a lower algorithmic complexity. We would also like to investigate how to integrate fluid properties such as viscosity and surface tension efficiently into our method and enhance the coupling between topological neighborhoods, simulation and reconstruction.

Chapter 5
Conclusion

Durant cette thèse, nous nous sommes intéressés à une représentation d'objets 3D particulière pour l'informatique graphique : les surfaces implicites. Cette représentation volumique est basée sur deux concepts mathématiques que sont les fonctions de champ scalaire et les opérateurs de composition. Afin d'expérimenter et valider nos travaux de recherche sur le contrôle de la forme de la surface résultant d'une opération de composition, nous nous sommes penchés sur l'utilisation des surfaces implicites dans deux domaines d'application : la modélisation géométrique et la simulation de fluides par particules.

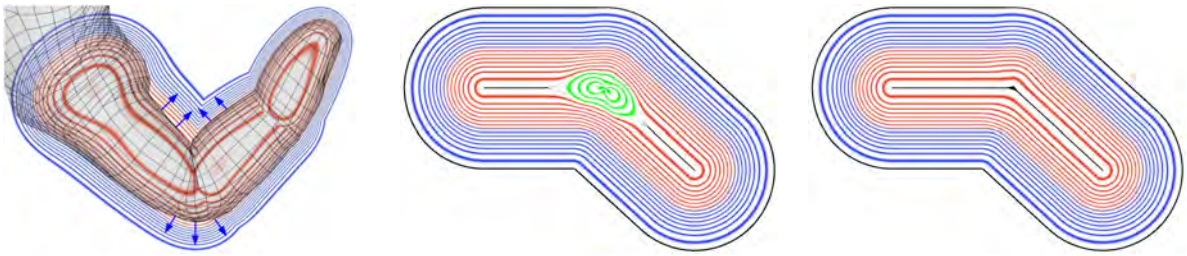
Pour ce qui est de la modélisation géométrique par surfaces implicites, nous avons étudié un modèle de surface implicite et son couplage avec des opérateurs de composition binaires. Ces opérateurs sont utilisés dans un système de modélisation incrémental dans lequel des éléments de matière sont assemblés, au travers d'opérations de type booléennes, afin de créer un objet complexe par ajout ou suppression de matière. Ainsi, nous présentons dans nos travaux de nouvelles contraintes que les fonctions de champ scalaire et les opérateurs de composition doivent respecter afin de rendre le modèle de composition unifié et consistant. Leur mise en œuvre s'est traduite en un nouveau modèle de fonctions de champ scalaire conjointement à un nouvel ensemble d'opérateurs de composition associés et basés sur la littérature afin d'offrir un contrôle fin du comportement de la composition. De plus, l'utilisation de ces deux contributions permet de restreindre la zone de variation du champ scalaire généré par la surface implicite résultante, notamment pour venir en creuser la délimitation lors d'une opération de différence afin que celle-ci ressemble plus à la surface. Nous avons conçu nos nouveaux opérateurs en utilisant la représentation 2D utilisée pour les opérateurs binaire. Cette représentation établit un parallèle visuel direct entre l'opérateur et son application. Elle permet une meilleure compréhension de ce qu'il se passe et est très utile pour la conception des opérateurs. Nous en avons d'ailleurs profité pour concevoir un nouvel opérateur spécifiquement adapté à la modélisation des détails fins sur la surface. Cet opérateur permet l'absorption progressive du champ scalaire généré par les détails, supprimant ainsi les dépressions indésirables dans le champ scalaire de l'objet final afin que celui-ci ressemble plus à un champ de distance à la surface.

En simulation de fluides par particules, nous avons plutôt investigué les opérateurs n-aires pour leur capacité à combiner un grand nombre de particules. Nous nous sommes ici surtout concentrés sur la cohérence temporelle de la topologie de la surface du fluide qui est reconstruite. Nous avons mis au point un nouvel opérateur n-aire basé sur un graphe de composition et permettant de réaliser un mélange asymétrique des particules au regard des fusions et séparations de parties de fluide. Ainsi, les artefacts de gonflements et de mélange à distance lorsque des parties de fluide passent à proximité les unes des autres sont évités et la surface reconstruite respecte la topologie du fluide simulé. La cohérence

temporelle est obtenue par l'utilisation de voisinages topologiques pour les particules à la fois dans la reconstruction de la surface pour éviter l'apparition de mélanges indésirables, et dans la simulation pour le calcul des interactions entre particules pour éviter notamment les interférences entre parties de fluide disjointes. Nous montrons aussi comment les voisinages topologiques, que nous utilisons comme graphe de composition, sont mis à jour au cours de la simulation pour prendre en compte les nombreuses fusions et séparations afin de gérer le comportement asymétrique du fluide. Enfin, pour garantir un temps de calcul raisonnable nous proposons plusieurs optimisations pour la reconstruction de la surface du fluide nous permettant de n'être que trois fois plus lents que la reconstruction standard par simple somme des contributions des particules.

Nous pensons que les travaux présentés dans cette thèse sont importants et offrent des contributions pertinentes quant à l'utilisation des surfaces implicites et leurs modèles de composition en informatique graphique. Tout d'abord, nous devons reconnaître qu'aujourd'hui les surfaces implicites ne constituent pas une représentation pratique pour concevoir des objets 3D, comparées aux surfaces de subdivision et surfaces paramétriques qui offrent des outils plus intuitifs pour la génération et surtout l'édition de la surface. Cependant, l'utilisation d'un système de modélisation par surfaces implicites en tant qu'application expérimentale nous a permis de valider à la fois nos modèles de fonctions de champ scalaire et d'opérateurs de composition et de rendre le processus de composition unifié et consistant.

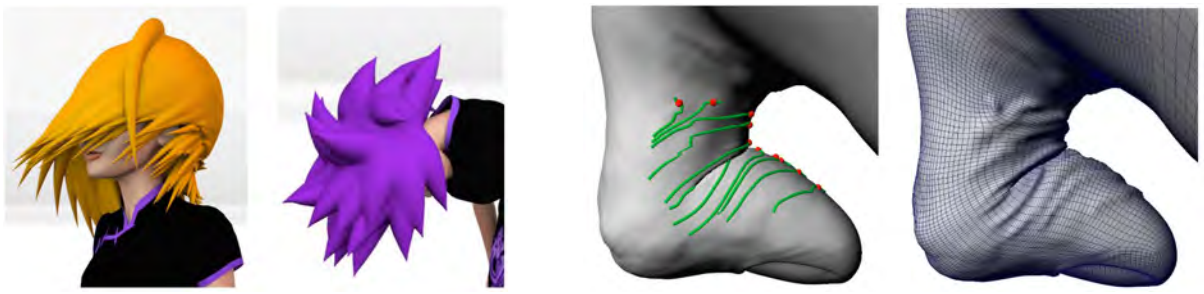
Ensuite, nos travaux démontrent l'importance de la qualité et des propriétés du champ scalaire aussi bien à l'extérieur qu'à l'intérieur de l'objet, qui sont indispensables pour beaucoup d'applications des surfaces implicites comme le contrôle de forme. Un très bon exemple est le couplage des surfaces implicites et des maillages pour l'animation de personnages [VBG+13, VGB+14], où les surfaces implicites décrivent le volume de parties du personnage représenté par un maillage. Durant la mise en mouvement, les sommets du maillage suivent la surface implicite à laquelle ils sont rattachés en utilisant le gradient du champ scalaire généré. Ainsi, notre représentation consistante permet d'améliorer ce suivi par la production d'un champ scalaire sans dépression et avec un gradient plus lisse, comme illustré en Figure 5.1.



(a) (b) (c)

Figure 5.1: *L'Implicit Skinning [VBG+13] utilise le gradient du champ scalaire pour projeter les sommets du maillage sur la surface (a). Nos opérateurs (c) suppriment les dépressions et discontinuités de gradient générées par la composition (b), rendant la projection plus stable.*

De plus, le modèle de composition pour le gonflement au contact utilisé ici peut être étendu à la modélisation des muscles grâce à notre opérateur d'absorption des détails. Les surfaces implicites peuvent aussi être utilisées pour représenter des chevelures [ZCG14] ou des surfaces de forme libre comme les plis pour l'animation de personnages ou la simulation de vêtements [TFR15, RPC+10] comme illustré Figure 5.2.



(a) (b)

Figure 5.2: *Les surfaces implicites peuvent être utilisées pour modéliser des chevelures de type manga (a) ou des plis sur la peau de personnages animés (b). Images issues de [ZCG14, TFR15].*

Enfin, en étendant l'utilisation d'un graphe de composition, nous avons conçu un nouvel opérateur n-aire capable de passer à l'échelle et permettant un contrôle de la topologie par le biais d'un mélange asymétrique, ce qui faisait défaut aux opérateurs de la littérature. Nous sommes persuadés que notre utilisation de voisinages topologiques est un pas important pour aller plus loin dans la simulation de fluides par particules. Par exemple, le couplage entre nos voisinages topologiques et la simulation peut amener à un meilleur contrôle des conditions aux bords, que ce soit pour la frontière du fluide ou la gestion de murs fins, mais aussi à un mouvement du fluide plus naturel concernant les croisements, fusions et séparations ou encore à une meilleure gestion des propriétés du fluide comme la vélocité, la viscosité ou la tension de surface.

Bibliography

[AA04] Marc Alexa and Anders Adamson. On normals and projection operators for surfaces defined by point sets. In Symposium on Point Based Graphics, Zurich, Switzerland, June 2-4, 2004. Proceedings, pages 149–155, 2004.

[AC02] Alexis Angelidis and Marie-Paule Cani. Adaptive implicit modeling using subdivision curves and surfaces as skeletons. In Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications, SMA '02, pages 45–52, New York, NY, USA, June 2002. ACM.

[AIAT12] G. Akinci, M. Ihmsen, N. Akinci, and M. Teschner. Parallel surface reconstruction for particle-based fluids. Computer Graphics Forum, 2012.

[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 and Applications (SMI'02), pages 137–144. IEEE Computer Society Press, May 2002.

[APKG07] Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J. Guibas. Adaptively sampled particle fluids. ACM Transactions on Graphics, 26(3), July 2007.

[BBCW10] Adrien Bernhardt, Loïc Barthe, Marie-Paule Cani, and Brian Wyvill. Implicit blending revisited. Proceedings of Eurographics, Computer Graphics Forum, 29(2):367–376, 2010.

[BDS+03] Loïc 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.

[BGB11] Haimasree Bhatacharya, Yue Gao, and Adam Bargteil. A level-set method for skinning animated particle data. In Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '11, pages 17–24, New York, NY, USA, 2011. ACM.

[BGB15] Haimasree Bhattacharya, Yue Gao, and Adam W. Bargteil. A level-set method for skinning animated particle data. IEEE Transactions on Visualization and Computer Graphics, 21(3):315–327, March 2015.

[BGC98] Loïc Barthe, Véronique Gaildrat, and R. Caubet. Combining implicit surfaces with soft blending in a CSG tree. In Proceedings of CSG Conference Series, pages 17–31, 1998.

[BGC01] Loïc Barthe, V. Gaildrat, and R. Caubet. Extrusion of 1D implicit profiles: Theory and first application. International Journal of Shape Modeling, 7:179–199, 2001.

[BHK14] Pierre Bénard, Aaron Hertzmann, and Michael Kass. Computing smooth surface contours with accurate topology. ACM Transactions on Graphics, 33(2), 2014.

[BK02] J. Bonet and S. Kulasegaram. A simplified approach to enhance the performance of smooth particle hydrodynamics methods. Applied Mathematics and Computation, 126(2-3):133–155, March 2002.

[Bli82] James F. Blinn. A generalization of algebraic surface drawing. ACM Transactions on Graphics, 1(3):235–256, July 1982.

[Blo97] J. Bloomenthal. Bulge elimination in convolution surfaces. 16(1):31–41, 1997.

[BMO+14] Jan Bender, Matthias Müller, Miguel A. Otaduy, Matthias Teschner, and Miles Macklin. A survey on position-based simulation methods in computer graphics. Computer Graphics Forum, 33(6):228–251, 2014.

[BS91] Jules Bloomenthal and Ken Shoemake. Convolution surfaces. In SIGGRAPH Comput. Graph., volume 25, pages 251–256. ACM, July 1991.

[BS95] Carole Blanc and Christophe Schlick. Extended field functions for soft objects. In Proceedings of Implicit Surfaces 1995, pages 21–32, 1995.

[BTT09] Markus Becker, Hendrik Tessendorf, and Matthias Teschner. Direct forcing for lagrangian rigid-fluid coupling. IEEE Transactions on Visualization an Computer Graphics, 15(3):493–503, 2009.

[Buh01] Martin Buhmann. A new class of radial basis functions with compact support. Mathematics of Computation, 70(233):307–318, 2001.

[BWdG04] Loïc Barthe, Bryan Wyvill, and Erwin de Groot. Controllable binary CSG operators for "soft objects". International Journal of Shape Modeling, 10(2):135–154, 2004.

[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 Proceedings of the 28th annual conference on Computer graphics and interactive techniques, SIGGRAPH '01, pages 67–76. ACM, 2001.

[CDR+15] A.J.C. Crespo, J.M. Domínguez, B.D. Rogers, M. Gómez-Gesteira, S. Longshaw, R. Canelas, R. Vacondio, A. Barreiro, and O. García-Feal. Dualsphysics: Open-source parallel CFD solver based on smoothed particle hydrodynamics (SPH). Computer Physics Communications, 187(0):204–216, 2015.

[CGBG13] Jiazhou Chen, Gael Guennebaud, Pascal Barla, and Xavier Granier. Non-oriented mls gradient fields. In Computer Graphics Forum, volume 32, pages 98–109. Wiley Online Library, 2013.

[CGD97] Marie-Paule Cani-Gascuel and Mathieu Desbrun. Animation of deformable models using implicit surfaces. Visualization and Computer Graphics, IEEE Transactions on, 3(1):39–50, 1997.

[DG95] Mathieu Desbrun and Marie-Paule Gascuel. Animating soft substances with implicit surfaces. In Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '95, pages 287–290, New York, NY, USA, 1995. ACM.

[dGWvdW09] Erwin de Groot, Brian Wyvill, and Huub van de Wetering. Locally restricted blending of blobtrees. Computers & Graphics, 33(6):690–697, 2009.

[DKT98] Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. In Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pages 85–94, 1998.

[Gas93] Marie-Paule Gascuel. An implicit formulation for precise contact modeling between flexible solids. In Proceedings of the 20th annual conference on Computer graphics and interactive techniques, SIGGRAPH '93, pages 313–320, New York, NY, USA, 1993. ACM.

[GBC+13] Olivier Gourmel, Loic Barthe, Marie-Paule Cani, Brian Wyvill, Adrien Bernhardt, Mathias Paulin, and Herbert Grasberger. A gradient-based implicit blend. ACM Transactions on Graphics, 32(2):1–12, April 2013.

[GDW+16] Herbert Grasberger, Jean-Luc Duprat, Brian Wyvill, Paul Lalonde, and Jarek Rossignac. Efficient data-parallel tree-traversal for blobtrees. Comput. Aided Des., 70(C):171–181, January 2016.

[GG07] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. In ACM SIGGRAPH 2007 papers, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[GGG08] Gaël Guennebaud, Marcel Germann, and Markus Gross. Dynamic sampling and rendering of algebraic point set surfaces. In Computer Graphics Forum, volume 27, pages 653–662. Wiley Online Library, 2008.

[GGP+15] Jean-David Génevaux, Eric Galin, Adrien Peytavie, Eric Guérin, Cyril Briquet, François Grosbellet, and Bedrich Benes. Terrain modelling from feature primitives. In Computer Graphics Forum, volume 34, pages 198–210. Wiley Online Library, 2015.

[GM77] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. Monthly Notices of the Royal Astronomical Society, 181(3):375–389, 1977.

[GW95] Andrew Guy and Brian Wyvill. Controlled blending for implicit surfaces using a graph. In Proceedings of Implicit Surfaces 1995, pages 107–112, 1995.

[HAC03] Samuel Hornus, Alexis Angelidis, and Marie-Paule Cani. Implicit modelling using subdivision-curves. Visual Computer, 19(2-3):94–104, May 2003.

[HC12] Evelyne Hubert and Marie-Paule Cani. Convolution surfaces based on polygonal curve skeletons. Journal of Symbolic Computation, 47(6):680–699, 2012. Advances in Mathematics MechanizationMathematics Mechanization.

[HH85] Christoph M. Hoffmann and John E. Hopcroft. Automatic surface generation in computer aided design. Technical report, Ithaca, NY, USA, 1985.

[HL03a] P. C. Hsu and C. Lee. Field functions for blending range controls on soft objects. Proceedings of Eurographics, Computer Graphics Forum, 22(3):233–242, 2003.

[HL03b] P-C Hsu and C Lee. The scale method for blending operations in functionally-based constructive geometry. In Computer Graphics Forum, volume 22, pages 143–158, 2003.

[HL08] S. Holzer and O. Labs. SURFEX 0.90. Technical report, University of Mainz, University of Saarbrücken, 2008. `www.surfex.AlgebraicSurface.net`.

[IOS+14] Markus Ihmsen, Jens Orthmann, Barbara Solenthaler, Andreas Kolb, and Matthias Teschner. Sph fluids in computer graphics. In Sylvain Lefebvre and Michela Spagnuolo, editors, Eurographics 2014 - State of the Art Reports. The Eurographics Association, 2014.

[JT02] Xiaogang Jin and Chiew-Lan Tai. Analytical methods for polynomial weighted convolution surfaces with various kernels. Computers & Graphics, 26(3):437–447, 2002.

[Kra] Matjuška Teja Krašek. Submission to the SURFER competition 2015 in Ljubljana of the Imaginary Project.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. SIGGRAPH Comput. Graph., 21(4):163–169, August 1987.

[Li07] Q. Li. Smooth piecewise polynomial blending operations for implicit shapes. Computer Graphics Forum, 26(2):157–171, 2007.

[LP04] Qingde Li and Roger Phillips. Implicit curve and surface design using smooth unit step functions. In Proceedings of the ACM symposium on Solid modeling and applications, SM '04, pages 237–242. Eurographics Association, 2004.

[Luc77] L.B. Lucy. A numerical approach to the testing of the fission hypothesis. Astronomical Journal, 82:1013–1024, December 1977.

[MCG03] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '03, pages 154–159, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[MGV11] I. Macêdo, J. P. Gois, and L. Velho. Hermite radial basis functions implicits. Computer Graphics Forum, 30(1):27–42, 2011.

[MM13] Miles Macklin and Matthias Müller. Position based fluids. ACM Transactions on Graphics (TOG), 32(4):104, 2013.

[MMTD07] Patrick Mullen, Alexander McKenzie, Yiying Tong, and Mathieu Desbrun. A variational approach to eulerian geometry processing. ACM Transactions on Graphics (TOG), 26(3):66, 2007.

[Mon92] J. J. Monaghan. Smoothed particles hydrodynamics. Annual review of astronomy and astrophysics, 30:543–574, 1992.

[MP89] Gavin Miller and Andrew Pearce. Globular dynamics: A connected particle system for animating viscous fluids. Computers & Graphics, 13(3):305–309, 1989.

[MS98] Jon McCormack and Andrei Sherstyuk. Creating and rendering convolution surfaces. In Computer Graphics Forum, volume 17, pages 113–120. Wiley Online Library, 1998.

[MSKG05] Matthias Müller, Barbara Solenthaler, Richard Keiser, and Markus Gross. Particle-based fluid-fluid interaction. In Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '05, pages 237–244, New York, NY, USA, 2005. ACM.

[Mül11] Matthias Müller. Overview of traditional surface tracking methods. ACM Siggraph 2011 courses, 2011.

[NHK+85] H. Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirakawa, and K. Omura. Object modeling by distribution function and a method of image generation. 1985.

[OC97] Agata Opalach and Marie-Paule Cani. Local deformation for animation of implicit surfaces. In Wolfgang Straßer, editor, Spring Conference on Computer Graphics (SCCG), Bratislava, Slovakia, June 1997.

[OF06] Stanley Osher and Ronald Fedkiw. Level set methods and dynamic implicit surfaces, volume 153. Springer Science & Business Media, 2006.

[OM93] Agata Opalach and Steve Maddock. Implicit surfaces: Appearance, blending and consistency. In In Fourth Eurographics Workshop on Animation and Simulation, pages 233–245, 1993.

[PAB+97] Elbridge Gerry Puckett, Ann S Almgren, John B Bell, Daniel L Marcus, and William J Rider. A high-order projection method for tracking fluid interfaces in variable density incompressible flows. Journal of Computational Physics, 130(2):269–282, 1997.

[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.

[PHK11] Sylvain Paris, Samuel W Hasinoff, and Jan Kautz. Local laplacian filters: edge-aware image processing with a laplacian pyramid. ACM Trans. Graph., 30(4):68, 2011.

[PPIK02] Galina Pasko, A Pasko, M Ikeda, and T Kunii. Bounded blending operations. In Proceedings of Shape Modeling International, 2002, pages 95–103, 2002.

[PPK05] Galina I. Pasko, Alexander A. Pasko, and Tosiyasu L. Kunii. Bounded blending for Function-Based shape modeling. IEEE Computer Graphics and Applications, 25(2):36–45, 2005.

[Ric73] A. Ricci. A constructive geometry for computer graphics. Computer Journal, 16(2):157–160, May 1973.

[RO85] Alyn Rockwood and J. C. Owen. Blending surfaces in solid modeling. In Proceedings of SIAM Conference on Geometric Modelling and Robotics, 1985.

[Roc89] A. P. Rockwood. The displacement method for implicit blending surfaces in solid models. ACM Transactions on Graphics, 8(4):279–297, October 1989.

[RPC+10] Damien Rohmer, Tiberiu Popa, Marie-Paule Cani, Stefanie Hahmann, and Sheffer Alla. Animation wrinkling: Augmenting coarse cloth simulations with realistic-looking wrinkles. ACM Transactions on Graphics, 29(5):157, December 2010.

[RS08] Martin Reimers and Johan Seland. Ray casting algebraic surfaces using the frustum form. In Computer Graphics Forum, volume 27, pages 361–370. Wiley Online Library, 2008.

[Sab68] M-A Sabin. The use of potential surfaces for numerical geometry. In Technical Report VTO/MS/153, British Aerospace Corp., Weybridge, U.K., 1968.

[She99] Andrei Sherstyuk. Kernel functions in convolution surfaces: a comparative analysis. The Visual Computer, 15(4):171–182, 1999.

[SOS04] Chen Shen, James F. O'Brien, and Jonathan Richard Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. ACM Transactions on Graphics, 23(3):896–904, 2004.

[SSP07] Barbara Solenthaler, Jürg Schläfli, and Renato Pajarola. A unified particle model for fluid–solid interactions. Computer Animation and Virtual Worlds, 18(1):69–82, 2007.

[TFR15] Fabio Turchet, Oleg Fryazinov, and Marco Romeo. Extending implicit skinning with wrinkles. In Proceedings of the 12th European Conference on Visual Media Production, CVMP '15, pages 11:1–11:6, New York, NY, USA, 2015. ACM.

[VBG+13] Rodolphe Vaillant, Loïc Barthe, Gaël Guennebaud, Marie-Paule Cani, Damien Rohmer, Brian Wyvill, Olivier Gourmel, and Mathias Paulin. Implicit skinning: Real-time skin deformation with contact modeling. ACM Trans. Graph., 32(4), July 2013.

[VGB+14] Rodolphe Vaillant, Gäel Guennebaud, Loïc Barthe, Brian Wyvill, and Marie-Paule Cani. Robust isosurface tracking for interactive character skinning. ACM Trans. Graph., 33(6), November 2014.

[Wen95] Holger Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. Advances in computational Mathematics, 4(1):389–396, 1995.

[Wen04] Holger Wendland. Scattered data approximation, volume 17. Cambridge university press, 2004.

[WGG99] Brian Wyvill, Andrew Guy, and Eric Galin. Extending the CSG tree - warping, blending and boolean operations in an implicit surface modeling system. Computer Graphics Forum, 18(2):149–158, June 1999.

[Wik] Wikipedia. Constructive solid geometry.

[Wil08] Brent Warren Williams. Fluid surface reconstruction from particles. Master's thesis, The University of British columbia, Canada, 2008.

[WMW86] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structure for soft objects. The Visual Computer, 2(4):227–234, February 1986.

[Wu95] Zongmin Wu. Compactly supported positive definite radial functions. Advances in Computational Mathematics, 4(1):283–292, 1995.

[WW00] Brian Wyvill and Geoff Wyvill. Better blending of implicit objects at different scales. ACM Siggraph 2000 presentation, 2000.

[WYY13] Xiaoyue Wu, Xubo Yang, and Yang Yang. A novel projection technique with detail capture and shape correction for smoke simulation. Computer Graphics Forum, 32(2pt4):389–397, 2013.

[YHB05] Alex Yvart, Stefanie Hahmann, and Georges-Pierre Bonneau. Hierarchical triangular splines. ACM Trans. Graph., 24(4):1374–1391, October 2005.

[YT10] Jihun Yu and Greg Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. In Proceedings of the 2010 ACM SIGGRAPH /Eurographics Symposium on Computer Animation, SCA '10, pages 217–225, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[YT13] Jihun Yu and Greg Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. ACM Transactions on Graphics, 32(1):1–12, February 2013.

[Zan13] Cédric Zanni. Skeleton-based implicit modeling and applications. Thesis, Université de Grenoble, December 2013.

[ZB05] Yongning Zhu and Robert Bridson. Animating sand as a fluid. ACM Transactions on Graphics, 24(3):965–972, July 2005.

[ZBQC13] Cédric Zanni, Adrien Bernhardt, Maxime Quiblier, and Marie-Paule Cani. Scale-invariant integral surfaces. Computer Graphics Forum, 32(8):219–232, December 2013.

[ZCG14] Cédric Zanni, Marie-Paule Cani, and Michael Gleicher. N-ary implicit blends with topology control. Computers & Graphics, October 2014.

Study of the Composition Models of Field Functions in Computer Graphics

Abstract:

Field functions are a powerful mathematical tool for surface representation in computer graphics. Despite the volume information they provide, combined with the composition models accompanying them, field functions are still used in only a few number of applications due to their limitations such as slow user interactions and a difficult shape control.

In this thesis we study these composition models in order to develop and improve them and make them efficient and relevant for computer graphics. We do so through two applications.

The first one is geometric modelling, where field functions represent object compounds that are combined pairwisely in an iterative creation process to design complex objects. We propose to unify and make consistent both the field function representation and the composition model to provide a more stable and artefact-free modelling process.

The second one is fluid simulation and reconstruction based on particles. Here, field functions represent contributions of the particles sampling the fluid volume. These contributions are then combined in a row to build the fluid surface. In this application, we propose to take the topology of the reconstructed surface into account when running the fluid simulation, thus avoiding an inappropriate behavior of the particles, and then of the simulated fluid.

Keywords:

Computer Graphics, Implicit Surfaces, Composition Operators, Modelling, Fluid Simulation

Florian Canezin

Study of the Composition Models of Field Functions in Computer Graphics

Thèse de doctorat soutenue le 08 Septembre 2016 à l'Université Toulouse 3 Paul Sabatier
Directeurs de Thèse: Loïc Barthe et Gaël Guennebaud

Résumé

Les fonctions de champ scalaire sont un outil mathématique puissant pour la représentation de surfaces en informatique graphique. Malgré l'information de volume qu'elles offrent, combiné aux modèles de composition qui les accompagnent, les fonctions de champ scalaire ne sont encore utilisées que dans très peu d'applications en raison de leurs limitations, telles qu'une interaction utilisateur lente et un contrôle de la forme de la surface difficile.

Dans cette thèse, nous étudions ces modèles de composition dans le but de les développer, de les améliorer et de faire en sorte qu'ils soient efficaces et pertinents pour l'informatique graphique. Pour cela, nous nous intéressons à deux applications.

La première est la modélisation géométrique, où les fonctions de champ scalaire représentent des composants d'objets qui sont assemblés par paires dans un processus de création incrémental pour construire des objets complexes. Nous proposons une représentation unifiée des fonctions de champ scalaire et du modèle de composition afin d'obtenir un processus de modélisation plus stable et sans artefacts.

La deuxième application à laquelle nous nous intéressons est la simulation et la reconstruction de fluides basées particules. Ici, les fonctions de champ scalaire représentent les contributions des particules qui échantillonnent le volume du fluide. Ces contributions sont alors combinées d'un coup pour reconstruire la surface du fluide. Nous proposons dans ce cadre de prendre en compte la topologie de la surface reconstruite dans la simulation, évitant ainsi un comportement inapproprié des particules, et donc du fluide ainsi simulé.

Mots clés

Informatique graphique, Surfaces implicites, Opérateurs de composition, Modélisation, Simulation de fluides

Thèse en informatique réalisée à l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier),
au sain de l'Institut de Recherche en Informatique de Toulouse (IRIT),
118 route de Narbonne 31062 TOULOUSE CEDEX 9