



THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

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

Discipline ou spécialité : *Informatique*

Présentée et soutenue par **Olivier Gourmel**

Le 18 juin 2015

Mélange de surfaces en temps réel : visualisation, contrôle des déformations et application à la modélisation.

JURY

Rapporteurs : Tamy Boubekur, Professeur, Telecom ParisTech
Gilles Gesquière, Professeur, Université de Lyon 2

Examineur : Jean-Michel Dischler, Professeur, Université de Strasbourg

Encadrants : Mathias Paulin, Professeur, Université Toulouse III
Loïc Barthe, Maître de Conférence HDR, Université Toulouse III

Ecole doctorale : Mathématiques Informatique Télécommunications de Toulouse
Unité de recherche : Institut de Recherche en Informatique de Toulouse - UMR 5505
Directeur(s) de Thèse : Mathias Paulin

Remerciements

Cette thèse a été l'occasion pour moi de découvrir le monde fascinant de l'informatique graphique, et je voudrais remercier tous ceux qui m'ont assisté et soutenu durant cette période de ma vie.

Tout d'abord merci à mes encadrants, Mathias Paulin et Loïc Barthe, de m'avoir orienté sur ce sujet comportant de nombreux challenges techniques. Les nombreuses connaissances scientifiques qu'ils m'ont enseignées, leurs conseils avisés et la patience dont ils ont fait preuve ont été cruciaux dans l'achèvement de ma thèse. Merci en particulier à Loïc pour son soutien et pour son aide inestimable dans l'accomplissement de ma thèse : c'est en grande partie grâce à lui que j'ai pu soutenir mes travaux avec succès.

Merci également aux membres du jury pour l'intérêt qu'ils ont montré envers mes travaux de recherche. Merci à mes rapporteurs, Tamy Boubekour et Gilles Gesquière, pour leurs retours constructifs sur ce manuscrit. Merci à Jean-Michel Dischler pour ses remarques et ses commentaires sur les perspectives de mes travaux.

Merci à tous mes anciens collègues et amis de l'IRIT : Vincent Forest, Anthony Pajot, Samir Turki, Mathieu Muratet, Philippe Ercolessi, Robin Bourianes, Rodolphe Vaillant et Jean-Patrick Rocchia pour les discussions techniques pointues et les bons moments passés ensemble.

Enfin, un grand merci à Andra Doran pour son aide, notamment sur la relecture du manuscrit, pour sa gentillesse et sa bonne humeur perpétuelle qui m'ont toujours redonné le sourire, et pour son soutien sans faille au travail et en dehors.

Résumé

Les surfaces implicites ont été perçues au cours des années 80, comme une alternative intéressante aux modélisations paramétriques des surfaces (NURBS, etc). Elles sont définies comme l'ensemble des points de même valeur d'un champ potentiel, c'est-à-dire la frontière de deux volumes. Ainsi elles possèdent des propriétés avantageuses dans le cadre de la modélisation géométrique : gestion automatique de la topologie, garantie de manipuler des entités manifold, possibilité de définir des transitions lisses entre des objets se fusionnant. Elles furent cependant délaissées au début des années 2000 en raison des contraintes qu'elles imposent : évaluation et affichage coûteux en temps de calcul, et forme des surfaces difficilement contrôlables. Les contributions de cette thèse proposent des solutions à ces problématiques de la modélisation par surfaces implicites.

Il est tout d'abord montré qu'une nouvelle structure d'accélération, combinant les propriétés d'une hiérarchie de volumes englobants et d'un Kd-Tree, permet d'accélérer l'affichage par lancer de rayons d'un grand nombre de surfaces implicites. Il est ainsi possible d'animer en temps réel une surface de type fluide, définie par les points d'isovaleur d'un champ potentiel obtenu par la somme de primitives simples.

Les opérateurs simples de composition de surfaces implicites, tels que la somme, permettent d'évaluer rapidement des champs potentiels combinant plusieurs milliers de primitives. Néanmoins, l'apparence organique des surfaces produites est difficile à contrôler. Cette thèse propose un nouveau type d'opérateur de composition, utilisant à la fois les valeurs et les gradients des champs potentiels sources, qui permet d'avoir beaucoup plus de contrôle sur la forme des surfaces produites tout en supprimant les effets indésirables des opérateurs classiques, tels que le gonflement à l'intersection de surfaces ou la fusion de surfaces proches.

Enfin il est montré comment ces opérateurs de mélange peuvent être utilisés pour déformer des surfaces de type maillage, animées par un squelette. Nous définissons un champ potentiel par composition de primitives implicites générées aux arêtes du squelette. A chaque déformation du squelette, le champ potentiel est lui aussi déformé par les opérateurs de composition choisis : ces déformations peuvent être reproduites sur le maillage en déplaçant chaque sommet du maillage jusqu'à la surface d'isovaleur correspondante à leur valeur de potentiel initiale. Cette technique permet d'obtenir rapidement des déformations plausibles au niveau des articulations des membres modélisés.

Abstract

Implicit surfaces have been considered during the eighties as a promising alternative to parametric surfaces (NURBS patches, etc...). They are defined as the set of points having the same value of a scalar field, thus splitting the space into two volumes. Their volumetric nature confers them interesting properties for geometric modeling : the topology of objects is handled automatically, geometries are guaranteed to be manifold and they can produce smooth blendings of objects easily. However, they were abandoned at the beginning of the 21st century due to the limitations they impose : they are computationally expensive to evaluate and to display, and the shape of the transition between objects is difficult to control. This thesis proposes new solutions to these problems in implicit surfaces modeling.

First of all, it is shown that the use of a new object-partitioning structure, mixing the properties of a bounding volume hierarchy and a Kd-Tree, makes it possible to raytrace a large number of implicit primitives at interactive frame rates. Therefore it allows real time visualization of fluid-like shapes, defined as an isosurface of a potential field computed as the sum of simple primitives.

Simple composition operators of implicit surfaces, such as the sum operator, allow a fast computation of a potential field combining thousands of primitives. Nevertheless, the shape of the resulting surfaces is organic and difficult to control. In this thesis, a new kind of composition operators is proposed, which takes both the value and the gradient of the source potential fields as input. These operators give much more control on the shape of the surfaces, and they avoid the classical problems of implicit surfaces composition, such as bulging at the intersection of two primitives or blending of surfaces at a distance.

Finally, a new skeleton-based animation technique is presented which reproduces the deformations of some implicit surfaces on a given mesh. We define a potential field as the composition of implicit primitives generated at the bones of the skeleton. Thus each motion of the skeleton will cause distortions in the associated potential field. These distortions can be reproduced on the mesh by moving each of its vertices to the isosurface of the potential field corresponding to their initial potential value. This technique is able to produce rapidly realistic deformations on the limbs of an articulated model of a body.

Table des matières

I	Tomographie optique	5
1	Présentation	7
1.1	Formalisme	8
2	Résolution du problème inverse	11
3	Résolution du problème direct	13
3.1	Résolution par méthode des éléments finis	14
3.2	Méthode de Monte Carlo	15
3.2.1	Path tracing	17
3.3	Sensibilités	19
3.3.1	Application au path tracing	19
4	Résultats	21
II	Surfaces implicites en modélisation géométrique	25
1	Introduction	27
1.1	Motivation	27
1.2	Résumé des contributions	29
1.3	Plan	30
2	Définition et rendu des surfaces implicites	33
2.1	Introduction et définitions	33
2.2	Méthodes de rendu de surfaces implicites	36

2.2.1	Polygonisation des surfaces	37
2.2.2	Méthodes de rendu par points	38
2.2.3	Lancer de rayons	39
3	Visualisation d'un grand nombre de surfaces implicites en temps réel	45
3.1	Contributions	45
3.1.1	Test d'intersection	45
3.1.2	Adaptation de la BVH	46
3.1.3	Hiérarchie de Volumes Englobants Redimensionnée	48
3.1.4	Optimisations	51
3.2	Implémentation et Résultats	54
3.2.1	Implémentation	54
3.2.2	Résultats	55
3.3	Conclusion	58
4	Opérateurs de mélange	61
4.1	Définition	61
4.2	Problèmes majeurs de la modélisation par surfaces implicites	65
4.3	Travaux antérieurs	66
5	Mélange contrôlé de surfaces implicites	71
5.1	Contrôle du mélange par le gradient	72
5.1.1	Continuité de l'opérateur	73
5.1.2	Forme générale du contrôleur	74
5.1.3	Opérateurs de composition	77
5.1.4	Évaluation des opérateurs	79
5.2	Applications	80
5.2.1	Mélange sans gonflement	80
5.2.2	Gonflement au contact	84
5.3	Discussion	87
6	Animation de maillages par surfaces implicites	89
6.1	Méthodes existantes	91
6.2	Skinning implicite	94
6.2.1	Principe de fonctionnement	94

6.2.2	Primitives implicites	95
6.2.3	Déplacement des sommets	96
6.2.4	Mélange des primitives	100
6.3	Résultats	100
6.3.1	Comparaison qualitative	100
6.3.2	Performances	104
6.4	Discussion	105
7	Conclusion	109
7.1	Récapitulatif des contributions	109
7.2	Discussion et travaux futurs	110
7.2.1	Rendu de surfaces implicites	110
7.2.2	Mélange de surfaces implicites	110
7.2.3	Animation de maillages par surfaces implicites	111
	Bibliographie	115
	A Equation des courbes d'ouverture	121
	B Précalcul des opérateurs de mélange	125
	Table des acronymes	127
	Table des figures	129
	Liste des tableaux	137

Avant-propos

Au cours de ma thèse, j'ai eu l'occasion de travailler sur deux thèmes distincts. Le premier sujet auquel j'ai consacré près d'un an et demi concerne une méthode d'imagerie médicale à l'étude depuis le début du 21ème siècle, à savoir la tomographie optique. Cette technique consiste à utiliser un spectre de rayonnement dans le domaine visible (proche infrarouge) pour détecter des tumeurs potentielles dans les tissus *mous*, *i.e.* qui ne contiennent pas d'os. L'intérêt de ce rayonnement est qu'il n'est pas nocif pour les tissus au contraire de celui utilisé dans d'autres techniques d'observation telles que l'observation par rayons X, ou bien la Tomographie par Emission de Positons qui nécessite l'injection d'un isotope radioactif. Malheureusement, les tissus biologiques sont fortement diffusants pour le rayonnement infra rouge de sorte qu'il est difficile de modéliser le comportement précis des photons au sein d'un tel milieu. Les méthodes traditionnelles de l'Etat de l'art utilisent pour la plupart des hypothèses simplificatrices sur la nature de la propagation des photons dans les tissus, mais ces hypothèses imposent des restrictions sur les cadres d'application de ces méthodes. Nous avons étudié la possibilité d'utiliser une méthode plus générale et non-restrictive, à savoir la méthode d'intégration de Monte-Carlo, mais celle-ci converge très lentement et génère un bruit résiduel qui affecte grandement la qualité des résultats.

N'ayant pas obtenus de résultats convaincants, j'ai alors saisi l'opportunité de travailler sur un deuxième sujet concernant la modélisation géométrique par surfaces implicites. La modélisation par surfaces implicites est une manière élégante de représenter des surfaces géométriques et possède de nombreux avantages (garantie d'avoir un objet manifold, pas d'artefacts liés à la discrétisation des surfaces, prise en compte automatique des changements de topologie au cours du temps, ...). Elle a toutefois été délaissée à la fin des années 90 en raison des difficultés et contraintes qu'elle impose :

- affichage des surfaces non immédiat, pouvant nécessiter un traitement préalable
- forme des surfaces difficilement contrôlable
- coût d'évaluation bien souvent non compatible avec des applications temps réel.

Durant cette thèse, je me suis attaqué à ces problèmes. Les contributions apportées

montrent que certains défauts des surfaces implicites peuvent être surmontés et résolus de manière simple, et j'espère convaincre le lecteur que les surfaces implicites ne doivent pas être enterrées définitivement.

Le manuscrit est ainsi décomposé en deux parties bien distinctes. La première, courte, concerne mes travaux sur la tomographie optique. La seconde, plus importante, est dédiée au reste de mes travaux effectués sur les surfaces implicites.

Première partie
Tomographie optique

1

Présentation

La tomographie optique est une technique d'imagerie qui consiste à retrouver les propriétés d'un objet à partir d'une série de mesures effectuées depuis l'extérieur de cet objet. Contrairement aux autres techniques de tomographie (TEP, scintigraphie), la tomographie optique utilise des sources lumineuses émettant des photons dans le bas du spectre visible (≈ 700 nm), ce qui la rend très attractive dans le domaine de l'imagerie médicale. En effet, d'une part ce rayonnement est inoffensif pour les tissus organiques, et la prise de mesures ne nécessite pas l'ingestion préalable d'un produit radioactif. D'autre part, ce type de rayonnement est peu coûteux à produire.

Dans les dispositifs expérimentaux, la lumière est transmise à la surface de l'objet par des fibres optiques. Les photons transmis interagissent avec la matière selon les propriétés du milieu et certains finissent par ressortir de l'objet. D'autres fibres optiques permettent de détecter ces photons en certains points de la surface.

Les deux principales interactions subies par la lumière dans un tissu organique sont l'absorption (qui se traduit par une diminution de l'intensité lumineuse) et la diffusion (un changement brutal de direction des rayons lumineux). L'influence de ces interactions varie selon le type de tissu observé et des variations locales (tumeur...) peuvent altérer les mesures de façon significative.

L'objectif de la tomographie optique est ainsi de retrouver les propriétés d'absorption et de diffusion en tout point du milieu étudié à partir des mesures réalisées à sa surface. Ce problème est appelé problème inverse. On appelle problème direct le problème qui consiste à calculer l'intensité lumineuse arrivant aux capteurs, les propriétés d'absorption et de diffusion du milieu étant connues. La résolution du problème inverse fait appel à des méthodes itératives, consistant à faire évoluer une solution de départ en fonction

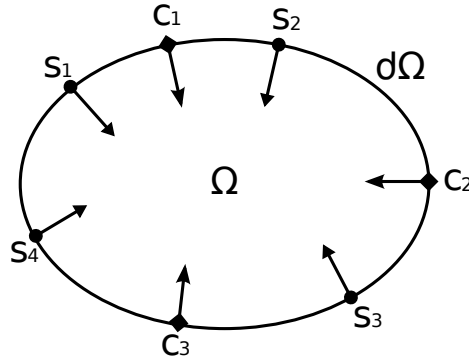


FIGURE 1.1 – Schématisation du domaine

des résultats du problème direct obtenus à chaque itération. Il est donc nécessaire pour résoudre le problème inverse de savoir résoudre le problème direct.

Dans une première partie, nous détaillerons le formalisme utilisé. Dans une seconde partie, nous introduirons les algorithmes permettant de résoudre le problème inverse. Enfin nous étudierons les algorithmes permettant de résoudre le problème direct.

1.1 Formalisme

On note Ω le domaine de l'objet observé et $\delta\Omega$ sa frontière. Pour tout point X de Ω , on appelle coefficient d'absorption $\mu_a(X)$ (resp. de diffusion $\mu_s(X)$) la probabilité qu'un photon soit absorbé (resp. diffusé) sur une unité de distance au point X . L'ensemble des sources lumineuses est noté S (cardinal N_S), celui des capteurs est noté C (cardinal N_C) (cf. fig. 1.1).

Le problème direct est défini de la façon suivante :

Etant donné $(\mu_a(X), \mu_s(X))$ pour tout X dans Ω , calculer $m_{i,j}$ la puissance échangée entre la source i et le capteur j pour tout $(i, j) \in S \times C$.

Et le problème inverse :

Etant donné $m_{i,j}$ pour tout (i, j) dans $S \times C$, retrouver $(\mu_a(X), \mu_s(X))$ pour tout X dans Ω .

Notons $a = \{\mu_a(X) \mid X \in \Omega\}$ et $b = \{\mu_s(X) \mid X \in \Omega\}$. Le problème inverse peut être vu comme le problème d'optimisation de la fonction :

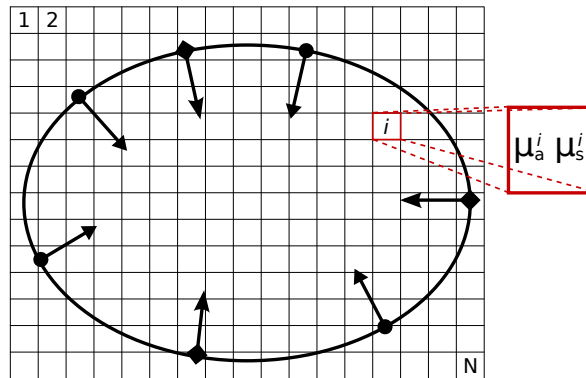


FIGURE 1.2 – La discrétisation du domaine : à chaque voxel i correspond un coefficient d'absorption μ_a^i et un coefficient de diffusion μ_s^i

$$I(a, b) = \sum_{j \in S} I_j(a, b)$$

avec

$$I_j(a, b) = \sum_{i \in C} (M_{i,j} - m_{i,j}(a, b))^2$$

où $M_{i,j}$ est la puissance échangée mesurée entre i et j et $m_{i,j}(a, b)$ la puissance échangée calculée entre i et j dans la configuration (a, b) .

Pour la résolution du problème inverse, le domaine Ω est discrétisé en N voxels. A l'intérieur de chaque voxel, les valeurs des coefficients μ_a et μ_s sont constantes (cf. fig. 1.2).

2

Résolution du problème inverse

Nous avons vu que le problème inverse peut s'écrire comme le problème d'optimisation suivant :

$$\min_{a,b} I(a,b) = \sum_{j \in S} I_j(a,b)$$

Pour résoudre ce problème, il est indispensable de pouvoir évaluer I en tout point (a,b) . Autrement dit il faut pouvoir résoudre le problème direct. Différentes méthodes d'optimisation permettent de résoudre le problème inverse en utilisant la résolution du problème direct :

1. Méthodes d'optimisation stochastiques

Ces méthodes utilisent des événements aléatoires pour guider la recherche d'une solution. Elles sont en général simples à mettre en oeuvre mais convergent plus lentement que les algorithmes déterministes. Parmi les algorithmes stochastiques les plus populaires, on peut citer le recuit simulé [SK83] ou encore les algorithmes génétiques.

2. Méthodes de Newton

Les algorithmes de type Newton recherchent le point où les dérivées de la fonction I à optimiser s'annulent. Pour cela, ils font évoluer une estimation β^s de la solution en utilisant l'approximation de Taylor à l'ordre 1 de I' autour de la solution :

$$\begin{aligned} I'(\beta^{s+1}) &= I'(\beta^s) + I''(\beta^s)h^s = 0 \\ \Rightarrow h^s &= -(I''(\beta^s))^{-1}I'(\beta^s) \end{aligned}$$

L'inconvénient de cette méthode est que le calcul de l'incrément h^s nécessite d'évaluer les dérivées secondes de I . Des évolutions de cet algorithme permettent de s'en passer, par exemple l'algorithme de Gauss Newton.

Etant données m fonctions $r_i, i = 1 \dots m$ de n variables $\beta = (\beta_1, \dots, \beta_n)$ l'algorithme de Gauss Newton permet de trouver le minimum de la fonction

$$I(\beta) = \sum_{i=1}^m r_i(\beta).$$

Comme toutes les méthodes de Newton, l'algorithme part d'une estimation initiale de la solution β^0 puis procède par itération :

$$\beta^{s+1} = \beta^s + \delta\beta$$

où

$$\delta\beta = -H^{-1}\nabla I(\beta_s)$$

avec

- H matrice hessienne de I $h_{i,j} = \frac{\partial^2 I}{\partial \beta_i \partial \beta_j}$
- $\nabla I = 2J^T r$: gradient de I en β^s
- J : matrice jacobienne des fonctions r_i : $j = \frac{\partial r_i}{\partial \beta_j}$

L'idée de Gauss est de faire une approximation de la hessienne à l'ordre 1 :

$$H \approx 2J^T J$$

Ainsi $\delta\beta = -(J^T J)^{-1} J^T r$ et seules les dérivées premières sont nécessaires pour calculer l'incrément. Toutefois, un problème apparaît lorsque $n > m$, autrement dit lorsque le nombre de voxels est supérieur au nombre de mesures effectuées. En effet dans ce cas la matrice $n \times n$ ($J^T J$) est non inversible car son rang est alors inférieur à n . Il faut alors utiliser une autre formulation de l'incrément, par exemple celui de la méthode de Levenberg-Marquardt :

$$\delta\beta = (J^T J + \lambda Id)^{-1} J^T r$$

où le paramètre λ est choisi arbitrairement, de façon à garantir le rang de la matrice ($J^T J + \lambda Id$) et à assurer une bonne vitesse de convergence de l'algorithme. L'utilisation de ces algorithmes en tomographie optique a été étudiée dans [KDP95, MS93].

3

Résolution du problème direct

Comme indiqué précédemment, la résolution du problème direct est une étape indispensable à la résolution du problème inverse. Le mouvement des photons se propageant dans le milieu est modélisé par l'équation de transfert radiatif (ETR) :

$$\frac{1}{c} \frac{\partial f(x, \omega, t)}{\partial t} + \omega \cdot \nabla_x f(x, \omega, t) = s(x, \omega, t) - (\mu_a(x) + \mu_s(x)) f(x, \omega, t) + \mu_s(x) \int_{4\pi} f(x, \omega', t) \phi(\omega, \omega') d\omega'$$

avec :

- f fonction de distribution des photons : $f(x, \omega, t)$ densité de photons en x se déplaçant dans la direction ω à l'instant t .
- ϕ fonction de phase du milieu : $\phi(\omega, \omega')$ probabilité qu'un photon se propageant dans la direction ω soit diffusé en ω' lors d'un évènement de diffusion.
- s terme source : $s(x, \omega, t) = 0 \forall x \notin S$

En tomographie optique, la méthode de résolution de l'ETR la plus utilisée est la méthode des éléments finis associée à une approximation de l'ETR dite approximation de diffusion. Cette méthode a l'avantage d'être rapide mais ne fonctionne que dans un cadre bien précis que nous allons décrire. Cette technique est détaillée dans [Arr99, MSD95]. Nous utilisons une méthode de résolution basée sur Monte-Carlo appliquée à la formulation intégrale de l'ETR, qui a l'avantage d'être valide dans tous les cas.

L'approximation de diffusion est une simplification de l'ETR que l'on obtient lorsque l'on fait les hypothèses suivantes :

- Le milieu est tel qu'il y a beaucoup plus d'évènements de diffusion que d'évènements d'absorption ($\mu_a \ll \mu_s$). Ainsi, après un certain nombre de diffusions, peu de

photons seront absorbés et la fonction de distribution sera presque isotrope.

- Dans un milieu fortement diffusant, la variation de densité de flux de photons est beaucoup plus lente que le temps moyen s'écoulant entre deux diffusions successives d'un même photon.

La première hypothèse permet de ne garder que les deux premiers moments de la décomposition en harmoniques sphériques de la fonction de distribution :

$$f(x, \omega, t) \approx \frac{1}{4\pi} F(x, t) + \frac{3}{4\pi} J(r, t) \cdot \omega$$

avec :

- $F(x, t) = \int_{4\pi} f(x, \omega, t) d\omega$ densité de photons en x à l'instant t
- $J(x, t) = \int_{4\pi} c\omega f(x, \omega, t) d\omega$ flux de photons en x à l'instant t

En substituant cette expression de f dans la RTE on peut obtenir les deux formulations suivantes :

$$\frac{1}{c} \frac{\partial F(x, t)}{\partial t} + \mu_a F(x, t) + \nabla \cdot J(r, t) = S(r, t) \quad (3.1)$$

$$\frac{1}{c} \frac{\partial J(x, t)}{\partial t} + (\mu_a + \mu'_s) J(x, t) + \frac{1}{3} \nabla F(x, t) = 0 \quad (3.2)$$

où $\mu'_s = (1 - g)\mu_s$ est le coefficient de diffusion réduit et $g = \int_{4\pi} \omega \cdot \omega' \phi(\omega, \omega') d\omega'$ est le facteur d'anisotropie de la fonction de phase.

La seconde hypothèse consiste à poser $\frac{\partial J}{\partial t} = 0$ soit en remplaçant dans (3.1) :

$$J(x, t) = -\frac{1}{3(\mu_a + \mu'_s)} \nabla F(x, t)$$

ce qui donne dans (3.2) :

$$\frac{\partial F(x, t)}{\partial t} + \mu_a F(x, t) - \nabla \cdot (\kappa \nabla F(x, t)) = S(x, t)$$

où $\kappa = \frac{1}{3(\mu_a + \mu'_s)}$ est le coefficient de diffusion.

3.1 Résolution par méthode des éléments finis

La méthode des éléments finis est un méthode de résolution numérique des équations aux dérivées partielles. Le domaine Ω est divisé en un maillage de P éléments formés par

D noeuds. La méthode des éléments finis consiste à approcher la solution $F(x, t)$ par une fonction continue par morceaux $F^h(x, t) = \sum_{i=1}^P F_i(t) \cdot u_i(x)$. La qualité de l'approximation dépend des fonctions u_i et du maillage choisis.

L'équation de diffusion s'écrit en termes d'éléments finis [MSD95, SRAD93] :

$$(K(\kappa) + C(\mu_a) + \zeta \cdot A)\Phi(t) + B\left(\frac{\partial \phi}{\partial t}\right) = q_0(t)$$

avec

- $\zeta = \frac{1}{2A}$ où A est un paramètre traduisant l'interaction lumineuse à la frontière $\partial\Omega$ [Aro93, MCC67]
- $K_{ij} = \int_{\Omega} \kappa(x) \nabla u_i(x) \cdot \nabla u_j(x) dx$
- $C_{ij} = \int_{\Omega} \mu_a(x) u_i(x) u_j(x) dx$
- $A_{ij} = \int_{\partial\Omega} u_i(x) u_j(x) dx$
- $B_{ij} = \frac{1}{c} \int_{\Omega} \mu_a(x) u_i(x) u_j(x) dx$

Soit en domaine fréquentiel :

$$(K(\kappa) + C(\mu_a) + \zeta \cdot A + i \cdot \omega \cdot B)\Phi = q_0$$

Le problème direct consiste alors à résoudre le système linéaire ci-dessus (avec une méthode de gradient conjugué par exemple).

Les limitations de cette méthode proviennent essentiellement de l'approximation de diffusion. Outre le cas $\mu_a \ll \mu_s$, l'approximation n'est plus valide lorsque le domaine contient des zones non-diffusantes ou, plus généralement, lorsque le milieu est optiquement mince (par exemple lorsque des capteurs sont placés près des sources, c'est à dire lorsque le libre parcours moyen de diffusion est non négligeable devant la distance source-capteur). C'est pourquoi nous avons choisi d'employer une autre méthode de résolution.

3.2 Méthode de Monte Carlo

La méthode d'intégration de Monte Carlo est une méthode permettant d'estimer numériquement une intégrale. Soit D un domaine et $f : D \mapsto \mathbb{R}$. On souhaite évaluer $I = \int_D f(x) dx$. Soient N variables aléatoires $X_1 \dots X_N \in D$ échantillonnées selon la même

densité de probabilité p . La variable aléatoire

$$J = \frac{1}{N} \sum_{i=1}^n \frac{f(X_i)}{p(X_i)}$$

appelée estimateur de Monte Carlo est un estimateur non biaisé de I . En effet

$$E[J] = \frac{1}{N} \sum_{i=1}^n \int_D \left(\frac{f(x)}{p(x)} \right) p(x) dx = I.$$

L'estimateur de Monte Carlo converge en $O(\sqrt{N})$. En pratique, le choix de la densité de probabilité p a beaucoup d'influence sur la vitesse de convergence. La densité de probabilité idéale $p(x) = \frac{f(x)}{I}$ (échantillonnage par importance) n'étant pas facile à évaluer dans la plupart des cas, il faut essayer de s'en approcher de manière intuitive. Nous utilisons la méthode d'intégration de Monte Carlo pour résoudre la formulation intégrale de l'équation de transfert radiatif :

$$L(x, \omega, t) = s(x_{\delta\Omega}, \omega, t) + \int_x^{x_{\delta\Omega}} \tau(x', x) \mu_s(x') \int_{4\pi} \phi(\omega, \omega') L(x', \omega', t - \frac{\|x' - x\|}{c}) d\omega' dx'$$

avec :

- $x_{\delta\Omega}$ la projection de x sur la frontière $\delta\Omega$ dans la direction $-\omega$
- $L(x, \omega, t)$ la radiance : puissance lumineuse en x arrivant dans la direction ω
- $\tau(x_1, x_2) = e^{-\int_{x_1}^{x_2} \mu_a(x) + \mu_s(x) dx}$: l'atténuation

En substituant l'expression de L dans sa définition :

$$L(x, \omega, t) = s(x_{\delta\Omega}, \omega, t) + \int_x^{x_{\delta\Omega}} \tau(x_1, x) \mu_s(x_1) \int_{4\pi} \phi(\omega, \omega_1) \left[s(x_{1\delta\Omega}, \omega, t) + \int_{x_1}^{x_{1\delta\Omega}} \tau(x_2, x_1) \mu_s(x_2) \int_{4\pi} \phi(\omega_1, \omega_2) [\dots] d\omega_2 dx_2 \right] d\omega_1 dx_1$$

On remarque alors que l'espace d'intégration est divisé en sous-espaces de chemins optiques comportant le même nombre de diffusions. L'expression de la luminance peut donc s'écrire comme une somme infinie d'intégrales :

$$L(x, \omega) = \sum_{i=0}^{\infty} \int_{D_i} f_i(y) dm_i(y)$$

où

- D_i est l'ensemble des chemins optiques comportant i diffusions et reliant le point x à une source

— $dm_i(x)$ la mesure sur l'espace des chemins comportant i diffusions

$$f_i(x) = \frac{\tau(x_0, x_1)}{\|x_0 - x_1\|^2} \phi(x_1 \vec{x}_0, x_2 \vec{x}_1) \frac{\tau(x_1, x_2)}{\|x_1 - x_2\|^2} \phi(x_2 \vec{x}_1, x_3 \vec{x}_2) \cdots \frac{\tau(x_i, x_{i+1})}{\|x_i - x_{i+1}\|^2} s(x_{i+1}, x_{i+1} \vec{x}_i)$$

si l'on note $x = x_0, x_1, \dots, x_{i+1}$.

On peut utiliser la méthode de Monte Carlo pour résoudre chacune de ces intégrales :

$$\int_{D_i} f_i(x) dm_i(x) = \frac{1}{N_i} \sum_{j=1}^{N_i} \frac{f_i(X_j)}{p_i(X_j)}$$

où les X_j sont des chemins optiques comportant i diffusions échantillonnées selon la densité de probabilité p_i choisie. Pour calculer la luminance en un point, il faut donc pouvoir échantillonner des chemins de longueurs arbitraires et évaluer une somme infinie d'intégrales.

3.2.1 Path tracing

Le path tracing [Kaj86] est une méthode itérative de génération de chemins : chaque segment d'un chemin optique est généré à partir du précédent à l'exception du dernier (qui est obtenu en reliant le dernier noeud généré à une source s_j). Plus concrètement, à partir du point x_0 et de la direction ω_0 (en supposant que l'on souhaite évaluer $L(x_0, \omega_0)$), on génère successivement un point de diffusion x_i selon une densité de probabilité $p_{x_{i-1}, \omega_{i-1}}$ puis une direction de diffusion ω_i selon une densité de probabilité $q_{x_i, \omega_{i-1}}$. A chaque point généré, la génération est stoppée avec une probabilité $P \in]0; 1[$ (mécanisme de roulette russe). La probabilité de générer un chemin $x = x_0, \dots, x_n$ est :

$$p(x) = P(1 - P)^{n-1} p_{x_0, \omega_0}(x_1) q_{x_1, \omega_0}(\omega_1) \cdots p_{x_{n-1}, \omega_{n-1}}(x_n) q_{x_n, \omega_{n-1}}(\omega_n).$$

Ainsi pour tout $n \in \mathbb{N}$, la probabilité de générer un chemin de longueur n est non nulle.

Il reste à choisir les densités de probabilité p_{x_i, ω_i} et $q_{x_i, \omega_{i-1}}$. Nous avons vu que pour optimiser la convergence de l'algorithme de Monte Carlo, il faut chercher à s'approcher de la densité de probabilité optimale $\frac{f_i(x)}{I}$. Par analogie avec l'expression de $f_i(x)$, on obtient $p_{x_i, \omega_i}(x_{i+1}) = \alpha \tau(x_i, x_{i+1})$ et $q_{x_i, \omega_{i-1}}(\omega_i) = \phi(\omega_{i-1}, -\omega_i)$ où α est tel que $\int_{x_i}^{x_i + \delta \Omega} p_{x_i, \omega_i}(x_{i+1}) dx_{i+1} = 1$. Intuitivement, les chemins possédant des segments courts

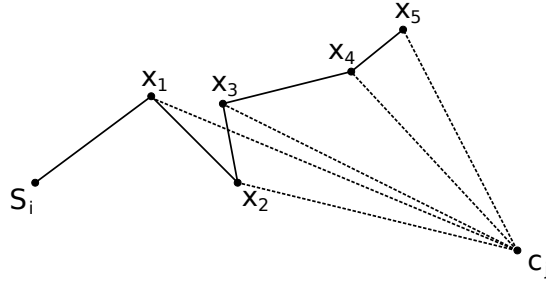


FIGURE 3.1 – Un ensemble de points générés permet d’explorer plusieurs chemins

doivent être échantillonnés plus souvent que les chemins possédant des segments longs car l’atténuation est plus importante sur ces derniers, ce qui est vérifié ici.

L’inconvénient de l’échantillonnage par path tracing est que la taille du dernier segment n’est pas prise en compte. D’autres méthodes d’échantillonnage plus efficaces sont donc possibles, mais un autre avantage du path tracing est la possibilité d’explorer plusieurs chemins avec un seul échantillonnage (cf. Figure 3.1). Cette caractéristique permet d’évaluer la luminance $L(x_0, \omega_0)$ en un point x_0 dans la direction ω_0 avec une seule estimation de Monte Carlo. En effet :

$$L(x_0, \omega_0) = \sum_{i=0}^{\infty} \int_{D_i} f_i(x) dm_i(x) = \sum_{i=0}^{\infty} \left(\frac{1}{N_i} \sum_{j=1}^{N_i} \frac{f_i(X_{i,j})}{p_i(X_{i,j})} \right)$$

où $X_{i,j}$ est le $j^{\text{ème}}$ chemin optique échantillonné comportant i diffusions. Si l’on choisit arbitrairement $N_0 = N_1 = N_2 = \dots = N$:

$$L(x_0, \omega_0) = \frac{1}{N} \sum_{j=1}^N F_j$$

avec

$$F_j = \sum_{i=0}^{\infty} \frac{f_i(X_{i,j})}{p_i(X_{i,j})}$$

qui peut être évalué pour chaque chemin $x_0 \dots x_n s_j$ généré :

$$F_j = \sum_{i=0}^{n-1} \frac{f_i(x_0 \dots x_i s_j)}{p_i(x_0 \dots x_i s_j)}$$

3.3 Sensibilités

Nous avons vu dans la section 2 qu'il est nécessaire de calculer les dérivées par rapport aux paramètres μ_a et μ_s pour pouvoir résoudre le problème inverse. Ces dérivées (appelées aussi sensibilités) peuvent être obtenues par la méthode des éléments finis [Arr99] mais le calcul de chaque sensibilité nécessite alors l'inversion d'un système linéaire. Un autre avantage de la méthode de Monte Carlo est de pouvoir évaluer ces dérivées très simplement et à faible coût. En effet, supposons que l'on souhaite évaluer la sensibilité de l'intégrale suivante $I = \int_D f(x)dx$ à un paramètre α tel que D ne dépend pas de α . D'après l'estimateur de Monte Carlo :

$$\frac{\partial}{\partial \alpha} I = \frac{\partial}{\partial \alpha} \left(\frac{1}{N} \sum_{i=1}^N \frac{f(X_i)}{p(X_i)} \right) = \frac{1}{N} \sum_{i=1}^N \frac{\partial}{\partial \alpha} \frac{f(X_i)}{p(X_i)}$$

On remarque qu'il est possible d'utiliser les mêmes échantillons X_i pour évaluer I et $\frac{\partial}{\partial \alpha} I$, ce qui permet d'économiser les coûts d'échantillonnage.

3.3.1 Application au path tracing

Nos échantillons sont de la forme x_0, \dots, x_n, s_j . La contribution d'un tel chemin peut s'écrire :

$$\frac{f_n(x_0, \dots, x_n, s_j)}{p_n(x_0, \dots, x_n, s_j)} = \frac{f_p^n(x_0, \dots, x_n)}{p_p^n(x_0, \dots, x_n)} \phi(x_n \vec{x}_{n-1}, s_j \vec{x}_n) \tau(x_n, s_j)$$

où les $\frac{f_p^k}{p_p^k}$ sont définis récursivement :

$$\frac{f_p^{k+1}(x_0, \dots, x_{k+1})}{p_p^{k+1}(x_0, \dots, x_{k+1})} = \frac{f_p^k(x_0, \dots, x_k)}{p_p^k(x_0, \dots, x_k)} \phi(x_k \vec{x}_{k-1}, x_{k+1} \vec{x}_k) \tau(x_k, x_{k+1})$$

Les sensibilités peuvent alors être calculées de façon récursive. En effet :

$$\frac{\partial}{\partial \alpha} \frac{f_p^{k+1}(x_0, \dots, x_{k+1})}{p_p^{k+1}(x_0, \dots, x_{k+1})} = \left(\frac{\partial}{\partial \alpha} \frac{f_p^k(x_0, \dots, x_k)}{p_p^k(x_0, \dots, x_k)} \right) \phi(x_k \vec{x}_{k-1}, x_{k+1} \vec{x}_k) \tau(x_k, x_{k+1}) +$$

$$\frac{f_p^k(x_0, \dots, x_k)}{p_p^k(x_0, \dots, x_k)} \frac{\partial}{\partial \alpha} (\phi(x_k \vec{x}_{k-1}, x_{k+1} \vec{x}_k) \tau(x_k, x_{k+1}))$$

Il est ainsi aisé de calculer successivement les sensibilités sur les chemins $\{x_0, x_1, s_j\}$, $\{x_0, x_1, x_2, s_j\}$, $\{x_0, x_1, x_2, x_3, s_j\}$, ...

4

Résultats

Nous avons effectué des tests de reconstruction sur des milieux synthétiques en deux dimensions homogènes. Dans ces milieux, nous avons ajouté une ou deux inclusions, c'est-à-dire des zones très locales où les coefficients d'absorption et de diffusion varient fortement. Pour chaque cas, nous avons simulé le problème direct puis effectué la résolution du problème inverse à partir des mesures simulées.

Dans notre configuration de test, les sources et les capteurs sont placés alternativement en cercle autour du milieu, conformément aux dispositifs expérimentaux réalisés dans [Arr99, MSD95] (cf. Figure 4.1). Pour la reconstruction, nous avons utilisé pour configuration initiale les caractéristiques du milieu de test sans la présence des inclusions.

Les résultats de reconstruction obtenus sont visibles en Figure 4.2. Sur le cas 8×8 , on peut remarquer que la position de l'occlusion a bien été retrouvée, mais qu'il n'a pas été possible de déterminer sa véritable nature. En effet, celle-ci représentait une variation du coefficient de diffusion, tandis que la reconstruction la fait apparaître également au

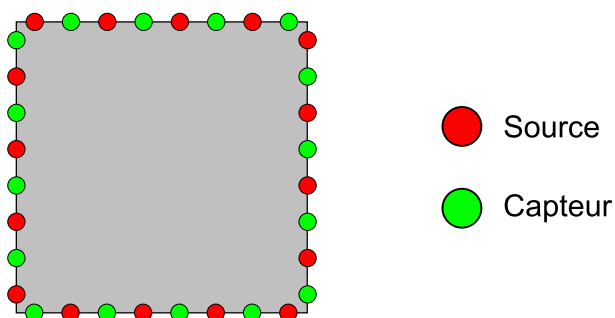


FIGURE 4.1 – Configuration des sources et des capteurs employée dans nos simulations.


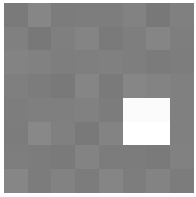

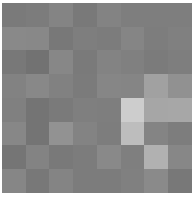

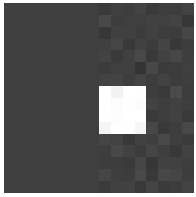
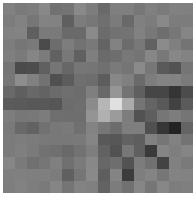
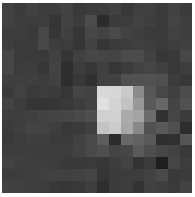

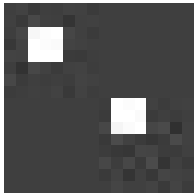

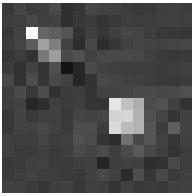
Résolution	μ_a réel	μ_s réel	μ_a reconstruit	μ_s reconstruit
8×8				
16×16				
16×16				

FIGURE 4.2 – Résultats de reconstruction obtenus sur les cas tests

niveau du coefficient d'absorption. Ce résultat est dû à une convergence de l'algorithme de Gauss-Newton dans un minimum local.

Outre la difficulté de distinguer la nature des inclusions, un autre problème de la méthode de reconstruction est que ces minima locaux peuvent être très éloignés de la solution du problème inverse, lorsque la résolution augmente. Ce problème est particulièrement visible dans le deuxième exemple en 16×16 où l'algorithme a échoué à retrouver les deux inclusions.

Devant les difficultés de convergence de la méthode de reconstruction, nous avons étudié différents moyens d'améliorer les résultats obtenus.

Une première approche fut d'utiliser une méthode de reconstruction en multi-résolution. Celle-ci consiste à résoudre plusieurs fois le problème inverse en augmentant la résolution de la grille entre deux résolutions successives. Chaque solution trouvée est alors rééchantillonnée dans la résolution suivante et est utilisée en tant que configuration initiale dans la méthode de Gauss-Newton pour le problème suivant. Ceci permet de rendre l'algorithme de résolution plus stable que dans le cas où la grille de plus grande taille est utilisée d'entrée.

Néanmoins, un inconvénient de cette technique est que le nombre de variables à déterminer (i.e. le nombre de coefficients dans la grille) augmente de manière exponentielle à chaque nouvelle itération. Le temps de calcul nécessaire à la résolution du problème inverse devient alors rapidement prohibitif, principalement en raison du temps nécessaire à la convergence de l'algorithme de Monte-Carlo dans la résolution du problème direct.

Nous avons alors recherché un moyen de réduire le nombre de variables en jeu dans la résolution du problème inverse. Nous avons formulé l'hypothèse que les inclusions possédaient une forme géométrique suffisamment simple pour pouvoir être décrites par quelques paramètres. Une méthode de résolution du problème inverse peut alors s'exprimer en deux étapes :

- Une étape de résolution par la méthode de Gauss-Newton sur une grille de faible taille, permettant de localiser grossièrement les inclusions.
- Une étape consistant à placer des objets géométriques simples modélisant ces inclusions dans le milieu puis à affiner les paramètres de ces objets via une seconde résolution de Gauss-Newton pour localiser précisément la position et la forme des inclusions.

Il reste à choisir une méthode de modélisation géométrique de ces inclusions. Nous avons choisi d'utiliser une modélisation par objets implicites. Celle-ci permet en effet de modéliser des objets complexes par assemblage d'un petit nombre de primitives de base, chacune étant définie par un faible ensemble de paramètres.

C'est donc ce contexte de recherche qui nous a amené à travailler sur la modélisation par surfaces implicites. Les travaux développés sont présentés dans un cadre plus général que la seule résolution du problème inverse en tomographie optique. Ainsi, les travaux portant sur l'intersection des rayons lumineux avec des surfaces implicites, nécessaire à l'utilisation du path tracing pour la méthode de Monte-Carlo, sont applicables dans le domaine plus général de la visualisation de primitives implicites. De même les travaux menés dans le domaine de l'assemblage d'objets implicites présentent un intérêt certain dans le domaine général de la modélisation géométrique par surfaces implicites.

Deuxième partie

Surfaces implicites en modélisation géométrique

1

Introduction

Ce chapitre présente les contributions de cette thèse apportées dans les domaines de la modélisation, de la visualisation et de l’animation des surfaces implicites.

Les surfaces implicites ont prouvé leur utilité en tant que représentation polyvalente, et ce dans un large domaine des disciplines graphiques incluant la vision assistée par ordinateur, la modélisation géométrique et l’animation. Elles permettent de manipuler simplement des surfaces évoluant au cours du temps et dont la topologie varie de manière complexe.

1.1 Motivation

La découverte d’une représentation numérique satisfaisante pour représenter les objets du monde réel dans un environnement numérique a fait l’objet de recherches durant des décennies. De nombreux types de représentations ont été développés pour répondre à la variété des contraintes et des exigences demandées. Parmi celles-ci, les représentations par maillages sont les représentations les plus communes pour représenter les surfaces, en raison de leur compacité et de leur simplicité. Un grand nombre de techniques de modélisation et de visualisation profitent ainsi de leurs avantages. Cependant, les représentations par maillages ont aussi leurs inconvénients. Celles-ci ne sont qu’une représentation linéaire par morceaux de la surface qu’elles représentent, et la modélisation d’une surface lisse avec une précision suffisante nécessite souvent une tessellation fine des polygones. Pour compenser la continuité C^0 de la surface, les normales en chaque sommet du maillage sont souvent incluses dans les données du maillage puis interpolées sur chaque face, une approximation qui peut se révéler être très grossière dans certains cas. Enfin ces représentations, comme toutes les représentations paramétriques, permettent la modélisation d’objets non-manifolds. Les surfaces représentées ainsi peuvent donc s’auto-intersecter ou présenter des

caractéristiques incompatibles avec les objets du monde réel.

En revanche, les surfaces implicites représentent les objets qu'elles modélisent de façon volumique. Elles forment ainsi une classification de l'espace en deux régions : l'espace situé à l'intérieur de l'objet et celui situé à l'extérieur. La surface située à la frontière de ces deux zones est alors manifold. Les informations de topologie de l'objet sont contenues implicitement dans cette représentation volumique et ne nécessitent pas de gestion explicite, au contraire des représentations par maillage. C'est pour cette raison que les surfaces implicites sont capables de représenter très simplement des objets dont la surface et la topologie sont dynamiques. Elles sont ainsi très adaptées à la modélisation de fluides visqueux, par exemple des vagues d'eau s'échouant sur la berge.

Cependant les représentations volumiques des surfaces implicites ont aussi leurs défauts. De par leur nature implicite, il est difficile d'extraire la surface qu'elles représentent. Ceci rend leur visualisation peu aisée. Les algorithmes de visualisation de surfaces implicites, par rastérisation ou lancer de rayon, sont ainsi bien moins rapides que leurs équivalents sur les maillages. En outre, leur utilisation en modélisation géométrique est peu intuitive. Les techniques de composition de surfaces implicites connues jusqu'alors sont difficilement contrôlables. Il est donc ardu d'ajouter des détails sur les surfaces et la réalisation d'objets complexes composés de milliers de primitives de tailles différentes est tout simplement impraticable. Enfin leur coût d'évaluation est généralement élevé, ce qui empêche leur utilisation dans les applications temps réel nécessitant un grand nombre d'évaluations, par exemple en animation. L'objectif de cette thèse est d'aborder certains de ces problèmes et de faire avancer l'état de l'art dans les domaines de la modélisation, la visualisation et l'animation des surfaces implicites.

La plupart des algorithmes présentés dans ce document ont été implémentés en exploitant la puissance des cartes graphiques (également appelées GPU de l'anglais *Graphics Processing Unit*) de la dernière génération. Les cartes graphiques ont été introduites en 1996 pour accélérer le processus de rastérisation des triangles dans les applications de visualisation 3D. Les performances et les capacités des GPUs furent ensuite améliorées d'année en année pour répondre aux besoins de l'industrie graphique, principalement dans le secteur des jeux vidéos. Une première évolution fut l'intégration du calcul de transformation et d'éclairage (T&L) sur les GPUs en 1999. Deux années plus tard, l'introduction des unités programmables appelées *Shaders* permit l'arrivée d'architectures plus flexibles. Les architectures unifiées d'aujourd'hui permettent ainsi de programmer totalement les

GPUs afin d'effectuer des traitements génériques [PF05].

1.2 Résumé des contributions

L'objectif de cette thèse a été de développer un ensemble de techniques permettant de rendre plus simple et plus robuste la modélisation géométrique par surfaces implicites. Les contributions réalisées sont les suivantes :

Hierarchie de volumes englobants redimensionnés : Le lancer de rayon est l'une des techniques les plus polyvalentes pour la visualisation de surfaces. Afin de réduire la complexité de calcul d'une intersection, il est courant de structurer les données des surfaces de manière hiérarchique selon des critères de segmentation ou de partition spatiale. Une nouvelle structure de donnée adaptée aux surfaces implicites à support compact (les plus utilisées en pratique) est présentée. Celle-ci permet d'améliorer l'efficacité des calculs d'intersection par rapport aux structures de données existantes.

Opérateurs de mélange contrôlés par le gradient : La modélisation géométrique par surfaces implicites est généralement réalisée par agrégation d'objets géométriques simples. Ceux-ci sont combinés entre eux de manière à réaliser de nouvelles surfaces implicites. Les fonctions de composition utilisées jusqu'alors pour réaliser ces combinaisons possèdent un certain nombre de défauts qui rendent difficile la modélisation précise d'un objet complexe. En utilisant les informations de gradient des primitives implicites, il devient possible de supprimer les défauts des opérateurs de compositions et ce de manière automatique. Ces informations de gradient peuvent également être utilisées pour augmenter la variété des compositions réalisables.

Animation de maillages par surfaces implicites : Les représentations par maillage sont très souvent utilisées en animation. Lorsque la topologie de l'objet à animer ne change pas, il suffit en effet de déplacer les sommets du maillage pour animer celui-ci. En revanche, les déformations à effectuer sont souvent complexes et les techniques connues nécessitent bien souvent l'intervention d'un infographiste a priori ou a posteriori pour obtenir un résultat convaincant. Il est par contre aisé de déformer un ensemble de surfaces implicites par utilisation d'opérateurs de composition adéquats. En réalisant un prototype de l'objet à déformer en un nombre limité de surfaces implicites, il devient alors possible de créer un modèle de déformation du maillage, les déformations appliquées au prototype étant reproduites sur le maillage par une méthode de suivi de gradient.

1.3 Plan

Les chapitres suivants présentent les travaux réalisés durant cette thèse. Le Chapitre 2 introduit les prérequis nécessaires à la compréhension de la première contribution : une nouvelle structure de segmentation spatiale adaptée aux primitives implicites à support compact permettant d'accélérer les tests d'intersection et de visualiser en temps réel un grand nombre de primitives implicites, présentée au Chapitre 3. Le Chapitre 4 introduit les notions avancées de compositions de surfaces implicites et explique les limitations des opérateurs de composition actuels. La seconde contribution de cette thèse présentée au Chapitre 5 consiste en de nouveaux opérateurs de composition de surfaces implicites. Ceux-ci ne souffrent pas des défauts des opérateurs existants et permettent de faciliter la modélisation géométrique d'objets complexes par des primitives implicites. Ces opérateurs sont utilisés dans le Chapitre 6 afin d'animer des maillages. La troisième contribution présentée dans ce chapitre est en effet une technique d'animation de maillages utilisant une représentation intermédiaire de l'objet à animer basée sur les surfaces implicites.

2

Définition et rendu des surfaces implicites

Ce chapitre présente les définitions de base des surfaces implicites et référence les différentes techniques de visualisation existantes. Les propriétés avancées des surfaces implicites sont introduites au Chapitre 4, mais ne sont pas nécessaires à la compréhension de la première contribution de cette thèse concernant l’affichage en temps interactif d’un grand nombre de surfaces implicites, présentée au Chapitre 3.

2.1 Introduction et définitions

La modélisation d’objets géométriques tridimensionnels complexes est souvent réalisée par agrégation de primitives simples, points, lignes ou polygones. Les représentations discrètes d’objets par maillage consistent ainsi pour la plupart en un assemblage d’un grand nombre de triangles ou de quadrilatères. Ces représentations sont très utilisées car elles permettent d’obtenir les meilleures performances d’affichage, mais sont complexes à déformer de façon précise et intuitive, notamment lors des changements de topologie de l’objet à modéliser. De manière générale, la modélisation d’objets dynamiques dont la topologie varie au cours du temps, par exemple des fluides, est un problème difficile. Les représentation classiques par maillage ne sont pas adaptées à ce type de surfaces, en raison des informations de connexité qu’elles intègrent et qu’il est peu aisé de mettre à jour lorsque les configurations topologiques changent.

Une méthode alternative de modélisation est basée sur l’utilisation de *champs potentiels*.

Définition 1 *Un champ potentiel $f : \mathbb{E}^3 \rightarrow \mathbb{R}$ est une fonction qui associe une valeur de*

potentiel $f(p)$ à chaque point p de l'espace euclidien.

Pour chaque *champ potentiel* et chaque nombre réel C , il est possible de définir un volume correspondant à l'ensemble des points dont la valeur du champ est supérieure ou égale à C . Cette définition du volume est alors *implicite* en ce sens qu'elle donne un moyen de tester l'appartenance de tout point à ce volume sans indiquer explicitement l'ensemble des points qu'il contient.

Définition 2 Soit un champ potentiel $f : \mathbb{E}^3 \rightarrow \mathbb{R}$ et $C \in \mathbb{R}$. Le volume implicite défini par le couple (f, C) est l'ensemble $V = \{p \in \mathbb{E}^3 \mid f(p) \geq C\}$.

De la même manière, il est possible de définir des surfaces de façon *implicite* en considérant les points situés à la frontière d'un *volume implicite*.

Définition 3 Soit un champ potentiel $f : \mathbb{E}^3 \rightarrow \mathbb{R}$ et $C \in \mathbb{R}$. La surface implicite définie par le couple (f, C) est l'ensemble $S = \{p \in \mathbb{E}^3 \mid f(p) = C\}$. Celle-ci est appelée l'isosurface de f de valeur C .

Le principal intérêt de cette représentation est qu'il est possible de fusionner plusieurs surfaces implicites en réalisant des compositions de leurs champs potentiels respectifs. Diverses transitions peuvent être réalisées entre les objets en fonction de l'opérateur de composition utilisé. Les premiers travaux se contentent de réaliser la somme de tous les *champs potentiels* de manière à raccorder de façon lisse les primitives implicites [WMW86, Bli82]. L'apparence des raccords entre primitives est alors uniquement contrôlable par les paramètres définissant les *champs potentiels*.

Cette représentation est efficace pour la modélisation de fluides car elle ne contient aucune information de connexité interne. Typiquement, de telles surfaces déformables sont modélisables simplement par un grand nombre de primitives implicites simples appelées *metaballs* [Blo97b] (cf. Figure 2.1).

Une *metaball* i est un champ potentiel radial centré en un point p_i . Ce champ est décrit par une fonction de densité f_i qui décroît quand on s'éloigne de p_i . Les lignes de champ d'une telle fonction sont illustrées en Figure 2.1. Les lignes de champ correspondent à des ensembles de points de même valeur de potentiel et sont circulaires dans le cas d'une *metaball* à 2 dimensions ou sphériques en 3D. Le champ potentiel g généré par N *metaballs* en un point x est :

$$g(x) = \sum_{i=0}^N f_i(x)$$

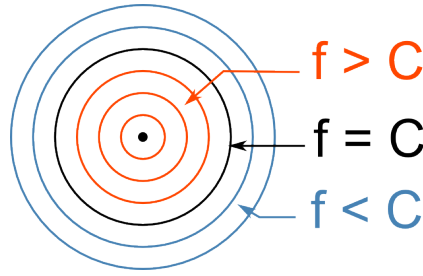


FIGURE 2.1 – Illustration des lignes de champs d'une *metaball* en 2D.



FIGURE 2.2 – Exemple de modélisation d'objets dynamiques à topologie variable à base de *metaballs* (Source : <http://www.planit3d.com>)

L'objet modélisé par ces *metaballs* est représenté par la surface obtenue à la résolution de l'équation

$$g(x) = C$$

pour une certaine isovaleur C choisie. Autrement dit, visualiser l'objet revient à visualiser l'isosurface du champ potentiel g de valeur C .

La *metaball* originale créée par Blinn [Bli82] est obtenue en définissant les fonctions de densité f_i par des gaussiennes :

$$f_i(x) = e^{-R_i \|x\|^2}.$$

D'autres formes communes utilisent des fonctions polynomiales par morceaux, par exemple :

$$f_i(x) = \begin{cases} \left(1 - \frac{\|x - p_i\|^2}{R_i^2}\right)^\alpha & \text{si } \|x - p_i\| \leq R_i \\ 0 & \text{autrement.} \end{cases} \quad (2.1)$$

Plus la valeur de α est élevée, plus la surface générée est lisse. Au contraire des *metaballs* de Blinn, ces fonctions de densité ont un *support compact*.

Définition 4 Une primitive implicite est à support compact s'il existe un volume compact V de \mathbb{E}^3 en dehors duquel son champ potentiel est constant et de valeur nulle.

Le champ potentiel d'une primitive à support compact est en général défini de façon à

prendre ses valeurs dans l'intervalle $[0; 1]$. L'isovaleur C générant la surface observée est également prise dans cet intervalle, en général égale à $\frac{1}{2}$. Ces primitives sont généralement obtenues à partir d'un objet géométrique simple, point (dans le cas des *metaballs*), ligne, ou polygone, appelé *squelette*. Le champ potentiel f est alors exprimé comme une fonction de la distance au squelette S :

$$\begin{aligned} f : \mathbb{E}^3 &\rightarrow \mathbb{R} \\ p &\mapsto f_d(d(p, S)) \end{aligned}$$

où d est la distance euclidienne du point p au squelette S . f_d est définie de manière à limiter le support du champ potentiel à un certain rayon R autour du squelette. Ainsi $f_d(r) = 0$ pour tout $r \geq R$. De façon à minimiser les oscillations du champ potentiel, la variation de f_d doit être uniforme sur $[0; R]$. Sa valeur est donc maximale en $r = 0$ et elle décroît sur $[0; R]$.

La surface est donc définie par l'ensemble des points p tels que $d(p, S) = f_d^{-1}(C)$ et le volume par l'ensemble des points p tels que $d(p, S) < f_d^{-1}(C)$.

L'influence limitée des primitives à support compact est un avantage en terme de modélisation. En effet, l'ajout d'une primitive n'a alors qu'une influence locale sur l'objet modélisé, ce qui permet leur utilisation dans un processus de modélisation incrémental. En outre, cette propriété peut être utilisée pour accélérer l'évaluation du champ potentiel $g(x)$ en tout point x en ignorant la contribution des primitives dont le support ne contient pas x .

De par leur simplicité, les *metaballs* permettent une modélisation rapide et efficace d'objets à topologie variable, mais leur visualisation est un problème difficile. En effet, leur nature implicite empêche de calculer facilement l'ensemble des points appartenant à l'isosurface observée.

2.2 Méthodes de rendu de surfaces implicites

Les techniques de rendu de surfaces implicites peuvent être classées en trois grandes familles : les approches par polygonisation, les approches de rendu par points et les approches par lancer de rayons (cf. Figure 2.3).

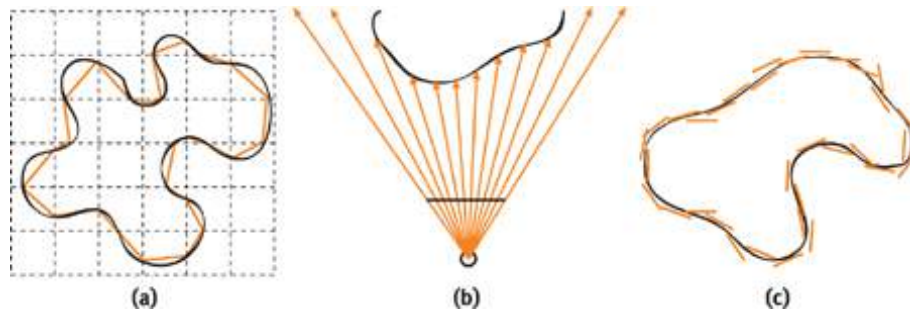


FIGURE 2.3 – Les différentes méthodes de visualisation de surfaces implicites. (a) Polygonisation. (b) Lancer de rayons. (c) Rendu par points.

2.2.1 Polygonisation des surfaces

Les techniques de visualisation par polygonisation consistent à calculer une approximation de l'isosurface par un maillage fin. Cette surface peut alors être visualisée en utilisant les techniques classiques de visualisation de maillage, par rastérisation ou lancer de rayon.

L'algorithme des *Marching Cubes* de Lorensen et Cline [LC87] est la méthode la plus employée pour extraire l'isosurface du champ potentiel. Elle se décompose en deux étapes :

1. Le champ potentiel est évalué à tous les sommets d'une grille régulière.
2. Pour chaque cellule de la grille, calculer un petit ensemble de triangles qui approxime l'isosurface à l'intérieur de celle-ci.

Cet algorithme s'appuie sur l'hypothèse selon laquelle la topologie de l'isosurface à l'intérieur d'une cellule ne dépend que de la valeur du champ potentiel en ses huit sommets. Plus précisément, on évalue pour chaque sommet s'il est à l'intérieur ou à l'extérieur de l'isosurface. Il y a ainsi 256 configurations possibles qui, par symétrie, peuvent se réduire à 15, et pour lesquelles on précalcule la triangulation correspondante (cf. Figure 2.4). Une fois la configuration choisie, les sommets des triangles sont ramenés sur l'isosurface en utilisant une approximation linéaire du champ potentiel à l'intérieur de la cellule.

Récemment, des adaptations de cette technique à l'architecture des GPU ont été développées. Yuri Uralsky [Ura06] utilise le geometry shader lors de l'étape de tessellation des cellules. Dyken *et al.* [Dyk08] optimisent l'utilisation du geometry shader par l'utilisation d'une hiérarchie de grilles qui permet de ne générer que les triangles visibles. Ces améliorations permettent l'utilisation des *Marching Cubes* dans le cadre du rendu en temps réel.

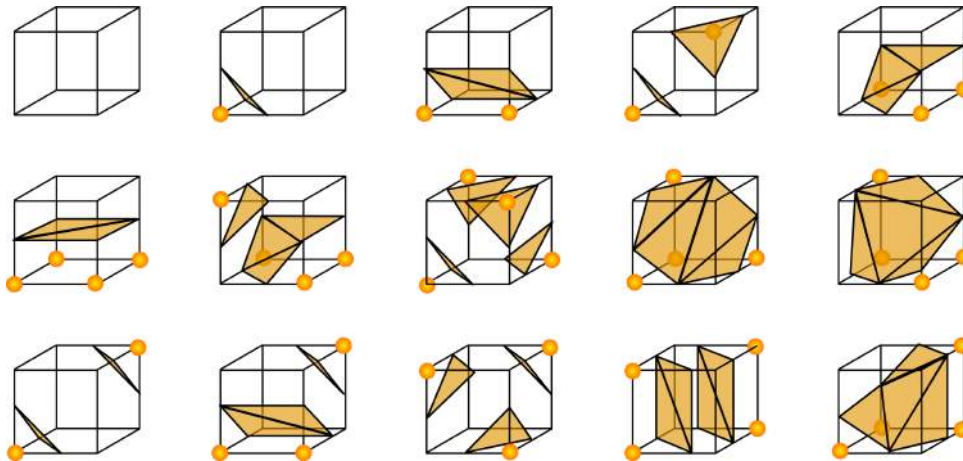


FIGURE 2.4 – Les 15 différentes configurations possibles dans l’algorithme des *Marching Cubes*.

Le principal défaut des *Marching Cubes* est que le maillage obtenu est fortement dépendant de la résolution d’échantillonnage utilisée. Ainsi, certains détails géométriques ou des petites primitives peuvent être manquantes si la résolution de la grille est trop faible. En outre, le coût d’évaluation du champ potentiel aux noeuds de la grille peut devenir rapidement problématique si la résolution employée est élevée. De plus, la triangulation produite est généralement de faible qualité.

2.2.2 Méthodes de rendu par points

Les méthodes de rendu par points consistent à générer sur la surface à visualiser une population de particules espacées de façon uniforme. La surface est alors approximée en générant un disque orienté en chacun des points échantillonnés, et sa visualisation peut ensuite être effectuée par rasterisation de ces disques.

Pour s’assurer de la répartition des particules sur la surface, celles-ci sont mises en mouvement par un système dynamique. Ce système est défini tel que chaque particule exerce des forces de répulsion sur les autres particules proches. Des contraintes sont posées de façon à ce que les particules ne quittent pas la surface lors de leur déplacement.

Dans l’implémentation originale proposée par Witkin et Heckbert [WH94], la population de particules obtenue après convergence du système dynamique peut servir d’initialisation au système de particules pour l’image suivante dans le cas de surfaces implicites en mouvement. L’exploitation de cette cohérence temporelle permet de réduire le temps de simulation du système dynamique. Le nombre de particules utilisé évolue dynamiquement

pour adapter l'échantillonnage lors d'une extension ou d'une contraction de la surface.

Le coût de la simulation du système peut néanmoins s'avérer élevé, une représentation fine de la surface pouvant nécessiter des centaines de milliers de particules. Kooten *et al.*[vKvdBT07] montrent qu'il est possible d'obtenir une visualisation en temps réel en utilisant le GPU lors du calcul de la simulation. Ils utilisent pour cela une grille régulière afin d'accélérer la recherche des voisins de chaque particule.

Un des défauts des méthodes de rendu par points est que des coupures peuvent apparaître à la surface, principalement sur les zones de forte courbure, en raison de l'approximation de la surface par des disques plans. Iwasaki *et al.*[IDYN06] utilisent la position des particules pour calculer une triangulation de haute qualité de la surface implicite. La visualisation de l'isosurface se fait alors par le maillage obtenu, ce qui permet d'éviter les artefacts visuels.

La surface obtenue en rendu par points est fortement dépendante de l'échantillonnage obtenu après la réalisation de la simulation de particules. Comme les méthodes par triangulation, les petits détails géométriques peuvent être manquants lors de la visualisation de la surface.

2.2.3 Lancer de rayons

La visualisation par lancer de rayons consiste à suivre le trajet inverse des rayons lumineux qui passent par chaque pixel de l'image. Pour chacun de ces rayons, on calcule leur première intersection éventuelle avec la surface, le point calculé étant ainsi le premier point de la surface *vu* par le pixel correspondant. Une des grandes forces de cet algorithme est qu'il est possible de lancer d'autres rayons à partir de ce point (appelés rayons secondaires) pour calculer des effets optiques secondaires complexes, par exemple des ombres ou des réflexions. Cette capacité est au coeur des algorithmes de visualisation par Monte Carlo ([Kaj86]).

Lors de l'utilisation de la technique de lancer de rayons pour la visualisation de surfaces implicites, deux difficultés entrent en jeu : le calcul de l'intersection des rayons avec la surface et l'évaluation efficace du champ potentiel pour satisfaire aux contraintes de visualisation en temps réel.

Calcul de l'intersection avec l'isosurface

De nombreuses méthodes pour évaluer l'intersection d'un rayon avec une surface implicite ont été développées. La technique de base appelée *Ray marching* consiste à évaluer le champ potentiel en des points successifs échantillonnés régulièrement le long du rayon. Un intervalle contenant la première intersection est trouvé dès qu'un changement de signe du champ potentiel est détecté. Une recherche linéaire permet ensuite de trouver une approximation très précise du point d'intersection.

Le choix du pas d'échantillonnage est primordial. En effet un pas trop grand risque d'empêcher la détection de l'isosurface si l'objet implicite à visualiser est trop fin. En revanche un pas trop petit augmente inutilement le nombre d'évaluations du champ potentiel. Des méthodes basées sur la condition Lipschitz permettent de faire varier le pas d'échantillonnage au cours de l'algorithme tout en garantissant la détection de la surface (Hart [Har93], Stander [SH94]). Toutefois, ces méthodes nécessitent que le champ potentiel soit une application Lipschitzienne :

$$\exists L \in \mathcal{R}, \forall (x, y) \in \mathcal{R}^3 \times \mathcal{R}^3 \quad \| f(x) - f(y) \| \leq L \| x - y \|$$

Dans le cas où l'équation du champ potentiel est polynomiale par morceaux, une méthode plus efficace proposée par Nishita et Nakamae [NN94] est de représenter le champ potentiel le long du rayon par une série de courbes de Bézier de même degré que le champ. Le problème du calcul de l'intersection avec l'isosurface devient alors le calcul de l'intersection d'une courbe de Bézier avec une droite. Les auteurs appliquent cette méthode avec succès dans le cas du rendu de *metaballs*. L'inconvénient de cette technique est qu'il faut segmenter les rayons de façon à ce que le champ potentiel soit polynomial sur chaque intervalle. Dans le cas où le champ potentiel est généré par des primitives à support compact, cela nécessite de calculer les intersections des supports avec chaque rayon, puis de les trier le long des rayons. Kanamori *et al.* [KSN08] parviennent à calculer efficacement ces intersections en rastérisant les supports des *metaballs* à l'aide du GPU. Plusieurs passes de rendu sont effectuées pour trier les intersection via la technique du *depth peeling*. Cette dernière technique qui utilise le GPU est très efficace et permet d'afficher des dizaines de milliers de *metaballs* en temps réel. Malheureusement, elle est limitée aux rayons primaires et ne peut être utilisée pour calculer des effets secondaires tels que des réflexions.

Structures d'accélération

Dans le cas où le champ potentiel est issu de plusieurs primitives à support compact, il est possible d'utiliser une structure d'accélération afin de diminuer le temps nécessaire

à son évaluation. Les structures les plus efficaces utilisent des arbres pour partitionner l'espace ou les primitives de façon hiérarchique. Cette partition vise à réduire le nombre de primitives à prendre en compte lors de l'évaluation du champ potentiel en un point donné de l'espace. De telles structures sont courantes en lancer de rayons en raison de leur grande efficacité. Les plus communes sont la hiérarchie de volumes englobants (ou *BVH*) et le *kd-tree*.

Une *hiérarchie de volumes englobants* réalise une partition des objets. Chaque objet de la scène est stocké dans une unique feuille de cette structure. Chaque noeud stocke un objet géométrique dont le volume englobe la totalité des objets contenus dans ses fils. La géométrie de ce volume doit être choisie de façon à réaliser un compromis entre deux objectifs. D'une part celle-ci doit être suffisamment simple pour réaliser rapidement des tests d'intersection avec les rayons lancés et ne pas encombrer la mémoire. D'autre part, il est souhaitable que ces volumes soient les plus compacts possibles afin de minimiser les tests d'intersection inutiles. La forme la plus commune est la *boîte englobante*, un parallélépipède rectangle dont les arêtes sont parallèles aux axes. En effet, celle-ci est peu coûteuse en terme d'espace mémoire, et simple à intersecter de manière robuste. Toutefois d'autres types de volumes ont été étudiés et peuvent s'avérer être mieux adaptés dans certains cas, par exemple des parallélépipèdes non alignés aux axes [GLM96] ou des polytopes [KHM⁺98]. Certaines autres formes ont été référencées dans [Hav04].

Une *BVH* peut-être construite rapidement de manière itérative, en divisant chaque noeud de la hiérarchie en deux de façon récursive à partir du volume englobant de toutes les primitives. Cette division peut être réalisée en choisissant un plan de séparation puis en divisant les primitives situées de part et d'autre de ce plan en deux ensembles, chacun formant l'un des deux fils du noeud courant. Ce plan de séparation doit être choisi de façon à minimiser le coût des tests d'intersection. La division doit s'arrêter lorsque le coût de traversée d'un noeud devient plus important que le coût d'intersection des primitives qu'il contient.

L'évaluation du coût des tests d'intersection est complexe. Elle peut être effectuée selon des méthodes de Monte-Carlo mais cela rend le temps de construction de la structure prohibitif. Toutefois, celle-ci peut-être estimée de façon heuristique. L'heuristique la plus utilisée est basée sur la surface des noeuds (*Surface Area Heuristic* ou *SAH*). Celle-ci indique que le coût d'intersection d'un noeud N contenant deux noeuds fils A et B est

$$C_N = \rho_A \times C_A + \rho_B \times C_B$$

où C_i est le coût de calcul d'une intersection dans le noeud i , et ρ_i est la probabilité qu'un rayon intersecte le noeud i sachant qu'il intersecte son noeud parent. ρ_i est estimé par le rapport A_i/A_N de la surface du volume englobant du noeud i sur la surface du volume englobant du noeud parent N . On considère habituellement que C_i est directement proportionnel au nombre de primitives : $C_i = \mathcal{O}(n)$.

Dans [Wal07], Wald propose de choisir le plan de séparation parmi les plans alignés aux axes. En utilisant une stratégie de type *binning*, il est possible de tester de nombreux plans de séparation et d'évaluer la *SAH* correspondante très rapidement. Ceci permet de construire une *BVH* sur des centaines de milliers de primitives en quelques dizaines de millisecondes. Cet algorithme a été porté sur GPU par [LGS⁺09], ce qui réduit son temps de construction d'environ 50%.

Un *kd-tree* est une structure partitionnant l'espace de manière hiérarchique. Chaque noeud divise l'espace en deux volumes disjoints via un plan de séparation aligné aux axes. Les feuilles de cet arbre contiennent toutes les primitives qu'elles intersectent.

Tout comme une *BVH*, la construction d'un *kd-tree* peut être réalisée par divisions récursives du noeud initial contenant toutes les primitives. Le plan de séparation doit être choisi de manière à minimiser les coût d'intersection, qui peuvent être approximés par la *SAH*. La méthode proposée par Wald et Havran dans [WH06] est analogue à la méthode présentée plus haut faisant appel au *binning* des primitives ([Wal07]).

Le principal avantage du *kd-tree* sur la *BVH* est que le non-recouvrement des noeuds d'un même niveau permet de réaliser des tests d'intersection en général plus rapidement. En effet, si les noeuds sont parcourus dans l'ordre de leur traversée, la traversée de l'arbre peut s'arrêter dès qu'une intersection est trouvée dans une feuille, ce qui ne peut pas être garanti lorsque l'on utilise une *BVH*.

La profondeur d'un *kd-tree* est en général plus importante que celle d'une *BVH*, ce qui a pour conséquence un coût de construction plus élevé. Toutefois, dans [ZHWG08], les auteurs montrent que la construction du *kd-tree* sur GPU peut être réalisée en temps réel pour des scènes contenant quelques dizaines de milliers de primitives.

Un autre inconvénient du *kd-tree* est que ses noeuds peuvent contenir beaucoup d'espace vide si les primitives sont relativement éloignées les unes des autres, surtout au début de la hiérarchie. Pour améliorer l'efficacité de la traversée, Havran [HHS06] propose d'uti-

liser des plans de séparation supplémentaires visant à isoler l'espace vide pour les noeuds situés sur les premiers niveaux de la hiérarchie. Malgré les niveaux supplémentaires introduits, cette technique se montre efficace pour réduire les coûts des tests d'intersection. Il faut toutefois 5 à 6 plans supplémentaires pour arriver au niveau de précision d'une boîte englobante, ce qui peut engendrer des branchements non désirables lors d'une utilisation sur des architectures parallèles telles que les GPU.

3

Visualisation d'un grand nombre de surfaces implicites en temps réel

Ce chapitre présente la première contribution de cette thèse dans le domaine du rendu d'un grand nombre de surfaces implicites en temps interactif. Nous nous focalisons dans ce chapitre sur les *metaballs* à support compact définies par l'Equation 2.1 du Chapitre 2, mais les techniques que nous avons développées sont compatibles avec tout type de primitive à support compact. Notre technique de rendu, présentée dans la section suivante, est basée sur le lancer de rayon. Il s'agit en effet de la méthode permettant d'obtenir la meilleure précision de rendu (les méthodes de polygonisation et de rendu par points peuvent manquer certains détails, cf. Section 2.2) et elle est, de plus, suffisamment générique pour gérer de manière uniforme les effets graphiques avancés (réflexions, ombres, etc.). Dans la section finale, nous discutons des détails de notre implémentation utilisant à la fois CPU et GPU et des résultats que nous avons obtenus.

3.1 Contributions

Nos travaux de recherche se sont focalisés sur l'adaptation et l'amélioration des structures d'accélération existantes aux primitives implicites. Nous présentons tout d'abord le test d'intersection rayon/isosurface utilisé, puis nous montrons comment utiliser efficacement ces structures d'accélération pour le rendu de surfaces implicites.

3.1.1 Test d'intersection

Notre méthode de détection d'une intersection entre un rayon et l'isosurface se déroule en deux étapes. Tout d'abord, nous recherchons un intervalle $[a; b]$ le long du rayon contenant uniquement la première intersection. Ensuite, nous réduisons cet intervalle via la

méthode de la sécante [MKF⁺04, NMHW02] jusqu'à atteindre une précision suffisante. La méthode de la sécante converge toujours si l'intervalle initial ne contient qu'une intersection.

Le problème revient donc à trouver un tel intervalle. Pour cela, nous supposons que dans l'éventualité où le rayon intersecte la surface, celui-ci passe à proximité du centre p_i d'une *metaball*. Ainsi, le champ potentiel au point p_{m_i} , projection de p_i sur le rayon, est positif. Nous fixons alors b au premier des p_{m_i} rencontrés par le rayon où le champ potentiel est positif et a à l'origine du rayon. Si $f(p_{m_i}) < 0$ pour tout i , nous considérons que le rayon n'intersecte pas la surface en raison de notre hypothèse.

Nous avons constaté que notre hypothèse était vérifiée pour la plupart des rayons. Cependant, elle peut devenir fautive pour certains rayons proches du plan tangent à la surface, et ainsi certains rayons sont détectés comme n'intersectant pas la surface à tort. Cependant, nous avons constaté que ces rayons tangents à la surface ne créent pas d'artefacts visuels, d'autant plus que des milieux denses en *metaballs* limitent l'apparition de silhouettes.

Une fois le point d'intersection calculé, nous obtenons la normale à la surface en ce point de manière standard, en calculant le gradient du champ potentiel.

3.1.2 Adaptation de la BVH

Les *metaballs* étant des primitives implicites à support compact, il est judicieux d'utiliser une structure d'accélération, pour rendre les calculs d'intersection plus rapides. Nous avons choisi d'utiliser une *BVH* pour sa rapidité de construction et sa capacité à isoler efficacement les zones d'espace vide.

Cependant, l'utilisation d'une *BVH* sur des primitives implicites à support compact nécessite certaines précautions. En effet, une *BVH* classique construite sur les supports des primitives implicites ne fonctionne pas car certaines primitives hors d'un noeud peuvent néanmoins avoir de l'influence sur l'isosurface contenue à l'intérieur de ce noeud. Nous appelons de telles primitives des *primitives de split* :

Définition 5 (Primitive de split) Une primitive de split d'un noeud N est une primitive qui n'appartient pas à N et dont le support intersecte le support d'une primitive de N .

Pour rendre la *BVH* fonctionnelle, il est nécessaire et suffisant que chaque feuille contienne les informations de ses *primitives de split*. Ces informations peuvent être générées aisément durant la construction de la structure.

Nous construisons notre *BVH* comme une *BVH* classique, à partir de sa racine. Comme dans [Wal07], pour chaque noeud nous évaluons plusieurs plans de séparation de façon efficace en utilisant la méthode du *binning*. Le meilleur plan de séparation est estimé par évaluation d'une *SAH*. Le coût d'intersection estimé dans le cas où le noeud est subdivisé est ensuite comparé au coût d'intersection estimé en transformant ce noeud en feuille afin de décider de l'arrêt ou non de la procédure de subdivision récursive. A chaque fois qu'un noeud est divisé en deux, nous évaluons puis stockons dans chaque fils les *metaballs de split* correspondantes. Pour cela, nous parcourons pour chaque fils les *metaballs de split* de leur noeud parent et les *metaballs* de l'autre fils. Si le support d'une de ces *metaballs* intersecte le support d'une *metaball* du noeud fils, nous l'ajoutons à la liste des *metaballs de split* de ce noeud fils (cf. Algorithme 1).

Algorithm 1: Construction d'une BVH pour des metaballs

```

1: BUILDNODE(nodeMballs, splitMballs, nodeBbox) {
2:   find best splitting plane  $s$  for nodeBbox using binning
3:   leftChildMballs, rightChildMballs  $\leftarrow \emptyset$ 
4:   leftChildBbox, rightChildBbox  $\leftarrow$  emptyBbox
5:   for each  $i$  in nodeMballs do
6:     if  $p_i$  is at the left of  $s$  then
7:       leftChildMballs  $\leftarrow$  leftChildMballs  $\cup \{i\}$ 
8:       leftChildBbox  $\leftarrow$  leftChildBbox  $\cup$  bbox( $i$ )
9:     else
10:      rightChildMballs  $\leftarrow$  rightChildMballs  $\cup \{i\}$ 
11:      rightChildBbox  $\leftarrow$  rightChildBbox  $\cup$  bbox( $i$ )
12:    end if
13:  end for
14:  leftSplitMballs  $\leftarrow \emptyset$ 
15:  for each  $i$  in splitMballs  $\cup$  rightChildMballs do
16:    if  $i$  overlaps the bounding sphere of any  $j \in$  leftChildMballs then
17:      leftSplitMballs  $\leftarrow$  leftSplitMballs  $\cup \{i\}$ 
18:    end if
19:  end for
20:  BUILDNODE(leftChildMballs, leftSplitMballs, leftChildBbox)
21:  execute the instructions 14 – 20, but for the right node }

```

La *BVH* accélère les tests d'intersection en réduisant le nombre de primitives utilisées lors du calcul de l'intersection : lorsque nous recherchons une intersection dans une feuille,

nous explorons uniquement les projections p_{m_i} de chaque metaball i appartenant à cette feuille et utilisons uniquement les *metaballs* et *metaballs de split* de la feuille pour calculer le champ potentiel de la feuille en tout point. Remarquons qu'il est inutile de tester la valeur du potentiel aux projections p_{m_j} des *metaballs de split* j car elles appartiennent à d'autres feuilles qui seront explorées si nécessaire.

Les résultats obtenus présentés en section 3.1.4 montrent que la BVH est une structure permettant d'accélérer efficacement le rendu. Cependant celle-ci perd de son efficacité lorsque la répartition des primitive est dense. En effet, du fait de la possibilité de recouvrement des feuilles, la même intersection peut être calculée plusieurs fois. Ceci devient problématique lorsque la densité des primitives est importante car chaque feuille possède alors de nombreuses *primitives de split* qui augmentent la complexité du calcul du champ potentiel. L'évolution que nous proposons permet de s'affranchir de ce problème.

3.1.3 Hiérarchie de Volumes Englobants Redimensionnée

Nous avons développé une nouvelle structure d'accélération baptisée *Hiérarchie de Volumes Englobants Redimensionnée* (alias *FBVH - Fitted Bounding Volumes Hierarchy*) réduisant le nombre de tests d'intersection redondants présents lors de l'utilisation d'une *BVH* classique comme indiqué dans la section précédente.

Définition 6 (FBVH) *Une FBVH est un arbre binaire dans lequel les boîtes englobantes de deux noeuds fils définissent une partition de la surface englobée par leur parent. Contrairement à une BVH, les noeuds du même niveau de cet arbre ne se superposent pas.*

Le fait que les noeuds d'un même niveau ne se croisent pas permet d'optimiser la traversée de la structure lors de l'évaluation d'une intersection. En effet, on peut arrêter le parcours de l'arbre dès qu'une intersection est trouvée, à la manière d'un *kd-tree*. Ceci permet de supprimer tous les tests redondants présents avec une *BVH*.

Les noeuds d'une *FBVH* ne correspondent plus au volume englobant des primitives qu'ils contiennent. A la place, ils déterminent une région de l'espace qui contient une portion de la surface générée par leurs *metaballs* et leurs *metaballs de split*. Il n'y a donc plus de réelle différence entre les *metaballs* et les *metaballs de split* d'un noeud : l'union des deux définit l'ensemble des *metaballs* dont le support intersecte la boîte englobante du noeud. Toutefois, nous continuons à faire la distinction entre ces deux ensembles car elle est nécessaire lors de la construction de la structure.

Cette nouvelle structure ressemble à un *kd-tree* de par sa construction et sa traversée. La principale différence entre les deux structures est la présence d'une boîte englobante à chaque noeud, qui permet d'englober précisément la surface des supports des primitives implicites lorsque celles-ci sont éparées. Ceci est essentiel car le coût d'intersection est élevé, comme précisé dans la Section 3.1.3. L'espace vide présent dans les noeuds d'un *kd-tree* peut être réduit en ajoutant des plans de séparation supplémentaires, comme indiqué en Section 2.2.3. Mais cela augmente le nombre de niveaux présents dans la hiérarchie ce qui est susceptible de provoquer des branchements supplémentaires lors du parcours de la structure.

Construction d'une FBVH

La construction d'une *FBVH* est proche de celle d'une *BVH*. La principale différence se caractérise par la présence d'une étape de redimensionnement des boîtes englobantes à la création des noeuds. Lors de la division d'un noeud, nous calculons une première boîte englobante pour chaque noeud fils obtenue en divisant la boîte englobante du noeud au niveau du plan de séparation (cf. Figure 3.1 (a)). Ensuite, nous distribuons les *metaballs* du noeud à son fils gauche ou son fils droit en fonction de leur position relative au plan de séparation, puis nous calculons les *metaballs de split* des deux fils. Pour cela nous parcourons l'ensemble des *metaballs de split* du noeud parent et des *metaballs* du fils droit et nous ajoutons celles dont le support intersecte le fils gauche à ses *metaballs de split* et nous répétons l'opération pour le fils droit (cf. Algorithme 2).

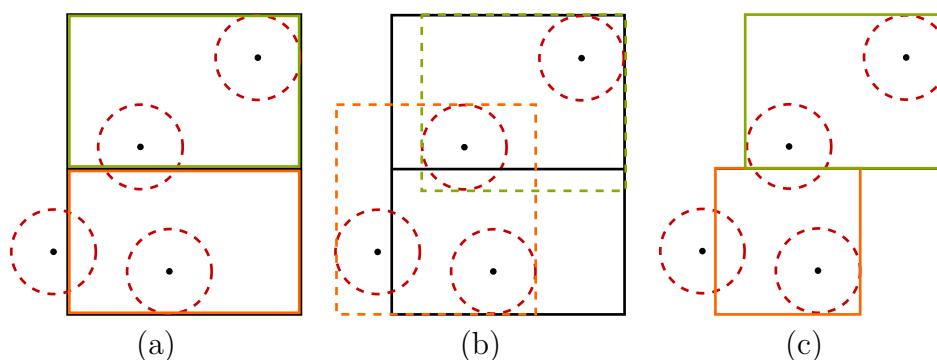


FIGURE 3.1 – Lors de la division d'un noeud, nous calculons les boîtes englobantes des metaballs et des metaballs de split (b). En calculant leur intersection avec les boîtes englobantes initiales (a), nous obtenons leurs nouvelles boîtes englobantes (c).

Nous passons ensuite à l'étape de redimensionnement des noeuds fils. Celle-ci est réalisée en calculant la boîte englobante de l'union des *metaballs* et des *metaballs de split* de chaque fil (cf. Figure 3.1 (b)). L'intersection des boîtes obtenues et des boîtes

Algorithm 2: Construction d'une FBVH

```
1: BUILDNODE(nodeMballs, splitMballs, nodeBbox) {
2: find best splitting plane  $s$  for nodeBbox using binning
3: leftChildMballs  $\leftarrow \emptyset$ , rightChildMballs  $\leftarrow \emptyset$ 
4: leftChildBbox  $\leftarrow$  nodeBbox.splitLeftAlong( $s$ )
5: rightChildBbox  $\leftarrow$  nodeBbox.splitRightAlong( $s$ )
6: leftMballBbox  $\leftarrow$  emptyBbox
7: rightMballBbox  $\leftarrow$  emptyBbox
8: for each  $i$  in nodeMballs do
9:   distribute  $i$  to either the left or right child node and update left and right
   metaball bounding boxes accordingly as in Algorithm 1
10: end for
11: leftSplitMballs  $\leftarrow \emptyset$ 
12: for each  $i$  in splitMballs  $\cup$  rightChildMballs do
13:   if  $i$  overlaps leftChildBbox then
14:     leftSplitMballs  $\leftarrow$  leftSplitMballs  $\cup \{i\}$ 
15:     leftMballBbox  $\leftarrow$  leftMballBbox  $\cup$  bbox( $i$ )
16:   end if
17: end for
18: leftChildBbox  $\leftarrow$  leftChildBbox  $\cap$  leftMballBbox
19: BUILDNODE(leftChildMballs, leftSplitMballs, leftChildBbox)
20: execute the instructions 11 – 19, but for the right node }
```

englobantes initiales permet d'obtenir une nouvelle boîte englobante pour les noeuds fils (cf. Figure 3.1 (c)).

SAH

Nous proposons une nouvelle heuristique du coût d'intersection adaptée à notre test d'intersection présenté en 3.1.1. Habituellement, le coût d'intersection d'une surface générée par un objet géométrique est le même dans chaque noeud, d'où $C_i = \mathcal{O}(n)$, où n est le nombre de primitives du noeud i . Cependant ceci n'est pas vérifié dans le cas de primitives implicites étant donné que le coût d'évaluation du champ potentiel dépend du nombre de primitives dans la feuille courante.

Lorsque nous recherchons une intersection dans une feuille, nous calculons les projections de tous les centres des *metaballs* de la feuille. Nous évaluons le champ potentiel en ces points, et chacune de ces évaluations est réalisée en $\mathcal{O}(n)$ opérations, n étant le nombre de primitives de la feuille. De ce fait, la recherche de l'intervalle initial contenant l'intersection est réalisée en $\mathcal{O}(n^2)$ opérations. Puis nous réalisons la méthode de la sécante afin d'isoler l'intersection. On peut supposer que le nombre d'opérations nécessaires à cette deuxième étape est constant pour atteindre une précision donnée. Nous considérons donc qu'elle est réalisée en $\mathcal{O}(n)$.

Cette analyse nous permet de déduire que le coût d'intersection d'une feuille devrait être :

$$C_i = n^2 + \lambda \times n$$

où n^2 correspond à la recherche de l'intervalle de base et $\lambda \times n$ à la méthode de la sécante. La valeur de λ est difficile à déterminer ; elle dépend du nombre d'étapes nécessaires à la convergence de l'algorithme. Cependant, il est clair que le terme en n^2 est dominant dans le cas de feuilles contenant beaucoup de *metaballs* (ce qui est le cas pour des ensembles denses de *metaballs*). Ainsi, nous pouvons fixer $\lambda = 0$ dans ces situations. Expérimentalement, nous avons observé que négliger $\lambda \times n$ dans tous les cas et fixer $C_i = n^2$ n'avait pas de réel impact négatif sur le temps de construction de la structure ou le temps de rendu.

3.1.4 Optimisations

Quelques optimisations de la structure peuvent être réalisées, permettant d'en augmenter encore l'efficacité. Nous présentons ici deux améliorations simples à mettre en oeuvre et qui se montrent efficaces en pratique.

Calcul des boîtes englobantes

Lorsque nous calculons la boîte englobante d'un ensemble de *metaballs*, nous construisons la boîte de taille minimale qui englobe les supports des primitives. Bien que robuste et simple à mettre en oeuvre, cette façon de procéder n'est pas optimale car l'isosurface observée se trouve généralement à une certaine distance de la frontière des supports. Il est donc possible de calculer une meilleure boîte englobante, dont les faces sont plus proches de l'isosurface, et qui garantirait de meilleures performances en réduisant le nombre de tests d'intersection effectués dans les feuilles de la structure.

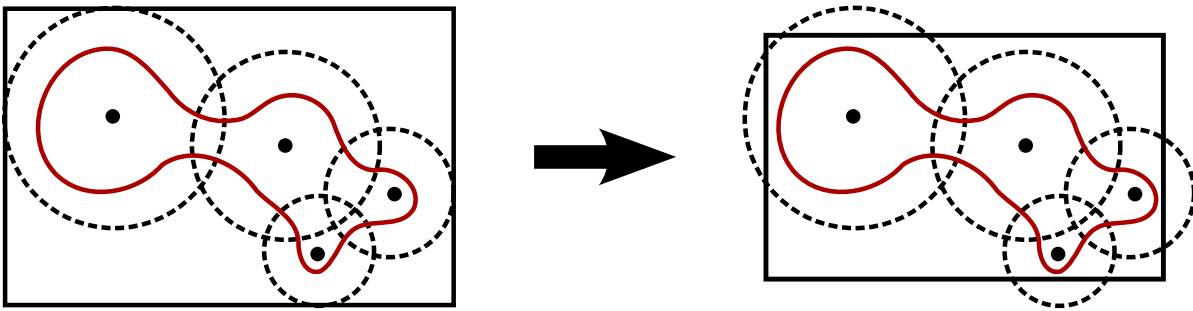


FIGURE 3.2 – La boîte englobante des supports des *metaballs* n'est généralement pas celle qui encapsule de plus près la surface.

Nous redimensionnons les boîtes englobantes des noeuds de manière conservative en procédant de la façon suivante. L'isosurface est au plus près de la frontière du support lorsque toutes les *metaballs* partagent le même centre, créant ainsi une sphère de rayon r tel que :

$$\sum_{i=1}^n f_i(r) = C \quad \text{où} \quad f_i(r) = \left(1 - \frac{r^2}{R_i^2}\right)^\alpha$$

Cette somme est majorée par la valeur $nf_K(r)$ où K est la primitive de rayon maximal appartenant au noeud. Ainsi, nous pouvons calculer une borne supérieure pour r :

$$r \leq r_{\max} = R_K \sqrt{1 - \left(\frac{C}{n}\right)^{\frac{1}{\alpha}}}. \quad (3.1)$$

Nous pouvons donc considérer que le volume englobant de chaque *metaball* i se limite à une sphère de rayon $r_i = \min(R_i, r_{\max})$ lors du calcul du volume englobant d'un groupe de *metaballs*.

Suppression des primitives de split inutiles

Les *metaballs de split* d'un noeud de la *FBVH* sont les primitives dont le support intersecte la boîte englobante de ce noeud. Cependant, certaines de ces primitives peuvent

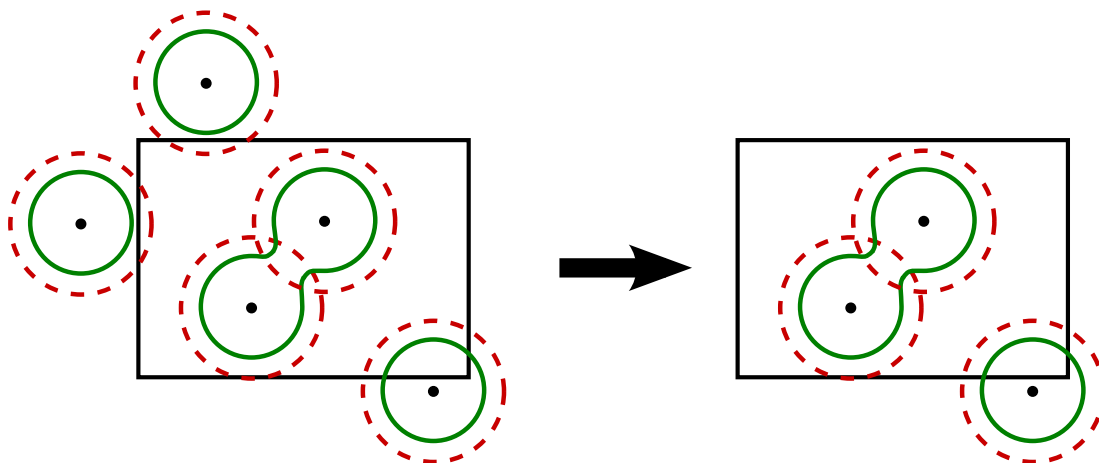


FIGURE 3.3 – Certaines *metaballs de split* dont le support (en rouge) intersecte un noeud de génèrent pas de surface (en vert) qui intersecte ce noeud. Il est donc inutile de les stocker dans ce noeud. En revanche, il est nécessaire de stocker toute *metaball* qui contribue à la surface englobée par ce noeud.

ne pas agir sur l'isosurface englobée par le noeud. C'est le cas si pour tout point x du support appartenant au noeud, le champ potentiel $f(x)$ est inférieur à l'isovaleur T (cf. Figure 3.3). Il est donc possible d'ignorer ces primitives lors de l'évaluation du champ potentiel, ce qui permet d'accélérer son évaluation. Concrètement, il suffit de les retirer de la liste des *metaballs de split* du noeud pour obtenir le résultat souhaité.

L'identification de toutes les primitives de split pouvant être supprimées de cette manière est difficile en raison de leur nature implicite. Cependant, il est possible de réaliser des tests conservatifs permettant d'en éliminer la plupart. Pour toute primitive de split i nous vérifions :

- que la sphère centrée à p_i et de rayon r_{\max} (cf. Equation 3.1) n'intersecte pas la boîte englobante du noeud.
- que pour toute autre *metaball* j du noeud,

$$\| p_i - p_j \| \geq R_i + r_{\max}.$$

Ceci garantit que la *metaball* i n'influence aucune surface générée par une autre *metaball* du noeud.

Si ces deux tests sont réussis, cela signifie que la primitive i peut être retirée de l'ensemble des *primitives de split* du noeud sans aucune conséquence sur la visualisation de l'isosurface.

3.2 Implémentation et Résultats

3.2.1 Implémentation

Nous avons écrit notre implémentation en langage C++ et en CUDA. La construction de la *FBVH* est réalisée sur CPU tandis que la traversée des rayons dans la structure et les tests d'intersections sont réalisés sur le GPU. Ceci nous permet de calculer la *FBVH* correspondant à l'image suivante de l'animation pendant que le rendu de l'image courante est calculé sur le GPU. La durée de construction étant généralement inférieure à la durée du calcul du lancer de rayons, cela permet de masquer son coût de calcul.

Pour bénéficier du cache mémoire, les positions des *metaballs* et les données des noeuds et des feuilles sont stockés sur le GPU en mémoire texture. La traversée de la *FBVH* sur le GPU est réalisée en utilisant une pile stockée en mémoire locale. L'animation de *metaballs* est réalisée sur GPU, puis transférée en mémoire locale. Tous les transferts mémoires entre CPU et GPU sont masqués par les temps d'exécution des noyaux de calcul CUDA, grâce à l'utilisation du *streaming* (cf Figure 3.4).

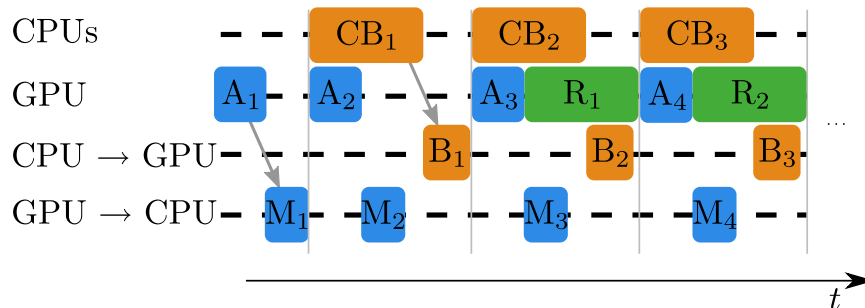


FIGURE 3.4 – Schéma de fonctionnement. L'animation des *metaballs* A_i est réalisée sur le GPU puis transférée sur la machine hôte (M_i). Celle-ci calcule alors la structure d'accélération correspondant à la position actuelle des *metaballs* (CB_i), puis transfère celle-ci sur la mémoire de la carte graphique (B_i). Le GPU possède alors les données nécessaires à l'exécution du rendu final (R_i), qui est réalisé pendant que le CPU calcule la structure d'accélération correspondant à l'image suivante (CB_{i+1}). La plupart des transferts mémoire sont ainsi masqués par le calcul, grâce à la technique du *streaming*.

Traversée

La traversée d'un rayon dans la *FBVH* n'est pas très différente d'une traversée de BVH. Lorsqu'un rayon intersecte un noeud, nous chargeons les boîtes englobantes des deux noeuds fils puis nous déterminons laquelle est intersectée en premier. Le second noeud intersecté est alors placé en pile, et la traversée se poursuit dans le premier noeud

intersecté. Dès qu’une intersection est détectée dans une feuille, nous arrêtons la traversée de la structure car les intersections éventuelles détectées dans les autres feuilles seraient nécessairement situées plus loin sur le rayon.

Le calcul de l’intersection dans une feuille de la *FBVH* est légèrement différent de celui utilisé dans une BVH présenté en section 3.1.2. Puisque les *metaballs de split* de la feuille courante ne sont pas nécessairement entièrement englobées dans une autre feuille, il est nécessaire de calculer également leur projection sur le rayon courant et d’évaluer le champ potentiel en ces points lors de la détermination de l’intervalle contenant la première intersection.

3.2.2 Résultats

Nos tests ont été réalisés sur un ordinateur équipé d’un processeur Intel Core 2 E6600 (2.66 GHz) et d’une carte graphique Nvidia GTX 280. Nous avons comparé les performances de notre *FBVH* à celles obtenues via l’utilisation d’une BVH classique sur des scènes diverses (cf. Table 3.1). Dans la plupart des scènes, notre *FBVH* permet un rendu bien plus performant, particulièrement sur les scènes contenant des regroupements denses de *metaballs*. Dans ces situations, comme par exemple la scène du bassin, les rendus effectués avec la *FBVH* peuvent être jusqu’à trois fois plus rapides qu’avec la BVH. Cette différence de performances s’amenuise au fur et à mesure que la densité spatiale des *metaballs* diminue, car le nombre de tests redondants effectués avec la BVH diminue de la même façon. Toutefois la *FBVH* reste néanmoins plus rapide dans toutes les configurations que nous avons testées.

Scène	nombre de metaballs	BVH				FBVH			
		temps de construction	fps			temps de construction	fps		
			min	max	avg		min	max	avg
Scattered	30,000	101 ms	4.6	12.4	7.4	79 ms	4.9	12.5	9.4
	100,000	160 ms	1.8	3.9	3.3	150 ms	2.3	9.1	4.3
Tornado	4,000	8.0 ms	3.6	5.4	4.6	5.5 ms	4	14.5	11
	128,000	500 ms	0.6	1.1	1.0	290 ms	1.9	2.6	2.4
Molecule 1	5,440	8.9 ms	3.2	8.6	5.8	9.9 ms	7.5	24.1	17.7
Molecule 2	5,440	8.9 ms	0.5	0.7	0.6	9.9 ms	1.2	1.4	1.3
Bassin	10,000	41 ms	1.5	2.3	1.9	40 ms	4.1	11.0	7.1

TABLE 3.1 – Performances du rendu de scènes variées composées de *metaballs* (cf. Figure 3.5) effectué à l’aide d’une structure d’accélération. La construction de la structure est complètement masquée par le temps de calcul nécessaire au rendu dans la plupart des cas.

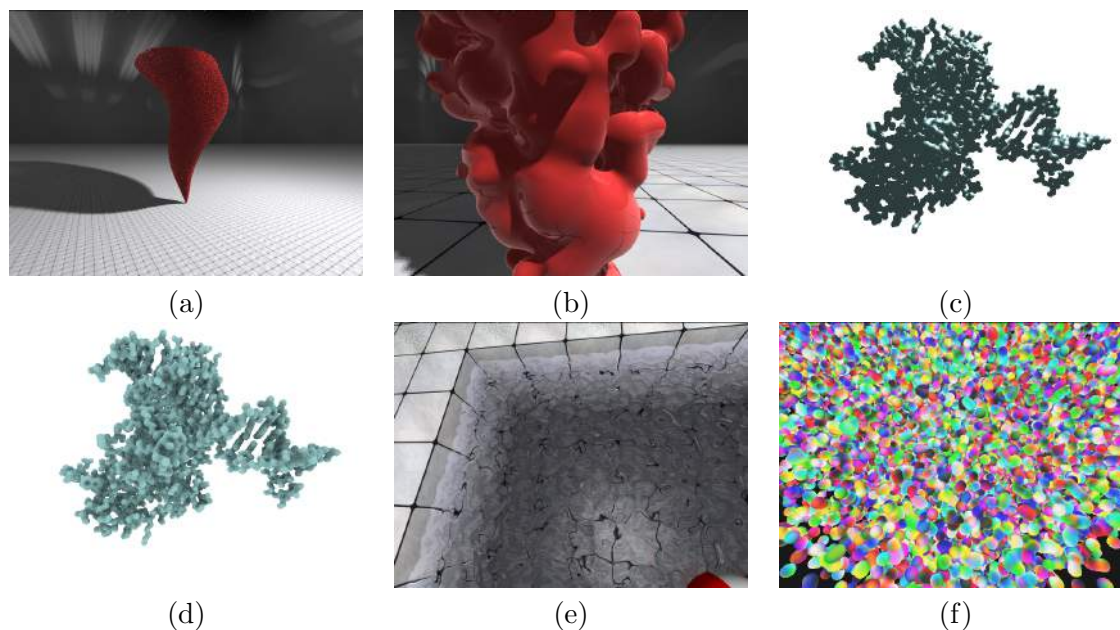


FIGURE 3.5 – Captures d'écran des différentes scènes de test : (a,b) Tornado, scène rendue avec 3 niveaux de réflexions ; (c) Molecule 1, rendue avec l'ombrage de Phong ; (d) Molecule 2, rendue en occlusion ambiante calculée en lançant 64 rayons d'ombre par pixel ; (e) Bassin, rendue avec simulation de la réflexion et de la réfraction de Fresnel ; (f) Scattered metaballs, rendue purement en lancer de rayons primaires (pas d'ombrage). Le rendu de toutes les scènes a été effectué en résolution 640×480 dans nos tests.

Il est à noter que bien que les durées de construction des deux structures soient du même ordre de grandeur, la *FBVH* est construite légèrement plus rapidement dans la plupart des cas. Cela peut sembler surprenant car on pourrait s'attendre à ce que l'étape de redimensionnement des noeuds introduise des calculs supplémentaires pénalisants. En réalité, le fait de réduire la taille des boîtes englobantes a pour effet de diminuer le nombre de *metaballs de split* des noeuds, ce qui accélère le temps de construction.

Nous avons mesuré le nombre moyen de noeuds traversés, tests d'intersection effectués et primitives utilisées dans les tests d'intersections par rayon (cf. Table 3.2). Comme on peut le voir, la *FBVH* permet d'éliminer efficacement les noeuds et primitives inutiles lors des tests d'intersection.

Globalement, on peut constater que l'utilisation d'une *SAH* en $\mathcal{O}(n^2)$ conduit à de bien meilleures performances que la *SAH* standard en $\mathcal{O}(n)$. Dans nos tests, la scène du bassin est 40% plus performante lorsque la *FBVH* est construite en utilisant notre *SAH*. Celle-ci permet en effet de réaliser des arbres plus équilibrés qui sont mieux adaptés à

	Nombre de traversées	Nombre de tests d'intersection	Nombre de metaballs prises en compte
BVH	9.3	1.2	83.5
FBVH	6.2	0.5	35.1

TABLE 3.2 – Nombre moyen de noeuds traversés, de tests d'intersection effectués, et de *metaballs* considérées lors des tests d'intersection par rayon sur la scène Tornado.

notre test d'intersection.

SAH	n	n^2
valeur moyenne de fps (FBVH sur scène du bassin)	5.1	7.1

TABLE 3.3 – Evaluation de l'intérêt de la nouvelle heuristique du coût d'intersection proposée en Section 3.1.3.

Nous avons également mesuré les effets de l'étape de redimensionnement des boîtes englobantes des noeuds présentée en Section 3.1.3 sur les performances lors de la visualisation. Pour ce faire, nous avons simplement omis cette étape lors de la construction de la *FBVH*, de sorte que la boîte englobante de chaque noeud fils fut obtenue en divisant la boîte englobante du noeud père au niveau du plan de séparation. La structure est alors similaire à un *kd-tree*. Sa traversée est cependant plus coûteuse car nous testons l'intersection avec les boîtes englobantes des noeuds au lieu de tester l'intersection avec les seuls plans de séparation. Néanmoins, les performances obtenues avec un *kd-tree* classique devraient être similaires à cette situation car le temps de traversée est faible face à la durée des tests d'intersection.

Scène	Sans l'étape de redimensionnement		Avec l'étape de redimensionnement	
	temps de construction	fps moyen	temps de construction	fps moyen
Molecule 1	8.7 ms	12.7	9.2 ms	14.6
Bassin	35 ms	3.2	37 ms	6.5

TABLE 3.4 – Bénéfices apportés par l'étape de redimensionnement des noeuds durant la construction de la *FBVH*.

Comme on peut le voir en Table 3.4, le nombre de tests inutiles éliminés par l'étape de redimensionnement conduit à de bien meilleures performances. Ce gain de performances augmente avec la densité de *metaballs* puisque le coût de calcul d'une intersection dans une feuille croît avec le nombre de primitives dans cette feuille. Les gains obtenus peuvent être de l'ordre de 100%. L'étape de redimensionnement est donc primordiale, d'autant plus qu'elle n'augmente pas particulièrement le coût de construction de la *FBVH*.

Scène	Optimisations (valeur moyenne de fps)			
	Aucune	Calcul des boîtes englobantes	Supression des metaballs de split inutiles	Les deux
Molecule 1	14.6	15.3	16.7	17.7
Pool	6.5	6.7	6.8	7.1

TABLE 3.5 – Bénéfices de l'utilisation des optimisations présentées en Section 3.1.4.

Les optimisations présentées en Section 3.1.4 apportent également une amélioration notable des performances (cf. Table 3.5). Cependant leur impact diminue lorsque la densité des primitives augmente. Ceci est logique compte tenu du fait que la valeur de r_{\max} augmente avec le nombre de *metaballs* de chaque noeud. Ainsi, il peut être préférable de ne pas les réaliser si les feuilles contiennent plus d'une douzaine de *metaballs*, étant donné que leur coût dans la construction de la *FBVH* augmente avec le nombre de primitives dans chaque noeud, en particulier le test d'élimination des *metaballs de split* inutiles.

3.3 Conclusion

Nous avons mis en place une nouvelle structure d'accélération pour le rendu de *metaballs*, compatible avec les effets lumineux avancés (ombres réflexions, etc...). L'augmentation des performances qu'elle apporte permet de visualiser sur GPU des centaines de milliers de *metaballs* à des fréquences de rafraîchissement interactives, voire temps réel. Cette *FBVH* permet des rendus bien plus rapides qu'une BVH classique dans le cas où la répartition des primitives est dense, tout en étant rapidement construite sur CPU de façon gloutonne par l'intermédiaire de notre *SAH*.

Néanmoins, le temps de construction de la *FBVH* peut devenir problématique dans le cas d'ensembles extrêmement denses de primitives, en raison du nombre important de *primitives de split*. Dans cette situation, pour les premiers niveaux de l'arbre, il est possible de ne pas réaliser de *binning* pour déterminer le plan de séparation, mais tout simplement de diviser les noeuds à la moitié de leur plus grand côté. Cela n'affecte que peu la qualité de l'arbre obtenu, mais accélère fortement sa construction.

Un autre problème est le coût des tests d'intersection dans les feuilles qui contiennent beaucoup de *metaballs* (plus de 50). A nouveau, ces situations se produisent lorsque la densité de primitives est extrêmement dense. L'isosurface est alors bien souvent située à l'extérieur de ces feuilles. Il serait intéressant de développer des approximations pour simplifier l'évaluation du champ potentiel à l'intérieur de ces noeuds.

4

Opérateurs de mélange

4.1 Définition

Le chapitre précédent nous a donné un aperçu de l'intérêt des surfaces implicites en modélisation : une surface complexe lisse peut être obtenue par l'agrégation d'un ensemble discret de primitives implicites. Les primitives que nous avons utilisées étaient des *meta-balls* et leur composition était simplement obtenue par la somme de leurs champs potentiels respectifs. L'opérateur de composition *somme* est celui utilisé par Blinn [Bli82], mais il est possible d'utiliser d'autres opérateurs de mélange afin d'obtenir diverses transitions entre les primitives. L'opérateur g_m de Ricci [Ric73] permet par exemple de généraliser l'opérateur de Blinn :

$$g_m(f_1, \dots, f_n) = (f_1^m + \dots + f_n^m)^{\frac{1}{m}}$$

Le paramètre m permet de contrôler la taille du mélange et l'on retrouve l'opérateur de Blinn en prenant $m = 1$. Ces opérateurs sont suffisants pour la modélisation de surfaces de type fluides mais pas pour la modélisation géométrique en général. Par exemple, la modélisation d'une arête franche entre deux primitives nécessite de pouvoir introduire des discontinuités d'ordre C^0 sur l'isosurface. Autrement dit, l'opérateur doit lui-même présenter une discontinuité de ce même ordre. Les opérateurs max et min de Sabin [Sab68] possèdent cette propriété et réalisent respectivement l'union et l'intersection booléenne d'un ensemble de primitives implicites :

$$g_{max}(f_1, \dots, f_n) = \max(f_1, \dots, f_n)$$

$$g_{min}(f_1, \dots, f_n) = \min(f_1, \dots, f_n)$$

Tous ces opérateurs sont des opérateurs *n-aires*, ce qui signifie que le résultat de la

composition ne change pas quel que soit l'ordre des paramètres f_1, \dots, f_n . Une autre manière de composer des primitives implicites consiste à définir un *arbre de composition* (CSG).

Définition 7 Un arbre de composition est un arbre binaire dont les feuilles sont les primitives implicites de base et les noeuds sont des opérateurs de composition binaires.

Les *arbres de composition* permettent ainsi de définir des objets complexes de manière hiérarchique : chaque noeud de l'arbre correspond au mélange de deux sous-parties de l'objet final.

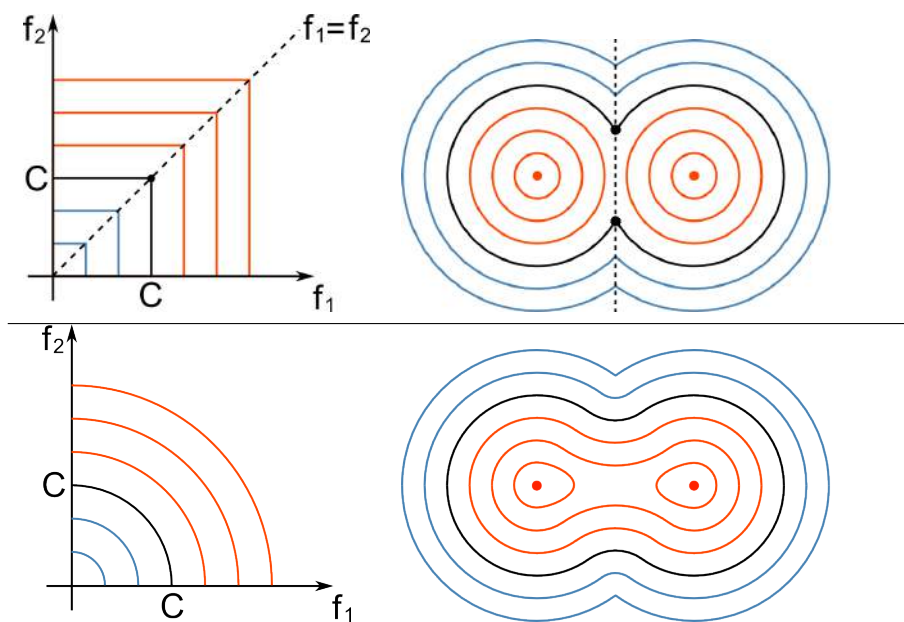


FIGURE 4.1 – Exemples d'opérateurs binaires de composition. En haut : opérateur d'union franche obtenu par la fonction $(f_1, f_2) \rightarrow \max(f_1, f_2)$. En bas : opérateur de mélange obtenu par la fonction $(f_1, f_2) \rightarrow \sqrt{f_1^2 + f_2^2}$. Sur chaque ligne, la figure de gauche représente les lignes de champ de l'opérateur sur le plan (f_1, f_2) et la figure de droite les lignes de champ obtenues en faisant la composition de deux *metaballs*. Notez la corrélation entre les lignes de champ sur les figures de gauche et de droite.

La forme générale d'un opérateur de composition binaire est $g(f_1, f_2)$, f_1 et f_2 étant les champs potentiels respectifs des primitives implicites à composer. Une manière graphique de visualiser un tel opérateur consiste à représenter ses lignes de champ sur un plan, l'axe des abscisses représentant les valeurs de f_1 et celui des ordonnées les valeurs de f_2 , comme illustré en Figure 4.1. On peut remarquer sur cette figure la correspondance entre les lignes de champ de l'opérateur de mélange et celles du champ correspondant à la composition de deux primitives implicites. En particulier, les lignes bleues, qui correspondent aux valeurs

de champ inférieures à l'isovaleur C , sont situées à l'extérieur de l'isosurface puisque le champ potentiel des primitives à support compact décroît lorsque l'on s'approche de la limite du support. À l'inverse, les lignes rouges correspondant aux valeurs de champ supérieures à C sont à l'intérieur de l'isosurface. La discontinuité de la différentielle de l'opérateur \max en $f_1 = f_2$ est bien visible sur le champ potentiel résultant, notamment sur l'isosurface (points en noir) : c'est ce qui est à l'origine de la formation d'une arête franche (cf. Figure 4.2 (a)). La Figure 4.2 illustre d'autres opérateurs de composition sur des cylindres implicites en 3 dimensions.

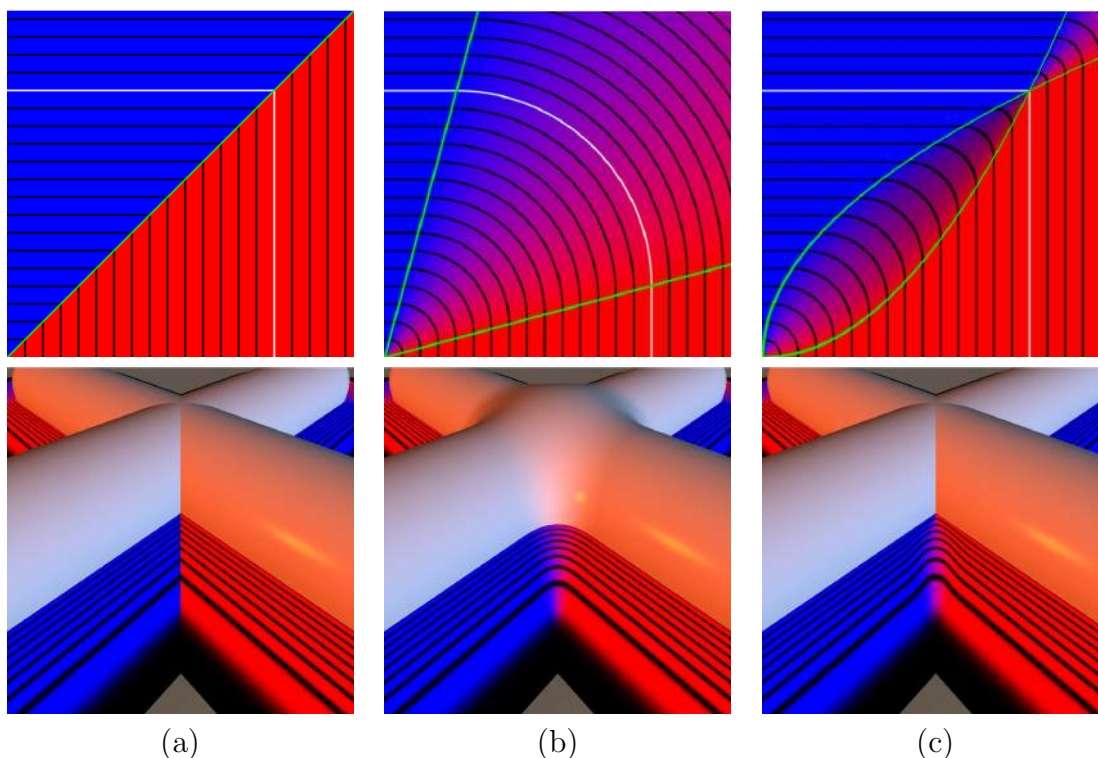


FIGURE 4.2 – Trois opérateurs de composition binaires g et leur application sur deux cylindres implicites positionnés en croix. En première ligne, le tracé des opérateurs où les axes représentent respectivement les valeurs des champs potentiels en entrée f_1 et f_2 . Les lignes noires représentent certaines iso-courbes de l'opérateur et la ligne blanche l'iso-courbe correspondant à la valeur C de l'isosurface observée. Le gradient de l'opérateur est représenté en rouge et bleu. La surface résultante, définie par $f = g(f_1, f_2) = C$ est visualisée en deuxième ligne. Les opérateurs représentés sont : (a) une union franche standard (\max), (b) un mélange lisse [BMDS02], (c) une union franche dont le champ potentiel est lisse en dehors de l'isosurface [BDS⁺02]. Les lignes vertes sur les opérateurs (b) et (c) représentent les frontières au delà desquelles $g(f_1, f_2) = \max(f_1, f_2)$.

La forme des opérateurs de composition est dépendante de la nature des primitives implicites. Pour les primitives à support global d'isovaleur 0, les compositions booléennes peuvent être obtenues par l'intermédiaire des R -*functions*, c'est-à-dire des fonctions

dont le signe ne dépend que des signes de leurs paramètres. Par exemple, l'opérateur d'union de Pasko [PASS95] qui réalise l'union franche de deux primitives implicites à support global est une R -*function*. Son expression est :

$$g(f_1, f_2) = f_1 + f_2 - \sqrt{f_1^2 + f_2^2}$$

L'isosurface résultante est la même que celle produite par l'opérateur *max* développé par Ricci [Ric73]. Cependant, hors de l'isosurface, le champ potentiel résultant est très différent : celui produit par l'opérateur de Pasko est de continuité C^∞ tandis que celui produit par l'opérateur de Ricci possède la même discontinuité C^0 partout où $f_1 = f_2$.

L'opérateur de Pasko [PASS95] est un opérateur d'*union propre* (de l'anglais *clean union*) : le champ potentiel généré par cet opérateur reste lisse en dehors de l'isosurface. Barthe *et al.*[BDS⁺03] ont également développé un opérateur d'*union propre* sur les primitives à support global dont l'aspect est configurable en dehors de l'isosurface.

Les opérateurs de composition peuvent également être utilisés pour créer des transitions douces entre les primitives. L'opérateur de mélange superelliptique introduit par Rockwood [Roc89] :

$$g(f_1, f_2) = 1 - \left[\frac{1 - f_1}{r_1} \right]_+^t - \left[\frac{1 - f_2}{r_2} \right]_+^t$$

où $[*]_+ = \max(0, *)$ et les paramètres r_1 , r_2 et t permettent de contrôler la forme du mélange. En modifiant l'expression de son opérateur d'union franche, Pasko [PASS95] a également produit un opérateur de mélange dont la forme est paramétrable :

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

Les trois paramètres a_0 , a_1 et a_2 sont les paramètres de contrôle du mélange.

Pour créer des opérateurs de composition de primitives à support compact, il est possible d'adapter les opérateurs spécifiques aux primitives à support global. Mais ceci peut avoir pour effet de déformer le champ potentiel à l'extérieur des supports des primitives, ce qui pose problème dans le cas de mélanges successifs. Les opérateurs de composition des primitives à support compact doivent respecter certaines propriétés de façon à ce que le mélange des primitives ne perturbe pas le champ potentiel situé en dehors des zones d'influence des primitives. En particulier tout opérateur de composition g doit vérifier la propriété suivante :

$$\forall (f_1, f_2) \in \mathbb{R}^2, g(f_1, 0) = f_1 \text{ et } g(0, f_2) = f_2$$

Barthe *et al.*[BWd04] proposent un opérateur de mélange réalisant une transition douce entre les primitives dont la taille est paramétrable. Pour cela, les lignes de champ issues de f_1 et celles issues de f_2 sont reliées par des arcs d'ellipses. La taille des arcs est déterminée par deux angles θ_1 et θ_2 (cf. Figure 4.2(b)). La continuité du champ potentiel résultant est G^1 .

4.2 Problèmes majeurs de la modélisation par surfaces implicites

L'utilisation de surfaces implicites associées à des arbres de composition semble être une alternative intéressante en modélisation statique, où les représentations par maillage ou paramétriques dominent. Il semble en effet intuitif de modéliser un objet de manière hiérarchique, en commençant par définir la forme générale de manière grossière puis en précisant localement chaque détail. Malheureusement, outre les difficultés de paramétrisation et de visualisation des surfaces générées par des primitives implicites, une critique majeure lancée envers ce type de représentation est qu'elles sont limitées à la création d'objets aux contours lisses, voire indistincts. La raison de cette critique est principalement liée aux opérateurs de composition utilisés jusqu'à aujourd'hui (présentés plus haut). En effet, ceux-ci souffrent tous de défauts qui rendent malaisée la modélisation géométrique par objets implicites :

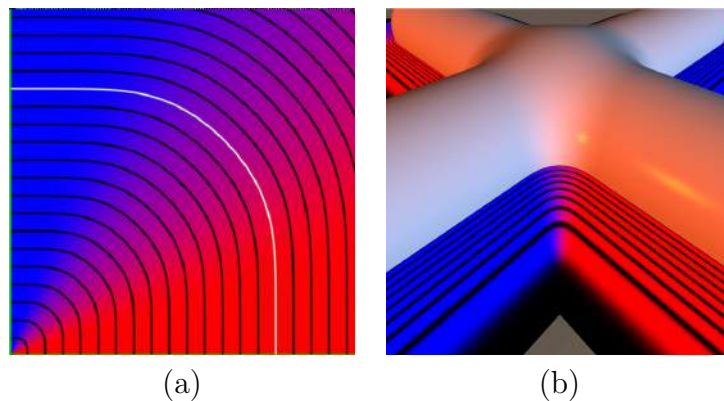


FIGURE 4.3 – Illustration du gonflement indésirable. Lors de l'utilisation d'un opérateur de mélange lisse classique (a), la surface résultante s'éloigne des surfaces initiales provoquant un gonflement. Celui-ci est particulièrement visible lorsque les primitives mélangées sont de même taille, comme le montre l'exemple des deux cylindres ci-dessus (b).

1. Le mélange de deux surfaces implicites peut provoquer des gonflement indésirables. Par exemple, une croix formée par le mélange de deux cylindres implicites situés dans le même plan sera nécessairement plus épaisse au niveau de l'intersection. En effet, une fonction de mélange lisse g est telle que $g(f_1, f_2) \geq \max(f_1, f_2)$, $\forall(f_1, f_2)$ (cf. Figure 4.3 (a)). En particulier, si les deux isosurfaces sont tangentes, i.e. $f_1 = f_2 = C$, $g(C, C) \geq C$ et ainsi la surface du mélange s'éloigne des surfaces initiales (cf. Figure 4.3 (b) et Figure 4.4 (a)).
2. Le mélange de surfaces implicites peut se produire même si les surfaces ne sont pas en contact. Ce problème du *mélange à distance* correspond en fait à la situation symétrique du problème précédent. Dans le domaine de la modélisation géométrique, ceci est un défaut rédhibitoire car il est alors impossible d'approcher deux sous-parties d'un même objet sans que celles-ci ne fusionnent. En animation, il est également souhaitable que le mélange de deux objets ne se produise pas avant que ceux-ci ne soient en contact, par exemple lors de la visualisation de gouttes d'eau tombant dans un réservoir (cf Figure 4.4 (b)).
3. Du fait du mélange à distance, l'augmentation du volume provoquée par les opérateurs de composition classiques peut boucher un orifice souhaité par le concepteur de l'objet, modifiant ainsi la topologie désirée (cf Figure 4.4 (c)).
4. Enfin les primitives implicites de petite taille subissent généralement de très fortes déformations lorsqu'elles sont mélangées à des primitives bien plus grandes, ce qui provoque l'estompement des détails modélisés en surface (cf Figure 4.4 (d)).

4.3 Travaux antérieurs

Quelques travaux réalisés précédemment ont identifié et tenté de résoudre les problèmes que nous avons recensés. Aucun cependant n'a permis de résoudre la totalité de ces problèmes de façon générale.

Le problème du gonflement indésirable a tout d'abord été abordé par Rockwood [Roc89] dans le domaine des primitives implicites construites à partir d'un squelette. Celui-ci a proposé de modifier l'opérateur de mélange superelliptique afin de paramétrer le rayon d'influence des primitives par le cosinus de l'angle entre les gradients des champs potentiels. Le champ potentiel résultant est de la forme :

$$f(p) = 1 - \max \left(0, 1 - \frac{d(p, S_1)}{r_1(1 - \cos\theta)} \right)^t - \max \left(0, 1 - \frac{d(p, S_2)}{r_2(1 - \cos\theta)} \right)^t$$

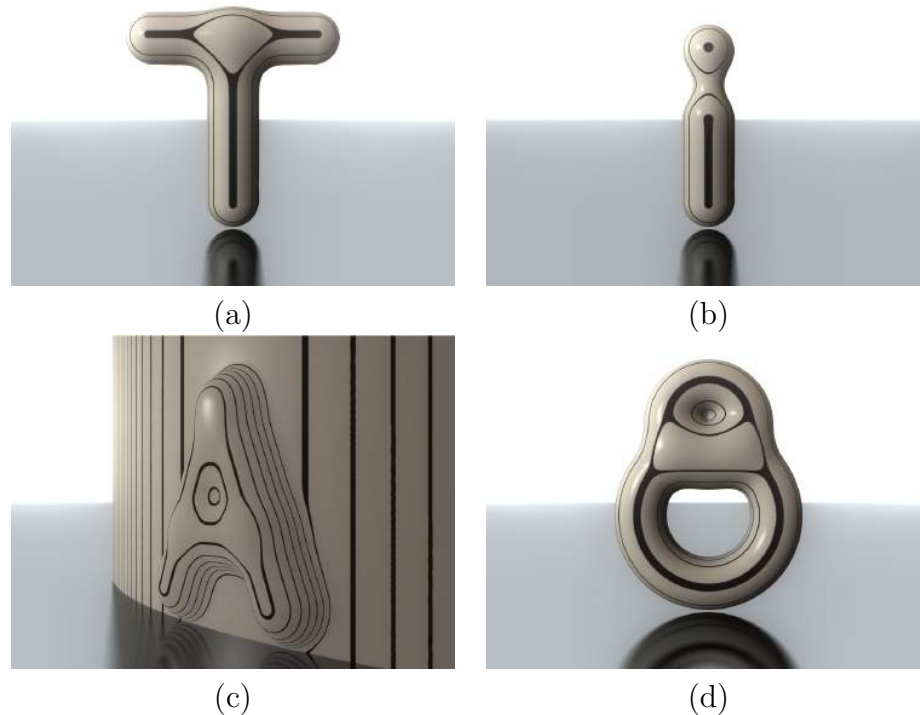


FIGURE 4.4 – Illustration des problèmes communs en composition de surfaces implicites. (a) Gonflement indésirable. (b) Mélange à distance. (c) Déformation des détails. (d) Changements indésirables de topologie.

où S_1 et S_2 sont les squelettes respectifs des primitives à mélanger, r_1 et r_2 leurs rayons d'influence, $d(p, S)$ la distance algébrique du point p au squelette S et θ l'angle entre les gradients des champs potentiels. Cet opérateur permet de résoudre effectivement le problème du gonflement indésirable pour le type de primitives étudié, mais pas celui du mélange à distance.

Bloomenthal [Blo97a] a étudié des solutions au problème du gonflement indésirable dans le cadre d'opérateurs de composition n -aires de primitives de convolution. Celles-ci interpolent une composition d'union franche et un mélange lisse, paramétrée par la position du point d'évaluation relativement aux squelettes. Cette technique permet de résoudre le problème du gonflement indésirable mais provoque un léger creux au niveau de la surface joignant les primitives initiales.

Le problème du mélange à distance a été traité plus récemment. La solution proposée par Bernhardt *et al.* [BBCW10] s'inspire du travail de Pasko *et al.* [PPK05] sur les mélanges localisés de surfaces implicites. Ces méthodes consistent à introduire une primitive implicite supplémentaire servant à délimiter la région dans laquelle le mélange doit s'effectuer.

La valeur du champ potentiel de cette primitive est ainsi utilisée en tant que facteur d'interpolation de deux compositions, une union franche et un mélange. Cette primitive implicite est construite de manière automatique, par détection de la ligne d'intersection des surfaces initiales, celle-ci étant ensuite utilisée comme squelette de la primitive. Le rayon d'influence de cette primitive est paramétré de façon à réduire l'estompement des petites primitives.

La méthode de Bernhardt *et al.* souffre cependant de plusieurs défauts. Les champs potentiels produits sont de continuité G^1 uniquement, ce qui peut entraîner des artefacts lors de la visualisation de réflexions sur les surfaces (cf. Figure 4.5). De plus, la technique d'extraction de la ligne d'intersection employée nécessite d'échantillonner celle-ci à partir des surfaces initiales, ce qui est relativement coûteux et rend difficile le passage à l'échelle de la solution, notamment pour les applications de modélisation où des ajouts consécutifs de plusieurs primitives risquent de générer de nombreuses intersections successives. Enfin cette technique ne résout pas le problème du gonflement indésirable et il serait difficile de l'adapter en ce sens. En effet, cela nécessiterait de construire des régions de mélange de forme complexe de sorte que celui-ci ne se produise qu'au niveau des zones de concavité entre les surfaces.

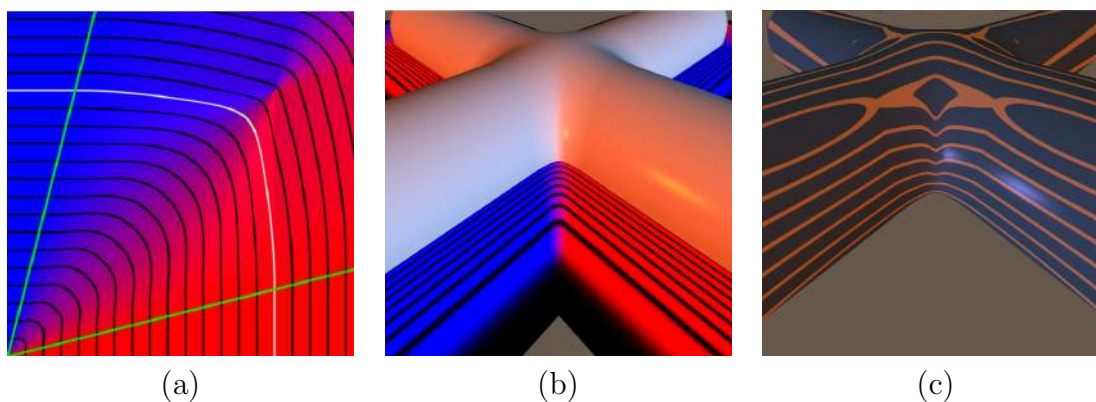


FIGURE 4.5 – L'opérateur de Bernhardt *et al.* interpole linéairement une union franche et un mélange lisse. En position intermédiaire, les isocourbes de cet opérateur présentent des distortions visibles en (a). La surface résultante représentée en (b) lors du mélange de deux cylindres est uniquement G^1 , ce qui entraîne des artefacts visuels lors de l'affichage de réflexions sur la surface (c).

Aucun des opérateurs de composition développés jusqu'à présent ne permet de résoudre la totalité des problèmes présentés en introduction. Nous avons construit une famille d'opérateurs de composition résolvant ces quatre problèmes de manière efficace. Contrairement à [PPK05, BBCW10], ils ne nécessitent pas l'introduction de primitives implicites supplémentaires. Le chapitre suivant présente et explique le fonctionnement de ces

opérateurs.

5

Mélange contrôlé de surfaces implicites

Notre famille d'opérateurs de composition repose sur l'idée de Rockwood consistant à utiliser l'angle entre les gradients des primitives pour contrôler l'importance du mélange. Ainsi, il est nécessaire que ces opérateurs soient paramétrables de manière à pouvoir passer de façon continue d'une transition franche à un mélange lisse entre les surfaces. Ce paramètre d'*importance du mélange* offre de nouvelles perspectives dans les systèmes de modélisations par surfaces implicites, augmentant la variété des transitions possibles entre les surfaces. Nous utilisons des fonctions de l'angle entre les gradients des primitives, que nous appelons *contrôleurs*, pour fixer la valeur de ce paramètre. Les *contrôleurs* sont des nouveaux outils de modélisation permettant de configurer les opérateurs de composition afin d'obtenir la transition souhaitée. Dans la suite du chapitre, nous montrons ainsi qu'un *contrôleur* judicieusement choisi permet de résoudre la totalité des problèmes majeurs en modélisation implicite constructive. Il est également possible de paramétrer un *contrôleur* de façon à créer localement une union franche et un mélange lisse au sein d'une même transition.

Notre méthode est générique et s'adapte à toutes les surfaces implicites représentées de manière fonctionnelle. Par la suite, nous ne considérerons que les primitives implicites à support local de la forme $f(p) = C$, f étant une fonction potentielle à support compact. Pour simplifier l'expression et la représentation de nos opérateurs nous fixons l'isosurface $C = \frac{1}{2}$, ce qui est la valeur habituelle pour ce type de primitives (par exemple, les *meta-balls*). Ces primitives sont les plus difficiles à manipuler car le résultat de leur composition doit conserver un support compact. Néanmoins, notre technique s'adapte aisément aux autres primitives implicites, comme illustré en Section 5.2 où les mêmes opérateurs sont utilisés sur des primitives de convolution à support global.

Ce chapitre est organisé de la façon suivante. Il est tout d'abord expliqué comment le gradient des champs potentiels peut être utilisé pour résoudre les problèmes liés au mélange des surfaces implicites. L'utilisation du gradient introduit des variations de continuité dans les champs potentiels qui sont discutées ensuite. Pour résoudre ces problèmes de continuité, des opérateurs de composition de classe C^∞ sont développés ainsi qu'une technique efficace d'évaluation sur GPU. Des exemples d'opérateurs contrôlés par le gradient sont présentés en fin de chapitre. Le premier est un opérateur de mélange lisse qui ne souffre pas des problèmes de mélange classiques. Le second est un opérateur qui forme une arête franche au niveau de la transition de deux objets et qui peut être utilisé pour simuler de manière automatique le contact d'objets élastiques ou mous.

5.1 Contrôle du mélange par le gradient

Comme indiqué précédemment, un opérateur de composition permettant de passer localement d'une union franche à une transition lisse est nécessaire pour définir un environnement de travail dans lequel les problèmes de mélange des surfaces implicites sont résolus. Formellement, cela signifie que cet opérateur est de la forme

$$f(p) = g(f_1(p), f_2(p), h(p)) \quad (5.1)$$

où $h \in \mathbb{R}$ est un paramètre variant spatialement et qui définit le degré de transition entre un opérateur d'union franche et l'opérateur de transition douce. Nous appelons ce paramètre un *contrôleur* de l'opérateur f :

Définition 8 *Un contrôleur h d'un opérateur g est une fonction continue de $\mathbb{R}^3 \rightarrow [0; 1]$. Celle-ci indique en chaque point le degré d'interpolation de deux opérateurs $O_0 : \mathbb{R}^2 \rightarrow \mathbb{R}$ et $O_1 : \mathbb{R}^2 \rightarrow \mathbb{R}$ tels que $\forall (f_1, f_2) \in \mathbb{R}^2$:*

$$O_0(f_1, f_2) = g(f_1, f_2, 0) \text{ et } O_1(f_1, f_2) = g(f_1, f_2, 1)$$

Dans le cas d'un *contrôleur* constant, l'opérateur g réalise ainsi un mélange radial classique (cf Figure 5.1(a)). Par la suite, on rendra l'opérateur g indissociable de son contrôleur h et noterons simplement :

$$g(f_1, f_2, h) = g(f_1, f_2)$$

Nous proposons de définir h comme une fonction de l'angle α entre les gradients des champs potentiels f_1 et f_2 passés en paramètre. Ainsi, le *contrôleur* $h : [0; \pi] \rightarrow [0; 1]$ est

défini par :

$$h(p) = h(\alpha) \text{ où } \alpha = \widehat{\nabla f_1(p), \nabla f_2(p)}.$$

En choisissant des opérateurs O_0 et O_1 définissant respectivement une transition douce et une union franche, il devient aisé de résoudre les problèmes de mélange des surfaces implicites.

Le problème du gonflement non désiré est provoqué par l'utilisation d'un opérateur de mélange réalisant une transition douce entre deux primitives aux endroits où leurs surfaces sont alignées, comme illustré sur la Figure 5.1(a). Pour supprimer ce gonflement, il faut limiter l'amplitude de la transition lorsque les gradients des deux surfaces sont proches en terme de direction, autrement dit lorsque $\alpha \approx 0$. Au cas extrême $\alpha = 0$, la transition réalisée doit même être une union franche. Pour cela, il suffit de fixer $h(0) = 1$, ainsi $g(f_1, f_2, h(0)) = O_1(f_1, f_2)$.

L'amplitude de la transition doit ensuite augmenter au fur et à mesure que l'angle entre les gradients augmente. Pour des raisons esthétiques, il est préférable que la taille de la transition soit maximale lorsque les surfaces sont orthogonales. Ceci est réalisé en faisant décroître le *contrôleur* h de 1 à 0 quand α varie de 0 à $\pi/2$ (cf. Figure 5.1(b)).

De manière analogue, le problème du mélange à distance est provoqué lorsqu'une transition douce a lieu entre deux surfaces opposées. Ce problème peut être résolu en faisant croître le *contrôleur* h de 0 à 1 lorsque α varie de $\pi/2$ à π . Ainsi une union franche est réalisée lorsque les surfaces se font face (cf. Figure 5.1(b)).

Le problème revient donc à définir des opérateurs contrôlables réalisant l'interpolation entre transition franche et transition douce et des contrôleurs adaptés aux types de surfaces implicites sous-jacents. Avant cela, nous présentons les problèmes de continuités inhérents à l'expression de notre opérateur g (cf. Equation 5.1).

5.1.1 Continuité de l'opérateur

Nos *contrôleurs* dépendent des gradients des champs potentiels initiaux f_1 et f_2 . Ainsi, si h et g sont de continuité suffisante, la continuité de $f = g(f_1, f_2, h)$ est soit celle de ∇f_1 , soit celle de ∇f_2 (il s'agit de la plus petite des deux). Dans ce cas, et à l'exception des cas triviaux tels qu'un *contrôleur* constant, nous perdons un ordre de continuité. Toutefois, cette perte est en général très locale car les fonctions définissant les champs potentiels initiaux sont généralement de classe \mathcal{C}^∞ par morceaux. Pour empêcher toute autre perte

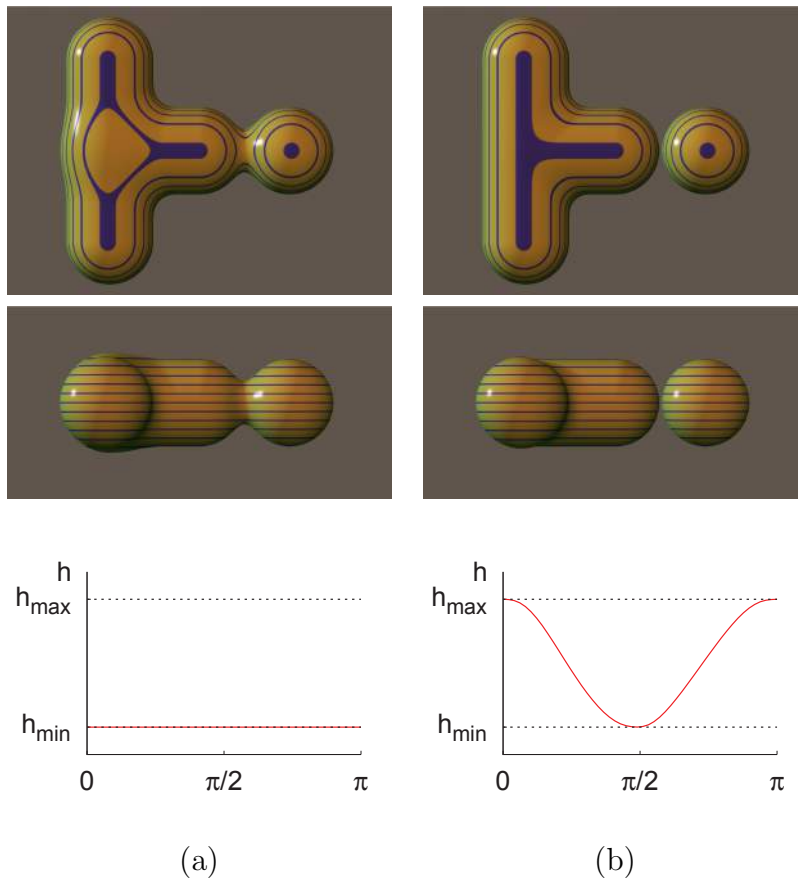


FIGURE 5.1 – (a) Mélange “radial” standard provoquant le gonflement indésirable et le mélange à distance. (b) L’amplitude de la transition est paramétrée par le *contrôleur* tracé en troisième ligne. Au dessus du point d’intersection des deux cylindres, l’angle entre les gradients $\alpha = 0$ et la transition est alors une union franche ($h = 1$). Lorsque l’on descend le long de la transition α augmente de 0 à $\pi/2$, le *contrôleur* h décroît de 1 à 0 ce qui augmente la taille de la transition.

de continuité (cf. Figure 5.2), nous avons employé des fonctions de classe \mathcal{C}^∞ pour définir les opérateurs de composition g et les *contrôleurs* h associés qui sont présentés dans les sections suivantes.

5.1.2 Forme générale du contrôleur

La vitesse à laquelle le *contrôleur* varie sur les intervalles $[0; \pi/2]$ et $[\pi/2; \pi]$ doit être paramétrable pour s’adapter aux champs potentiels initiaux. En effet, si celle-ci n’est pas adéquate, des gonflements peuvent apparaître de part et d’autre d’une transition. Ce phénomène est typiquement produit lorsque l’ouverture de la transition est trop rapide dans les zones où α est proche de 0 ou de π (cf. Figure 5.3(b)).

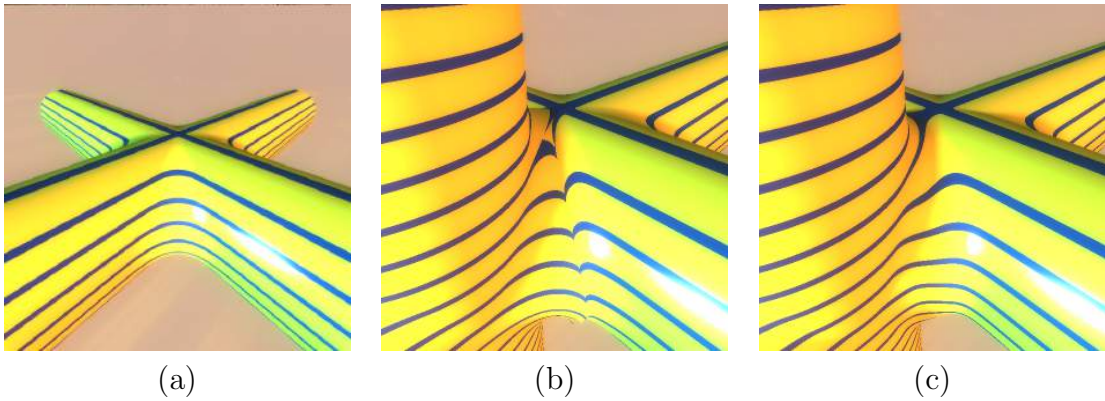


FIGURE 5.2 – (a) Deux cylindres sont composés avec un opérateur de mélange de classe G^1 contrôlé par les gradients. La transition reste lisse malgré la perte de continuité occasionnée. (b) Un troisième cylindre est ajouté au résultat de la composition précédente, à l'aide du même opérateur de mélange. Le champ potentiel résultant perd alors un degré de continuité supplémentaire, qui se traduit par l'apparition d'un arête franche C^0 au sein de la transition. (c) Même scène où les cylindres sont composés avec un opérateur de classe C^∞ qui n'engendre pas de perte de la continuité.

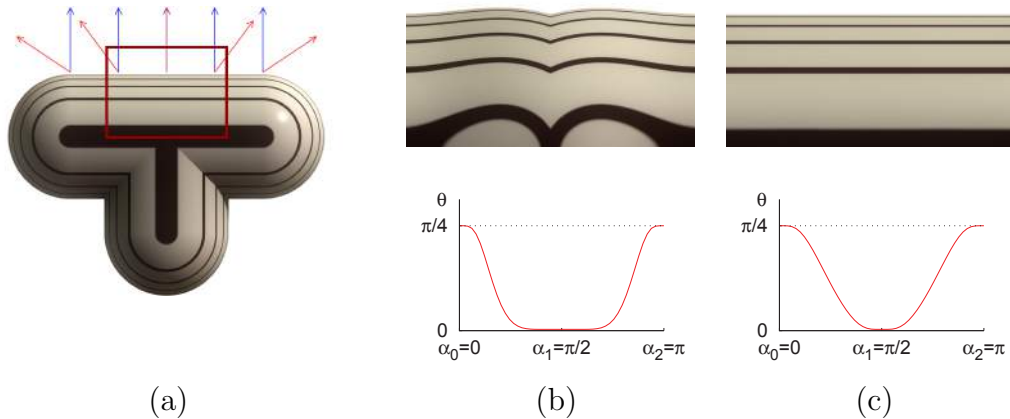


FIGURE 5.3 – Illustration des gonflements non désirés liés à un mauvais paramétrage du *contrôleur*. (a) Les gradients respectifs des champs potentiels des cylindres horizontal et vertical sont représentés en bleu et rouge. (b) Lorsque les paramètres de pente w_0 et w_1 du contrôleur sont trop élevés (ici $w_0 = w_1 = 3$), l'ouverture de la transition devient trop rapide et provoque des gonflements de part et d'autre de la transition. (c) Une valeur adaptée de w_0 et w_1 permet de résoudre le problème (ici $w_0 = w_1 = 1$). Celle-ci est à déterminer en fonction du type de primitive implicite utilisé.

Huit paramètres sont utilisés pour définir nos *contrôleurs*. Ceux-ci sont les trois valeurs d'angle α_i ($i = 0, 1, 2$), les trois valeurs du contrôleur associées $h_i = h(\alpha_i)$ ($i = 0, 1, 2$) et deux paramètres supplémentaires de contrôle de pente w_0 et w_1 . Un contrôleur défini par ces paramètres :

- est constant et égal à h_0 sur $[0, \alpha_0]$
- varie continûment de h_0 à h_1 sur $[\alpha_0, \alpha_1]$
- varie continûment de h_1 à h_2 sur $[\alpha_1, \alpha_2]$
- est constant et égal à h_2 sur $[\alpha_2, 1]$

Les paramètres w_0 et w_1 définissent la vitesse de variation de h respectivement sur $[\alpha_0; \alpha_1]$ et sur $[\alpha_1; \alpha_2]$.

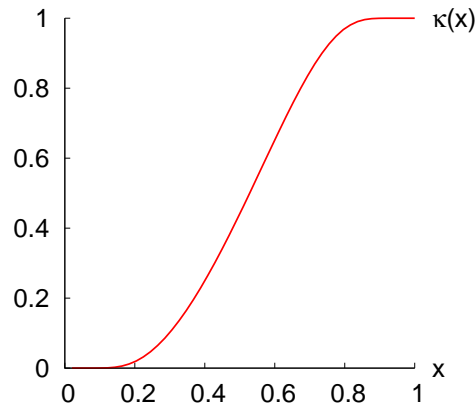


FIGURE 5.4 – Tracé de la fonction κ .

Pour garantir la continuité C^∞ du contrôleur, la valeur de celui-ci sur les intervalles $[\alpha_0; \alpha_1]$ et $[\alpha_1; \alpha_2]$ est obtenue par mise à l'échelle de la fonction $\kappa : [0; 1] \rightarrow [0; 1]$:

$$\kappa(x) = 1 - \eta(1 - \eta(x)) \text{ avec } \eta(x) = e^{1-\frac{1}{x}}$$

Celle-ci est parfaitement plate aux extrémités $x = 0$ et $x = 1$ (toutes ses dérivées sont nulles en ces deux points) (cf. Figure 5.4). L'équation de h devient :

$$h(\alpha) = \begin{cases} h_1 & \alpha \leq \alpha_0 \\ \left(\kappa \left(\frac{\alpha - \alpha_1}{\alpha_0 - \alpha_1} \right) \right)^{w_0} (h_0 - h_1) + h_1 & \text{si } \alpha \in]\alpha_0, \alpha_1[\\ \left(\kappa \left(\frac{\alpha - \alpha_1}{\alpha_2 - \alpha_1} \right) \right)^{w_1} (h_2 - h_1) + h_1 & \text{si } \alpha \in]\alpha_1, \alpha_2[\\ h_2 & \text{sinon} \end{cases}$$

Du fait des propriétés de la fonction κ , tous les *contrôleurs* de cette forme sont de classe C^∞ . En pratique il convient de régler à la main les deux paramètres w_0 et w_1 en fonction de la famille de primitives implicites employée. La forme d'un *contrôleur* est alors définie en fixant la valeur des trois couples $(\alpha_i, h(\alpha_i))$. Des exemples d'applications sont données en Section 5.2.

5.1.3 Opérateurs de composition

Tout comme les *contrôleurs*, les opérateurs de composition g doivent être de classe C^∞ pour minimiser la perte de continuité expliquée en 5.1.1. Nous avons établi une méthode générique de conception des opérateurs. Celle-ci permet de définir de façon intuitive de nouveaux opérateurs de mélange contrôlables tout en respectant ces contraintes de continuité.

Comme expliqué en section 5.1, le *contrôleur* h est utilisé pour faire varier la composition g d'une union franche à une autre transition choisie, par exemple une transition douce. L'interpolation entre ces deux opérateurs est définie par l'intermédiaire d'un angle θ appelé *angle d'ouverture de l'opérateur* g . Lorsque $\theta = 0$, l'opérateur g est dit *ouvert* et correspond à une transition de largeur maximale. Lorsque $\theta = \pi/4$, l'opérateur est dit *fermé* et correspond à une union franche (cf. première rangée de la Figure 5.5). Ainsi, l'angle θ varie en fonction de l'angle α des gradients des champs f_1 et f_2 par l'intermédiaire du *contrôleur* $h : h(\alpha) = \tan(\theta)$.

La variation entre les deux transitions lorsque θ varie est obtenue de la façon suivante. Pour chaque valeur de $\theta \in [0; \pi/4]$, nous définissons les *courbes d'ouverture* $k_\theta(f_1)$ et $k_\theta(f_2)$ de g . Celles-ci délimitent la zone du support de g à l'intérieur de laquelle la transition choisie s'effectue (typiquement, une transition douce), zone visible en gris sur la deuxième rangée de la Figure 5.5. Notez la façon dont la forme des *courbes d'ouverture* varie avec l'angle θ : celles-ci délimitent une zone de transition maximale lorsque $\theta = 0$ qui diminue quand θ augmente puis finit par se cintrer en 0.5 lorsque $\theta = \pi/4$, formant alors un opérateur d'union franche. L'équation des *courbes d'ouverture* est donnée en Annexe A. Elles sont toutes de classe C^∞ (à l'exception de $k_{\pi/4}$ en $\frac{1}{2}$) ce qui permet de garantir la continuité de l'opérateur g .

Les valeurs de g dans la région bornée par les *courbes d'ouverture* k_θ sont définies par l'intermédiaire d'un opérateur $\bar{g} : \mathbb{R}^2 \rightarrow \mathbb{R}$, définissant le type de transition souhaitée. L'équation de g est alors :

$$g(f_1, f_2) = \begin{cases} \max(f_1, f_2) & \text{si } f_1 < k_\theta(f_2) \text{ ou } f_2 < k_\theta(f_1) \\ \bar{g}(f_1, f_2) & \text{sinon} \end{cases}$$

Finalement \bar{g} est défini par une *fonction de profil de déformation* $\bar{p} : \mathbb{R} \rightarrow \mathbb{R}$ retournant

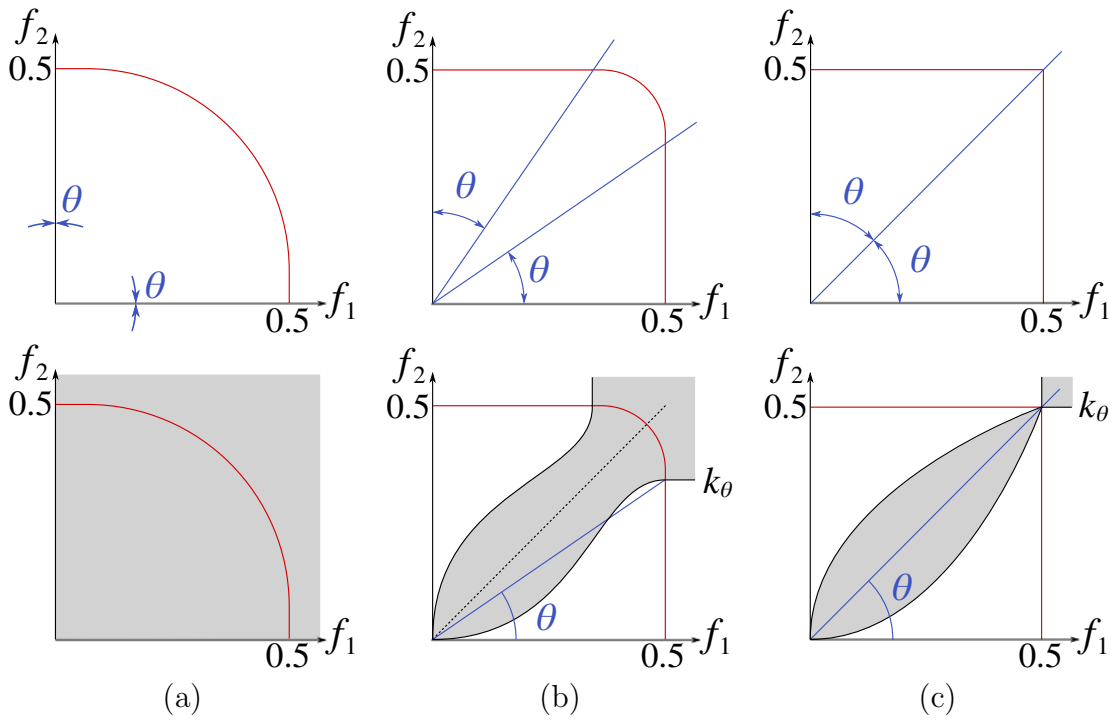


FIGURE 5.5 – Première ligne : L'angle d'ouverture θ définit la largeur de la transition au niveau de l'isovaleur correspondant à la surface observée (ici $\frac{1}{2}$). la courbe rouge représente l'isovaleur $\frac{1}{2}$ d'un opérateur de mélange $g(f_1, f_2, \theta)$. Deuxième ligne : Les courbes d'ouverture k_θ définissent les frontières de la zone de transition de l'opérateur g . (a) Opérateur complètement ouvert ($\theta = 0$). (b) Angle d'ouverture intermédiaire. (c) Opérateur fermé ($\theta = \frac{\pi}{4}$).

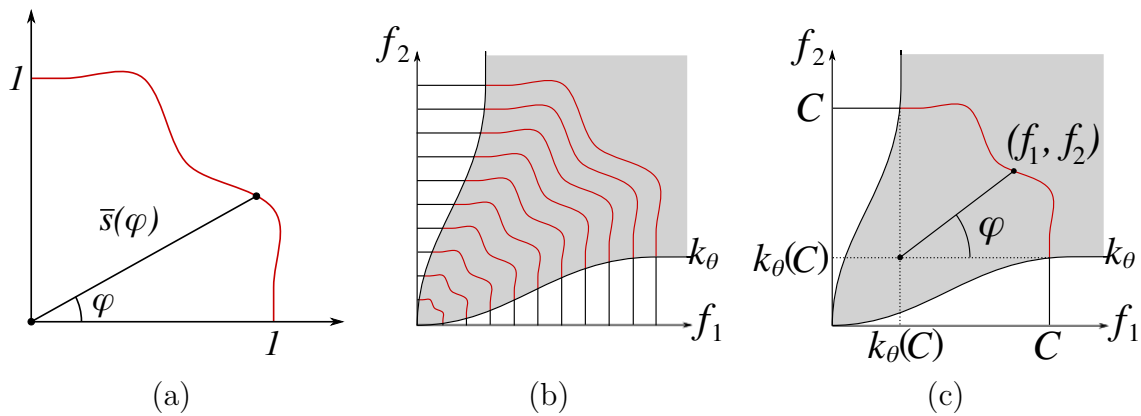


FIGURE 5.6 – (a) Le Profil de déformation \bar{p} définit la forme des isocourbes de la transition. Pour chaque valeur de ϕ , $\bar{p}(\phi)$ représente la distance de l'origine à l'isocourbe de valeur 1 de l'opérateur g en position pleinement ouverte (i.e. pour $\theta = 0$). (b) Isocourbes de l'opérateur g pour un angle $\theta > 0$. (c) Évaluer g en un point (f_1, f_2) quelconque nécessite de résoudre l'équation 5.2.

la distance entre l'origine et l'isocourbe 1 de \bar{g} pour chaque valeur de l'angle polaire ϕ (cf. Figure 5.6). Ainsi, un arc de cercle est obtenu en fixant $\bar{p}(\phi) = 1 \forall \phi$.

Évaluer l'opérateur g en un point (f_1, f_2) revient alors à trouver la valeur C solution de l'équation (cf. Figure 5.6) :

$$\frac{\sqrt{(f_1 - k_\theta(C))^2 + (f_2 - k_\theta(C))^2}}{\bar{p}(\phi)(C - k_\theta(C))} = 1, \quad \phi = \arctan \frac{f_2 - k_\theta(C)}{f_1 - k_\theta(C)}. \quad (5.2)$$

Cette expression est difficilement exploitable en pratique. Nous proposons une autre méthode d'évaluation de l'opérateur g en section 5.1.4.

Les *courbes d'ouverture* étant fixées, l'opérateur g est uniquement défini par le *profil de déformation* \bar{p} choisi. Pour satisfaire les contraintes de continuité de g avec la fonction max au niveau des *courbes d'ouverture*, le *profil de déformation* doit satisfaire aux conditions suivantes :

- $\bar{p}(0) = \bar{p}(\pi/4) = 0.5$
- \bar{p} est plate en 0 et en $\pi/4$ (toutes ses dérivées successives sont nulles en ces points).

Du fait de la continuité C^∞ des *courbes d'ouverture* k_θ , ces conditions sont suffisantes pour garantir la continuité C^∞ de l'opérateur g ainsi construit. Dans la section suivante, nous illustrons l'utilisation des *profils de déformation* par la création d'opérateurs de compositions variés.

5.1.4 Évaluation des opérateurs

L'évaluation d'un opérateur g exprimé par l'intermédiaire des *courbes d'ouverture* et des *profils de déformation* n'est pas aisée. En effet, celui-ci n'a pas d'expression analytique. Nous proposons donc d'évaluer g par interpolation de valeurs précalculées. La procédure d'échantillonnage est présentée en Annexe B. Une fois celle-ci réalisée, l'interpolation linéaire des huit valeurs nécessaires à l'évaluation de g en un triplet de paramètres (f_1, f_2, h) peut se faire rapidement et automatiquement sur GPU par l'intermédiaire d'une texture 3D.

Une attention particulière doit être apportée à la zone du support de g à précalculer. Il est nécessaire que celle-ci soit suffisamment grande pour garantir que l'évaluation de g reste possible même après plusieurs mélanges consécutifs. En réalité, il est nécessaire de précalculer g dans la surface contenue dans l'ensemble $[0; \frac{1}{2}] \times [0; \frac{1}{2}]$ uniquement. En effet, les *courbes d'ouverture* k_θ sont constantes et égales à $\frac{\tan(\theta)}{2}$ sur $[\frac{1}{2}; +\infty]$ ce qui permet

d'évaluer g de façon analytique sur l'ensemble $[\frac{1}{2}; +\infty] \times [\frac{1}{2}; +\infty]$. La valeur de g est alors obtenue par l'expression :

$$g(f_1, f_2) = \frac{\sqrt{(f_1 - f_\theta)^2 + (f_2 - f_\theta)^2}}{\bar{p}(\phi)} + f_\theta \quad (5.3)$$

où $f_\theta = \frac{\tan(\theta)}{2}$ et $\phi = \text{atan}(\frac{f_2 - f_\theta}{f_1 - f_\theta})$.

Le processus final d'évaluation de $g(f_1, f_2)$ est présenté en Algorithme 3.

Algorithm 3: Evaluation d'un opérateur de composition

```

1:  $g(f_1, f_2)$  {
2:   if  $f_1 < \frac{1}{2}$  et  $f_2 < \frac{1}{2}$  then
3:     lire et retourner la valeur de  $g$  en texture.
4:   else
5:     if  $f_1 > f_\theta$  et  $f_2 > f_\theta$  then
6:       calculer et retourner la valeur de  $g$  selon l'équation 5.3
7:     else
8:       retourner  $\max(f_1, f_2)$ 
9:     end if
10:  end if
11: }
```

L'évaluation du gradient de g est réalisée de façon analogue. Les dérivées partielles $\frac{\partial g}{\partial f_1}$ et $\frac{\partial g}{\partial f_2}$ sont précalculées et stockées dans des textures 3D. La valeur du gradient est ainsi approximée par :

$$\nabla g(f_1, f_2) = \frac{\partial g}{\partial f_1} \nabla f_1 + \frac{\partial g}{\partial f_2} \nabla f_2$$

5.2 Applications

Dans cette section, nous présentons des exemples d'opérateurs de composition créés avec notre méthode de conception développée en section 5.1.3. Ceux-ci sont définis à partir de leur fonction de profil \bar{p} et leurs contrôleurs associés h .

5.2.1 Mélange sans gonflement

L'une des grandes forces des primitives implicites dans les systèmes de modélisation est la capacité de mélanger de façon lisse des objets de formes variées, la jonction entre deux objets faisant apparaître une surface de courbure régulière. Un bon opérateur de mélange lisse est obtenu en utilisant un *profil de déformation* dont les variations de courbure sont les plus faibles possibles, tout en respectant les contraintes énoncées en section 5.1.3 (cf.

Figure 5.7). Un arc de cercle présente une variation de courbure minimale, mais ne satisfait pas à ces contraintes.

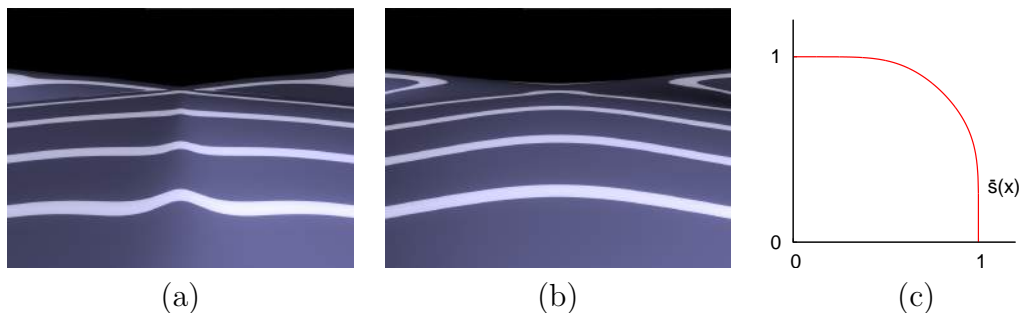


FIGURE 5.7 – Visualisation de lignes de réflexion sur une surface obtenue par mélange de deux cylindres implicites. (a) Opérateur de mélange de Bernhardt *et al.*[BBCW10] présentant une oscillation de la courbure. (b) Opérateur obtenu en utilisant notre *profil de déformation* \bar{p}_b . (c) Tracé de \bar{p}_b .

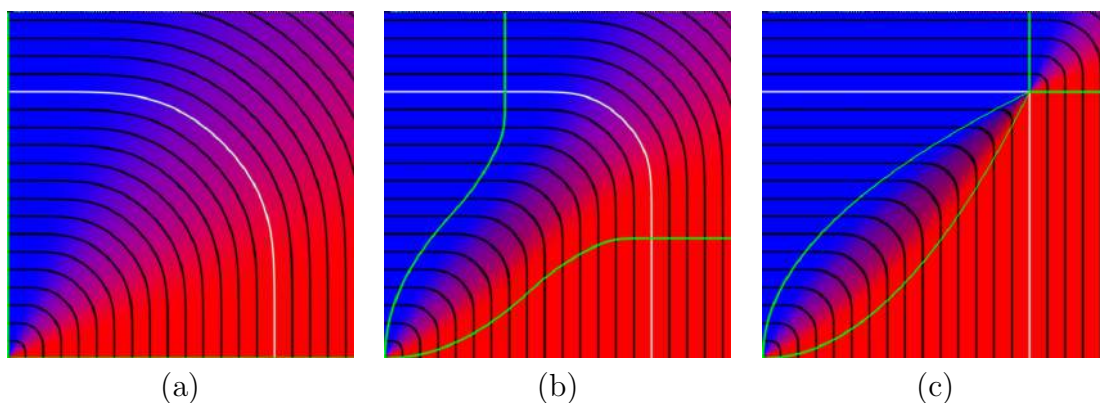


FIGURE 5.8 – Isocourbes de l'opérateur de mélange g_b pour différentes valeurs de θ . (a) Opérateur ouvert : $\theta = 0$. (b) Transition intermédiaire : $\theta = \frac{\pi}{8}$. (c) Transition franche : $\theta = \frac{\pi}{4}$.

La *fonction de profil* que nous avons utilisé est illustrée en Figure 5.7 (c). On peut constater que celle-ci est plate en $\phi = 0$ et en $\phi = \frac{\pi}{2}$, et sa courbure augmente de façon monotone sur l'intervalle $[0; \frac{\pi}{4}]$ puis diminue sur $[\frac{\pi}{4}; \frac{\pi}{2}]$. Ce *profil de déformation* est obtenu de la façon suivante.

Considérons la transformation $T : C^\infty(\mathbb{R}) \rightarrow C^\infty(\mathbb{R})$

$$T : v \mapsto u^{-1} \circ v \circ u$$

où $u : x \mapsto e^x - 1$. Cette transformation retourne la représentation de n'importe quelle

fonction sur une échelle logarithmique. En l'appliquant à l'hyperbole d'équation $z(x) = \frac{1}{x}$, nous obtenons une fonction de classe C^∞ dont les branches convergent très rapidement vers les axes. La forme du *profil de déformation* que nous utilisons est obtenue par mise à l'échelle et décalage d'une application supplémentaire de T à $T(z)$:

$$\bar{p}(x) = 1 - T^2(z)(e - x.e).e^{-1}$$

Il reste à exprimer \bar{p} sous forme polaire $\bar{p}(\phi)$. Cette expression étant peu commode à évaluer de façon analytique, nous la précalculons de façon discrète. Ceci est réalisé en échantillonnant la courbe en des points $X_i = (x_i, \bar{p}(x_i))$, puis en évaluant pour chaque X_i l'angle correspondant ϕ_i et sa valeur associée $\bar{p}(\phi_i)$. Par la suite, pour évaluer $\bar{p}(\phi)$, nous recherchons l'intervalle $[\phi_i; \phi_{i+1}]$ contenant ϕ de manière dichotomique, puis estimons $\bar{p}(\phi)$ par interpolation linéaire de $\bar{p}(\phi_i)$ et $\bar{p}(\phi_{i+1})$.

Notre *profil de déformation* ainsi créé n'est pas parfaitement plat en $\phi = 0$ et en $\phi = \pi/2$. Cependant, ses dérivées sont nulles avec une erreur relative $\epsilon = 10^{-5}$, ce qui est suffisant pour nos besoins. Pour des applications de visualisation, cet opérateur permet de générer des transitions plus esthétiques que l'opérateur de Bernhart *et al.*(cf. Figure 5.7 et 5.8).

Il ne nous reste plus qu'à définir des *contrôleurs* pour déterminer le comportement de notre opérateur de mélange. Nous proposons deux *contrôleurs*. Le premier, *Camel*, permet de résoudre les problèmes de mélange entre surfaces implicites mentionnés en introduction, en utilisant la méthode expliquée en Section 5.1.3. Il est ainsi adapté au contexte de modélisation industrielle CADG.

Des comparaisons entre un opérateur de mélange classique et notre opérateur contrôlé par le *Camel* sont présentés en Figure 5.9. Le mélange généré par notre opérateur ne forme pas de gonflement lorsque les surfaces en entrée sont parallèles, tout en réalisant une transition douce entre les objets. Il résout également le problème du mélange à distance de deux primitives implicites. En outre, il permet de conserver la topologie des objets de départ. Les détails ne sont pas estompés comme on peut le constater sur la Figure 5.9 (c) où deux objets implicites de tailles très différentes sont mélangés.

Le deuxième *contrôleur*, l'*Organique*, a été développé pour les besoins d'un système de modélisation interactif à base de surfaces de convolution à support global. Il était

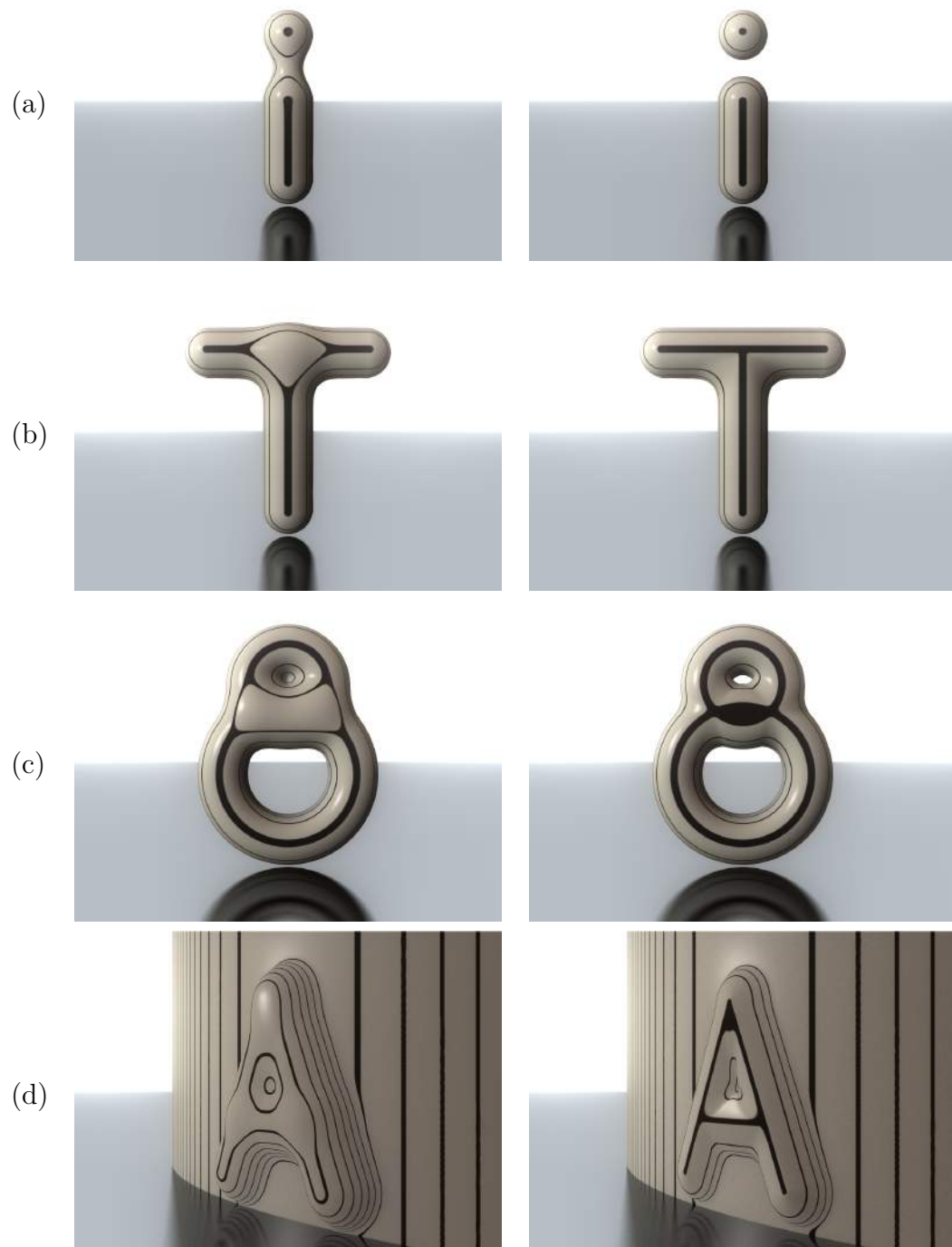


FIGURE 5.9 – Première colonne : illustration des problèmes classiques rencontrés lors de la composition de surfaces implicites ; (a) mélange à distance, (b) gonflement indésirable, (c) modification de la topologie (d) estompement des détails. Deuxième colonne : tous ces problèmes sont résolus en utilisant notre opérateur de mélange g_b associé au contrôleur *Camel* visible en Figure 5.10 (a).

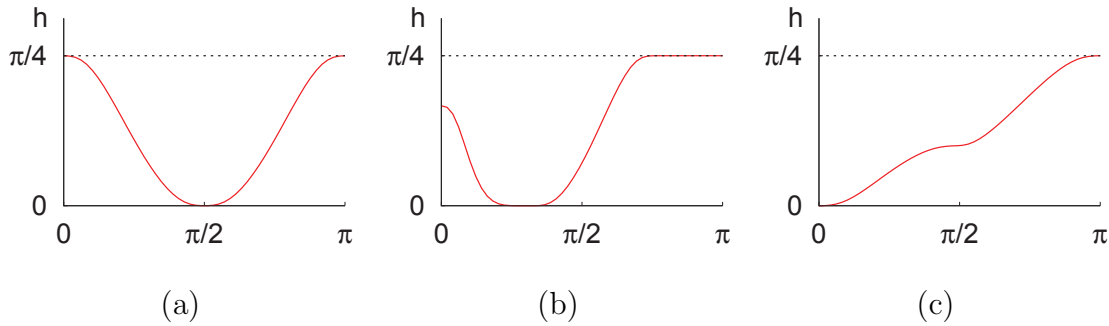


FIGURE 5.10 – Tracé des trois *contrôleurs* utilisés. (a) Le *Camel*, permettant de résoudre les problèmes liés aux mélange de surfaces implicites. (b) L’*Organique* qui est utilisé dans la modélisation de structures organiques par surfaces de convolution. (c) Le *Contact* utilisé conjointement avec l’opérateur de *gonflement au contact* pour simuler les collisions entre objets mous.

important de pouvoir réaliser simultanément des transitions douces et franches entre deux objets, par exemple lors du placement du nez d’un personnage sur son visage. Cet effet peut être obtenu par l’utilisation d’un contrôleur asymétrique : les surfaces se mélangent lorsque leurs gradients pointent dans la même direction, alors qu’une union franche est utilisée quand les surfaces tendent à se faire face (cf. Figure 5.11).

	α_0	α_1	α_2	h_0	h_1	h_2	w_0	w_1
Camel	0	$\pi/2$	π	$\pi/4$	0	$\pi/4$	1	1
Organique	0	$\pi/3$	$3\pi/4$	$\pi/6$	0	$\pi/4$	3	1
Contact	0	$\pi/2$	π	0	$\pi/10$	$\pi/4$	1	0.7

TABLE 5.1 – Valeur des paramètres utilisés pour définir nos contrôleurs.

5.2.2 Gonflement au contact

Pour illustrer la variété des opérateurs de mélange qu’il est possible de réaliser, nous avons conçu un opérateur de *gonflement au contact* permettant de simuler les déformations observables lors de la collision de deux objets *mous*, c’est-à-dire composés d’une matière élastique comme de la mousse. Lors du contact de deux de ces objets, un léger surélevement apparaît de part et d’autre de la zone de contact. Ceci peut être réalisé avec un *profil de déformation* adapté, celui-ci définissant la forme de la transition entre les objets.

Nous avons obtenu un tel profil de déformation en modifiant le profil correspondant à une transition franche de façon à former, sur chaque primitive, des gonflements dont la

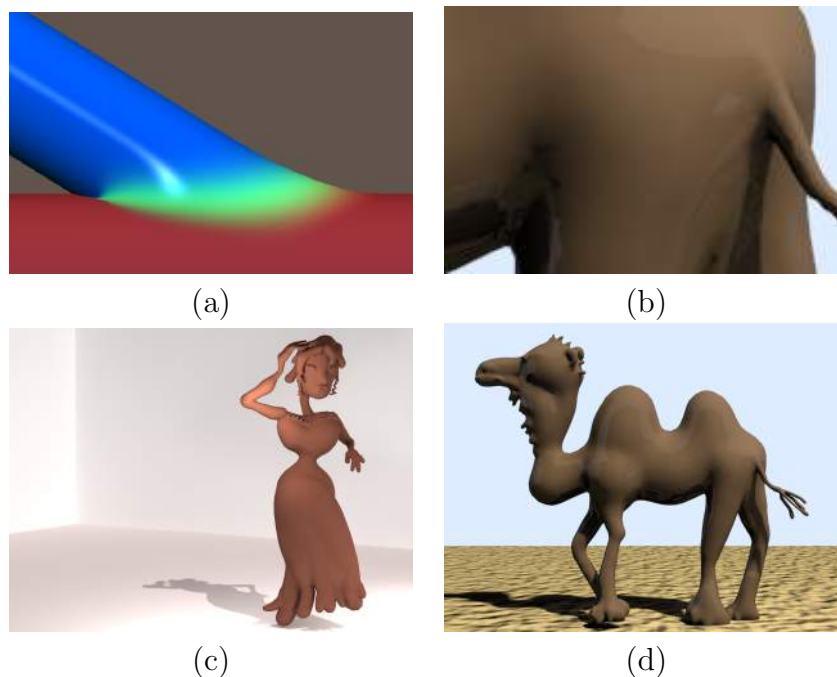


FIGURE 5.11 – Utilisation de l'opérateur de mélange g_b avec le contrôleur *Organique* pour la modélisation interactive. (a) Illustration de cet opérateur sur deux cylindres. Remarquez la présence simultanée d'une arête franche et d'un mélange lisse dans la transition. (b) La transition obtenue par cet opérateur est particulièrement adaptée lors de la fixation d'un membre sur un corps, par exemple lors de l'ajout de la queue au corps du chameau. (c) et (d) montrent deux modèles réalisés interactivement en mélangeant des primitives implicites simples avec cet opérateur.

taille est paramétrable. Une équation paramétrique de ce profil est :

$$\bar{p}_c(t) = \begin{cases} x_c(t) = \begin{cases} t & \text{si } x \geq 1 \\ 1 - K e^{1-\frac{1}{2-t}}(t-1) & \text{sinon} \end{cases} \\ y_c(t) = \begin{cases} 1 - K e^{1-\frac{1}{t}}(1-t) & \text{si } x \geq 1 \\ 2-t & \text{sinon} \end{cases} \end{cases}$$

le paramètre t variant entre 0 et 2. $K \in [0; 1]$ est le paramètre déterminant la taille des gonflements (cf. Figure 5.12). Pour passer à la forme polaire, nous procédons comme précédemment, en précalculant un grand nombre d'échantillons et en réalisant l'évaluation par recherche dichotomique. Ce *profil de déformation* est parfaitement plat en $\phi = 0$ et $\phi = \pi/2$ et est de classe C^∞ partout sauf en $\phi = \pi/4$ de façon à former le repli entre les deux surfaces en contact (cf. Figure 5.12).

Le comportement de cet opérateur est défini par le contrôleur *Contact* qui réalise une transition franche (pas de déformation) en absence de contact entre les objets, c'est-à-dire

lorsque les gradients s'opposent. Au fur et à mesure que les primitives pénètrent l'une dans l'autre, l'angle entre les gradients diminue et la taille du gonflement augmente. Ceci est obtenu en ouvrant l'opérateur quand l'angle entre les gradients diminue.

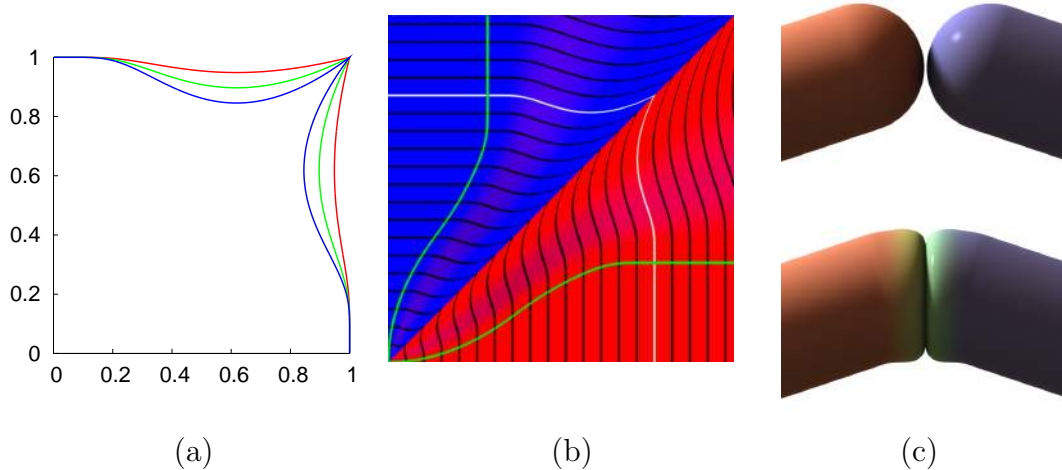


FIGURE 5.12 – (a) Tracés de \bar{p}_c pour $K = 0.25$ (en rouge), $K = 0.5$ (en vert) and $K = 0.75$ (en bleu). (b) L'opérateur de composition correspondant g_c pour $h = \frac{\pi}{8}$. (c) Application de cet opérateur utilisé avec le contrôleur *Contact* présenté en Figure 5.10 (c) illustrant la déformation obtenue lors de la collision de deux cylindres.

Nous avons utilisé cet opérateur pour simuler le fonctionnement d'une *lava lamp*. Le *gonflement au contact* est utilisé pour mélanger les bulles de fluide qui se déplacent le long de la lampe et retombent dans le réservoir, donnant ainsi l'illusion d'un contact souple entre les objets. Lorsque les bulles quittent le réservoir, celles-ci sont mélangées avec un opérateur de mélange pour donner l'apparence d'un fluide en déformation (cf. Figure 5.13).



FIGURE 5.13 – Application du mélange contrôlé par gradient à l'animation. La *lava lamp* est réalisée en composant plusieurs primitives implicites avec deux opérateurs, le *gonflement au contact* et un opérateur de mélange pour les bulles quittant la base.

5.3 Discussion

Nous avons réalisé une nouvelle famille d’opérateurs de mélange de surfaces implicites. Ces opérateurs permettent de contrôler précisément la forme de la transition entre les surfaces, augmentant ainsi le potentiel d’utilisation des surfaces implicites dans le domaine de la modélisation géométrique ou de l’animation.

Notre approche est basée sur l’utilisation des gradients des champs potentiels afin de contrôler le mélange. Ainsi, nos opérateurs ne sont plus uniquement fonctions de la valeur des champs potentiels, mais aussi de leurs gradients. Cette propriété peut introduire des changements de continuité, mais en pratique ceux-ci sont très localisés et maîtrisables (cf. 5.1.1). En outre, cela nous permet de résoudre les problèmes courants de la modélisation par objets implicites : les problèmes de gonflements indésirables et de mélanges à distance sont aisément résolus par l’utilisation d’un contrôleur adéquat, tel le *Camel*. Un effet de bord du *Camel* est que les petits détails en surface ne sont pas estompés en cas de mélange avec une primitive bien plus grande. Il est, de plus, possible de réaliser des transitions douces et franches au sein du même mélange.

Nos opérateurs peuvent être rapidement évalués en utilisant la mémoire texture du GPU. Par rapport à un opérateur de mélange classique, le seul surcoût provient de l’évaluation des gradients qu’il est nécessaire de réaliser à chaque composition. A chaque évaluation d’un champ potentiel f en un point p , nous calculons également son gradient $\nabla f(p)$ de façon systématique. Ceci permet de conserver une évaluation de l’arbre de mélange (CSG) en $\mathcal{O}(n)$, n étant le nombre de noeuds de l’arbre.

La principale limitation de notre travail est que nos opérateurs sont limités à des compositions binaires. Ceci empêche leur utilisation dans le cas où de nombreuses primitives doivent être mélangées simultanément de façon symétrique, par exemple dans les simulations de fluides basées sur des particules. La conception d’opérateurs de composition n -aires reste un problème difficile qui pourra faire l’objet de travaux futurs.

Dans le chapitre suivant, nous présentons un exemple d’application de nos opérateurs de composition dans le cadre de la déformation de maillage. Nous montrons comment le contrôle du mélange de primitives implicites peut être utilisé pour contrôler la déformation d’un maillage animé par un squelette.

6

Animation de maillages par surfaces implicites

Nous avons vu au cours des chapitres précédents que l'utilisation de surfaces implicites combinée à nos opérateurs de composition permettent de modéliser diverses transitions. En particulier, le contrôleur *Organique* offre la possibilité de combiner à la fois des transitions douces et franches au sein d'un même mélange, et permet une approximation intéressante des déformations de surfaces telles que des replis de peau.

Typiquement, ce genre d'effet est difficile à réaliser avec les représentations par maillage. D'une part les données de connexité entre sommets donnent une information de voisinage tangentiel et non de voisinage global. Or ce type de déformation est justement dû à la proximité d'autres surfaces. D'autre part l'utilisation de maillages nécessite de veiller au bon maintien de leur topologie, particulièrement lorsque de fortes déformations sont appliquées. Néanmoins il est illusoire de vouloir remplacer les maillages par les surfaces implicites. Celles-ci ont leurs propres limitations, ne serait-ce que pour citer l'application de textures, une technique largement répandue dans tous les domaines liés à l'infographie et qui permet d'ajouter du détail à une surface à faible coût.

Nous avons alors eu l'idée d'utiliser conjointement ces deux représentations. En effet, puisqu'il est simple de déformer des surfaces implicites par l'utilisation d'opérateurs de composition adaptés, une idée naturelle est d'utiliser la composition de surfaces implicites en tant que support de déformation de maillages : c'est l'objet des travaux présentés dans ce chapitre, qui ont été réalisés au cours des huit derniers mois de ma thèse.

L'animation de personnages est généralement réalisée par l'intermédiaire d'une structure appelée *squelette*. Cette structure est composée de segments rigides appelés *os* liés entre eux par des *articulations*. Le *squelette* est en général créé dans une configuration initiale appelée *pose de repos* qui correspond au maillage non déformé. L'animation du maillage ou *skinning* consiste alors à reproduire sur le maillage les déformations appliquées au *squelette*. Celles-ci sont réalisées par rotation des *os* autour des *articulations*, ce qui a pour effet de changer la *pose* du *squelette*, c'est-à-dire sa configuration.

Une caractéristique souhaitée durant le *skinning* d'un personnage est la reproductibilité. Concrètement, si l'on déforme le *squelette* d'une *pose* p_a à une *pose* p_b , puis qu'on le ramène dans la configuration p_a , le maillage doit avoir retrouvé sa forme initiale. Ceci implique que pour chaque *pose* p du *squelette*, il doit correspondre une et une seule déformation du maillage.

Chaque *os* du *squelette* est habituellement représenté par sa longueur et un repère dont l'origine est située à l'une de ses extrémités, et dont l'un des axes pointe vers l'extrémité opposée. Une représentation équivalente est obtenue par la transformation permettant de passer de ce repère au repère objet. Celle-ci peut être exprimée sous la forme d'une matrice de taille 4×4 par l'intermédiaire des coordonnées homogènes.

Nous utilisons les notations suivantes. La *pose initiale* est notée 0. La transformation qui, appliquée à un point exprimé dans le repère de l'*os* i dans la configuration p , exprime ce même point dans le repère objet est notée T_i^p . v_j^p désigne la position du sommet j du maillage exprimée dans le repère objet lorsque le *squelette* est dans la *pose* p .

La forme la plus simple de *skinning* est obtenue en associant chaque sommet du maillage à un *os* du *squelette*. Le maillage est alors déformé en appliquant à chaque sommet la déformation qu'a subie l'*os* correspondant. Les coordonnées des sommets sont alors calculées par l'expression suivante :

$$v_j^p = T_i^p \cdot (T_i^0)^{-1} \cdot v_j^0$$

où i est l'*os* associé au sommet v_j .

Cette technique est appelée *skinning rigide* du fait que chaque sommet est immobile dans le repère de l'*os* qui lui est associé. Elle est peu adaptée à l'animation de personnages du fait de sa raideur.

6.1 Méthodes existantes

La technique la plus simple et la plus utilisée pour le skinning de personnages en temps réel est le *skinning linéaire*, également appelée *SSD* (de l'anglais Skeletal Subspace Deformation). Celle-ci a été introduite par Magnetat *et al.*[MtLTM88] afin d'étudier les déformations d'un modèle de main.

Le *skinning linéaire* associe à chaque sommet v_j et à chaque os i un *poids de skinning* $w_{i,j}$. La transformation appliquée à chaque sommet est obtenue par combinaison linéaire des transformations appliquées aux os en utilisant les poids correspondant :

$$v_j^p = \sum_{i=1}^b w_{i,j} \cdot T_i^p \cdot (T_i^0)^{-1} \cdot v_j^0$$

Les poids sont tels que $\forall j \sum_{i=1}^b w_{i,j} = 1$. En général, le nombre d'os définissant la position de chaque sommet est compris entre 1 et 4, ce qui implique que la plupart des poids $w_{i,j}$ sont nuls. Le calcul de la transformation du maillage est ainsi extrêmement rapide.

Le *skinning linéaire* permet d'obtenir des déformations plus élastiques que le *skinning rigide*, qui sont donc plus adaptées à la modélisation des mouvements des membres humains. Le choix des poids est un paramètre déterminant dans le réalisme des animations obtenues. Un choix standard consiste à utiliser des poids dont la valeur est proportionnelle à la distance géodésique des sommets aux os les plus proches, mais il est bien souvent nécessaire de retoucher les poids à la main pour obtenir un résultat satisfaisant.

Le défaut majeur du *skinning linéaire* est qu'il provoque une grande perte de volume lorsque le *squelette* est configuré dans des poses trop éloignées de la *pose au repos*. Les situations typiques observées sont l'effondrement d'un coude (cf. Figure 6.1 (a)) et l'effondrement sous torsion (cf. Figure 6.1 (b)). Celles-ci sont provoquées car cette technique interpole linéairement des matrices de transformation, chacune correspondant à une rotation autour d'une articulation. Or, cette interpolation n'est pas équivalente à l'interpolation des paramètres de ces rotations. Un autre inconvénient est la difficulté à contrôler le skinning, le choix des poids ne permettant pas toujours de contrôler l'aspect de la déformation souhaitée.

Malgré cela, le *skinning linéaire* reste très populaire en raison de sa simplicité et de sa rapidité. De nombreux auteurs ont travaillé à améliorer cette technique. Mohr et Gle-

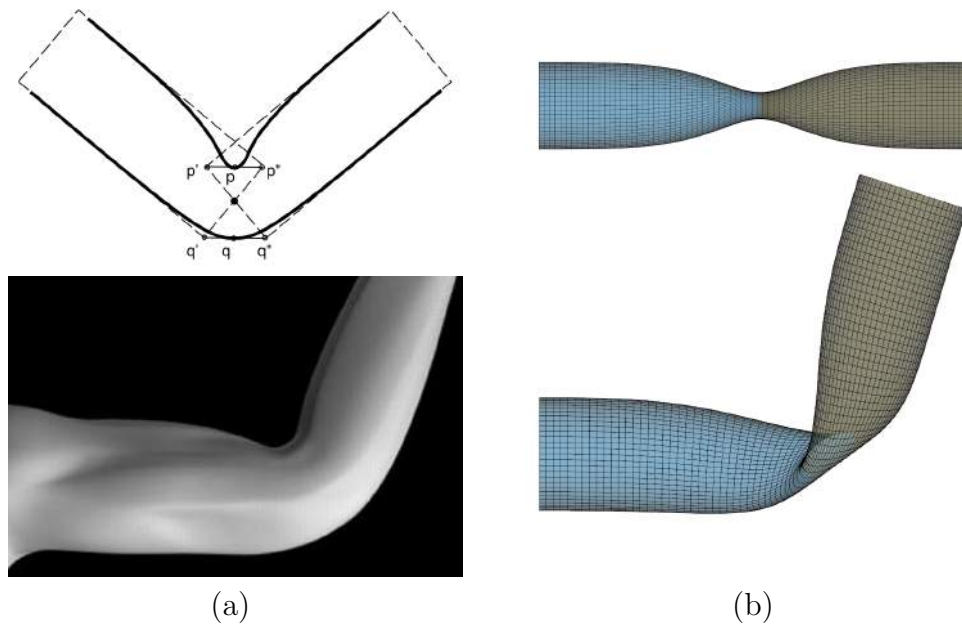


FIGURE 6.1 – Principaux défauts du skinning linéaire. L'interpolation linéaire des transformations crée (a) un creux à la jonction de deux membres en cas de flexion et (b) un amincissement des membres en cas de torsion.

cher [MG03] parviennent à limiter les pertes de volume par l'ajout d'*os* supplémentaires au *squelette* à l'extrémité des *os* existants. Le problème du *skinning linéaire* étant justement causé par l'interpolation linéaire des transformations, certains auteurs proposent des schémas d'interpolations alternatifs [MTCSP04] [Kv05] [KCvO08]. Par exemple, l'algorithme présenté par Kavan *et al.*[KCvO08] interpole les paramètres des rotations en utilisant une représentation alternative des transformations basée sur les quaternions en espace des nombres duaux. Ces approches permettent de résoudre en grande partie les problèmes d'effondrement du maillage, mais ne permettent pas de contrôler l'aspect de la déformation.

Une autre approche de skinning consiste à utiliser plusieurs *poses au repos*. Les données initiales sont ainsi définies par un ensemble de paires (p_i, M_i) constituées d'une *pose* p_i et de la configuration du maillage associé M_i .

Une manière d'utiliser ces données est de calculer automatiquement les *poids de skinning* du *skinning linéaire*. En effet, la détermination des poids est en général longue et peu intuitive lorsqu'elle est réalisée manuellement. Les techniques proposées par des auteurs tels que Wang *et al.*[Wea02] et Merry [MMG06] consistent à calculer les *poids de skinning* de manière à minimiser l'écart obtenu avec les maillages M_i lorsque le squelette est en *pose* p_i . Cependant le nombre de poses doit être suffisant et celles-ci doivent être choisies

soigneusement de façon à ce que le problème de minimisation à résoudre ne soit pas sous-contraint. De manière analogue, Kavan *et al.*[KCO09] optimisent les poids de skinning de manière à approximer un skinning non-linéaire. Le temps de calcul est alors inférieur au *skinning* utilisant les quaternions en espace des nombres duaux avec une qualité similaire.

L'approche proposée par Lewis *et al.* dans [LCF00] est d'obtenir la position des sommets par interpolation directe des *poses au repos*. Ces *poses* pouvant être choisies de façon arbitraire, l'interpolation est réalisée par des fonctions à base radiale (RBF). Une phase d'apprentissage est ainsi nécessaire afin de déterminer les poids de RBF propres à chaque sommet. Cette approche permet de contrôler l'aspect de la déformation, et de simuler ainsi les gonflements des muscles et les replis de la peau. Cependant elle nécessite un travail supplémentaire de la part des artistes, qui doivent réaliser plusieurs maillages du même modèle (un par *pose*). De plus, l'empreinte mémoire du modèle est beaucoup plus grande car l'ensemble des maillages M_i doit être stocké en mémoire en plus des poids des RBF.

Dans [WPP07], Wang *et al.* proposent de réaliser un modèle de prédiction des déformations à partir des (p_i, M_i) . Les déformations locales sont observées par l'intermédiaire de descripteurs appelés *deformation gradient* qui stockent la déformation de chaque triangle via une matrice 3×3 . L'hypothèse formulée par Wang est que les composantes de rotation de ces déformations dépendent linéairement d'une et une seule articulation, et les composantes de mise à l'échelle sont fonctions linéaires de deux articulations au plus. La loi de déformation $D(p)$ de chaque triangle en fonction de la *pose* du *squelette* est alors obtenue selon une régression par moindres carrés. Le maillage est ensuite reconstruit de manière à minimiser l'écart avec la déformation prévue $D(p)$. Cette dernière étape est réalisée sur GPU, le problème d'optimisation linéaire obtenu étant résolu par factorisation de Cholesky, ce qui permet d'obtenir des temps d'exécution compétitifs avec le *SSD*. Cette technique est plus robuste que celle proposée par Lewis et nécessite moins de *poses* p_i pour obtenir le résultat souhaité. Cependant, l'empreinte mémoire reste importante en raison du stockage des *deformation gradients*. Elle est en générale 10 fois plus importante que dans le cas du *skinning linéaire*. Le temps d'apprentissage par régression peut également être conséquent (plusieurs dizaines de minutes pour des maillages de l'ordre de 20000 polygones).

D'autres méthodes de skinning permettent de corriger automatiquement les pertes de volumes potentielles survenant par exemple lors du repli d'une articulation. Ces méthodes produisent ainsi des gonflements ou rides sur le maillage afin d'augmenter le réalisme de

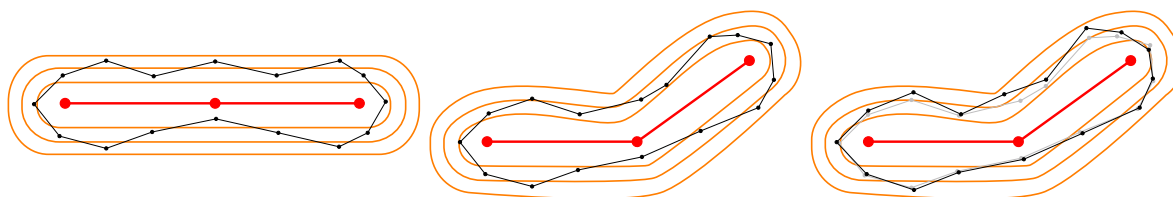


FIGURE 6.2 – Principe du *skinning implicite*. (a) Une primitive implicite est générée à chaque os du squelette. Ces primitives sont mélangées par un opérateur de composition qui définit la forme de la déformation de l’articulation correspondante. (b) Lorsque le squelette est déformé, une première transformation est appliquée au maillage qui positionne les sommets de manière approximative. (c) Les sommets sont déplacés de façon à rejoindre les isosurfaces correspondantes. Il suivent donc les déformations du champ potentiel qui sont définies par l’opérateur de mélange utilisé.

la déformation [vFTS08] [RHC09]. Malheureusement ces approches sont très coûteuses en temps de calcul. Les problèmes d’interpénétration de la surface peuvent être résolus en utilisant la méthode de préservation de volume par intégration de champs vectoriels d’Angelidis *et al.*[AS07]. Toutefois cette méthode s’intègre difficilement dans les chaînes d’animation standards car elle nécessite de réaliser une intégration temporelle des sommets du maillage.

6.2 Skinning implicite

Nous avons développé une méthode de skinning utilisant les surfaces implicites. Comme le *skinning linéaire*, celle-ci ne requiert qu’une *pose au repos* et ne nécessite pas d’apprentissage. Notre technique se base sur le squelette pour calculer la déformation à appliquer de manière automatique, mais celle-ci peut également être paramétrée facilement. Elle permet de simuler rapidement et de manière réaliste les articulations simples du corps humain (doigts, coude, etc..)

6.2.1 Principe de fonctionnement

Le *skinning implicite* fonctionne de la manière suivante. Nous générons une primitive implicite à partir de chaque *os* du squelette. Nous les mélangeons ensuite en utilisant des opérateurs de compositions adaptés, l’arborescence du *squelette* définissant l’ordre de mélange des primitives.

Nous estimons la valeur du champ potentiel généré par le squelette en chaque sommet v_j du maillage. Celle-ci est notée c_j . L’idée de la méthode du *skinning implicite* est que

les valeurs de potentiel c_j doivent être constantes, quelle que soit la *pose* du squelette. Lorsque la *pose* change, le champ potentiel généré par le squelette change également. Nous déplaçons alors chaque sommet v_j de manière à ce qu'il rejoigne l'isosurface de valeur c_j (cf. Figure 6.2). En fonction des primitives implicites et des fonctions de mélange utilisées, divers effets peuvent être obtenus. Nos opérateurs de mélange définis au chapitre précédent permettent ainsi de simuler les replis et gonflements de peau qui apparaissent lors de la contraction d'un membre.

6.2.2 Primitives implicites

Le choix du type de primitives implicites générées à partir du *squelette* est essentiel au bon fonctionnement du *skinning implicite*. Leur mélange avec l'opérateur de composition choisi doit en effet être le reflet de la déformation souhaitée au niveau du maillage, afin d'en faciliter la conception. Ceci conduit à rechercher deux propriétés au niveau du champ potentiel :

1. Autant que possible, la forme des isosurfaces doit se rapprocher de la forme du maillage. Dans l'idéal, tous les sommets doivent se situer sur la même isosurface. Il faut donc chercher à minimiser l'écart entre les c_j .
2. La déformation souhaitée doit apparaître de manière homogène sur les isosurfaces c_j du champ potentiel de façon à être reflétée sur le maillage. Ces isosurfaces doivent ainsi être parallèles. Pour cela, le champ potentiel doit présenter un caractère lisse.

En plus de ces propriétés, il est souhaitable que l'évaluation du champ potentiel soit rapide, car les évaluations sont nombreuses lors de l'étape de suivi d'isosurfaces.

Nous proposons d'utiliser des *cylindres implicites* à support compact générés par des segments superposés aux *os* correspondant. En effet, la forme de ceux-ci correspond approximativement à la forme des membres du corps humain (bras, jambes, doigts), et leur extrémité arrondie est adaptée à la représentation des *coudes* au niveau des articulations. De plus, leurs isosurfaces sont toutes parallèles et leur champ potentiel lisse (d'ordre G^1). En outre, ces primitives sont suffisamment simples pour être évaluées rapidement.

Le champ potentiel généré par le cylindre issu de l'os o_k en un point p est :

$$b_k(p) = \left(1 - \left(\frac{d(p, o_k)}{R_k}\right)^2\right)^\alpha \quad (6.1)$$

où $d(p, o_k)$ est la distance du point p au segment formé par l'os o_k et R_k le rayon d'influence de la primitive. Ce rayon d'influence doit être choisi de manière à minimiser l'écart entre

les valeurs de c_j des sommets v_j situés à proximité de l'os o_k . En effet, la surface implicite générée par le squelette doit modéliser au mieux la surface représentée par le maillage, sous peine que les effets obtenus avec les opérateurs de mélange ne soient pas bien reproduits sur le maillage. Le squelette joue ainsi un rôle capital dans la bonne représentativité de la surface et pour obtenir de bons résultats, il faut que les os soient autant que possible localisés au centre de leurs membres respectifs.

Cette étape est réalisée de la façon suivante. Nous commençons par déterminer pour chaque sommet v_j l'os du squelette qui lui est le plus proche. Ceci nous permet de segmenter les sommets en fonction des os, et pour chaque os o_k , il lui est associé un ensemble de sommets V_k . Nous déterminons ensuite pour chaque os o_k le rayon d'influence R_k tel que le potentiel moyen des sommets qui lui sont associés soit égal à $\frac{1}{2}$. Ceci est obtenu en calculant la distance moyenne des sommets de V_k à O_k et en inversant la formule 6.1 :

$$R_k = \frac{d_{\text{avg}}^k}{\sqrt{1 - (\frac{1}{2})^{\frac{1}{\alpha}}}}$$

où

$$d_{\text{avg}}^k = \frac{1}{\text{Card}(V_k)} \sum_{v_j \in V_k} d(v_j, o_k)$$

6.2.3 Déplacement des sommets

Lors d'une modification de la configuration du *squelette*, il faut déplacer les sommets de façon à ce qu'ils rejoignent à nouveau l'isosurface du champ potentiel généré par le squelette qui leur est associé, ce processus permettant de répercuter sur le maillage les transformations du champ potentiel. Le processus de déplacement des sommets s'effectue en deux étapes. Dans une première étape, nous positionnons les sommets dans une configuration initiale correspondant à la pose courante du *squelette*. Cette étape est réalisée de manière déterministe, à partir du maillage initial. Elle permet de satisfaire à la contrainte de reproductibilité énoncée plus haut. Dans une seconde étape, nous déplaçons chaque sommet v_j de manière itérative en utilisant les gradients du champ potentiel jusqu'à l'isosurface c_j . Ces deux étapes se parallélisent aisément sur GPU, les sommets étant traités indépendamment.

La première étape joue un rôle capital dans la convergence de la seconde. En particulier, celle-ci doit respecter la topologie initiale du maillage. Si celle-ci est modifiée, par exemple si deux sommets sont croisés, les changements seront répercutés à l'issue de la deuxième étape, du fait du caractère lisse du champ potentiel. Cette mise en configura-

tion initiale ne peut donc pas être effectuée avec un *skinning rigide*, car celui-ci réalise systématiquement un croisement de sommets lors d'un mouvement du squelette autour d'une articulation. À la place, nous avons décidé d'utiliser un *skinning linéaire*. En effet, celui-ci respecte la topologie initiale du maillage dans la limite de mouvements raisonnables des articulations. Notons qu'un repli d'une articulation sur elle-même finit toujours par déclencher une auto-intersection de la surface du maillage, quels que soient les poids de skinning utilisés. Toutefois ces auto-intersections surviennent pour des rotations extrême des articulations. Si l'on se place dans le cadre des limites des articulations du corps humain, il est aisé de définir des poids de skinning de façon à ce qu'aucune auto-intersection ne survienne.

La seconde étape consiste à déplacer chaque sommet v_j de manière à rejoindre l'isosurface du champ potentiel c_j qui lui est associée. Une manière triviale de procéder consiste à marcher le long du gradient du champ potentiel selon un pas de faible longueur ϵ . Toutefois, il se peut que rejoindre l'isosurface soit impossible de cette manière, un sommet pouvant se retrouver dans un puits de potentiel sans pouvoir en sortir. C'est par exemple le cas lors de la simulation d'un repli de peau formé par une arête franche.

Pour résoudre ce problème, nous arrêtons la convergence de l'algorithme dès qu'un changement de variation du gradient est détecté. Nous procédons alors à une recherche locale de la valeur de potentiel la plus proche de c_j par dichotomie.

Un autre problème de la méthode consistant à marcher le long du gradient du champ potentiel est que les fluctuations du champ potentiel peuvent entraîner des variations importantes de la densité des sommets à la surface du maillage. Ainsi des dilatations excessives de la surface du maillage peuvent survenir dans les zones de mélange des champs potentiels générés par les os. Ce phénomène est amplifié par le fait que l'étape de *skinning linéaire* a tendance à éloigner les sommets de leur isosurface associée dans les zones situées au creux des articulations (cf. Figure 6.3).

Ce problème peut être résolu en modifiant la marche des sommets de la seconde étape. En effet, la cause de ce phénomène est que le gradient du champ potentiel dirige les sommets vers les points des isosurfaces les plus proches de leurs positions initiales. Or, ces positions initiales sont obtenues à l'issue du processus de *skinning linéaire*, qui souffre des problèmes d'effondrement de maillage. Ceux-ci se traduisent par un déplacement excessif des sommets orthogonalement à l'axe de rotation des articulations. Afin de ne pas subir la perte de densité à l'issue de la deuxième étape, il est nécessaire de compenser ce

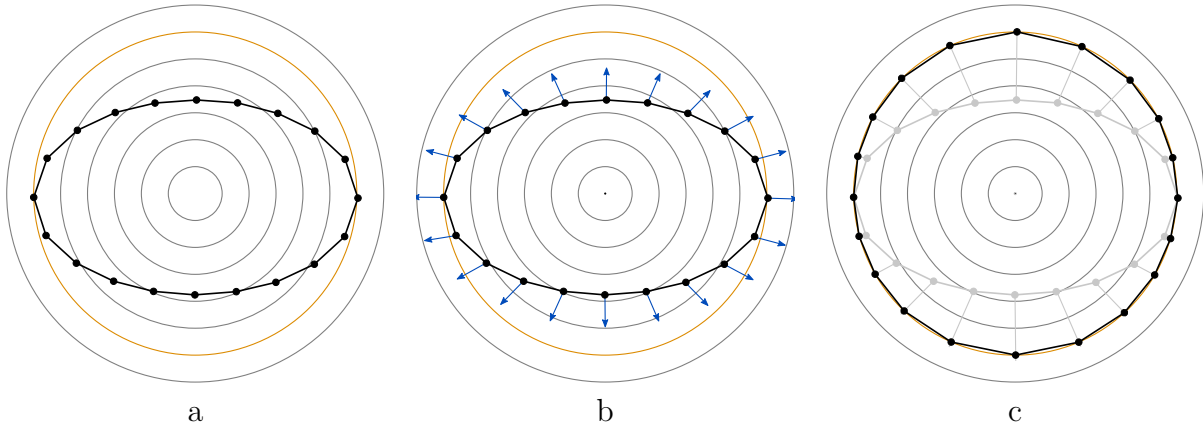


FIGURE 6.3 – Une marche triviale le long du gradient du champ potentiel (b) peut entraîner une variation de la densité des sommets (c) relativement au maillage initial (a).

déplacement. Pour cela, nous calculons pour chaque sommet la composante de rotation de la transformation qu'il a subie lors du *skinning linéaire*. Nous effectuons ensuite la marche en favorisant la composante du gradient orthogonale à l'axe de cette rotation.

La composante de rotation de la transformation $T_j = \sum_{i=1}^b w_{i,j} \cdot T_i^p \cdot (T_i^0)^{-1}$ appliquée à un sommet v_j est obtenue en effectuant une décomposition polaire de T_j :

$$T_j = O_j \cdot R_j$$

où O_j est une matrice orthogonale et R_j une matrice symétrique. La matrice orthogonale est calculée par une méthode itérative simple. Considérons la suite de matrices M_j^k définie par :

$$M_j^0 = T_j \text{ et } M_j^{k+1} = \frac{1}{2} \left[M_j^k + ((M_j^k)^{-1})^T \right]$$

De par les propriétés des matrices orthogonales et symétriques, la suite de matrices M_j^k converge vers O_j quand $k \rightarrow +\infty$. Cette convergence est rapide et une dizaine d'itérations en moyenne permettent d'obtenir O_j avec une erreur relative de 10^{-5} . L'axe de la rotation est alors donné par le vecteur :

$$r_j = (O_{j3,2} - O_{j2,3}, O_{j1,3} - O_{j3,1}, O_{j2,1} - O_{j1,2})$$

Nous effectuons ensuite l'étape de marche des sommets. L'effondrement du maillage est particulièrement prononcé au creux des articulations, qui correspondent aux zones où le gradient du champ potentiel est orthogonal à l'axe de rotation de l'articulation. Dans ces zones, le déplacement doit être effectué davantage selon la composante du gradient orthogonale à l'axe de rotation que selon sa composante tangentielle. Nous proposons de

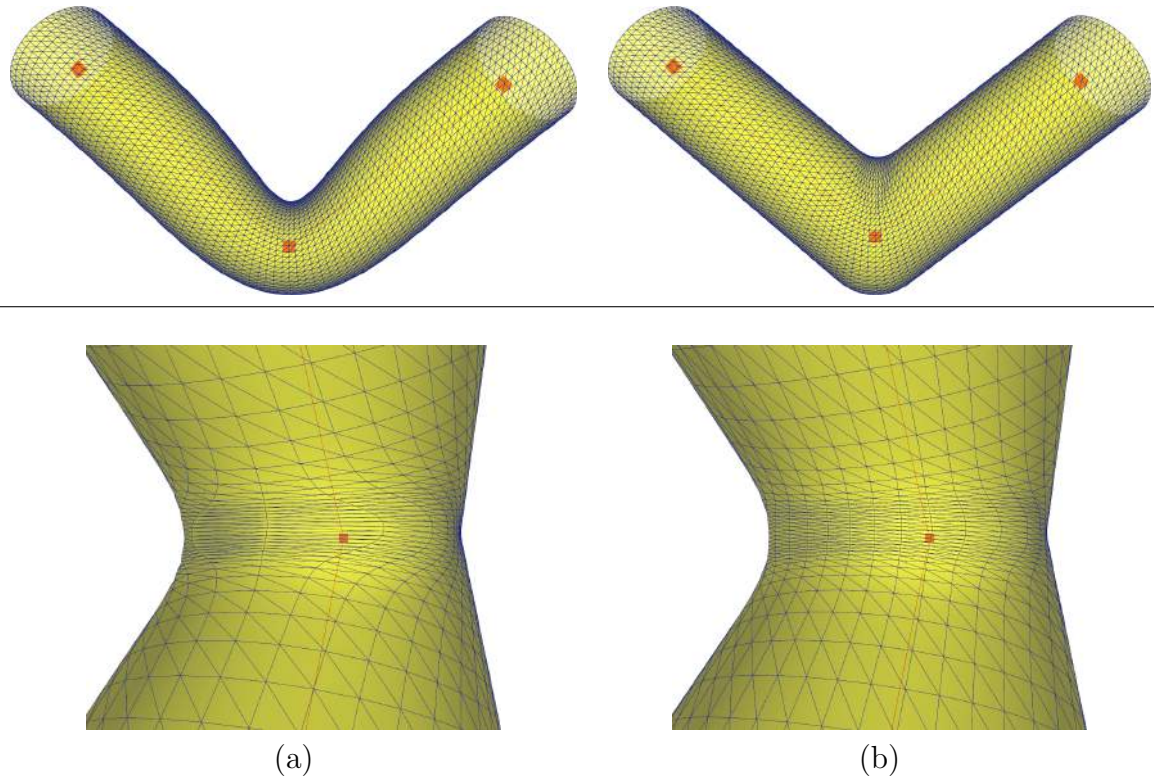


FIGURE 6.4 – Première ligne : skinning d'un maillage de cylindre par un squelette comportant une articulation. (a) Skinning classique par SSD. (b) Skinning implicite. Le champ potentiel utilisé a été obtenu par composition des cylindres implicites centrés sur les os du squelette. Deuxième ligne : (a) Résultat du skinning implicite au creux de l'articulation lorsque le déplacement des sommets est effectué dans la direction du champ potentiel. Une perte de densité est visible. (b) Résultat obtenu lorsque le déplacement est effectué selon le vecteur de déplacement décrit dans l'équation 6.2.

déplacer chaque sommet v_j dans la direction du vecteur

$$d_j = \nabla f |_{v_j} + ((1 - u_j) \cdot u_j) \cdot r_j \text{ où } u_j = \nabla f |_{v_j} \cdot r_j \quad (6.2)$$

Dans nos observations, l'utilisation de cette formule permet de maintenir la densité radiale des sommets en surface (cf. Figure 6.4). Notons qu'une rotation autour d'une articulation entraîne toujours une compression longitudinale des sommets situés au creux de cette articulation et une dilatation des sommets situés à l'opposé de ce creux. Ces phénomènes traduisent les compressions et dilatations locales de la surface, celles-ci étant d'autant plus prononcées que le centre de l'articulation est éloigné de la surface. Ils se produisent d'ailleurs naturellement au niveau de toutes les articulations du corps humain.

6.2.4 Mélange des primitives

Comme expliqué en Section 6.2.1, les fonctions de mélange utilisées pour composer les cylindres implicites doivent être configurées de manière à réaliser les déformations adéquates au niveau du maillage. Un simple opérateur d'union en *max* permet de réaliser un coude possédant une arête franche au niveau de la transition (cf. Figure 6.5 (a)). Des effets plus avancés tels que le gonflement d'un muscle peuvent être obtenus en utilisant les opérateurs de compositions contrôlés par gradient présentés au Chapitre 5.

Ainsi, l'opérateur de *mélange sans gonflement* permet d'obtenir automatiquement un comportement analogue à celui d'un coude humain. Lorsque les membres sont étirés la peau est tendue. Lorsque le coude se plie, il n'y a pas de repli visible dans un premier temps du fait de l'élasticité de la peau, un comportement similaire à celui obtenu en zone de transition douce de notre opérateur. Ce n'est qu'à partir d'un certain angle que le pli se marque : c'est le comportement que nous observons en mode de transition franche (cf. Figure 6.5 (b)).

L'opérateur de *gonflement au contact* permet quant à lui de simuler les articulations courtes du type de celles reliant les phalanges d'une main : le pli apparaît très rapidement lorsque le doigt se replie. Un gonflement se produit alors de part et d'autre de l'articulation qui résulte du déplacement de la chair. Notre opérateur de mélange permet de modéliser ce comportement tout en réduisant les phénomènes d'auto-intersection au niveau du pli (cf. Figure 6.5 (c)).

6.3 Résultats

6.3.1 Comparaison qualitative

La figure 6.6 illustre notre technique de skinning dans le cadre de l'animation d'un modèle de main. Dans cet exemple, les cylindres implicites issus des différents os du squelette sont composés à la manière d'un CSG, selon la hiérarchie définie par le squelette. Les primitives implicites situées au même niveau dans la hiérarchie représentent un ensemble solidaire de plusieurs os. Nous les composons avec l'opérateur de *mélange sans gonflement* afin d'obtenir un champ potentiel lisse. Les primitives résultantes sont composées de père à fils avec l'opérateur *gonflement au contact* afin de modéliser les replis de peau lors de la flexion d'un membre.

En comparaison avec l'approche SSD classique, le skinning implicite permet d'obte-

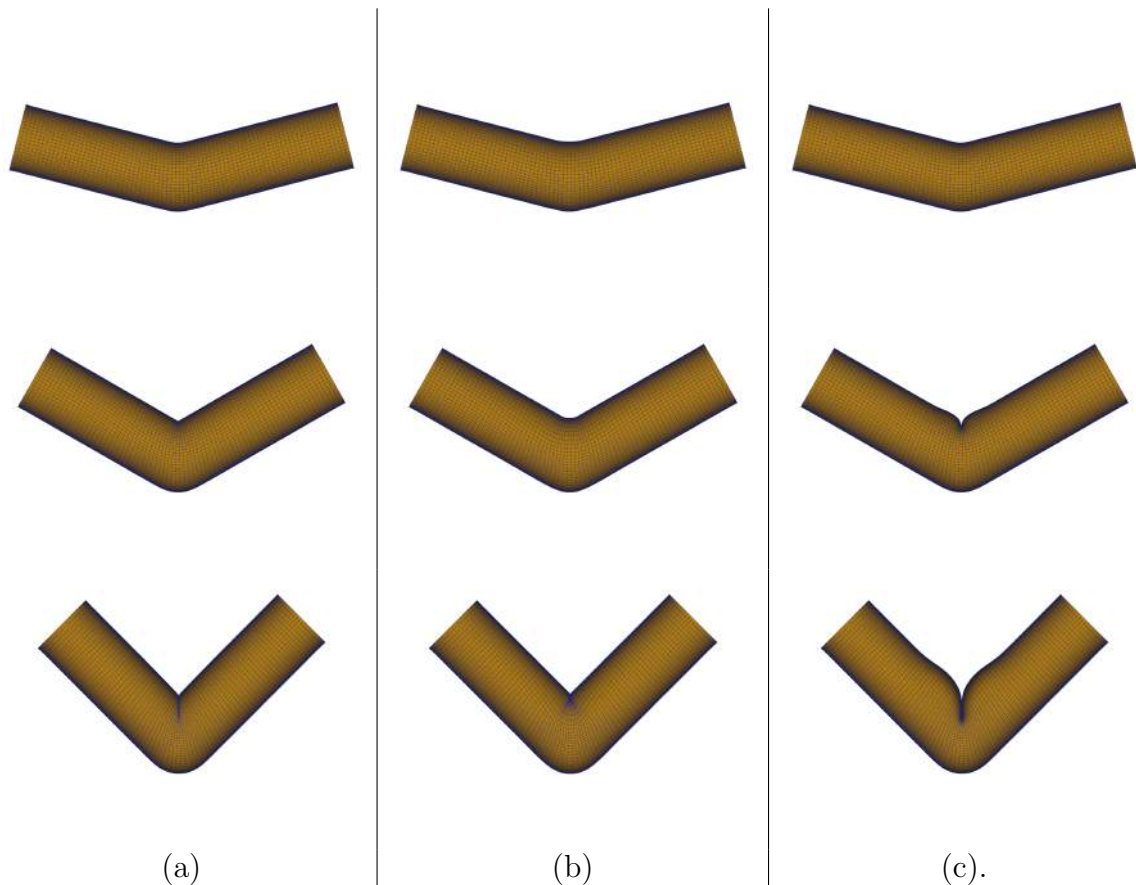


FIGURE 6.5 – Exemples de déformations modélisables au niveau de la jonction de deux membres modélisés par des cylindres. (a) Transition franche obtenue par l'opérateur d'union *max*. (b) Transition douce qui se réduit au fur et à mesure que l'articulation se plie jusqu'à l'apparition d'une transition franche. Celle-ci est obtenue en utilisant l'opérateur *Mélange sans gonflement* présenté au Chapitre 5. Elle modélise l'effet obtenu sur l'articulation d'un coude humain. (c) Transition franche faisant apparaître des bourrelets de part et d'autre de l'articulation. Elle est obtenue par l'opérateur *Gonflement au contact* du Chapitre 5 et permet de modéliser les replis de peau visibles sur les articulation des doigts par exemple.

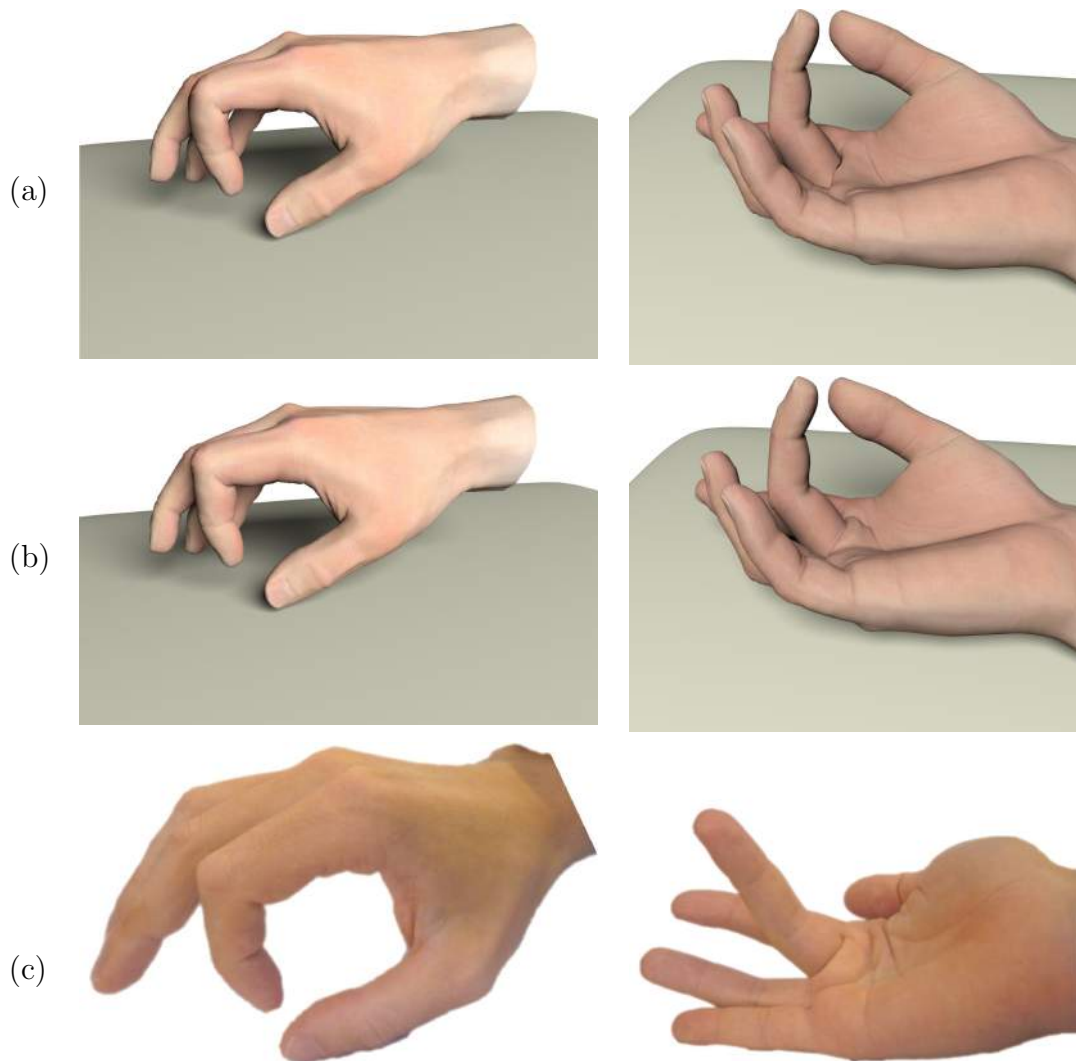


FIGURE 6.6 – Résultats obtenus dans le cadre de l’animation d’une main. (a) Skinning par SSD. (b) Skinning par utilisation de cylindres implicites. (c) Photos d’une vraie main. Les cylindres implicites composés avec l’opérateur de gonflement au contact permettent de conserver la partie supérieure des phalanges rigide et d’obtenir des replis de peau plus réalistes.

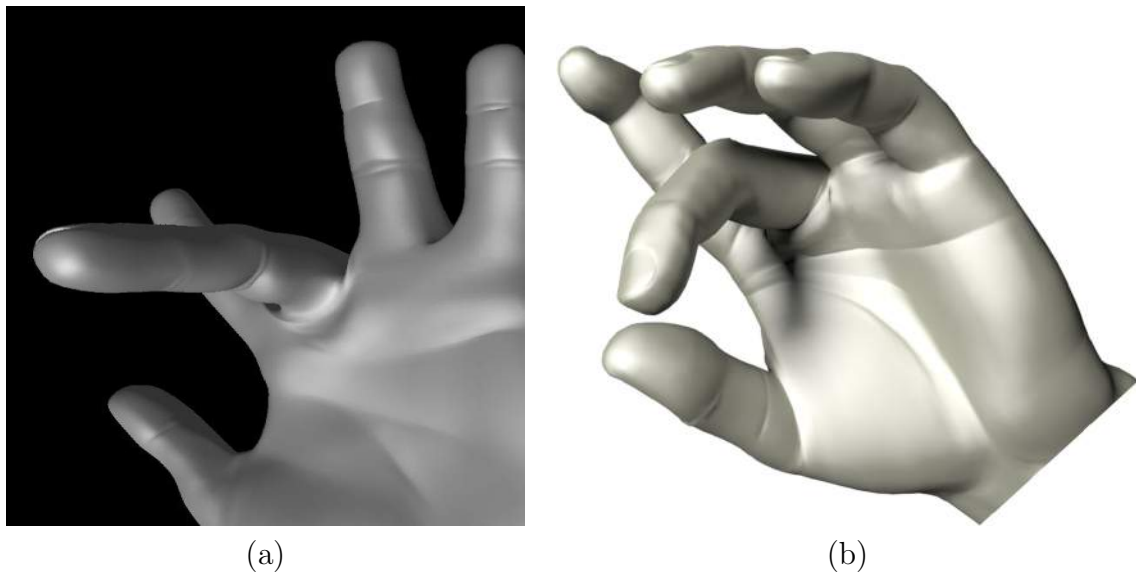


FIGURE 6.7 – Comparatif avec une autre méthode de skinning existante. (a) Résultat obtenu grâce à la technique de Angelidis et al. (b) Résultat obtenu avec le skinning implicite.

nir un comportement beaucoup plus proche de la réalité. Le skinning linéaire dégrade fortement la surface : les parties intérieures des doigts se creusent (*collapsing elbow*) et les parties extérieures se courbent. De plus, le repli de peau qui survient naturellement entre la paume et les doigts n'est pas visible. Notre technique permet en revanche de résoudre automatiquement tous ces problèmes : l'extérieur des doigts reste rigide tandis que l'intérieur gonfle légèrement, et le repli à la jonction de la paume est bien apparent.

Ce repli de peau pose problème pour la plupart des techniques de skinning basées sur un squelette. La figure 6.7, montre le repli obtenu avec la technique du *kynodynamic skinning* de Angelidis *et al.*[AS07]. Celui-ci n'est pas bien localisé : il est situé bien en dessous de la jonction. Avec notre technique, le repli est positionné à un endroit plus réaliste.

Certains cas peuvent toutefois poser problème avec notre approche. Tout d'abord, pour que le comportement de la surface soit bien traduit par l'opérateur de mélange, il est nécessaire que les sommets qui la composent soient localisés à proximité de la même isosurface du champ potentiel. Si la déviation est trop importante, les déformations du champ potentiel liées à la composition des cylindres implicites ne seront pas transmises correctement au maillage. La figure 6.8 illustre une situation à problèmes : la peau située à la jonction de la mâchoire et de la gorge du dinosaure est plus éloignée du squelette que la peau située au milieu de sa gorge. En conséquence, le repli de peau simulé par l'opérateur de mélange n'est pas localisé au bon endroit.

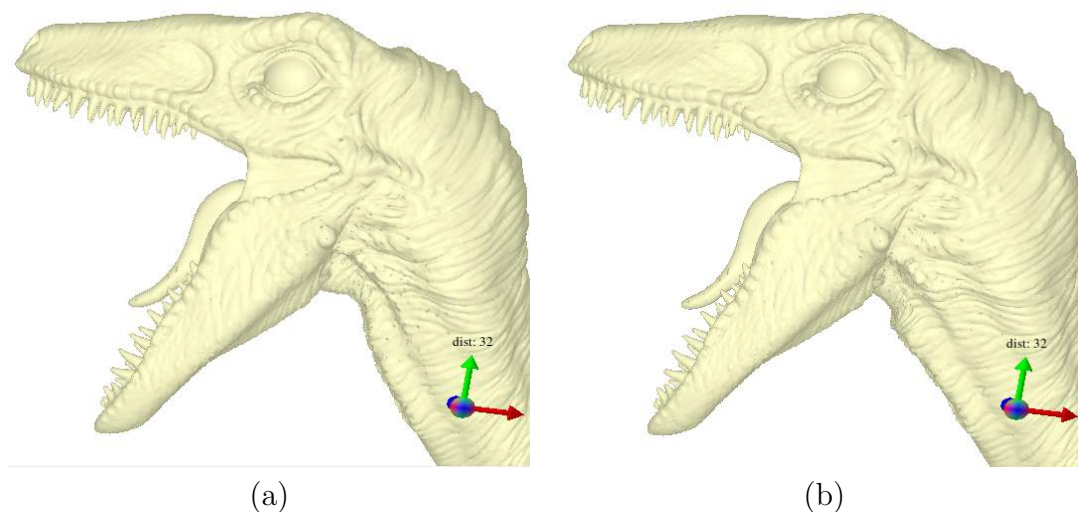


FIGURE 6.8 – Illustration d’un problème survenant lors de l’utilisation de cylindres implicites. (a) L’ouverture de la mâchoire du dinosaure crée un phénomène similaire au *collapsing elbow* à la jonction de la tête et du cou avec du skinning linéaire. (b) Le skinning implicite ne corrige pas totalement le problème car la surface n’est pas bien représentée par les isosurfaces des cylindres implicites.

Pour corriger ce problème, il est nécessaire d’avoir recours à des primitives implicites plus représentatives de la surface originale que les cylindres implicites que nous utilisons. Une solution serait d’ajouter manuellement des os supplémentaires qui génèreraient de nouveaux cylindres implicites dans les zones mal représentées par les cylindres existants. L’inconvénient de cette approche est qu’elle casse l’aspect automatique de notre technique. De plus, l’ajout de cylindres supplémentaires implique de réaliser davantage de compositions, ce qui peut augmenter le temps de calcul du champ potentiel de façon notable.

6.3.2 Performances

Notre nouvelle technique de skinning est hautement parallélisable, le traitement de chaque sommet du maillage étant effectué de manière indépendante. Elle est ainsi particulièrement adaptée aux cartes graphiques modernes. Notre implémentation est réalisée intégralement en CUDA et permet d’obtenir des performances interactives avec des modèles composés d’une centaine de milliers de polygones (cf. Table 6.1). Le skinning implicite reste tout de même particulièrement coûteux en comparaison avec le skinning linéaire. Ceci est dû au fait que l’étape de marche des sommets peut nécessiter plusieurs dizaines d’évaluations du champ potentiel avant convergence, en fonction de la position initiale

des sommets issus du skinning linéaire et du pas ϵ choisi.

scène	nombre de sommets	nombre moyen d'évaluations	temps de traitement (ms)
hand	31750	12.3	67
raptor head	302120	23.1	980

TABLE 6.1 – Performances du skinning implicite sur les scènes de test présentées (configuration : Intel Core 2 E6600 (2.66 GHz) + Nvidia GTX 280).

6.4 Discussion

Une solution au problème de la mauvaise approximation de la surface par les cylindres implicites serait d'approximer la surface représentée par le maillage avec des fonctions implicites telles que celles utilisées dans les algorithmes de reconstruction de surface : Radial Basis Functions (RBF) [MGV], Algebraic Point Set Surfaces (APSS) [GG07] ou Moving Least Squares (MLS) [FCOS05]. Il n'est pas nécessaire que la surface implicite capture les détails du maillage, mais juste qu'elle en conserve la forme générale. Or ce type de primitives permet de générer une surface à partir d'un nombre arbitraires de points d'échantillonnage : plus le nombre de points est grand, meilleure est la précision de la surface reconstruite, et inversement, un nombre réduit de points permet d'éliminer les détails à haute fréquence. Ainsi, en échantillonnant le maillage situé à proximité de chaque membre, nous pourrions obtenir une modélisation par surfaces implicites beaucoup plus précise. Le coût d'évaluation du champ potentiel serait toutefois plus élevé que dans le cas de simples cylindres implicites (complexité linéaire avec le nombre de points d'échantillonnage). Pour conserver des performances en temps réel, le champ potentiel associé à chaque membre pourrait être précalculé et stocké dans un atlas de textures en 3D.

Cette approche a été proposée par Rodolphe Vaillant *et al.* [VBG⁺13] qui a poursuivi ces travaux dans le cadre de sa thèse. Ses résultats montrent que l'utilisation de HRBF (*Hermite Radial Basis Functions* [MGV11]) au lieu de cylindres implicites pour l'approximation de la surface du maillage permet d'améliorer considérablement la qualité de la transition entre deux membres sur les articulations les plus complexes (épaule, bassin, etc.). Le précalcul et le stockage des HRBF en texture 3D permet une évaluation très rapide des champs potentiels avec une précision suffisante pour un coût en mémoire restreint (environ 10 Mo pour le stockage de HRBF d'un squelette à 21 os).

Rodolphe [VBG⁺13] a, de plus, utilisé la méthode de Newton-Raphson pour l'étape

de déplacement des sommets au lieu d'une marche à pas constant. Celle-ci converge de manière quadratique et le nombre d'itérations requis est donc beaucoup moins élevé, ce qui permet de limiter le nombre d'évaluations du champ potentiel. Cependant, dans certains rares cas, la méthode de Newton peut dépasser fortement le point de convergence. Une solution pourrait être de se baser sur les propriétés lipschitziennes des fonctions HRBF utilisées pour borner l'amplitude de déplacement des sommets à chaque itération.

7

Conclusion

A leur introduction dans les années 80, les surfaces implicites étaient considérées comme une alternative intéressante aux représentations paramétriques des surfaces. Les qualités inhérentes à l'utilisation de champs potentiels (transitions automatiques entre objets, gestion automatique des changements de topologie en animation) suggéraient leur adoption future au sein des systèmes de modélisation. Cependant il s'avérait que les défauts de ce type de représentation étaient trop contraignants, en particulier le coût en temps de calcul du rendu des surfaces et la difficulté à contrôler la forme des transitions. Les travaux présentés dans cette thèse montrent que l'impact de ces défauts peut être réduit de manière à ce que les surfaces implicites puissent devenir une alternative viable aux autres représentations surfaciques.

7.1 Récapitulatif des contributions

- Les représentations par maillage restent les plus rapides à visualiser, notamment grâce aux processeurs graphiques dédiés. Cependant nous avons montré que la visualisation en temps réel d'un grand nombre de surfaces implicites (100000) est possible grâce à une nouvelle structure d'accélération (FBVH). L'utilisation conjointe du CPU et du GPU permet de masquer le temps de construction de cette structure en contexte d'animation.
- Le mélange contrôlé par gradient augmente la flexibilité d'utilisation des surfaces implicites en modélisation. D'une part il permet de résoudre les problèmes généraux de composition de surfaces implicites (gonflement à la jonction, mélange à distance, suppression des détails et changements indésirables de la topologie). D'autre part il permet de réaliser un contrôle précis sur la zone de transition. Il est ainsi possible d'obtenir plusieurs effets intéressants en modélisation et en animation tels que le

gonflement au contact.

- Un exemple d'application pratique de ces opérateurs de mélange est le skinning implicite : en faisant suivre les fluctuations du champ potentiel aux sommets d'un maillage, les transitions entre les primitives implicites sont répercutées sur ce maillage. Ceci est possible car en effectuant un mélange de champs potentiels on transforme de façon similaire toutes les isosurfaces de ces champs. Cette transformation est alors visible sur le maillage même si les sommets ne sont pas initialement sur la même isosurface.

7.2 Discussion et travaux futurs

7.2.1 Rendu de surfaces implicites

Nos travaux sur la visualisation de surfaces implicites ont été appliqués à la visualisation d'un unique type de primitives. Ces primitives sont combinées entre elles à l'aide d'un opérateur n-aire. Il est aisé d'adapter la structure FBVH à n'importe quel type de surface implicite à support compact. En revanche, le remplacement de l'opérateur de composition n-aire par une composition de type CSG nécessiterait certaines optimisations pour conserver une évaluation efficace. En effet, le CSG impose un ordre hiérarchique d'évaluation des fonctions potentielles de chaque primitive et l'évaluation de la fonction potentielle résultante a une complexité linéaire avec le nombre de primitives total de la scène. Or, l'expression de cette fonction potentielle peut être simplifiée au sein de chaque noeud de la FBVH : seules les primitives appartenant à ce noeud (i.e. les primitives dont le support intersecte la boîte englobante du noeud) peuvent en modifier la valeur. Une évaluation efficace du champ potentiel nécessiterait donc de calculer une expression simplifiée de l'opérateur de composition au sein de chaque feuille de la FBVH à l'issue de sa construction.

7.2.2 Mélange de surfaces implicites

Le mélange de surfaces implicites contrôlé par le gradient permet de s'affranchir des problèmes de composition usuels, mais possède toutefois certaines limitations. Tout d'abord les paramètres de la fonction d'ouverture, en particulier les paramètres de pente w_0 et w_1 , doivent être réglés manuellement en fonction du type des primitives implicites utilisées afin de ne pas créer d'artefacts supplémentaires dans la composition. Il serait intéressant d'explorer diverses manières de calculer ces coefficients de manière automatique afin de limiter le travail de l'utilisateur. Ensuite, le mélange est fortement dépendant

des fluctuations du gradient des primitives implicites. Si un des gradients présente de fortes variations locales, par exemple à l’approche d’un point singulier où le gradient est nul, le mélange résultant peut présenter un aspect inattendu. Une solution pourrait être de lisser le gradient en calculant sa valeur moyenne dans une certaine sphère autour des points d’évaluation, mais cela pénaliserait fortement le coût en temps de calcul des évaluations du champ potentiel.

Les opérateurs de mélange introduits dans cette thèse ont été généralisés par Canezin *et al.*[CGB13] à l’intersection et la différence de primitives implicites. A cet effet, il a été nécessaire de borner l’opérateur de mélange. En effet, pour garantir la consistance du champ potentiel résultant de compositions successives, tous les opérateurs utilisés durant la composition doivent être bornés sur leur propre intervalle de définition. On peut choisir par exemple de considérer des opérateurs de mélange $g : [0, 1]^2 \rightarrow [0, 1]$ dans le cas où les primitives implicites utilisées génèrent des champs potentiels à valeur dans $[0, 1]$. Les opérateurs d’union g_{\cup} proposés par Canezin *et al.*[CGB13] respectent cette propriété, ce qui permet de déduire de nouveaux opérateurs d’intersection et de différence :

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

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

Ces nouveaux opérateurs permettent d’offrir aux opérateurs contrôlés par gradient la flexibilité requise à leur utilisation au sein des systèmes de modélisation par CSG.

7.2.3 Animation de maillages par surfaces implicites

L’utilisation de primitives implicites pour l’animation d’un maillage permet de transmettre automatiquement à ce maillage les déformations issues de l’opérateur de composition utilisé. Ce nouvel outil peut grandement simplifier le travail nécessaire à l’animation d’un personnage : le choix et le paramétrage d’un opérateur de composition permet de régler très facilement le comportement de la transition entre une surface lisse et un repli de la peau lors de la flexion d’une articulation pour un effet visuel convaincant. De plus, cette méthode ne souffre pas des problèmes de perte de volume habituellement rencontrés avec les méthodes de skinning traditionnelles.

Chaque nouvel opérateur de mélange offre de nombreuses possibilités de transition au niveau des articulations par l’intermédiaire des contrôleurs. Afin d’aider les utilisateurs à obtenir la forme souhaitée lors du pli d’un membre, il serait intéressant de permettre la création d’opérateurs de mélange par tracé (*sketching*) de *profils de déformation*. Ceci

permettrait d'obtenir simplement des effets complexes (ex : rides) sans avoir à trouver une formulation mathématique adéquate du profil recherché.

Enfin les développements effectués sont à l'état de prototype : à l'heure actuelle, trouver les bons paramètres des opérateurs pour obtenir la transition souhaitée nécessite de procéder par tâtonnements. La création d'un véritable environnement d'édition permettant de modifier à la volée le type et les paramètres des contrôleurs utilisés à chaque articulation faciliterait énormément cette étape et pourrait prouver que cette technique d'animation peut être utilisée par n'importe quel utilisateur sans connaissance préalable des théories mathématiques utilisées.

Bibliographie

- [Aro93] R Aronson. Photon migration and imaging in random media and tissues. In *Proc. SPIE 1888*, 1993.
- [Arr99] S R Arridge. Optical tomography in medical imaging. *Inverse problems*, 1999.
- [AS07] Alexis Angelidis and Karan Singh. Kinodynamic skinning using volume-preserving deformations. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '07, pages 129–140, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.
- [BBCW10] Adrien Bernhardt, Loic Barthe, Marie-Paule Cani, and Brian Wyvill. Implicit blending revisited. *Proc. of Eurographics, Computer Graphics Forum*, 2010.
- [BDS⁺02] L. Barthe, N. A. Dodgson, M. A. Sabin, B. Wyvill, and V. Gaildrat. Two-dimensional potential fields for advanced implicit modeling operators, 2002.
- [BDS⁺03] L. Barthe, N. A. Dodgson, M. A. Sabin, B. Wyvill, and V. Gaildrat. Two-dimensional potential fields for advanced implicit modeling operators. *Computer Graphics Forum*, 22(1) :23–33, 2003.
- [Bli82] James F. Blinn. A generalization of algebraic surface drawing. In *Proceedings of the 9th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '82, pages 273–, New York, NY, USA, 1982. ACM.
- [Blo97a] J. Bloomenthal. Bulge elimination in convolution surfaces. In *Computer Graphics Forum*, volume 16, pages 31–41, 1997.
- [Blo97b] Jules Bloomenthal. *Introduction to Implicit Surfaces*. Morgan Kaufmann, August 1997.
- [BMDS02] L. Barthe, B. Moran, N. A. Dodgson, and M. A. Sabin. Interactive implicit modelling based on C1 reconstruction of regular grids. *International Journal of Shape Modeling*, 8(2) :99–117, décembre 2002. Geometric modeling, 3D grid reconstruction, Implicit surfaces.

- [BWd04] Loïc Barthe, Brian Wyvill, and Erwin de Groot. Controllable binary csg operators for “soft objects”. *International Journal of Shape Modeling*, 10(2) :135–154, 2004.
- [CGB13] Florian Canezin, Gaël Guennebaud, and Loïc Barthe. Adequate Inner Bound for Geometric Modeling with Compact Field Function. *Computer & Graphics (proceedings of SMI 2013)*, 37(6) :565–573, July 2013.
- [Dyk08] Dyken, C., Ziegler, G., Theobalt, C. and Seidel, H.-P. High-speed Marching Cubes using HistoPyramids. In *Computer Graphics Forum 27*, number 8, pages 2028–2039, 2008.
- [FCOS05] Shachar Fleishman, Daniel Cohen-Or, and Cláudio T. Silva. Robust moving least-squares fitting with sharp features. *ACM Trans. Graph.*, 24(3) :544–552, July 2005.
- [GG07] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. *ACM Trans. Graph.*, 26(3), July 2007.
- [GLM96] S. Gottschalk, M. C. Lin, and D. Manocha. Obb-tree : A hierarchical structure for rapid interference detection, 1996.
- [Har93] John C. Hart. Sphere tracing : Simple robust antialiased rendering of distance-based implicit surfaces. Technical report, School of EECS, Washington State University, 1993.
- [Hav04] Herman Haverkort. *Results on geometric networks and data structures*. PhD thesis, Utrecht University, NL, 2004.
- [HHS06] Vlastimil Havran, Robert Herzog, and Hans-Peter Seidel. On fast construction of spatial hierarchies for ray tracing. Research Report MPI-I-2006-4-004, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, June 2006.
- [IDYN06] Kei Iwasaki, Yoshinori Dobashi, Fujiichi Yoshimoto, and Tomoyuki Nishita. Real-time rendering of point based water surfaces. In *Computer Graphics International 2006*, pages 102–114, June 2006.
- [Kaj86] J T Kajiya. The rendering equation. In *Computer Graphics(Proceedings of SIGGRAPH’86)*, volume 20, 1986.
- [KCO09] Ladislav Kavan, Steven Collins, and Carol O’Sullivan. Automatic linearization of nonlinear skinning. In Eric Haines, Morgan McGuire, Daniel G. Aliaga, Manuel M. Oliveira, and Stephen N. Spencer, editors, *SI3D*, pages 49–56. ACM, 2009.

-
- [KCvO08] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O’Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph.*, 27 :105 :1–105 :23, November 2008.
- [KDP95] H Jiang K D Paulsen. Spatially-varying optical property reconstruction using a finite element diffusion equation approximation. *Med. Phys.*, 1995.
- [KHM⁺98] James T. Klosowski, Martin Held, Joseph S. B. Mitchell, Henry Sowizral, and Karel Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, 4 :21–36, 1998.
- [KSN08] Yoshihiro Kanamori, Zoltan Szego, and Tomoyuki Nishita. GPU-based fast ray casting for a large number of metaballs. *Computer Graphics Forum (Proc. of Eurographics 2008)*, 27(3) :351–360, 2008.
- [Kv05] Ladislav Kavan and Jiří Žára. Spherical blend skinning : A real-time deformation of articulated models. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games, I3D ’05*, pages 9–16, New York, NY, USA, 2005. ACM.
- [LC87] William E. Lorensen and Harvey E. Cline. Marching cubes : A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques, SIGGRAPH ’87*, pages 163–169, New York, NY, USA, 1987. ACM.
- [LCF00] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation : a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques, SIGGRAPH ’00*, pages 165–172, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.
- [LGS⁺09] Christian Lauterbach, Michael Garland, Shubhabrata Sengupta, David Luebke, and Dinesh Manocha. Fast BVH construction on gpus. *Comput. Graph. Forum*, 28(2) :375–384, 2009.
- [MCC67] P F Zweifel M C Case. *Linear Transport Theory*. New York : Addison-Wesley, 1967.
- [MG03] Alex Mohr and Michael Gleicher. Building efficient, accurate character skins from examples. *ACM Trans. Graph.*, 22 :562–568, July 2003.
- [MGV] Ives Macedo, Joao Paulo Gois, and Luiz Velho. In *SIBGRAPI*, pages 1–8. IEEE Computer Society.
- [MGV11] Ives Macedo, Joao Paulo Gois, and Luiz Velho. Hermite radial basis functions implicits. *Comput. Graph. Forum*, 30(1) :27–42, 2011.

- [MKF⁺04] Gerd Marmitt, Andreas Kleer, Heiko Friedrich, Ingo Wald, and Philipp Slusallek. Fast and Accurate Ray-Voxel Intersection Techniques for Iso-Surface Ray Tracing. In *Vision, modeling, and visualization 2004 (VMV-04)*, pages 429–435, 2004.
- [MMG06] Bruce Merry, Patrick Marais, and James Gain. Animation space : A truly linear framework for character animation. *ACM Trans. Graph.*, 25 :2006, 2006.
- [MS93] D T Delpy M Schweiger, S R Arridge. Application of the finite-element method for the forward and inverse models in optical tomography. *J. Math. Imag. Vision*, 1993.
- [MSD95] M Hiraoka M Schweiger, S R Arridge and D T Delpy. The finite element model for the propagation of light in scattering media : Boundary and source conditions. *Med. Phys.*, 1995.
- [MTCSP04] N. Magnenat-Thalmann, F. Cordier, H. Seo, and G. Papagiannakis. Modelling of bodies and clothes for virtual environments. In *CAVW04, invited paper*, pages 201–208. IEEE Publisher, July 2004.
- [MtLTM88] Nadia Magnenat-thalmann, Richard Laperrire, Daniel Thalmann, and Université De Montréal. Joint-dependent local deformations for hand animation and object grasping. In *In Proceedings on Graphics interface 88*, pages 26–33, 1988.
- [NMHW02] A. Neubauer, L. Mroz, H. Hauser, and R. Wegenkittl. Cell-based first-hit ray casting. In *VISSYM '02 : Proceedings of the symposium on Data Visualisation*, pages 77–86. Eurographics Association, 2002.
- [NN94] Tomoyuki Nishita and Eihachiro Nakamae. A method for displaying metaballs by using bézier clipping. *Comput. Graph. Forum (Proc. of Eurographics '94)*, 13(3) :271–280, 1994.
- [PASS95] Alexander Pasko, Valery Adzhiev, Alexei Sourin, and Vladimir Savchenko. Function representation in geometric modeling : Concepts, implementation and applications, 1995.
- [PF05] Matt Pharr and Randima Fernando. *GPU Gems 2 : Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems)*. Addison-Wesley Professional, 2005.
- [PPK05] Galina I. Pasko, Alexander A. Pasko, and Tosiyasu L. Kunii. Bounded blending for Function-Based shape modeling. *IEEE Comput. Graph. Appl.*, 25(2) :36–45, 2005.

-
- [RHC09] Damien Rohmer, Stefanie Hahmann, and Marie-Paule Cani. Exact volume preserving skinning with shape control. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '09*, pages 83–92, New York, NY, USA, 2009. ACM.
- [Ric73] A. Ricci. A Constructive Geometry for Computer Graphics. *The Computer Journal*, 16(2) :157–160, May 1973.
- [Roc89] Alyn Rockwood. The displacement method for implicit blending surfaces in solid models. *ACM Transactions on Graphics*, 8(4) :279–297, 1989.
- [Sab68] M.A. Sabin. The use of potential surfaces for numerical geometry, 1968.
- [SH94] Barton T. Stander and John C. Hart. A lipschitz method for accelerated volume rendering, 1994.
- [SK83] M P vecchi S Kirkpatrick, C D Gelatt. Optimization by simulated annealing. *Science*, 1983.
- [SRAD93] M Hiraoka S R Arridge, M Schweiger and D T Delpy. A transport-backtransport method for optical tomography. *Med. Phys.*, 1993.
- [Ura06] Yuri Uralsky. Dx10 : Practical metaballs and implicit surfaces. In *Game Developers Conference*, 2006.
- [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) :125, 2013.
- [vFTS08] Wolfram von Funck, Holger Theisel, and Hans-Peter Seidel. Volume-preserving mesh skinning. In Oliver Deussen, Daniel A. Keim, and Dietmar Saupe, editors, *VMV*, pages 409–414. Aka GmbH, 2008.
- [vKvdBT07] Kees van Kooten, Gino van den Bergen, and Alex Telea. Point-based visualization of metaballs on a gpu. In Hubert Nguyen, editor, *GPU GEMS 3*, chapter 7. Addison-Wesley, 2007.
- [Wal07] Ingo Wald. On Fast Construction of SAH based Bounding Volume Hierarchies. In *Proceedings of the 2007 Eurographics/IEEE Symposium on Interactive Ray Tracing*, 2007.
- [Wea02] Xiaohuan Corina Wang and et al. Multi-weight enveloping : Least-squares approximation techniques for skin animation, 2002.
- [WH94] Andrew Witkin and Paul S. Heckbert. Using particles to sample and control implicit surfaces. pages 269–278, 1994.

- [WH06] Ingo Wald and Vlastimil Havran. On building fast kd-trees for ray tracing, and on doing that in $o(n \log n)$. In *IN PROCEEDINGS OF THE 2006 IEEE SYMPOSIUM ON INTERACTIVE RAY TRACING*, pages 61–70, 2006.
- [WMW86] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structure for *oft* objects. *The Visual Computer*, 2(4) :227–234, 1986.
- [WPP07] Robert Y. Wang, Kari Pulli, and Jovan Popović. Real-time enveloping with rotational regression. *ACM Trans. Graph.*, 26, July 2007.
- [ZHWG08] Kun Zhou, Qiming Hou, Rui Wang, and Baining Guo. Real-time kd-tree construction on graphics hardware. In *ACM SIGGRAPH Asia 2008 papers*, SIGGRAPH Asia '08, pages 126 :1–126 :11, New York, NY, USA, 2008. ACM.

A

Equation des courbes d'ouverture

Afin d'obtenir un champ potentiel lisse à l'intérieur de l'isosurface, nous définissons $k_\theta(x) = \tan(\theta)/2 \forall x \in [\frac{1}{2}, +\infty]$. La définition de k_θ à l'extérieur de l'isosurface fait intervenir la fonction suivante :

$$\lambda_\theta(x) = \begin{cases} \frac{1+\tan(\theta)}{4} & \text{si } x \geq \frac{1}{2} \\ x & \text{si } x \leq \frac{\tan(\theta)}{2} \\ \frac{1-\tan(\theta)}{4} \phi\left(2\frac{2x-\tan(\theta)}{1-\tan(\theta)}\right) + \frac{\tan(\theta)}{2} & \text{sinon} \end{cases}$$

Où ϕ est une fonction définie de manière à garantir la continuité de classe C^∞ de la fonction λ_θ (Figure A.1 (a)). Elle est construite de la manière suivante :

$$\phi^{-1}(x) = x + 1 - w(x)$$

où w est la fonction utilisée en tant que *profil de déformation* \bar{p} défini à la Section 5.2.1. La fonction ϕ ne possède pas de forme analytique connue, mais elle peut être néanmoins évaluée numériquement par recherche dichotomique.

La fonction λ_θ nécessite d'être mise à l'échelle d'un facteur $\frac{2\tan(\theta)}{1+\tan(\theta)}$ afin de correspondre à la valeur de k_θ au point $x = 0.5$. Cependant cette expression fait dégénérer l'expression du champ potentiel en une union max lorsque $\theta = \frac{\pi}{4}$ (Figure A.1 (b)). Pour éviter cela, il suffit de mettre la fonction λ_θ au carré dans l'expression de k_θ :

$$k_\theta(x) = \frac{\tan(\theta)}{2} \left(\frac{4}{1+\tan(\theta)} \lambda_\theta(x) \right)^2$$

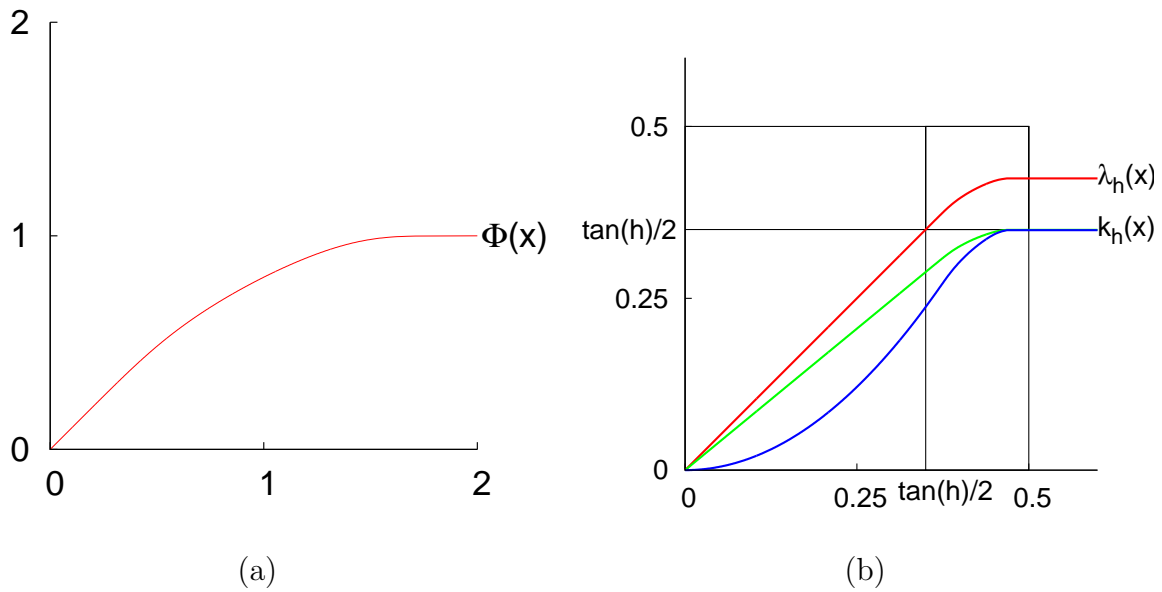


FIGURE A.1 – (a) La fonction ϕ est plate en $x = 0$ et $x = 2$, ce qui garantit que la fonction k_θ est de classe C^∞ . (b) La fonction λ_θ en rouge réalise la jonction C^∞ de deux fonctions affines grâce à la fonction ϕ visible dans le petit carré noir. Après mise à l'échelle, la fonction verte est toujours lisse mais génère un opérateur d'union non lisse quand $\theta = \frac{\pi}{4}$. L'expression finale de k_θ (en bleu) corrige ce problème.

B

Précalcul des opérateurs de mélange

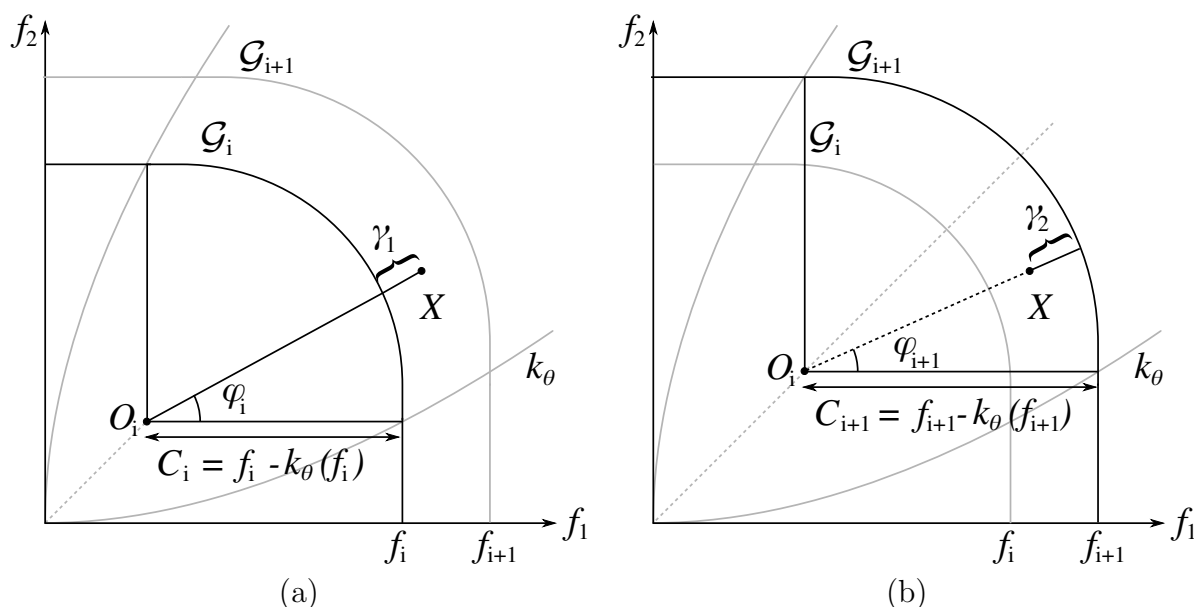


FIGURE B.1 – Calcul des paramètres d'interpolation l_{X_i} et $l_{X_{i+1}}$ pour l'évaluation de g_θ au point d'échantillonnage $X = (f_{1m}, f_{2n})$.

Pour calculer une représentation discrète de g , nous réalisons l'évaluation de g_θ aux points d'échantillonnage réguliers (f_{1i}, f_{2j}) , $i, j \in [0; N]$ de la manière suivante. Tout d'abord, nous définissons $g_\theta(f_{1i}, f_{2j}) = \max(f_{1i}, f_{2j})$ pour chaque couple i, j tel que $f_{1i} < k_\theta(f_{2j})$ ou $f_{2j} < k_\theta(f_{1i})$. Puis pour chaque $i \in [0; N]$, nous considérons les deux isocourbes L_i et L_{i+1} de g_θ correspondant aux valeurs respectives f_{1i} et f_{1i+1} . Pour chaque point d'échantillonnage X situé entre ces deux isocourbes, $g_\theta(X)$ est estimé en interpolant f_{1i} et f_{1i+1} linéairement entre L_i and L_{i+1} (Figure B.1). Cette interpolation est réalisée de la manière suivante. Nous calculons les deux valeurs :

$$\begin{aligned}l_{X_i} &= \|X - O_i\| - \bar{p}(\varphi_i) \cdot C_i \\l_{X_{i+1}} &= \bar{p}(\varphi_{i+1}) \cdot C_{i+1} - \|X - O_{i+1}\|\end{aligned}$$

qui sont respectivement les distances de X   L_i (Figure B.1(a)) et de X   L_{i+1} (Figure B.1(b)). Dans ces expressions, le point O_i a pour coordonn es $(k_\theta(f_{1i}), k_\theta(f_{1i}))$ et $C_i = f_{1i} - k_h(f_{1i})$. L'expression de $g_\theta(X)$ est alors :

$$g_\theta(X) = \frac{l_2 \cdot f_{1i} + l_1 \cdot f_{1i+1}}{l_1 + l_2}.$$

Table des acronymes

API	Interface de programmation (<i>Application Programming Interface</i>)
APSS	Méthode de reconstruction par ajustement local de surfaces quadriques (<i>Algebraic Point Set Surface</i>)
BVH	Hiérarchie de volumes englobants (<i>Bounding Volume Hierarchy</i>)
CAD	Conception Assistée par Ordinateur (<i>Computer Aided Design</i>)
CAGD	<i>Computer Aided Geometric Design</i>
CAO	Conception Assistée par Ordinateur
CPU	Processeur central (<i>Central Processing Unit</i>)
CSG	Géométrie de construction de solides (<i>Constructive Solid Geometry</i>)
CUDA	API propriétaire des cartes graphiques nVidia (<i>Compute Unified Device Architecture</i>)
FBVH	Hiérarchie de volumes englobants redimensionnée (<i>Fitted Bounding Volume Hierarchy</i>)
GPU	Processeur graphique (<i>Graphics Processing Unit</i>)
HRBF	Fonction à base radiale de Hermite (<i>Hermite Radial Basis Function</i>)
MLS	Méthode des moindres carrés mobiles (<i>Moving Least Squares</i>)
RBF	Fonction à base radiale (<i>Radial Basis Function</i>)
SAH	Heuristique d'aire surfacique (<i>Surface Area Heuristic</i>)
SSD	Skinning linéaire (<i>Skeleton Subspace Deformation</i>)
T&L	Transformation et éclairage (<i>Transform and Lightning</i>)

Table des figures

1.1	Schématisation du domaine	8
1.2	La discrétisation du domaine : à chaque voxel i correspond un coefficient d'absorption μ_a^i et un coefficient de diffusion μ_s^i	9
3.1	Un ensemble de points générés permet d'explorer plusieurs chemins	18
4.1	Configuration des sources et des capteurs employée dans nos simulations.	21
4.2	Résultats de reconstruction obtenus sur les cas tests	22
2.1	Illustration des lignes de champs d'une <i>metaball</i> en 2D.	35
2.2	Exemple de modélisation d'objets dynamiques à topologie variable à base de <i>metaballs</i> (Source : http://www.planit3d.com)	35
2.3	Les différentes méthodes de visualisation de surfaces implicites. (a) Polygonisation. (b) Lancer de rayons. (c) Rendu par points.	37
2.4	Les 15 différentes configurations possibles dans l'algorithme des <i>Marching Cubes</i>	38
3.1	Lors de la division d'un noeud, nous calculons les boîtes englobantes des metaballs et des metaballs de split (b). En calculant leur intersection avec les boîtes englobantes initiales (a), nous obtenons leurs nouvelles boîtes englobantes (c).	49
3.2	La boîte englobante des supports des <i>metaballs</i> n'est généralement pas celle qui encapsule de plus près la surface.	52
3.3	Certaines <i>metaballs de split</i> dont le support (en rouge) intersecte un noeud de génèrent pas de surface (en vert) qui intersecte ce noeud. Il est donc inutile de les stocker dans ce noeud. En revanche, il est nécessaire de stocker toute <i>metaball</i> qui contribue à la surface englobée par ce noeud.	53

3.4	Schéma de fonctionnement. L'animation des <i>metaballs</i> A_i est réalisée sur le GPU puis transférée sur la machine hôte (M_i). Celle-ci calcule alors la structure d'accélération correspondant à la position actuelle des <i>metaballs</i> (CB_i), puis transfère celle-ci sur la mémoire de la carte graphique (B_i). Le GPU possède alors les données nécessaires à l'exécution du rendu final (R_i), qui est réalisé pendant que le CPU calcule la structure d'accélération correspondant à l'image suivante (CB_{i+1}). La plupart des transferts mémoire sont ainsi masqués par le calcul, grâce à la technique du <i>streaming</i>	54
3.5	Captures d'écran des différentes scènes de test : (a,b) Tornado, scène rendue avec 3 niveaux de réflexions ; (c) Molecule 1, rendue avec l'ombrage de Phong ; (d) Molecule 2, rendue en occlusion ambiante calculée en lançant 64 rayons d'ombre par pixel ; (e) Bassin, rendue avec simulation de la réflexion et de la réfraction de Fresnel ; (f) Scattered metaballs, rendue purement en lancer de rayons primaires (pas d'ombrage). Le rendu de toutes les scènes a été effectué en résolution 640×480 dans nos tests.	56
4.1	Exemples d'opérateurs binaires de composition. En haut : opérateur d'union franche obtenu par la fonction $(f_1, f_2) \rightarrow \max(f_1, f_2)$. En bas : opérateur de mélange obtenu par la fonction $(f_1, f_2) \rightarrow \sqrt{f_1^2 + f_2^2}$. Sur chaque ligne, la figure de gauche représente les lignes de champ de l'opérateur sur le plan (f_1, f_2) et la figure de droite les lignes de champ obtenues en faisant la composition de deux <i>metaballs</i> . Notez la corrélation entre les lignes de champ sur les figures de gauche et de droite.	62
4.2	Trois opérateurs de composition binaires g et leur application sur deux cylindres implicites positionnés en croix. En première ligne, le tracé des opérateurs où les axes représentent respectivement les valeurs des champs potentiels en entrée f_1 et f_2 . Les lignes noires représentent certaines iso-courbes de l'opérateur et la ligne blanche l'iso-courbe correspondant à la valeur C de l'isosurface observée. Le gradient de l'opérateur est représenté en rouge et bleu. La surface résultante, définie par $f = g(f_1, f_2) = C$ est visualisée en deuxième ligne. Les opérateurs représentés sont : (a) une union franche standard (\max), (b) un mélange lisse [BMDS02], (c) une union franche dont le champ potentiel est lisse en dehors de l'isosurface [BDS ⁺ 02]. Les lignes vertes sur les opérateurs (b) et (c) représentent les frontières au delà desquelles $g(f_1, f_2) = \max(f_1, f_2)$	63

4.3	Illustration du gonflement indésirable. Lors de l'utilisation d'un opérateur de mélange lisse classique (a), la surface résultante s'éloigne des surfaces initiales provoquant un gonflement. Celui-ci est particulièrement visible lorsque les primitives mélangées sont de même taille, comme le montre l'exemple des deux cylindres ci-dessus (b).	65
4.4	Illustration des problèmes communs en composition de surfaces implicites. (a) Gonflement indésirable. (b) Mélange à distance. (c) Déformation des détails. (d) Changements indésirables de topologie.	67
4.5	L'opérateur de Bernhardt <i>et al.</i> interpole linéairement une union franche et un mélange lisse. En position intermédiaire, les isocourbes de cet opérateur présentent des distortions visibles en (a). La surface résultante représentée en (b) lors du mélange de deux cylindres est uniquement G^1 , ce qui entraîne des artefacts visuels lors de l'affichage de réflexions sur la surface (c). . . .	68
5.1	(a) Mélange "radial" standard provoquant le gonflement indésirable et le mélange à distance. (b) L'amplitude de la transition est paramétrée par le <i>contrôleur</i> tracé en troisième ligne. Au dessus du point d'intersection des deux cylindres, l'angle entre les gradients $\alpha = 0$ et la transition est alors une union franche ($h = 1$). Lorsque l'on descend le long de la transition α augmente de 0 à $\pi/2$, le <i>contrôleur</i> h décroît de 1 à 0 ce qui augmente la taille de la transition.	74
5.2	(a) Deux cylindres sont composés avec un opérateur de mélange de classe G^1 contrôlé par les gradients. La transition reste lisse malgré la perte de continuité occasionnée. (b) Un troisième cylindre est ajouté au résultat de la composition précédente, à l'aide du même opérateur de mélange. Le champ potentiel résultant perd alors un degré de continuité supplémentaire, qui se traduit par l'apparition d'un arête franche C^0 au sein de la transition. (c) Même scène où les cylindres sont composés avec un opérateur de classe C^∞ qui n'engendre pas de perte de la continuité.	75
5.3	Illustration des gonflements non désirés liés à un mauvais paramétrage du <i>contrôleur</i> . (a) Les gradients respectifs des champs potentiels des cylindres horizontal et vertical sont représentés en bleu et rouge. (b) Lorsque les paramètres de pente w_0 et w_1 du contrôleur sont trop élevés (ici $w_0 = w_1 = 3$), l'ouverture de la transition devient trop rapide et provoque des gonflements de part et d'autre de la transition. (c) Une valeur adaptée de w_0 et w_1 permet de résoudre le problème (ici $w_0 = w_1 = 1$). Celle-ci est à déterminer en fonction du type de primitive implicite utilisé.	75
5.4	Tracé de la fonction κ	76

5.5	Première ligne : L'angle d'ouverture θ définit la largeur de la transition au niveau de l'isovaleur correspondant à la surface observée (ici $\frac{1}{2}$). la courbe rouge représente l'isovaleur $\frac{1}{2}$ d'un opérateur de mélange $g(f_1, f_2, \theta)$. Deuxième ligne : Les courbes d'ouverture k_θ définissent les frontières de la zone de transition de l'opérateur g . (a) Opérateur complètement ouvert ($\theta = 0$). (b) Angle d'ouverture intermédiaire. (c) Opérateur fermé ($\theta = \frac{\pi}{4}$).	78
5.6	(a) Le Profil de déformation \bar{p} définit la forme des isocourbes de la transition. Pour chaque valeur de ϕ , $\bar{p}(\phi)$ représente la distance de l'origine à l'isocourbe de valeur 1 de l'opérateur g en position pleinement ouverte (i.e. pour $\theta = 0$). (b) Isocourbes de l'opérateur g pour un angle $\theta > 0$. (c) Évaluer g en un point (f_1, f_2) quelconque nécessite de résoudre l'équation 5.2.	78
5.7	Visualisation de lignes de réflexion sur une surface obtenue par mélange de deux cylindres implicites. (a) Opérateur de mélange de Bernhardt <i>et al.</i> [BBCW10] présentant une oscillation de la courbure. (b) Opérateur obtenu en utilisant notre profil de déformation \bar{p}_b . (c) Tracé de \bar{p}_b	81
5.8	Isocourbes de l'opérateur de mélange g_b pour différentes valeurs de θ . (a) Opérateur ouvert : $\theta = 0$. (b) Transition intermédiaire : $\theta = \frac{\pi}{8}$. (c) Transition franche : $\theta = \frac{\pi}{4}$	81
5.9	Première colonne : illustration des problèmes classiques rencontrés lors de la composition de surfaces implicites; (a) mélange à distance, (b) gonflement indésirable, (c) modification de la topologie (d) estompement des détails. Deuxième colonne : tous ces problèmes sont résolus en utilisant notre opérateur de mélange g_b associé au contrôleur <i>Camel</i> visible en Figure 5.10 (a).	83
5.10	Tracé des trois contrôleurs utilisés. (a) Le <i>Camel</i> , permettant de résoudre les problèmes liés aux mélange de surfaces implicites. (b) L' <i>Organique</i> qui est utilisé dans la modélisation de structures organiques par surfaces de convolution. (c) Le <i>Contact</i> utilisé conjointement avec l'opérateur de gonflement au contact pour simuler les collisions entre objets mous.	84

5.11	Utilisation de l'opérateur de mélange g_b avec le contrôleur <i>Organique</i> pour la modélisation interactive. (a) Illustration de cet opérateur sur deux cylindres. Remarquez la présence simultanée d'une arête franche et d'un mélange lisse dans la transition. (b) La transition obtenue par cet opérateur est particulièrement adaptée lors de la fixation d'un membre sur un corps, par exemple lors de l'ajout de la queue au corps du chameau. (c) et (d) montrent deux modèles réalisés interactivement en mélangeant des primitives implicites simples avec cet opérateur.	85
5.12	(a) Tracés de \bar{p}_c pour $K = 0.25$ (en rouge), $K = 0.5$ (en vert) and $K = 0.75$ (en bleu). (b) L'opérateur de composition correspondant g_c pour $h = \frac{\pi}{8}$. (c) Application de cet opérateur utilisé avec le contrôleur <i>Contact</i> présenté en Figure 5.10 (c) illustrant la déformation obtenue lors de la collision de deux cylindres.	86
5.13	Application du mélange contrôlé par gradient à l'animation. La <i>lava lamp</i> est réalisée en composant plusieurs primitives implicites avec deux opérateurs, le <i>gonflement au contact</i> et un opérateur de mélange pour les bulles quittant la base.	86
6.1	Principaux défauts du skinning linéaire. L'interpolation linéaire des transformations crée (a) un creux à la jonction de deux membres en cas de flexion et (b) un amincissement des membres en cas de torsion.	92
6.2	Principe du <i>skinning implicite</i> . (a) Une primitive implicite est générée à chaque os du squelette. Ces primitives sont mélangées par un opérateur de composition qui définit la forme de la déformation de l'articulation correspondante. (b) Lorsque le squelette est déformé, une première transformation est appliquée au maillage qui positionne les sommets de manière approximative. (c) Les sommets sont déplacés de façon à rejoindre les iso-surfaces correspondantes. Il suivent donc les déformations du champ potentiel qui sont définies par l'opérateur de mélange utilisé.	94
6.3	Une marche triviale le long du gradient du champ potentiel (b) peut entraîner une variation de la densité des sommets (c) relativement au maillage initial (a).	98

6.4	Première ligne : skinning d'un maillage de cylindre par un squelette comportant une articulation. (a) Skinning classique par SSD. (b) Skinning implicite. Le champ potentiel utilisé a été obtenu par composition des cylindres implicites centrés sur les os du squelette. Deuxième ligne : (a) Résultat du skinning implicite au creux de l'articulation lorsque le déplacement des sommets est effectué dans la direction du champ potentiel. Une perte de densité est visible. (b) Résultat obtenu lorsque le déplacement est effectué selon le vecteur de déplacement décrit dans l'équation 6.2.	99
6.5	Exemples de déformations modélisables au niveau de la jonction de deux membres modélisés par des cylindres. (a) Transition franche obtenue par l'opérateur d'union <i>max</i> . (b) Transition douce qui se réduit au fur et à mesure que l'articulation se plie jusqu'à l'apparition d'une transition franche. Celle-ci est obtenue en utilisant l'opérateur <i>Mélange sans gonflement</i> présenté au Chapitre 5. Elle modélise l'effet obtenu sur l'articulation d'un coude humain. (c) Transition franche faisant apparaître des bourrelets de part et d'autre de l'articulation. Elle est obtenue par l'opérateur <i>Gonflement au contact</i> du Chapitre 5 et permet de modéliser les replis de peau visibles sur les articulation des doigts par exemple.	101
6.6	Résultats obtenus dans le cadre de l'animation d'une main. (a) Skinning par SSD. (b) Skinning par utilisation de cylindres implicites. (c) Photos d'une vraie main. Les cylindres implicites composés avec l'opérateur de gonflement au contact permettent de conserver la partie supérieure des phalanges rigide et d'obtenir des replis de peau plus réalistes.	102
6.7	Comparatif avec une autre méthode de skinning existante. (a) Résultat obtenu grâce à la technique de Angelidis et al. (b) Résultat obtenu avec le skinning implicite.	103
6.8	Illustration d'un problème survenant lors de l'utilisation de cylindres implicites. (a) L'ouverture de la mâchoire du dinosaure crée un phénomène similaire au <i>collapsing elbow</i> à la jonction de la tête et du cou avec du skinning linéaire. (b) Le skinning implicite ne corrige pas totalement le problème car la surface n'est pas bien représentée par les isosurfaces des cylindres implicites.	104

A.1	(a) La fonction ϕ est plate en $x = 0$ et $x = 2$, ce qui garantit que la fonction k_θ est de classe C^∞ . (b) La fonction λ_θ en rouge réalise la jonction C^∞ de deux fonctions affines grâce à la fonction ϕ visible dans le petit carré noir. Après mise à l'échelle, la fonction verte est toujours lisse mais génère un opérateur d'union non lisse quand $\theta = \frac{\pi}{4}$. L'expression finale de k_θ (en bleu) corrige ce problème.	122
B.1	Calcul des paramètres d'interpolation l_{X_i} et $l_{X_{i+1}}$ pour l'évaluation de g_θ au point d'échantillonnage $X = (f_{1m}, f_{2n})$	125

Liste des tableaux

3.1	Performances du rendu de scènes variées composées de <i>metaballs</i> (cf. Figure 3.5) effectué à l'aide d'une structure d'accélération. La construction de la structure est complètement masquée par le temps de calcul nécessaire au rendu dans la plupart des cas.	55
3.2	Nombre moyen de noeuds traversés, de tests d'intersection effectués, et de <i>metaballs</i> considérées lors des tests d'intersection par rayon sur la scène Tornado.	57
3.3	Évaluation de l'intérêt de la nouvelle heuristique du coût d'intersection proposée en Section 3.1.3.	57
3.4	Bénéfices apportés par l'étape de redimensionnement des noeuds durant la construction de la <i>FBVH</i>	57
3.5	Bénéfices de l'utilisation des optimisations présentées en Section 3.1.4. . . .	58
5.1	Valeur des paramètres utilisés pour définir nos contrôleurs.	84
6.1	Performances du skinning implicite sur les scènes de test présentées (configuration : Intel Core 2 E6600 (2.66 GHz) + Nvidia GTX 280).	105